ViPLab— A Virtual Programming Laboratory for Mathematics and Engineering

Th. Richter & S. Rudlof & D. Boehringer\textsuperscript{1} & P. Grube
Computing Center University of Stuttgart
Stuttgart, Germany
{richter,rudlof,boehringer,pascal.grube}@rus.uni-stuttgart.de

Ch. Grüninger & R. Helmig
Institute for Modelling Hydraulic and Environmental Systems - University of Stuttgart
Stuttgart, Germany
{christoph.grueninger,rainer.helmig}@iws.uni-stuttgart.de

Ch. Rohde
Institute for Mathematics - University of Stuttgart
Stuttgart, Germany
christian.rohde@mathematik.uni-stuttgart.de

H. Bernlöhr
Free IT
Stuttgart, Germany
heiko.bernloehr@freeit.de

C.-D. Munz & A. Stock
Institute of Aerodynamics and Gas Dynamics - University of Stuttgart
Stuttgart, Germany
{munz,stock}@iag.uni-stuttgart.de

Conference Topic: Mathematics and Engineering Education
Keywords: MATLAB, Programming, Virtual Laboratory, Online Courses, Simulation

1 BACKGROUND AND INTRODUCTION
Due to the change of the Diploma to the Bachelor/Master system, courses at German universities had been redesigned: While the Master degree provides the necessary skills for working as a scientist in research, the Bachelor degree is supposed to teach students all

\textsuperscript{1} Corresponding Author
D. Boehringer, boehringer@rus.uni-stuttgart.de
necessary skills to work successfully in applied fields and production. As such, knowledge of computer algebra systems, numerical methods and usage of software tools is of ever increasing importance. Lectures in the Bachelor studies thus include material on both the underlying mathematics as well as the usage and handling of such tools for quite a while. However, it turned out that simply providing exercises, skeleton code, installation media and the necessary software licences for students is not sufficient to make a lecture successful and teach students all necessary skills. In our experience, a lot of time and effort is often spend on installing the software on students’ computer systems, maintain the installations, download the homework and get the individual computer systems to work. Furthermore, as students had to send homework solutions by email, they had to be checked by the teaching staff manually, and required careful adjustment of the installations to ensure interoperability between the students’ and the university machines.

To improve this situation and address the above issues, the University of Stuttgart setup the Virtual Programming Lab (ViPLab) project; the goal of this project is to implement online access to all software tools required for the Bachelor study and integrate these tools into the eLearning infrastructure of the university. Installation of software should be unnecessary, and Internet access plus a standard web browser should be all students need to provide for preparing homeworks. The online interface should be appealing, and help focus on the task at hand: numerics, or programming in general. At the same time it should provide an integrated programming environment in the sense that compiling, sending in solutions or visualizing simulations remains as simple as possible, allowing students to focus on the assignment rather than the interface. Furthermore, access to homework assignments and hand-over of students’ solutions should all be possible within the Learning Management System (LMS) of the university.

The software infrastructure for this project was implemented by the Computer Center (RUS) of the university which also installed and maintains the numerics cluster that runs the programs handed in by students and teachers. Users of the system are currently the Institute of Aerodynamics and Gas-dynamics, Aerospace Engineering and Geodesy (IAG), the Institute of Hydraulic Engineering (IWS), and the Institute for Applied Analysis and Numerical Simulation (IANS). The project is challenged by the various needs of its diverse user group and had to be designed to supply access to manifold mathematical tools, such as MATLAB, Octave, C or C++ compilers or specialized numerical simulation frameworks. The project was entirely funded by student tuitions; program code and project outcomes are available under an Open Source Licence.

The ViPLab is of course not the first of its kind: It is in a long tradition of virtual laboratories that were designed to provide simulations to complement traditional hands-on courses for students in the engineering studies, see for example [3] and [4, 1, 2]. As far as computer science goes, the VPL module for Moodle [8] provides a similar plug-in for one popular Learning Management System. Quite unlike VPL, the ViPLab infrastructure is generic and works on any SCORM [5] compliant LMS infrastructure, and it is build around a very scalable and flexible load balancing architecture that, according to our tests, can handle several hundred students at once — see also section 4. As far as assessment software for mathematics is concerned, similar commercial products are available as well. One of them is MapleTA [6], which, however, is restricted to one particular back-end and thus not quite as flexible as ViPLab. Despite that, it does not allow free programming in the same degree as ViPLab does but rather provides an automatic assessment system based on machine generated exercises students should answer online.

This article is structured as follows: In the next section, the technical infrastructure of the project is described which is followed by some examples. Following that, insight into
experiences with the system is given and student voices are presented. In the last section, we provide an outlook into future developments of the ViPLab.

![Diagram of ViPLab architecture](image)

**Figure 1**: The architecture of the ViPLab. For both student and teacher the infrastructure is solely operated through a browser. The actual computation is run on a back-end which may support one or many numerical software packages, compilers or simulation software. Communication between the components is entirely based on HTTP/REST.

## 2 SOFTWARE ARCHITECTURE

The ViPLab architecture, see Fig. 1 consists of three modules: A front-end Java Applet that runs on the students’ machine and provides an editor for the source code and several visualization toolkits for rendering plots and graphics; a load balancing middle-ware, called the eLearning Community Server [11] (ECS), which collects requests from the front-end and passes them over to the numbercrunching back-end. The ECS also provides a simple database that keeps all homework assignments in the form of code-templates and back-end configurations. The third component consists of one or several back-ends which run the actual numerical software, let it be a C or C++ compiler, a mathematical or numerical software like GNU Octave or MATLAB or a simulation framework like DuMux [9, 10].
"Solution":
"exerciseModifications":
"elements":
[

{"value":"int main(int,char**) 
{printf("Hello World\n"); 
return 42;}\n",
"identifier":"Id3"
}
],
"ID":4711,
"exercise":
"https://vip.uni-stuttgart.de/numlab/exercises/0815",
"postTime":"2012-03-10 23:07:09 +0100"
}

Figure 2. A student solution encoded in JSON; only the editable code parts are transmitted here, the exercise identifier allows the back-end to merge this solution with the exercise template.

The communication between them is based on REST [12], a stateless protocol in the tradition of http or HTTPS using a trivial text encoding called JSON [14]. That is, every homework “exercise” consists of a single database entry at the ECS middle-ware encoded in JSON and transmitted to the corresponding student front-end, see Fig. 3. A homework exercise is basically a template consisting of editable, non-editable (constant) or invisible code segments which has to be completed by the student to form a running program.

Figure 3. The student front-end, here on a simple linear algebra exercise. The student solution is on white on the left-hand side, non-editable parts of the template greyed out top-left, the output of the system on the right.

In addition, the data base entry does not only define the code parts and their visibility, it also defines the interpreter or compiler, its options, the linker and the program environment in general. ViPLab is thus not constraint to a specific back-end configuration but is able to describe a variety of setups ranging from compiler driven execution as C or C++ up to interpreter languages such as Octave. The back-end, polling for new incoming student solutions at the ECS, merges the code-parts, performs a syntax and validity check and then executes the compiled binary or the MATLAB code, forwarding any results back to the front-
end. Back-ends are typically instances of virtual machines running in a large computer cluster, as currently employed by the Computing Center of the University of Stuttgart. Thus, upscaling the system in critical situation, e.g. near the end of the semester, to provide more computing power and to serve more students is quite easy.

The front-end is a Java Applet running in the students’ browser; it is delivered by the Learning Management System of the university as part of a SCORM module — basically HTML code plus JavaScript for the communication between the LMS and the Applet — which contains a URL reference to the Applet code and the configuration of the Applet. This configuration states both the URL of the ECS middle-ware and a database ID identifying the code template – the database id defining the homework – to load into the front-end.

In addition to the student Applet, a more powerful “teacher” front-end is also supplied which is used to create the code-templates, i.e. homework assignments, see Fig. 4, which also includes features to design and test assignments, and to create the final SCORM package.

![Figure 4. The teacher front-end showing the design of a simulation exercise. To the bottom right an interactive 3D plot rendering the result. The same visualization is also available in the student Applet.](image)

3 INTEGRATION OF THE ARCHITECTURE

Integration of the overall architecture into the eLearning infrastructure of the University of Stuttgart based on the Ilias system was an important goal to begin with. Many commercial systems like MapleTA [6, 7] are stand-alone, but as such also separate from the remaining infrastructure and cause additional work as soon as they need to be linked to the university system. For this, we depend on the SCORM standard [5] and the teacher frontend provides the functionality to create SCOs that are readily uploaded into the LMS.

A second challenge we had to address is that of visualization. Visualization and usability were identified as mission critical milestones of the whole project. Unfortunately, visualization toolkits offered by the deployed numerical software are of little use as they would be only visible on the back-end server. Thus, we developed replacement functions for MATLAB and Octave which overload the build-in visualization commands, most notably the plot command, and which communicate the data to be plotted to the front-end via the ECS. The currently available tools, namely a line (2D) plot, a grid (contour) plot and a 3D plot cover most applications we have in the current lectures. Impressions from the currently available interactive plotting tools are depicted in the right-hand side of figures 3 and 4.
4 EXPERIENCES

The system described above has been implemented by the Computing Center starting in 2009 and, after prototypical deployment in 2010, is now in production use by several institutes of the university, namely the IAG, IWS and IANS. A relatively new user is the University of Linköping in Sweden where students use ViPLab to program in Open-Modelica [17]. Before starting the production server, the computing center ran a stress test to estimate the performance of the overall system design. According to this test, two desktop machines operating as back-ends were handle up to about 200 students simultaneously, demonstrating that the overall design was sound and stable.

After the successful test and a prototypical testing by volunteering students, the system was rolled out to the university; by far the largest user group is formed by the students of the undergraduate Numerical Mathematics course offered by the IAG, approximately 300 students currently participate here. A smaller number of students use ViPLab at the IWS and the IANS.

Not quite unexpected, we found a couple of technical issues during the first days of deployment, namely that the Virtual Private Network manager we depended on in the first design was too hard to handle for most students, and that the network connectivity was not as good as expected in some lecture halls. The lack of an Java update on some Apple systems also cut off some students from the system. However, we were able to overcome these difficulties quickly by optimizing the system and lowering the requirements on the Java VM.

5 USER FEEDBACK

The IAG collected student feedback on the system, again as part of the Numerical Simulations course. As part of the regular course schedule, three weekly tutorial exercise courses were offered by two student teachers; in these exercise sessions, the teachers provided an introduction into working with ViPLab using MATLAB. The total number of students per course was approximately twenty. At the end of the course, we asked the students participating in the exercises to evaluate the following statements, providing rates from “1” to “6” on the following questions, where “1” denotes a “I agree completely” and “6” I disagree completely:

1. My state of knowledge in Matlab was good at the beginning of the lecture
2. Programming in VIPLab was easy for me.
3. The exercises were helpful for solving the homeworks.

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agree</td>
<td>1</td>
<td>1.9%</td>
<td>13.9%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.9%</td>
<td>23.8%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.9%</td>
<td>22.8%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12.3%</td>
<td>29.7%</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>84.0%</td>
<td>9.9%</td>
</tr>
<tr>
<td>Disagree</td>
<td>6</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Count</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agree</td>
<td>106</td>
<td>101</td>
<td>106</td>
</tr>
<tr>
<td>Disagree</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>4.75</td>
<td>2.98</td>
<td>2.25</td>
</tr>
</tbody>
</table>

Table 1: Student ViPLab Evaluation in the Summer Term 2011
We collected overall 107 answers to this questionnaire, the result of which is depicted in table 1. Individual statements of the students on the question “What did you like most in the course?” showed that ViPLab and the exercises are appreciated a lot by the students. To complete the picture, we provide a list of student quotes on the system:

- “The exercises helped a lot to improve the comprehension.”
- “The Matlab-Beginner-Tutorial was very helpful.”
- “Not only theory but also practical problems (exercises).”
- “Working with my own laptop!”
- “Lots of fun with the tutorial exercises.”

In the meantime, students take the availability of the system as self-evident by now. We should also note that reactions on the compiler-based FORTRAN courses run in the last decade were quite critical, though a precise evaluation of the reasons is in so far complicated as we not only switched from a compiler-based setup to ViPLab, but also from FORTRAN to MATLAB. One observation we made, however, is that ViPLab can be used right from scratch whereas it usually took several weeks to get students acquainted with the compiler.

6 OUTLOOK

While the architecture is currently deployed and accepted for homeworks, we plan to also implement electronic exams in the future. The challenges here are, however, mostly non-technical but rather organizational. For example, legal requirements for such exams need to be clarified, and we must ensure sufficient availability of computer hardware for this application.

Future versions may also provide additional visualization tools, as demanded by the users, and may provide further back-ends, support for Java is for example currently in the planning phase; another area of interest could be to support long running simulations.

7 ACKNOWLEDGEMENTS

The authors want to thank the Computing Center for partially sponsoring the project and the hardware.

REFERENCES


