PlayLearn: A Platform for the Development and Management of Learning Experiences in Location-Based Mobile Games

Dimitris Makris, Konstantinos Makris, Polyxeni Arapi, Stavros Christodoulakis
Laboratory of Distributed Multimedia Information Systems and Applications (TUC/MUSIC)
Technical University of Crete
Chania, Greece
e-mail: {dmakris, makris, xenia, stavros}@ced.tuc.gr

Abstract—**Mobile location-based games, exploiting the unique capabilities of mobile devices, such as camera, GPS and compass, can have high learning potential. On one hand, they present a very attractive form of learning for modern students, even very young ones, who have already developed their skills in computer games and are very familiar with the use of mobile devices. On the other hand, location-based games provide a unique opportunity for education since they connect an area with a story, and their activities may result in social, experiential and situated learning. These characteristics can make them a powerful tool in a number of applications, including education, nature and museum exploration, city sightseeing, natural disasters awareness and prevention training. In this paper, we present the design and implementation of PlayLearn, a platform for the development and management of learning experiences in mobile location-based games, consisting of: (a) an authoring tool, supporting the creation and management of games, scenario editing, user interface customization and organization of gaming activities, and (b) a mobile application, compatible with most state-of-the-art mobile devices and platforms, supporting the play of games created by the authoring tool. Our implementation supports the Experience API, allowing the activities that happen as part of gaming (learning) experiences to be recorded, tracked and shared in a Learning Record Store.**

Keywords- mobile educational games; location-based games; learning experiences; experiential learning.

I. INTRODUCTION

Location-based services on mobile phones have become very popular in recent years. These services have been developed to provide location-based information and can be used to create games that exploit the geographical location of the user or other people. A -so called- mobile location-based game [1] is a type of pervasive game in which the game-play evolves and progresses via a player's location. Such games must provide some mechanism to allow the players report their location by some kind of localization technology like GPS. The difference between a video game and a location-based game focusing on the same story is that the later has a closer connection between game and reality.

In terms of the main objective, mobile location-based games may be categorized as games that are created for fun, for learning or for mixed objectives [2]. All these categories of games seem to have a higher learning potential. On one hand, they present a very attractive form of learning for modern students, even very young ones, who have already developed their skills in computer games and are very familiar with the use of mobile devices. On the other hand, location-based games provide a unique opportunity for education since they connect a geographic area or object with a story. The physical and cultural surroundings are an integral part of the game space, and the location of the gamers is a key aspect of the game-playing activity. By visiting the actual locations, the story becomes more authentic and leads to better educational results. Essentially, games of this kind allow the user to collect data from the real world and assign them to the game's map. De Souza et al. [3] observed that these activities produce learning that is social, experiential and situated. The combination of informal learning and mobile outdoor games can be seen as a relevant arena for conducting novel learning activities that involve learners in different tasks including physical motion, problem solving, inquiry and collaboration [4].

Learning experiences are those events and activities from which we learn by experience and can identify, to a certain extent, what we have learned [5]. Different learners have different characteristics and preferences (e.g., learning style, educational level, background knowledge) affecting how learning experiences might be organized in terms activities and learning material to achieve certain goals. Experiential learning, according to Kolb [6], can exist without a teacher and relates solely to the meaning-making process of the individual's direct experience. Knowledge is continuously gained through personal and environmental experiences.

The Experience API (xAPI) is an eLearning software specification that allows learning content and systems to interoperate in a manner that records and tracks all types of learning experiences [7]. Results of learning experiences are stored in a Learning Record Store (LRS), which may exist in a traditional Learning Management System (LMS), or on its own. xAPI does not require a learning experience to take place in any particular medium (mobile, desktop, tablet), offline or online, or in any particular system. By collecting and analyzing xAPI for a specific learner, a picture of the learner's activities, achievements, competencies and interests can be created, drawing on experiences using multiple devices and activities [8]. Exposing data through the xAPI provides a means for interoperability and enables innovation of learning content, experiences, and systems [9].

The use of xAPI with mobile devices can leverage learning by enabling the capturing of activities from diverse learning experiences that exploit the capabilities of the mobile platform. This may lead to new kinds of learning experiences and a much wider adoption of mobile-based performance support [9]. For instance, if a learning design was predicated on students taking pictures of examples of a particular phenomenon, and then sharing and discussing these with other students, the xAPI enables the various activities to be recorded and tracked in an LRS in the form of activity streams. Later, the teacher could retrieve this information, initiate a discussion with the students in class, or even improve the learning design of the game according to the results to fit better the needs of the learners. These are only some of the many uses and benefits of adopting xAPI.

In this paper, we present the design and implementation of PlayLearn, a platform for the management of learning experiences in mobile location-based games, consisting of: (a) an authoring tool, supporting creation and management of games, scenario editing, user interface customization and organization of gaming activities, and (b) a mobile application, compatible with most state-of-the-art mobile devices and platforms, supporting the play of games created by using the authoring tool. Our implementation supports the xAPI, allowing the activities that happen as part of gaming experiences to be recorded, tracked and shared.

The rest of this paper is structured as follows: Section II presents the related work. Section III specifies the model for describing educational games. Section IV presents the architecture that has been designed and implemented, while Section V provides some insight on the implementation of the platform. Finally, Section VI summarizes the presented work and sketches some perspectives for future extensions.

II. Related Work

Geocaching [10] is an outdoor treasure hunt activity where players try to find hidden caches using coordinates. Activities are supported by a desktop and mobile application, enabling the searching and previewing of geocaches, and providing basic navigational assistance. PlayLearn, on the other hand, supports a variety of gaming activities and allows the user to create and share his own game. Moreover, it provides the means to organize location-based events, offering an in depth gaming experience to the participants.

WHAIWHAI [11] is an interactive story based on a gaming platform and developed to offer a way of exploring the less touristic and unknown city places. Although games are customizable in terms of difficulty, time limit and number of players, WHAIWHAI is limited to exploration activities. On the contrary, PlayLearn provides tools for creating and playing various types of games, exploited under different contexts and targeting users of certain age groups.

Tourality [12] is a location-based game where the user's goal is to visit several spots of certain interest. In multiplayer mode, the game focuses on time, requiring the user to maintain a high score and compete either by participating in a team or alone. Although it enhances competitiveness and cooperation, it focuses strictly to these characteristics leaving aside any in depth educational activity that a game may provide. Using PlayLearn, on the other hand, a user is able to create games consisted of a variety of educational activities. The player apart from visiting places, is able to browse educational material, answer riddles, perform metadata annotation tasks and search for hidden treasures.

AnswerTree [13] is a collaborative mobile location-based educational game designed to teach 8-12 year old players about trees and wildlife. The game is designed around collecting virtual information cards about notable trees by answering questions. Collaboration is encouraged by the fact that solutions to these questions are obtainable through sharing knowledge with other cardholders. Apart from this, AnswerTree is a static game targeting users of specific profile, interests and goals.

ARIS [14] is a platform for creating and playing mobile games, tours and interactive stories. Its authoring tool provides the ability to specify the game location and create quests, but lacks of functionality for organizing gaming events. Unlike PlayLearn's authoring application, the game creator cannot customize the user interface of the game. Another important limitation is that ARIS game player application is designed as a mobile application available only for iPhones. On the contrary, we provide a native game player for the majority of mobile phone operating systems including iOS, Android and Windows. PlayLearn's game player application is also available through the browser.

III. Model

The model for the creation of location-based educational games has been designed with flexibility and extensibility in mind, to be able to describe a variety of game types. Figure 1 presents the core model entities of Playlearn.

The *Users* of the system are the game creators and the

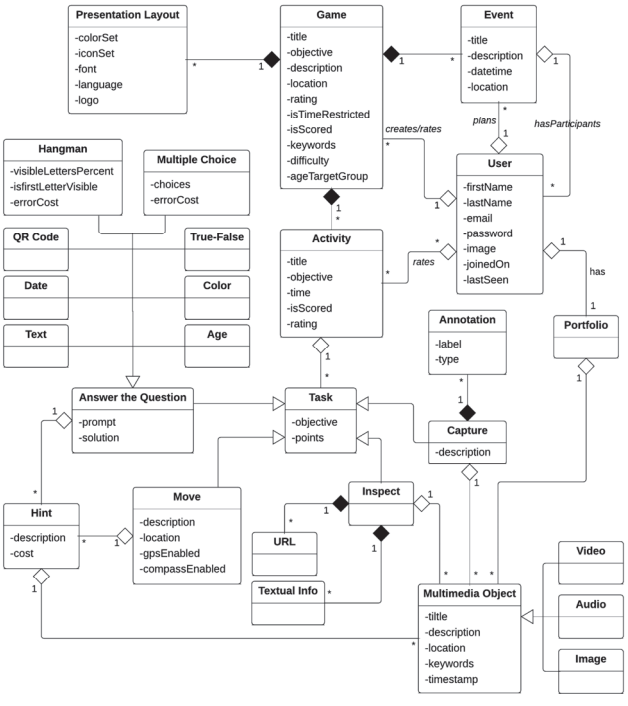

Figure 1. PlayLearn core model

players. The *Game* class represents the games which can be created, managed and played by *Users* using PlayLearn. A *Game* has a goal and consists of several descriptive and presentation metadata, while it is possible for the creator to adapt its *Presentation Layout* (color set, icon set, fonts, language, logo). A *Game* consists of a sequence of *Activities* with certain objectives that have to be tackled by the player to support the game goal. *Activities* consist of *Tasks* which are considered as the actions to be performed in order to achieve the respective activities' objectives. There are currently four types of *Tasks*:

- *Move:* the user needs to navigate in order to reach a specific destination.
- *Inspect:* the user has to read a document or any piece of *Textual Information*, view an image/video or listen to a sound (*Multimedia Object*).
- *Answer the Question:* the user should provide an answer for a given question. The type of question can be *Multiple Choice, True–False, Text* (the player should provide his answer in plain text), *Hangman* (the player needs to find a hidden word), and *QR Code* (the player should scan a pattern code, after searching for it in a specified location range).
- *Capture:* the user has to take a photo, or record a video/sound. The task may require the *Annotation* of the captured *Multimedia Object* with metadata.

The *Multimedia Object* class represents multimedia objects of type: video, image, text, and audio. Each type is depicted as a different class, holding its own descriptive attributes. The *Audio* class represents the multimedia objects of type sound. Optionally, it contains a GPS point specifying the location that has been recorded. The *Image* class represents the multimedia objects that are of type image. Optionally, it contains a GPS point with information about the location where it was captured, as well as other descriptive metadata. The *Video* class represents the multimedia objects that are of type video. Optionally, it contains a list of GPS points with information about the location where it was recorded, as well as any other information provided by the camera. The list of GPS points can be used to recreate the path that the user took for the duration of the capturing. Each User has a *Portfolio* that corresponds to a library with *Multimedia Objects*, which are used in games creation.

Games can be used for the organization of gaming *Events*, which can be created and shared by *Users* in order to promote gaming activities in certain locations. Each *Event* may refer to a specific game, location and date/time. Additionally, the list of the event participants can be either open or restricted to specific user groups. Since Games can be bounded to a specific place, such information along with game classification, target group, rating and difficulty can be used for game searching and filtering.

Figure 2 presents a simple game example for a visit of preliminary school pupils to a Botanical garden to explore olive trees, consisting of three activities corresponding to the pre-visit, visit and post-visit phases.

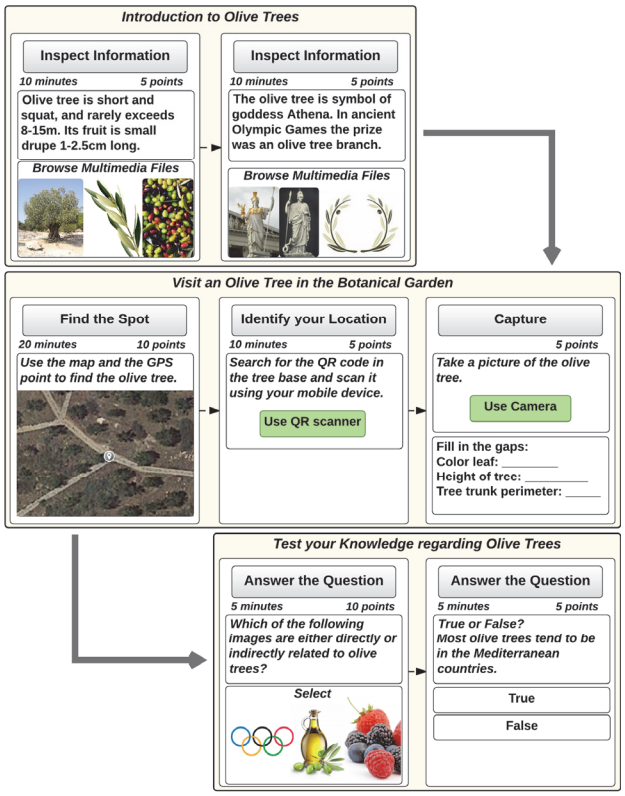

Figure 2. A simple game example for a visit to the Botanical garden

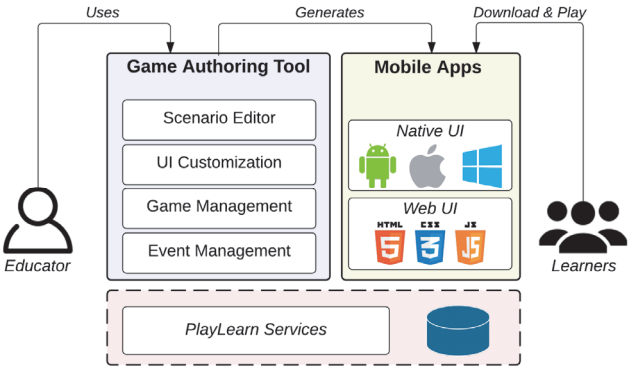

Figure 3. PlayLearn conceptual architecture

IV. ARCHITECTURE

Built as a web application, the system adopts the Rich Internet Application (RIA) principles, which promote the development of web applications as desktop applications, performing business logic operations both on the client and server side. The client side logic operates within the web browser running on a user's computer, while the server side logic operates on the web server hosting the application.

Figure 3 presents the conceptual architecture of the PlayLearn platform. The main parts of this architecture are: (a) The *Game Authoring Tool* used by the Educator for the creation of games, providing functionality for game scenario design, game and event management, and UI customization, and (b) the *Mobile Apps* providing both Native UI to support

mobile devices with different OS (Android, iOS, Windows) and a Web UI, which are used by the Learners to download and play the games created by the Authoring Tool during events organized by the Educators. The *Game Authoring Tool* and the *Mobile Apps* are supported by a number of services and repositories that are described in detail in the following paragraphs.

The overall system architecture is presented in Figure 4. For the development of the application we adopted several design patterns [15]. The use of well-established and documented design patterns speeds up the development process, since they provide reusable solutions to the most common software design problems [16][17]. The Model-View-Controller (MVC) design pattern [18][19] and the Observer pattern were used on the client side, and a multi-tier architecture was implemented on the server side, which are described in the following sections.

A. *Server Side*

The Server Side part of our platform adopts a multi-layered architectural pattern consisting of three basic layers (Figure 4): *Service Layer*, *Business Logic Layer* and *Data Layer*. This increases system's maintainability, reusability of the components, scalability, robustness, and security.

The *Service Layer* controls the communication between the client-side logic and the server-side logic, by exposing a set of services (operations) to the client-side components [20]. These services comprise the middle-ware concealing the application business logic from the client and have been built as RESTful [21][22]. The basic system services are:

- *CRUD Services*, facilitating the creation, retrieval, update and deletion of games, events associated with games, and users.
- *External Access Services*, providing the means for the client side and external applications to use the data of the system.
- *Multimedia Access Services*, enabling access to the uploaded multimedia files and their thumbnails.
- *LRS Services*, facilitating the persistence of statements generated by the use of xAPI.

The *Business Logic Layer*, also known as *Domain Layer*, contains the business logic of the application and separates it from the Data Layer and the Service Layer. In more detail:

- The *Game Management Module* is responsible for the game management, as well as for the activity and task (de)composition in our system.
- The *Event Management Module* is responsible for the event management.
- The *Search Management Module* handles the search and filter queries posed on our dataset and delivers the obtained results to the appropriate component of the Service Layer.
- The *Multimedia Management Module* is responsible for managing the persistence and serving of multimedia files, as well as for performing basic metadata extraction and thumbnail generation.
- The *LRS Parser/Manipulator Module* is responsible for the persistence and accessing of gaming results that have been collected and obtained during a gaming activity.

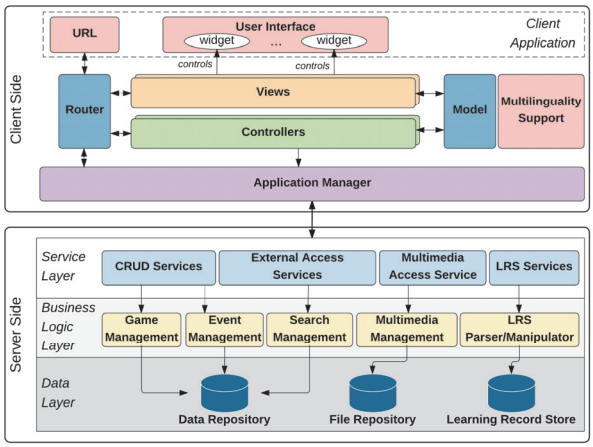

Figure 4. PlayLearn system architecture

The *Data Layer* accommodates external systems used to index and persist both data and multimedia files, such as:

- *Data Repository*, storing all the data of the system.
- *File Repository*, persisting multimedia files and thumbnails.
- *Learning Record Store*, archiving the collected results of gaming activities.

B. *Client Side*

The Client Side of PlayLearn application is responsible for the interaction with the user. It refers to both the authoring tool (web application) and the player application (mobile application). All the actions performed by an individual are handled by client side logic which undertakes the presentation of the information as well as the communication with the server. In order to achieve a high level of decoupling between the components forming the client logic, we adopted the Model View Controller (MVC) design pattern, as well as the Observer pattern. The use of MVC introduces the separation of the responsibilities for the visual display and the event handling behavior into different entities, named respectively, View and Controller.

The *Model* refers to the business objects which are being used by our system. When the system needs to present information about a business object, the client side requests the respective information from the server side using the services that the later exposes. Similarly, when an update on the Model needs to be persisted, the client side sends the updated Model to the server side, triggering the indexing and storage of the business objects by the appropriate modules and external systems.

The *Views* are responsible for the presentation of information in the user interface. Each view controls a number of widgets on the application's graphical user interface. It consists of several handlers that are responsible for listening user actions, as well as HTML templates that define the presentation of the widgets.

The *Controllers* are the modules that respond to the user input and interact with the Views in order to perform any change on the user interface. Furthermore, they maintain the Model and change it appropriately. Every View has a dedicated Controller managing, handling and propagating any changes that are to be performed or have already been performed to the user interface. Moreover, there are several cases where a "composite" Controller manages a number of other Controllers in order to create complex widgets.

The *Router* is used for deep-linking URLs to controllers and views. It manages the URL of the client browsers, providing a different path to each distinct interface, without raising a browser event that will force a reload on the whole page. When the URL changes the Router analyzes the new path and handles the transition to the new View. This is performed using mappings between the different URLs supported in the system, the Controllers and the Views.

The *Multilinguality Support* module manages the translation of the user interface elements through the use of certain configuration files. The process is easily adaptable and the system can be extended to support any language with minimum effort. Currently the graphical user interfaces of the system have been translated in English and Greek.

V. IMPLEMENTATION

PlayLearn has been successfully implemented as described in previous sections. Its server side is based on Java and makes extensive use of the Spring Framework in order to tackle certain backend aspects like data access. For persistency storage, MongoDB is used. One of the reasons that led us to choose MongoDB is that most of our data does not conform to a rigid relational schema. Hence, we cannot bind it in the structure of a relational database and we need flexibility. Due to the fact that MongoDB allows us to store parts of our data in different forms with minor effort, our back end is considered compatible with LRS and xAPI.

Regarding the client side, both the game authoring tool and the player application are based on the latest web-application standards, rely on the JavaScript programming language and make extensive use of the AngularJS framework. Moreover, they have been created in order to match different user requirements, and thus their user interfaces are implemented differently in order to match the goals of their stakeholders. Apart from JavaScript, the user interface layout has been built using HTML5 and CSS3. The player application has been packaged as a native mobile application with the use of Phonegap, making it compatible with all the major mobile platforms. This allows our application to run without the need of internet access, or loading its source each time that the user accesses it. Additionally, the use of Apache Cordova allowed us to provide more functionality by using various native platform features that are otherwise unavailable to web applications.

Figure 5 presents the graphical user interface of the game authoring tool. Specifically, it shows the use of the scenario editor while the user creates a new task of type “Capture” to populate an already existing activity of a game named “Explore Olive Trees”. The left side of the user interface is

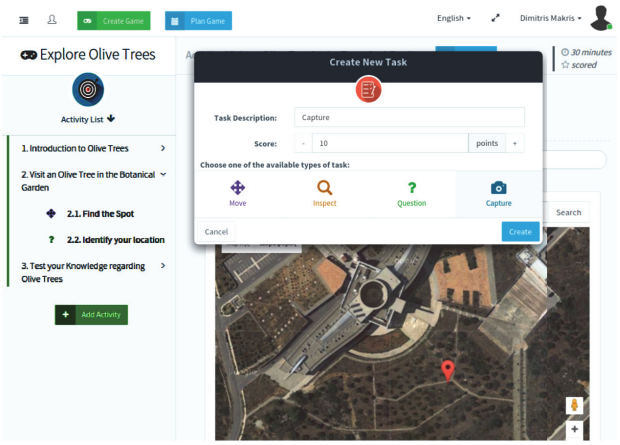

Figure 5. Game authoring tool (web application)

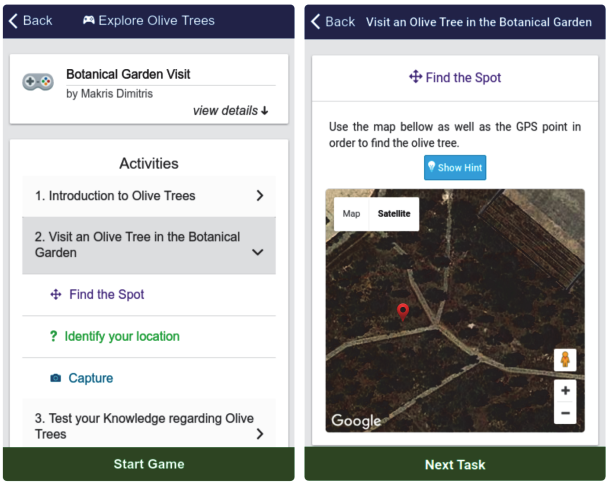

Figure 6. Player application (mobile application)

used for presenting the game activity list, while the right side is used as the main working area for customizing activity tasks. The top bar is used for the main menu inspection, user interface customization (in terms of language and layout), and personal settings editing. Figure 6 presents the graphical user interface of the mobile application that was generated by the game authoring tool in Figure 5. The screenshot on the left side shows the preview presented to the user just before starting the game. It includes basic information such as the event, the game, the creator and the list of activities that need to be completed. On the other hand, the screenshot on the right has been taken during playing the game and shows an activity task of type “Move” requiring the user to use the map in order to navigate to a specific spot of interest.

VI. CONCLUSIONS

In this paper, we presented the design and implementation of PlayLearn, a platform for the development and management of learning experiences in mobile location-based games. PlayLearn provides: (a) an authoring tool supporting the creation and management of

games, scenario editing, user interface customization and organization of gaming activities, and (b) a mobile application supporting the play of games created using the authoring tool. The player application is compatible with most state-of-the-art mobile devices/platforms, while both tools have been designed with flexibility and extensibility in mind. The underlying model supports a great variety of basic building blocks that can be exploited by a user in order to create a great range of complex and structured location-based gaming experiences. The activities that happen as part of these gaming (learning) experiences are recorded, tracked and shared in a LRS by supporting xAPI. Both applications have been evaluated for their usability through the use of pluralistic walkthroughs and extensive paper prototyping.

The work presented in this paper is part of the EVANDE project learning infrastructure. EVANDE (Enhancing Volunteer Awareness and education against Natural Disasters through E-learning) [23] is a European project co-funded under the Union Civil Protection Mechanism. It aims to create a new learning tool to educate and train civil protection volunteers and local authorities' staff through identification of best practices and knowledge, develop a web platform and tools to host e-learning courses, games and training activities, as well as to organize and implement local-based dissemination and training actions. Promoting exploration with mobile location-based educational games is vital in teaching players about specific areas where a crisis can happen [24] and practicing on best practices to prevent and respond on the most effective manner benefiting the maximum from human and technical resources.

Our future work includes the following: (a) direct connection with repositories persisting observational data like GBIF, BioCASE and Natural Europe [25], in order to easily enrich the educational content of a game, (b) direct connection with well known cultural heritage repositories like Europeana [26], and (c) enabling the creation of observations during the play of a game [27], as well as their further dissemination to related repositories.

ACKNOWLEDGMENT

The work presented in this paper is partially funded in the scope of the EVANDE (Enhancing Volunteer Awareness and education against Natural Disasters through E-learning) project, co-funded under the Union Civil Protection Mechanism, Grant Agreement ECHO/SUB/2014/693261.

REFERENCES

[1] Location-based game. [Online]. Available from: https//en.wikipedia.org/wiki/Location-based_game.

[2] N. Avouris and N. Yiannoutsou, "A review of mobile location-based games for learning across physical and virtual spaces," Journal of Universal Computer Science, 2012, vol 18,(15), pp. 2120-2142.

[3] De. Souza, E. Silva, and G. C Delacruz, "Hybrid Reality Games Reframed Potential Uses in Educational Contexts," Games and Culture, Sage Publications, 2006.

[4] D. Spikol and M. Milrad, "Physical activities and playful learning using mobile games," Research and Practice in Technology Enhanced Learning, 2008, vol. 3 (3), pp. 275–295, doi: 10.1142/S1793206808000562

[5] P. Arapi, N. Moumoutzis, M Mylonakis, and S. Christodoulakis, "A Framework and an Architecture for Supporting eLearning Applications on top of Multimedia Digital Libraries," DELOS Conference on Digital Libraries, 2007.

[6] D. A. Kolb, Experiential learning. [Online]. Available from: https://en.wikipedia.org/wiki/Experiential_learning.

[7] J. Delano and A. Shahrazad, "Using the Experience API to Track Learning," Infoline (Numbered), vol. 1304, American Society for Training & Development, 2013.

[8] P. Durlach, J. Poltrack, K. Murray, and D. Regan, "Modernizing Education," MT2, 2013, vol. 18(5).

[9] K. Murray, P. Berking, J. Haag, and N. Hruska, "Mobile Learning and ADL's Experience API," Connections 12(1), 2012, pp. 45–49.

[10] Geocaching. [Online]. Available from: https://www.geocaching.com/

[11] WHAIWHAI. [Online]. Available from: http://www.whaiwhai.com/en/

[12] Tourality. [Online]. Available from: http://www.tourality.com/

[13] A. Moore, J. Goulding, E. Brown, and J. Swan, "AnswerTree – a Hyperplace-based Game for Collaborative Mobile Learning," mLearn Conference, 2009.

[14] ARIS Games. [Online]. Available from: http://arisgames.org/

[15] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[16] D. Alur, D. Malks, and J. Crupi, Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition). Prentice Hall, 2 edition, 2003.

[17] M. Fowler, Patterns of Enterprise Application Architecture. Addison-Wesley, Boston, ersteauflage edition, 2003.

[18] T. Reenskaug, Models - Views - Controllers. Technical report, Technical Note, Xerox Parc, 1979.

[19] T. Reenskaug, The Model-View-Controller (MVC) Its Past and Present, 2003.

[20] G. Alonso, Web Services: Concepts, Architectures and Applications. Springer, 2004.

[21] C. Pautasso, O. Zimmermann, and F. Leymann, "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision," The 17th international conference on World Wide Web, ACM, New York, 2008, pp. 805–814.

[22] R. T. Fielding and R. N. Taylor, Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology (TOIT), 2002, 2(2):115–150.

[23] EVANDE . [Online]. Available from: http://www.evande.eu

[24] I. Di Loreto, S. Mora, and M. Divitini, "Collaborative Serious Games for Crisis Management: An Overview," The IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012.

[25] G. Skevakis, K. Makris, V. Kalokyri, P. Arapi, and S. Christodoulakis, "Metadata Management, Interoperability and Linked Data Publishing Support for Natural History Museums," International Journal on Digital Libraries (IJDL), 2014.

[26] K. Makris et al., "Federating Natural History Museums in Natural Europe," The 7th Metadata and Semantics Research Conference (MTSR), Special track on Metadata & Semantics for Cultural Collections & Applications, Thessaloniki, 2013.

[27] C. Tsinaraki, G. Skevakis, I. Trochatou, and S. Christodoulakis, "MoM-NOCS: Management of Mobile Multimedia Nature Observations using Crowd Sourcing," The 11th International Conference on Advances in Mobile Computing & Multimedia (MoMM), Vienna, 2013.