This is a postprint version of the following published document:

Gallego-Romero, J.M., Alario-Hoyos, C., Estévez-Ayres, I. et al. Analyzing learners' engagement and behavior in MOOCs on programming with the Codeboard IDE. *Education Tech Research Dev* 68, 2505–2528 (2020).

DOI: [10.1007/s11423-020-09773-6](https://doi.org/10.1007/s11423-020-09773-6)

Analyzing learners' engagement and behavior in MOOCs on programming with the Codeboard IDE

Jesús Manuel Gallego-Romero, Carlos Alario-Hoyos, Iria Estévez-Ayres, Carlos Delgado Kloos
Universidad Carlos III de Madrid, Avenida Universidad 30, 28911 Leganés (Madrid), Spain
{jegalleg@pa, calario@it, ayres@it, cdk@it}.uc3m.es

Corresponding Author E-mail: calario@it.uc3m.es

ABSTRACT

MOOCs (Massive Open Online Courses) can be enhanced with the so-called learning-by-doing, designing the courses in a way that the learners are involved in a more active way in the learning process. Within the options for increasing learners' interaction in MOOCs, it is possible to integrate (third-party) external tools as part of the instructional design of the courses. In MOOCs on computer sciences, there are, for example, web-based IDEs (Integrated Development Environments) which can be integrated and that allow learners to do programming tasks directly in their browsers without installing desktop software. This work focuses on analyzing the effect on learners' engagement and behavior of integrating a third-party web-based IDE, *Codeboard*, in three MOOCs on Java programming with the purpose of promoting learning-by-doing (learning by coding in this case). In order to measure learners' level of engagement and behavior, data was collected from Codeboard on the number of compilations, executions and code generated, and compared between learners who registered in Codeboard to save and keep a record of their projects (registered learners) and learners who did not register in Codeboard and did not have access to these extra features (anonymous learners). The results show that learners who registered in Codeboard were more engaged than learners who did not register (in terms of number of compilations and executions), spent more time coding and did more changes in the base code provided by the teachers. The main implication of this study suggests the need for a trade-off between designing MOOCs that allow a very easy and anonymous access to external tools aimed for a more active learning, and forcing learners to give a step forward in terms of commitment in exchange for benefitting from additional features of the external tool used.

Keywords

MOOCs; programming; learners' engagement; Codeboard; learning analytics; edX

1. INTRODUCTION

MOOCs are an opportunity for learners from all over the world to study online courses from prestigious universities, although research studies indicate that people who already have a higher education degree and come from upper socioeconomic levels are the ones who take greater advantage from these courses (Hansen & Reich, 2015). At the same time, MOOCs are serving to rethink the digital education strategy in many higher education institutions and, in some cases, are also serving to outsource professional master's degrees at a cost, leaving aside the term "open" of the acronym MOOC (Reich & Ruipérez-Valiente, 2019). In fact, 2019 was the year of the consolidation of MOOC-based degrees and microcredentials programs associated with MOOCs (called Specializations in Coursera, MicroMasters and Professional Certificates in edX, Nanodegrees in Udacity, etc.), and the incorporation of more and more universities to offer MOOCs, adding up to more than 13,500 MOOCs offered, more than 900 universities offering MOOCs, and more than 100 million learners worldwide, according to Class Central (Shah, 2019).

Most MOOCs share a basic instructional design based on video lectures with complementary formative and summative exercises, typically automatic correction or peer-to-peer correction exercises, organized in weeks or modules (Margaryan, Bianco, & Littlejohn, 2015). At the same time, most MOOCs suffer the problem of high dropout rates reaching a typical number of 90-95% among enrollees (Li, Gao, Li, Xiong, Wen, et al., 2016). The fact that MOOCs are open and do not have an entry fee contributes to learner dropout (Evans, Baker, & Dee, 2016) (De Freitas, Morgan, & Gibson, 2015); actually, a good part of this high dropout rate is inevitable (Gütl, Rizzardini, Chang, & Morales, 2014). Nevertheless, the fact that the instructional design is poor or that the level and contents of the MOOC are not what the learners are looking for are also other relevant factors that contribute to these high dropout rates (Khalil & Ebner, 2014).

Several authors have discussed in the literature how to improve the design of MOOCs (Yousef, Chatti, Schroeder, & Wosnitza, 2014), and there is a widespread agreement that learners who do more activities and participate more in the course forums learn

more than learners who just watch video lectures and read documents (Ferguson & Sharples, (2014) (Koedinger, Kim, Jia, McLaughlin, & Bier, 2015) (Wang, Yang, Wen, Koedinger, & Rosé, 2015) (Rai & Chunrao, 2016). An analysis of the most popular MOOCs, with the aim to understand learner engagement, stressed that one of the main factors that contributes to increasing learners' engagement is a MOOC design which promotes active learning and problem-based learning (Hew, 2016). However, the main platforms for the deployment of MOOCs, Coursera, edX, FutureLearn, etc., barely include built-in tools aimed at promoting active learning and the realization of interactive activities, beyond the typical multiple response quizzes or similar exercises for automatic correction, or a few context-specific exercises for certain specific topics (Gilbert, 2015).

One approach to promote active learning in MOOCs with the aim to achieve greater learner engagement is the integration of (third-party) external tools as part of the instructional design of MOOCs. This idea has already been discussed in the literature through works such as that of Aleven, Baker, Blomberg, Andres, Sewall, et al. (2017), who integrated two external tools in edX with the aim to "facilitate the addition of learning-by-doing activities". In this way it is possible to complement the built-in tools available in MOOC platforms with context-specific (external) tools. This integration can be done through ad hoc software developments for the platform where the MOOC is deployed (for example, in edX XBlocks can be developed to add new components to the MOOC platform (Freire, del Blanco, & Fernández-Manjón, 2014) (Volchek, Romanov, & Mourontsev, 2017)), or through widely accepted integration standards such as the IMS LTI (Learning Tools Interoperability) (IMS LTI, 2019), which is adopted by Coursera, edX and some of other major MOOC platforms (Aleven et al., 2017) (Funieru, & Lăzăroiu, 2016). Actually, edX was the pioneer in encouraging full integration of third-party external tools in MOOCs among major MOOC platforms (Godwin-Jones, 2014), and so there are more related works in the literature on edX.

In the case of MOOCs in the area of computer science, which typically include programming activities in which learners need to develop some code, the integration of development environments to carry out these practical activities can be highly beneficial, and also a way to promote active learning (Krugel & Hubwieser, 2017). Although some platforms such as edX give partial support for JavaScript problems (Gilbert, 2015), the support to activities in other programming languages is scarce or non-existent (Staubitz, Klement, Renz, Teusner, & Meinel, 2015). It is always possible for each learner to install the necessary development environment locally on his or her computer, although this can lead to problems arising from the installation process, such as differences between software versions, incompatibility of libraries, etc., which can make it difficult to finally do the required activities (Meyer, 2017). Moreover, support from teachers to learners in the MOOC forums is very valuable and should not be used to solve some of these technical challenges which are not in the core of the course (Hollands, & Tirthali, 2014).

A popular development environment that has been integrated in several programming MOOCs in a seamlessly way through the IMS LTI standard is Codeboard (Morales-Chan, et al., 2017) (Meyer, 2017) (Alario-Hoyos et al., 2018). Codeboard is a web-based development environment which can be used to design programming activities (assignments) in multiple programming languages (Java, C, C++, Python...) (Codeboard, 2020). These activities can be integrated as regular activities in the learning sequences of a MOOC thanks to the IMS LTI standard, and do not require learners to install anything in their computers. Codeboard supports concurrent access of thousands of learners to the same programming activity, which is a critical non-functional requirement for its integration in MOOCs. Teachers provide a base code structure in each activity and can also include test cases so that these can be used to grade learners (Chi, Allen, & Jones, 2016). Learners must work on the base code provided by the teachers and develop a code which fulfils the instructions provided in each Codeboard activity.

In fact, learners using Codeboard integrated into a MOOC have two options for managing their Codeboard activities (Codeboard, 2020). The first and simplest option is doing nothing in particular, that is going to the corresponding Codeboard activity in the learning sequence of the MOOC, and appear as an *anonymous learner*; every time an anonymous learner loads a Codeboard activity the base code provided by the teachers will be shown (no matter if the learner did some changes in the code of that Codeboard activity in the past); this is the most seamless form of integration of Codeboard in a MOOC. The second option is to register in Codeboard at any time (this being an additional registration process to that of the MOOC); every time a *registered learner* loads a Codeboard activity in the learning sequence of the MOOC the latest version of his/her modified code will be shown; this option is a little less seamless, although once the learner signs in for a Codeboard activity, he or she will be signed in for the remaining Codeboard activities.

The second option allows the learner to keep his/her progress in each Codeboard activity and denotes, in principle, a higher level of engagement on the learner side, which is essential in MOOCs designed to promote active learning and learning-by-doing. In this context, it is relevant to try to make a more exhaustive analysis of the level of engagement and coding behavior of the two types of learners which can be found in MOOCs on programming that integrate Codeboard as a tool to support code development activities: *anonymous learners* and *registered learners*. The following two research questions (RQs), which arise in this context, will be addressed through the study presented in this paper.

- RQ1: What are the differences between anonymous learners and registered learners in terms of their level of engagement (measured in number of accesses, correct and incorrect compilations, and correct and incorrect executions of code) in the MOOCs of programming which integrate Codeboard as a support tool for code development activities?

- RQ2: What are the differences between anonymous learners and registered learners in terms of their coding behavior (measured in changes in the code and time dedicated to coding) in the MOOCs of programming which integrate Codeboard as a support tool for code development activities?

In order to answer these two research questions, a study is conducted using three MOOCs whose purpose was to introduce Java programming, and which were deployed in edX, integrating a total of 117 Codeboard activities. For this study it was necessary to install a local instance of Codeboard and do some modifications in the code of Codeboard to collect the necessary data aimed to answer these two research questions.

The remainder of this paper is structured as follows. Section 2 presents the literature review, including a discussion on publications related to the integration of external tools in learning platforms to support learner-centered models in general, the integration of external tools in MOOCs on programming in particular, and the integration of Codeboard to specifically support interactive programming activities. Then, Section 3 introduces the research methods, including the environment used in this study, Codeboard, as well as the data collected and analyzed in order to answer the research questions. Sections 4 and 5 present and discuss the results of the analysis. Finally, Section 6 presents the conclusions of this study and the future lines of work.

2. LITERATURE REVIEW

This section presents the related work using a top-down approach starting with a discussion of publications on learners' engagement and behavior in MOOCs, continuing with a discussion on the integration of third-party external tools in learning platforms to support learner-centered learning models, in general, and continuing with a discussion of the integration of third-party external tools in MOOCs, and particularly on MOOCs on programming, and finalizing with the discussion of the integration of Codeboard to support interactive programming activities (both in MOOCs and in other learning environments).

2.1 Learners' engagement and behavior in MOOCs

The engagement level and behavior of learners enrolled in MOOCs are very diverse, mainly due to the heterogeneity in their backgrounds, previous knowledge and motivations. Several authors have tried to characterize MOOC learners according to their level of engagement and behavior. [Ramesh, Goldwasser, Huang, Daumé III, and Getoor, \(2013\)](#) first classified MOOC learners in: 1) active engagement (those who show explicit signs of engagement such as posting in forums or submitting assessments); 2) passive engagement (those who watch lectures or view/subscribe to posts in forums); and 3) disengaged (those who show a significant decrease in their activity throughout time). Another pioneer related research article published by [Hill \(2013\)](#) classified MOOC learners according to five profiles: 1) no-shows (those who register in the MOOC but never log in); 2) observers/lurkers (those who log in but take no assessments); 3) drop-ins (those who take some assessments but do not attempt to complete the MOOC); 4) passive participants (those who see the MOOC as content to consume); and 5) active participants (those who participate in all the course activities). The authors [\(Alario-Hoyos, Pérez-Sanagustín, Delgado-Kloos, Parada G., & Muñoz-Organero, 2014\)](#) built upon that work [\(Hill, 2013\)](#) classifying MOOC learners in seven profiles in the case of instructor-paced MOOCs: 1) no-shows, 2) observers, 3) drop-ins, 4) latecomers, 5) drop-in latecomers, 6) non-engaged and 7) engaged. This classification took into account the moment of arrival/departure to/of the course for learners who do not follow the MOOC from beginning to end (drop-ins, latecomers, drop-in latecomers) and also the participation in the course discussions (e.g., in the course forums) and contribution to enrich the course for those who did finish the MOOC (non-engaged, engaged). [Maldonado-Mahauad, Pérez-Sanagustín, Kizilcec, Morales, and Munoz-Gama \(2018\)](#) used data mining tools to extract behavioral patterns of MOOC learners based on self-regulated learning strategies, classifying MOOC learners in three groups: 1) comprehensive learners (those who follow the sequence structure of the course materials as defined by the instructor of the MOOC); 2) targeting learners (those who engage with specific course content directly aimed at helping them pass the course assessments); and 3) sampling learners (those who exhibit a more erratic and less goal-oriented behavior); this third group always underperforms the first two. A similar three-group classification for MOOC learners was proposed by [Xu and Yang \(2016\)](#): 1) video watching (learners who prioritize the knowledge gained and therefore similar to the above mentioned comprehensive learners); 2) certification earning (learners who prioritize the certificate gained and therefore similar to the above mentioned sampling learners); and 3) course sampling (learners who just check part of the course contents and therefore similar to the above mentioned sampling learners).

Several research articles have carried out interventions with the aim to increase learners' engagement in MOOCs, although it is important to note that a non-negligible number of the learners enrolled in each MOOC have no real intention of taking the course, so no action will increase the engagement or modify the behavior of these learners anyway [\(Hill, 2013\)](#) [\(Gütl, Rizzardini, Chang, & Morales, 2014\)](#). Some interventions from the course instructional design which can increase learners' engagement include: 1) the definition of microsteps to simplify the learning process and increase learners' ability to succeed [\(Kizilcec, Piech, & Schneider, 2013\)](#); 2) the promotion of a growth mentality among learners [\(Kizilcec, Piech, & Schneider, 2013\)](#); 3) the setup of questions in forum discussions aimed at triggering a debate where everyone can express their opinion [\(Sunar, White, Abdullah, & Davis, 2016\)](#); and 4) the promotion of innovative active learning activities that go beyond the

capabilities of the MOOC platforms (Hew, 2016). All in all, in the case of some of the learners enrolled in a MOOC it is possible to modify their behavior and increase their engagement with the course, and for this reason this work focuses on one of this possibilities which is the integration of external tools in order to design and develop activities that promote active learning and support learning models which are more centered in the learner.

2.2 Integration of external tools in learning platforms to support learner-centered learning models

For many years educational systems focused on content, with the teacher's main job being to create content and/or divide it into smaller pieces that could be manageable for the context of a one-hour class (Norman, & Spohrer, 1996). However, research in the area of education has put the focus on the learner in the last two decades (Henson, 2003), since learner-centered education, including active learning and its achievement through strategies such as project-based learning or collaborative learning, is generally more effective than traditional passive instruction (teacher-centered education) (Prince, 2004). However, as education moved from the face-to-face classroom to online and blended learning models, some teachers returned to passive instruction, sometimes due to the difficulty of promoting learners' interaction in the online world, partially caused by the lack of context-specific built-in tools aimed at promoting active learning in the main platforms used to support online or blended learning environments (Ally, 2008) (Kim, & Bonk, 2006) (Kumar, Gankotiya, & Dutta, 2011). With the advent of MOOCs, advances related to the promotion of active learning in online environments were again slowed down, hampered by the lack of context-specific built-in tools aimed to promote interaction in the MOOC platforms, this time with the additional difficulty of needing scalable tools able to support thousands of learners working simultaneously (Bali, 2014) (Koedinger, Kim, Jia, McLaughlin, & Bier, 2015).

The problem of the lack of adequate tools to promote active learning in online environments (in general), and later in MOOCs (in particular), has been extensively studied in the literature. One of the main solutions used has been to develop *ad hoc* tools for specific platforms, as long as these platforms were open source or had an API that allowed connecting the tool developed with the platform. For example, Moodle already has a collection of more than 1,600 plugins which can be added to a local Moodle installation (Moodle Plugins, 2020), while Open edX also has a public directory with XBlocks which can be added to this platform when installed locally to offer MOOCs (XBlocks Directory, 2020); both repositories include general-purpose and context-specific tools which may be useful for teachers depending on the course they teach.

There have also been attempts to propose general purpose architectures and protocols to be used with multiple tools and multiple online learning platforms (Wilson, Daniel, Jugel, & Soi, S., 2011) (Alario-Hoyos, Bote-Lorenzo, Gómez-Sánchez, Asensio-Pérez, Vega-Gorgojo, & Ruiz-Calleja, 2013) (Sirkiä, & Haaranen, 2017). The authors (Alario-Hoyos, & Wilson, 2010) carried out some time ago a comparison of architectures and protocols for the integration of external tools in online learning platforms. Among them, IMS LTI (Learning Tools Interoperability) became a standard, and has been the most accepted solution among tool providers and platform providers (Severance, Hanss, Hardin, 2010) (IMS LTI, 2020). Several articles published in the literature have taken advantage of the adoption of IMS LTI to integrate new tools into online learning environments, including general-purpose tools (Forment, Guerrero, Mayol, Piguillem, Galanis, et al., 2012), games (Fontenla, Pérez, & Caeiro, 2011) (Freire, del Blanco, & Fernández-Manjón, 2014), or context-specific tools such as programming environments (Jurado, & Redondo, 2016), among others.

All in all, there have been numerous attempts to apply learner-centered learning models in online and blended education by integrating external tools into online platforms, including MOOC platforms. Although, unfortunately, this is not something that widespread, it is true that it is becoming more and more popular. From the average teacher's point of view, it is necessary to analyze the most suitable tools, from those available, for the course to be taught, and maybe try to integrate external tools using, for example, IMS LTI. Next subsection narrows the discussion focusing on the case of MOOCs, and specifically on the case of MOOCs on programming, this being a relevant area with an important number of related courses offered in the main MOOC platforms.

2.3 Integration of external tools in MOOCs on programming

According to Class Central (Shah, 2019), technology-related topics stand out as some of the most popular topics in which universities are offering MOOCs. Among the technology-related topics, MOOCs focused on teaching programming, or in which programming is an important tool for achieving learning objectives, also stand out. MOOCs should be designed, in general, to promote active learning (Hew, 2016), but in the case of MOOCs on programming or in MOOCs where programming is an essential part, the instructional design should be even more focused on promoting active learning, through problem resolution and learning-by-doing (or rather learning-by-coding). Learning programming requires applying fixed syntax rules, as well as practicing the acquired skills through the development of programs and algorithms (Brusilovsky, Kouchnirenko, Miller, Tomek, 1994) (Matthíasdóttir, 2006). Therefore, passive learning based on viewing video lectures, reading documents, or solving multiple choice activities is definitely not enough in MOOCs on programming, and appropriate tools that facilitate the resolution of coding problems should be used too. Unfortunately, the main MOOC platforms hardly incorporate built-in

programming tools, although there have been several successful cases of integration of external tools in MOOCs to support programming activities (Ben-Ari, 2013).

In the case of interpreted programming languages, such as Python or JavaScript, it is easier for MOOC platforms to incorporate tools that support these languages. This is the case of the integration of CodeSkulptor (Tang, Rixner, & Warren, 2014) in Coursera or edX (Warren, Rixner, Greiner, & Wong, 2014) (Hew, 2016), which allows programming directly from the browser in Python while programs are saved in the cloud (Ben-Ari, 2013). As for the evaluation of programming activities carried out in interpreted programming languages, the work of Pu et al. (Pu, Narasimhan, Solar-Lezama, & Barzilay, 2016) stands out for automatic correction of small Python assignments, being able to fix syntactic and semantic errors.

Regarding integration of tools in MOOCs for other specific programming languages, the integration of Eiffel4MOOC in the MOOC on “Introduction to Programming” by ETH Zurich (Switzerland) in 2013 was a relevant milestone as this integrated environment allowed learners, for the first time, to do programming activities in Eiffel, directly in the browser, compiling and executing Eiffel code in the cloud (Meyer, 2017). As another example, Király et al. (Király, Nehéz, & Hornyák, 2017) developed an automatic grader for Java programs which were submitted to their MOOC platform, called MeMOOC, built on a local instance of Open edX (the open-source platform developed by edX, freely available to be installed by any institution which wants to offer MOOCs).

With regard to tools or development environments that support multiple programming languages and that have been used in the context of MOOCs on programming, it is worth mentioning the INGINious platform (Derval, Gego, Reinbold, Frantzen, & Van Roy, 2015), which acts as an automatic grader and allows providing feedback to learners for programs written in C++, Java, Scala or Python. INGINious can be integrated in edX through a plug-in to grade learners’ work submitted to the MOOC, although it has also been used with regular on-campus learners from Université Catholique de Louvain (Belgium). Finally, CodeOcean (Staubitz, Klement, Teusner, Renz, & Meinel, 2016) is another tool developed at the Hasso Plattner Institute (Germany) to facilitate the compilation and execution of programs in MOOCs. This tool has been tested to some extent in three MOOCs offered through the openHPI MOOC platform, testing its potential scalability through simulations.

All in all, there have been some interesting attempts to integrate external tools and development environments to support MOOCs on programming. However, these attempts have either been specific for a programming language or have not been adopted across several MOOCs provided by several institutions. The exception to this in the literature is Codeboard (Codeboard, 2020), a web-based development environment that has been integrated into several programming MOOCs offered by several institutions to support programming activities in multiple programming languages, and that has also been useful in other educational contexts to teach and learn how to program.

2.4 Integration of Codeboard to support programming activities

The first courses in which Codeboard was integrated to support the realization of programming activities directly from the browser, compiling and running them on the cloud, were the MOOCs entitled “Introduction to Programming” and “Computing: Art, Magic, Science?”, offered in edX by ETH Zurich, the institution where Codeboard was developed (Meyer, 2017). Codeboard changed the way in which programming was taught, both in online and in blended courses, and also the interaction between teachers and learners in several ways (Meyer, 2017): 1) it removed the difficulties of setting up a proper development environment for courses on programming; 2) instructors could provide a base code with the level of detail they wanted for learners to fill in the missing parts (this is especially useful when learners are starting to program); 3) the base code provided could serve to guide learners who later must write similar codes themselves, learning through imitation; 4) learners could ask for help to teaching assistants just providing the link to their projects, instead of sending a ZIP file with the all their code, thus reducing the administrative workload and response time from teaching assistants. Additionally, Antonucci et al. (Antonucci, Estler, Nikolić, Piccioni, & Meyer, 2015), also from ETH Zurich, added a hint button for learners who got lost when coding so that they received increasingly detailed hints. Learners found helpful this extra functionality to guide them through the process of building their solution.

Some authors have also used Codeboard also to support face-to-face or blended courses either using Codeboard directly through the official website or integrating it with traditional learning platforms, as it supports also the integration with, for instance, Moodle (Meyer, 2017). For example, Chi et al. (Chi, Allen, & Jones, 2016) used the functionality of Codeboard that serves to evaluate learners’ code through a test-driven approach, providing the corresponding test cases which were used as grading templates, obtaining promising outcomes on the use of Codeboard. España-Boquera et al. (España-Boquera, Guerrero-López, Hermida-Pérez, Silva, & Benlloch-Dualde, 2017) replaced their conventional IDE on a programming course by Codeboard, detecting an improvement on learners’ final marks, which could suggest also learning gains.

Once again in the field of MOOCs, Škorić et al. (Škorić, Pein, & Orehovački, 2016) did a search for “web IDE” and “online IDE”, with the aim to look for a development environment to be integrated in their MOOC on programming. From the 35 results obtained, adding the requirements of open source, free to use, support to different programming languages, and interoperability with MOOC platforms, and using the Analytic Hierarchy Process method, they decided to choose Codeboard.

These authors studied the usefulness and ease of use of Codeboard through questionnaires collecting learners' perceptions and concluded that learners had a positive attitude towards Codeboard and were willing to use it in order to learn to program.

Likewise, Morales-Chan et al. (Morales-Chan, et al., 2017) also integrated Codeboard in several MOOCs on programming (specifically on mobile development with Android) deployed in edX. These authors collected self-reported information from surveys to evaluate learners' motivation and perception of the usefulness of Codeboard. The results of this study showed that learners saw Codeboard as a useful tool for learning programming and also that this tool increased learners' motivation and engagement with the MOOC.

Furthermore, the authors of this paper (Alario-Hoyos et al., 2018) also integrated Codeboard in several MOOCs on introduction to programming deployed in edX, finding out also that learners see Codeboard as a useful tool to learn programming. In addition, the above mentioned study compared the differences in the use of Codeboard when the MOOCs in which it is integrated run in an instructor-paced (synchronous) mode and in a self-paced (asynchronous) mode, concluding that the MOOC delivery mode did not affect the interaction of learners with Codeboard in terms of overall number of accesses, compilations, and executions, but that learners engaged more with Codeboard (had a greater preference to register in Codeboard rather than work as anonymous learners) when the MOOCs were released in an instructor-paced (synchronous) mode.

All in all, it is possible to conclude that Codeboard is a useful tool with promising results on learners' motivation and engagement when seamlessly integrated in MOOCs to support coding activities. However, there are still research gaps to be addressed, such as the differences between anonymous learners and registered learners in terms of engagement and coding behavior, which are the research questions addressed in this study. It is important to note that Codeboard does not have an API for data collection, and only some high-level data can be obtained directly through its web interface. Therefore, in order to collect the appropriate data to answer the research questions of this study, it is necessary to install Codeboard locally and also to modify its code to get relevant data.

3. RESEARCH METHODS

This section describes the research methods for this work. Firstly, Codeboard is introduced. Codeboard is the third-party external tool integrated into three MOOCs offered through edX. This is the learning environment upon which this research has been conducted. The pedagogical and technical aspects in relation to Codeboard which are necessary to understand the research study are also presented next. The process of data collection and analysis in order to answer the research questions is also presented in this section.

3.1 THE CODEBOARD IDE

Codeboard is an IDE (Integrated Development Environment) for on-cloud programming that has been used to enhance teaching and learning in different educational settings and for different programming languages (Chi, Allen, & Jones, 2016), (España-Boquera, Guerrero-López, Hermida-Pérez, Silva, & Benlloch-Dualde, 2017), (Morales-Chan, et al., 2017). The teacher creates and sets up a Codeboard activity and learners can use their preferred web browser to code directly in Codeboard without installing any additional software locally. In this way, learners can focus on the challenges of programming (syntax, algorithms, etc.) instead of in the logistics and administrative burden of setting up the appropriate development environment.

Every Codeboard activity has a different link (URL). The teacher can share the link with the learners who will receive the base code to be modified. Each learner receives a different instance of the Codeboard activity to work individually on the base code even though they all access the same link for the each Codeboard activity. Once the learner has accessed the base code through the web browser, s/he can compile the code, see if there are any compilation error messages (and correct them if so), execute the code, and even run tests on the code with input data and expected results (Meyer, 2017).

3.1.1 Pedagogical Aspects

Codeboard was integrated into three 5-week MOOCs on Java programming offered on edX. These three MOOCs formed a sequence of courses in a professional development program and their titles were: “*Introducción a la programación en Java: empezando a programar*” (“Introduction to Java programming: Starting to code”), “*Introducción a la programación en Java: escribiendo buen código*” (“Introduction to Java programming: Writing good code”) and “*Introducción a la programación en Java: estructuras de datos y algoritmos*” (“Introduction to Java programming: Fundamental data structures and algorithms”). In addition to videos and automatic correction exercises, the teachers of these three MOOCs created 117 Codeboard activities distributed throughout the several weeks: 38 activities in the first MOOC, 25 activities in the second MOOC, and 54 activities in the third MOOC. Considering that each MOOC had 5 weeks, that makes 7.8 Codeboard activities per week on average. Instructors designed the MOOCs with the aim to promote learning by doing week after week adding multiple Codeboard activities.

Codeboard activities were only formative activities, so they were not considered when calculating the final course grade for the learners. All the Codeboard activities included instructions on what to do, and a base code that learners had to modify by adding methods or classes. The solution to all Codeboard exercises was provided at the end of each week. Figure 1 shows an example of a Codeboard activity integrated in a learning sequence (“4.2 Binary Trees”) of the fourth week (“Week 4: Trees”) in the third MOOC. This learning sequence contains five Codeboard activities, all of them represented with a book icon in the top bar of Figure 1.

Learners enrolled in the MOOCs could work on the Codeboard activities in two different ways. First, learners could register in Codeboard (*registered learners*). This registration process is different from the edX registration process, which means that learners could use different usernames in the MOOCs and in Codeboard. Second, learners could work on the Codeboard activities without registering, that is in an anonymous way (*anonymous learner*). Registered learners have their code in the Codeboard activity associated with their accounts. If a registered learner modifies the base code provided by the teacher, the modified code will be stored in his/her account. If that registered learner returns to the same Codeboard activity later, s/he will get the version with the latest changes. Anonymous learners cannot save the changes they made in the code. Every time an anonymous learner clicks on a Codeboard activity in the MOOC s/he will get the base code provided by the teachers. Therefore, registered learners and anonymous learners behave differently when deciding how to face the Codeboard activities (deciding whether or not to register). The research questions that are addressed in this study refer to analyzing the differences between registered learners and anonymous learners with respect to their level of engagement and coding behavior in the Codeboard activities.

3.1.2 Technical Aspects

Codeboard is an open-source project available in GitHub ([Codeboard on GitHub, 2016](#)). A local instance of Codeboard was installed in a local server to collect the necessary data to answer the research questions of this paper. The Codeboard project consists of three Node.js applications which are deployed on Docker:

- *Codeboard* is the core of the project. It is divided into back-end (server-side) and front-end (client-side) and uses Node.js and AngularJS, respectively. The front-end takes care of the communication with the end-user. The back-end is in charge of the communication with the two databases of the application (MySQL and MongoDB).
- *Mantra* receives instructions from the Codeboard component to compile and run programs (e.g., Java programs).
- *Kali* receives instructions from the Codeboard component related to testing programs.

The *Codeboard* project has two databases:

- *MySQL*. It stores structural data, such as user accounts, project information, base code files, etc.
- *MongoDB*. It stores “activity logs” with information about each action performed by each user (compilations and executions, etc.): 1) a unique activity identifier (*_id*); 2) the project on which the action is performed; 3) the user who performs the action (if registered, or “anonymous” otherwise); 4) the type of action; 5) the start time (*startTime*) of the action; 6) the end time (*endTime*) of the action; 7) and the programming language used. Nevertheless, activity logs do not contain information about the result of the action, that is, whether the compilation was correct or incorrect and what error message was generated in the latter case, or the program output when running the program. Therefore, a new field called *description* was added in the MongoDB database in order to store the result of the compilation or execution of code.

In addition, Codeboard was modified to store the code compiled or executed by the learner in order to be able to undertake further analysis. This code is associated to the activity identifier (*_id*).

Codeboard was integrated into the above mentioned three MOOCs. This integration was possible thanks to the IMS LTI standard ([Severance, Hanss, Hardin, 2010](#)), which is a lightweight and widely adopted standard in education, supported natively by both edX and Codeboard. The platform, edX in this case (“LTI consumer”), embeds a tool, Codeboard in this case (“LTI provider”), through its URL and two additional data called “secret” and “key”, presenting a seamless user experience of the tool in a single web page of the platform (see example in Figure 1). It is very relevant to highlight that Codeboard was designed to scale up so that a Codeboard activity can receive requests from thousands of learners at the same time which is essential for an external tool to work properly when integrated into a MOOC.

Figure 1. Screenshot of a Codeboard activity integrated in a MOOC on Java Programming offered on edX. The top bar represents a sequence of activities, instructions for the Codeboard activity are below, and the Codeboard activity is seamlessly integrated in the lower half.

3.2 METHODOLOGY

This subsection details the participants in the experiment, the data collection process from the local installation of Codeboard, which was modified for the purpose of this study, and the data analysis process used to answer the research questions of this paper

3.2.1 Participants

The three MOOCs on which this research was conducted were part of a series of courses on “Introduction to Programming with Java”, were offered through edX and were in Spanish. These three MOOCs had 12,344 registered participants overall. Some overall demographic characteristics of the enrollees were: 1) largest group by gender, male (78%); 2) largest group by age, 25-30 years old; 3) largest group by level of studies, Bachelor’s Degree completed; 4) countries with more enrollees: Mexico, Spain and Colombia, in this order.

There are two important considerations that need to be considered about this overall number of enrollees. Firstly, the period of data collection was 40 days, between May 20 and June 30, 2018, which corresponds to the last days in which the MOOCs were available (the three MOOCs actually started in February, April and May 2018, respectively). Therefore, not all the learners enrolled in the MOOCs were active during the data collection period. This decision was made to try to minimize possible scalability problems which may happen when integrating third-party external tools in MOOCs, as the server where the local installation of Codeboard was hosted was not necessarily prepared to serve hundreds of compilation/execution requests in parallel, and in the last days of the MOOCs learners’ level of activity is much lower. Secondly, the Codeboard activities available among the three MOOCs were formative only, and learners could choose between registering in Codeboard (*registered learners*) or not (*anonymous learners*). 28 learners registered at Codeboard. Due to these considerations, the actual number of participants could be accurately determined, but was between the number of registered learners in Codeboard and the number of registered learners in the three MOOCs.

3.2.2 Data collection

Prior to the data collection period, the MOOC learners used the official Codeboard installation, which does not allow for low-level data collection. During the data collection period, MOOC learners used the local Codeboard installation modified for this study, which does allow for the collection of the required low-level data. The Codeboard activities were the same in both periods (only the link to the Codeboard installation changed), so that the learners would not notice any difference. During the data collection period, low-level data was collected from any of the 117 Codeboard activities distributed among the three MOOCs offered in edX.

The local Codeboard installation was the source for data collection. It is important to note that the data was not collected from the edX platform as it is not possible to identify the learners in both edX and Codeboard due to the type of integration used. In particular, four types of information were collected from Codeboard: 1) activity logs with all the actions referred to learners’ requests for code compilations/executions (MongoDB database); 2) access logs with the number of accesses of each learner to each Codeboard activity (MongoDB database); 3) code files organized in folders with the latest code that each learner sent for compilation / execution for each Codeboard activity (Codeboard file system); and 4) Java templates with the base code for each Codeboard project (MySQL database). In total, 5117 different activity logs were collected, with 2126 access logs, for the 117 Codeboard activities.

3.2.3 Data analysis

Data was cleaned up and then analyzed through Python scripts. More specifically, data from the MongoDB database (activity logs and access logs) was first exported to CSV files. The MySQL database where Java templates were stored was handled directly using SQL queries. These Java templates were compared with the code files stored in the folders created after the last action performed by each learner per project (compilation/execution). For each of these sources of information the data was separated according to whether these data had been generated by *anonymous learners* or learners who had registered in the new local Codeboard installation (*registered learners*). The data analysis comprises three parts:

1. *Comparison of the number of accesses and actions by anonymous and registered learners (RQ1).* Actions include code compilations and executions with and without errors. Practically all the base codes provided by the teachers already compiled correctly (except for a couple of Codeboard activities whose purpose was precisely that learners identified compilation errors).
2. *Comparison of the number of actions and time in each working session by anonymous and registered learners (RQ2).* Actions include code compilations and executions with and without errors. The time is calculated since the first action (always a code compilation) to the last action in the same Codeboard activity. If the learner leaves the current Codeboard activity and/or moves to another Codeboard activity, then the current session is closed.

3. *Comparison of the base code and the code generated by anonymous and registered learners in each Codeboard activity (RQ2).* The Levenshtein distance (Levenshtein, 1966) is used as a metric for code similarity, where the higher the value (in a scale from 0 to 100), the greater the similarity between the two codes. Teachers provided a base code in all the Codeboard activities and learners had to improve it. Although the code structure is expected to remain, the changes to be added in the code are important to solve the proposed Codeboard activity.

4. RESULTS

This section presents the results aimed to answer the two research questions of this study. Data was collected and analyzed as indicated in the previous section, comparing the level of engagement and coding behavior of the anonymous and registered learners who worked in the Codeboard activities in the three introductory MOOCs on Java programming.

4.1 RQ1: Differences in the level of engagement between anonymous and registered learners

At the end of the data collection period, the actions stored in the Codeboard database were analyzed, separating compilations and executions, and adding the number of accesses to Codeboard activities. 28 learners registered in Codeboard and accessed Codeboard activities, although, only 26 of them compiled or executed code. Overall, 2,126 accesses to Codeboard activities were counted, 1,736 (81.7%) of them from anonymous learners, and 390 (18.3%) from registered learners. A total of 5,110 actions (compilations or executions) were counted, 3,748 (73.3%) from anonymous learners and 1,362 (26.6%) from registered learners. Within the total number of actions, 732 actions resulted in errors: 613 compilation errors (21.6 % of the total number of compilations) and 119 executions errors (5.2% of the total number of executions of code). This information is presented in Table 1.

Table 1. Number of collected events by event type and learner type.

Event Type		Anonymous learners	Registered learners	All learners
<i>Accesses</i>	<i>Total Accesses</i>	1,736	390	2,126
<i>Actions</i>	<i>Total Actions</i>	3,748	1,362	5,110
	<i>Compilations</i>			
	<i>Successful</i>	1,653	565	2,218
	<i>With errors</i>	375	238	613
	<i>Executions</i>			
	<i>Successful</i>	1,612	548	2,160
	<i>With errors</i>	108	11	119

A more detailed analysis, comparing ratios between number of compilations/executions and number of accesses to Codeboard activities, was done to try to draw conclusions on the level of engagement of learners. More engagement with the Codeboard activities would be associated with a higher level of these ratios, as learners would work more per access on average. Table 2 shows the ratios between number of compilations/executions and number of accesses, and also the ratios of compilations/executions with errors and total number of compilations/executions, respectively. A ratio of 1.168 was obtained for anonymous learners dividing number of compilations between number of accesses, while this ratio was much higher, 2.059, in the case of registered learners. Similar results were obtained when calculating the ratio between the number of executions and the number of accesses, obtaining a ratio of 0.991 for anonymous learners and a ratio of 1.449 for registered learners. Therefore, on average, registered learners compiled and executed their code more times every time they accessed to Codeboard activities.

With the data stored after the expansion of Codeboard it was also possible to identify the number of errors that each learner made while working in the Codeboard activities, and to relate this number of errors with the total number of actions. As almost all the base codes provided by the teachers in the Codeboard activities were free of compilation/execution errors, a learner that did a compilation or execution error suggests that he or she was working to modify the code provided. Table 2 shows that the ratio of compilations with errors and total number of compilations is 0.296 for registered learners and 0.185 for anonymous learners. Due to the simple nature of the proposed exercises there were little execution errors, leading to low error ratios: 0.02 for registered learners and 0.063 for anonymous learners.

Table 2. Metrics comparing anonymous learners and registered learners.

	Anonymous learners	Registered learners
<i>Compilations/Accesses</i>	1.168	2.059
<i>Executions/Accesses</i>	0.991	1.449
<i>Compilations with errors/Total compilations</i>	0.185	0.296
<i>Executions with errors/Total executions</i>	0.063	0.02

In conclusion, the analyzed data suggest that registered learners have a greater level of engagement with Codeboard activities in the MOOCs on programming from which data have been collected. This is supported by the largest number of ratios in compilation/accesses, executions/accesses and compilations with errors/total compilations. This finding is not surprising at all, given that registered learners have already taken a step forward to show their engagement by opening a Codeboard account and signing in before starting their work on the Codeboard activities integrated in the MOOCs.

4.2 RQ2: Differences in the coding behavior between anonymous and registered learners

The second research question takes the analysis one step further in order to compare the coding behavior between anonymous learners and registered learners. For this purpose, activity logs were analyzed according to their identifiers so that work sessions per project of anonymous learners and work sessions per project of registered learners could be compared. Table 3 shows the number of actions per session (compilations and executions) as well as the average duration of each session, taking into account the timestamps for each action in the session. There were 1,036 sessions in total (820 from anonymous learners and 216 from registered learners). Interestingly, registered learners performed 6.35 actions per session on average, while anonymous learners performed only 4.57 actions, with a significant difference. The data collected also shows that registered learners spent more time per session than anonymous learners: 250.88 seconds for registered learners versus 148.58 seconds for anonymous learners, also with a significant difference. For those sessions in which learners compiled the code, the ratio of compilations with errors and total number of compilations, and the ratio of total number of executions and total number of compilations per session were also calculated. Registered learners had a higher ratio of compilations with errors and total number of compilations per session, which is consistent with the global result obtained in the case of RQ1 (although here the analysis is done per session instead of globally). Anonymous learners, however, had a higher ratio of number of executions and number of compilations, which indicates that the additional actions of the registered learners per session on average were mainly intended to compile the code. The comparisons were supported by a t-test between the groups of anonymous learners and registered learners, using the sessions as independent samples. It is important to take into account that it was not possible to do the comparison directly between learners as no information was available on the number of anonymous learners. Therefore, sessions had to be chosen as the independent outputs of learners' work. Assumptions of normal distribution of samples and homogeneity of variances (Levene's test) were checked before applying the t-test.

Table 3. Comparison between anonymous learners and registered learners taking into account all the sessions

	Anonymous learners			Registered learners			<i>t-test</i> <i>p-value</i>	
	Mean	SD	Number of sessions	Mean	SD	Number of sessions		
<i>Number of actions per session</i>	4.57	5.52	820	6.35	8.19	216	-3.78	0.0001
<i>Time per session (seconds)</i>	148.58	312.51	820	250.88	422.41	216	-3.95	<0.0001
<i>compilations with errors / total compilations per session</i>	0.091	0.243	819	0.177	0.293	211	-4.40	<0.0001
<i>total executions per session / total compilations per session</i>	0.922	0.359	819	0.798	0.422	211	4.28	<0.0001

Nevertheless, the learners did not necessarily work in all the above-mentioned sessions (in some sessions the learners simply compiled and executed the base code provided but without modifying it). Therefore, it is relevant to filter the sessions in which the learners actually modified the code provided by the teachers. As the code used in the last action (compilation/execution) is saved locally in the local installation of Codeboard, this code could be compared with the base code provided by the teachers, discarding those sessions in which learners did not change the code at all. Table 4 shows the number of actions per session in which the code was changed and the time per session for both anonymous and registered learners and in this case, there are no significant differences. However, even in this case, registered learners still had a higher ratio of compilation errors and total number of compilations per session, and again this result is consistent with that obtained in the case of RQ1. Likewise, anonymous learners still had a higher ratio of total number of executions and total number of compilations, which again indicates that the additional actions of the registered learners per session on average are mainly intended to compile the code. An interesting fact is the percentage of sessions where learners did changes in the code compared to the total number of sessions: 41.34% of the sessions for anonymous learners, and 62.5% of the sessions for registered learners.

Table 4. Comparison between anonymous learners and registered learners taking into account only the sessions in which the base code was modified

	Anonymous learners			Registered learners				
	Mean	SD	Number of sessions	Mean	SD	Number of sessions	<i>t-test</i>	<i>p-value</i>
<i>Number of actions per session</i>	7.88	7.33	339	8.64	9.58	135	-0.92	>0.1
<i>Time per session (seconds)</i>	322.62	417.7	339	365.06	472.9	135	-0.96	>0.1
<i>compilations with errors / total compilations per session</i>	0.187	0.315	339	0.263	0.33	134	-2.35	0.0187
<i>total executions per session / total compilations per session</i>	0.852	0.327	339	0.708	0.402	134	4.02	<0.0001

Finally, the Levenshtein distance, as a measure of similarity between the code modified by the learner and the base code provided by the teachers, was calculated for each of the coding files the learners worked with. It is important to bear in mind that in the Codeboard activities there could be one or several coding files. Table 5 shows the Levenshtein distance in the case of anonymous learners and registered learners, obtaining a lower value on average in the case of registered learners, which means that registered learners did more changes in the code on average. The comparisons were again supported by a t-test between the groups of anonymous learners and registered learners, using the files as independent samples. Since no information was available on the number of anonymous learners, files had to be compared as the independent outputs of learners' work. Assumptions of normal distribution of samples and homogeneity of variances (Levene's test) were checked before applying the t-test.

Table 5. Comparison between anonymous learners and registered learners taking into account Levenshtein distance

	Anonymous learners			Registered learners				
	Mean	SD	Number of files	Mean	SD	Number of files	<i>t-test</i>	<i>p-value</i>
<i>Levenshtein distance</i>	98.52	4.6	1069	95.53	9.15	298	7.74	<0.0001

In conclusion, the analysis shows that registered learners and anonymous learners had different coding behaviors when working in Codeboard activities in the three MOOCs under analysis. Considering all the sessions, registered learners worked more than anonymous learners, although this difference decreased when comparing sessions which actually entailed changes in the code. Levenshtein distance also contributed to seeing these differences in the work done by registered learners as compared to that of anonymous learners.

5. DISCUSSION

Several research articles have characterized MOOC learners according to their level of engagement (Ramesh, Goldwasser, Huang, Daumé III, & Getoor 2013) (Hill, 2013) (Maldonado-Mahauad, Pérez-Sanagustín, Kizilcec, Morales, & Munoz-Gama, 2018). Several research articles have also tried to increase learners' level of engagement in MOOC through interventions (Kizilcec, Piech, & Schneider, 2013) (Sunar, White, Abdullah, & Davis, 2016), such as the promotion of innovative active learning activities that go beyond the capabilities of the MOOC platforms (Hew, 2016). In the case under study, that intervention was the integration of a third-party external tool, Codeboard, to allow learners to do programming exercises directly from their browsers, thus extending the capabilities of the MOOC platform used, edX in this case. The chosen intervention relies on several research articles which stress the importance of engaging learners in programming courses through projects and practical activities, as opposed to simply reviewing lectures (or video lectures in the case of MOOCs) (Hansen, & Eddy, 2007) (Lister & Leaney, 2003) (Sarpong, Arthur, & Amoako, 2013).

The first research question is addressed using as metrics for the level of engagement the number of accesses, the number of correct and incorrect code compilations and the number of correct and incorrect code executions by MOOC learners in Codeboard activities. It is therefore important to justify the selection of these metrics to validate the results obtained. Some authors have already used the number of access events (Su, Ding, & Lai, 2017) or the number of compilation attempts (Fwa, & Marshall, 2018) to model learners' engagement in computer programming activities. This study considered not only the number of access events and the number of compilation attempts (either correct or incorrect), but also the number of code executions (either correct or incorrect). Both code compilation and code execution are actions that learners need to explicitly do by clicking two different buttons in Codeboard activities, and in order to see the output of the program developed. Nevertheless, an important limitation in relation to this first research question refers to the data sources. Integrating a third-party external tool in a MOOC entails some technological trade-offs. In this case, the trade-off meant that for scalability reasons the main logic was on the server side of the external tool, and only the results of the code compilations/executions were displayed to the learner (on the client side). The required simplicity on the client side meant that additional data that might have been interesting to collect, such as keystrokes or even learners' head motion (Fwa, & Marshall, 2018), could not be captured for this analysis.

The second research question is addressed analyzing learners' coding behavior through their coding sessions (and the actions performed in each of these coding sessions) and through the changes done by learners in the code skeleton provided by the instructors. Changes in the code skeleton provided by the instructors were calculated through the Levenshtein distance. The Levenshtein distance is a well-known metric that has been used by some anti-plagiarism systems in order to detect similarities between codes (Noh, Kim, & Jung, 2006) (Sheneamer, & Kalita, 2015) and even between texts written in natural language (De Lucia, Scanniello, & Tortora, 2004). In the case under study, the Levenshtein distance offers a first high level view of whether the learners actually worked on the code or not. The instructors have prepared the Codeboard activities in a guided way and with incremental difficulty, so it can be assumed that the learners who worked on these activities and solved them were learning programming. Nevertheless, an important limitation in relation to this second research question refers to the metric used. Additional metrics could have been used by analyzing in detail the files sent for compilation and execution by the learners, such as method calls, or loops/conditionals used. However, given that this study collects data from introductory MOOCs on programming and that templates with the code skeleton were provided by the instructors so that learners were only expected to complete a small part of the code to make it work as required, no major differences would be expected in general in the resulting code implemented by the learners.

6. CONCLUSIONS AND FUTURE WORK

This work has studied the level of engagement and coding behavior of two types of learners (anonymous learners and registered learners) when working with Codeboard activities integrated in three introductory MOOCs on Java programming. The data showed a higher level of engagement of registered learners in terms of the ratios of compilations and accesses, and executions and accesses, compared to anonymous learners. Registered learners also carried out a larger number of actions per session (understanding actions as code compilations and executions), and spent more time with the code, although the differences with anonymous learners was only significant when all the sessions were considered (including those sessions in which the code skeleton provided by the teachers was modified and in which this code skeleton was not modified). Finally, the code of registered learners was modified to a greater extent than that of anonymous learners as calculated through the Levenshtein distance between modified code and base code. The results of this work have implications on the design of MOOCs in which external tools are integrated with the aim to promote active learning and learning-by-doing. These implications have been discussed in some prior research articles which highlight the need for incorporating more sophisticated assessment types in MOOCs (Godwin-Jones, 2014). There is a trade-off between providing the simplest access to the external tools with less

features (i.e., not being able to save the changes in the code in the case of Codeboard), and getting less commitment on learners' side, or providing a more complicated access to the external tools in exchange for more features and getting more commitment on learners' side (i.e., creating a Codeboard account and logging in to be able to save the changes in their code). Therefore, more features can be beneficial for learners, but if this means more obstacles to access and use the external tool, then fewer learners will benefit from these features. For example, if instructors want to push learners to register in the external tool, then some of the activities that must be done in the external tool could be considered in the summative evaluation of the course (provided that the external tool can be connected to the MOOC evaluation system, something that IMS LTI in principle would allow). Furthermore, it could be interesting for tool developers to provide single sign-on between the MOOC platform and the external tool so that the MOOC learners are automatically registered in the external tool with their MOOC account. In this way it would be possible to link the learner's performance in the MOOC and in the external tool.

This work has several important limitations which lead to future lines of work. First, the need for modifying the code of Codeboard and the difficulties found in this process meant that the data collection process could not start earlier. This entailed collecting a limited amount of data from the last 40 days of the three MOOCs, which was the period with the lowest learners' activity. Because of this, the amount of data collected, especially from registered learners, was also quite limited. Therefore, it would be necessary to carry out a complementary study in which the collection of data from Codeboard activities could be expanded to a longer period of time. Second, it was not possible to match the work done by learners in the MOOCs (in term of watching videos and doing automatic correction exercises) and the work done by learners in Codeboard, since the latter allowed anonymous access, and even for registered learners the usernames in Codeboard and edX did not necessarily match. Therefore, it is not possible to determine if learners who registered in Codeboard were also more engaged than anonymous learners in the activities that did not involve the use of Codeboard (i.e., those that were done directly through the edX platform, such as watching videos and doing automatic correction exercises). An interesting future line of work would be to explore ways in which to try to match the work done by learners in a third-party external tool such as Codeboard and in edX in order to analyze if the level of engagement detected in Codeboard also holds when working in edX.

Other lines of future work include comparing the results obtained in this study with similar studies for other MOOCs on programming which also integrate Codeboard, and with MOOCs on programming which are offered with different delivery modes (instructor-paced and self-paced modes). Finally, an interesting future line of work would be to categorize the types of compilation and execution errors found by learners and stored in the extended version of Codeboard and use this information to improve the design of the programming activities with Codeboard, providing appropriate hints or recommendations to learners for each activity.

ACKNOWLEDGMENTS

The work received partial support from FEDER/Ministerio de Ciencia, Innovación y Universidades – Agencia Estatal de Investigación through project Smartlet (TIN2017-85179-C3-1-R), from the eMadrid Network, which is funded by the Madrid Regional Government (Comunidad de Madrid) with grant No. P2018/TCS-4307 and from the European Commission through Erasmus+ projects LALA (586120-EPP-1-2017-1-ES-EPPKA2-CBHE-JP), InnovaT (598758-EPP-1-2018-1-AT-EPPKA2-CBHE-JP) and PROF-XXI (609767-EPP-1-2019-1-ES-EPPKA2-CBHE-JP). This publication reflects the views only of the authors and funders cannot be held responsible for any use which may be made of the information contained therein.

REFERENCES

- Aleven, V., Baker, R., Blomberg, N., Andres, J. M., Sewall, J., Wang, Y., & Popescu, O. (2017, May). Integrating MOOCs and Intelligent Tutoring Systems: edX, GIFT, and CTAT. In *5th Annual Generalized Intelligent Framework for Tutoring Users Symposium* (pp. 11-21).
- Ally, M. (2008). Foundations of educational theory for online learning. In T. Anderson (Ed.), *The Theory and Practice of Online Learning* (2nd ed., pp. 15-44). Athabasca, Alberta: Athabasca University Press.
- Antonucci, P., Estler, C., Nikolić, D., Piccioni, M., & Meyer, B. (2015, June). An incremental hint system for automated programming assignments. In *2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 320-325). ACM.
- Alario-Hoyos, C., & Wilson, S. (2010, November). Comparison of the main alternatives to the integration of external tools in different platforms. In *Proceedings of the International Conference of Education, Research and Innovation, ICERI* (pp. 3466-3476).
- Alario-Hoyos, C., Bote-Lorenzo, M. L., Gómez-Sánchez, E., Asensio-Pérez, J. I., Vega-Gorgojo, G., & Ruiz-Calleja, A. (2013). GLUE!: An architecture for the integration of external tools in Virtual Learning Environments. *Computers & Education*, 60(1), 122-137.

- Alario-Hoyos, C., Pérez-Sanagustín, M., Delgado-Kloos, C., Parada G., H. A. & Muñoz-Organero, M. (2014). Delving into participants' profiles and use of social tools in MOOCs. *IEEE Transactions on Learning Technologies*, 7(3), 260-266.
- Alario-Hoyos, C., Estévez Ayres, I. M., Gallego Romero, J. M., Delgado Kloos, C., Fernández Panadero, M. C., Crespo García, R., et al. (2018). A study of learning-by-doing in MOOCs through the integration of third-party external tools: comparison of synchronous and asynchronous running modes. *Journal of Universal Computer Science*, 24(8), 1015-1033.
- Bali, M. (2014). MOOC pedagogy: glean good practice from existing MOOCs. *Journal of Online Learning and Teaching*, 10(1), 44-56.
- Ben-Ari, M. M. (2013). MOOCs on introductory programming: a travelogue. *ACM Inroads*, 4(2), 58-61.
- Brusilovsky, P., Kouchnirenko, A., Miller, P., Tomek, I., (1994, June). Teaching Programming to Novices: A Review of Approaches and Tools. In *1994 World Conference on Educational Multimedia and Hypermedia (ED-MEDIA)* (pp. 103-110).
- Chi, H., Allen, C., & Jones, E. (2016, July). Integrating Computing to STEM Curriculum via CodeBoard. In *International Conference on Computational Science and Its Applications* (pp. 512-529). Springer, Cham.
- Codeboard (2020). Retrieved April 2020, from: <https://codeboard.io/>
- De Freitas, S. I., Morgan, J., & Gibson, D. (2015). Will MOOCs transform learning and teaching in higher education? Engagement and course retention in online learning provision. *British Journal of Educational Technology*, 46(3), 455-471.
- De Lucia, A., Scanniello, G., & Tortora, G. (2004, April). Identifying Clones in Dynamic Web Sites Using Similarity Thresholds. In *Sixth International Conference on Enterprise Information Systems* (pp. 391-396).
- Derval, G., Gego, A., Reinbold, P., Frantzen, B., & Van Roy, P. (2015, May). Automatic grading of programming exercises in a MOOC using the INGINIOUS platform. In *European MOOCs Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCs'15)* (pp. 86-91).
- España-Boquera, S., Guerrero-López, D., Hermida-Pérez, A., Silva, J., & Benlloch-Dualde, J. V. (2017, July). Analyzing the learning process (in Programming) by using data collected from an online IDE. In *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)* (pp. 1-4). IEEE.
- Evans, B. J., Baker, R. B., & Dee, T. S. (2016). Persistence patterns in massive open online courses (MOOCs). *The Journal of Higher Education*, 87(2), 206-242.
- Ferguson, R., & Sharples, M. (2014, September). Innovative pedagogy at massive scale: teaching and learning in MOOCs. In *European Conference on Technology Enhanced Learning (EC-TEL 2014)* (pp. 98-111). Springer, Cham.
- Fontenla, J., Pérez, R., & Caeiro, M. (2011, April). Using IMS Basic LTI to integrate games in LMSs - Lessons from Game•Tel. In *2011 IEEE Global Engineering Education Conference (EDUCON)* (pp. 299-306). IEEE.
- Forment, M. A., Guerrero, M. J. C., Mayol, E., Piguillem, J., Galanis, N., García-Peñalvo, F. J., & Conde, M. Á. (2012). Docs4Learning: Getting Google Docs to work within the LMS with IMS BLTI. *Journal of Universal Computer Science*, 18(11), 1483-1500.
- Freire, M., del Blanco, Á., & Fernández-Manjón, B. (2014, April). Serious games as edX MOOC activities. In *2014 IEEE Global Engineering Education Conference (EDUCON)* (pp. 867-871). IEEE.
- Funieru, L. M., & Lăzăroiu F. (2016). Massive Open Online Courses (MOOCs): A Comparative Analysis of the Main Platforms. *Informatica Economică*, 20(2), 35-45.
- Fwa, H. L., & Marshall, L. (2018). Modeling engagement of programming students using unsupervised machine learning technique. *GSTF Journal on Computing*, 6(1), 1-6.
- Gilbert, M. A. (2015). *edX E-Learning Course Development*. Birmingham, UK: Packt Publishing Ltd.
- Godwin-Jones, R. (2014). Global reach and local practice: The promise of MOOCs. *Language Learning & Technology*, 18(3), 5-15.
- Gütl, C., Rizzardini, R. H., Chang, V., & Morales, M. (2014, September). Attrition in MOOC: Lessons learned from drop-out students. In *International workshop on learning technology for education in cloud* (pp. 37-48). Springer, Cham.
- Hansen, S., & Eddy, E. (2007). Engagement and frustration in programming projects. *ACM SIGCSE Bulletin*, 39(1), 271-275.
- Hansen, J. D., & Reich, J. (2015). Democratizing education? Examining access and usage patterns in massive open online courses. *Science*, 350(6265), 1245-1248.
- Henson, K. T. (2003). Foundations for learner-centered education: A knowledge base. *Education*, 124(1), 5-16.

- Hew, K. F. (2016). Promoting engagement in online courses: What strategies can we learn from three highly rated MOOCs. *British Journal of Educational Technology*, 47(2), 320-341.
- Hill, P. (2013). Emerging Student Patterns in MOOCs: A Graphical View. *e-Literate*. Retrieved April 2020, from: https://eliterate.us/emerging_student_patterns_in_moocs_graphical_view
- Hollands, F. M., & Tirthali, D. (2014). Resource requirements and costs of developing and delivering MOOCs. *The International Review of Research in Open and Distributed Learning*, 15(5), 113-133.
- IMS LTI (2019). Learning Tools Interoperability Core Specification, IMS Final Release, Version 1.3. Retrieved April 2020, from: <https://www.imsglobal.org/spec/lti/v1p3/>
- Jurado, F., & Redondo, M. A. (2016). IMS-LTI and web-services for integrating Moodle to an eclipse-based distributed environment for learning to program. *International Journal of Engineering Education*, 32(2), 1007-1014.
- Khalil, H., & Ebner, M. (2014, June). MOOCs completion rates and possible methods to improve retention-A literature review. In *EdMedia+ Innovate Learning* (pp. 1305-1313). AACE.
- Kim, K. J., & Bonk, C. J. (2006). The future of online teaching and learning in higher education. *Educause Quarterly*, 29(4), 22-30.
- Király, S., Nehéz, K., & Hornyák, O. (2017). Some aspects of grading Java code submissions in MOOCs. *Research in Learning Technology*, 25, 1-16.
- Kizilcec, R. F., Piech, C., & Schneider, E. (2013, April). Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *Proceedings of the third international conference on learning analytics and knowledge* (pp. 170-179).
- Koedinger, K. R., Kim, J., Jia, J. Z., McLaughlin, E. A., & Bier, N. L. (2015, March). Learning is not a spectator sport: Doing is better than watching for learning from a MOOC. In *Second ACM conference on learning@ scale* (pp. 111-120). ACM.
- Krugel, J., & Hubwieser, P. (2017, April). Computational thinking as springboard for learning object-oriented programming in an interactive MOOC. In *2017 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1709-1712). IEEE.
- Kumar, S., Gankotiya, A. K., & Dutta, K. (2011, April). A comparative study of Moodle with other e-learning systems. In *2011 3rd International Conference on Electronics Computer Technology* (Vol. 5, pp. 414-418). IEEE.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10, 707-710.
- Li, W., Gao, M., Li, H., Xiong, Q., Wen, J., & Wu, Z. (2016, July). Dropout prediction in MOOCs using behavior features and multi-view semi-supervised learning. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 3130-3137). IEEE.
- Lister, R., & Leaney, J. (2003, January). Introductory programming, criterion-referencing, and bloom. In *34th SIGCSE technical symposium on Computer science education* (pp. 143-147).
- Maldonado-Mahauad, J., Pérez-Sanagustín, M., Kizilcec, R. F., Morales, N., & Munoz-Gama, J. (2018). Mining theory-based patterns from Big data: Identifying self-regulated learning strategies in Massive Open Online Courses. *Computers in Human Behavior*, 80, 179-196.
- Matthíasdóttir, Á. (2006, June). How to teach programming languages to novice students? Lecturing or not. In *International Conference on Computer Systems and Technologies-CompSysTech* (Vol. 6, pp. 15-16).
- Margaryan, A., Bianco, M., & Littlejohn, A. (2015). Instructional quality of massive open online courses (MOOCs). *Computers & Education*, 80, 77-83.
- Meyer, B. (2017). Fourteen years of software engineering at ETH Zurich. *arXiv preprint arXiv:1712.05078* (pp. 1-118).
- Moodle Plugins (2020). Retrieved April 2020, from: <https://moodle.org/plugins/>
- Morales-Chan, M., de la Roca, M., Alario-Hoyos, C., Barchino-Plata, R., Medina, J. A., & Hernández-Rizzardini, R. (2017, November). Perceived usefulness and motivation students towards the use of a cloud-based tool to support the learning process in a Java MOOC. In *2018 International Conference MOOC-Maker (MOOC-Maker)* (pp. 73-82).
- Noh, S. Y., Kim, S., & Jung, C. (2006). A Lightweight Program Similarity Detection Model using XML and Levenshtein Distance. In *FECS* (pp. 3-9).
- Norman, D. A., & Spohrer, J. C. (1996). Learner-centered education. *Communications of the ACM*, 39(4), 24-27.
- Prince, M. (2004). Does active learning work? A review of the research. *Journal of engineering education*, 93(3), 223-231.

- Pu, Y., Narasimhan, K., Solar-Lezama, A., & Barzilay, R. (2016, October). *sk_p: a neural program corrector for MOOCs*. In *2016 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity* (pp. 39-40). ACM.
- Rai, L., & Chunrao, D. (2016). Influencing factors of success and failure in MOOC and general analysis of learner behavior. *International Journal of Information and Education Technology*, 6(4), 262-268.
- Ramesh, A., Goldwasser, D., Huang, B., Daumé III, H., & Getoor, L. (2013, December). Modeling learner engagement in MOOCs using probabilistic soft logic. In *NIPS workshop on data driven education* (pp. 1-7).
- Reich, J., & Ruipérez-Valiente, J. A. (2019). The MOOC pivot. *Science*, 363(6423), 130-131.
- Sarpong, K. A. M., Arthur, J. K., & Amoako, P. Y. O. (2013). Causes of failure of students in computer programming courses: The teacher-learner Perspective. *International Journal of Computer Applications*, 77(12), 27-32.
- Severance, C., Hanss, T., & Hardin, J. (2010). IMS Learning Tools Interoperability: Enabling a mash-up approach to teaching and learning tools. *Technology, Instruction, Cognition and Learning*, 7(3-4), 245-262.
- Sirkkä, T., & Haaranen, L. (2017). Improving online learning activity interoperability with acos server. *Software: Practice and Experience*, 47(11), 1657-1676.
- Shah, D. (2019). By the Numbers: MOOCs in 2019. Retrieved April 2020, from: <https://www.classcentral.com/report/mooc-stats-2019/>
- Sheneamer, A., & Kalita, J. (2015, December). Code clone detection using coarse and fine-grained hybrid approaches. In *2015 IEEE seventh international conference on intelligent computing and information systems (ICICIS)* (pp. 472-480). IEEE.
- Škorčić, I., Pein, B., & Orehovački, T. (2016, May). Selecting the most appropriate web IDE for learning programming using AHP. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 877-882). IEEE.
- Staubitz, T., Klement, H., Renz, J., Teusner, R., & Meinel, C. (2015, December). Towards practical programming exercises and automated assessment in Massive Open Online Courses. In *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 23-30). IEEE.
- Staubitz, T., Klement, H., Teusner, R., Renz, J., & Meinel, C. (2016, April). CodeOcean - A versatile platform for practical programming exercises in online environments. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 314-323). IEEE.
- Su, Y. S., Ding, T. J., & Lai, C. F. (2017). Analysis of students engagement and learning performance in a social community supported computer programming course. *Eurasia Journal of Mathematics, Science and Technology Education*, 13(9), 6189-6201.
- Sunar, A. S., White, S., Abdullah, N. A., & Davis, H. C. (2016). How learners' interactions sustain engagement: a MOOC case study. *IEEE Transactions on Learning Technologies*, 10(4), 475-487.
- Tang, T., Rixner, S., & Warren, J. (2014, March). An environment for learning interactive programming. In *45th ACM technical symposium on Computer science education* (pp. 671-676). ACM.
- Volchek, D., Romanov, A., & Mouromtsev, D. (2017, November). Towards the semantic MOOC: Extracting, enriching and interlinking e-learning data in open edX platform. In *International Conference on Knowledge Engineering and the Semantic Web* (pp. 295-305). Springer, Cham.
- XBlocks Directory (2020). Retrieved April 2020, from: <https://openedx.atlassian.net/wiki/spaces/COMM/pages/43385346/XBlocks+Directory>
- Yousef, A. M. F., Chatti, M. A., Schroeder, U., & Wosnitzer, M. (2014, July). What drives a successful MOOC? An empirical examination of criteria to assure design quality of MOOCs. In *2014 IEEE 14th International Conference on Advanced Learning Technologies* (pp. 44-48). IEEE.
- Wang, X., Yang, D., Wen, M., Koedinger, K., & Rosé, C. P. (2015, June). Investigating How Student's Cognitive Behavior in MOOC Discussion Forums Affect Learning Gains. In *2015 International Conference on Educational Data Mining (EDM)* (pp. 226-233). International Educational Data Mining Society.
- Warren, J., Rixner, S., Greiner, J., & Wong, S. (2014, March). Facilitating human interaction in an online programming course. In *45th ACM technical symposium on Computer science education* (pp. 665-670). ACM.
- Wilson, S., Daniel, F., Jugel, U., & Soi, S. (2011, June). Orchestrated user interface mashups using w3c widgets. In *International Conference on Web Engineering* (pp. 49-61). Springer, Berlin, Heidelberg.

Xu, B., & Yang, D. (2016). Motivation classification and grade prediction for MOOCs learners. *Computational intelligence and neuroscience*, 2016, 1-7.