uAdventure: The eAdventure reboot

Combining the experience of commercial gaming tools and tailored educational tools

Ivan J. Perez-Colado, Victor M. Perez-Colado, Ivan Martínez-Ortiz, Manuel Freire-Moran, Baltasar Fernández-Manjón

Dept. of Software Engineering and Artificial Intelligence, Facultad de Informática, Universidad Complutense de Madrid Madrid, Spain

{ivanjper, victormp}@ucm.es, {imartinez, manuel.freire, balta}@fdi.ucm.es

***Abstract*—Educational games (aka serious games, SG) are powerful educational contents. However, they are costly to develop, and once developed, SGs become dependent on software and hardware combinations that may become obsolete, such as Adobe Flash or Java Applets. Addressing these problems would allow a much greater use of SGs in education. The eAdventure authoring tool, developed by the e-UCM research group, addressed high development costs, and resulted in the creation of multiple SGs in collaboration with different institutions. However, eAdventure's Java Applets have become increasingly difficult to run due to platform obsolescence. To maintain the benefits of the eAdventure platform and user base, we have created new platform called uAdventure: an SG editor built on top of the game engine Unity that allows for the creation of educational adventure games without requiring programming. Since Unity is supported on a majority of platforms (including mobile). By developing SGs with uAdventure, the games become future-proof, as they can be updated and retargeted for new platforms as required. In this sense, uAdventure improves the lifecycle of SGs by reducing both authoring and maintenance costs.**

Keywords—serious games; Unity 3D; learning analytics; geolocalization

I. INTRODUCTION

Educational simulations, gamification approaches and educational games in general (hereinafter referred to as serious games, SGs) have been attracting lot of attention by education and training communities due to their intrinsic motivating and engaging characteristics [1]. SGs have been applied in different educational settings and application domains, most notably in business management, health, and military. SGs are used to facilitate training of abilities tactics that are specifically suited to be acquired in role-playing educational settings, such as conflict management or business negotiation skills. Moreover, SGs can be used to acquire or refresh a required ability that is infrequent or rare in real life (medicine) or poses security or cost difficulties (military). In addition to education and training purposes, SGs are also used in other domains such as public health or environmental protection to improve awareness or to affect behaviors.

As a result of the increasing use and acceptance of SGs, a niche market is being created around their development and maintenance. This market is expected to grow over 16% annually between 2015 and 2020 [2]. For instance, CodinGame, an online SG to learn programming, has around half a million registered users, two thousand of which are online at any given instant[3].

The process of developing and procuring SGs can be classified using the following categories: i) *Commercial Off-the-Shelf (COTS)*, ii) *Pre-existing Serious Game*s and iii) *Tailored Serious Games*. This classification is based on the alignment of the learning goals of the educators against the actual learning goals (if any) that are part of the SG's actual design. Generally, the greater the degree of personalization, the better the game will suit educators' needs. As we get closer to a fully-personalized game, the development process of the game will become more complex and costly (per user).

In some cases, an already available commercial game would suit the educator's needs. These COTS games, when applied in an adequate educational context, can have a relevant impact. For example, Zoo Tycoon[1] is a game focused around building and running successful zoo scenarios where players learn concepts related to economics or business management; and has been used as a SG to help teach these concepts. The economical advantage of this approach is clear: the game does not need to be developed, only licensed. However, the effective application of this approach also requires an additional effort to design the pedagogical scenarios and a degree of teacher support while the game is deployed as an SG.

Pre-existing Serious Games are commercial SGs that are usually supported by training and consultancy companies that may offer a customization layer. This customization layer, is usually offered as an *a la carte* service, where the organization/ instructor can choose a set of learning goals and the company provides a set of SGs or a single SG that cover the required competencies or learning goals. This approach can be considered as a *Games as a Service*, where the learning organization/instructor does not have to either develop or maintain the game, for a fee. Other advantages include streamlined integration with the learning process, or assessment tools for monitoring and behavior analysis. An example of this genre is Classcraft[2], where the classroom is transformed into a

[1] https://en.wikipedia.org/wiki/Zoo_Tycoon

[2] https://www.classcraft.com

978-1-5090-5466-4/17/$31.00 ©2017 IEEE 25-28 April 2017, Athens, Greece

role-playing game that improves students' collaboration and teamwork.

Finally *tailored serious games*, despite having the greatest potential effectiveness, also tend to require large, multidisciplinary development teams and, therefore, have high production costs. However, for simpler SGs, it is feasible to provide educators with an easy to use platform, with limited features, that can allow them to create their own SGs; or at least to modify or localize it to their needs.

The eAdventure (eA) platform [4] provides an authoring tool that allows for SG development without requiring advanced computer skills (i.e. programming). In addition, the eA platform allows for the reuse and repurposing of preexisting eA games. This allows educators to customize the games for a specific educational setting. For example, changing the students' evaluation included in the game or simply changing graphic assets of the game elements to customize their aspect to the specific learning context.

However, the eA platform was launched in 2007, and although it has received several updates, its underlying technology has run into technical limitations. In particular, eA's targeted the Java Plugin, then present in most browsers. However, as described in JEP 289[3], the Java Plugin is to be deprecated in newer versions of Java; and, as of this writing, most browsers already ship with this component either absent or disabled. To address these concerns, and to streamline the development of new educational features, we have designed uAdventure (uA), an evolution of eA.

The rest of the paper is structured as follows; section II provides insight into eA analyzing the limitations that required an evolution from both technical and educational perspectives. Section III and IV describe uA and the new features and educational opportunities that it offers. Finally, section V provides some conclusions and future lines of work.

II. From eAdventure to uAdventure

A. eAdventure

Developing games, and especially SGs, requires addressing and balancing both educational and technological aspects. One of the first aspects to consider is the platform or technology used to develop the game. For example, using a professional game engine such as *Unity* or *Unreal Engine* requires a game development team with good programming background; and this will prevent most teachers from participating in the SG's core development team.

Another aspect the adequate balance between engagement and education [5]. On the one hand, if the game is too focused on the educational facets it may not be engaging enough for the student. On the other hand, if the game is too narrowly focused on the fun facet, the student may not learn enough, yielding only limited educational value.

Within games there are several genres, and not all of them are equal from an educational perspective. Dickey [6] and Amory [7] have identified point-and-click adventure games as one of the most suitable genres for SG, due to the presence of elements such as a slow pace, reflection, study of the environment, and problem-solving [8]. There are specific commercial editors for creating these kind of games, such as *Adventure Game Studio*[4]*, Adventure Maker*[5] or *3D Adventure Studio*[6] [9]. However, these platforms do not usually include support for SG educational elements (e.g. evaluation).

eA is a platform for simplifying the development of 2D point-and-click adventure serious games [4]. In eA, educators can actively participate in the development of the SG side-by-side with the game developers or even develop the SG themselves without requiring programming skills. This way, educators are at the core of the development process taking care of the educational aspects that fits for their own needs.

eA has four characteristics that make it especially suited for the creation of educational SGs: i) game development can be achieved with small budgets, ii) maximizes the Return on Investment (ROI), iii) customizes the learning experience and iv) facilitates both distribution and evaluation. One of the most expensive aspects of game development is the creation of graphic assets. With eA, it is possible to use pictures taken with any digital camera to create the graphic assets. eA games are editable, meaning that existing games can always be opened with the editor to either customize them for specific learning scenarios, or to update them for new requirements.

In addition, eA games can be packaged as an e-learning standard content package that includes the game as a Java applet making possible to distribute it using a Learning Management System (LMS). Furthermore, if the LMS it is compatible with the ADL SCORM [10] e-learning specification the eA games can send back assessment results to the LMS.

B. Why does eAdventure need a refurbishment?

eA design started in 2007, and the first version was delivered in early 2008 as an open source project. The project has been maintained since then by different contributors, mainly by researchers that have been actively using the platform. These contributors have often added features requested by educators participating in the eA community. Nevertheless, there were some architectural and technological decisions that were correct at that time, but are now hindering the maintenance and evolution of the platform.

eA was built with Java technology to take advantage of the *"build once run anywhere"* motto. Java technology powered both the eA editor (used in game development) and the games themselves, which could be deployed in different platforms without changes. This way it is possible to alleviate, or completely avoid, compatibility issues between the computer used by the educator (or the development team) to create the SG and the usual heterogeneous environment that can be found at school labs (including security restrictions). In addition, this

3 February 2016's JEP 289: http://openjdk.java.net/jeps/289

[4] http://www.adventuregamestudio.co.uk/

[5] http://www.adventuremaker.com/

[6] http://3das.noeska.com/

multiplatform nature and the ability of eA games to be customized fostered the collaboration between educators in a community of practice by sharing SGs.

Although Java's *"build once run anywhere"* motto may still hold for the desktop platform, desktop computers, once the main platform in school, have since disappeared from multiple institutions, displaced by tablets. Desktop Java is generally unavailable on tablets and smartphones: although Java as a language is still relevant (due to the Android platform), development in Java is not longer providing effective multiplatform support. Moreover, mobile devices offer new ways to interact with the device (touch, accelerometer, geolocation, etc.) that open new opportunities to create ubiquitous educational experiences that should be used in SGs.

eA SG deployment relies on Java applets (i.e. Java programs that run inside the browser) to facilitate SG distribution and integration into existing e-learning platforms. When eA was created, e-learning content interoperability was mainly focused on distribution of web content. The use of Java applets in eA facilitates the reuse of e-learning standards allowing SGs to be packaged as an IMS-CP content [11]. This allows students to access and play the game just like any other web content. The eA platform could also take advantage of ADL's SCORM e-learning standard to communicate back with the LMS, mainly to store SG progress and assessment information inside the hosting Learning Management System (LMS).

However, due to the continuous security problems [12] in Java Applets in particular and in browsers' plugins in general, most of the browsers have already deprecated the Java applets or are in the process of deprecating them [13].

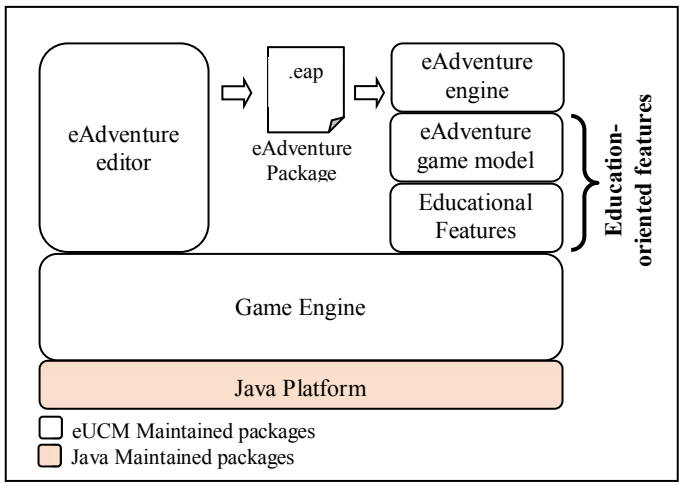

Fig. 1 eAdventure's main components

The eA platform, besides being an open source point-and-click editor, complements the usual game platform features with educational features (e.g. assessment, e-learning standards support) that make eA still unique and relevant (see Fig. 1). Although only two of eA's components have been specifically designed for educational purposes, considerable effort is required to maintain the platform as a whole. This imbalance it is dangerous for the maintenance and survival of an open source research project with limited resources (developers). In particular, keeping up with technological evolution makes it harder to add and update the educational features that teachers expect and demand.

Nonetheless, we consider that eA's main ideas are still valid and relevant to the educational community. Therefore, we decided build a new SG platform, which we describe in the following section.

III. UADVENTURE

uA development aims to achieve the following goals: i) to address the technical issues of the eA platform, ii) to face the project management/survival issues, iii) to provide a solid base platform to build new educational features and iv) maintain compatibility with eA (at least importing previous eA games).

uA is built on top of Unity mainly because it has become the most popular game development platform [14]. One of the reasons of this popularity is its multi-platform support[7] including desktop (Windows, Mac OS and Linux), consoles (PlayStation, Xbox, PSVita or Nintendo 3DS), web and mobile (Android and iOS). Unity has attracted attention both from big companies like Blizzard with "Hearthstone"[8], Nintendo with "PokemonGO"[9] or Ubisoft with "Grow Up!"[10], and is also one of the top choices for so-called "indie" developers[11]. The "indie" label is generally used to refer to low-budget games, and in this sense encompasses most SG developments. Compared to the other choices, Unity stands out for its simplicity and extensibility, rich documentation, and a very active community of developers.

In addition, the choice of Unity was an answer to project management issues. uA can take advantage of a preexisting strong platform and community. Thus uA development time can be focused on the development of those characteristics that made eA unique. Moreover, due to the use of a well-known platform, finding contributors or hiring developers to create new features should be greatly simplified.

Furthermore, the expansion of mobile technologies has created new opportunities to apply them in education. As we move into mobile technologies, new forms of "anytime and anywhere" learning are possible. Learning can happen at any moment with a range of devices: at home with the PC, at school with a tablet, or just outside during the free time with the smartphone.

uA have been created in a context where previously developed eA games have serious deployment problems. This new uA framework, built on top of Unity, allows eA projects to be imported, allowing them to be tested and modified on the new execution core. This extends, supports, and simplifies their lifecycles. After opening a game in uA, and using the *Unity*

[7] https://unity3d.com/unity/multiplatform
[8] http://eu.battle.net/hearthstone/en/
[9] http://www.pokemongo.com/
[10] https://www.ubisoft.com/en-GB/game/grow-up/
[11] https://unity3d.com/public-relations

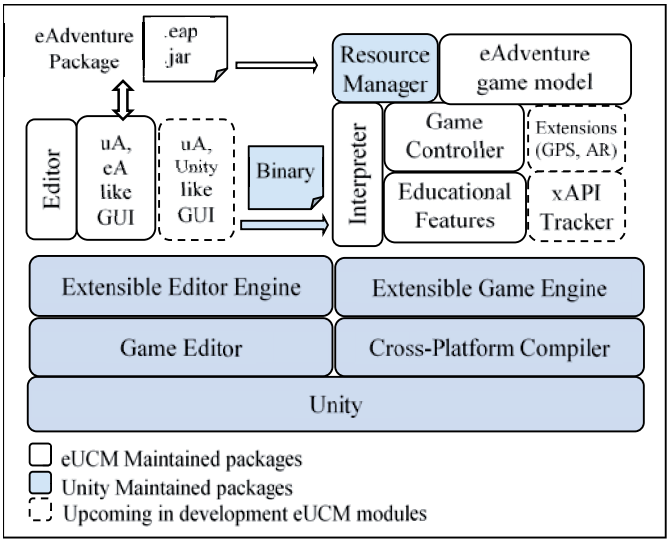

Fig. 2: uAdventure modules architecture, including Unity provided modules.

compilation tools, it can be exported as a standalone game for a range of different platforms and operating systems.

A. uAdventure Architecture

eA is composed of several layers (see Fig. 1), such as editor, game player, representation core and project builder. In tightly-coupled, complex software solutions developers need to cope with all the complexity; and in this case, developers wishing to extend eA needed to be familiar with all of these layers. In contrast, uA is composed only of two main layers: *Interpreter* and *Editor*. Most of the previous ad-hoc eA layers have been replaced by Unity's standard layers. Thus, uA will be easier to maintain and evolve.

The uA module architecture is presented in Fig. 2. It shows how the graphical editor (GUI), the interpreter and the emulator are built on top of Unity. New features are highlighted in Fig. 2 with dashed lines: the xAPI tracker, extensions such as location support, and a future Unity-like GUI.

Compared to the previous architecture, it may seem more complex, since there are more modules. However, this is misleading: even though there are more e-UCM maintained modules, they are organized into only two layers: *Editor* and *Interpreter*. The rest of the e-UCM maintained modules are smaller and easy to maintain compared to the eAdventure's packages (Editor and Game Engine) shown in Fig. 1, whose cores have been replaced with the Extensible Editor Engine and the Extensible Game Engine provided by Unity.

Therefore, the two main layers that compose uA are: *Interpreter* and *Editor*. The *Editor* is focused on creating a game specification that, in conjunction with the required game resources, will make it playable. It uses the eA data model plus some new additions, like new map scenes and new assessment specification [15]. The *Interpreter* relays on Unity's representation core and translates all eA model components into Unity game elements.

B. The uAdventure editor

The *Editor* allows users to modify the game specification, manage and create new game resources. Some key aspects of eA have been recreated in uA, such as the main graphical user interface, including menus, windows and general appearance. This way, it will be easier for the current eA users to migrate to

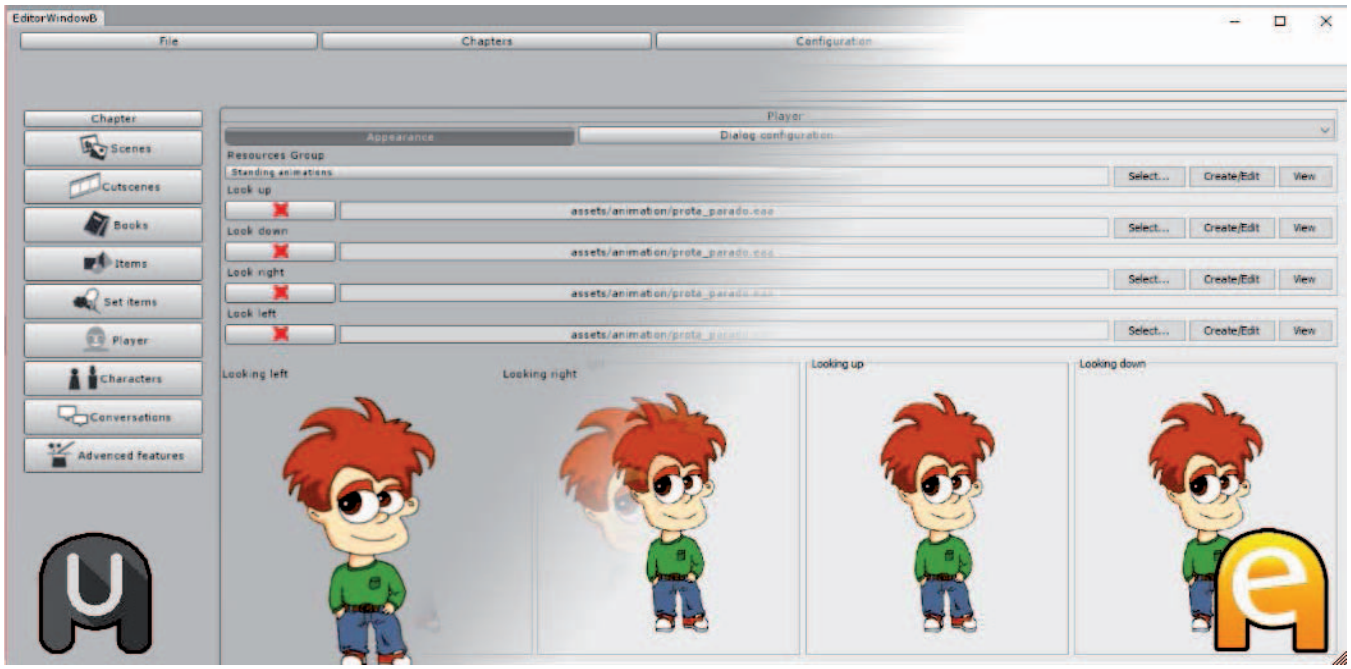

Fig. 3: Fade transition from players appearance view on uAdventure (left) to previous players appearance view on eAdventure (right). The editor is quite similar in both systems.

uA. However, uA will also provide a brand-new Unity-Unified GUI. This new GUI will be based on the idea of tighter integration with the Unity editor, and is inspired by the most used Unity windows, such as the Scene preview and object (model) hierarchy, Unity animator, or element inspector. This enables greater modularity and allows game developers familiarized with Unity to take advantage of uA's added value for educational videogames. A comparison between the old eA GUI and the new uA GUI is presented in Fig. 3. In this case, an eA user that is just starting out with uA can quickly become productive: the interaction with the GUI is very similar to what they already know, making for a smoother learning curve.

As uA evolves inside Unity, it will be able to benefit from other Unity features, such as 3D assets (downloadable from the Unity Asset Store), the physics engine, navigation meshes for movement, or native animation state machines. Games will result richer and developers should need less time to produce them. In addition, Unity's native features are already well documented, can be reused between multiple games and distributed as packages, and will be improved and maintained as Unity evolves.

One of the improvements of uA is multimedia support with the usage of the *FFmpeg* project for export-time video conversion. Multimedia resources including video and audio are complex to deal with when deploying a game in different platforms. uA addresses those problems, not only when playing the game (using Unity's multimedia players), but also during game creation. It uses FFmpeg for converting videos on-the-go, increasing, for example, the number of supported video formats, from AVI (DIVX 4, XVID, DIVX3low, S-Mpeg 4 V2, DIVX5) and MPG (MPEG1), to almost every video format available. This will greatly simplify the inclusion of new assets in games.

Therefore, using Unity as a support layer has not only allowed us to include new features; but also has improved the development and maintenance of uA. On one hand, development requires a deep knowledge of the Unity platform and experience in extending. On the other hand, some of the most time-consuming maintenance aspects are no longer required (e.g. coping with changes in mobile platforms, multimedia playback) as they are provided by Unity. This allows e-UCM's resources to focus on improving and increasing the educational features and the added value offered by uA.

C. The uAdventure interpreter: an emulator for eAdventure

The *Interpreter* loads game description file and makes it playable. Taking advantage of Unity's features, the *Interpreter* extends the Extensible Game Engine, developing behaviors for every element that is visible or plays a role in the game.

Furthermore, the *Interpreter* also manages the *GameState*, including the state of the *flags*, *variables*, *global-states*, and other game elements such as the *inventory* or the *current scene*. Finally, it also orchestrates, relying on Unity's *MonoBehaviour* lifecycle, the execution of the game.

The *Interpreter* is not only responsible for representation, but also for handling user interaction. Video games developed with uA can be exported to a variety of platforms, which raises a serious issue: different platforms imply different interaction methods. uA supports different interaction methods, including traditional mouse and keyboard as well as touch screen environments. The framework provides developers with some tools for making the game easier to play depending on user's runtime environment. For example, when a game is *exported* for PC/Mouse-Keyboard system, the game will be presented like older eA games, with different cursors, floating names, and lists of answers at the bottom of the screen. On other hand, when a game is exported for us in a Tablet/Touch system, interactive elements glow periodically to capture the player's attention, because tablet players can no longer "mouse-over" scenes to find interactive elements.

uA also includes a multi-platform eA game *emulator* that can take a ".jar" executable eA game (an already-exported, ready-to-run eA game) and run it on every platform available within uA. This emulator runs the previous games, but supports new uA features (provided by the Interpreter). This includes better animations, more effects, improved and adapted GUI, or the ability to play eA games on mobile devices. The emulator allows users to browse their filesystem, and easily import those games they want to play on their preferred platform. Previously launched games will be saved in the emulator with its configuration that will enable learners to use old games on pre-configured devices.

D. Migrating SGs from eAdventure to uAdventure

eA's SG life cycle has been greatly improved with the implementation of the uA *Interpreter* as most of the games developed with eA in the past are now supported by uA. This prevents the considerable effort that went into designing, implementing and testing these games from being lost. As the game is now difficult to deploy or does not run in new environments it would have otherwise required a complete redevelopment to achieve the same degree of support. That is, the use of uA avoids the high cost in time and money that would otherwise be required to reimplementation these older games in new platforms.

There is an actual case study of adapting a game to a new game engine, where an experienced game developer of the e-UCM research group reproduced the First Aid [16] game inside Unity. The production took almost 50 hours of development, and even with that effort, only 85% of the game was supported (some game situations were ignored). Based on this data, we can estimate that 60 to 80 hours of development are necessary to fully rebuild a simple eA game into a new game engine, assuming that the developer responsible has access to the initial detailed design document, and without factoring extensive user testing, which may reveal additional problems. This is a significant cost for the maintenance of a previously produced SG, where in many cases there is no budget at all for on-going maintenance. Indeed, the 6 educational games developed by the educational organization CATEDU[17] using eA are in this situation. Thanks to uA, this cost can be reduced to the few minutes it takes to import the project into uA and generate new versions. Indeed, we have taken the First Aid [16] game and done exactly that; and are currently piloting it again in different

schools in Madrid to test the learning analytics features described in the next section.

IV. NEW EDUCATIONAL FEATURES IN UADVENTURE

In terms of educational features, there are few differences between uA and eA. This section will focus on description of the two main educational features that are being implemented in uA: i) learning analytics as an assessment tool and ii) geolocated games.

A. Learning Analytics for SGs

The evaluation of a formative activity that uses a SG poses significant challenges. Usually the teacher has little control over, or information of, the learner's activity through the process of playing a serious game [1]. This also happens usually when using other eLearning technologies in the classroom, such as MOOC or LMS. Some research results show that the more teachers know about the learners' behavior using a learning tool, the better teachers can control that learning tool [18].

eA's assessment feature was based on conditions that tracked player actions inside the game such as "The player has failed test X" or "Player refuses to accept help from Non-Player Character Y" [11]. These conditions, although useful for evaluation, were one of the most complex features to master for developers of eA games. This requires mastering conditions and, to some extent, to have computational thinking abilities to understand how to link the game states that have a correlation with the educational learning goal; or with the skill that is put into practice while playing the game. Moreover, the eA editor allows authors to create multiple assessment profiles (sets of conditions) [19]. The evaluation outcome was a textual report, which had to be interpreted in terms of actual learning outcomes by an instructor.

In addition, evaluation based on pre-set conditions implies that, the more the authors want to evaluate, the more time they must spend creating conditions. Still, there are some aspects of the learner's behavior that are lost because conditions can only track scenarios that are devised before the learning activity takes place. Finally, this approach works for small groups of learners, but as the groups become larger, the difficulty of supporting corrective interventions during the learning activity increases: more learners result in more the individual documents that must be read and analyzed.

A complementary approach to condition based assessment, is based on the application of big data and analytics techniques where learners' gameplay data is harvested, processed and presented in a dashboard to the educator. This approach can be used to provide metrics that will help the instructor to assess learners or even discovering game design errors [20], [19].

In order to apply big data and analytics techniques, all gameplay interactions must first be collected. Instead of creating a custom format to represent theses interactions, we use an existing e-learning specification called xAPI (eXperience API) [21]. In xAPI, traces of interactions (called statements) are composed of: i) an *actor*, that defines who performed the action, ii) a *verb*, that defines the action between the actor and the object, iii) an *object*, that defines the thing that was acted on, iv) and optional properties such as *result*, representing a measured outcome related to the statement, or a *timestamp* of when did that happen [22]. Under this umbrella it is indeed possible to represent gameplay actions; but in order to facilitate the development of analytic tools that automate insights of the learners' gameplay the e-UCM team, in collaboration with the ADL initiative (developers of x-API), authored a specific xAPI application profile for use with serious games [15].

The reasons for using xAPI are twofold. On the one hand, this allows uA SG to interact with existing e-learning tools that are compliant with the xAPI specification and also with, for example, a Learning Record Store (a storage service for xAPI statements). On the other hand, and even more important, gameplay sessions will be compatible with future xAPI tools, for example, it will be possible to apply new learning analytics tools to gameplay sessions recorded in the past.

For example, with the current xAPI support (although incipient) it is possible to integrate the SGs created with uAdventure with the Learning Analytics application created as part of the H2020 RAGE[12] and BEACONING[13] projects.

B. Embracing Ubiquity: geolocated games

One of the main advantages of the uA and its supporting platform is the ability to create SGs for mobile devices. Moreover, these devices usually include a plethora of sensors that are unique to the mobile platform.

Getting into mobile allows games to be played anywhere and anytime. This opens new opportunities to use SGs in different educational scenarios, and using alternative interaction mechanisms. For example, MIT's MitAR[14] is a serious game platform that uses augmented reality and geolocation [23]; although it was never successfully ported to today's major mobile platforms. More recently, 2016 was the year of the engaging AR mobile game "PokemonGo", which not only attracted large numbers of users, but also indirectly had an impact in the players' wellness [24].

uA is starting to take advantage of the GPS, compass and camera sensors that are already available to Unity games. This will allow the creation of geolocated games that were beyond the capabilities of eA, such as gymkhanas, interactive tours, or cultural visitor guides. This new feature has been implemented adding a new type of *scene* to the uA data model. Until now, a game scene was a metaphor to represent a place and environment where players or their avatars interact with the game world. This new type of scene actually is a virtual representation of the real environment where the game takes place. This new *map scene* (and its companion editor) provides access to a map to define POIs (points of interest), regions, or even routes to link all the existing mechanics to the geolocation gaming paradigm.

Finally, in order to be effective indoors, QR code scan support was added. QR codes are easily generated, and can be

[12] http://www.rageproject.eu/

[13] http://beaconing.eu/

[14] http://web.mit.edu/mitstep/projects/mitar-games.html

placed on interesting locations to be scanned by those playing on a mobile device; however, their use requires access to the device's in-built camera from within the game. In addition to enabling indoor use, support for QR codes also allows games to be changed or adapted to new locations by moving the (physically placed) QR codes around. For instance, one of most well-known games in the eA community is the "Fire Protocol" game[15], originally created as a demonstrator of eA's capabilities, and also as a drill for the actual fire protocol procedure in the Computer Science building at Complutense University of Madrid. Although the already uses pictures of actual locations inside the building, forcing the players to physically move around the actual locations depicted in the game provides a much greater degree of immersion.

V. Conclussions and Future work

The eAdventure (eA) platform has reached its technological limit due to Java platform's limitations supporting mobile and web-based deployments. However, the main goals and advantages of the eA platform are still relevant for the SG community. eA's SG life cycle has been greatly improved with the implementation of uAdventure (uA) as most of the games developed with eA in the past are now supported by uA.

The uA project addresses these technological limitations, and provides new educational capabilities that were not envisioned when eA was created. In order to address current and future technological challenges, uA is built on top of the Unity platform, which provides solid technical underpinnings, together with a strong user base and support in the game development community.

The uA tool has been created around two main elements: the *Editor* and the game *Interpreter*. The editor provides easy-to-use game authoring tool for non-programmers that is very similar to eA user interface, but includes some new features. Some of them are: touchscreen interaction support, Learning Analytics support, and geo-positioned game support with or without maps. Touchscreen support includes features that make playing point-and-click adventure games easier on touchscreens. Learning analytics support is provided by an integrated xAPI tracker that generates traces of what the learner does during gameplay. Finally, geo-positioning support uses mobile GPS and wireless capabilities, and new map scenes. Use of Unity has also allowed us to make multiple game representation improvements, giving games a better look and feel. Furthermore, by taking advantage of Unity's geolocation capabilities, uA will allow the creation of games such as gymkhanas or historical city tours.

The interpreter has been built using the very same model used in eA, in order to maintain the compatibility with games created with eA and facilitate the migration to the new platform. In the case-study presented in Section III-D, it can reduce up to 85% percent of the maintenance and migration costs associated with making a pre-existing SG capable of running on new platforms.

Integration of uAdventure with the real-time Learning Analytics platform used in the H2020 RAGE and BEACONING projects is ongoing. This LA platform can populate learning dashboards from incoming flows of xAPI traces, and can display alerts and warnings when learners enter risky areas or evidence anomalous behavior.

Although uA still requires significant work before public release, we believe that uA with the capabilities described in this paper will be a worthy heir of eA's legacy. Hopefully, the present work and the work done in the H2020 projects RAGE and BEACONING (where the e-UCM team participates) will facilitate a more widespread adoption of SGs in tomorrow's classrooms.

Acknowledgment

We would like to thank Piotr Marszal for his coding contribution on the first beta of uAdventure. This research has been partially financed by the Regional Government of Madrid [eMadrid S2013/ICE-2715], by the Ministry of Education [TIN2013-46149-C2-1-R] and by the European Commission [RAGE H2020-ICT-2014-1-644187, BEACONING H2020-ICT-2015-687676].

References

[1] M. Nichols, "A theory for eLearning," *Educational Technology and Society*, vol. 6, no. 2. pp. 1–10, 2003.

[2] "Serious Game Market worth $5,448.82 Million by 2020." [Online]. Available: http://www.marketsandmarkets.com/PressReleases/serious-game.asp.

[3] "CodinGame: General Leaderboard." [Online]. Available: https://www.codingame.com/leaderboards/global?column=codingpoints&value=all.

[4] J. Torrente, Á. Del Blanco, E. J. Marchiori, P. Moreno-Ger, and B. Fernández-Manjón, "e-Adventure: Introducing Educational Games in the Learning Process," in *IEEE Education Engineering (EDUCON) 2010 Conference*, 2010, pp. 1121–1126.

[5] M. Prensky, "Digital game-based learning," *Comput. Entertain.*, vol. 1, no. 1, p. 21, Oct. 2003.

[6] M. D. Dickey, "Game Design Narrative for Learning: Appropriating Adventure Game Design Narrative Devices and Techniques for the Design of Interactive Learning Environments," *Educ. Technol. Res. Dev.*, vol. 54, no. 3, pp. 245–263, Jun. 2006.

[7] A. Amory, "Building an Educational Adventure Game: Theory, Design and Lessons," *J. Interact. Learn. Res.*, vol. 12, no. 2, pp. 249–263, 2001.

[8] R. Van Eck, "Building Artificially Intelligent Learning Games," in *Games and Simulations in Online Learning*, IGI Global, 2007, pp. 271–307.

[9] P. Moreno-Ger, J. L. Sierra, I. Martínez-Ortiz, and B. Fernández-Manjón, "A documental approach to adventure game development," *Sci. Comput. Program.*, vol. 67, no. 1, pp. 3–31, Jun. 2007.

[15] http://e-adventure.e-ucm.es/course/view.php?id=29

[10] J. Torrente, P. Moreno-Ger, I. Martínez-Ortiz, and B. Fernandez-Manjon, "Integration and deployment of educational games in e-learning environments: The learning object model meets educational gaming," *Educ. Technol. Soc.*, vol. 12, no. 4, pp. 359–371, 2009.

[11] P. M. Ger, "eAdventure: Serious games, assessment and interoperability," in *2014 International Symposium on Computers in Education (SIIE)*, 2014, pp. 231–233.

[12] G. McGraw and E. W. Felten, *Java Security: Hostile Applets, Holes&Amp;Antidotes*. New York, NY, USA: John Wiley & Sons, Inc., 1996.

[13] "NPAPI deprecation: developer guide." [Online]. Available: https://www.chromium.org/developers/npapi-deprecation.

[14] F. Messaoudi, G. Simon, and A. Ksentini, "Dissecting games engines: The case of Unity3D," in *2015 International Workshop on Network and Systems Support for Games (NetGames)*, 2015, vol. 2016–Janua, pp. 1–6.

[15] Á. Serrano-Laguna, I. Martínez-Ortiz, J. Haag, D. Regan, A. Johnson, and B. Fernández-Manjón, "Applying standards to systematize learning analytics in serious games," *Comput. Stand. Interfaces*, vol. 50, pp. 116–123, Feb. 2017.

[16] E. J. Marchiori, G. Ferrer, B. Fernandez-Manjon, J. Povar-Marco, J. F. Suberviola, and A. Gimenez-Valverde, "Video-game instruction in basic life support maneuvers," *Emergencias*, vol. 24, no. 6, pp. 433–437, 2012.

[17] CATEDU, "E-ADVENTURES. Download page.," 2011. [Online]. Available: http://www.catedu.es/webcateduantigua/index.php/descargas/e-adventures.

[18] A. Ravenscroft, "Designing E-learning Interactions in the 21st Century: revisiting and rethinking the role of theory," *Eur. J. Educ.*, vol. 36, no. 2, pp. 133–156, Jun. 2001.

[19] W. L. Wong, C. Shen, L. Nocera, E. Carriazo, F. Tang, S. Bugga, H. Narayanan, H. Wang, and U. Ritterfeld, "Serious video game effectiveness," in *Proceedings of the international conference on Advances in computer entertainment technology - ACE '07*, 2007, no. January, p. 49.

[20] Á. del Blanco, J. Torrente, P. Moreno-Ger, and B. Fernández-Manjón, "Enhancing Adaptive Learning and Assessment in Virtual Learning Environments with Educational Games," in *Intelligent Learning Systems and Advancements in Computer-Aided Instruction*, IGI Global, 2013, pp. 144–163.

[21] M. Manso Vazquez, M. Caeiro Rodriguez, and M. Llamas Nistal, "Development of a xAPI application profile for self-regulated learning requirements for capturing SRL related data," in *2015 IEEE Global Engineering Education Conference (EDUCON)*, 2015, pp. 358–365.

[22] "xAPI-About @ github.com." [Online]. Available: https://github.com/adlnet/xAPI-Spec/blob/master/xAPI-About.md#partone.

[23] B. Coulter, E. Klopfer, J. Sheldon, and J. Perry, "Discovering Familiar Places," in *Games, Learning, and Society*, C. Steinkuehler, K. Squire, and S. Barab, Eds. Cambridge: Cambridge University Press, 2016, pp. 327–354.

[24] T. Althoff, R. W. White, and E. Horvitz, "Influence of Pokémon Go on Physical Activity: Study and Implications," *J. Med. Internet Res.*, vol. 18, no. 12, p. e315, Dec. 2016.