Requirements for educational games in MOOCs

Manuel Freire, Iván Martínez-Ortiz, Pablo Moreno-Ger, Baltasar Fernández-Manjón
Dept. Software Engineering and Artificial Intelligence
Universidad Complutense de Madrid, Facultad de Informática
C/ Profesor Jose Garcia Santesmases, 9 28040 Madrid, Spain
{manuel.freire, imartinez, pablom, balta}@fdi.ucm.es

Abstract—**MOOCs are disrupting educational technology by democratizing access to high quality courses offered by some of the most prestigious universities. However, due to low entry cost and multiple other factors, MOOCs suffer from low retention rates: making MOOCs more interactive and engaging is therefore an important goal. Educational games provide environments that mix both immersion and quick feedback cycles, and we consider that games also have the potential to improve interaction and assessment in MOOCs environments; indeed, many game-like simulations have already been successfully applied within MOOCs in domains such as electronic circuits (schematic circuit simulator) and biology (foldit, eyewire). In this work, we analyze multiple such experiences and propose a catalogue of best practices that could contribute to the successful integration of educational games into MOOC platforms.**

Keywords—SGs, MOOCs, integration, best practices

I. Introduction

Large-scale e-learning, in the form of Massive Online Open Courses (MOOCs), is in the process of transforming education. MOOCs offer a new opportunity for democratising and simplify the access to high-quality long life continuous education. Platforms such as edX (www.edx.org), Coursera (www.coursera.org), FutureLearn (www.futurelearn.com), FUN (www.france-universite-numerique.fr) and Iversity (www.iversity.org), among others, are quickly gaining in participating institutions and student enrollments – requiring only an internet-connected PC from students.

Since students are not bound by physical location, tuition or contracts to the courses that they participate in, the market for students is highly competitive, both within and among platforms. Course and platform creators are therefore on the lookout for ways to differentiate themselves and increase the impact of their (highly-scalable) courses. The lack of student barriers also results in low retention rates, as students can easily decide to drop a course with minimal consequences. For example, course 2013's edX MOOC for 8.02x in Electricity and Magnetism found that only a 25.1% of its initially-active students took the final exam [1]. Retention rates for total registered students, regardless of activity level, are much lower, typically under 10% [1]–[3]

While many aspects can affect student retention and satisfaction, we consider that assessment is a key element. Now, most of the MOOC assignments are multiple choice, short answer or programming assignments that are graded automatically by the MOOC system or essays that are evaluated by peers. However, there are different problems because the kind of exercises that can be automatically graded is limited and peer evaluation requires extra effort for the students and could discourage retention in the MOOC. Therefore, we are interested in new types of highly-interactive simulations and games that have been used in MOOCs from the onset. EdX, with over 65 member institutions as of early 2015, has included specific support for the protein-folding game fold.it [4] exercise, along with circuit-schematics editor/simulator and a chemical-formula and protein-sequence editors. More recently, they have added access to the Eyewire crowd-sourced neuron mapping game. This game has already delivered, as of early 2015, more than 3.9 million cubes (each one a stack of brain images, which the player traces to reconstruct the 3D structure of the neuronal connections contained within), and lead very relevant results that has been published in high-impact scientific journals such as Nature. Eyewire has already announced that they plan to extend integration to other systems, using a public API and more delivery mechanisms to simplify integration of their serious game into different platforms (one assumes, MOOCs included).

The main goal of this paper is to analyze the issues and constraints that serious games and simulations face for inclusion as part of MOOC courses, as well as highlight recent trends in the area. The integration can be analyzed from different perspectives, in particular, the depth of the integration, the issues related to deployment and the technical perspective.

II. Integration options in Moocs

Although there are many competing MOOC platforms, few make their underlying source-code open for external inspection (and often improvement). In this section, we analyze the approaches towards complex activity integration found in two open-source MOOC platforms, edX (used by over 65 higher learning institutions worldwide) and OpenMOOC (with around 7 institutions as of early 2015).

A. *Integration options in edX*

edX supports a large number of different activity types and origins, most notably integrating LON-CAPA exercises and access to libraries of such exercises [5]. LON-CAPA exercises are themselves somewhat extensible, as it is possible to link to external servers for customized grading by using the "external-response" tag [6]. Some of these advanced exercise types are included in the default edX distribution, such as those involving circuits, protein sequence, or chemical molecule design. Extending LON-CAPA with new exercise types generally requires adding server-side code to allow these new types to be rendered in the student's browser; although edX's open-source nature and explicit support for extensibility (via the XBlocks mechanism) makes this easier than in other platforms, it still requires considerable expertise.

Once developed, XBlocks (both LON-CAPA related and otherwise) can be shared, to be installed by administrators in other edX deployments. Several such XBlocks are available on GitHub[1]. As server-side components, XBlocks have direct access to the students' profile, and can even include customized "authoring views" for use when their corresponding activities are added or modified within edX's Studio course-authoring tool. For example, Microsoft has developed an XBlock that simplifies the inclusion of OfficeMix lectures as edX activities[2]. Preliminary work exploring the inclusion of serious games as edX XBlocks has also been successful.

A lightweight alternative to adding functionality by extending XBlocks, currently in development, is the use of a specialized type of LON-CAPA exercises by using a new "custom-response" tag called JSInput[3]. Within this tag, external HTML pages (which can be made arbitrarily complex via JavaScript) can be rendered and evaluated. Although JSInput exercises cannot contain specialized authoring or server-side code, reuse of JSInput activities requires only copying and pasting snippets of XML code pointing to the relevant pages.

Another source of rich activities is through IMS Learning Tools Interoperability (LTI) support [7]. Currently, version 1.2 is supported in edX. The use of LTI is widespread in traditional Virtual Learning Environments such as Moodle or Sakai; and it is possible to package complex games and simulations as LTI. However, LTI expects communication between activity and MOOC to take place only at the start and end of the activity; and therefore, makes it difficult to receive and act on learning analytics data generated during rich activities such as games or simulations; this disadvantage is shared with use of JSInput LON-CAPA activities.

B. Integration options in OpenMOOC

OpenMOOC relies on simple HTML forms for exercises, and does not provide modular extension hooks. Adding exercise "nuggets" (as evaluated activities are termed in OpenMOOC) requires changing the MOOC server's source-code. Even then, the open-source nature of the project would simplify applying these changes to other OpenMOOC installations.

[1] https://github.com/edx/edx-platform/wiki/List-of-XBlocks

[2] https://github.com/OfficeDev/xblock-officemix

[3] http://edxpdrlab.readthedocs.org/en/latest/course_data_formats/jsinput.html

III. Serious Games in MOOCs

A. Integrating the game in the educational design

Regarding the course design, we consider that the most promising approach is to treat the serious game as a new type of activity or exercise. From an educational design perspective, the integration of a SG as an activity inside a MOOC can be simplified to answering the following questions:

1. Where do the students' download / play the game ?
2. Is it possible to extract students' score from the game?
3. Is it possible to extract students' progress during the gameplay?

To answer the first question, we need to analyze the nature of the SG, that is, the game platform (desktop, mobile/tablets, web based, etc.). MOOC course's activities are usually web based activities; however it is not always possible to have a web based game that fits the educational needs, in particular, when the course's author reuses a previously existing game. In addition, even when the SG is web based, the course's author must know/understand the technical details regarding the deployment of the game itself, that is, uploading the game and all the game assets to the MOOC platform and creating the entry web page where the game it is initialized. Both cases pose some difficulties for educators that do not have strong computer science skills.

Unlike other multimedia resources included inside a MOOC that must be complemented with a traditional questionnaire or a reflection activity to evaluate acquired knowledge, SGs are a valuable assessment tool in and of themselves [8], providing both frequent and immediate feedback to the player during gameplay paired with optional summary feedback, usually as a score, when the player finishes the game or reaches a milestone. This summary feedback can be a unique value or a defined set of values (e.g., score, time spent playing, actual time in particular game tasks, number of errors, etc.) that can be used to automatically evaluate the student and, thus, alleviate the workload of the MOOC's instructors. In order to use students' scores, the course's author must configure how these scores are sent back to the MOOC platform. This, poses two problems, first the game developer has to provide a means to integrate their games into a MOOC, that is, the game developer must know and develop a communication module for each MOOC platform, and second the course's author still needs to pair the SG gameplays with the course, again, requiring to have a high level of technical expertise.

Additionally, in-game evaluation can be performed without the student being aware of it (a practice known as "Stealth Assessment"[8]). Game-players frequently report [9] entering a state of flow [10], which implies optimal conditions for learning, but a traditional questionnaire can break the flow of learning. Implicit (or stealth) evaluation during game-play does not have such disruptive effects on the game's immersion, and is therefore highly valuable to preserve student motivation. Moreover, sometimes the final result (e.g. score) is not the most relevant assessment tool, but the continuous achievements and progress during the gameplay, that is, the assessment process takes into account not only the results but the timeline these achievements.

B. Game deployment

Since MOOCs are delivered via web to the student's browser, all MOOC activities can be split into a server-side portion and a client-side portion. The server-side portion of each activity is in charge of configuring and rendering the activity; and later on, recording the student's interactions with it. The client-side portion is in charge of actually presenting the activity to the student, and interacting with the student during its execution. Client-side interaction may or may not involve querying the activities' server-side component.

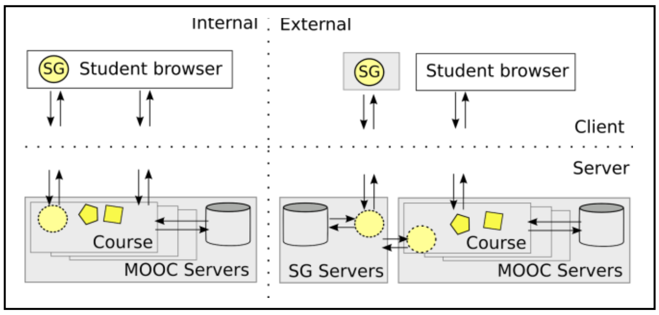

Fig. 1. Deployment models

The image depicts two major decisions for game-platform integration. In the left-hand side, the game-client and the SG server is integrated within the MOOC; in the right-hand side, the game-client and the SG server are external to the MOOC. As depicted by the vertical split, these decisions can be taken independently for both the game-client and the game-server, with different advantages and disadvantages for each.

An in-browser game client requires the game client to work in a browser; this restricts the range of possible languages and platforms to those that can be embedded into browsers, such as Flash or Java applets (available as browser plugins). Recent advances in web standards and scripting, and increased browser support for these standards (eg.: HTML5 with WebGL), as well as diminishing support for plugins, have driven adoption of HTML-only game clients.

On the other hand, stand-alone game clients allow the greatest freedom during game design, and potentially greater game performance in terms of speed, at the cost of having to build and distribute many more game versions: one for each supported platform such as Windows, Mac or Linux. With the increased availability of cheap computational devices and communications, access to many online resources cannot be presumed to be through a desktop computer with keyboard and mouse; support for mobile, touch-enabled devices such as Android tablets or Apple iPads adds additional platforms to support or target. The proliferation of platforms has made multiplatform game development (where a single game can be deployed to several platforms) a very appealing choice; for example, the Unity 3D cross-platform game development engine boasts a 45% market share as of early 2015 [11].

When the game's server-side component is hosted in the MOOC, the game behaves (from the MOOC's course author's point of view) as just another activity. An alternative is to use an external game server, and use a proxy to that server to send and gather information. Although this involves additional complexity in order to manage user sessions and access data, it is important in applications where significant functionality or data is only made available on request. This also allows game servers to track player performance internally, without needing to query the MOOC. Conversely, it is now the responsibility of the MOOC to use game server APIs to inquire on student performance and interactions after (or even during) their use of the game.

In the case of the edX integrations of EyeWire[4] and fold.it[5], both use external servers, allowing their clients to load puzzles on demand. In the case of EyeWire, puzzles are composed of layers of microscope images of brain tissue; while, for fold.it, puzzles require folding proteins into a stable lowest-energy state. While the EyeWire client is web-based, using WebGL for its graphics and Javascript for interaction, the fold.it client has higher computational demands, and is therefore external, with versions available for Windows, MacOS X and Linux platforms.

IV. Technical Issues in Serious Game Deployment

The previous perspectives, in turn, raise questions related to the technical feasibility of the different approaches, which take a greater relevance when the massive potential size of MOOC deployments is taken into account.

A. SG execution

The core SG's execution may be located either in the server, the client's computer or a mixed approach. The least taxing for the servers is (obviously) to deploy the game to the students' computers and run it there.

However, this typically requires the students to go through different installation procedures (e.g. a platform-dependent executable, as in the case of Fold.it) or to be able to support specific technologies and plugins in their browsers (e.g. a Java plugin for edX's LON-CAPA applets for exercises involving proteins and genes).

Both approaches typically impose additional technical requirements in those target computers, which they may or may not be able to meet, given the varied populations that typically participate in these courses.

In addition, and depending on the course's profile, this may also be a challenge in terms of assessment, since the game is run locally and the host system is unaware of how the students interact with the game.

For this reason, it also feasible to run the code in the server. The code is bundled in the MOOC infrastructure and the students' computers act as thin clients. This has the advantage of allowing a great degree of control and insight into how each

[4] https://eyewire.org

[5] http://fold.it/portal

TABLE I. BEST PRACTICES

Component dimension	Technical	Educational
game	• Support standardized packaging formats, providing search-friendly metadata. • Simplify integration into host courses and platforms.	• Use student profiles to customize learning experience. • Provide information on student actions and progress to the MOOC platform. • Allow authoring access to the educational content, so that course creators can tweak it (OER spirit).
course	• Integrate guest activity into general course progression provide activity with student profile, • Reflect in-game decisions in updated profile.	• Provide educational context for SG, both before and after. • Select the SG activities that best advance the educational goals of the course.
platform	• Clear APIs for guest activities. • Support several (incremental) integration levels.	• Collect and facilitate analysis of data from guests, allowing, for example, student leaderboards to be generated. • Support A/B testing of game variants.

student is playing, and often reduces the technical requirements to run the game. However, this is often impossible given the enormous communication overhead and the risk of vulnerabilities in the game code that could potentially affect the hosting server.

As a combination of those two ideas, the most interesting approach is probably a mixed scenario: games that have both a server and a client component for each exercise. The games then run on the students' computers, but send periodic updates to the server. The server can collect detailed information and gameplay traces for assessment and tracking purposes.

In this approach, all games could share the same server-side component, and communicate with the host following specific APIs common to all deployed games. This improves compatibility and helps in maintaining costs at a reasonable level.

B. *Setting up a communication channel*

When taking the mixed approach (with a game component in the client and another in the server), the next technical question to be solved is when and how the games are expected to communicate with the server.

The simplest and most common approach is to communicate at the end of the gameplay session, so that the game can report back information about the session for tracking and assessment purposes. However, it is also very common to have an initial communication as soon as the game is launched, in order to set communication parameters and a global configuration for the play-through.

In addition, especially when gathering detailed tracking information, deferring the transmission of results to the end of the play-through presents some issues: the student may not complete the playthrough (therefore loosing valuable intermediate tracking data) and the endgame data submissions may be too large. To alleviate this, the communication may contemplate intermediate submissions of partial data to be collected by the server.

C. *Game authoring and modification*

The development of the games also presents important technical barriers that require attention. Most game developments are closed products developed by third parties. But while the model is common, this makes games obsolete much earlier, since they cannot accommodate even minor updates.

This prompts for the creation of games that are easy to maintain and modify, therefore ensuring the future updatability of those games. However, this will often require easy-to-use authoring environments that do not present significant technical barriers when it comes to making changes in the game.

Ideally, the games should fit on the requirement of the Open Education Resources movement's "4 Rs": reuse, redistribute, revise, remix [12]. In this way, the investment is better protected against future changes, the community benefits from previous works and the costs can be driven down.

When taken to the extreme, the idea of "easy to create, easy to modify" games also opens the door for further models such as those identified by Copper et al. [4], who detected the preference of users towards user-created playing tools on Fold.it.

V. BEST PRACTICES

Based on our previous analysis, we propose the following catalogue of best practices (see Table I). We subdivide the responsibilities into three components, game, course and platform.

Usability issues and the deployment models described in section III, and the technical difficulties described in section IV are not completely new in the e-learning field, the have a close relation to the very same problems that affected Learning Management Systems (LMS) years ago.

To alleviate the aforementioned limitations in the context of LMS, there were proposed different e-learning standards and specifications, specifically, del Blanco et al. [13] argue for 3 types of integration of SGs within a LMS. Although MOOC platforms are not LMS and hence the very same specifications may not be applicable (or even not desirable) it is clear that it is needed a common set of agreements between game developers and MOOC developers in order to take the most advantage assessment features of SGs. In addition, this agreements will led to the simplification of the authoring perspective (hiding most of technical details for the end user), fostering the reutilization of the SG and allowing the tailoring of the SG to the student profile.

SGs is that can generate a detailed set of the student performance data that can be collected inside the MOOC just to track student progress, for example, in-game decision can lead to the issuing of a badge that will be reflected in the user profile, or to suggest additional activities (or even other MOOC courses).

MOOC platform developers are usually focused on the usability and scalability of the core modules of the platform. Still having a clear defined integration model for third party activities and providing incremental levels of integration can attract SG developers and thus attracting educators.

VI. Conclusions and Future work

According to the latest editions of the NMC Horizon Report, MOOCs and Serious Games are two of the most relevant trends in higher education, with engagement rates being the most relevant hindering factor in the former [14], [15]. The potential synergies for both are significant, with MOOCs being an ideal platform for serious games deployment and serious games being an ideal medium to increase engagement rates.

However, the challenges are also significant: serious games often need to tackle deployment issues even when applied on a small stage, and the scale of MOOCs requires especial attention to this issue. In contrast, the other common issue in serious games initiatives is the excessive development cost. In this case, the scale is a favorable factor, since a single game can reach wider audiences, therefore improving the potential return on the investment. When developed (and deployed) with adequate care to interoperability factors, the investment is further protected by allowing deployment across different courses and platforms.

In this work we have analyzed different experiences in simulators and serious games used as MOOC activities, as well as the most common technical issues that these deployments face. From this analysis, we have proposed a set of best practices and a brief overview of the current standardization landscape at the intersection of serious games and MOOCs.

Given the comparatively short history of MOOCs as a medium, platform maturity is much lower than that of traditional VLEs, and considerable research is still necessary to confirm our initial findings. However, given the numbers involved in MOOCs and the rate of growth of current deployments, the future is exciting and the wait for new initiatives data will not be long.

Acknowledgment

The e-UCM research group has been partially funded by Regional Government of Madrid (eMadrid S2009/TIC-1650, S2013/ICE-2715), by the Ministry of Education (TIN2010-21735-C02-02, TIN2013-46149-C2-1-R), by the RIURE Network (CYTED 513RT0471) and by the European Commission (SEGAN 519332-LLP-1-2011-1-PT-KA3-KA3NW and RAGE H2020-ICT-2014-1-644187).

References

[1] S. Rayyan, D. T. Seaton, J. Belcher, D. E. Pritchard, and I. Chuang, "Participation And performance In 8 . 02x Electricity And Magnetism□: The First Physics MOOC From MITx," *Phys. Educ.*, vol. 11 October, 2013.

[2] D. Seaton, Y. Bergner, I. Chuang, P. Mitros, and D. Pritchard, "Towards Real-Time Analytics in MOOCs," *ceur-ws.org*, 2013. [Online]. Available: http://ceur-ws.org/Vol-985/paper3.pdf.

[3] S. Haggard, Centre for Distance Education, and Observatory on Borderless Higher Education, "The Maturing of the MOOC: Literature Review of Massive Open Online Courses and Other Forms of Online Distance Learning," *BIS Res. Pap. 130*, no. 130, pp. 1–122, 2013.

[4] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players, "Predicting protein structures with a multiplayer online game.," *Nature*, vol. 466, no. 7307, pp. 756–760, 2010.

[5] G. Kortemeyer, W. Bauer, D. Kashy, E. Kashy, and C. Speier, "The LearningOnline Network with CAPA Initiative," *31st Annu. Front. Educ. Conf. Impact Eng. Sci. Educ. Conf. Proc. (Cat. No.01CH37193)*, vol. 2, 2001.

[6] LON-CAPA Group, "LON-CAPA Author's Tutorial and Manual," 2014. [Online]. Available: https://s10.lite.msu.edu/adm/help/author.manual.pdf.

[7] IMS Global Consortium, "Learning Tools Interoperability Implementation Guide Final Version 1.1," 2012. [Online]. Available: http://www.imsglobal.org/LTI/v1p1/ltiIMGv1p1.html.

[8] V. J. Shute, "Stealth Assessment in Computer-Based Games to Support Learning," in *Computer Games and Instruction*, S. Tobias and J. D. Fletcher, Eds. Information Age Publishers, 2011, pp. 503–523.

[9] B. Hoffman and L. Nadelson, "Motivational engagement and video gaming: A mixed methods study," *Educ. Technol. Res. Dev.*, vol. 58, no. 3, pp. 245–270, 2010.

[10] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. New York: Harper and Row, 1990.

[11] Unity 3D, "Unity 3D Fast Facts," 2015. [Online]. Available: http://unity3d.com/public-relations.

[12] J. Hilton, D. Wiley, J. Stein, A. Johnson, and J. Hilton III, "The four 'R's of openness and ALMS analysis: frameworks for open educational resources," *Open Learn. J. Open Distance Learn.*, vol. 25, no. 1, pp. 37–44, 2010.

[13] Á. del Blanco, E. J. Marchiori, J. Torrente, I. Martínez-Ortiz, and B. Fernández-Manjón, "Using e-learning standards in educational video games," *Comput. Stand. Interfaces*, vol. 36, no. 1, pp. 178–187, Nov. 2013.

[14] L. Johnson, S. Adams Becker, M. Cummins, V. Estrada, A. Freeman, and H. Ludgate, "NMC Horizon Report: 2013 Higher Education Edition," Austin, Texas, USA, Texas, USA, 2013.

[15] L. Johnson, S. Adams Becker, V. Estrada, and A. Freeman, "NMC Horizon Report: 2014 Higher Education Edition," Austin, Texas, USA, 2014.