Lecturers' and Students' Experiences with an Automated Programming Assessment System

Clemens Sauerwein
University of Innsbruck,
Innsbruck, Austria
clemens.sauerwein@uibk.ac.at

Simon Priller
University of Innsbruck,
Innsbruck, Austria
simon.priller@uibk.ac.at

Martin Dobiasch
University for Continuing Education Krems,
Krems, Austria
martin.dobiasch@donau-uni.ac.at

Stefan Oppl
University for Continuing Education Krems,
Krems, Austria
stefan.oppl@donau-uni.ac.at

Michael Felderer
University of Innsbruck,
Innsbruck, Austria
michael.felderer@uibk.ac.at

Ruth Breu
University of Innsbruck
Innsbruck, Austria
ruth.breu@uibk.ac.at

Abstract

Assessment of source code in university education has become an integral part of grading students and providing them valuable feedback on their developed software solutions. Thereby, lecturers have to deal with a rapidly growing number of students from heterogeneous fields of study, a shortage of lecturers, a highly dynamic set of learning objectives and technologies, and the need for more targeted student support. To meet these challenges, the use of an automated programming assessment system (APAS) to support traditional teaching is a promising solution. This paper examines this trend by analyzing the experiences of lecturers and students at various universities with an APAS and its impact over the course of a semester. In doing so, we conducted a total number of 30 expert interviews with end users, including 15 lecturers and 15 students, from four different universities within the same country. The results discuss the experiences of lecturers and students and highlight challenges that should be addressed in future research.

Keywords: Automated Programming Assessment System, Empirical Study, Expert Interviews

1. Introduction

In the course of the ongoing digitization of economy and society, programming education at universities is gaining an increasingly important role. While in the past programming played an important role mainly in the natural sciences, its relevance is increasing in almost all other scientific fields. For example, in fields like economics, social sciences, or humanities, a growing need of programming skills can be observed.

Accordingly, as part of their strategy for digitization in teaching and research, universities are including new formats such as supplementary programs or minors with a focus on programming. As a result, the group of students who acquire programming skills during their studies will broaden considerably and become more heterogeneous. However, the challenge lies not only in the rapidly growing number of students but also in their heterogeneous prior knowledge, the shortage of teachers, and highly dynamic learning objectives and technologies (Mekterovic and Brkic, 2017).

Accordingly, solutions are needed to support lecturers in supervising a large number of students with heterogeneous backgrounds. These solutions should provide students with automated feedback and targeted support in learning programming. Automated programming assessment systems (APAS) are used for overcoming problems associated with manually managed programming assignments, such as objective and efficient assessments in large classes and providing timely, targeted, and helpful feedback (Mekterovic and Brkic, 2017). During the last years, a large variety of APAS appeared on the market (Keuning et al., 2016, 2018; Mekterović et al., 2020), like Checkpoint (English and English, 2015), JACK (Goedicke et al., 2008) or ArTEMiS (Krusche and Seitz, 2018). The main objectives of these APAS are to motivate students, provide a well-founded overview of the learning progress, improve the quality of teaching and students' contributions, minimize the programming entry hurdle, standardize and objectify the feedback and decrease dropout rates (Keuning et al., 2018; Mekterović et al., 2020).

However, there has been very little research examining lecturers' and students' experiences related to the use of an APAS. For example, only studies regarding students' perceptions of these systems (Barra et al., 2020; Gordillo, 2019; Restrepo-Calle, Ramírez Echeverry, and González, 2019; Restrepo-Calle et al., 2020; Rubio-Sánchez et al., 2014) and investigations regarding their usability (Pettit, Homer, Gee, et al., 2015) have been conducted. Furthermore, anecdotal evidence suggests that an APAS is typically developed and used at only one university for a specific programming course and is rarely used at various universities within different courses (Amelung et al., 2011; Ullah et al., 2018). To address these research gaps, we would like to answer the following research question: *What are lecturers' and students' experiences with an automated programming assessment system used at different universities?*

To answer this research question, we conducted an intervention study in which we used an APAS at four different universities in seven different programming courses in Austria over a period of one semester. In doing so, we used ArTEMiS (Krusche and Seitz, 2018) as an APAS and primarily focused on introductory courses in programming with a total number of approximately 517 students from a wide variety of fields of study. Our intervention study was accompanied by a series of expert interviews at the end of the semester. A total of 15 interviews with lecturers and 15 interviews with students were conducted. Experiences, impacts, and challenges related to the use of an APAS were identified. Our study showed that teachers and students are satisfied with the system in use. However, it identified a need for improvement regarding the integration of existing learning management systems, the usability of online editors, the creation of suitable programming tasks, and the avoidance of the trial-and-error behavior of students.

The remainder of this paper is structured as follows. Section 2 discusses related work regarding comparable studies in the field. Section 3 provides background information on our project and why we used ArTEMiS as an APAS at different universities located in Austria. Section 4 describes the applied research methodology. Section 5 presents lecturers' and students' experiences with an APAS. In Section 6 we discuss the results and outline limitations of the research at hand. Finally, Section 7 provides a conclusion and outlook on future work.

2. Related Work

Recent research has been conducted in higher education to understand and support teachers and students in learning programming (Luxton-Reilly et al., 2018). Several studies have examined how students approach programming assignments (Pettit, Homer, Gee, et al., 2015) and the problems that make learning to program difficult (Gomes and Mendes, 2007; Jenkins, 2002). In order to solve these problems, tools have been developed in this context (Marin et al., 2017; Souza et al., 2016; Ullah et al., 2018). In particular, the use of and research on APAS has increased. Based on a comprehensive review of existing literature in the field Pettit et al. showed that the use of an APAS might influence the learning success and teaching experience (Pettit, Homer, Holcomb, et al., 2015). Moreover, other researchers demonstrated that the iterative learning of programming with an APAS is perceived as very supportive by students (Yan et al., 2020). For example, the advantages of an APAS include more objective grading of programming tasks Poženel et al., 2015, the use of gamification to increase student engagement (Coore and Fokum, 2019), or the use of logging data to analyze student learning behavior (Ihantola et al., 2015; Knobbout and Van Der Stappen, 2020) and code quality (H. M. Chen et al., 2020). Additionally, APAS have also been developed to provide a meaningful distributed learning environment and students' perceptions regarding these systems have been investigated (Daradoumis et al., 2019). In the wake of the COVID-19 pandemic, the relevance of these APAS has continued to grow (Barra et al., 2020). Queirós et al. already conducted an extensive study in Portuguese universities to understand how programming teaching is approached in their higher education, demonstrating the need for automation (Queirós et al., 2020). Furthermore, other researchers have investigated the usability of an automated test data generator (Romli et al., 2015). Another strain of research investigates the assessment of programming assignments or even competency based assessment (Galan et al., 2019; Vargas et al., 2019).
In contrast to related work, we analyze the experiences with an APAS that becomes an integral part of programming teaching at seven universities in Austria. Moreover, we adopt a user-centered perspective that includes lecturers and students alike. In doing so, we interviewed a heterogeneous group of lecturers (i.e., more or less experienced) and students (i.e., from different study programs) from four different universities. We thus contribute to advancing the state-of-the-art by explicitly examining the heterogeneous end user experiences with APAS that emerge from the needs for different target groups and disciplinary backgrounds when engaging in learning to program.

3. Background Information

In order to align and standardize the teaching of programming at Austrian universities and to support teachers and students, the CodeAbility Austria project was initiated. The goal of the project is the deployment and extension of an APAS with learning analytics functionalities to enable adaptive learning and support just-in-time teaching. In order to find a suitable APAS for the project, we evaluated various systems, like Checkpoint (English and English, 2015), JACK (Goedicke et al., 2008), ArTEMiS (Krusche and Seitz, 2018) regarding their functionalities, software architecture, support and license. Based on this evaluation, we selected ArTEMiS (Krusche and Seitz, 2018) because it provides all basic functionalities of an APAS (i.e., managing, testing, and providing feedback on programming exercises), is programming language agnostic, documents the interactions of users, is a web application, is still supported, is open source and expandable. For this intervention study, ArTEMiS serves as a prototype for APAS in general. Our investigations showed that the addressed functionalities in our study are widely available in other APAS and the focus on evaluation was put on functionalities rather than the actual form of implementation. Therefore, the experiences identified are not specific to ArTEMiS but can be considered for APAS in general.

4. Applied Research Methodology

In order to investigate lecturers' and students' experiences with an APAS used at different universities, we carried out an intervention study over the period of one semester. Our intervention study consisted of a *preparation* (see Section 4.1), *operation* (see Section 4.2) and *evaluation phase* (see Section 4.3).

4.1. Preparation Phase

As part of the CodeAbility Austria project, we announced a call for study participation among the seven participating universities. Four of the seven universities voluntarily agreed to use the central project instance of the APAS ArTEMiS (Krusche and Seitz, 2018) in selected introductory programming courses. The respective universities chose the responsible lecturers and courses themselves. In total 15 lecturers, including four professors, three postdocs, three PhD students, and five teaching assistants, with an average teaching experience of approximately 9 years and mainly programming experience in C, Java, and Python participated in the study. To ensure that all 15 faculty are familiar with the use of the APAS, we held an introductory workshop. After his workshop, all lecturers verbally ensured being able to prepare and conduct their courses on the APAS. This preparation consisted of entering suitable programming assignments with corresponding test cases into the system for automatic assessment. In order to ensure a fair assessment, lecturers were instructed to formulate test cases covering all possible implementation options of the respective assignment.

4.2. Operation Phase

After the *preparation phase* the APAS was used at four different universities in seven courses. In total, approximately 517 freshmen from different fields of studies participated. Amongst others, they included bachelor students of computer science and business informatics but also students from supplementary programs of computer science or other study programs with a strong focus on digitization. The seven courses were introductory courses to programming with an average number of approximately 73.8 students. Programming languages of the courses (n) C (n = 1), Java (n = 3), and Python (n = 3) were taught. In order to run these courses with the support of the APAS, in total 38 C, 168 Java, and 17 Python programming assignments were entered into the system during the *preparation phase*.

Students had to solve the programming assignment using the APAS on a mandatory basis. Thereby, students had the option to solve the exercise with their integrated development environment (IDE) or integrated APAS editor. Finally, students had to submit their solved exercises on the APAS. The submissions were automatically assessed by executing the predefined test cases on the system. Based on the test execution results, feedback was provided to the students. In addition, the platform provided services for exercise distribution and submission management to the respective lecturer.

4.3. Evaluation Phase

After the *operation phase* we conducted 30 individual expert interviews including the aforementioned 15 lecturers and 15 students. Each of the four participating universities was asked to randomly select four different students that participated during the *operation phase* in at least one of the seven courses. In this context, it is worth mentioning that one university was only able to recruit three students instead of four. In addition, only students who passed the respective course were selected, as no failed students could be contacted.

The goal of these expert interviews was to create an understanding of the use of the APAS, the acceptance of the platform, its influence on the learning process, and compile suggestions for improvement. We developed a detailed protocol including concrete interview questions (e.g. *Which functionalities of the APAS were used?*) to guide the interviewer and to guarantee the reproducibility, objectivity, and comparability of the interviews. All interviews were conducted virtually by at least one author of this publication. The interviews lasted approximately half an hour and were audio recorded. Finally, we compiled the recordings and notes into transcripts. These interview transcripts were analyzed to produce qualitative summaries and extract information relevant to our research (Campbell et al., 2013). In doing so, we performed an inductive categorization based on the discussed functionalities of an APAS during the interviews. Three authors of this paper performed this categorization. If a conflict occurred we discussed it and, if necessary, we reached a consensus by majority voting.

5. Results

In the following section, we present the experiences gained by *lecturers* (see Section 5.1) and *students* (see Section 5.2) while using the APAS.

5.1. Lecturers' experiences

During the interviews, the lecturers shared their experiences, regarding the *course management*, *preparation of exercises*, *monitoring of the learning progress*, *impacts on students' solutions*, *evaluation of exercises* and *usability*, while using the APAS ArTEMiS.

Course management: Most of the participating universities handle their course management in their respective learning management system (LMS), like OLAT or Moodle. Accordingly, a few lecturers raised concerns regarding the introduction of an additional system as their university was using a LMS in almost all courses. In this context, several lecturers mentioned that a thorough integration of the APAS into their university's LMS would be appreciated as this would reduce the number of extra tools lecturers and students have to use. Moreover, an integration between existing systems would also enable the APAS to make use of functionalities for managing individual students or groups of students. In addition, due to the diverse nature of the examined courses and their exercises, the participants highlighted the need for functionalities to map their course structure in the APAS. While some courses featured one bigger exercise per week, or even multiple weeks, other courses consisted of several smaller exercises per week. For the latter, lecturers expressed the need of grouping several exercises to a larger unit such as an exercise sheet. However, it has to be pointed out that most other features of the APAS, e.g., discussing exercises or requesting/giving additional feedback, were not used in the courses since these features were covered by the respective LMS of the universities. These examples illustrate once again the need for close integration between APAS and LMS for course management.

Preparation of exercises: Before the pilot phase, the lecturers reported a steep learning curve while preparing the exercises in the APAS. In this context, lecturers reported that creating relevant and meaningful tests with associated feedback might be cumbersome. For example, introductory exercises heavily rely on Input-Output (IO) tests which makes it difficult to write a test case for every possible input or output. In addition, a minimal deviation in the output of a student's solution might result in a failed test case even though the solution is correct. Moreover, there may be unforeseen failed tests or error messages from the APAS that could confuse students. For example, a failed test due to a timeout caused by a system scanner waiting for input might lead to the following error message: *'Your program did not finish within 5 seconds'*. This error message could be hard for students to understand since the same error message might appear if students write an infinite loop. In addition, a meaningful test case contains an appropriate feedback message as to why a test failed. In this context, participants have reported that it is difficult to define messages with helpful and meaningful feedback. In order to address this challenge, participants suggested that it would be helpful to import exercises with good and meaningful test cases from another LMS. However, this approach proved to be labor intensive as most of the test cases had to be adapted to the logic of the APAS.

Monitoring of learning progress: The interviews showed that an APAS is very useful for monitoring the learning progress. In this regard, lecturers reported that they gained better insight into each student's knowledge because an APAS can analyze the evolution of a student's submission and provide an overview of all solutions with a corresponding score. For example, the contributions of individual students can be tracked by their commit history and typical student problems can be identified early on. In addition, individual students can be compared to the performance of all students. Thus, monitoring learning progress can allow lecturers to respond promptly to problems as they arise and provide targeted support to students.

Impacts on students' solutions: Lecturers have noticed positive and negative impacts on students' solutions by the APAS. On the one hand, lecturers mentioned that the quality of submitted solutions improved significantly, which might be traced back to the fact that students' solutions must conform to a certain template and meet predefined test cases. In this context, it is worth mentioning that the quality of solutions is related to the quality of test cases defined by lecturers. On the other hand, they criticized that due to the predefined test cases, the creativity of solutions is limited. In particular, this plays a role especially in advanced topics, where the focus is on finding creative solutions. For example, in the case of an exercise where a design pattern is searched, the test cases might provide information about the expected solution and thus limit creativity or bias students. Unfortunately, the APAS was sometimes too strict while assessing students' solutions. For example, they had to adapt their correct solutions to meet the test cases. Moreover, a trial-and-error pattern of students solving the exercises could be observed. In other words, this means that students keep trying around with their solutions until all test cases are passed. In this context, one participant stated that a student might learn more by failing a test case and getting (individual) feedback from the lecturer rather than iteratively modifying the program until the tests are passed. In other words, it was reported that students kept modifying their solutions until all test cases are passed.

Exercise evaluation: Lecturers highlighted that the APAS reduced the time needed for correcting exercises. Moreover, the automated assessment of students' solutions is beneficial since it solves the problem of missing teaching assistants to some extent. In this context, a participant pointed out that it has become increasingly difficult to find teaching assistants for evaluating and grading the exercises. Consequently, if a teaching assistant might grade all submitted solutions by hand, it might decrease the quality of teaching. Therefore, an APAS provides a solution for automated and objective assessment of students' solutions, it increases the efficiency of assessment and allows lecturers to focus on their core tasks of teaching. In addition, several lecturers stated that the use of the APAS leads to a reduction of discussion about grades and scores with students. However, participants also noted that a plagiarism check across different programming languages would be desirable in order to detect plagiarism early.

Usability: Regarding the usability of the APAS lecturers highlighted the need for an API, a feature-rich online editor, and tight integration into the respective development environment (e.g., IntelliJ IDEA). The desired API could automate certain tasks (e.g., grading of students, exporting a list of participants or solutions, updating of assignment specifications, ...) and enable the integration of the APAS into existing LMS. In addition, a feature-rich online editor would allow teaching programming without the need for an IDE. This could avoid installing and learning the functionalities of an IDE and direct the primary focus on learning a programming language. Last but not least, the participants highlighted the need for a tight integration of the APAS into the used IDE. This could be realized by implementing plugins for the respective IDEs. In this context, the participants mentioned that such a plugin should automate the submission of solutions and present the feedback of the APAS accordingly.

5.2. Students' experiences

During the interviews, the students shared their experiences, regarding the *solving of programming exercises*, *feedback*, *automated assessment* and *used functionalities*, while using the APAS ArTEMiS.

Solving of programming exercises: In order to solve the programming exercises provided by the APAS, students had two possibilities to solve them, via the online editor or in their local IDE. Our interviews showed that the online editor was used very little because of the lack of certain functionalities (e.g., auto-completion, debugging features, ...). According to these limitations, only a few students used it to fix minor issues that were identified by the automated tests or to solve introductory programming exercises (e.g., hello-world programs). As the semester progressed, it became apparent that all students used their IDE instead of the online editor. This choice was motivated by more extensive functionalities, a much more user-friendly programming experience through customization options (e.g., dark mode), highlighting of syntax errors, or a larger editor window. Moreover, students stated that it was much more convenient to run their code on their local machine instead of triggering the whole continuous integration pipeline of the APAS. This might be traced back to the fact that the evaluation of solutions by the APAS takes quite some time. Moreover, in this context, the students wished for a more in-depth introduction on how their solutions are evaluated by the APAS since the whole submission approach and the underlying continuous integration pipeline lacked transparency.

Feedback: Students highlighted immediate feedback as the greatest strength of APAS but felt that feedback could be improved or made more precise. In particular, the possibility of not having to wait for feedback from the lecturer after submitting a solution was emphasized very positively. This way, students can see whether a programming assignment has been completely misunderstood or where their mistakes are. The fact that the tests pointed them to certain mistakes (e.g., division by zero, null reference,...) was perceived as helpful and students indicated that it improved their learning performance. In general, students reported that the feedback made them feel even more mentally engaged in the task, allowing them to correct their mistakes directly. In this context, students reported that they felt more independent because they could correct their own mistakes without the help of a lecturer. They also indicated that they were more engaged with their code and learned a lot by correcting their own mistakes. However, this behavior leads to the trial-and-error pattern observed by the lecturers. In addition, several students pointed out that in other courses, they do not know if their solutions are correct when they submit their exercises. Accordingly, they feel nervous before the classes where the submitted exercises are discussed. Students reported that this problem is solved by the immediate certainty of having successfully submitted the solution into the APAS. In their opinion, this is another aspect of why immediate feedback is perceived as beneficial. However, the students stated that the instant feedback can be improved. For example, some error or feedback messages were incomprehensible or inaccurate. Moreover, students mentioned that it would be also beneficial to get exact error locations instead of just stating that a test failed. In this context, students emphasized that they would appreciate a concrete description of the test case so that they can investigate why a certain test fails. Sometimes they were not able to identify the cause and hence were unable to fix it. As already discussed, in general, this might be traced back to the fact that the definition of appropriate test cases with meaningful feedback represents a major challenge for lecturers.

Automated Assessment: After the solutions have been successfully submitted to the APAS, the automated assessment takes place. In general, students commented positively that submitting and reviewing assignments through the APAS provides a comparable and objective evaluation. This might be attributed to the fact that the solutions are now assessed by a single instance and not by different lecturers as before. One criticism, however, was that it was not possible to submit incomplete code or partial solutions. For some students, this meant that not all submissions could be evaluated.

Used functionalities: Finally, we interviewed the students regarding their *scope of use* of the automated programming assessment system. Since the students primarily work on the exercises in their local integrated development environment and use the continuous integration pipeline to submit their solutions to the automated programming assessment system it becomes apparent that the system is perceived and used as a submission platform. Other functionalities of the system, such as sending questions or complaints about the assignments and assessment, were not used. One reason for this might be that the universities already use established learning management systems that are used for all courses. Registration for courses takes place via these systems. All results are also listed there. Quizzes and communication with teachers also take place partly via these tools or in the respective seminars.

6. Discussion

While both teachers and students were satisfied with the APAS, it became evident that the system does not yet fulfill all of the needs of both parties. In the following, we discuss the *key findings* (see Section 6.1 to 6.4) with *recommendations for future work* (see Section 6.5). As already discussed in Section 3, our key findings are not limited to the APAS used, but also apply to APAS in general. In addition, we outline potential limitations of the research at hand (see Section 6.6).

6.1. Limited user acceptance and integration into existing systems

Our investigations showed that lecturers and students did not use the full functional scope of the APAS and heavily rely on their university's LMS (e.g., OLAT, Moodle,...). For example, many features of the APAS like asking questions or requesting feedback were not used by the students. This was mainly because they were instructed by their lecturer to use email communication or accustomed to using the LMS of their university. In this context it is worth mentioning that the use of LMS at universities has become even more widespread as a result of the COVID-19 pandemic. At most universities they are the primary tool for course management, course operation, exams, and student assessment. Accordingly, the use of the APAS meant that lecturers and students had to accept an additional system and switch between different systems to perform their tasks. In order to address these issues, it is highly recommended that an APAS offers functionalities or APIs to facilitate integration into existing systems, like LMS. However, former research (Amelung et al., 2011) showed that this is a major challenge. Besides this, further empirical investigations are needed to understand the factors influencing user acceptance and satisfaction of APAS.

6.2. Limited functionalities of code editors

The expert interviews showed that the APAS's integrated code editor has limited functionalities although this might put students attention on learning programming instead of challenging them with a locally installed integrated development environment (IDE). Additionally, our investigations have shown that integrating an APAS into a local IDE can be difficult due to missing plugins. However, students prefer locally installed IDEs due to their extensive range of functions and the possibility to customize them. Nevertheless, an integrated development environment is another system that students must install on their operating system and become familiar with. Accordingly, an integrated online editor would remedy the situation, focus the attention on learning programming and not introduce another system in the learning environment. Especially for freshmen or students from other disciplines, this removes the first hurdle to programming. Accordingly, an improvement in the functionality of integrated code editors in APAS is required to achieve the necessary acceptance by students and thus provide a full-fledged system for programming learning.

6.3. Challenging preparation of assignment with corresponding test cases

During the interviews, lecturers shared their experiences of preparing exercises with corresponding test cases and feedback. Similar to previous research (Ala-Mutka, 2005; Cerioli and Cinelli, 2008; Enström et al., 2011), they stated that it is not a trivial but an elaborate, difficult and time-consuming task. Especially the definition of good test cases and valuable feedback was a challenge for some lecturers. This was also illustrated by the fact that students would like to see improvements in terms of feedback and error reporting. In addition, as also highlighted by P. M. Chen, 2004 the creation of test cases is made more difficult, as they might limit or bias the creativity of students. For example, students might see the solution to the problem based on the test cases and focus their implementations only on fulfilling these test cases. Accordingly, there is room for improvement in the process of creating exercises with corresponding test cases and feedback. Due to the fact that several lecturers from different universities are involved in the CodeAbility Austria project, it would be conceivable to implement an exchange of exercises and a corresponding peer-review process to increase the quality of exercises and test cases. Moreover, such collaboration was already mentioned by the lecturers as an expectation regarding the APAS in an earlier project phase. In the course of this collaboration, the expertise of several lecturers could be drawn upon or standardized exercises for certain problems could be created jointly. Therefore, it would be worth considering introducing and establishing an exchange platform for teaching materials and exercises. In a further step, this platform could be expanded and pursue a crowdsourcing approach involving lecturers outside the CodeAbility Austria project.

6.4. Trial-and-error behavior of students

Last but not least, similar to previous studies (Edwards, 2004; Karavirta et al., 2006; Restrepo-Calle, Ramirez Echeverry, and Gonzalez, 2019) our investigations showed a trial-and-error behavior of students while solving exercises with the automated programming assessment system. This can also be attributed to the fact that a test-driven approach is taken when solving and assessing the exercises. Students indicated that after a failed attempt, they searched for solutions on the Internet or kept trying until the tests were passed, which could be annoying. In this context, lecturers mentioned that they would like to monitor the learning process in order to understand what the problems were. Furthermore, monitoring of failed attempts could also help to identify and prevent trial-and-error behavior as this approach has been shown to be detrimental for students' performances (Restrepo-Calle, Ramirez Echeverry, and Gonzalez, 2019). Based on the knowledge of failed attempts and the competencies to be taught, individual learning paths could be defined. In other words, the system should provide support for typical failures or problems which might occur when solving the exercises. For example, this support could be advanced exercises, additional learning materials, or tutorials. To support the approach of individualized learning paths, the respective course should be mapped in the system as a competency graph. For each task, the student's competence gain is checked by the predefined test cases. If a competence has not been acquired, the student is given another task and the aforementioned support to acquire it. This approach results in individual learning paths that are aligned with the competency graph.

6.5. Directions for future Work

Based on the key findings (see Sections 6.1 to 6.4), the following directions for future work emerge: (1) *Design, implementation, and evaluation of concepts to integrate APAS into commonly used LMS* (cf. 6.1), (2) *Empirical investigations on factors influencing user acceptance and satisfaction of APAS* (cf. 6.1), (3) *Design, implementation and evaluation of an integrable full-featured code editor for APAS.* (cf. 6.2), (4) *Design, implementation and evaluation of a sharing platform to exchange standardized assignments with corresponding test cases for APAS* (cf. 6.3), (5) *Application and evaluation of crowdsourcing concepts to create assignments with corresponding test cases for APAS* (cf. 6.3) and (6) *Development, implementation and evaluation of concepts for individualized learning paths in APAS based on students' learning traces and competency graphs* (cf. 6.4). As mentioned earlier, the proposed directions are not limited to our APAS. Rather, they are intended as topics for future research and suggestions for improving all APAS.

6.6. Limitations

The research at hand might be limited by a *(i) selection bias of participants*, a *(ii) selection bias of courses*, the *(iii) used APAS* and *(iv) limited experience of lecturers and students with an APAS*. In order to limit *(i)* we asked all universities involved in the project and their lecturers to voluntarily participate in our empirical investigations (cf. Section 4.1). Moreover, we asked them to randomly select four students for the interviews (cf. Section 4.3). Accordingly, we had limited influence on the participants who agreed to take part in the empirical investigations. It is worth mentioning that those students who did not pass the respective course, unfortunately, did not participate in the expert surveys and interviews. This might possibly influence the results by introducing a more positive attitude towards the automated programming assessment system. The results might be limited by *(ii)*, since we primarily studied the use of the APAS by freshmen in introductory courses. This was also the primary goal of our research. Future research will also look at the use of APAS by advanced programmers in advanced software engineering courses. Limitation *(iii)* is only present to a certain extent, since we evaluated various APAS at the beginning of the project (cf. Section 3) and our investigations showed that the offered basic functionality is similar for all analyzed APAS. Furthermore, it should be noted that the primary goal of our work is to report on the lecturers' and students' experiences with basic functionalities of an APAS and not to evaluate the actual implementation of functionalities. Last but not least, in order to counteract limitation *(iv)* we conducted an introductory workshop for lecturers on how to use the APAS ArTEMiS (cf. Section 4.1). Moreover, students had the possibility to participate in an interactive tutorial provided by the APAS.

7. Conclusion & Outlook

In this paper, we provided an intervention study on lecturers' and students' experience with an APAS used at four different universities within the same country. In total 15 lecturers and 517 freshmen from different fields of studies participated in the study over the period of one semester. In order to learn more about their experiences 30 expert interviews, including 15 lecturers and 15 students, were conducted. Our investigation showed that an automated programming assessment system needs a tight integration in existing LMS and IDEs to increase user satisfaction. The interview results suggest that collaborative creation and peer-reviewing of exercises with corresponding test cases and feedback might increase exercise quality. Furthermore, individual learning paths might avoid the trial-and-error behavior of students while solving exercises. Our future work will focus on developing a user satisfaction model for APAS, approaches to crowdsource, share and peer-review programming exercises, and a concept for implementing individual learning paths based on competency models.

Acknowledgments

The CodeAbility Austria project has been funded by the Austrian Federal Ministry of Education, Science and Research (BMBWF).

References

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer science education*, *15*(2), 83–102.

Amelung, M., Krieger, K., & Rösner, D. (2011). E-assessment as a service. *4*(2), 162–174. https://doi.org/10.1109/TLT.2010.24

Barra, E., López-Pernas, S., Alonso, Á., Sánchez-Rada, J. F., Gordillo, A., & Quemada, J. (2020). Automated assessment in programming courses: A case study during the covid-19 era. *Sustainability*, *12*(18), 7451. https://doi.org/10.3390/su12187451

Campbell, J. L., Quincy, C., Osserman, J., & Pedersen, O. K. (2013). Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. *Sociological Methods & Research*, *42*(3), 294–320.

Cerioli, M., & Cinelli, P. (2008). Grasp: Grading and rating assistant professor. *Proceedings of the ACM-IFIP IEEIII 2008*, 37–51.

Chen, H. M., Nguyen, B. A., Yan, Y. X., & Dow, C. R. (2020). Analysis of learning behavior in an automated programming assessment environment: A code quality perspective. https://doi.org/10.1109/ACCESS.2020.3024102

Chen, P. M. (2004). An automated feedback system for computer organization projects. *IEEE Transactions on Education*, *47*(2), 232–240.

Coore, D., & Fokum, D. (2019). Facilitating Course Assessment with a Competitive Programming Platform. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 449–455. https://doi.org/10.1145/3287324.3287511

Daradoumis, T., Marquès Puig, J. M., Arguedas, M., & Calvet Liñan, L. (2019). Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming. *Computers and Education*, *128*, 159–170. https://doi.org/10.1016/j.compedu.2018.09.021

Edwards, S. H. (2004). Using software testing to move students from trial-and-error to reflection-in-action. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 26–30. https://doi.org/10.1145/971300.971312

English, J., & English, T. (2015). Experiences of using automated assessment in computer science courses. *Journal of Information Technology Education: Innovations in Practice*, *14*, 237–254.

Enström, E., Kreitz, G., Niemelä, F., Söderman, P., & Kann, V. (2011). Five years with kattis – using an automated assessment system in teaching. *2011 Frontiers in Education Conference (FIE)*, T3J-1-T3J–6.

Galan, D., Heradio, R., Vargas, H., Abad, I., & Cerrada, J. A. (2019). Automated Assessment of Computer Programming Practices: The 8-Years UNED Experience. *IEEE Access*, *7*, 130113–130119. https://doi.org/10.1109/ACCESS.2019.2938391

Goedicke, M., Striewe, M., & Balz, M. (2008). *Computer aided assessments and programming exercises with jack* (tech. rep.). ICB-Research Report.

Gomes, A., & Mendes, A. J. N. (2007). Learning to program-difficulties and solutions. *International Conference on Engineering Education*, 283–287.

Gordillo, A. (2019). Effect of an instructor-centered tool for automatic assessment of programming assignments on students' perceptions and performance. *Sustainability (Switzerland)*. https://doi.org/10.3390/su11205568

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. Á., Sheard, J., Skupas, B., Spacco, J., Szabo, C., & Toll, D. (2015). Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. *Proceedings of the 2015 ITiCSE on Working Group Reports*, 41–63. https://doi.org/10.1145/2858796.2858798

Jenkins, T. (2002). ON THE DIFFICULTY OF LEARNING TO PROGRAM. *3rd Annual LTSN-ICS Conference,Loughborough University*.

Karavirta, V., Korhonen, A., & Malmi, L. (2006). On the use of resubmissions in automatic assessment systems. *Computer science education*, *16*(3), 229–240.

Keuning, H., Jeuring, J., & Heeren, B. (2016). Towards a systematic review of automated feedback generation for programming exercises. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 41–46.

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)*, *19*(1), 1–43.

Knobbout, J., & Van Der Stappen, E. (2020). Where is the Learning in Learning Analytics? A Systematic Literature Review on the Operationalization of Learning-Related Constructs in the Evaluation of Learning Analytics Interventions. *IEEE Transactions on Learning Technologies*, *13*(3), 631–645. https://doi.org/10.1109/TLT.2020.2999970

Krusche, S., & Seitz, A. (2018). Artemis: An automatic assessment management system for interactive learning. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 284–289.

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: A systematic literature review. *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology* *in Computer Science Education*, 55–106. https://doi.org/10.1145/3293881.3295779

Marin, V. J., Pereira, T., Sridharan, S., & Rivero, C. R. (2017). Automated personalized feedback in introductory java programming moocs. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 1259–1270. https://doi.org/10.1109/ICDE.2017.169

Mekterovic, I., & Brkic, L. (2017). Setting up automated programming assessment system for higher education database course. *International Journal of Education and Learning Systems*, *2*.

Mekterović, I., Brkić, L., Milašinović, B., & Baranović, M. (2020). Building a comprehensive automated programming assessment system. *IEEE Access*, *8*, 81154–81172. https://doi.org/10.1109/ACCESS.2020.2990980

Pettit, R., Homer, J., Holcomb, K., Simone, N., & Mengel, S. (2015). Are automated assessment tools helpful in programming courses? *ASEE Annual Conference and Exposition, Conference Proceedings*, *122*.

Pettit, R., Homer, J., Gee, R., Mengel, S., & Starbuck, A. (2015). An Empirical Study of Iterative Improvement in Programming Assignments. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 410–415. https://doi.org/10.1145/2676723.2677279

Poženel, M., Fürst, L., & Mahnič, V. (2015). Introduction of the automated assessment of homework assignments in a university-level programming course. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 761–766. https://doi.org/10.1109/MIPRO.2015.7160373

Queirós, R., Pinto, M., & Terroso, T. (2020). Computer Programming Education in Portuguese Universities. *OpenAccess Series in Informatics*, *81*(21), 1–11. https://doi.org/10.4230/OASIcs.ICPEC.2020.21

Restrepo-Calle, F., Ramirez Echeverry, J. J., & Gonzalez, F. A. (2019). Continuous assessment in a computer programming course supported by a software tool. *Computer Applications in Engineering Education*, *27*(1), 80–89.

Restrepo-Calle, F., Ramírez Echeverry, J. J., & González, F. A. (2019). Continuous assessment in a computer programming course supported by a software tool. *Computer Applications in Engineering Education*, *27*(1), 80–89. https://doi.org/https://doi.org/10.1002/cae.22058

Restrepo-Calle, F., Ramirez-Echeverry, J. J., & González, F. (2020). Using an interactive software tool for the formative and summative evaluation in a computer programming course: An experience report. *Global Journal of Engineering Education*, *22*(3). http://www.wiete.com.au/journals/GJEE/Publish/vol22no3/06-Echeverry-J.pdf

Romli, R., Sulaiman, S., & Zamli, K. Z. (2015). Improving Automated Programming Assessments: User Experience Evaluation Using FaSt-generator. *Procedia Computer Science*, *72*, 186–193. https://doi.org/10.1016/j.procs.2015.12.120

Rubio-Sánchez, M., Kinnunen, P., Pareja-Flores, C., & Velázquez-Iturbide, Á. (2014). Student perception and usage of an automated programming assessment tool. *Computers in Human Behavior*, *31*, 453–460. https://doi.org/https://doi.org/10.1016/j.chb.2013.04.001

Souza, D. M., Felizardo, K. R., & Barbosa, E. F. (2016). A Systematic Literature Review of Assessment Tools for Programming Assignments. *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, 147–156. https://doi.org/10.1109/CSEET.2016.48

Ullah, Z., Lajis, A., Jamjoom, M., Altalhi, A., Al-Ghamdi, A., & Saleem, F. (2018). The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education*, *26*. https://doi.org/10.1002/cae.21974

Vargas, H., Heradio, R., Chacon, J., De La Torre, L., Farias, G., Galan, D., & Dormido, S. (2019). Automated Assessment and Monitoring Support for Competency-Based Courses. *IEEE Access*, *7*, 41043–41051. https://doi.org/10.1109/ACCESS.2019.2908160

Yan, Y.-X., Wu, J.-P., Nguyen, B.-A., & Chen, H.-M. (2020). The impact of iterative assessment system on programming learning behavior. *Proceedings of the 2020 9th International Conference on Educational and Information Technology*, 89–94. https://doi.org/10.1145/3383923.3383939