Research Article

Paula Correia Tavares*, Elsa Ferreira Gomes, Pedro Rangel Henriques, Diogo Manuel Vieira

Approaches to manage and understand student engagement in programming

https://doi.org/10.1515/edu-2022-0005
received November 15, 2019; accepted September 27, 2021.

Abstract: Computer Programming Learners usually fail to get approved in introductory courses because solving problems using computers is a complex task. The most important reason for that failure is concerned with motivation; motivation strongly impacts on the learning process. In this paper we discuss how techniques like program animation, and automatic evaluation can be combined to help the teacher in Computer Programming courses. In the article, PEP system will be introduced to explain how it supports teachers in classroom and how it engages students on study sessions outside the classroom. To support that work, students’ motivation was studied; to complement that study, a survey involving students attending the first year of Algorithms and Programming course of an Engineering degree was done. It is also presented a tool to analyse surveys, using association rules.

Keywords: Automatic Evaluation; Educational Data Mining; Motivation; Program Animation; Association Rules.

1 Introduction

It is known that students have problems learning how to program, and consequently they fail in Computer Programming courses (Proulx, 2000). This fact is corroborated by the international community of Programming teachers that decided in 2020 to meet in a conference--ICPEC, International Computer Programming Education Conference--to discuss the problem and promote the development of pedagogical methods and tools to overcome some of the problems. Its proceedings (ICPECa, 2020) gather a collection of relevant papers concerned with approaches to overcome the difficulties faced by Computer Programming students. The success of the referred forum led that community to organize a new edition in the present year (ICPECb, 2021) where around twenty authors presented new tools and studies, namely those of the invited Speakers: “The revolution in computing education at school: opportunity and challenge” by Simon Peyton-Jones; “Learning programming in school: the teacher’s toolkit” by Sue Sentence; and “Challenges of teaching novice computer programming - Studies and reflections” by Anabela Gomes.

The reasons of this failure are not consensual (Robins, 2015). However, it is known that: students give up and do not finish a given programming task as soon as they find a difficulty; they do not understand the requirements; they cannot devise the necessary algorithm to solve the problem; they do not master the programming language to implement properly the algorithm.

Proulx (Proulx, 2000) identified some of the most important difficulties faced by students when a traditional learning method is used, namely:

- Understanding the problem, because they are not experts nor knowledgeable on the subject or they are unable to understand the problem statement (correctly interpreting its meaning);
- Thinking logically, to split the problem into simpler components and design the necessary algorithm to solve properly the problem;
- Learning fully the syntax and the semantics of the programming language chosen to implement the algorithm.

One challenge for teachers and an important factor in learning is to motivate students (Gomes et al., 2018). It is important to understand why students do lack motivation (Almeida, 2012). For example, when students have to complete their homework, they can do it for (at least) two different reasons. They can either do it for the sake of teacher’s approval, which will not present any interest or
curiosity in the content of the task itself or, instead, they can be motivated by the challenge and the knowledge it will bring to them.

The aim of this work can be summarized in three points: 1) the understanding of the reasons behind the difficulties that occur in the process of teaching/learning of programming, 2) the study of computer-supported approaches to avoid these difficulties and 3) the suggestion of ways of combining all these approaches to increase students curiosity and interest (Tavares, Henriques & Gomes, 2016).

Furthermore, we present two techniques to support the teaching of programming: Program Animation and Systems for Automatic Evaluation of the Programs. The Program Animation permits the visual simulation, helping the understanding of the algorithms and programs and the Systems for Automatic Evaluation of the Programs motivates students on their working with an immediate feedback as soon as they finish writing a program. According to our experience, it is very important to motivate students to practice programming exercises early in the course. It is also important to give them an immediate feedback in this phase (Verdú et al., 2011), indicating the result and, in the case of being wrong, to explain the cause of the error ad how to correct it.

Taking into account the extensive research done, a web-based learning support tool, called PEP (“Plataforma para o Ensino da Programação” or Platform to Programming Learning), was developed to help the teachers and students either in the classroom or at home. Students' motivation was then carefully studied to support that development. Complementary, a survey to first year students of Algorithm and Programming course of an Engineering degree was performed. Along the paper, we show how Association Rules (AR) can be used to mine the data collected through the inquiry to extract relations among reasons influencing extrinsic motivation.

The article is organized in seven sections. Section 2 describes the theory of student’s motivation and its relevance for achieving success in learning how to program; in section 3 we refer to the learning process itself; in section 4 we describe the two techniques that we used in our work (animation and automatic evaluation); in section 5 we describe an experiment combining existing animation and automatic evaluation tools and we present an online platform for learning how to program and in section 6 we apply the association rules based tool FERA (“Ferramenta de data mining para dados educacionais”, or a data mining tool for educational data) to study the data obtained from students’ surveys to understand engagement.

2 Student’s Motivation

Several theories have been developed to explain motivation from the beginning of the history of psychology as a science. Because it is a complex phenomenon, the subject has been studied under different prisms. Some of them claim that people are motivated by material rewards, others by increasing their power and prestige in the world, or by an interesting work, enriched environments, recognition, or to be respected as an individual. The fact is that humans in general have very complex needs and desires. Motivation is one of the keys to understand the human behaviour; it acts on the thought, attention, emotion and action of the Human Being, involving desire, effort, dreams and hope (Williams & Williams, 2011) (Almeida, 2012) (Ochman, Podpora & Kawala-Janik, 2015) (Reeve, Deci &Ryan, 2004) (Pintrich, 2003).

One of the main reasons of the students' failure in the introductory programming courses is the lack of motivation during the learning process (Mohanarajah, 2018). To motivate students is a big challenge for a teacher (Callahan, 2010) and the motivation needed to involve each student in a given task depends on their belief for success and on the importance that the student gives to it.

According to the theory of self-determination, the motivation can be intrinsic or extrinsic, having different degrees that vary in a continuum range from demotivation (absence of motivation) to intrinsic motivation, going through extrinsic motivation (Ryan & Deci, 2000). The intrinsic motivation does not need any external factor. It derives from the student herself/himself as the dedication, competence, willingness and ability to accomplish a task. Extrinsic motivation is the result of external factors, such as the resources that the student has, the incentives, and the environment where it develops her/his tasks (Silva et al., 2014). Both work together and the result will set the student’s behaviour, as shown in the Figure 1 (Tavares, Henriques & Gomes, 2017). The motivational and emotional factors have a direct relation with the learning process. So, teachers must understand it to improve their role in the classroom (Schunk, Meece & Pintrich, 2014).

3 Learning Programming

Programming is, first of all, solving problems using a computer. This task requires the definition of strategies to choose and organize elementary instructions to carry out the operations necessary to find the solutions. These instructions need to be translated to computer commands written in a programming language.
In that context, usually students misunderstand two different notions, learning programming, and learning the syntax and semantics of a particular programming language.

As said above, programming is much more than writing a code in each language; it also involves other steps as the understanding of the problem statement, the algorithm development, or the test of the program.

The authors of the guidelines for introductory Computer Science courses (ACM/IEEE, 2013) argue that most of the courses have a “programming focus” character (here “programming” means “coding”). In that document, they alert the importance that students do not perceive computer science as only learning a particular programming language but the concepts of learning how to program. They also sustain an interesting discussion on “Programming Paradigm and Choice of Language” recommending the lecture of alternative programming paradigms. Also, Gomes (Gomes, 2010) showed that the adopted paradigm and the language used strongly influence the learning process.

The resolution of a complex problem can be achieved in successive steps solving simpler problems; this means that the programming activity is an iterative process; so that important evidence must be considered when planning the teaching approach to be followed in Computer Programming courses. This way of solving complex problems should be included in the teaching planning and activities, so that the problems proposed can also be incremented step by step, until reaching a certain complexity.

Because of previous experiences, we think that to practice (writing some programs that solve different problems) is a possible way to learn programming. Given this, students can acquire problem-solving strategies that are crucial to assure their success in learning programming. Therefore, to actually improve the students’ ability to solve problems it is necessary that the students exhibit an active behaviour, instead of a passive one; they must be motivated to do exercises by themselves and develop their own programs. Unfortunately, it is well known that in most cases, when teachers assign students a problem to be solved, they are not capable of starting the task. In other cases, the student breaks the initial inertia, but rapidly becomes discouraged and gives up when she/he faces the first trouble (Proulx, 2000).
3.1 Techniques to support the teaching of programming

Continuing the discussion in the previous section, concerning the essence of teaching/learning ‘Computer Programming’, we divide this section in two parts: one, to present the concept ‘Program Animation’ -- principle, strategies, and supporting tools; the other, to discuss the real impact of ‘Feedback’ on the teaching-learning process -- tools for the ‘Automatic Evaluation of programs’, that validate programs running them and checking the output produced, will be analysed in order to appraise the possibility of their integration in the teaching process.

3.1.1 Animation

The animation of an algorithm is a dynamic visualization of its major abstractions. It is important because it provides the ability to visualize the logic of an algorithm (Pereira, 2002). The animation can be composed of a set of interactive views. Pedagogically spoken, (Korhonen, 2003) states that instead of presenting implementation details of an algorithm, it is more interesting to illustrate the logic and behaviour of it. As such, the students understand better the more complex but important concepts while being stimulated to make progress in their activity (Hundhausen, Douglas & Stasko, 2002).

The use of graphical interfaces has been a concern for several authors, as they use them for better communication between the user and the computer, not just using a textual form (Hansen, Narayanan & Schrimpsher, 1999) (Stasko & Kehoe, 1996) (Hundhausen & Douglas, 2000). The motivation for this approach lies on the potential of the human visual system. In that context there is still a research question regarding how to use that approach to actually help students coping with complex concepts. An animation is a clear way of showing behaviours (Pereira, 2002); so, the animation of an algorithm is a dynamic visualization of its abstractions.

There are several tools aimed at assisting students on learning programming; these tools introduce basic algorithmic concepts through a pleasant environment (Tavares, Henriques & Gomes, 2015). The use of animation improves the quality of the class materials and the learning process becomes simpler since the presentation of abstract concepts is more didactic. It is clear that if the student acquires a good knowledge basis, her/his performance increases producing better results (Santos & Costa, 2006).

Figure 2 illustrates the idea above, showing a screenshot of the animation of a Java program produced by one well-known tool in the area, Jeliot System (Silva et al., 2009).

The screenshot presented in Figure 2 exhibits one of the steps in the animation process of a program that checks if a given number ‘n’ is even. The screen that can be seen in that picture is a visualization of the execution of a ‘if’ instruction included in the program to verify if the given number ‘n’ is multiple of 2; the sequence of images, like the one shown, corresponding to the execution of each program statement, produces the desired animation of the program under study.

3.1.2 Automatic Evaluation

In the learning process, it is important that students practice and solve programming exercises by themselves. However, this approach requires the review, by the teacher, of each solution developed by students. Prompt feedback is very important for the improvement of knowledge and student’s motivation. In large classes and with few lecture hours, individual feedback may consume too much teacher’s time, and this approach could become inadequate (Queirós & Leal, 2015).

To support automatic evaluation for programming problems, online submission systems have been developed (Queirós & Leal, 2012). These systems also help students to improve the performance of programming (Verdú et al., 2011). As such, it shall be provided systems for solving problems with an automatic evaluation able to return immediate feedback. Examples of such systems are Boss (Heng et al., 2005), Mooshak (Leal & Silva, 2008) and EduJudge (Verdú et al., 2011). These tools can be used in teaching activities not only by their online submission purpose but also because allow students to incorporate tests in their work (Tavares, Henriques & Gomes, 2015). They compare the output produced by the submitted answers (programs) against the expected output (repeating the verification for a set of input/output test cases) and produce a grading, but, at the same time, these systems help the teacher to involve students through precise and rapid feedback increasing the motivation of students. Figure 3 shows a Mooshak screenshot illustrating the simplicity and ease of its interface. In addition to the feedback of exercises, shown in the central window, it is possible to see, on the top window, the different options offered to the students (exercise selection, submission, visualization of the results, etc.).
To assess the impact of combining animation and automatic evaluation tools we first conduct an experiment using the currently available computing resources Jeliot and Mooshak described above. Afterwards, we propose a system, PEP, that supports the teacher in lab classes and the student’s study outside the classroom. The overall aim is to improve the students’ engagement and motivation. PEP design follows two principles: (i) first provide means for an easier understanding of given programs, and then...
provide support to write new programs; (ii) to instigate students to practice regularly programming, providing immediate feedback. To be successful in that approach it is required that the teacher chooses an easy to use powerful Animation tool, as well as a friendly Automatic Evaluation System (AES) that returns a complete feedback of the errors found the quality of the code.

3.2.1 Exploring existing resources

This section is devoted to the presentation of the experiment conducted to obtain the user perception regarding our proposal. There are many details concerning its design, implementation and analysis that are fully described in (Tavares, Henriques & Gomes, 2016). To avoid repetitive statements, we just present a brief description in the rest of this section.

As mentioned in the paper referred above, Jeliot and Mooshak were the platforms used for our classroom experiments. This experiment is described in four steps.

Step 1: For each topic, the teacher prepared three similar exercises.

Step 2: For the first exercise, the teacher analysed with the students the problem statement, and then asked them to solve the problem and test the solution produced using the Automatic Evaluation System (AES) selected. Optionally, the teacher also discussed with each student the feedback received.

Step 3: Finally, the teacher provided her/his solution for the exercise and the student uses the AES to animate the execution of the given program in order to carefully analyse and understand the correct solution and its behaviour.

Step 4: Steps 2 and 3 were repeated for the remaining exercises.

To design the experiment so far introduced, the first step was to choose the teaching topic (included in the syllabus of a Computer Programming Course) to be worked out in the experimental lesson as well as to prepare the set of exercises to be solved. For that purpose, it was necessary to define a rigorous plan for the lesson to guide the students step by step. The plan will help them to understand what they are asked to do and how they should proceed.

Twenty-eight 1st year students were involved in these experiments (all of them attend Engineering degrees in various areas other than Computer Science). Two classrooms were prepared to split the students into two parts, one teacher in each room. Teachers remained in the classroom all time, helping students, observing carefully how the session evolves, and to record the entire experiment. The class duration (2 hours) was controlled so that all tasks could be performed during the session. The total time for each exercise was rigorously controlled: resolution and automatic evaluation with Mooshak; visualization/animation of the correct solution with Jeliot.

The class started by preparing students for the session during the first half hour. For the rest of the class (90 minutes), students have half an hour for each exercise, fifteen minutes to develop and test the solution and fifteen minutes to animate the correct solution.

The teachers who were in the classroom were able to observe that both sessions went smoothly, as the first objective; everything ran as planned, all tasks were accomplished, and no incidents were registered. In our experiment, we could observe that students completed the third exercise, which was not easier than the others, faster (smaller number of submissions per student) and with better performance (total number of accepted submissions).

A simple survey, with three questions, was made at the end of the experiment; all the students were asked to fulfill a questionnaire. In most of the answers given, students said that their biggest problem was in coding. However, 30% of students had difficulty in understanding the statements. In programming, the interpretation of statements is a problem, and Mooshak can help students with this task. To understand quickly where an exercise is right or wrong is very important for students. They also mentioned the importance of displaying the error type and the opportunity to correct it and resubmit the exercise. Finally, most students considered important the animation offered by Jeliot. Students also said that they got a better understanding of the algorithm and it was relevant because it explained well the problem solving in an incremental and interactive way.

This experiment was useful to understand how to improve the performance of future tests. For instance, in the future we intend to introduce the three exercises at the beginning and allow students to choose the time intervals to use in each of them. In this way, they can explore the animation more deeply if they think it is beneficial to sediment knowledge before progressing to a new implementation.

3.2.2 PEP System

To enable the combination of these two techniques, PEP (“Plataforma para o Ensino da Programação”), a web-based system was built to support teachers during their
Approaches to manage and understand student engagement in programming

laboratory classes and, support students on their study sessions at home.

With PEP, the teacher prepares two sets of exercises, similar with one another, for each one of the topics that will be taught and presents them with the support of the Animation technique. This process can be divided in three parts:

1. The analysis of the problem and its resolution is presented to the student: at this time, the teacher uses the exercises in the first set to discuss their statement and resolution. Algorithms that solve the exercises are outlined and shared.

2. The student tries to solve the problem: at this time, the student will solve the exercises of the second set and test them using an automatic evaluation system.

3. Discussion: at this time, the teacher and the students discuss the process and feedbacks given by the evaluator.

PEP platform allows the teacher to create and maintain the exercises grouped by topics and organized by level of difficulty; then she/he can decide the exercises to be used in each session; so, at last, she/he is able to plan the sessions. PEP also allows the student to run one or more sessions of a particular topic; during each session the student can animate the exercises and solve them getting immediate feedback. The teacher will receive information about the performance of each student working session (date and time, sequence of solved exercises, time spent, etc.). As can be seen in Figure 4 (Tavares, Henriques & Gomes, 2017), PEP is composed by two main components: the Back-office (BO) and the Front-office (FO).

BO supports the tasks of database management with the questions. Only teachers access this component to manage exercises (create, edit and delete), plan sessions, and analyse the sessions submitted by the students. BO has two more components: (i) the compiler that reads the formal specification of each session (written in a DSL, created specifically for this purpose) and produces the code necessary to mount the sessions in the FO; (ii) the analysis module that recovers from the database the information related to each session of each student and presents it to the teacher to allow him to assess each student’s learning process.

Figure 5 (Tavares, Henriques & Gomes, 2017) shows the possible interaction flow offered to the students in the interface developed. A simple access control mechanism is implemented, requiring a simple and traditional registration process. After choosing the session, two working options are offered: (part 1) that allows the study of previously implemented programs using animation techniques; (part 2) that allows students to develop their own programs and check them immediately supported by an automatic evaluation system. It is recommended to run part 2 only after part 1 is complete.
During Part 1 (Program Animation), the student must read the problem statement, maybe complemented with a short description and comments about the resolution (the algorithm and data structures needed to solve the challenge); the program that implements the solution (in the present version, Java code) is also provided fully implemented. The student shall read and analyse carefully the code, and then she/he must use the Animator available (at present, Jeliot) to run the program and watch an animation that shows how the program behaves while executing the commands written to find the problem solutions.

In Part 2 (Program Evaluation), the student only sees the description of the problem she/he has to solve. Then she/he shall analyse in depth the problem statement and plan a strategy (an algorithm and data structure) to adequately solve the stated problem. After that she/he shall write the Java code that implements the devised algorithmic resolution. The next step is to resort to the Automatic Assessment Tool (Mooshak system in the present version) to compile, run and verify whether her/his program properly solves the problem, or not.

PEP FO is visually described through screenshots displayed in Figures 6 and 7.

PEP is simple and user-friendly Web application developed aiming at incremental motivation and stimulates self-study activities, to improve the Computer Programming learning process.

After developing a prototype it was necessary to assess its quality in order understand how it satisfies the educational requirements. For that purpose we have searched for software tools evaluation approaches. QEF, Quantitative Evaluation Framework (Escudeiro, 2007) (Escudeiro, Bidarra & Escudeiro, 2010), was the assessment method chosen because it allows for the evaluation of a tool in any phase of its development cycle. As deeply detailed in (Tavares, 2018), more than fifty requirements, organized in three dimensions (ergonomic, pedagogical and technical-functional) also structured in factors, were quantified. Applying the formulae proposed by QEF authors, a global grade of 70% was obtained.
Figure 6: Access to Part1 and Part2 in a session.

Figure 7: Part1-navigation buttons to invoke the Animator.
In order to understand the particular difficulties and motivation of the 1st year students attending the Programming course (APROG), we collected 176 answers from an anonymous survey. Questions focused on student motivation: If she/he considered not attending higher education, if the program was her/his first choice, if she/he knew how to program before, if she/he is repeating APROG, if programming is a difficult task, if she/he prefer to work in group, if she/he is motivated, if she/he thinks exercises are hard to understand, if she/he would like to have feedback of the exercises, if she/he needs external motivation to study, if she/he would like an online training platform, if APROG lab classes are good and if the school is good to study.

FERA tool (Gomes et.al, 2019) was proposed to help the teacher to better understand the students’ motivation and learning process and then to identify the appropriate actions for improvement. This tool intends to support the analysis of students’ surveys applying association rules and predicting student’s behaviour in order to improve learning performance in programming learning. Generating a set of association rules, it allows the teacher to visualize them with graphics and charts. According to (Garcia et.al, 2011), Association Rules (AR) is one of the most used data mining techniques. Data mining has been applied in educational research (Karkhanis & Dumbre, 2015), (Mohamad & Tasir, 2013). In particular, Association Rule Mining (ARM) can be used to analyse the data gathered in a survey, allowing a more in-depth comprehension of the interactions between the identified variables. One problem of ARM is the large number of rules produced. This tool helps to identify the most interesting rules generated, visualize them using graphics and it also allows the generation of reports by user demand.

The tool uses the apriori algorithm from arules package, with the parameters selected by the user (support, confidence, and maximum length). Support is an indication of how frequently the itemset appears in the dataset; Confidence is an indication of how often the rule has been found to be true and Maximum length is the maximum number of conditions in the rules. All collected data were analysed using FERA tool to obtain association rules and visualize with more detail the more interesting set of rules (Figure 8).

In Figure 9, we can see a confidence/support plot for a chosen subset of the discovered rules for a group of students. The box shows the detail of a rule. The colour scale represents Lift, the ratio of the observed support to that expected if two itemsets were independent.

Each group of rules can also be visualized as an influence graph between the involved variables. The circles represent rules (diameter is support and colour is lift) while the items are represented as labels. In Figure 10, we can see the result when we choose the variable ‘I need external motivation to study’.

Figure 8: Screen shot of the application FERA for browsing Association Rules.
This application also allows the generation of an HTML report with all the information (Association Rules extracted, a plot a graph of the association rules for the selected parameters).

These visualizations enable a deeper understanding of all aspects of a rule set extracted (and their post-processing) and, in particular, to understand the students’ motivation. For example, one of the rules extracted reveals...
that the lack of motivation throughout the semester seems to be related to the difficulty of the subject and the need of external incentives, with 0.92 of confidence (Tavares, Henriques & Gomes, 2018). With this knowledge, teachers can better understand what motivates (or not) students and act more efficiently. They can also design new surveys and use FERA to support the formulation of new hypothesis iteratively.

4 Conclusion

The teaching/learning process in Computer Programming introductory courses strongly demands new pedagogical approaches, and in this article we have presented our proposal to improve that process. Concerning that process, a combination of two existing methods – the animation of programs, and their automatic evaluation – was discussed. PEP (acronym for Platform for Teaching Programming), a tool designed and implemented by us, was described because we propose its use to support teachers in classrooms or students during homework to increment motivation to learn programming more effectively. The evaluation of PEP prototype carried on following QEF approach, as referred in section 5, encouraged us to plan the development of a final version in the near future.

Finally, we presented FERA, a tool that assists in the analysis of student surveys. We have used surveys from programming students in engineering schools to evaluate the tool capacity to extract interesting rules. FERA was useful to understand the relationship between the student’s behaviour and their motivation.

As future work, we intend be do other experiments and more students’ surveys to collect richer data. FERA will be used to analyse the collected data. The tool will be refined according to the feedback received. We also intend to apply other data mining approaches to assess the relationship between identified variables and student’s engagement in programming.

References

ACM/IEEE, 2013. Computer Science Curricula (2013). Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Final Report.

Almeida, D. (2012). A motivação do Aluno no Ensino Superior: um estudo exploratório. Dissertação apresentada ao programa de Mestrado em Educação da Universidade Estadual de Londrina.

A. Gomes, W. Ke, C. Lam, M. J. Marcelino and A. Mendes, “Student motivation towards learning to program,” 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 2018, pp. 1-8, doi: 10.1109/FIE.2018.8659134.

Callahan, M. (2010). How Do I Motivate My Students?. Teaching Resources, Texas Tech University.

Escudeiro, P. (2007). Avaliação da Qualidade em conteúdos Digitais. Escudeiro, P., Bidarra, J. and Escudeiro, N. (2010). Evaluating Educational Software.” In Systemics, Cybernetics and Informatics, Volume 8, n. 2.

Garcia, E., Romero,C., Ventura, S., Castro, C. (2011). A collaborative educational association rule mining tool. In The Int. and Higher Education, 14, Issue 2, 77-88, ISSN 1096-7516.

Gomes, A. (2010). Difficulties of learning computer programming. Contributions to the understanding and resolution, Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução. Dissertação submetida à Universidade de Coimbra para obtenção do grau de “Doutor em Engenharia Informática”.

Gomes, E., Vieira,D., Tavares, P., Henriques, P. (2019). Association Rules-based Tool for Educational Data mining (FERA). In Conference: Academic Success in Higher Education, CACHE 2019, Porto, Portugal.

Hansen, S., Narayanan, N., Schrimpscher, D. (1999). Helping Learners Visualize and Comprehend Algorithms. Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA’99).

Heng, P., Joy, M., Boyatt, R., Griffiths, N. (2005). Evaluation of the BOSS Online Submission and Assessment System.

Hundhausen, C., Douglas, S. (2000). Using Visualizations to Learn Algorithms: Should Students Construct Their Own, or View an Expert’s?. Proceedings 2000 IEEE International Symposium on Visual Languages IEEE Computer Society Press, Los Alamitos.

Hundhausen, C., Douglas, S., Stasko, J. (2002). A Meta-Study of Algorithm Visualization Effectiveness. Journal of VisualLanguages and Computing 13, 259-290.

ICPCEa, (2020). First International Computer Programming Education Conference Proceedings, June 25-26, 2020, ESMAD, Vila do Conde, Portugal. Queirós, R., Portela, F., Pinto, M., Simões, A. (Ed.). Schloss Dagstuhl - Leibniz-Zentrum fur Informatik (Publisher), OASIs. Volume. 81, ISBN: 978-3-95977-153-5.

ICPCEa, (2021). Second International Computer Programming Education Conference Proceedings, May 27-28, 2021, Universidade do Minho, Braga, Portugal. Henriques, P., Portela, F., Queirós, R., Simões, A. (Ed.). Schloss Dagstuhl - Leibniz-Zentrum fur Informatik (Publisher), OASIs. Volume. 91, ISBN: 978-3-95977-194-8.

Karkhanis, S. P., Dumbre, S.S. (2015). A Study of Application of Data Mining and Analytics in Education Domain. Int. Journal of Computer Applications, Volume.120, No.22, 23-28.

Korhonen, A. (2003). Visual Algorithm Simulation. Dissertation for the degree of Doctor of Science in Technology. At Helsinki University of Technology (Espoo, Finland), November 2003.

Leal, J., Silva, F. (2008). Using Mooshak as a Competitive Learning Tool.

Mohamad, S. K., Tasir, Z. (2013) “Educational Data Mining: A Review”, In Procedia - Social and Behavioral Sciences, Volume.97, 320-324, ISSN1877-0428.

Mohanarajah, S. (2018). Increasing intrinsic motivation of programming students: Towards fix and play educational
Approaches to manage and understand student engagement in programming

Ochmann, K., Podpora, M., Kawala-Janik, A. (2015). Increasing the motivation of learning in HMI workshops with the use of simple embedded platforms. In Proceedings of Quaesti 2015 Conference. EDIS-Publishing Institution of the University of Zilina, Slovak Republic, 3(), 257-260. DOI: 10.18638/quaesti.2015.3.1.246.

Pereira, M. (2002). Systematization of Programs Animation, Sistematização da Animação de Programas. Dissertação submetida à Universidade do Minho para obtenção do grau de doutor em Informática, ramo Tecnologia da Programação.

Pintrich, P. R. (2003) A motivational science perspective on the role of student motivation in learning and teaching contexts. Journal of Educational Psychology, 95(), 667-686.

Proulx, V. (2000). Programming patterns and design patterns in the introductory computer science course. Proceedings of the thirty-first. SIGCSE technical symposium on Computer science education, 80-84. New York.

Queirós, R., Leal, J. (2012). Exercises Evaluation Systems - An Interoperability Survey. In Proceedings of the 4th International Conference on Computer Supported Education (CSEDU), Volume 1, 83-90. Porto.

Queirós, R., Leal, J. (2015). Ensemble: An Innovative Approach to Practice Computer Programming. In R. Queirós (Ed.), Innovative Teaching Strategies and New Learning Paradigms in Computer Programming, Hershey 2015, PA: Information Science.

Reeve, J., Deci, E. L., Ryan, R. M., (2004). Self-determination Theory: A dialectical framework for understanding sociocultural influences on student motivation. In D. M. McInerney & S. Van Etten (Eds.), Big theories revisited 31-60. Greenwich, CT: Information Age.

Robins, A. (2015). Editorial. Computer Science Education, 25(2), 115-119. https://doi.org/10.1080/08993408.2015.1034350

Ryan, R., Deci, E. (2000). Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. American Psychologist, Vol. 55, 68-78.

Santos, R., Costa, H. (2006). Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. INFOCOMP – Journal of Computer Science, Lavras/MG – Brasil, volume 5, n. 1, 41-50.

Schank, D., Meece, J., Pintrich, P. (2014). Motivation in Education: Theory, Research, and Application. 4th edition.

Silva, M., D’Emery, R., Neto, J., Bezerra, Y. (2009). Programming structures: A Experiment with Jeliot. Estruturas de Programação: um Experimento com Jeliot. IX Jornada de Ensino Pesquisa e Extensão (JEPEX) da UFRPE.

Silva, T., Mascarenhas, L., Medeiros, C., Sousa, E. (2014). A Motivação no Ensino Superior: Um Estudo com Alunos dos Cursos de Administração e Direito. Journal of Management Analysis, Volume 3, 104-113. Fortaleza.

Stasko, J., Kehoe, C. (1996). Using Animations to Learn about Algorithms: An Ethnographic Case Study. Technical Report GIT-GVU-96-20, September 1996.

Tavares, P., Henriques, P., Gomes, E. (2016). Computer Techniques to Increase Students Engagement in Programming. 8th International Conference on Computer Supported Education, (CSEDU 2016), Rome, Italy.

Tavares, P., Henriques, P., Gomes, E. (2017). A Computer Platform to Increase Motivation in Programming Students – PEP. 9th International Conference on Computer Supported Education (CSEDU 2017), Porto, Portugal.

Tavares, P., Henriques, P., Gomes, E. (2018). Studying Programming Students Motivation using Association Rule. 10th International Conference on Computer Supported Education (CSEDU 2018). Funchal, Portugal.

Tavares, P. (2018). O Impacto da Animação e da Avaliação Automática na Motivação para o Ensino da Programação. Doctoral dissertation submitted to Universidade do Minho.

Verdú, E., Regueras, L., Verdú, M., Leal, L., Castro, J., Queirós, Q. (2011). A distributed system for learning programming online. Computers & Education 58, 1–10.

Williams, K., Williams, C. (2011). Five Key Ingredients for Improving Student Motivation. Research in Higher Education Journal, 104-122.