







# Can I Code? User Experience of an Assessment Platform for Programming Assignments

Anne Münzner   

Center for Human-Computer Interaction, University of Salzburg, Austria

Nadja Bruckmoser 

University of Salzburg, Austria

Alexander Meschtscherjakov   

Center for Human-Computer Interaction, University of Salzburg, Austria

---

## Abstract

Learning a programming language is a difficult matter with numerous obstacles for university students – but also for their lecturers. Assessment tools for programming assignments can support both groups in this process. They shall be adapted to the needs of beginners and inexperienced students, but also be helpful in long-term use. We utilised an adapted version of the Artemis system as an assessment platform for first-year computer science students in the introductory programming course. To examine the students' user experience (UX) over the semester, we conducted a three-stage online questionnaire study (N=42). We found that UX evolves over the semester and that platform requirements and problems in its usage change over time. Our results show that newcomers need to be addressed with caution in the first weeks of the semester to overcome hurdles. Challenges shall be added as the semester progresses.

**2012 ACM Subject Classification** Human-centered computing

**Keywords and phrases** Programming tool, user experience, student evaluation, programming assignment

**Digital Object Identifier** 10.4230/OASICS.ICPEEC.2021.18

**Category** Short Paper

**Funding** CodeAbility is funded by the Federal Ministry of Education, Science and Research as part of the structural funds for higher education.

## 1 Introduction

Learning to program can be a difficult task. Especially in the university context, when students are in the first semesters of their studies, they are not yet routinized and experienced with academic learning and problem solving and a completely new stage of life has often started with the study period [16, 7]. Students often have different levels of knowledge in programming. Some have already been taught solid basics in school, others have already experimented in the personal field, and still others have no previous experience at all.

But it is not easy for the lecturers either. They have to serve these different knowledge bases and are primarily occupied with creating and correcting suitable tasks, often for a large number of students. They have to create structures and incentives that make students do what is most important for learning a programming language: repeated intensive practice [13].

To address this problem, we have launched the CodeAbility project at Austrian universities which aims to standardize university programming education across Austria and create an infrastructure with an online learning platform, didactic concepts, and an exchange platform for teachers. The primary goal is to develop the online learning platform that integrates into the existing classroom using blended learning and, in the future, uses programming learning analytics to enable individual learning paths. We want to develop a platform that acts as a helping tool on the difficult path of learning to program and is also perceived as such.



© Anne Münzner, Nadja Bruckmoser, and Alexander Meschtscherjakov;  
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 18;  
pp. 18:1–18:12



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Since frustration and fear are true learning killers [1], these must be avoided at all costs. Especially students who have never had contact with programming or computer science in general, have fears and doubts which create barriers that keep these people from learning. Especially for this group, it is important to convey a sense of self-efficacy, competence, security, and autonomy through the tool. This means not only avoiding frustration due to poor usability but specifically promoting positive experiences because lack of motivation is one of the main barriers to learning to program [1, 8].

At the moment, we are exploring what functionalities and content the platform should offer. Since the experience with delivery systems in higher education has been very positive [20] and support was needed most in this area, the functionality of the delivery platform was the first one we implemented and tested. In the first pilot phase of the project, Artemis, the automated assessment management system for interactive learning, was used in various courses at Austrian universities to create and submit exercises. Artemis was developed at the Technical University of Munich, Germany [10]. It is especially suited to supervise large groups and has been successfully used in MOOCs [11]. We have used the system at the University of Salzburg in the courses “Introduction to Programming with Java” and “Introduction to Programming with Python”. Both courses are aimed at students in the first bachelor’s semester of computer science and DIG (Digitization Innovation Society).

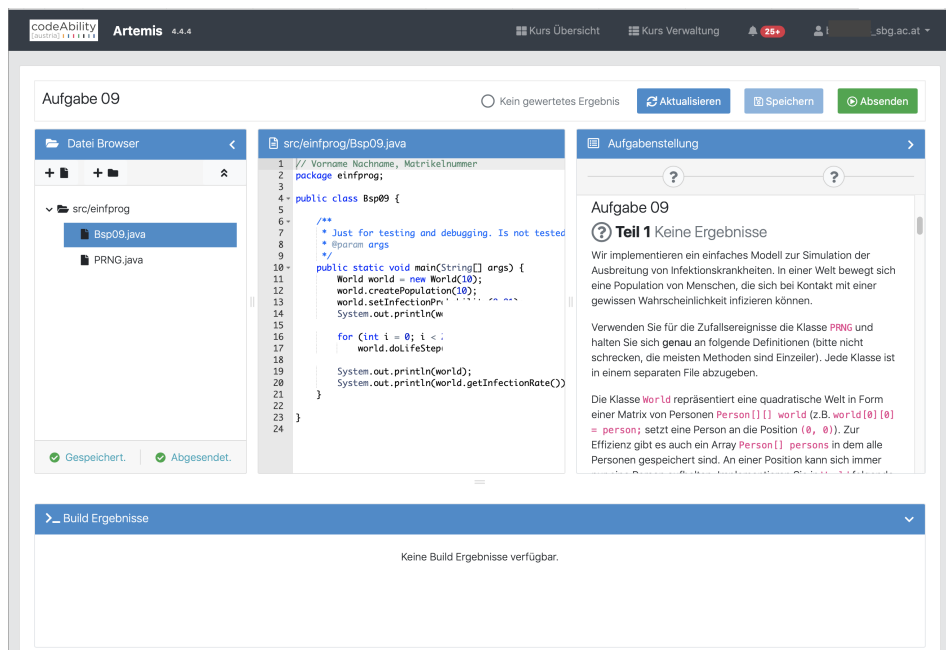
Figure 1 provides a screenshot of the Artemis platform as it was experienced by the students. An assignment was given that had to be submitted by a certain date. The interface includes a file browser (left), an area for the code that has to be developed (center), and the description of the assignment (right).

Today, there are many different assessment tools for programming assignments that save time and effort through automation [19] and have already been successfully used for programming courses at universities [2]. The requirements of lecturers and students in this context have also already been investigated [16].

As already stated, one of the problems concerns the motivation, i.e. also the emotional level of the students. We aim to develop a platform that also addresses this problem. Therefore, our main concern was to learn more about the students’ user experience (UX). It is already known that the UX of programs evolves over time with increasing experience [9]. For us, it was important to find out to what extent this also applies to our learning platform. As mentioned before, students are at a point in time where they are confronted with many new challenges, so it is particularly relevant to identify entry problems. But also the needs of students with growing experience should be recognised to optimally support different phases of learning programming. This leads to the following research questions:

- *How does the user experience of the learning platform changes during the first semester?*
- *How do requirements and reported problems change over the semester?*
- *How should deployed learning platforms be evaluated to capture the needs of all students?*

To answer these questions, we conducted a user study at the University of Salzburg with students who used the platform in their first semester. For this purpose, we sent out online questionnaires at three different times in the winter semester 20/21 (October 2020 - January 2021). Our study showed that the UX of the platform developed over the semester. We were given insights into existing problems with the platform that especially novice programmers found obstructive. Moreover, we were able to identify functions and aspects that students found particularly supportive. In the following, we will reflect on related work. We will describe the applied method in detail and report results. Lastly, we will discuss findings and provide answers to our research questions.



**Figure 1** This screenshot of the Artemis platform shows a file browser (on the left), a java code (in the center), and a student assignment (on the right). At the bottom information and errors would be reported. The green button at the top right allows the student to submit the code. An automated analysis (e.g., syntax errors, compilability) is performed and the student would get instant feedback at the bottom.

## 2 Related work

Solutions to the challenges of teaching programming are being explored in many countries through higher education institutions. Queirós et al. [16], for example, conducted a study in Portuguese universities to gain a deeper understanding of the programming teaching process in such higher education institutions. Their results show the need for helping tools that support teachers as well as students and automate parts of this process.

Cardoso et al. [2] conducted a study on the use of automated assessment tools for first semester students and demonstrated that the tools support and motivate students to learn programming.

The automated assessment system we used was Artemis, developed at the Technical University of Munich. In a quantitative study, its scalability, feedback under 10 seconds and good usability were proven [10]. The latter resulted from the observation that novice programmers are able to use the platform without problems. In addition, students gave positive feedback on the usefulness of the platform.

This proves that the use of automated assessment tools at universities is useful for teachers and students. For us, it was important to investigate not only the usability but also the UX in more detail in order to optimize the use of this tool, gain further insights into the needs of students over the semester, and thus gather knowledge for further development in the CodeAbility project.

There have been numerous studies on the change of UX and usability over time [15, 9]. Kujala et al. [12] describe a change in the context of an application over time under what they call long-term UX. It is the nature of learning to change learners through the growth of

knowledge and skills that build upon each other. It is a process with which the context, the target group of students changes. Since the tool accompanies the students in this process for months, the question of long-term UX is particularly relevant. When dealing with user experience in the context of learning, eudaemonic well-being is also important, as the enjoyment of the immediate activity is not always present or visible, but this activity can lead to increased well-being in the long run [5].

### 3 Methodology

#### 3.1 Objective of the study

With our study, we wanted to gain deeper insights into the student perspective on the platform. Thus, the goal of the study was not simply to evaluate the existing platform, but to find out, with respect to the CodeAbility project, how students perceived the use of the platform, what is perceived as helpful, and where we need to change aspects. Since the platform is intended to support students as they get started in programming and to work for students who are less familiar with computer science, we evaluated the UX over the course of the semester. We were interested in whether the programming learning platform is rated differently over this time, for example, whether students rate it better with increasing experience or not. Thus, the overall objective of the study was the following: *Does the UX change over the semester and do students have different needs and issues with the platform at different points in time?*

#### 3.2 Procedure

To evaluate UX, we used questionnaires that were sent to the students via the course instructors on three different dates during the semester. Typically, students had weekly classes and weekly assignments during the semester (i.e. over a four-month period). In the lectures, the students learned basic programming constructs such as data types, if-cases, and loops. A weekly task on the topic of the last lecture was then published via the Artemis platform. These assignments were divided into two subtasks that build on each other. Students could score 0 points if neither task passed the tests, 5 points if the first task passed the tests, and 10 points if both subtasks passed the tests. The students had to solve it in the course of the week. This task then had to be submitted via the Artemis platform either using the online editor or the version management system GIT. An instant check of the assignment was performed automatically and the student would get feedback from the Artemis platform. This feedback is based on the test cases created by the teachers. In addition to the compiler output, the students also received output on other tests, such as input output tests, and thus knew directly whether their solution passed all the tests. Students could resubmit their assignments as often as they wanted until the deadline. Thereafter, the solutions and the scores achieved were viewed by the lecturers and possibly assessed manually. On the platform, students could view their submitted assignments and achieved points.

The first survey was sent out six weeks after the start of the semester so that any initial difficulties were still fresh, but the students had already gained some experience with the platform. After another month a second questionnaire was sent out to students. The last questionnaire was sent out at the end of the semester after another six weeks.

We used the same questionnaire for all three dates. Demographic data from the participating students were collected in the first part of the questionnaire. The main purpose of this was to gain knowledge about the exact target group and find out whether it is a

relatively heterogeneous group or if there were differences among participants that need to be addressed. Here, we also asked questions about the students' previous experience in order to identify any correlations between this and the user experience of the platform.

The main part of the questionnaire was the *User Experience Questionnaire UEQ+*. The original UEQ was developed by Laugwitz et al. [14] and targeted at the assessment of UX of interactive systems in general. It measures the subjectively perceived impression of the users, mapped in six different dimensions: *Attractiveness*, *Efficiency*, *Transparency*, *Controllability*, *Stimulation*, and *Originality* [17]. The UEQ+ is a modular extension of the UEQ. It is composed of 26 bipolar questions rated using a 7-point Likert scale (coded from -3 to +3). The UEQ+ allows selecting various dimensions that are relevant for the respective product. We have ultimately used the scales *Attractiveness*, *Efficiency*, *Perspicuity*, *Dependability*, *Usefulness* and *Clarity* as UX dimensions.

The following part of the questionnaire included questions that further addressed the context of the application. These were questions about the platform's support for learning the programming language, the submission of assignments, and related error messages. These items were self-designed and had to be answered using a 10-level Likert scale from one (not at all) to 10 (very good).

The third part of the questionnaire consisted of open-ended questions about obstacles and difficulties as well as advantages, suggestions for improvement and wishes. They not only aimed to measure the user experience but also to provide possible justifications for the evaluation of this in the UEQ+ [18]. In addition, questions were intended to reveal further problems and potentials in the interaction at different points in the semester and provide the opportunity to obtain more qualitative answers [6].

### 3.3 Results

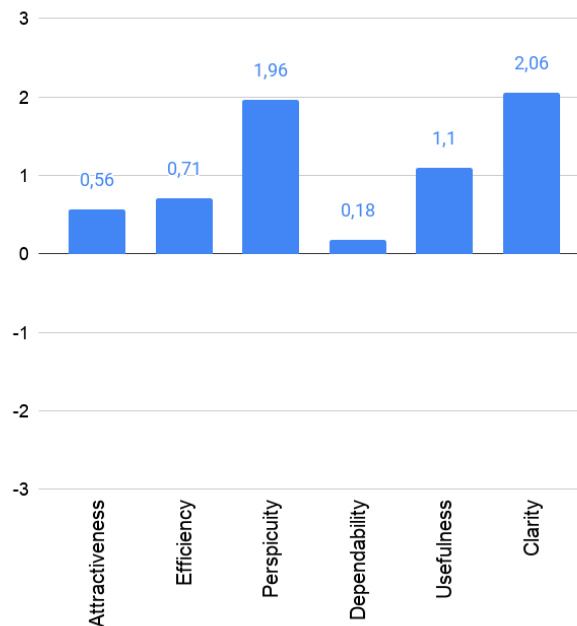
#### 3.3.1 First Survey

A total of 61 students participated in the first survey. Of these, 42 students responded to the questionnaire in full. Since the introductory courses are intended for the first semester, 54.34% of the participants are in the first semester of their studies. The remaining participants are distributed among the other semesters.

12.8 % of the participants had no computer science classes in school and the majority with 31.9 % had 2 years of computer science classes in school. However, there are also some students (8.5 % of the participants) who have had computer science lessons in school for as long as eight years. This means that the previous school experiences were quite different. Figure 2 shows the results of the first UEQ+ questionnaire. Overall all six dimensions were rated positively (i.e. mean values above 0 on the scale between -3 and +3). *Dependability* scored worst with a value of 0.18. *Attractiveness* with 0.56 and *Efficiency* with 0.71 did not score very high either. *Usefulness* scored better with a value of 1.1. The best scores were *Perspicuity* with 1.96 and *Clarity* with 2.6. So far, we have not found any differences in the rating of the UEQ+ between students with different levels of prior knowledge.

The answers to the context-related questions were striking. They were answered very diversely. To the question "Artemis supports me in working on the homework", 14.3 % answered with *not at all* (0 points on the Likert scale) and the same number answered with *very well* (10 points on the Likert scale). The answer to the question "The error messages in the homework helped me" was similarly varied. Here, the proportion of those who voted neutral or worse (one to five points on the Likert scale) predominated.

## 18:6 Can I Code?



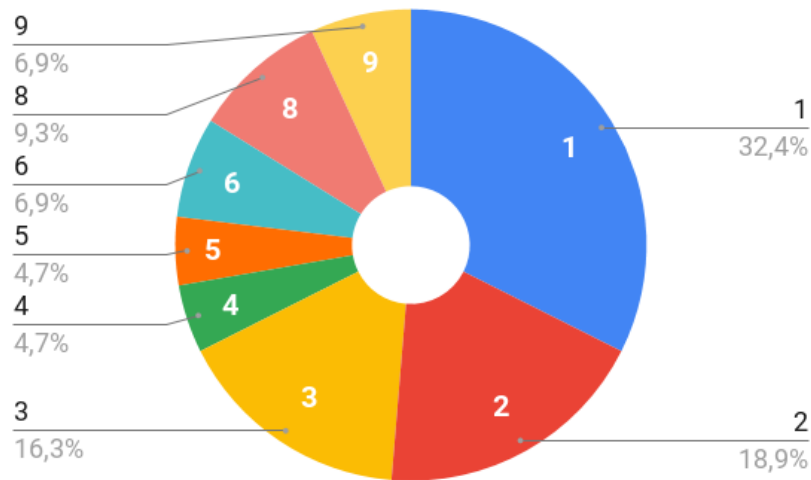
■ **Figure 2** Results of the UEQ+ of the first phase of the study. Ratings were given on a scale between -3 and +3. Mean values were all positive. *Dependability* scored worst. *Attractiveness* and *Efficiency* did not score very high. *Usefulness* scored better. The best scores were *Perspicuity* and *Clarity*.

However, the proportion of neutral to poor evaluations predominates in the question “Artemis has supported me in learning the programming language”. As shown in figure 3, 32.4 % of the participants stated that Artemis did not support them at all in this respect. Again, for all three questions, we have not yet found any differences in perceived support between students with different levels of prior knowledge.

In order to interpret these numbers, we shall have a look into the open questions. When asked what obstacles were encountered when working on and handing in the exercises, the majority of the participants criticized the quality of the error messages for the programmed exercises. These were described as inaccurate or incomprehensible. This problem is also reflected in the students’ wishes for improved and more detailed error messages that are also understandable for beginners.

Some students also found the submission via GIT difficult, especially for the first task, as they had to deal with the version management system in addition to the new algorithmic thinking. This problem is also reflected in the students’ requests for instruction for GIT.

But which aspects did the students rate positively? The immediate feedback of the platform on the submitted task was often praised as positive, even if the quality of the error messages was often criticized. A total of 24 of the 42 participants mentioned this as positive in the open questions. The students liked the fact that they could immediately see where their programs contained errors and correct them before the actual submission. In addition, they immediately had the certainty that their program corresponded to the created tests. So they did not have to wait for the lecturers to correct their solution.



■ **Figure 3** Results of the first study on the question: “Artemis has supported me in learning the programming language”. From 1 (not at all) to 10 (very good).

### 3.3.2 Development over time

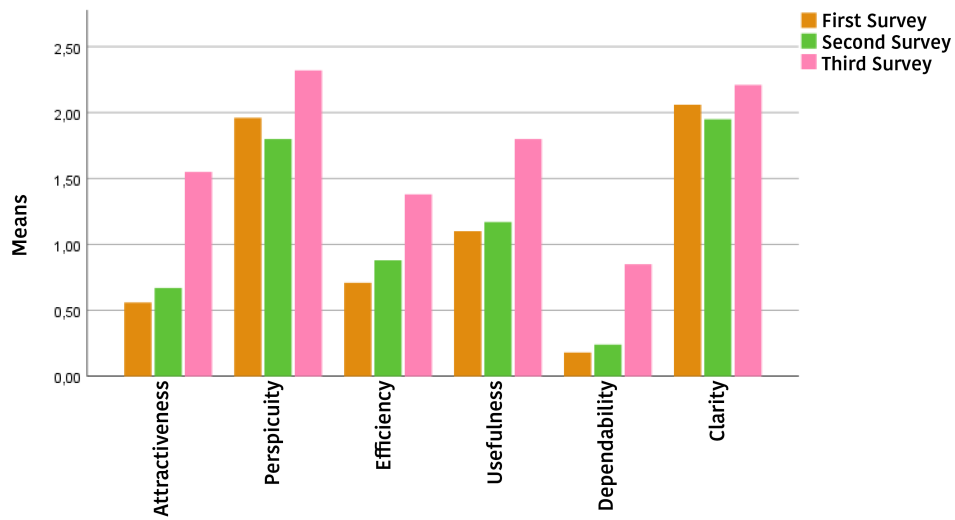
The comparison with the subsequent surveys showed that the UX of the platform increases in all the scales of the UEQ+ we examined as the semester progressed. In the second survey, the UEQ+ was still rated similarly to the first survey round. Either no or only minor improvements could be demonstrated. In the third round of the survey, greater improvements were detected. We saw a big increase in the dimensions *Attractiveness*, *Efficiency*, and *Usefulness*. Figure 4 shows detailed results of the UEQ+ over time.

The ratings of contextual questions also improved. In the last survey, for example, 11.5% did answer *not at all* to the question “Artemis supported me in learning the programming language”. For comparison: at the beginning of the study, 32.4% of the participants did so. The improvement can be seen in figure 5. The number of participants neglecting capability of the platform to support learning programming decreased significantly. Also, the support of the platform in the processing of the homework was rated better as the semester progressed as shown in figure 6. In the first survey participants were evenly spread across the answer possibilities (1 to 10). In the third survey, only a minority of participants stated that the platform would not be supportive for the assignments.

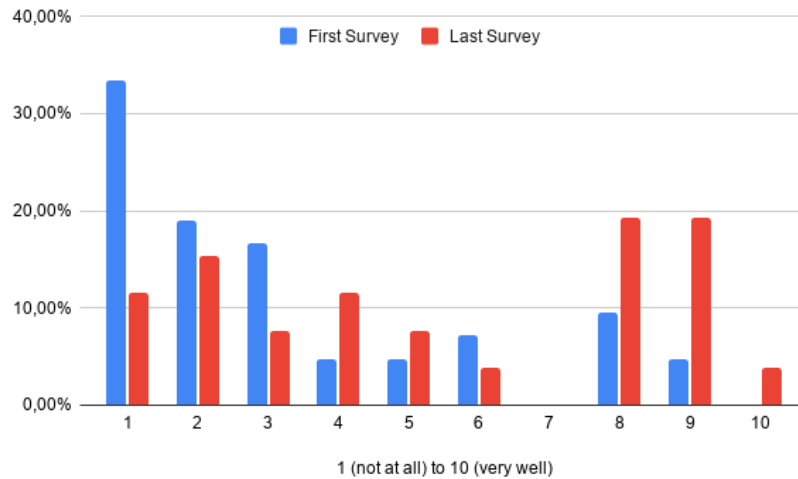
So are the problems identified in the first survey only the entry barriers described by Mendoza and Novick [15]? The answers to the open questions at the end of the questionnaire could give some clues. Now among the problems the lack of design freedom in the programming tasks, which does not allow own approaches since the system only accepts solutions that have been designed exactly according to the specifications, is mentioned much more often. However, experienced students would like to implement their own or more creative solutions. Even though the error outputs were still criticized, the complaints were more about the inaccuracy of these outputs.

The immediate feedback was also repeatedly emphasized as positive in these survey rounds. Furthermore, the students found it helpful to have all assignments in one place and thus to have an overview also of homework already handed in.

## 18:8 Can I Code?

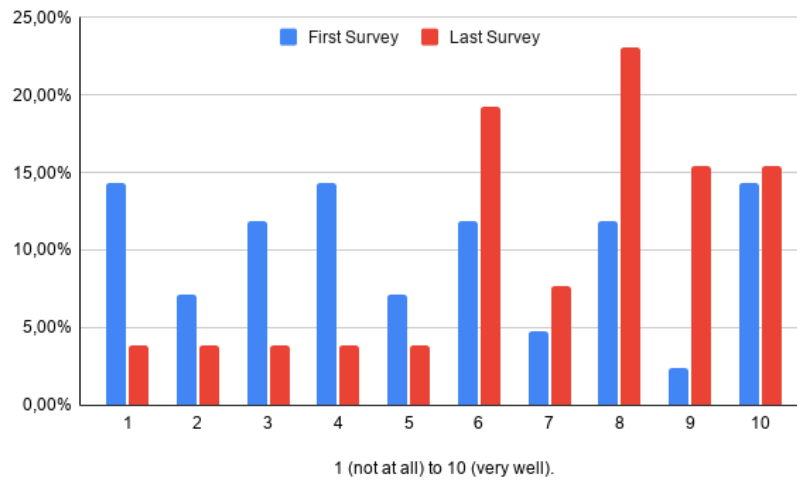


■ **Figure 4** Results of the UEQ+ mean values over time. Mean values were taken from the three points in time of the study when the UEQ+ was asked. Especially the dimensions attractiveness, efficiency, usefulness, and dependability were rated higher at the end of the semester than at the beginning or during the semester.



■ **Figure 5** Detailed results on the statement "Artemis supported me in learning the programming language". Results show an increase in the capability of the platform as a support for programming language learning between the first and third survey.





**Figure 6** Detailed results on the statement “Artemis supported me in working on the assignments”. Results show an increase in the platform support for homework completion between the first and the third survey.

## 4 Discussion

In our study, we found that the user experience of using the Artemis platform at the beginning of the semester was lower compared to later points in the semester. In the beginning, the platform was not perceived as a support for learning to program by the majority of students. This was due to the fact that in addition to the new concepts of programming and algorithmic thinking that the students had to deal with at this point, they also had to learn and understand how to use and operate the Artemis platform. In addition to recognizing the workflow, completely new concepts and tools such as the version management system GIT and error output had to be understood.

On the one hand, this finding is gratifying, as it shows that the UX is perceived as more positive in the longer term and that the platform functions as a support for learning to program. Even though this development of the user experience has been proven over time in several studies, this finding is still very critical for us and no reason to rest on it, because as mentioned at the beginning, we are also concerned with simplifying the entry into programming.

### 4.1 Recommendations

If you want to analyze the user experience of learning platforms in higher education, our clear recommendation is to do this not only once and only at the end of the semester. Especially initial difficulties, which may have already led to leaving the course, can no longer be recorded here.

For many students, the online learning platform is their first point of contact with programming. The resulting responsibility and the potential to arouse students’ interest in programming and to create motivation for the rest of the semester should be used. Here the chance exists to get negative expectations and prejudices about programming [3, 8] out of the way.

Following the possibility-driven design approach [4], the potentials of an interactive learning platform should be used to create truly positive experiences here. According to our study, in addition to the clarity of the tasks and observation of one's progress, one of these potentials is the possibility of receiving immediate feedback. Isn't that also the beauty of programming and logic? Let's remember the moment when (maybe after a long time of debugging) the compiler no longer gives an error message and the algorithm does what it is supposed to do. Programming can be fun and this should be conveyed right at the beginning. Obstacles that stand in the way of this experience must be removed.

Of course, the platform must also be supportive for learning success in long-term use. This includes, for example, the use of GIT since it is a reality in later practical work on the job. In the same way, searching for errors and evaluating error messages that are often inaccurate or incomprehensible is part of a programmer's everyday work. These concepts must be learned, but should not overwhelm students all at once in their first programming assignment. They should be introduced gradually. For example, error messages could be filtered initially. In any case, the question of how feedback should be designed to support students at different levels of experience is worth exploring. In addition, as students become more experienced, the platform should also provide the opportunity for their own more creative approaches to solving the programming tasks, because the ability to find alternatives and to work on programming problems with different approaches is also important.

So it is obvious that a programming learning platform for students should function as a dynamic companion on the way of learning to program. With concepts like Programming Learning Analytics and Gamification, we want to achieve this. Whatever concepts we introduce in the future as part of the CodeAbility project, we will always pay special attention to what impact they have on the entry of still inexperienced students who have just started programming.

## 4.2 Limitations

In the pilot phase, the Artemis platform was used to create and submit weekly assignments and provide feedback on their correctness. We conducted our study only in the two courses that used the platform at the University of Salzburg (N=42). Thus, our results cannot be generalized to online learning platforms. This would require a further increase in the number of study participants. Furthermore, the platform used was limited to its function as a submission system for students. It remains to be investigated whether a learning platform with more extensive functionalities, such as teaching content, would be evaluated differently. Not to be neglected here is also the fact that the platform was used to submit weekly compulsory assignments, which influenced the grading of the students.

## 5 Conclusion

To support students and teachers at Austrian universities in learning and teaching programming, we started the CodeAbility project. In the first pilot phase of the project, we used the assessment tool Artemis to automate the creation and submission of assignments. In our study, we examined the user experience over the semester. According to our research questions, we found the following:

- *At the end of the semester, the user experience of the platform is rated better than at the beginning of the semester in all scales of the UEQ+, i.e. Attractiveness, Efficiency, Perspicuity, Dependability, Usefulness and Clarity. However, real differences in the evaluation could not yet be determined in the second, but only in the third survey at the end of the semester.*

- *At the beginning of the semester, in addition to the new computational thinking, students still struggle with initial difficulties in using the platform. Later in the semester, the problems with the platform and the students' demands change. Students no longer feel overwhelmed with new concepts and rather want more freedom in programming.*
- *To represent the user experience for the entire students, it makes sense to investigate of the user experience at least two different times, i.e. at the beginning and the end of the semester. This procedure captures both the entry problems of programming beginners and the needs of advanced students, which can differ considerably.*

In a future study, it would be helpful to conduct this survey with a larger number of students e.g. the other universities involved in the CodeAbility project.

On the one hand, our findings can be helpful for the evaluation of learning platforms, as it shows the requirement for repeated user experience studies. On the other hand, they can be useful for the design of learning platforms, as our findings show the need to address the different needs at the beginning and later in the semester. At the beginning of the semester, any obstacles that make it even more difficult to get started with programming should be eliminated. Then, as the semester progresses, additional concepts that students will need later should continue to be introduced. The platform should thus evolve along with the students.

---

## References

- 1 Christine Bruce, Lawrence Buckingham, John Hynd, Camille McMahon, Mike Roggenkamp, and Ian Stoodley. The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, 9, January 2010. doi:10.28945/1183.
- 2 Marílio Cardoso, António Vieira de Castro, Álvaro Rocha, Emanuel Silva, and Jorge Mendonça. Use of Automatic Code Assessment Tools in the Programming Teaching Process. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 4:1–4:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.4.
- 3 Sapna Cheryan, Allison Master, and Andrew Meltzoff. Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in psychology*, 6:49, February 2015. doi:10.3389/fpsyg.2015.00049.
- 4 Pieter Desmet and Marc Hassenzahl. *Towards Happiness: Possibility-Driven Design*, pages 3–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-25691-2\_1.
- 5 Sarah Diefenbach and Marc Hassenzahl. *Psychologie in der nutzerzentrierten Produktgestaltung*. Springer-Verlag Berlin Heidelberg, January 2017. doi:10.1007/978-3-662-53026-9.
- 6 Susan Farrell. Open-Ended vs. Closed-Ended Questions in User Research, 2016. URL: <https://www.nngroup.com/articles/open-ended-questions/>.
- 7 Anabela Gomes and António José Nunes Mendes. Learning to program-difficulties and solutions. *International Conference on Engineering Education*, 2007.
- 8 Tony Jenkins. ON THE DIFFICULTY OF LEARNING TO PROGRAM. In *3rd Annual LTSN-ICS Conference, Loughborough University*, 2002.
- 9 Evangelos Karapanos, Marc Hassenzahl, and Jean-Bernard Martens. User experience over time. In *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08*, page 3561, New York, New York, USA, 2008. ACM Press. doi:10.1145/1358628.1358891.
- 10 Stephan Krusche and Andreas Seitz. ArTEMiS - An automatic assessment management system for interactive learning. In *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018. doi:10.1145/3159450.3159602.

- 11 Stephan Krusche and Andreas Seitz. Increasing the Interactivity in Software Engineering MOOCs - A Case Study. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019. doi:10.24251/hicss.2019.915.
- 12 Sari Kujala, Marlene Vogel, Anna E. Pohlmeyer, and Marianna Obrist. Lost in time: The meaning of temporal aspects in user experience. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 559–564, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2468356.2468455.
- 13 Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, page 14–18, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1067445.1067453.
- 14 Bettina Laugwitz, Martin Schrepp, and Theo Held. Konstruktion eines Fragebogens zur Messung der User Experience von Softwareprodukten. In *Mensch und Computer 2006*, pages 125–134. OLDENBOURG WISSENSCHAFTSVERLAG, München, 2006. doi:10.1524/9783486841749.125.
- 15 Valerie Mendoza and David G. Novick. Usability over time. In *Proceedings of the 23rd annual international conference on Design of communication documenting & designing for pervasive information - SIGDOC '05*, page 151, New York, New York, USA, 2005. ACM Press. doi:10.1145/1085313.1085348.
- 16 Ricardo Queirós, Mário Pinto, and Teresa Terroso. Computer Programming Education in Portuguese Universities. *OpenAccess Series in Informatics*, 81(21):1–11, 2020. doi:10.4230/OASICS.ICPEC.2020.21.
- 17 Maria Rauschenberger, Martin Schrepp, and Jörg Thomaschewski. User experience mit fragebögen messen - durchführung und auswertung am beispiel des ueq. In *In Usability Professionals Konferenz 2013*, September 2013.
- 18 Heike Sandkühler, Martin Schrepp, and Jörg Thomaschewski. Ux messung mithilfe des ueq+ frameworks. In Christian Hansen, Andreas Nürnberger, and Bernhard Preim, editors, *Mensch und Computer 2020 - Workshopband*, Bonn, 2020. Gesellschaft für Informatik e.V. doi:10.18420/muc2020-ws105-244.
- 19 Draylson M. Souza, Katia R. Felizardo, and Ellen 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)*, pages 147–156. IEEE, April 2016. doi:10.1109/CSEET.2016.48.
- 20 Anne Venables and Liz Haywood. Programming students need instant feedback! In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20*, ACE '03, page 267–272, AUS, 2003. Australian Computer Society, Inc.