VELS: VHDL E-Learning System for Automatic Generation and Evaluation of Per-Student Randomized Assignments

Martin Mosbeck
Institute of Computer Technology,
TU Wien, Vienna, Austria
martin.mosbeck@tuwien.ac.at

Daniel Hauer
Institute of Computer Technology,
TU Wien, Vienna, Austria
daniel.hauer@tuwien.ac.at

Axel Jantsch
Institute of Computer Technology,
TU Wien, Vienna, Austria
axel.jantsch@tuwien.ac.at

***Abstract*—Learning digital design with VHDL requires extensive practice with solving many assignments independently, which is difficult to provide in a university setting with high student-teacher ratio. Automated assessment systems can help by facilitating a flexible, satisfying learning environment for students.**

This paper describes the VHDL E-learning system VELS, its implementation, and our experience over the last three years with approximately 1000 students. VELS is a flexible system with key features including a uniform task system with parameterized tasks, non-static testbenches, communication over email, a tool that aids in creating new tasks, flexible configuration possibilities with different course modes, exchangeable simulator backend and multi language support.

I. Introduction

As today's electronic hardware is becoming ever increasingly complex, comprehensive knowledge of hardware modeling with description languages like VHDL is becoming crucial. To learn how to design hardware with the help of hardware modeling languages, students need active involvement with the language and tools to grasp the used abstractions and concepts and be allowed to make errors.

Like software programming, learning hardware modeling requires practice with adequate tasks. Students have to iteratively improve their knowledge by trying different approaches on given tasks. This implies the need for adequate, frequent and ideally instant feedback on students' solution designs. Providing feedback and adequate tasks is a common problem area due to large classes and different types of learners concerning pace, space and study time scheduling.[1, 16]

To address this problem field and to provide an improved learning experience to students, we have developed a VHDL E-Learning System (VELS) at the Institute of Computer Technology, TU Wien, Austria with the following key features:

- The system is usable for students via email.
- Tasks are parameterized, so different students will rarely get exactly the same task.
- A task creator tool aids teachers to easily build new tasks.
- The system supports a variety of course configurations for different target audiences.
- The used simulator is exchangeable with low effort.
- It supports multi language task descriptions to use one task in multiple courses held in different languages.
- Testbenches are non-static, each submission is tested by a new testbench.

Implementation and incorporation of enhancements from using the system for three years with approximately 1000 students have lead us to a modular, flexible system for teachers and students. VELS is open source, licensed under the GPLv2 and can therefore be used and extended by other institutions. Up to now 20 different tasks ranging from a simple truth table, modeling parts of a CPU to the modeling of a system communicating with other systems via a synchronous protocol have been implemented.

VELS was inspired by the Eudyptula Challenge [4], "a series of programming exercises for the Linux kernel, that started from a very basic 'Hello world' kernel module, moving up in complexity to getting patches accepted into the main Linux kernel source tree."[4]. Students interact with the system via email to get tasks, solve given tasks and submit them to receive feedback. The multi threaded submission system *autosub* [21] handles submissions, collects statistics and interacts with the task system. A task itself is a sum of scripts and templates that use the common task system facilities.

Task specific generators create parameterized task descriptions for each student. Randomized parameters guarantee an individual task description and the language template system allows different languages for the same task. Task specific testers communicate over a uniform simulator abstraction interface with a chosen simulator (e.g. ghdl) and assemble meaningful feedback for students. Teachers can configure courses over a web configuration interface, monitor students' progress and create new tasks aided by a task creator tool.

This paper describes VELS and its implementation and the experience we have made during the last three years deploying it in undergraduate and graduate courses. Section II provides an overview of automatic assessment in general and comparable VHDL systems. Section III describes VELS, the implementation choices we made, its parts and how the system and its tools can be used. Our experience deploying

the system for three years are summarized and illustrated in Section IV. Finally, Section V concludes the paper with a summary of challenges we faced, points out limitations, and lists plans for further improvement.

II. Related Work

Learning to model hardware using a hardware modeling language involves acquiring complex knowledge combined with acquisition of practical skills. Novice designers lack the competences related to designing and implementing problem solutions, skills that can only be acquired through continuous formative assessment that assists in gaining a deeper understanding of the target field [16, 17]. Research on automatic assessment of programming exercises is an ongoing research field since more than 50 years [8] as it offers a chance to overcome issues related to manual assessment of exercises [2]. With manual assessment each student solution has to be evaluated, tested and graded by an instructor. Manual assessment is a time consuming task resulting in an enormous workload for large classes. Given exercises tend to be big, span several topics and the assessment procedure shows difficulties ensuring consistency and accuracy [18].

Automatic assessment can provide many small exercises and consistent immediate feedback to students resulting in a motivating environment, that allows to incrementally acquire practical skills and considers diversity factors between students such as working habits and prior knowledge [1]. Surveys show that automatic assessment is a factor to improve students performance and learning experience [15]. Nevertheless deploying automatic assessment entails new challenges as tasks, testing and feedback have to be of high quality to be useful to students [20, 16]. Tasks have to be specified more precisely as no instructor can compensate for ambiguous task descriptions [16]. Test cases have to be carefully chosen to ensure adequate and complete test coverage [18]. This implies that part of the saved workload shifts to designing good tasks [20]. Altogether automatic assessment systems have to be carefully designed to be flexible, easy to use, extendable and allow task reuse.

The following surveys [9, 19, 18] give a good overview of existing systems, without distinguishing between hardware modeling assignments and classical programming assignments due to their similarity. Specifically for hardware modeling language courses several systems have been proposed.

Kumar et al. [13] developed a light-weight IDE based on jEdit, GHDL and GTKWave that allows local simulation of given tasks and submission to an Apache Tomcat server based platform that checks the submission concerning correctness. Multiple submissions of different users can be handled in parallel. Tasks are static and stored in a problem database that can be accessed through a teacher server applet. Students submissions are checked with predefined test vectors against a model solution design.

Gutièrrez et al. [6, 7] developed an extension "CTPracticals" to the common Moodle learning management system [14], which they use to teach hardware modeling in the context of computer architecture. CTPracticals both offers an interface for students and instructors. Students solve given statical assignments called practicals by uploading their solutions in a zip file. Submissions are checked by testers consisting of a fixed testbench, a command script to interface a simulator backend and a file containing expected outputs and other needed files. Students access feedback concerning their submissions over the Moodle system, instructors can manage teams, access several statistics and update grades. The system offers a multi language feature for task descriptions, an extensive logging system and is flexible concerning what language the solutions are written in and what simulation backend is used (e.g. Matlab, Modelsim VHDL, Logisim circuits).

Jelemenska et al. [11] also implemented their assessment system inside the Moodle environment. Students are partitioned into groups and can access tasks and upload solutions over Moodle. Instructors create tasks and assign them to assignment categories. Automatic random association between users and a task of an assignment category aims to prevent students in the same group to get the same task. Similarly to Gutierrez et al.[6, 7] this system uses predefined testbenches and comparison to a reference model. A later version [12] also incorporates a content aspect comparing significant parts of students solutions to the reference so students can be given points if they "used components, keywords or parts of the language that were required by the assignment"[12].

Compared to these systems VELS introduces a few new key features:

- The task system enables to design parameterized tasks so that students get different versions of a task.
- All communication is handled via email, a technology all students know well.
- A uniform interface makes it easy to create support for a new simulator backend, without having to change tasks.
- A tool that intuitively assists in defining new tasks significantly reduces the time to design a new task.
- A flexible course configuration web interface allows a wide variety of different courses. Courses can be held in strict task queue or request mode, tasks can have own start and end date, chosen language and simulator backend.
- As a new testbench is created for each submission, consecutive submissions are tested differently, ensuring that students' submissions fulfill the task specification.

III. The VELS System

VELS is an E-learning system built from multiple parts working together: 1) a multi threaded submission deamon system called *autosub*, 2) the task system with common interfaces and conventions, 3) a VHDL task creator tool, 4) the course configuration and status web interface VELS_WEB and 5) existing tasks.

The following subsections depict the parts of the system to give an overview on how the system works, can be configured and used. Detailed implementation and instructions on how to use the system can be found in the freely available codebase

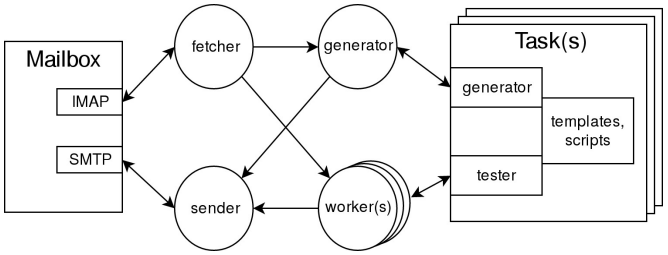

Fig. 1. Overview of the different threads of the *autosub* system and how they work together with the task system and email mailbox.

[21] and the system usermanual [22]. VELS runs on a Linux server system and has been tested on Debian 7, 8 and 9.

A. Autosub submission system

The *autosub* submission system is the core of VELS. It is the communication endpoint for both incoming and outgoing emails and therefore is in charge of fetching, processing and archiving students' emails. It also has to manage users, per user customized tasks and submissions and communicate with task specific generators and testers to generate tasks and test user submissions.

Autosub is built as a multi-thread server daemon implemented in Python3 with each thread being in charge of a subtask. This approach was taken to ensure progress even if testing of a submission takes long. An overview of the different threads and how they work together can be seen in Figure 1. A fetcher thread is in charge of fetching student emails and deciding which system actions to take depending on the email's subject. If the user has to be provided with a task description the generator thread is notified, which in turn communicates with the generator of the requested task. If a user provides a submission for a task one of multiple worker threads is notified to communicate with a tester. As testing is the bottleneck of the system, multiple worker threads ensure parallel testing of multiple users' submissions. The fetcher, worker and generators can produce textual output and attachments which the sender thread assembles to an email and sends to the student who initiated the action. If the *autosub* system does not understand which action to take, the student is informed on how to use the system. Users whose email address is not whitelisted are prevented from using the system. *Autosub* is equipped with a central logging system to monitor and quickly identify the source of errors.

Figure 2 depicts the typical flow triggered by a user requesting a task. The user sends an email with the subject "Request Task <N>", with <N> being the unique task number of the task. This triggers the task specific generator, that creates a parametrized variant of the task identified by the task parameters. Consequently the triple (user id, task number, task parameters) is saved in the *UserTasks* database and together with the help of templates a language specific description file, a specific entity file and an empty behavior file is created. All these files are saved on the server and assembled to an email which is sent to the user. To submit a solution for a task, the user sends an email with the subject "Result Task <N>" and attaches his or her solution behavior file to start the testing flow (Figure 3). The task specific tester is triggered to generate a custom testbench for the user specific task variant. The testbench, the entity file and the submitted behavior file then are fed to a chosen simulator backend, controlled by the common tester via a simulator interface, that abstracts the specifics of the chosen simulator. If the test passes, an email with subject "Success Task <N>" is sent to the user. In case the test fails an email with subject "Failure Task <N>" containing simulator error messages, specific behavior feedback and, optionally, a signal wavefile is assembled and sent to the user.

B. Task system & creating a new task

To use a task with *autosub* it has to follow a minimal uniform structure. This base structure ensures uniformity and streamlines the creation of new tasks. Each task is located in its own directory and at minimum has to supply a generator and a tester. In case of the already implemented tasks, both are executable shell scripts which call other helper scripts and use task template files.

A generator has multiple responsibilities: generate random task parameters for a task, create VHDL files, provide a task description, save the created files and add information to the database to identify user, task files and task parameters. A tester is given the student's submission and the task information to generate a custom testbench with randomized test vectors, test the student's design and indicate correctness of the submission or create feedback on what is wrong.

Tasks in VELS evolve around the model that students have to program the behavior of one or multiple entities to solve a given problem. Therefore the first step to create a new task is to establish the general parameters of the task: a unique name and the entities, the behavior of which the student has to design including their ports. Next the VELS task creator wizard can be used to generate a skeleton version of the task.

As the wizard creates all the mandatory files and fills them with both task independent content and the given general task parameters, the user can focus on the individual key points of the task. These points include:

- Complete the task description by adding the concrete problem definition (including text, figures, tables, etc.) and support all wanted languages.
- Implement the task generating logic in the provided skeleton generator including the task specific parameters which will later be randomized during the task generation.
- Complete all entity templates which should be given to the students.
- Implement the testbench generating logic in the provided skeleton tester and program a testbench template. Designing the testbench and its feedback is the key feature for a good task.

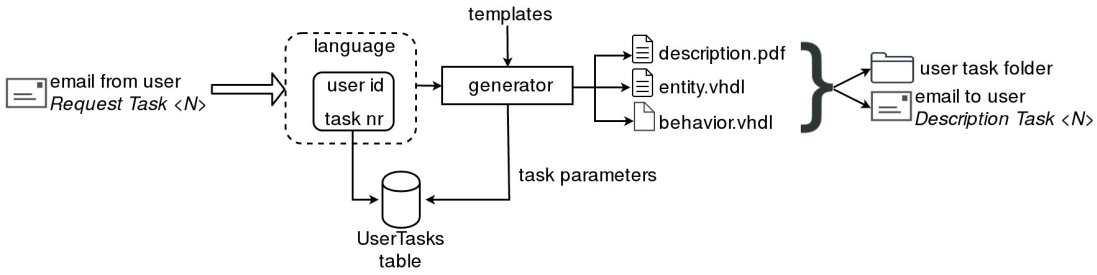

Fig. 2. System flow of a student requesting a task.

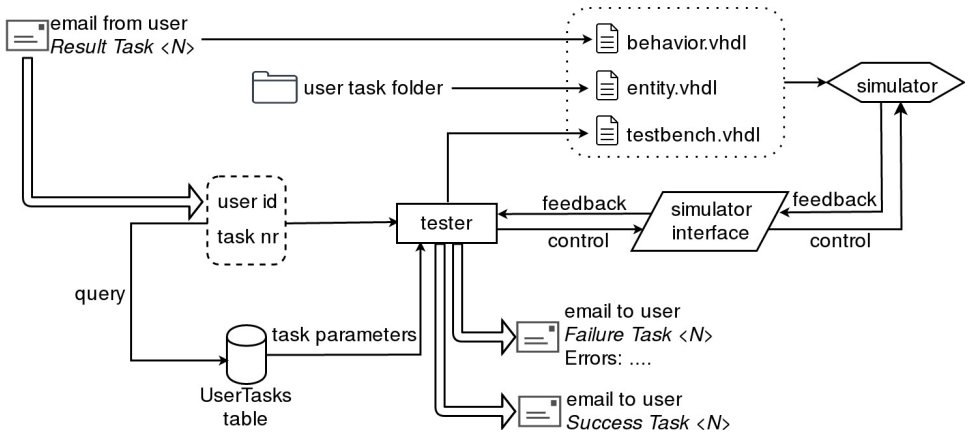

Fig. 3. System flow of a student submitting a task solution.

C. Course configuration and status

A course operator of a new course has to configure VELS at two points: a system configuration file and VELS_WEB, the course configuration and status web interface. While the configuration file is used to configure connection credentials to the email server and general system parameters, VELS_WEB is used to configure the task list for the course and monitor students' progress and course statistics.

VELS offers two distinctive course modes: linear mode and request mode. Linear mode allows courses which want to force a strict queue of tasks. Upon registering the student gets the first task of the queue. A student only advances to the next task if he or she solved the current one. The second course mode offered by VELS is the request mode. It gives students more freedom by allowing to individually request tasks. Tasks of a course have to be assembled in a task list, with each task having its own configuration consisting of: a unique task number, start time and deadline, name of an existing VELS task, language and simulator backend. Individual start times, deadlines and the two modes of VELS allow the creation of many different course types. Moreover, VELS can be used as a system to hold exams in a closed laboratory environment.

Concerning status and monitoring both the *autosub* submission system and VELS_WEB offer a number of functionalities. Per student progress can be accessed via VELS_WEB to see which tasks a student is doing, how many submissions a student has done per task and which submission was the first successful. Used testbenches and each submission are saved by the *autosub* system to enable analysis of students' learning experience. In addition, VELS_WEB offers statistics at a global scale, for example the success ratio for each task.

IV. Deployment and Results

A. Deployment and usage

VELS has been used at the Institute of Computer Technology, TU Wien, Austria during the last three years and its continuous improvements were based on the students and lecturers feedback. VELS has been implemented in two different courses, namely "Microcomputer" and "Digital integrated circuits". While "Microcomputer" is an undergraduate course in the "Electrical Engineering and Information Technology" bachelor program held in German, "Digital integrated circuits" is a graduate course in the "Embedded Systems" master program held in English. Consequently these two courses presuppose quite different foreknowledge and have different learning targets.

In the "Microcomputer" course, students have their first contact with hardware description languages thus the focus is to convey the basic knowledge of VHDL and its main constructs. The implemented VELS tasks therefore have to be very simple to support the students' first VHDL experiences (e.g. multiplexer, Pulse Width Modulation (PWM) generation, etc.). The tasks were continuously introduced during the semester based on the lecture progress and related code examples were

presented in the accompanying lecture. The VELS course configurations have varied over the last three years, including both the request and linear mode. During the first two years the tasks had been mandatory for being admitted to the end exam. In the last year this mode was changed and it was by the students' own choice to complete the tasks. VELS was also used for the end exam in the "Microcomputer" course. During the first two years the students had to pass one given VELS task. In the last year's exam students were provided with two tasks out of 14 available tasks and had to pass at least one of them. In total approximately 750 students signed in for the "Microcomputer" VELS course over the last three years and 600 students used it in their end exam.

As the "Digital integrated circuits" course is part of the master program, the students already have some foreknowledge about hardware description languages. The learning target therefore focuses on the deeper understanding of VHDL and the VELS tasks need to be more complex (e.g. ALU, single cycle CPU, synchronous communication etc.). Similar to the "Microcomputer" course students where provided with a number of VELS tasks during the semester, but VELS was not used for the end exam. In total, approximately 200 students signed in for the "Digital integrated circuits" VELS course over the last three years.

Figure 4 gives a graphical overview of VELS activity for the course "Digital integrated circuit" during the winter semester 2017. VELS handled over 3000 received emails. The figure affirms that automated assessment with VELS ensures continuous learning of students.

Table I illustrates the students' submission behavior and the variability in the difficulty of tasks. For the eight assignments in the course "Microcomputer" during the winter semester 2017 the table shows the number of submissions, the number of successful submissions and the number of successful students. Students did not have to finish these tasks to pass the course, but the tasks were recommended as a preparation for the end exam. Although the tasks have not been mandatory, VELS was faced with a big number of submissions resulting in over 15.000 fetched emails. While the average success rate was high, a clear difference in the complexity of the particular tasks can be observed. By comparing the number of correct submissions and the number of students, who passed a task, it can be seen that some students even have requested and completed the same task multiple times, indicating that students enjoyed interacting with the system to improve an already correct solution.

B. Results and experience

The evaluation of the two courses shows very good acceptance of the VELS system. Both students and lecturers quickly felt confident with the system and its interface. The email based system makes VELS independent of the used operating system and therefore supports all the different students' personal setups. Combined together with online simulators (e.g. EDAplayground [3]) the students were able to use the VELS system on any PC setup without the need to install a complex development tool.

TABLE I
SUBMISSIONS AND SUCCESS RATES OF THE COURSE "MICROCOMPUTER" DURING THE WINTER SEMESTER 2017

	Submissions			Students		
Task nr.	Sum	Passed	Rate	Sum	Passed	Rate
1	1215	317	26 %	242	195	81 %
2	905	326	36 %	242	185	76 %
3	623	294	47 %	242	188	78 %
4	1967	313	15 %	242	176	73 %
5	1318	240	18 %	242	160	66 %
6	1411	115	8 %	242	84	35 %
7	795	247	31 %	242	147	61 %
8	1805	141	7 %	242	80	33 %

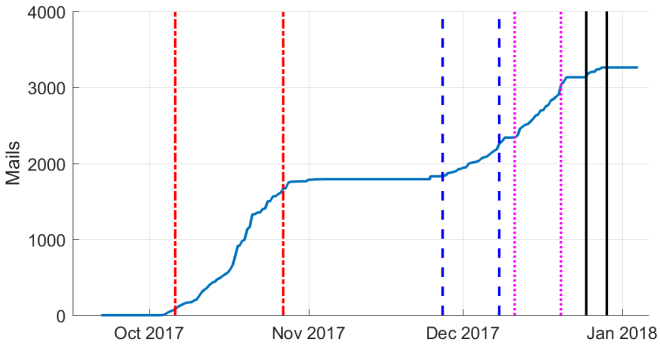

Fig. 4. Number of received emails during the winter semester 2017 for the "Digital integrated circuits" course. Each vertical line pair represents the start- and end-date of one task group.

Summarizing the lecturers feedback, the following should be noted:

- Maintaining the system and providing new tasks during the semester is easy and intuitive.
- VELS configuration is flexible, allowing courses with different languages, modes and simulator backends.
- The task creator tool saves time and ensures uniform task setups.
- Creating new tasks and especially their testbenches is the key point for a fruitful use of VELS and takes some time and experience. Students have to be provided with meaningful error feedbacks.
- Supervising the students is still necessary and therefore the possibility to ask and answer questions on specific tasks is a very useful feature of VELS.

Students feedback was collected with the universities internal feedback system. Notable comments (translated from German) include:

- *"I especially liked ... the voluntary practice opportunity via VELS."*
- *"I especially liked ... the VELS exercises were good for practicing."*
- *"Sometimes the VELS comments are not really helpful, if no explicit tips are given about what behavior is wrong."*

V. Conclusion & Future Work

After three years of continuous development and improvements VELS now is a fully operational VHDL E-learning system. The email based interface, individually randomized tasks, multilingual task descriptions, different course modes, individual textual submission feedback and the task creator wizard have led to a good reception among both students and lecturers. While VELS helps the students to continuously improve their skills by providing small tasks with an instant assessment, it does not replace human teaching. Students still need to learn the basic structures and constructs of hardware description languages, but can enhance and strengthen their knowledge with E-learning systems like VELS.

One very important point for an optimal outcome of VELS is the quality of each task. Compared to manual assessment the task description and specifications must be more detailed, unambiguous and easy to understand. Beside the task description special attention has to be payed on designing the testbench and its feedback algorithm. Students need to get a comprehensive feedback from the system.

The points mentioned above also indicate the limits of VELS. The size of a task is limited by the possibility to create an unambiguous description with an optimal feedback. The larger and complexer a task gets, the more possible mistakes have to be considered and the bigger the feedback algorithm has to be. In any case, it will be impossible to take every potential submission mistake into account and therefore a human contact point should be provided in parallel.

VELS is open source, licensed under the GPLv2 and still getting continuously enhanced. Some further improvement ideas include: building a synthesis tool interface for testers to allow e.g. feedback concerning resource consumption, a plagiarism checker, partial grading based on correct coding approaches, an interface to make the system usable with Moodle and bundling the system as a Debian package.

Acknowledgment

VELS is a big project that would not have been possible without the contributions of following people: Andreas Platschek provided the basis for the first version of autosub, Hedyeh Kholerdi and Gilbert Markum implemented tasks and Gilbert Markum devised the simulator abstraction interface.

VELS received funding as a VHDL E-Learning Platform from the Institute of Computer Technology[10], autosub received funding as an E-Learning Best Practice Project from the FH Campus Wien Teaching Support Center [5].

References

[1] M. Amelung, K. Krieger, and D. Rösner. "E-Assessment as a Service". In: *IEEE Transactions on Learning Technologies* 4.2 (Apr. 2011), pp. 162–174.

[2] B. Cheang et al. "On automated grading of programming assignments in an academic institution". In: *Computers & Education* 41.2 (2003), pp. 121–131.

[3] *EDA Playground*. https://www.edaplayground.com/.

[4] *Eudyptula Challenge*. http://eudyptula-challenge.org/. Accessed: 2018-04-04.

[5] *FH Campus Wien Teaching Support Center, Austria*. https://www.fh-campuswien.ac.at/.

[6] E. D. Gutiérrez et al. "An Experience of e-assessment in an Introductory Course on Computer Organization". In: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, pp. 1436–1445.

[7] E. Gutiérrez et al. "A new Moodle module supporting automatic verification of VHDL-based assignments". In: *Computers & Education* 54.2 (2010), pp. 562–577.

[8] J. Hollingsworth. "Automatic Graders for Programming Classes". In: *Commun. ACM* 3.10 (Oct. 1960), pp. 528–529.

[9] P. Ihantola et al. "Review of recent systems for automatic assessment of programming assignments". In: cited By 165. 2010, pp. 86–93.

[10] *Institute of Computer Technology, TU Wien, Austria*. https://www.ict.tuwien.ac.at/.

[11] K. Jelemenská and P. Čičák. "Improved Assignments Management in Moodle Environment". In: *INTED2012 Proceedings*. 6th International Technology, Education and Development Conference. Valencia, Spain: IATED, May 2012, pp. 1809–1817.

[12] K. Jelemenska, P. Cicak, and M. Gazik. "VHDL models e-assessment in Moodle environment". In: *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*. Nov. 2016, pp. 141–146.

[13] A. Kumar, R. C. Panicker, and A. Kassim. "Enhancing VHDL learning through a light-weight integrated environment for development and automated checking". In: *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. Aug. 2013, pp. 570–575.

[14] *Moodle Learning Management System*. https://moodle.org/.

[15] R. Pettit et al. "Are automated assessment tools helpful in programming courses?" In: vol. 122nd ASEE Annual Conference and Exposition: Making Value for Society. 122nd ASEE Annual Conference and Exposition: Making Value for Society. 2015.

[16] V. Pieterse. "Automated Assessment of Programming Assignments". In: *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*. CSERC '13. Arnhem, Netherlands: Open Universiteit, Heerlen, 2013, 4:45–4:56.

[17] A. Robins, J. Rountree, and N. Rountree. "Learning and Teaching Programming: A Review and Discussion". In: *Computer Science Education* 13.2 (2003), pp. 137–172.

[18] R. Romli, S. Sulaiman, and K. Z. Zamli. "Automatic programming assessment and test data generation a review on its approaches". In: *2010 International Sym-*

posium on Information Technology. Vol. 3. June 2010, pp. 1186–1192.

[19] D. M. Souza, K. R. Felizardo, and E. F. Barbosa. "A Systematic Literature Review of Assessment Tools for Programming Assignments". In: *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. Apr. 2016, pp. 147–156.

[20] T. Staubitz et al. "Towards practical programming exercises and automated assessment in Massive Open Online Courses". In: *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. Dec. 2015, pp. 23–30.

[21] *VELS and autosub*. https://github.com/autosub-team/autosub.

[22] *VELS usermanual*. https://github.com/autosub-team/autosub/blob/master/doc/doc_pdf/usermanual.pdf.