Accepted for publication on Personal and Ubiquitous Computing, DOI: 10.1007_s00779-017-1063-8

Integration scenarios of virtual worlds in Learning Management Systems using the MULTIS approach

Leonel Morgado
INESC TEC, CIAC, LEAD, and Universidade Aberta
Coimbra, Portugal
leonel.morgado@uab.pt
+351 22 209 4199
ORCID: 0000-0001-5517-644X

Hugo Paredes, Benjamim Fonseca, Paulo Martins
INESC TEC and UTAD, Universidade de Trás-os-Montes e Alto Douro
Vila Real, Portugal
ORCID: 0000-0002-4274-4783, 0000-0002-0850-9755, 0000-0002-3040-9080

Álvaro Almeida, Andreas Vilela, Bruno Pires, Márcio Cardoso
UTAD, Universidade de Trás-os-Montes e Alto Douro
Vila Real, Portugal, Country

Filipe Peixinho, Arnaldo Santos
Altice Labs
Aveiro, Portugal

***Abstract*— This work further clarifies how the MULTIS architecture can be used for integration of virtual worlds in Learning Management Systems (LMS) for organizational management of e-learning activities, as an extension to a previous work published in the proceedings of VEAI 2016. Current LMS provide minimal support for educational use in an organizational context, and other integration efforts assume educators are inside the virtual world, accessing the LMS as an external service. Our approach enables educators to setup and manage virtual world activities from within the traditional LMS Web interface as an integral part of the overall educational activities of a course. The MULTIS architecture foresees several alternative communication channels between LMS and virtual worlds, including the spooling of automated clients or "bots", and the flexibility to inject code if necessary and possible. In this work, we detail the application of this architecture and its approach in several sample scenarios, based on previous analysis of integration requirements. It is the result of a joint effort by academic and corporate teams, implemented and tested in the Formare LMS for OpenSimulator and Second Life Grid virtual world platforms.**

Keywords- LMS; virtual worlds; integration; OpenSim; OpenSimulator; Second Life

1. INTRODUCTION

Lack of integration between carrying out and setting up/managing educational activities in virtual worlds is a factor hampering the widespread deployment of virtual worlds in education and training [1]. To set up and manage the activities in these environments, teacher/trainers must deal with an encumbering variety of administrative and technical tasks: login credentials managed separately; having the teacher/trainer setup trainee/student tracking in the virtual world, and then link it back to the learning management system (LMS); object repositories need to be managed in the virtual world by teachers/trainers with no connection with other learning materials stored in the LMS; the list goes on and on.

Our perspective is that this current status of virtual worlds' use in education and training sees each teacher/trainer as an island, isolated from modern organizational information systems and support services. That is, each teacher/trainer currently needs to technically set up the virtual world space and its activities and then sort out any

Acknowledgment
Partly funded by Altice Labs (MULTIS II project) and by Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020", in the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

connection plug-ins with other systems. Either that or at least worry about having a technical team to do all these tasks. And technical teams within education/training institutions may see that as an offshoot: even for technicians, virtual world management is not as streamlined as one would expect from a core operational information system.

Our approach inverts this perspective: teachers/trainers should be able to specify and supervise educational activities in virtual worlds as a seamless part of their overall teaching-learning plan, without the need for custom technical interventions. Just as they do, in fact, in traditional e-learning platforms: teachers/trainers concerns should be with the educational/training content and dynamics, not with the technical/computational issues. To realize this perspective, we conducted a software engineering research effort, using as a prototype the integration of Second Life Grid (SLG) [2] and OpenSimulator (OpenSim) [3] platforms into a corporate-oriented LMS system, Formare [4].

In this work, we demonstrate the use of the software architecture named MULTIS [5], which we developed for this purpose, by providing operational details. This architecture foresees several alternative communication channels between LMS and virtual worlds, including the spooling of automated clients or "bots", and the flexibility to inject code if necessary and possible. The evaluation of this architecture is then performed using illustrative scenarios, one of the traditional and most-employed methods in design science research [6]. I.e., we apply our artifact (the MULTIS architecture) to several real-world situations, to illustrate its suitability and utility.

The architecture and prototype were developed for SLG/OpenSim virtual worlds and the Formare LMS, but the overall approach holds the potential for expanding as a generic approach to integrate virtual worlds/serious games with LMS platforms in general. This work is an extension to an earlier one [7], adding considerable new details to the final section, and presenting several cases where the problem/solution vectors are employed to implement integration features.

2. Related work

This section is structured as follows: first we present the background on integration requirements, establishing a panorama of what features and functionalities should be accomplished in such a process; then we summarize the achievements of previous integration attempts, explaining their approaches and limitations.

2.1. Integration requirements

While the literature provides extensive documentation of virtual world use in education and training [8], typically only the actual virtual world activities are reported. There are some integration efforts, summarized in the next section, but few actual accounts or surveys of integration requirements with other educational systems, including LMS [9]. We recently determined a set of such requirements, by conducting content and thematic analysis of documents produced during four years of cooperation between academia and industry (e-learning provider) [9]. The results were 39 requirements and 54 sub-requirements, each supported by individual instances of documentation that originated it, and organized under 9 categories and several subcategories (ibid.). In Table 1 we present these categories, for the convenience of the reader.

Table 1 – Integration requirements – Categories [9]

Cat.	Subct.	Description
C1	-	Privacy of training sessions
C2	-	Record and replay behaviors of actors and other elements
''	C2.1	Recording the full events of a 3D session or generic 3D space
''	C2.1.1	Recording the behaviors of other elements
''	C2.2	Replaying the full events of a 3D session
''	C2.2.1	Replay the events in 3D
''	C2.2.2	Replay the events in 2D
C3	-	Support for virtual world content development
''	C3.1	3D space features manageable independently
''	C3.2	Support for at least 31 concurrent users
C4	-	Automated support for Administration
''	C4.1	Automated support for the administrative flow
''	C4.1.1	Tools/methods to track deployment & user adoption
''	C4.2	Federated authentication, LMS/virtual world platforms

Cat.	Subct.	Description
‘’	C4.2.1	LMS users may use preexistent SL/OpenSim usernames
C5	-	Automated support for trainers and trainees
‘’	C5.1	Specific-purpose applications to support trainers and trainees
‘’	C5.2	Trainer should have control over trainee's audio
‘’	C5.3	Orientation support for trainees
‘’	C5.4	Ability to manage access to interaction with 3D objects
‘’	C5.5	Alternative avatar appearance identification features
‘’	C5.6	Support for training about the use of virtual worlds
C6	-	Access to the LMS data and services in the 3D space
C7	-	Integration of virtual world data in the LMS
‘’	C7.1	LMS accepts choreographies provided by trainees or trainers
‘’	C7.2	LMS accepts 3D models provided by trainees or trainers
‘’	C7.3	Ability to annotate the raw data from a session recording
C8	-	LMS must be the source of control and management over educational activities in virtual worlds
C9	-	Alternatives for voice communication in the 3D platform

2.2. INTEGRATION EFFORTS

Possibly the best-known effort for integration of SL/OpenSim virtual worlds and LMS is the SLOODLE project [10], which employed Moodle [11] as the focus LMS platform, and has over 10 years of development, providing a large array of integration features and tools [12]. But there have also been other, short-lived projects. For instance, the BbSL project, developed in 2008-2009, aimed to “manage, administrate and facilitate any hybrid Second Life / Blackboard Learn instructional experience” [13]. Another examples was the special-purpose LMS called Vushi, whose website was active between 2010-2012 (acc. Internet Archive Wayback Machine, [14]), but some of its features and operation can still be seen in its YouTube channel, which remains available at the time of writing of this document [15].

Other efforts focused on integration of specific issues, rather than overall systems integration. For instance, tracking attendance [16], orchestrating avatar choreographies [17], or responding to control requests issued from virtual world scripts [18]. In this regard, there are similarities with the field of remote physical laboratories, where efforts have been made to orchestrate collaboration [19] and current concerns include federating authentication and conducting laboratory tasks under control of the LMS [20].

Both SLOODLE and the shorter-lived BbSL and Vushi projects have in common their perspective of the role of the teacher/trainer: it is centered on the virtual worlds, not on the LMS. Specifically, the trainer needs to set up tools and features within the virtual world and these can then be linked to the LMS, to access the information stored there. For example, these are SLOODLE’s instructions for using its presenter tool [21]:

“*To setup the presenter:*

1. *Create a Presenter Activity on your moodle website*
2. *Create your slideshow, by adding images, or webpages, or videos as links in the bottom section*
3. *once complete, rez a presenter in Second Life*
4. *Click on it, to authenticate it with your moodle website*
5. *Once its* (sic) *been authenticated, click on the presenter again, to download the saved configuration*
6. *If the land is owned by a group, you must deed the presenter to that group*
7. *Once the Presenter has been deeded, it will automatically* (…) *after 5 minutes. If you don't want to wait 5 minutes for it to check if it has been deeded or not, you can just touch the Presenter for it to jump to the next step.*
8. *Now your presenter is ready.*
9. *Press play on your media settings.*
10. *You should now be able to see your presentation*”

All steps between 3 and 7 are in effect technical setup issues of the virtual world platform that the trainer should not have to deal with, since they take up time and effort from the educational activities. This perspective on integration sees the virtual world platform as the central point for managing the educational activities, with the LMS being a secondary external service,

hence we call it virtual-world-centric. We propose adopting an LMS-centric perspective, where the LMS is the central point for managing educational activities, of which the virtual world is but one location. Following this perspective, a presentation activity, once created on the LMS, should be ready and available for the trainer/teacher and trainees/students to use within the virtual world. This reflects several requirements' categories of Table 1, such as categories C5-C8.

Our approach enables the LMS to go inside a virtual world platform and set up actions that trainers typically must do themselves when using other current approaches (e.g, SLOODLE). This could be achieved by custom development, but that would couple the LMS code with the code of the virtual world platform. The coupling would tie the LMS to a customized version of a virtual world platform, rather than enable it to keep up with the ever-changing diversity of platforms that have emerged and are emerging regularly. Therefore, our approach, dubbed MULTIS (the name of the project where it was created), takes advantage of the fact that virtual worlds have a core feature: they enable remote users to login and interact with the world. While in game-oriented worlds this interaction can be quite limited, in creation-oriented worlds such as those supported by SLG or OpenSim platforms it can achieve most of the necessary setup tasks. Instead of laying these tasks upon the users, as in SLOODLE, BbSL, Vushi or other systems, the MULTIS software architecture enables LMS systems to spool a pool of automated clients, known as 'bots', to perform the tasks a user would, and combine this with other more efficient communication and control channels, if available. Trainers/trainees use the LMS interface and the LMS can use this MULTIS bot-spooling approach to conduct any associated setup tasks. This software architecture, particularly its bot-spooling approach, was presented in an earlier work [5], and is summarized in section III.A.

3. The MULTIS approach

3.1. Overall architecture

As mentioned above, the bot-spooling approach of the MULTIS architecture aims to allow integration of LMS systems with a large diversity of virtual world platforms and serious games, without requiring custom development on the virtual world side. That is, to enable integration even if virtual worlds and serious games do not provide application programming interfaces (APIs) or other services for external systems. As we put forward in our seminal work on the MULTIS architecture, "any online virtual world platform needs to provide login systems for clients. Thus, an LMS system can log into the virtual world platform (…) using automated clients" [11], typically known as 'bots'. To avoid bottlenecking systems integration through a single bot, we then introduced the bot-spooling approach.

Fig 1 presents the MULTIS architecture. The general LMS functions are represented by the "LMS logic" module. Any LMS that wishes to implement this architecture needs to provide and/or use the services associated with this module (in our current implementation, we employed the Formare LMS). When a virtual world task needs to be performed, this module sends a request to the "Bot scheduler" module, which chooses an appropriate bot. The request is then converted into bot commands, issued to the "Bot logic" module, which keeps track of running code threads and current open connections with the virtual world for controlling the 'bot' avatars. The bot avatars are represented in the figure as the "Avatar/Bot" module inside the virtual world server. The bot logic module then carries out the commands with the bot chosen by the scheduler module.

While bots enable the MULTIS architecture to circumvent the lack of an API, the architecture can also take advantage of the existence of such an API or some other level of interconnection services provided by the virtual world platforms. Some platforms even allow users or administrators to provide code add-ons that can communicate with external systems. For instance, in SLG and OpenSim, end-users can provide scripts for virtual world elements with Web-based communications

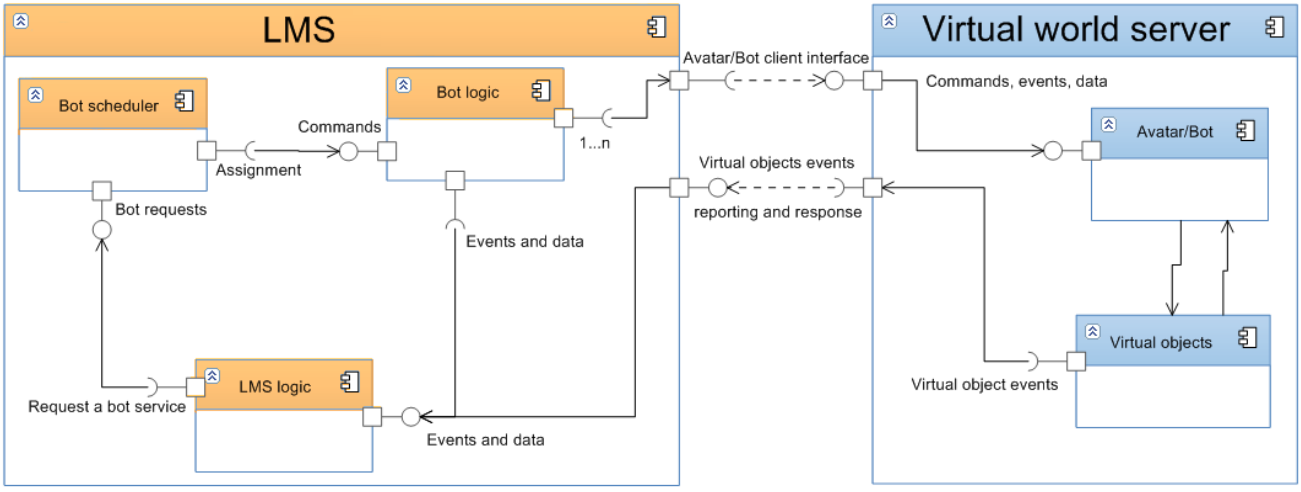

Fig. 1 – The MULTIS architecture [5]

capabilities. For this reason, the MULTIS architecture foresees interfaces both for bot-mediated actions and for other forms of interconnection (Fig. 1). The following section explains how to use this architecture.

3.2. THE FOUR PROBLEM/SOLUTION VECTORS

Using the MULTIS architecture, we addressed the requirements presented in section 2.1 using four problem/solution vectors, which we describe briefly. While the problems are generic, the proposed solutions are based on SLG/OpenSim virtual worlds.

Problem Vector 1: storage of proprietary virtual world content in the LMS. Nowadays there are quasi-interchangeable data formats for content such as 3D models and skeletal poses, but often platforms also employ proprietary data formats, such as SLG/OpenSim link sets (i.e., groups of objects), scripted objects, avatar clothing, and more. **Solution:** we employ automated avatars as data stores for the LMS. The LMS has the credentials for these avatars and controls them programmatically to receive or pickup proprietary formats.

Problem Vector 2: placing LMS-stored virtual world content in the virtual world. Once the proprietary content is within the bot data store, it needs to be placed in the world for users. **Solution:** from problem-solution vector 1, bots act as data stores for the LMS. When necessary, commands are issued to the bot spool as requests, and assigned to a bot, which will log into the virtual world and place the requested content. If necessary, bots being part of the data store exchange content among themselves programmatically.

Problem Vector 3: receiving virtual world events and data in the LMS. To log data, respond to events and in general update the system status, the LMS must receive notification of events and collect data. **Solution:** we deployed a two-pronged approach. When the speed of data collection or event reporting isn't critical, we use scripts in objects reporting data and events to Web services in the LMS. These scripts can be part of user interaction objects (placed using the vector 2 solution), part of specific invisible objects (id.) or injected into objects as necessary (see vector 4 solution, ahead). When data collection requires more timely responses, the LMS assigns a data collection request to the bot spool (see vector 2 solution). When virtual world platforms provide APIs, their services can be chosen by timeliness and combined with these two approaches.

Problem Vector 4: use the LMS to control the behavior of the virtual world. Following the requirements list [9], there is a plethora of situations were settings need to be adjusted, be it creating/deleting private voice chat groups, changing training room features, changing interaction permissions, resetting tools, etc. **Solution:** the LMS issues the necessary tasks to the bot spool, which then employs the bots to achieve them. This includes adding objects, injecting scripts, issuing private channel parameters and more. A complementary approach when timing is flexible is to have some scripts issue events to LMS Web services keeping the simulation/class state, and decide on necessary outcomes. If these are achievable through parameter passing, the LMS Web services respond with those, and the response can also include commands to be relayed to other scripts. This is the Pinheiro et al. method [18].

4. USING THE PROBLEM/SOLUTION VECTORS TO IMPLEMENT SPECIFIC FEATURES

The list of requirements [9] is quite extensive, so we selected three cases to illustrate how the problem-solution vectors were used to implement specific features. The core concern was keeping the LMS Web interface at the helm of decision-making: the place where trainers go for control of the virtual world sessions and activities. In Table 2, we summarize how each case demonstrates the use of the solution vectors.

TABLE 2 – SAMPLE CASES AND SOLUTION VECTORS EMPLOYED

Case	Requirement	Solution Vectors
1	Training session space features specifiable on creation	Solution Vector 1 Solution Vector 2
2	3D objects should have user role-based permissions	Solution Vector 3
3	Recording actors' behaviors as a 3D choreography	Solution Vector 4

4.1. CASE 1: REQUIREMENT R2F-2 (IBID.), "TRAINING SESSION SPACE FEATURES SPECIFIABLE ON CREATION"

In this case, 'features' can be simple items, such as chairs, or interactive elements, such as voting booths, simulators or games. One such element is a presenter tool for virtual world slideshows, which we will use in this work for clarity, so that readers can compare it with the operation of the SLOODLE presenter tool, described in section 2.1. Figure 2 shows two such spaces we created. Both of which have slideshow panels on the right side.

Fig. 2 – Training spaces with simple items and interactive elements [6]

Typically, virtual world slideshow presenters employ a pair of interactive objects: a control board and a display. Instead of having the trainer or a support person create them manually in the virtual world, we used solution vector 1 and stored these objects in the LMS, in the bot data store. When a trainer specifies in the LMS Web interface that a slideshow presenter is required for a session, we use solution vector 2: before the session starts, two requests are issued by the LMS to its bot spool, one for the placement of each object. The LMS knows the virtual world coordinates of the session space, since it was also created by the LMS when the teacher/trainer requested it in the LMS Web interface. Therefore, to create the slideshow presenter, the LMS logic includes the coordinates in the requests it issues to its bot spool. The LMS bot spool logic then selects the bots, logs them into the virtual world platform, and accomplishes the setup of the slideshow display and control board.

To clarify, in Figures 3 and 4, we present the sequence of execution of both solution vectors for this case. We are using OpenSim/SLG parlance and specifics, but this could take place in other platforms, by resorting to their specific approaches to providing and transmitting virtual objects and triggering scripts.

It all starts when the developer of a virtual slideshow presenter (the set of two objects, board and display) wants to provide it to the LMS, to render it available for users. This is the part that is solved using solution vector 1. As shown in Figure 3, first the developer accesses a virtual world location managed by the LMS and "drops" the slideshow presenter set into a preexisting object which was scripted for accepting deliveries: we call it an "object delivery station". The script inside this virtual object reports the event to the LMS logic, typically by calling a Web service. The LMS logic responds by asking the bot scheduler to collect the new item into the bot store. To do this, the bot scheduler assigns this task to a bot, whose logic logs into the virtual world, accesses the delivery station and collects the slideshow presenter set dropped within. Now that the bot has it in its inventory, the successful collection (and its data descriptors) is reported to the bot scheduler, which in turn reports to the LMS logic. The slideshow presenter set is now part of the LMS data store and can be accessed by users via the Web or the virtual world. One such case of Web-based use is shown at the bottom of Figure 3: the developer can access the LMS Web interface and check that the virtual slideshow presenter set is indeed there, possibly providing extra information and metadata.

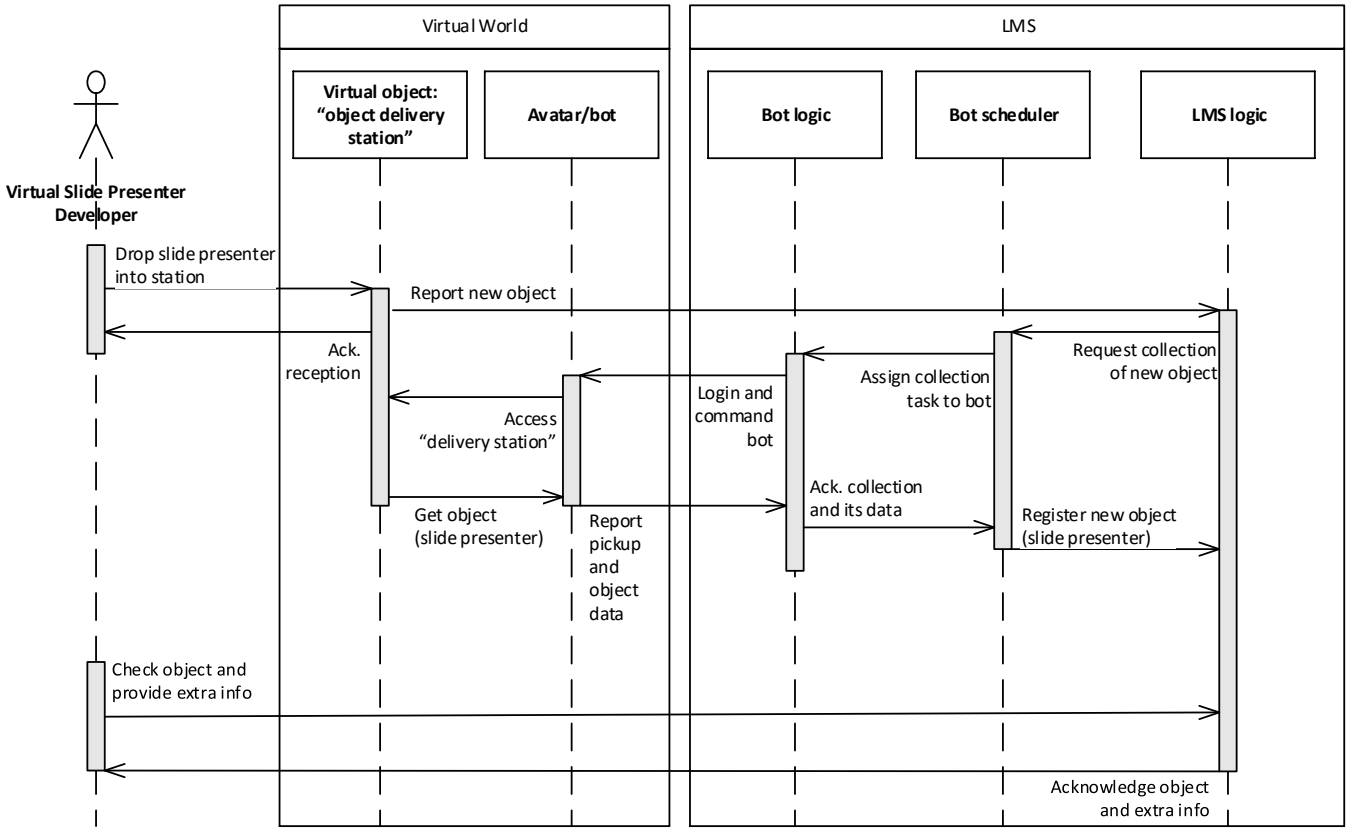

Fig. 3 – Sequence diagram for solution vector 1 applied to Case 1

Then another user (a trainer or training designer) will want to use the slideshow presenter set in a training session, inside the virtual world. This is the part solved using solution vector 2. As shown in Figure 4, this user will access the Web-based list of available training tools while setting up a virtual training session. From that list, the user will select the slideshow presenter set and provides any required settings (e.g., which presentation will be displayed, which users can control it, where it should be located in the virtual training room, etc.). The LMS logic proceeds by asking the bot scheduler to place the slideshow presenter into the virtual training room. To do this, the bot scheduler assigns this task to the bot whose inventory holds the presenter (or, if necessary for better scheduling, orders the transfer of inventory between bots). The bot logic logs into the virtual world, accesses the desired training room location and places the presenter within it, in the specified location. Once any setup-up scripts run in the presenter and its setup is complete, the successful deployment is reported to the bot scheduler, which in turn reports it to the LMS logic. The slideshow presenter set is now available in the virtual world and can be access by virtual training users. One such case of Web-based use is shown at the bottom of Figure 4: the trainer/training designer enters the virtual worlds and uses the presenter.

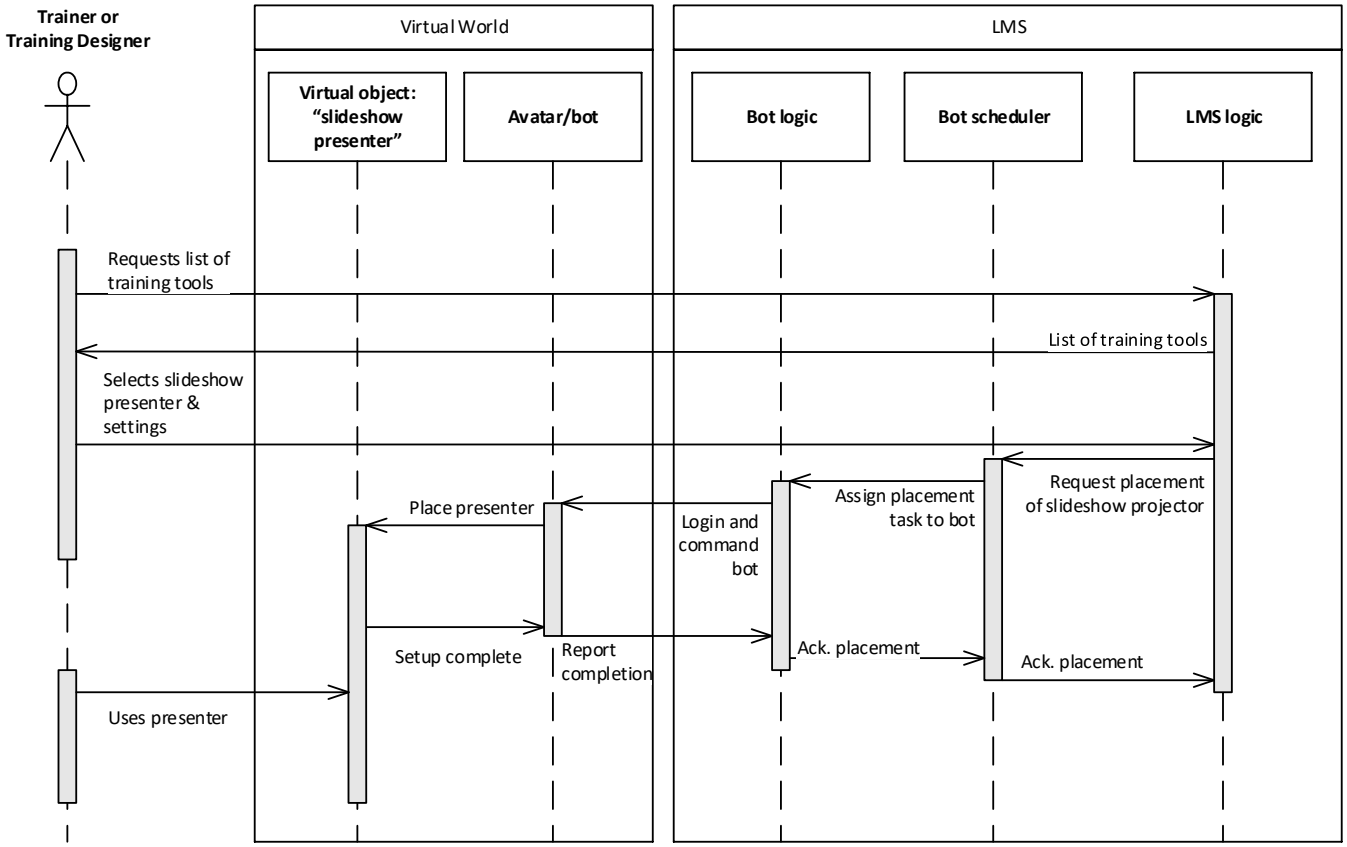

Fig. 4 – Sequence diagram for solution vector 2 applied to Case 1

4.2. CASE 2: REQUIREMENT R2G-6 (IBID.), "3D OBJECTS SHOULD HAVE USER ROLE-BASED PERMISSIONS"

In this case, 'permissions' are typically related to the kind of interactions that a participant can have with a 3D object. One such object might be the slideshow control board of the previous case. Suppose it may only be controlled in the virtual world by the trainer or by a student that is currently presenting. Roles are an LMS-specific logic (trainer, student, presenter, etc.), independent from the actual virtual world platform. The trainer edits the roles for a session in the Web interface, and the virtual world behavior must comply.

Figure 5 presents part of such a LMS Web page: each line deals with a different user. We cropped usernames, but the center column (heading: "Perfil", meaning "profile") identifies their roles as Tutor or student ("Aluno") for a course. For a given virtual world session, the Status column allows the trainer to specify each user's role for that session, and likewise for specific objects: Moderator, Participant, or Non-Participant ("Não Participa[nte]"). The remaining buttons are for convenience (e.g., "todos moderam", meaning "all can moderate") or for other features not discussed here (e.g., "repor avatar", meaning "reset avatar").

Fig. 5 – LMS Web page details for managing roles in the virtual space, translations are provided in the text [6]

In this case, we used solution vector 3: the interaction script in the slideshow control board, when touched by a user's avatar (trainer, trainee, etc.), issues an event to a Web service in the LMS, which matches the avatar ID with the LMS user ID, decides whether that user has the adequate role to use the board and responds accordingly (allowing or disallowing the interaction). The board script then uses the response to ignore the usage attempt or to respond to it. This sequence of execution is shown in Figure 6.

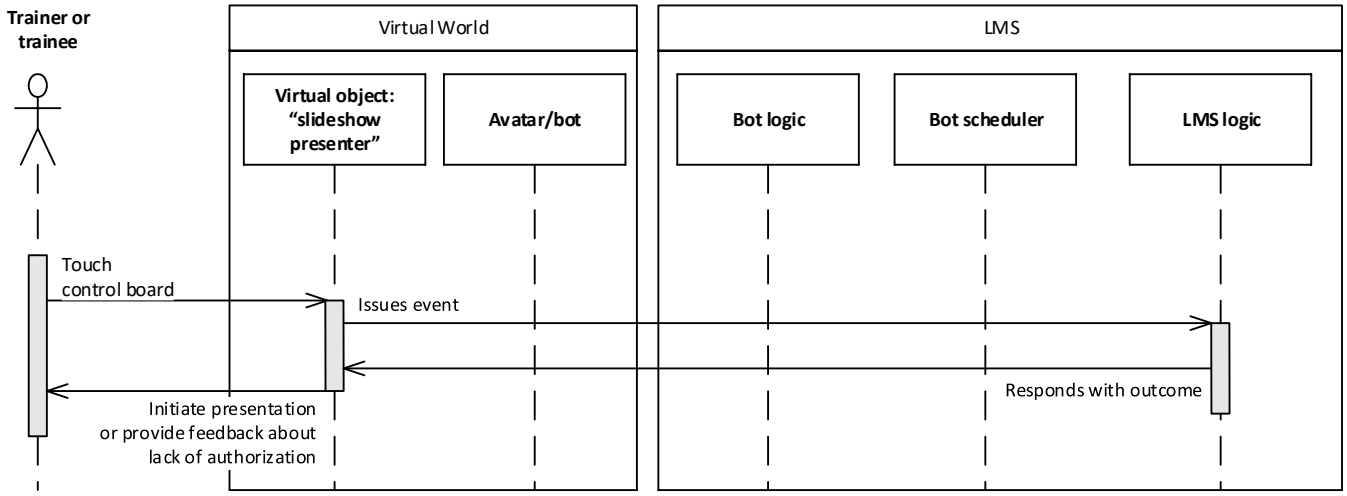

Fig. 6 – Sequence diagram for solution vector 3 applied to Case 2

4.3. CASE 3: REQUIREMENT R2A (IBID.), "RECORDING ACTORS' BEHAVIORS AS A 3D CHOREOGRAPHY"

In this context, 'actors' are any participants in a virtual world training session or class, regardless of whether they are human-controlled or computer-controlled. And 'choreography', in this case, is the set of actions that participants have performed, including not only their motions, but their conversation, their interaction with virtual elements, and other aspects, such as facial demeanor, body gestures, etc. For instance, supposing one is conducting a training session with a role-play situation, the recording of actors' behaviors as a choreography should enable a later review of the performance and events, supporting reflecting learning approaches per methods such as after-action review [22].

This can be achieved by employing a combinations of data collection methods. Silva et al.'s approach [17] is striving to make this independent from the LMS and other information systems, by describing these methods in ontologies and creating a managing system separating the LMS from the concerns of data acquisition. But this separation is beyond the scope of the current document. Here we demonstrate how the LMS can be integrated with specific platforms – OpenSim or SLG – enabling it to operate without end user intervention.

In OpenSim or SLG, choreography data can be collected by a combination of methods: scripts in objects can detect interaction and report it as event to LMS Web services (employing solution vector 3) or even detect the presence and location of user avatars, if timing isn't critical (e.g., for tracking attendance [16]). LMS-controlled bots can collect live data about the behavior of other avatars, such as users, for more time-dependent cases. In both cases, we are employing solution vector 4. Since the LMS manages the session and the virtual space, it knows which objects are present. When the trainer requests the recording of a choreography, the LMS can use both approaches. For instance, it can issue requests to the bot spool to inject an event-reporting script into each object or to use bots to collect live data. When the teacher/trainer requests that choreography recording stops, the LMS again issues the necessary requests to its bot spool: deleting the injected scripts and logging out the data-collection bots.

Notice that the same solution vector 4 can also be used to setup live data collection bots adequately. For instance, "dressing" bots as invisible avatars, or locating them beyond the visual reach of default user cameras (but still within tracking distance) [23]. Or conversely, by dressing them as video-recording operators or as "You are being recorded" signs.

To clarify, in Figure 7 we detail the setup and beginning of choreography recording: it all starts when a trainer specifies, as part of setting up a training session, that choreography recording should be available. The LMS asks the bot spooler to place recording features in a virtual training space. If these are scripted objects or scripts for injecting into pre-existing objects, solution vector 2 can be used (top half of Figure 7): the bot spooler assigns the placement of injection to a bot, which carries out the task. When the time comes for choreography recording, the scripted objects or injected scripts start sending live data to the LMS logic.

However, if the "recording feature" is a more time-critical data collection by a bot, the bottom half of Figure 7 comes into play, using solution vector 4: at the assigned moment for choreography recording, the bot spooler assigns a bot to perform live data collection; its logic performs the log in and commands for live data collection, feeding it to the LMS logic.

The exact moment when choreography recording starts is something the trainer or training designer can specify: For instance, it could be that an entire training session needs to be recorded, in which case choreography recording can be initiated at a set time. Or it could be that the recording is initiated by interaction with a specific virtual object, in which case solution vector 3 (Figure 6) is used to provide the necessary event, which initiates the activities in solution vector 4 (Figure 7).

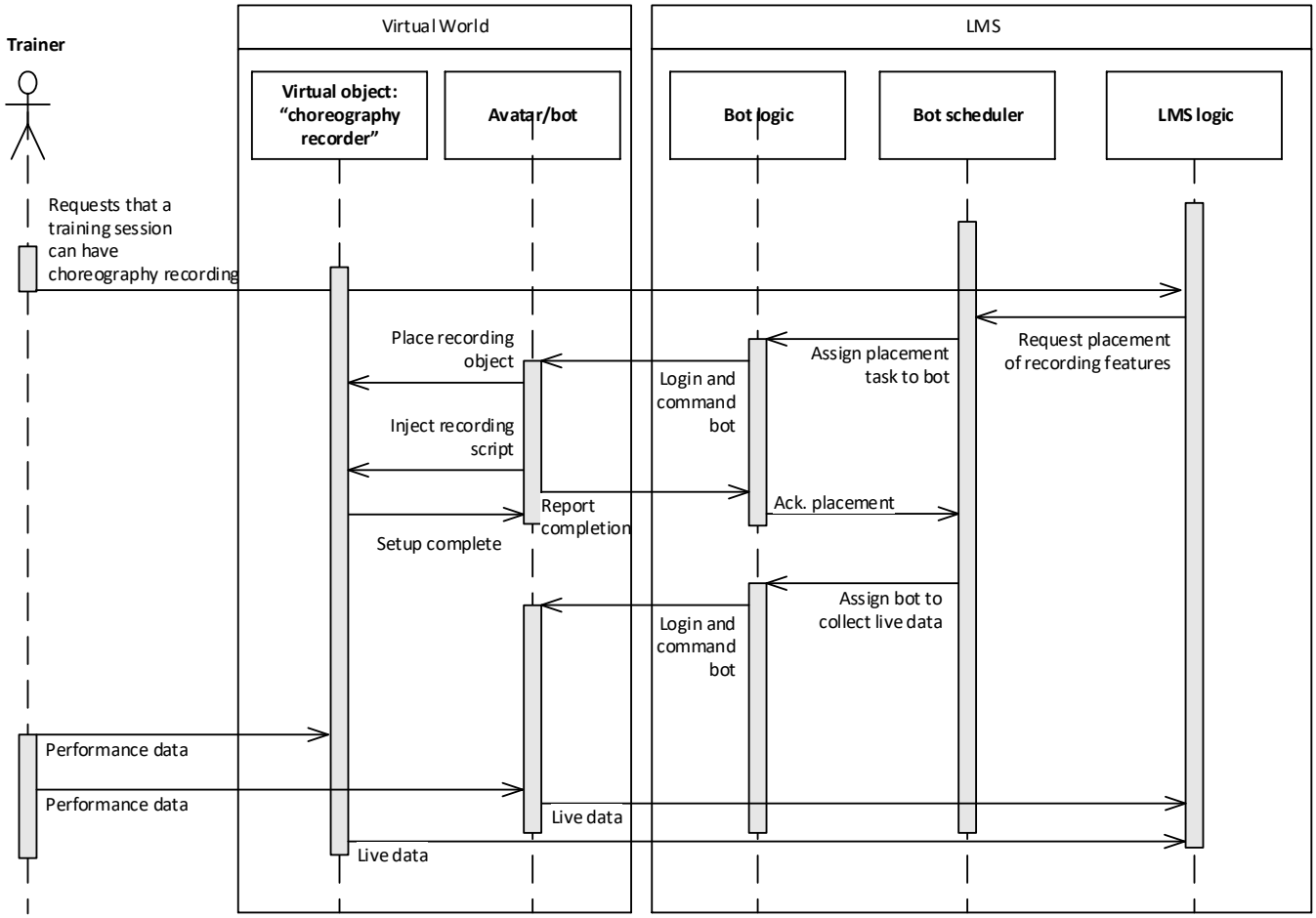

Fig. 7 – Sequence diagram for solution vector 4 applied to Case 3: initiating choreography recording

The completion of recording takes place in much the same way: either the trainer/training designer set it up to end at a specific time or as the outcome of interacting with an object. In either case, the LMS logic will use solution vector 4 to conclude the recording and clean up. This is shown in Figure 8. The LMS logic requests that the bot scheduler terminates the choreography recording, and this results in two different actions: bots conducting live collection of time-critical data are logged out and other bots are used to interact with recording objects, either to stop their recording actions or to delete any recording scripts that had previously been injected.

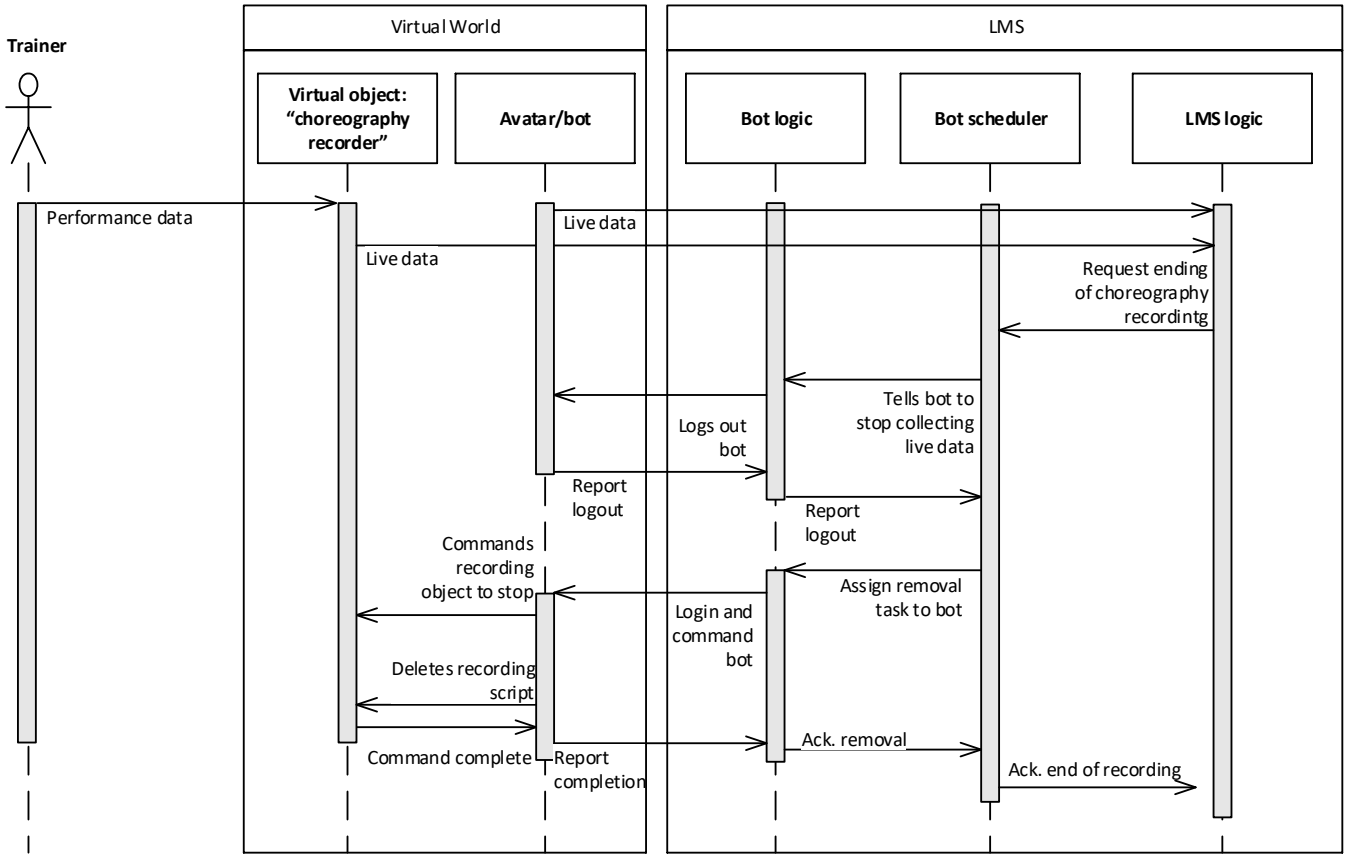

Fig. 8 – Sequence diagram for solution vector 4 applied to Case 3: concluding choreography recording and cleaning up.

5. CONCLUSIONS

The provided solution vectors have been devised from research using the OpenSim/Second Life Grid technological ecosystem, and hence bear some dependencies to it. Notwithstanding, the core control aspects of each case take place in the LMS, outside the virtual world and independent from it: knowing the state of session content, participants, roles, enabled features, etc. While other platforms don't provide such a powerful range of interaction capabilities to users (and hence bots), there are often alternatives that can follow the same rationale. For instance, HighFidelity [24] enables remote execution of scripts, rather than inside its servers; Activeworlds [25] enables developers to link automated clients to collect data and provide interaction. The bot spool approach could, conceivably, be expanded to use a variety of such interaction approaches, abstracting the complexity from the core LMS control processes. That is, expanding to the overall interaction process the proposal by Silva et al. for storing and replaying avatar choreographies across different virtual world platforms [17].

This approach was implemented and tried out in a large-scale training course for a major telecommunications operator in Brazil. While there are no published accounts, we can testify that it was technically sound and operational, but curtailed by the need for massive teacher/trainer and student/trainee training in the use of virtual worlds. Hence, we alert the reader for the need to combine the technical solution presented in this document with a wider educational/training framework for deployment. Field research with end users is needed to improve and refine this architecture and indeed the requirements from which it emerged.

Finally, while the approach was designed for virtual world platforms, it may also hold potential for integration of serious games with LMS. Currently, serious games development is disconnected from LMS in much the same way virtual worlds were, but we hope efforts such as the our own described herein help map a path towards the full integration of not only virtual worlds but also serious games with LMS, and from that enable more widespread use of these technologies in education and training.

6. REFERENCES

[1] Morgado L, Manjón B, Gütl, C (2015) Guest editorial: overcoming the technological hurdles facing virtual worlds in education: the road to widespread deployment, Educ Technol & Soc, 18(1):1-2

[2] Second Life Official Site - Virtual Worlds, Avatars, Free 3D Chat. http://secondlife.com

[3] OpenSimulator, http://www.opensimulator.org

[4] Formare, http://www.formare.pt

[5] Morgado L, Paredes H, Fonseca B, Martins P, Almeida A, Vilela A, Peixinho F, Santos A (2016) A Bot Spooler Architecture to Integrate Virtual Worlds with E-learning Management Systems for Corporate Training, J Univers Comput Sci 22(2):271-297. doi: 10.3217/jucs-022-02-0271

[6] Peffers K, Rothenberger M, Tuure t, Vaezi R (2012). Design Science Research Evaluation. In Peffers K, Rothenberger M, Kuechler B (eds) Design Science Research in Information Systems. Advances in Theory and Practice. DESRIST 2012. Lecture Notes in Computer Science, vol 7286. Springer, Berlin, Heidelberg, pp 398-410. doi: 10.1007/978-3-642-29863-9_29

[7] Morgado L, Paredes H, Fonseca B, Martins P, Almeida Á, Vilela A, Pires B, Cardoso M, Peixinho F, Santos A (2016) Integrating Virtual Worlds with Learning Management Systems: The MULTIS Approach. In Garcia-Blas J, Carretero J, Ray I, Jin Q, Georgalas N (eds.) Proceedings 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 8th International Symposium on Cyberspace and Security, IUCC-CSS 2016, 14-16 December 2016, Granada, Spain. IEEE Computer Society, Los Alamitos, CA, USA, pp 167-172

[8] Ghanbarzadeh R, Ghapanchi A, Blumenstein M, Talaei-Khoei A (2014) A decade of research on the use of three-dimensional virtual worlds in health care: a systematic literature review, J Med Internet Res, 16(2):e47. doi: 10.2196/jmir.3097

[9] Morgado L, Paredes H, Fonseca B, Martins P, Antunes R, Moreira L, Carvalho F, Peixinho F, Santos A (2016) Requirements for the use of virtual worlds in corporate training - Perspectives from the post-mortem of a corporate e-learning provider approach of Second Life and OpenSimulator. In Allison C, Morgado L, Pirker J, Beck D, Richter J, Gütl C (eds.) iLRN 2016 Santa Barbara - Workshop, Short Paper and Poster Proceedings from the Second Immersive Learning Research Network Conference, Santa Barbara, CA, USA. Technischen Universität Graz, Graz, Austria, pp 18-29

[10] Kemp J, Livingstone D (2006) Putting a Second Life «Metaverse» Skin on Learning Management Systems. In Kemp J, Livingstone D (eds.) Proceedings of the First Second Life Education Workshop, Part of the 2006 Second Life Community Convention, August 18th-20th 2006, Fort Mason Centre, San Francisco. University of Paisley, Paisley, UK, pp 13-18

[11] Moodle - Open-source learning platform | Moodle.org, https://moodle.org/

[12] Sloodle User Documents - SLIS Second Life Wiki, https://www.sloodle.org/docs/Sloodle_User_Documents

[13] Fillwalk J, interview reported by Werner G (2009). Online toolset will allow educators to maximize use of Second Life and Blackboard, https://apps.bsu.edu/CommunicationsCenter/Story.aspx?CategoryID=81&MessageGuid=FB31E866-0086-4089-8728-CF2BDD88549C&OptIn=Y

[14] Internet Archive Wayback Machine, "http://vushi.org", https://web.archive.org/web/20120715000000*/http://vushi.org/

[15] vushination, https://www.youtube.com/user/vushination

[16] Madeira A, Sequeira P, Morgado L, Gonzaga L (2010) Controlo da Assiduidade em Aulas Efectuadas no Second Life, RISTI - Rev Iber Sistemas Tecnol Inf, 5:87-100

[17] Silva E, Silva N, Morgado L (2014) Model-Driven Generation of Multi-user and Multi-domain Choreographies for Staging in Multiple Virtual World Platforms. In Ait Ameur Y, Bellatreche L, Papadopoulos G A (eds) Model and Data Engineering, 4th International Conference, MEDI 2014, Larnaca, Cyprus, September 24-26, 2014, Proceedings. Springer International Publishing, Cham, Switzerland, pp 77-91. doi: 10.1007/978-3-319-11587-0_9

[18] Pinheiro A, Fernandes P, Maia A, Cruz G, Pedrosa D, Fonseca B, Paredes H, Martins P, Morgado L, Rafael J (2014) Development of a mechanical maintenance training simulator in OpenSimulator for F-16 aircraft engines, Entertain Comput, 5(4):347-355. doi: 10.1016/j.entcom.2014.06.002

[19] Jailly B, Gravier C, Preda M, Fayolle J (2011) Interactive mixed reality for collaborative remote laboratories. In MTDL '11 Proceedings of the third international ACM workshop on Multimedia technologies for distance learning. ACM, New York, NY, USA, pp 1-6. doi: 10.1145/2072598.2072600

[20] Al-Khanjari Z, Al-Roshdi Y (2015) Developing virtual lab to support the Computer Science Education in Moodle. In 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV 2015), Bangkok, Thailand, 25-27 February 2015. IEEE, Piscataway, NJ, USA, pp 186-191. doi: 10.1109/REV.2015.7087290

[21] SLOODLE Presenter - SLIS Second Life Wiki, https://www.sloodle.org/docs/SLOODLE_Presenter

[22] Morrison J, Meliz L (1999) Foundations of the After Action Review Process. Special Report 42, United States Army Research Institute for the Behavioral and Social Sciences, Washington, DC, USA

[23] Vilela A, Cardoso M, Martins D, Santos A, Moreira L, Paredes H, Martins P, Morgado L (2010) Privacy challenges and methods for virtual classrooms in Second Life Grid and OpenSimulator. In Debattista K, Dickey M, Proença A, Santos L P (eds.) 2nd International Conference on Games and Virtual Worlds for Serious Applications, VS-GAMES 2010, Braga, Portugal, 25-26 March 2010. IEEE Computer Society, Los Alamitos, CA, USA, pp 167-174. doi: 10.1109/VS-GAMES.2010.30.

[24] High Fidelity. https://highfidelity.io/

[25] ActiveWorlds: Home of the 3D Internet since 1995. https://www.activeworlds.com/