Lessons Learned from the Development of a Mobile Learning Game Authoring Tool

Pierre-Yves Gicquel, Iza Marfisi-Schottman, Sébastien George

Draft: Pierre-Yves Gicquel, Iza Marfisi-Schottman, Sébastien George, "Lessons Learned from the Development of a Mobile Learning Game Authoring Tool », *Proceedings of the Games and Learning Alliance Conference, GALA*, 27th to 29th of November 2019, Athens, Greece.

Lessons Learned from the Development of a Mobile Learning Game Authoring Tool

Pierre-Yves GICQUEL[1], Iza MARFISI-SCHOTTMAN[1] and Sébastien GEORGE[1]

[1] Le Mans Université, EA 4023, LIUM, 72085 Le Mans, France
{ pierre-yves.gicquel, iza.marfisi, sebastien.george}@univ-lemans.fr

Abstract. Students and schools are increasingly equipped with smartphones and tablets. These mobile devices can enhance teaching in many ways. Mobile Learning Games (MLGs) for example, have shown great potential for increasing student's motivation and improving the quality of situated learning. For the past few years, the research community has been working on authoring tools that allow teachers to create and distribute their own MLGs. The development of these authoring tools is challenging and time consuming and even more so if the objective is for these tools to actually be used in classrooms. The Design-Based Research (DBR) paradigm was precisely developed to address these central issues of Technology Enhanced Learning. It involves co-designing and testing with end-users from the beginning of the project. Although DBR increases the acceptance of new educational tools, it also adds several challenges, including the complexity of involving teachers and students in real-world situations and creating several versions of the tools that will be improved iteratively. In this paper, we aim at providing design principles and practical guidance on the way to develop such authoring tools, based on our experience. We conclude on lessons learned from this project and discuss some systematic issues we faced.

Keywords: game-based learning, authoring tool, mobile learning, situated learning, design-based research

1 Designing Mobile Learning Game Authoring tools

Authoring tools for Mobile Learning Games (MLGs) provide teachers with a simple way of designing and using applications that fit their specific needs. Teachers especially like using MLGs for educational outing and field trips. Indeed, MLGs combine all the ingredients necessary to attract students' attention and engage them in learning activities [1]–[3]: game mechanics such as competition, rewards and exploration, situated learning in real contexts and the physical effort necessary to go to the right place. Several MLG authoring tools have recently been proposed [4], [5]. The conception, development and administration of such authoring tools are arguably challenging and time-consuming tasks for researchers.

The first challenge is related to the complex nature of MLGs itself. Despite an active research community, no MLG model has reached a consensus and the know-how required to select the best game mechanics to create pedagogically effective MLGs

remains difficult to pinpoint. In addition, the main objective of authoring tools is to fit the needs of a large number of users. The MLG model used by the authoring tool therefore needs to be as generic as possible while still providing enough structure and guidance to help teachers.

Moreover, designing authoring tools that actually meet the needs of real teachers is very challenging. Many researchers such as Virvou and Eythimios [6] were confronted to the difficulty of adapting the user interface to the needs of teachers because they waited for the end of the project to involve end-user. The use of Design-Based Research (DBR) seems like a good alternative to maximize the acceptability of authoring tools in class [7] [8]. This movement started in the 1990s and was developed to address several central issues of Technology Enhanced Learning (TEL) such as the need to find solutions with the end-users (teachers and students) and the need to study learning phenomena in the real world rather than in a laboratory.

Although DBR is a powerful paradigm for addressing these needs, it also brings major technical challenges for the development of TEL tools. First of all, DBR implies the co-design of tools with the end-users in an iterative way. Researchers therefore need to develop not one, but several prototypes, in close collaboration with the end-users. In the case of MLG authoring tools, this means that the authoring tool and the underlying MLG model will be modified, several times, until they meet the teachers' needs. DBR also emphasizes the fact that the tools should be tested in real-world learning environments. The MLGs, designed by teachers with the authoring tool, therefore need to support many simultaneous connections during field trips. Finally, DBR strongly depends on the help of teachers. If they want to benefit from their full collaboration, researchers must provide robust tools that teachers will be able to use, long after the research project is over.

As we have shown, the design of MLG authoring tools with DBR raises many technical challenges. Yet, to our knowledge, very few researchers have provided theoretical or practical guidelines for the development of such tools. In this paper, we propose design principles and an example of software architecture, based on our experience developing a MLG authoring tool, named MOGGLE, with DBR.

In the second section of this paper, we will detail the constraints related to developing authoring tools and supporting DBR. In the third section, we propose several design principles that are adapted to these constraints and illustrate them with the architecture we set up for MOGGLE. Section four provides elements of validation of our proposition through the numerous iterations and modifications that MOGGLE was put through before reaching a final version. Finally, in section five, we conclude on lessons learned from the development of MOGGLE, and discuss systematic issues we faced at different stages of the project.

2 Challenges Brought by Design Based Research

Design-Based Research is “a systematic but flexible methodology aimed to improve educational practices through iterative analysis, design, development, and implementation, based on collaboration among researchers and practitioners in real-

world settings, and leading to contextually-sensitive design principles and theories" [12]. The use of DBR is paramount for designing tools that meet the needs of teachers and that will actually be used in class. However, DBR also comes with a set of constraints that complicates the development of these tools. In this section we identify the constraints that DBR sets on the development of MLG authoring tools.

2.1 User-Centered Design

DBR gives a central position to the end-users. The design of MLG authoring tools should therefore be carried out with teachers, who will use the authoring tool to create MLGs, and students, who will play these MLGs. Usability and utility are therefore a central concern. In terms of usability, the authoring tool should be **simple enough for teachers to create MLGs rapidly**. In order to feel at ease with these tools and confident enough to use them in class, teachers also need to be able to control what the final MLG will look like for the students. This is especially true when designing MLGs, that are new to them. This implies developing a **preview or testing system** that can immediately show what the final MLG will look like. In addition, like any educational material, MLGs need to be adjusted before reaching the satisfactory final version. The authoring tool therefore needs to offer **the means to modify MLGs** to facilitate incremental design. Finally, authoring tools need to offer maximum utility to teachers. In other terms, they should allow them to create MLGs that are adapted to their educational field trips. Such educational outings are used in many domains such as geology, botany, history, archaeology but also arts and sports. For many of these domains, it is important for the MLGs to offer specific activities such as plant and rock identification or augmented reality. The authoring tool therefore needs to **offer the possibility of adding new types of domain specific activities**.

2.2 Iterative Design

DBR supports the idea that the theoretical learning model and the learning tools are gradually improved through iterative co-design and testing with end-users. In the context of MLG authoring tools, two models are at stake: the MLG model and the authoring tool model. Indeed, there are many different ways of designing an authoring tool based on the same MLG model. Given the fact that DBR encourages multiple iterations to reach a satisfying version of these models and the limited resources available for research projects, the **initial models need to be highly modular and expandable** in order to build on them and not start from scratch each time.

2.3 Tests in Real-World Situations

Finally, the last important constraint that is brought by DBR, is the fact that learning tools need to be tested in real-world situations. This means that the MLGs designed by the teachers, with the authoring tools, need to be **robust** enough to withstand field trips with 30 or more students, who will probably not use the MLGs as they were initially intended to be. Furthermore, the genericity of MLG authoring tools needs to

be tested with teachers in several different domains. Given the complex organization of field trips, it is not rare for several tests to overlap. It should therefore be possible **to set up independent instances of MLG authoring tools** for each user group.
In the next section, we propose several design principles that support these three major constraints and illustrate them with the software architecture uses for MOGGLE.

3 Design Principles to Support Design-Based Research

MOGGLE (MObile, Geolocated Games for Learning) is composed of two applications: **MOGGLE-Editor** (referred to as Editor) that allows teachers to create their MLGs and **MOGGLE-Player** (referred to as Player), which allows students to play these MLGs (Figure 1). The instances of MLGs are stocked in a shared database.

Fig. 1. MOGGLE MLG authoring tool

The first key decision, taken at the beginning of the project, was to use **only web technologies** to develop the Editor and the Player. There are several reasons for this choice. First of all, web applications can be accessed from all types of devices and operating systems (PC, Mac, Android, IOS, Windows tablets…). This is very advantageous, given the fact that the types of devices available in schools are very heterogeneous. Secondly, web applications do not require complex installation. Users simply need to have access to the Internet and connect themselves via their usual web browser. This definitely facilitates the use of tools for teachers and students in a learning context. One might think that the need to have a connection can be a constraint but some web technologies allow connection breaks by using a cache system (local storage) while allowing data to be transferred when the connection is restored. Furthermore, from an ergonomic point of view, responsive web apps are getting closer and closer to native applications. Finally, web applications can be updated very easily without requiring any actions from the end-users. Knowing that DBR implies many iterations and updates, this seemed like an important advantage. The choice of web technologies was justified for MOGGLE but seems well adapted for the development of any TEL tool, especially if they are designed with the DBR paradigm. We therefore propose **to only rely on Web technologies as our first design principle**.

Below, we will propose seven other design principles to support the constraints detailed in the last section. Note that we will not detail the basic design principles de-

rived from software engineering such as system versioning, server security or programming design patterns.

3.1 User-Centered Design

Interactive preview of elements being designed

As mentioned in the previous section, it is important for teachers to able to preview what their MLG will look like on students' smartphones and test its interactions. Ideally, this preview should be available while they are designing the elements, so that they can adjust them directly in order to obtain the desired interface. We therefore advocate the integration of an **interactive preview of the MLG elements** into the MLG authoring tool.

This design principle was used for MOGGLE. As shown in the figure 2, teachers can test the player interface (on the top right) while they are designing the MLG activities. To offer this interactive preview, we used the same web component for the Player and the Editor's preview system. The interactive preview is updated, in real time, when the teachers fill in the form on the left. This was done with two-way data binding. In *HTML/JavaScript*, data binding refers to the binding of a variable declared in the main script directly to an HTML tag's attribute (i.e. the attribute "value" of an input tag *<input value={{boundVariable}}>*). When the attribute is modified, the value of the variable is also modified. By using the same variable in another tag's attribute element, real time synchronization between functionally independent components can be achieved.

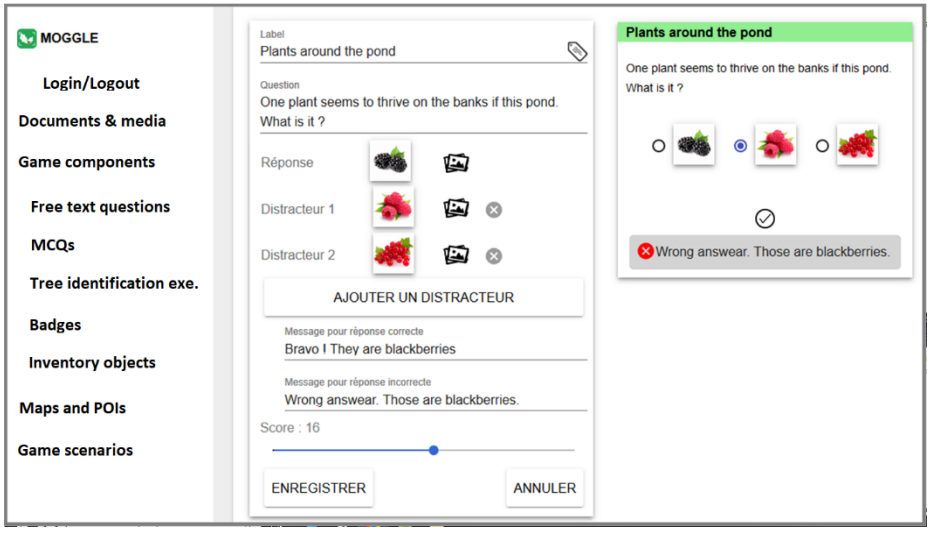

Fig. 2. MOGGLE-Editor interface for creating an MCQ with the interactive preview

Incremental design

As for any type of educational content, teachers will need to adjust their MLG's scenario several times, until they reach the desired product. For instance, after testing the

MLG, the teacher might want to change an image or add educational feedback. It is therefore important, in terms of user experience, to support incremental design by **providing functionalities to incrementally edit and test MLGs**.

This design principle was implemented in MOGGLE using the dual server architecture presented in Figure 3. Each instance of MOGGLE is composed of two separate virtual machines: the Editor server and the Player server. The Editor and the Player both share the same MLG model and communicate through a unique MLG database. When an element is created or updates in the Editor, it is immediately updated in the database which is used by the Player. Teachers can therefore immediately test the new versions of their MLGs, while editing them.

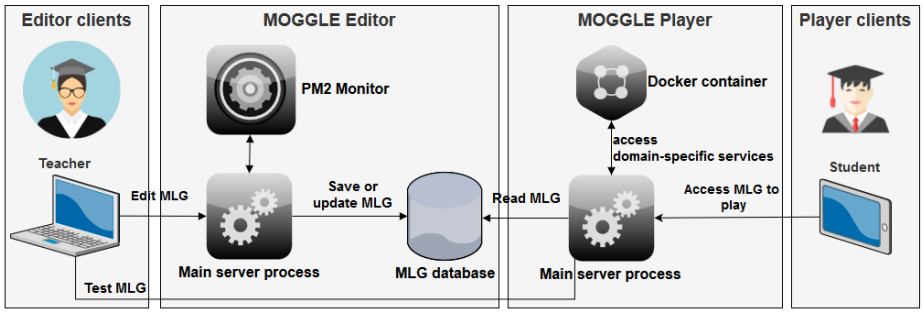

Fig.3. MOOGLE architecture

Integration of Domain-Specific External Services

MLG authoring tools aim to be used in a large variety of real-world learning environments. In order for these tools to be widely acceptance by teachers, they must be adaptable to their numerous educational needs. We therefore propose the following design principle: **the architecture of MLG authoring tools should allow the integration of domain-specific external services.**

Implementing this design principle into MOGGLE was a necessity. This tool was developed in the context of the ReVeRIES project[1], for which the main objective was to create outdoor botanical games. One of the most important skills in botanical science is the ability to identify the species of a tree. In order to create MLG activities that would help students master tree identification, we integrated the *FOLIA* application[2] [9] into MOGGLE. *FOLIA* performs tree recognition based on pictures of a leaf, using shape extraction algorithms. The integration of *FOLIA* offered several engineering challenges. First of all, it was not possible to run *FOLIA* on the client side because it is a binary executable file. It also requires multiple dependencies that need to be compiled manually. In order to avoid unnecessary dependencies and a tedious compilation process for each new deployment, we chose to encapsulate *FOLIA* in a *Docker* container accessible as a web service (figure 3). *Docker* containers are similar to virtual machines, they encapsulate an operating system and applications and can be run

1 http://reveries-project.fr/
2 https://apps.apple.com/fr/app/folia/

on a host system with precise resources allocation (CPU, RAM, disk space). This method can easily be applied to other existing services such as speech recognition or Augmented Reality to create domain specific MLG activities without compromising the security, robustness and genericity of the authoring tool.

3.2 Iterative Design for Mobile Learning Game Authoring Tools

Extensible Data Model and Document-Based Data Persistence

DBR encourages iterative co-design. As shown in section 2, it is therefore crucial for the MLG model to be designed with future extensions in mind. Such extensions inevitably require the flexibility of the data persistence mechanism. Relational databases are not well adapted for handling structural modifications. Indeed, their schemas are fixed at the beginning and are very time consuming to modify. NoSQL databases, on the other hand, do not store the data in fixed schemas. Instead of using tables, they store the data in *documents*. A *document* is an association of key-values, similar to *JSON*, with loose constraints on the existence of keys and on the type of the values. Thus in a NoSQL database, it is possible to add new kinds of *documents* without impacting what previously exists. As a design principle we therefore propose that MLG authoring tools should use **extendable MLG models and a document-based data persistence system.**

In the case of MOGGLE, we used a highly modular and adaptable MLG model, described in previous work [10], and the document-oriented database *MongoDB* that offers a high-level performance on a large number of documents.

Modular Interfaces

As seen previously, iterative co-design will refine the MLG model but also the Editor's and Player's user interfaces. The interfaces should therefore offer **maximum modularity by minimizing functional dependencies** between its different parts.

This design principle was implemented in MOGGLE by using web components to represent each level of granularity of a MLG. At the lowest level, MLGs are composed of elementary resources (images and videos) that can be combined into multimedia documents. These resources and documents can then be used to create situated activities such as multiple-choice questions, free text questions or tree identification exercises. These activities can then be associated to a point of interest, at a specific location, in order to create MLG units. Finally, these units are organized to form a MLG. Each level being independent, it is easy to modify the design process of a MLG according to the user needs. Technically, this design process is supported by the use of *web-components* that allow the definition of *custom elements* which are usable directly as HTML tags (e.g. <*div/*>,<*span/*>). We defined tags for each level of granularity above, the high-level tags (e.g. <*mlg-unit/*>) being structured with generic fillers, to avoid functional dependencies with lower level tags.

3.3 Tests in Real-World Situations

Server Fallback Mechanism

A very important constraint brought by DBR is the necessity to provide teachers with a consistent access to the application. When performing daily changes on a code, even proper testing cannot guarantee against a failure of the server. Such failures could have serious consequences on the collaboration with end-user: if teachers lose access to the MLGs they spent hours designing, they will likely be reluctant to use the system again. As a design principle we therefore propose to **set up a fallback mechanism in case of server crash**.

In MOGGLE, we use a *PM2 Monitor* (figure 3) as process manager and fallback mechanism in case of server failure. *PM2* monitors the state and the resources used by a process in real time. Additionally, *PM2* can detect when a process is in failure state and perform a proper restart of this process.

Separate Group-Based Web Portals

Teachers put a significant amount of effort into understanding how to design MLGs. It is therefore important not to change the version of the tools they use during this process. The use of distinct group-based web portals allows to perform incremental changes on the MLG authoring tool without disturbing the end-users. Each web portal is accessible through a unique URL which is provided to a group of users (usually teachers from the same school). This way, two groups are able to work at the same time on two distinct versions of the authoring tool. We therefore advocate the use of **separate web portals for each users' group.**

This design principle was implemented in MOGGLE through the use of virtual machines. Each new machine, associated with a unique subdomain of our main domain reveries-project.fr (e.g. moggle1.reveries-project.fr).

4 Iterative Experimental Validation of MOGGLE

In this section we provide elements of experimental validation of MOGGLE regarding the DBR constraints we identified.

The first constraint was the User-Centered Design. We integrated a preview system for the MLG elements in the Editor and propose an incremental MLG design process, meaning that teachers can easily test the MLG being designed and modify them until they reach satisfaction. We gathered elements of validation for these principles through a qualitative evaluation of MOGGLE involving five teachers from various domains: history, geography, mathematics and foreign language. They were asked to design a MLG adapted to their teaching domain after a short introduction to MOGGLE. They all managed to create a MLG adapted to their specialty in less than an hour, with minimal assistance from the experimenter. The iterative MLG design process we proposed, proved to be useful since the teachers directly tested their games and updated them. The integration of domain specific tools was evoked by the history teacher. She questioned the possibility to integrate Augmented Reality layer of visualization to provide reconstitution of ancient buildings.

The second constraint is to use an iterative design in MLG authoring tool. We proposed a modular design pattern for the client side in order to simplify modifications resulting of these iterative circles. This modular design pattern based on web component was proven useful for integrating new features. The integration of *Youtube* videos was added after using MOGGLE with a natural park manager and the scoring mechanism was changed after another iteration. Other features, such as the integration of sound files in multimedia documents, were removed because they were never used. The modular design pattern, and the generic data structure we proposed, allowed us to add and remove these features with only minor changes.

The third constraint was to be able to test MOGGLE in real-world learning environments. We proposed the use of group-based web portal for that feature. These portals were deployed for each user group which facilitated the test of MOGGLE with several groups at the same time. We are currently in the process of deploying portals for two new groups of teachers. The ability to deploy up-to-date web portals on demands proved to be very useful to offer a private space to end-users but also for testing purposes.

5 Lessons Learned and Perspectives

In this article, we identify the three main constraints related to the design of Mobile Learning Game (MLG) authoring tools with Design-Based Research (DBR) and propose design principles. These design principles result from our experience in developing MOGGLE, a MLG authoring tool, designed in collaboration with several groups of teachers and that went through many iterative cycle of refinement.

The **first design principle we propose is to rely only on web technologies**. This design principle, justified in detail section 3. From this quite generic principle we derive seven design principles (Table 1). We illustrate technological implementation of each principle in terms of software architecture.

Table 1. Design principles for the development of Mobile Learning Game authoring tools

DBR Constraints	**Proposed design principles**	**MOGGLE software architecture**
User-centered design	Interactive preview of elements being designed	Web components & data binding
	Incremental MLG design	Shared database
	Integration of domain specific external services	Web services & Docker
Iterative design	Extensible data model & document-based database	Extensible MLG model & MongoDB
	Modular design patterns on the client side	Web components
Test in real-world situations	Server Fallback Mechanism	PM2
	Separate Group-Based Web Portals	Separate virtual machines for each user group

However, we faced some systematic issues, related to the incredibly fast evolution of web technologies between the project beginning and today [11]. At the beginning of the project, in early 2016, we choose *Polymer 1.0* that was recently released by Google and was very promising for the use of web-components. *Polymer 2.0* was released in the beginning of 2017 with no backward compatibility and *Polymer 3.0* was released in 2018 equally without backward compatibility. Due to limitations in human resources we had to keep using *Polymer 1.0* even though it is now deprecated.

A related systematic issue is the unpredictability frameworks trends. *VueJS* and *ReactJS* (which are also component-oriented framework) now largely surpass *Polymer*, although initially these two frameworks were not as promising. The question of framework evolution and deprecation creates many concerns in the web developer community while few publications focus on this question. This question raises important practical challenges in TEL and would deserve further investigations.

References

1. I. Marfisi-Schottman and S. George, "Supporting Teachers to Design and Use Mobile Collaborative Learning Games", in Proceedings of the International Conference on Mobile Learning, Madrid, Spain, 2014, pp. 3-10.
2. N. Bianchi-Berthouze, "Understanding the Role of Body Movement in Player Engagement", Hum. Comput. Interact., vol. 28, no 1, pp. 40-75, 2013.
3. F. Bellotti, B. Kapralos, K. Lee, P. Moreno-Ger, and R. Berta, "Assessment in and of Serious Games: An Overview", Adv Hum-Comp Int, vol. 2013, p. 1:1–1:1, janv. 2013.
4. Karoui, I. Marfisi-Schottman, and S. George, "Mobile Learning Game Authoring Tools: Literature Review, Synthesis and Proposal", in Proceeding of the Game and Learning Alliance Conference, Utrecht, Netherlands, (2016), pp. 281-291.
5. Karoui, I. Marfisi-Schottman, and S. George, "JEM iNVENTOR: a Mobile Learning Game Authoring Tool based on a Nested Design Approach", in Mobile Learning European conference, Larnaca, Cyprus, (2017), pp. 1-4.
6. M. Virvou, and A. Eythimios, "Mobile educational features in authoring tools for personalised tutoring." Computers & Education 44(1) pp. 53-68, (2005).
7. A. Collins, D. Joseph, and K. Bielaczyc, "Design Research: Theoretical and Methodological Issues", J. Learn. Sci. 13(1), pp. 15-42, (2004).
8. E. Sanchez, R. Monod-Ansaldi, C. Vincent, and S. Safadi-Katouzian, "A praxeological perspective for the design and implementation of a digital role-play game", Educ. Inf. Technol., 22(6), pp. 2805-2824, (2017).
9. S. Bertrand, G. Cerutti, and L. Tougne, "Bark Recognition to Improve Leaf-based Classification in Didactic Tree Species Identification", in International Conference on Computer Vision Theory and Applications, VISAPP, Porto, Portugal, (2017), pp. 435-442.
10. I. Marfisi-Schottman, P.-Y. Gicquel, A. Karoui, and S. George, "From Idea to Reality: Extensive and Executable Modeling Language for Mobile Learning Games", in Proceedings of the European Conference on Technology Enhanced Learning (EC-TEL), Lyon, France, 2016, pp. 428-433.
11. A. Pano, D. Graziotin, and P. Abrahamsson, "Rationale leading to the adoption of a JavaScript framework", arXiv preprint arXiv:1605.04303 (2016).
12. F. Wang, M. J. Hannafin, "Design-based Research and Technology-enhanced Learning Environments", Educational Technology Research and Development, Vol. 53, No. 4, pp. 5-23. (2005)