Open access • Journal Article • DOI:10.1016/J.COMPEDU.2018.09.021

Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming — Source link

Thanasis Daradoumis, Thanasis Daradoumis, Joan Manuel Marquès Puig, Marta Arguedas ...+1 more authors

Institutions: University of the Aegean, Open University of Catalonia

Published on: 01 Jan 2019 - Computers in Education (Pergamon)

Computers & Education 128 (2019) 159–170

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming

Thanasis Daradoumis[a,b,*], Joan Manuel Marquès Puig[a], Marta Arguedas[a], Laura Calvet Liñan[a]

[a] Department of Computer Science, Multimedia and Telecommunications, Open University of Catalonia, Rambla Poblenou 156, 08018, Barcelona, Spain

[b] Department of Cultural Technology and Communication, University of Aegean, University Hill, 81100, Mytilini, Greece

ARTICLE INFO

Keywords:
Distributed programming assignment
Automatic assessment
Online learning
Students' perceptions

ABSTRACT

Designing an automated assessment tool in online distributed programming can provide students with a meaningful distributed learning environment that improves their academic performance. However, it is a complex and challenging endeavor that, as far as we know, has not been investigated yet. To address this research gap, this work presents a new automated assessment tool in online distributed programming, called DSLab. The tool was evaluated in a real long-term online educational experience by analyzing students' perceptions with the aim of improving its design. A quantitative analysis method was employed to collect and analyze data concerning students' perceptions as to whether using the DSLab tool was really a worthwhile experience. Our study shows that the DSLab tool includes acceptable utility and efficiency features. It also identifies factors that influence current design efficiency with the aim of improving DSLab design by suggesting new functionalities and ideas.

1. Introduction

It is widely accepted that programming and algorithms are considered difficult domains for students, and teachers usually face serious difficulties associated with teaching these subjects; online teaching of programming in particular constitutes one of the major challenges in teaching computer science courses (Jenkins, 2002; Lahtinen, Ala-Mutka, & Järvinen, 2005; Watson & Li, 2014). These difficulties increase when students have to work with complex algorithms such as those used in distributed programming (DP).

Although several automated assessment tools have been developed over the years to assess programming assignments, this still remains a labor-intensive task (Pettit, Homer, Holcomb, Simone, & Mengel, 2015). This is especially evident in distributed programming, where, as far as we know, there is hardly any tool for self-assessing distributed programs in real long-term online educational settings. As a consequence, we developed DSLab, a tool that students can use for self-assessment of the distributed applications they build using a real distributed deployment under realistic conditions.

We argue that the benefits of online students employing this tool are manifold and may include the following: (i) their solutions are assessed in a consistent manner and they obtain immediate feedback, which allows them to know whether their code works correctly; (ii) they strive for a better solution and, therefore, better learning, by submitting their code multiple times and improving it

* Corresponding author. Department of Computer Science, Multimedia and Telecommunications, Open University of Catalonia, Rambla Poblenou 156, 08018, Barcelona, Spain.

E-mail addresses: adaradoumis@uoc.edu (T. Daradoumis), jmarquesp@uoc.edu (J.M. Marquès Puig), martaarg2@gmail.com (M. Arguedas), lcalvetl@uoc.edu (L. Calvet Liñan).

https://doi.org/10.1016/j.compedu.2018.09.021
Received 20 May 2018; Received in revised form 8 September 2018; Accepted 26 September 2018
Available online 27 September 2018
0360-1315/

using the tool's feedback; and (iii) they are provided with a facility which informs them of the grade they would obtain at the end of the execution. Since their assignment consists of different phases, students can decide to stop when they have reached the desired grade (an interesting side benefit for online students who simultaneously study and work, which allows them to manage their work load).

The rest of the paper is organized as follows: Section 2 presents related work on automated programming assessment tools, emphasizing the lack of such tools in distributed programming; then it defines the purpose of our study and research question. Section 3 describes our automated assessment tool. Section 4 presents the implementation and experimentation of our tool. The results of this experimental study are presented in Section 5. In Section 6, we discuss and analyze these results with regard to the research question. Finally, conclusions and future work are presented.

2. Literature review

A large variety of automated programming evaluation systems have been developed for many years. This has already been discussed in an early review of systems for automatic assessment of programming assignments (Ihantola, Ahoniemi, Karavirta, & Seppälä, 2010). However, most of them had a limited lifespan due to the short-term or limited nature of the overall system. Later, the need to effectively assess the programming skills of MOOC students has shifted interest to more practical, intuitive and automated online assessment of programming (Staubitz, Klement, Renz, Teusner, & Meinel, 2015).

A typical example of a programming assessment system is the open-source tool Git, which students use to submit and check their codes continuously until the code meets the assignment requirements (Kelleher, 2014; Lawrance, Jung, & Wiseman, 2013; Loeliger & McCullough, 2012). However, Git is not a fully automated tool, since there are several tasks that teachers and students have to perform manually. An extension of Git has been provided by Chen, Chen, Hsueh, Lee, and Li (2017), which offers more automation, but their tool has not been tested in real long-term educational settings.

Recent assessment tools utilize new mechanisms for analyzing and assessing programming assignments, examining the type and nature of student errors and providing meaningful feedback. Such mechanisms are: Edit distance (Barker-Plummer, Dale, & Cox, 2012), control flow graph (CFG) similarity measurement (Vujošević-Janičić, Nikolić, Tošić, & Kuncak, 2013), or Euclidian and Levenshtein word distance (Stajduhar & Mausa, 2015). However, all these systems fail to detect ‘informal’ errors, and the most important flaw is the lack of a systematic way to cope effectively with the whole formative assessment process that makes the most sense to both teachers and students. Our work presents an initial effort to fill this gap by analyzing all those factors and features that can contribute toward a systematic development of an efficient and useful assessment tool in distributed programming.

Virtual programming lab systems and other programming assessment plug-ins have also been implemented over Course Management Systems (CMS), like Moodle, such as: The Virtual Programming Lab (VPL) developed by Rodríguez-del-Pino, Rubio-Royo, and Hernández-Figueroa (2012), a tool for automated evaluation of VHDL and Matlab programming assignments (Ramos, Trenas, Gutiérrez, & Romero, 2013), and an online compiler and plagiarism detection tool for assessing programming assignments (Kaya & Özel, 2014). However, these tools are still far from providing an environment for a systematic assessment of programming.

Automated assessment tools have also been implemented as part of bigger intelligent tutoring systems (ITS). As such, ITSs are capable of employing systematic assessment and grading methods (Baker & Rossi, 2013), allowing students to correct their errors in real time (Malmi et al., 2004), and offering them appropriate formative feedback to enhance their knowledge construction (Clark, 2012; Heffernan & Heffernan, 2014). In addition, ITS can update students’ models and make them aware of their misconceptions, knowledge gaps and their own learning more accurately and in a timely manner (Grivokostopoulou, Perikos, & Hatzilygeroudis, 2017).

Several online scoring systems have also been developed for different programming environments in recent years, such as automatic scoring systems for C programming language (Fernández-Alemán, 2011; Li et al., 2010). Similar systems were also developed for assessing introductory Python programming problems (Fangohr, O'Brien, Prabhakar, & Kashyap, 2015; Singh, Gulwani, & Solar-Lezama, 2013). Finally, online automated scoring systems were implemented for Java programming assignments (Kitaya & Inoue, 2016; Shamsi & Elnagar, 2012). These systems constitute concrete solutions for particular learning situations conditioned by the specific programming environment used. They have also been tested in controlled studies.

Robinson and Carroll (2017) describe an open-source online learning platform that provides new methods for automated formative and formal summative assessment, allowing both standard compiler/interpreter feedback and customized contextual instructor guidance. The system can also keep track of students' actions and the resulting analysis can provide information about student participation and progress. However, a real appreciation of the system's utility and efficiency is lacking, since an in-depth evaluation of the perceptions of its users is needed.

Hundt, Schlarb, and Schmidt (2017) introduced a web application for the automated assessment of parallel programming assignments. Students can submit any number of solutions until a predefined deadline. For each submission they receive immediate feedback that informs them which tests they have passed. The instructor can also monitor and provide interactive supervision for each individual submission over the whole time span.

Finally, Pettit et al. (2015) conducted an interesting survey into the real usefulness of automated assessment tools (AAT) in programming courses. They discovered that AATs are helpful in improving student learning, supporting instructors, and assuring assessment accuracy. However, they also found that students’ opinions about the helpfulness of AATs are inconclusive.

Indeed, despite the large number of automated assessment tools and published papers, the above survey showed that only a small percentage of the current research work includes formal results about the effects of tool use. Hence, the authors underline the need for more rigorous experiments that can explore the effects of AATs on student learning in more depth. Students' opinions are extremely

important in order to determine the factors and features which are necessary for improving AATs' design and use, especially to decrease negative student perceptions.

2.1. Purpose of the study

Although automated assessment tools for programming assignments have a positive effect on students' academic success, more systematic research is needed to understand how, to what extent, and under what circumstances these tools can influence students' academic performance. This is especially needed in the area of online distributed programming where, to the best of our knowledge, no systematic study exists. To fill this gap, this study examines the utility and efficiency of an automated assessment tool in a distributed programming environment based on the analysis of the perceptions of students in a real, long-term online learning experience. An in-depth quantitative analysis of data coming from students answering a questionnaire is performed. In particular, the study seeks to answer the following research question:

- Research question (RQ) – Did the students perceive that using the DSLab tool was really a worthwhile experience?

3. Proposed tool

DSLab is a web-based tool for assessing distributed applications automatically under realistic conditions in real deployment. The tool has the following characteristics:

- **Automatic assessment**: Students upload the source code to be assessed (different instances of the code are executed interacting with one another). If the final state among the instances is consistent, the assignment is graded as correct; otherwise as not correct.
- **Feedback**: It indicates whether the assignment is correct or not. In case of failure, it reveals which data structures were not consistent. In addition, students can access the final state of all instances as well as the execution logs created by each instance while executing. More details about execution logs are presented at the end of this section.
- **Interaction of student's implementation with instructor's correct implementation**: Half of the computers participating in the assessment of the assignment run the students' implementation, while the other half run the instructors' implementation. This gives us more certainty about the correctness of the students' implementation. In the current version of the tool, an assignment is assessed by running the student's code as well as the teacher's code, interacting with each other. A student's implementation code is correct if its final state is equal to the final state of the teacher's implementation of the code. Therefore:
  - o Teachers upload a solution to the assignment.
  - o A correct student solution to the assignment, when running in a distributed environment, provides the same result as the teacher's solution. The teacher's output can be either a predefined value or it can be calculated while running the student's solution. This second option (calculate the teacher's solution while running student's code) allows different execution parameters and condition values to be used to evaluate each assignment.
- **Execution in real deployment** (different computers) **under realistic conditions**: We force the failure of some computers following different failure patterns.
- **Unlimited submissions**: The final grade is the highest grade obtained from among all submissions.
- **Ability to run several assignments simultaneously**: DSLab permits several students to submit the assignment simultaneously.

DSLab has the following components (Fig. 1):

- **Web interface**: It allows students to upload the code they have implemented, to compile and build it with the rest of the project as well as submit it to be assessed. Instructors can access the submissions of all users and check each student's best submission.
- **Database**: It stores the code uploaded to DSLab and the result of all submissions.
- **Front-end and LSim**: These are in charge of deploying and running the assignments on a set of computers. The front end receives the specification that describes the characteristics of the assignment to be executed (number of instances of each piece of code to be run, initialization parameters, etc.) and runs it in LSim. LSim is a tool we developed to test students' assignments on a set of distributed computers (Fig. 1, right side in blue). It has two parts: (a) *LSim framework*, which automatically deploys and runs the application on the computers; (b) *LSim Library*, which automates the initialization, coordination, collection of results and collection of log messages. More details about the architecture and functionalities of the proposed tool are presented in the tool website[1].

3.1. Logging functionality

As mentioned above, when a submission finishes, the student can check if it was correct or not. If it was not correct, the student is informed which data structure was incorrect as well as about the final state of each instance involved in the assessment of the assignment. Most often, this information is not enough to identify the implementation errors (when the assignment failed), or to

[1] http://dpcs.uoc.edu/projects/dslab.

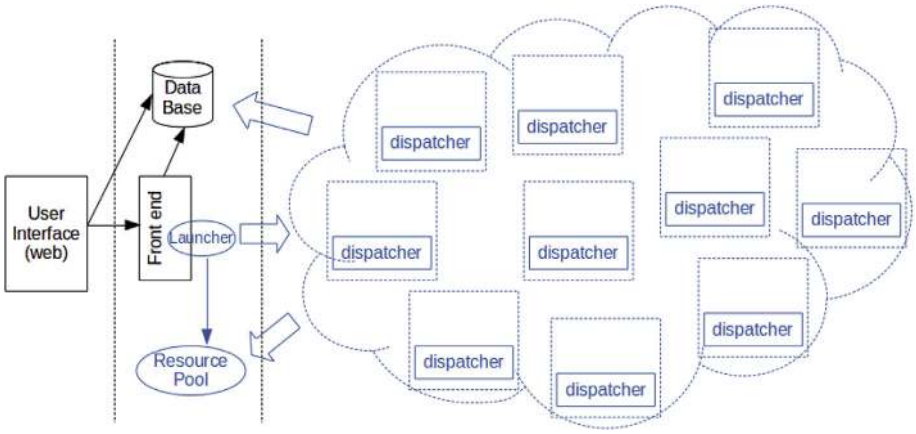


Fig. 1. Architecture of DSLab tool.

improve the efficiency of the code (if it passed). Therefore, we included a logging functionality that allows students to record the events (action, state, value of variable, communication session with another instance, etc.) they think could help them understand the execution of the assignment. Students can access the logs generated by each instance.

Similarly to *log4j*, a popular logging utility from Apache Software Foundation, we use six levels of log (in decreasing order of severity): FATAL, ERROR, WARN, INFOR, DEBUG and TRACE. This allows students to categorize each trace they want to collect and, once the execution is finished, access the level of detail they consider will help them understand how the execution evolved during the assessment of the assignment.

4. Implementation and experimentation

4.1. Case study description

Our experiment was performed in an undergraduate Distributed Systems online course at our university. Students had to implement a distributed algorithm, upload and execute their code to DSLab and assess its correctness. The multiple submissions capability of DSLab – which allows students to improve their code using the feedback from previous executions – supports iterative learning (Suleman, 2008).

The assignment consisted of several implementation phases that were assessed using DSLab (phases: 2, 3 and 4.1). Each phase is an extension of the previous one, has an increasing level of difficulty and is assessed separately, as it uses different configuration and execution conditions.

4.2. Participants and procedure

The course was carried out fully online for seven weeks. A total of 110 adult students participated, divided into three classes. All students performed the same assignment. Among the students, 13 were female (11.81%) and 97 were male (88.18%).

The assignment could be implemented individually or in pairs (formed by students themselves), applying the same task and assessment criteria. Students could decide which option they preferred. From the 110 students that participated in the experience, 65.85% decided to do the assignment individually and 34.15% in pairs.

In each assignment phase, students could submit their code as many times as they wished. For each submission they received DSLab assessment and feedback, and thus could look at the execution logs to check for possible errors.

At the end of the whole process, students answered a customized questionnaire that was specifically designed to respond to our RQ and allowed us to collect quantitative data for our analysis.

4.3. Data collection

The questionnaire was finally answered by 54 students, that is, about 49% of the participants. The questionnaire aimed to examine *students' perceptions of the utility of the tool* (RQ) and included 19 items divided into four categories (as shown in Table 1): five items related to tool use (PU, standing for Perception – Use), seven items related to tool feedback (PF, standing for Perception – Feedback), and seven items related to tool efficiency and objectivity (PEO, standing for Perception – Efficiency-Objectivity). We used a five-point Likert-type scale ranging from 1 (Strongly disagree) to 5 (Strongly agree), requiring a quantitative answer.

The statistical techniques employed in the analysis were descriptive statistics, calculating relative frequencies (%), as well as bivariate correlation and analysis of variance to find relationships between the variables under study to answer our research question. The presentation of statistical results is shown in the Results section below.

Table 1
Questionnaire of nineteen (19) question items related to students' perceptions of the utility of the tool (RQ).

Students' Perceptions of the utility of the tool	
Use of the DSLab automated assessment tool	
PU1	There are more than two things that you liked about the tool. What were they? 1.2.
PU2	The more you used the tool, the more you liked it.
PU3	Being able to make several submissions encouraged you to work to obtain a higher grade.
PU4	Do you consider it counterproductive to be able to make several submissions, allowing you to obtain a grade that you consider satisfactory, and therefore stop working on the rest of the exercises (to dedicate yourself to other things)?
PU5	Under the same conditions, if you had the option to choose between doing the assignment with or without DSLab tool, would you choose to use the tool?
***F**eedback provided by the DSLab tool*	
PF1	Have you found the logs and the execution results provided by the tool useful to help you determine why your program was working incorrectly?
PF2	Has it been useful for you to have the DSLab tool inform you so quickly of whether the submission of each phase of the assignment was correct or not?
PF3	Do you think the tool's logs are accurate?
PF4	Do you think the tool's logs were adaptable to your knowledge level?
PF5	Do you think the logs provided by the tool have conditioned your implementation? (That is, at some point you modified a part of your source code, even though it was not wrong, just because it was better suited to the feedback logs generated by the tool.)
PF6	The logs and the result associated with each execution have helped you know how your code should have behaved.
PF7	Do you think that if the tool had a more extensive and detailed level of logs it would be much more useful for helping to improve your code?
***E**fficiency-**O**bjectivity*	
PEO1	Do the self-assessed assignments meet your expectations about the course requirements?
PEO2	The use of DSLab tool has allowed you to obtain a more robust solution than if you had worked without using the tool.
PEO3	Using the tool has reduced your workload.
PEO4	Using the tool has saved you time.
PEO5	Do you think that the use of the DSLab tool makes the notion of correctness for an assignment too strict?
PEO6	Do you think the tool should be used in more assignments in the future?
PEO7	All in all, do you think that a system that allows you to test different solutions and provides quick feedback on your performance of the task is valuable to you?

5. Results

5.1. Reliability statistics and multivariate normality

To ensure the reliability of data collection, we applied Cronbach's alpha coefficient to the student data. We obtained a Cronbach's alpha of .85, which is higher than 0.70, thereby reinforcing the reliability of our items.

In addition, the skewness and kurtosis of each variable were examined to check for multivariate normality. To achieve this, the absolute values of skewness and of kurtosis must not exceed the allowed maximum both for univariate skewness (2.0) and univariate kurtosis (7.0). Our calculations showed that there was no critical problem regarding multivariate normality, i.e. the data for all the variables are normally distributed.

5.2. The descriptive statistic measures

The results obtained from descriptive statistics are related with our research question:

RQ –Did the students perceive that using the DSLab tool was really a worthwhile experience?

Regarding the descriptive statistic measures we performed in this analysis, on the one hand we consider the *median* (or weighted average) since it is less sensitive than the mean to oscillations of the values of the variable, while not being affected by dispersion. On the other hand, we consider *mode* which, as it relies only on frequencies, gives us the most common values in a data set. This is very useful in our interpretations, in which we want to measure both knowledge improvement after training and the most common score among participants.

5.2.1. The results with regard to the RQ

We analyzed students' perceptions of the utility of the tool in relation to three parameters: (1) the students' experience of using DSLab (PU2-PU5), (2) the feedback provided by the tool (PF1-PF7), and (3) the efficiency-objectivity of the tool (PEO1- PEO7).

(1) Regarding the use of the tool, the analysis is based on the descriptive statistic measures of items PU2-PU5 in Table 2 and Fig. 2.

More specifically, almost 54% of the students clearly expressed that *the more they used the tool, the more they liked it* (PU2, with mode = 4 and median = 4). However, a number of students (around 30%) did not agree with this statement.

Table 2
Frequency data concerning *Students' **P**erceptions* based on DSLab ***U**se* (**PU**).

Frequency data concerning *Students' **P**erceptions* based on DSLab ***U**se* (**PU**)

	PU2		PU3		PU4		PU5	
	Fr.	%	Fr.	%	Fr.	%	Fr.	%
1	6	11.1	4	7.4	30	55.6	2	3.7
2	10	18.5	3	5.6	10	18.5	4	7.4
3	9	16.7	6	11.1	5	9.3	6	11.1
4	27	50.0	20	37.0	5	9.3	16	29.6
5	2	3.7	21	38.9	4	7.4	26	48.1

	PU2	PU3	PU4	PU5
Min	1	1	1	1
Max	5	5	5	5
Mean	3.17	3.94	1.94	4.11
Median	4.00	4.00	1.00	4.00
Mode	4	5	1	5

Fig. 2. The descriptive statistic measures concerning *Students' **P**erceptions* based on DSLab ***U**se*.

Certainly this matter raises the need for a more extensive analysis taking into account the students' background in programming and transcript of academic records, since this is one of the last courses in the students' academic career. In any case, we did manage to obtain students' comments during the semester that we used DSLab.

From the students' comments, we deduced that those with good programming skills found DSLab useful since they could assess their implementation code and know what grade they would obtain at any time. In addition, the tool allowed them to test their implementation code against the instructor's implementation. In this way, they could identify and resolve any minor implementation errors they may have and, therefore, strive to obtain a grade according to their knowledge, skills and available time.

Among students with poorer programming skills, we identified two main attitudes:

(a) Students who appreciated that the tool tested their implementation exhaustively; this allowed them to identify and resolve problems that they initially had not considered and, consequently, achieve better learning.
(b) Students who did not like that their implementation was executed against the instructor's solution; in that case, the tool detected implementation errors that students initially had not taken into account when they were testing it on their own local computer (i.e. these errors did not appear because they did not test that specific case). In most of these cases, students were unhappy to discover and resolve these errors, since it took them considerable time and effort.

As concerns the latter, we should note that if a teacher had graded the students' code, they would have received the same results; however, this process should have been much slower. In fact, the teachers tried to explain this situation to these students so as to reduce their anxiety and make them realize that the extra time the tool required them to spend should lead them to achieve a more functional code.

The students' perception of the usefulness of the tool greatly increased in items PU3 and PU5 (mode = 5 and median = 4). In both items, more than 75% of the students manifested a clear preference for *choosing the option to use the tool instead of working without it* (PU5) as well as for the *tool feature that enabled them to make several submissions which encouraged them to work to obtain a higher grade* (PU3). The latter is further confirmed by almost 75% of the students who stated that *they did not consider it counterproductive to be able to make several submissions, allowing them to obtain a grade they considered satisfactory, and thus they had the choice to stop working on the rest of the exercises and pay more attention to other things* (PU4, with mode = 1 and median = 1).

In conclusion, more than 75% of the students chose to use the tool to debug their codes during the activity, motivated by the ease of use and the possibility of obtaining better results and grades, without feeling apologetic for not completing the rest of the exercises. This fact is also depicted in the qualitative answers they provided in item PU1 which asked them to *express two things that they liked most about the tool.* Some representative opinions are the following: "It's wonderful to see that it works against a real distributed system". "After one or two uses you get it, it's really simple to use and effective". "It's easy to use, since it has not been necessary to

Table 3
Frequency data concerning learning *Students*' ***Perceptions*** based on ***Feedback*** (**PF**).

Frequency data concerning learning *Students' **Perceptions*** based on ***Feedback*** (**PF**)

	PF1		PF2		PF3		PF4		PF5		PF6		PF7	
	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%
1	5	9.3	2	3.7	6	11.1	9	16.7	9	16.7	3	5.6	2	3.7
2	8	14.8	2	3.7	3	5.6	5	9.3	12	22.2	10	18.5	5	9.3
3	15	27.8	5	9.3	13	24.1	17	31.5	15	27.8	13	24.1	13	24.1
4	21	38.9	15	27.8	29	53.7	17	31.5	13	24.1	21	38.9	21	38.9
5	5	9.3	30	55.6	3	5.6	6	11.1	5	9.3	7	13.0	13	24.1

employ either a manual or any other help". "I like it because I can upload different projects as tests; furthermore, it not only lets you see whether the result is correct or not, but it also lets you see the different messages it facilitates as well as the automatic code correction".

(2) Regarding the feedback provided by DSLab, the results are presented in Table 3 and Fig. 3.

These results show that more than 50% of the students agreed that *the logs and the result associated with each execution helped them know how their code should have behaved* (PF6, mode = 4 and median = 4). However about 25% did not share this opinion. Moreover, less than 50% of the students convincingly stated that *the logs and the execution results provided by the tool were useful enough to help them determine why their program was working incorrectly* (PF1, with mode = 4 and median = 3). In fact, several students indicated that they would have preferred a more specific automated code correction facility. It is also worth mentioning that almost 25% of students expressed their disagreement with the usefulness of the tool logs and execution results for helping them debug their code.

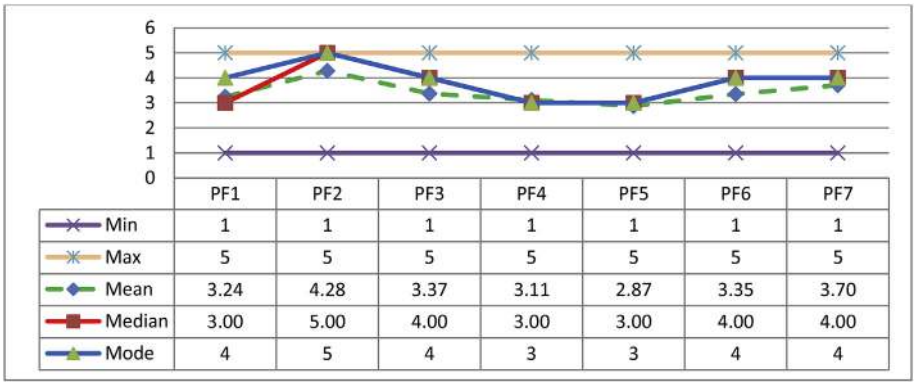
The above result is in line with the fact that, on the one hand, the vast majority of students (more than 83%) agreed that *the response of the tool was always quick and informative of the detected faults* (PF2, mode = 5 and median = 5). On the other hand, a fairly large number of students (63%) also agreed that *if the tool had a more extensive and detailed level of logs it would be a much more useful aid to improve their code* (PF7, mode = 4 and median = 4).

Furthermore, nearly 60% of the students affirm that the tool's logs provided them with precise information regarding the execution of their codes (PF3, mode = 4 and median = 4). However, just 42% of students firmly believed that *the tool's logs were adaptable to their knowledge level* (PF4, mode = 3 and median = 3); more than 25% claimed the contrary.

Finally, students' opinion was somewhat divided when they were asked whether *the logs provided by the tool had conditioned their implementation* (PF5, mode = 3 and median = 3). Indeed, 39% of students – compared with 33% – perceived that the tool was not influencing them to adapt their code to the feedback hints when they had submitted an acceptable code. Consequently, at this point it is uncertain whether the tool feedback influences students to follow a standard coding pattern, which ultimately hinders their own creativity.

In conclusion, the analysis of these results shows that although the tool feedback was useful and helpful in general, it presents some problems and has some limitations that need to be improved:

a) The logs should provide more extensive information that is more related to the detected errors and not to the different steps of the code execution.
b) The logs should provide students with more specific information about how these errors make their code behave erroneously and then suggest possible solutions to correct them.
c) The wording and terminology used in the logs should be better adapted to the students' knowledge level, since they are currently difficult to understand.



	PF1	PF2	PF3	PF4	PF5	PF6	PF7
Min	1	1	1	1	1	1	1
Max	5	5	5	5	5	5	5
Mean	3.24	4.28	3.37	3.11	2.87	3.35	3.70
Median	3.00	5.00	4.00	3.00	3.00	4.00	4.00
Mode	4	5	4	3	3	4	4

Fig. 3. The descriptive statistic measures concerning learning *Students*' ***Perceptions*** based on ***Feedback***.

Table 4
Frequency data concerning *Students' **P**erceptions* on ***E**fficiency-**O**bjectivity* (**PEO**).

Frequency data concerning *Students' **P**erceptions* on ***E**fficiency-**O**bjectivity* (**PEO**)

	PEO1		PEO2		PEO3		PEO4		PEO5		PEO6		PEO7	
	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%	Fr.	%
1	1	1.9	3	5.6	10	18.6	7	13.0	8	14.8	5	9.3		
2	3	5.6	3	5.6	11	20.4	14	25.9	17	31.5	2	3.7	2	3.7
3	8	14.8	8	14.8	17	31.5	12	22.2	15	27.8	7	13.0	7	13.0
4	25	46.3	22	40.7	13	24.1	16	29.6	11	20.4	16	29.6	20	37.0
5	17	31.5	18	33.3	3	5.6	5	9.3	3	5.6	24	44.4	25	46.3

(3) Regarding students' perceptions of the efficiency and objectivity of the tool (PEO1- PEO7), the results are presented in Table 4 and Fig. 4.

These results show that the great majority of students (nearly 78%) agreed that *the self-assessed assignments met their expectations about the requirements of the course* (PEO1, mode = 4 and median = 4), since they were able to check whether the assignment worked in real time. Moreover, 74% of students confirmed that *the use of the DSLab tool allowed them to produce a more robust solution than if they had done so without using the tool* (PEO2, mode = 4 and median = 4).

However, only around 30% expressed with certainty that *using the tool reduced their workload* (PEO3, mode = 3 and median = 3). Just over 30% were neutral in this aspect, whereas 39% of students did not notice any workload reduction from using the tool. Consequently, the same number of students (39%) ascertained that *they did not save time by using the tool* (PEO4, mode = 4 and median = 3), against an equal number of students (39%) who mentioned important time savings, principally because of the tool's ease of use.

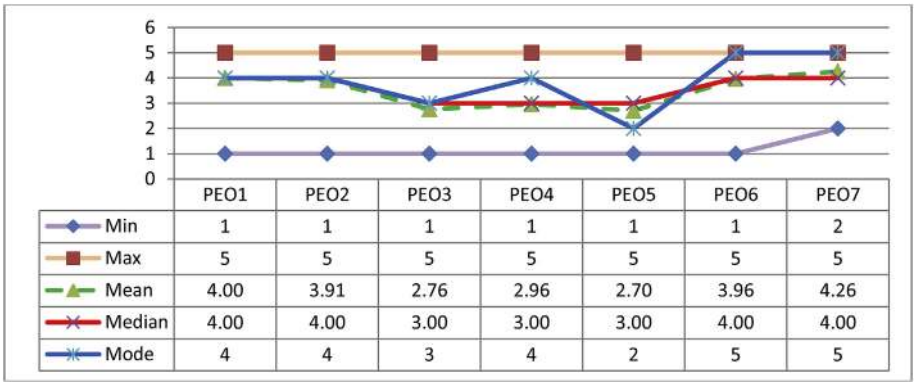
Furthermore, just 26% of the students agreed that *the use of the DSLab tool makes the evaluation of whether the assignment is correct or not too strict* (PEO5, mode = 2 and median = 3). Instead, more than 46% of students perceived that *the use of the DSLab tool did not make the notion of correctness for an assignment too strict*, whereas the rest of the students (around 28%) were neutral. As such, 74% of students – compared with 14% – think that *the tool should be used in more assignments in the future* (PEO6, mode = 5 and median = 4), since it constitutes a good simulation artifact for a real distributed environment. This motivates students and gives them a clear idea of their performance level on the course. Finally, more than 83% of students asserted that *DSLab is a system that allows them to test different solutions and provides quick feedback on their performance of the task, a fact that is valuable to them* (*PEO7*, mode = 5 and median = 4). Less than 4% had a negative opinion about it.

In conclusion, students find that the DSLab tool is efficient and objective because:

a) It helps them carry out self-assessment of their assignments until they meet the course requirements, thus obtaining a robust solution.
b) It allows them to test different solutions and provides quick feedback on their task performance.
c) It was rather flexible when assessing the correctness of an assignment, by accepting codes that were correct even though they were not following the tool coding pattern exactly.

For these reasons, students positively value the use of the DSLab tool and express their preference to continue using the tool in future assignments.

However, for an important number of students (almost 40%), the use of the tool has not reduced their workload, nor has it saved



	PEO1	PEO2	PEO3	PEO4	PEO5	PEO6	PEO7
Min	1	1	1	1	1	1	2
Max	5	5	5	5	5	5	5
Mean	4.00	3.91	2.76	2.96	2.70	3.96	4.26
Median	4.00	4.00	3.00	3.00	3.00	4.00	4.00
Mode	4	4	3	4	2	5	5

Fig. 4. The descriptive statistic measures concerning *Students' **P**erceptions* on ***E**fficiency- **O**bjectivity*.

them time in the resolution of the assignment.

The latter is related with the conclusions drawn for parameter 2, feedback. For this parameter, students would have preferred more extensive and specific information related to the detected errors and more guidance for error correction, adapted to their knowledge level. This fact would certainly have contributed to reducing students' workload and saving them time in resolving the assignment.

Point c) can be contrasted with item PF5 (Table 3). In this case, the multiple submissions facility of the tool as well as the type of feedback provided have directed several students to follow a standard coding pattern (coding conventions), which may have hindered their own creativity. Despite this, the tool shows some flexibility by accepting student programs that deviate slightly from standard coding conventions, but that still represent correct programs (we call them 'non-standard correct codes').

6. Discussion

In the previous section, the presentation of questionnaire results based on descriptive statistics provided us with insights into the use of the DSLab tool and its effects, but mainly about the *perceptions of students on the utility of the tool* (RQ). Based on these results, in this section we attempt to take a closer look at our research question within the context of the existing literature.

6.1. Analysis for the RQ

With respect to our research question, to provide a more comprehensive response regarding *the students' perceptions about the utility of the tool*, we analyzed the Pearson correlations between the three dependent variables: the **u**se of the tool (PU), the tool **f**eedback (PF), and the **e**fficiency-**o**bjectivity of the tool (PEO) with respect to the independent variable *students' perceptions of the utility of the tool* (P).

Our results are broadly consistent with previous research that has shown students' appreciation of the opportunity to submit and check their codes continuously until they meet the requirements of the assignments (Kelleher, 2014; Lawrance et al., 2013; Loeliger & McCullough, 2012).

In particular, regarding the use of the tool (PU), the results presented in Table 5 show that there is a strong positive relationship between variables PF1, PF2, PF5, PF6 and PU2, PU3, PU5.

This strong positive relationship means that, the more students used the tool (PU2), the more beneficial they found the tool feedback, valuing its fast response and gaining more knowledge. Although the feedback sometimes conditioned their implementation, it helped them know how their code should have behaved. Moreover, the type of feedback received encouraged students to make several submissions and work to obtain a higher grade (PU3), which resulted in them considering the tool an important asset for carrying out their assignments (PU5). Here, it is worth mentioning the negative or low relationship between variables PF3 and PF7 and all PU variables (except PF7 and PU3), which confirms the lack of accuracy of the tool logs and the need for more explicit, precise, detailed and meaningful feedback (Barker-Plummer et al., 2012; Stajduhar & Mausa, 2015). Finally, the adaptability of the tool logs to students' knowledge levels (PF4) worked better if students used the tool more (PU2); however, higher adaptability to student knowledge is necessary in order to allow students to take better advantage of the tool's multiple submission facility and achieve better performance (Heffernan & Heffernan, 2014).

As regards efficiency-objectivity variables (PEO), there is a strong positive relationship between variables PEO1, PEO2, PEO3, PEO4, PEO6 and PU2, PU5. This confirms our previous conclusions that the more students used the tool (PU2), the more they fulfilled

Table 5
The correlations between ***U**se of the DSLab (PU)* and ***F**eedback (PF) - **E**fficiency-**O**bjectivity (PEO)* items.

The correlations between ***U**se of the DSLab (PU)* and ***F**eedback (PF) - **E**fficiency-**O**bjectivity (PEO)* items

	PU2	PU3	PU4	PU5
PF1	0.55**	0.32*	0.04	0.47**
PF2	0.33*	0.30*	-0.14	0.37**
PF3	0.26	0.15	-0.28*	0.22
PF4	0.30*	0.18	-0.26	0.35*
PF5	0.32*	0.38**	0.04	0.36**
PF6	0.47**	0.33*	-0.09	0.37**
PF7	0.01	0.35*	0.12	-0.07
PEO1	0.38**	0.17	-0.02	0.60**
PEO2	0.50**	0.40**	-0.06	0.59**
PEO3	0.32*	0.48**	0.12	0.51**
PEO4	0.47**	0.51**	0.07	0.57**
PEO5	0.08	0.06	0.16	-0.11
PEO6	0.30*	0.38**	-0.19	0.52**
PEO7	0.20	0.40**	-0.28*	0.50**

course requirements (PEO1) and the greater possibility they had of achieving a robust solution (PEO2). It also confirms that students will not only use the tool again to do their assignments (PU5) but they will also use it for other courses and assignments (PEO6). Indeed, DSLab could be used in any course where students have to learn to implement an algorithm. For example, in addition to distributed courses (which is the main focus of this work), it could be also used in an algorithm and data structures course to assess the implementation of different algorithms (e.g. quicksort, binary tree, etc.).

This trend echoes findings from other studies indicating improvement in student programming skills (Chen et al., 2017; Helmick, 2007; Higgins, Gray, Symeonidis, & Tsintsifas, 2005). However, our analysis makes an important contribution to the literature by suggesting that the more students used the tool (PU2), the more they felt it reduced their workload (PEO3) and saved them time (PEO4), especially the latter. This perception increased greatly when students thought of using the tool if they had to do the assignment again (PU5). Moreover, PU3 (*making several submissions encouraged students to work to obtain a higher grade*) has a strong positive relationship with the aim of obtaining: a robust solution (PEO2), workload reduction (PEO3) and time saving (PEO4). These are aspects that students would probably not have achieved if they had worked without the tool or submitted their work just once to the teacher. Moreover, students developed more confidence and appreciation for the tool utility (PEO6 & PEO7). While more research is necessary to confirm and better understand these associations, it may not be too soon to suggest that familiarization and constant activity with the tool constitute factors that positively influence workload reduction and time saving. Also, the negative relationship between PU4 and PEO7 confirms the findings of previous studies (Blumenstein, Green, Fogelman, Nguyen, & Muthukkumarasamy, 2008; Joy, Griffiths, & Boyatt, 2005; Wang, Su, Ma, Wang, & Wang, 2011) that students did not consider it counterproductive to make several submissions until they obtained a satisfactory grade. In our case, students could stop working on the rest of the exercises (to dedicate themselves to other things) when they were satisfied with their grade. Finally, in contrast to Rodríguez-del-Pino et al. (2012), where the students' program output should conform exactly to the specified solution output of their tool, our analysis notes that there are not any significant relationships between PEO5 and PU variables. This reasserts students' perception that our tool shows some flexibility in accepting non-standard correct codes.

Regarding the feedback provided by the tool (PF), the results presented in Table 6 show that there is a strong positive relationship between variables PF1, PF2, PF3, PF4, PF6 and variables PEO1, PEO2, PEO4, PEO6, PEO7, as well as PEO3 and PEO5 to a lesser extent.

In general, the findings of this study on feedback usefulness and its contribution to making students perceive tool efficiency and objectivity are in accordance with results from other studies (Clark, 2012; Hundt et al., 2017; Kordaki, Miatidis, & Kapsampelis, 2008; Robinson & Carroll, 2017). In fact, our results suggest that feedback helped students determine why their program was working incorrectly (PF1), allowed them to know rapidly whether their assignment was correct or not (PF2), showed an acceptable level of accuracy (PF3), was adaptable to their knowledge level (PF4), and helped them know how their code should have behaved (PF6). Consequently, the current feedback type encouraged students to perceive that using the DSLab tool was really a worthwhile experience (RQ), since it generally fulfilled their expectations about the course requirements (PEO1), allowed them to obtain a more robust solution than if they had worked without using the tool (PEO2), saved them time (PEO4), gained their confidence (PEO6 & PEO7), and, to a lesser extent, reduced their workload (PEO3) and permitted some flexibility in accepting non-standard correct codes (PEO5). Considering the survey conducted by Pettit et al. (2015), which found that students' opinions about the helpfulness of automated assessment tools in programming courses are inconclusive, this is reflected in variables PF5 and PF7 of our study. The latter did not show any significant relationship with the efficiency-objectivity variables, which reconfirms one of the biggest limitations of our current feedback type – its lack of more extensive and detailed information about students' errors – which would certainly have been much more useful in helping to improve their code. Furthermore, the only significant positive relationship that exists between variables PF5 and PEO2 implies that, on the one hand, the feedback provided by the tool conditioned students' implementation, which in turn limited their own creativity, and, on the other hand, it allowed students to obtain a more robust solution. Thus, these findings suggest that future research should investigate the format, content and style of tool feedback in order to achieve a better equilibrium between tool efficiency and objectivity.

Table 6
The correlations between ***Feedback from the DSLab (PF)*** and ***Efficiency-Objectivity (PEO)*** items.

The correlations between ***Feedback from the DSLab (PF)*** and ***Efficiency-Objectivity (PEO)*** items

	PF1	PF2	PF3	PF4	PF5	PF6	PF7
PEO1	0.35*	0.39**	0.19	0.38**	0.13	0.42**	-0.23
PEO2	0.60**	0.30*	0.33*	0.48**	0.33*	0.48**	0.03
PEO3	0.42**	0.37**	0.11	0.14	0.12	0.22	0.24
PEO4	0.54**	0.40**	0.14	0.29*	0.23	0.36**	0.23
PEO5	0.13	-0.07	0.41**	-0.00	0.24	0.19	0.04
PEO6	0.50**	0.49**	0.43**	0.43**	0.19	0.40**	-0.04
PEO7	0.26	0.51**	0.42**	0.27	0.13	0.31*	-0.11

7. Conclusion and limitations

This work presented a new automated assessment tool in online distributed programming, called DSLab, and analyzed a variety of parameters from students' points of view that may influence the design of such a tool. Designing an automated assessment tool in online distributed programming that can provide students with a truly worthwhile learning experience and improve their academic performance is a complex and challenging endeavor that, to the best of our knowledge, has not yet been investigated in this area. Our study, which was carried out in real long-term online educational settings, showed that the DSLab tool includes acceptable utility and efficiency features. It has also identified factors that influence and improve DSLab design so that a new version of the tool could meet these criteria in a more effective way and thus enhance students' learning experience.

Limitations of the current work provide challenging opportunities for future research. First, more descriptive and explanatory feedback that is more adaptable to students' knowledge level should be generated; students should be helped in overcoming difficult situations and improving their code; and students' implementation code must not be conditioned to follow a standard coding pattern, thus restricting their own coding creativity. Second, the use of the tool should focus more on issues that are very important for online higher education students, such as workload reduction and time saving. Currently, only around 30% of students expressed with certainty that using the tool reduced their workload, whereas around 40% affirmed important time savings. Our analysis showed that improvement on both issues was evident only when students achieved a high level of familiarization and constant activity with the tool. Moreover, workload reduction and time saving were clearer to students who strived to obtain a more robust solution to meet course requirements and gain a better grade, a fact that would otherwise have been more difficult to attain if they had worked without using the tool. Nevertheless, both aspects remain high-priority issues to improve in subsequent versions of the tool. Third, though some students used the tool collaboratively in pairs, the current tool design lacks specific features that would support effective collaborative work and learning. Finally, at a technical level, the tool needs some improvements to correct log inconsistencies and failures that prevent some files from being generated as expected.

References

Baker, R. S., & Rossi, L. M. (2013). Assessing the disengaged behaviors of learners. In R. Sottilare, A. Graesser, X. Hu, & H. Holden (Vol. Eds.), *Design recommendations for intelligent tutoring systems: Vol. 1*, (pp. 155–166). Orlando, FL: Learner Modeling. U.S. Army Research Lab.

Barker-Plummer, D., Dale, R., & Cox, R. (2012). Using edit distance to analyse errors in a natural language to logic translation corpus. *Proceedings of the 5th international conference on educational data mining* (pp. 19–21). (Chania, Greece).

Blumenstein, M., Green, S., Fogelman, S., Nguyen, A., & Muthukkumarasamy, V. (2008). Performance analysis of GAME: A generic automated marking environment. *Computers & Education, 50*(4), 1203–1216.

Chen, H.-M., Chen, W.-H., Hsueh, N.-L., Lee, C.-C., & Li, C.-H. (2017). ProgEdu - an automatic assessment platform for programming courses. In Meen, Prior, & Lam (Eds.). *Proceeding of the 2017 IEEE international conference on applied system innovation* (pp. 173–176). (Sapporo, Japan).

Clark, I. (2012). Formative assessment: Assessment is for self-regulated learning. *Educational Psychology Review, 24*(2), 205–249.

Fangohr, H., O'Brien, N., Prabhakar, A., & Kashyap, A. (2015). *Teaching Python programming with automatic assessment and feedback provision*Technical Report arxiv:1509.03556.

Fernández-Alemán, J. L. (2011). Automated assessment in a programming tools course. *IEEE Transactions on Education, 54*(4), 576–581.

Grivokostopoulou, F., Perikos, I., & Hatzilygeroudis, I. (2017). An educational system for learning search algorithms and automatically assessing student performance. *International Journal of Artificial Intelligence in Education, 27*(1), 207–240.

Heffernan, N. T., & Heffernan, C. L. (2014). The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education, 24*(4), 470–497.

Helmick, M. T. (2007). Interface-based programming assignments and automatic grading of java programs. *ACM SIGCSE Bulletin, 39*(3), 63–67.

Higgins, C. A., Gray, G., Symeonidis, P., & Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *Journal on Educational Resources in Computing, 5*(3), 5.

Hundt, C., Schlarb, M., & Schmidt, B. (2017). Sauce: A web application for interactive teaching and learning of parallel programming. *Journal of Parallel and Distributed Computing, 105*(2017), 163–173.

Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. *Proceedings of the 10th koli calling international conference on computing education research* (pp. 86–93). Koli, Finland: ACM.

Jenkins, T. (2002). On the difficulty of learning to program. *Proceedings of the 3rd annual conference of the LTSN centre for information and computer sciences: Vol. 4*, (pp. 53–58). United Kingdom: Loughborough University.

Joy, M., Griffiths, N., & Boyatt, R. (2005). The boss online submission and assessment system. *Journal on Educational Resources in Computing, 5*(3), 2.

Kaya, M., & Özel, S. A. (2014). Integrating an online compiler and a plagiarism detection tool into the Moodle distance education system for easy assessment of programming assignments. *Computer Applications in Engineering Education, 23*(3), 363–373.

Kelleher, J. (2014). Employing Git in the classroom. *Computer applications and information systems (WCCAIS), 2014 world congress* (pp. 1–4). (Hammamet, Tunisia).

Kitaya, H., & Inoue, U. (2016). An online automated scoring system for java programming assignments. *Int. J. Inform. Educ. Technol. 6*(4), 275–279.

Kordaki, M., Miatidis, M., & Kapsampelis, G. (2008). A computer environment for beginners' learning of sorting algorithms: Design and pilot evaluation. *Computers & Education, 51*(2), 708–723.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14–18.

Lawrance, J., Jung, S., & Wiseman, C. (2013). Git on the cloud in the classroom. *Proceedings of the 44th ACM technical symposium on computer science education* (pp. 639–644). New York: ACM.

Li, J., Pan, W., Zhang, R., Chen, F., Nie, S., & He, X. (2010). Design and implementation of semantic matching based automatic scoring system for C programming language. *Lecture Notes in Computer Science, 6249*, 247–257.

Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development.* Sebastopol, CA: O'Reilly Media Inc.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education, 3*(2), 267–288.

Pettit, R. S., Homer, J. D., Holcomb, K. M., Simone, N., & Mengel, S. A. (2015). Are automated assessment tools helpful in programming courses? *Proceedings of the 122nd ASEE annual conference & exposition* (pp. 1–20). (Seattle, WA).

Ramos, J., Trenas, M. A., Gutiérrez, E., & Romero, S. (2013). E-assessment of Matlab assignments in Moodle: Application to an introductory programming course for engineers. *Computer Applications in Engineering Education, 21*(4), 728–736.

Robinson, P. E., & Carroll, J. (2017). An online learning platform for teaching, learning, and assessment of programming. *Global engineering education conference* (pp. 547–556). Athens, Greece: IEEE.

Rodríguez-del-Pino, J. C., Rubio-Royo, E., & Hernández-Figueroa, Z. J. (2012). A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features. *Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government* (pp. 80–85). Las Vegas, USA: CSREA Press.

Shamsi, F., & Elnagar, A. (2012). An intelligent assessment tool for students' Java submissions in introductory programming courses. *Journal of Intelligent Learning Systems and Applications, 4*(1), 59–69.

Singh, R., Gulwani, S., & Solar-Lezama, A. (2013). Automated feedback generation for introductory programming assignments. *Proceedings of the 34th ACM SIGPLAN conference on programming language design and implementation* (pp. 15–26). Seattle, Washington, USA: ACM.

Stajduhar, I., & Mausa, G. (2015). Using string similarity metrics for automated grading of SQL statements. *Proceedings of the 38th IEEE international convention on information and communication Technology, electronics and microelectronics* (pp. 1250–1255). Opatija, Croatia: IEEE.

Staubitz, T., Klement, H., Renz, J., Teusner, R., & Meinel, C. (2015). Towards practical programming exercises and automated assessment in Massive Open Online Courses. *Proceedings of the 2015 IEEE international conference on teaching, assessment, and learning for engineering* (pp. 23–30). Zhuhai, China: IEEE.

Suleman, H. (2008). Automatic marking with Sakai. *Proceedings of the 2008 annual conference of the south African Institute of computer scientists and information technologists on IT research in developing countries* (pp. 229–236). Wilderness, South Africa: ACM.

Vujošević-Janičić, M., Nikolić, M., Tošić, D., & Kuncak, V. (2013). Software verification and graph similarity for automated evaluation of students' assignments. *Information and Software Technology, 55*(6), 1004–1016.

Wang, T., Su, X., Ma, P., Wang, Y., & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. *Computers & Education, 56*(1), 220–226.

Watson, C., & Li, F. W. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 ACM conference on innovation & Technology in computer science education* (pp. 39–44). Uppsala, Sweden: ACM.

Thanasis Daradoumis holds a PhD in Computer Science from the Polytechnic University of Catalonia. He combines his role as Associate Professor at the University of the Aegean, Department of Cultural Technology and Communication with his collaboration at the Open University of Catalonia within the Department of Computer Science, Multimedia and Telecommunications. He is also researcher at the Internet Interdisciplinary Institute (IN3) as well as at the eLearn Center (eLC). His research interests are: Emotional Intelligence, Alternative (Holistic) Education, Learning Analytics, E-learning, Collaborative, Affective and Adaptive Systems, CSCL. He has been advisor of over 20 MSc theses and 7 PhD already completed. He has published over 110 Scopus-indexed papers, from which over 20 are articles in ISI-JCR journals. He is a member of the editorial board of several international conferences and journals, whereas he has coordinated or participated in various National and European R&D projects.

Joan Manuel Marquès holds a PhD in Computer Science from the Universitat Politècnica de Catalunya. He is an Associate Professor at Computer Sciences, Multimedia & Telecommunication Studies at Universitat Oberta de Catalunya a (UOC) since 1997. His research interests include the design of scalable and cooperative Internet services and applications, privacy and e-learning.

Marta Arguedas has been a Lecturer at the program of Technical Engineering in Computer Systems of La Almunia Polytechnic University School ascribed to the University of Zaragoza, Spain, involved in the areas of Computer Architecture and Technology as well as of Computer Languages and Systems until September 2012. She holds a PhD in Education and ICT (eLearning) from the Open University of Catalonia, a Bachelor degree in Technical Engineering in Computer Systems from La Almunia Polytechnic University School and two Masters, a Master in Education and ICT (e-learning) and a Master in Multimedia Applications, from the Open University of Catalonia, Spain. Currently, her research interests are: Emotional Intelligence, Learning Analytics, E-learning, Collaborative, Affective and Adaptive Systems, Interaction Analysis, CSCL, CSCW, Knowledge Management.

Laura Calvet holds a PhD in Network and Information Technologies from the Open University of Catalonia (UOC). She has been a lecturer at the Autonomous University of Barcelona, the University of Barcelona, the International University of Valencia, and the Euncet Business School, involved in the areas of Statistics, Mathematics, Industrial Economics, and Econometrics. From September 2017, she works as lecturer in the Department of Computer Science, Multimedia and Telecommunication (EIMT) at the UOC. Her research is related to Applied Statistics, Metaheuristics, Simulation, and e-Learning.