CHAPTER

κPAX: EXPERIENCES ON DESIGNING A GAMIFIED PLATFORM FOR SERIOUS GAMING

13

D. Riera, J. Arnedo-Moreno
Open University of Catalonia, Barcelona, Spain

1 INTRODUCTION

Games are something that all civilizations through time have in common. Even though the games themselves may vary, we all share the concept of play as a free activity outside of the "ordinary" or "serious" world. Nevertheless, in almost every case, they are conceived as greatly engaging activities. In fact, Huizinga defined human beings as *homo ludens* in 1955 (Huizinga and Hull, 1949). In recent years, the evolution of digital technologies and the rise of video-games as a medium has increased the prominence of the role of games in society, as well as their the recognition of capability to provide enjoyment and engagement (Brumels and Blasius, 2008). A new ecosystem has arisen, populated by the so-called Z generation (Prensky, 2001), people who have grown up playing video-games and feel at home in a society where games are prevalent in daily life. In a world with an increasing number of such citizens, the concepts of serious games and gamification arise almost organically. However, even though both concepts share several characteristics, and sometimes overlap in some parts of the literature, it is important to take into account that each one has its own meaning and goals.

On the one hand, serious games push entertainment into the background, looking for another focus, usually training or learning. It is very difficult to say precisely when serious games were born since, from ancient times, they have been a tool used to facilitate learning. From a formal standpoint, one of the first written references is Clark Abt's discussion of the term in his 1970s book, Serious games (Abt, 1970). There, he states "We are concerned with serious games in the sense that these games have an explicit and carefully thought-out educational purpose and are not intended to be played primarily for amusement", giving us a first attempt at a formal definition of the concept.

On the other hand, the goal of gamification is engagement. This concept has evolved considerably since its inception in 2002 (Pelling, 2011), Nick Pelling defined it as the application of a game-like accelerated user interface designed to make electronic transactions both enjoyable and fast. Nowadays, it also encompasses different aspects of a game experience, and has slowly started to encroach on playful experiences and anything game-related, such as serious games themselves. Nevertheless, the most widespread definition in the literature could be summarized as "the use of game design elements in non-game contexts" (Deterding et al., 2011).

In this paper, we present our experiences in the design and development of *κPAX*[1]: kudos, *Plataforma d'Aprenentatge en Xarxa* (*kudos—Network Platform for Learning*). This project was born from mixing the prevalence of social networks (eg, Facebook, LinkedIn, etc.), the rising number of electronic devices, portable and nonportable (eg, smart-phones, tablets, consoles, computers, etc.), and the entrance of serious games into the spotlight. The chosen means to join this triad is gamification, both as an engagement mechanic and, tracing back to the roots of its definition, as a video-game-like feedback mechanism. From here, the main goal has been twofold: (a) the design and implementation of an open source platform easily extensible by independent heterogeneous modules (including games, plugins, simulators, etc.), which can be the result of other research or innovation projects, or external developers contributions, and (b) the creation of a social network for games-based learning over the technological platform.

For a good understanding of the platform's purpose, it is necessary to introduce a bit on our institution, the *Universitat Oberta de Catalunya* (UOC, *Open University of Catalonia*)[2]. This is a public Internet-centered university created in 1994 in Barcelona. That makes it one of the first universities in the world to be completely online based, with more than 60,000 students and 50,000 graduates in fields like IT, psychology, education, law, management, communication and humanities, among others. The Internet-only nature of our institution and the digital literacy of our students makes it a good match for *κPAX*'s approach to e-learning.

The platform is built upon four ideas that take into account the idiosyncrasies of UOC:

1. Providing support for the training and assessment of certain skills or proficiencies
2. The current state of technology, which adds a new dimension to the spatial and temporal asynchronism of traditional online learning, thanks to the latest trends in mobile devices
3. Using tools based in ubiquitous technology, such as social networks, that allow people to be connected to others everywhere
4. An interest in applying the results of innovative projects across different fields, courses and studies.

In addition, the impact of this project goes beyond the UOC's campus, as *κPAX*, being a universal open learning tool, accepts internal and external users.

The result of our work has been the design and implementation of this technological platform and social network. The code of the platform has been published under the GNU General Public License (GPL) Version 2[3] so that any contributor can see, improve and complete the source code if so desired. Also, the addition of new games to the platform is fully open both to the UOC community (eg, faculty, management staff, students, designers, etc.) and to external programmers too. The main benefit of *κPAX* is that educators are provided with a platform that easily allows the distribution of serious games on different topics. Students get feedback on their skill mastery progress through their performance on all games in the platform.

[1]http://kpax.uoc.edu/elgg.
[2]http://www.uoc.edu.
[3]http://www.gnu.org/licenses/gpl-2.0.html.

This paper is structured as follows. First of all, Section 2 introduces the use of gamification and serious games as supports to the learning processes both from a conceptual and practical standpoint. *κPAX*'s conceptual design, based on the three pillars of system requirements, the specific gamification design and the website and interface definition, is presented in Section 3. Then, in Section 4, we discuss the decisions taken during the architectural design, mainly focusing on the services definition, the internal database and some key aspects on security. Finally, Section 5 concludes this work and provides some insights into our current and future work.

2 BACKGROUND

Serious games and gamification have many things in common, both being based on the application of a game design process to create a product that is not purely for entertainment. In this regard, they have been successfully applied in an educational context many times, their benefits have been tested and proved (Hamari et al., 2014).

Formally defining the designing of a "game" is a topic of contention in the literature. However, we consider Whitton's (2009) approach very useful. Instead of thinking in terms of "game/not game," it may be more useful to talk about game-like properties:

- *Competition*: There is a winner/loser scenario
- *Challenge*: The activity itself is not trivial
- *Fantasy*: There is and imaginary environment component
- *Exploration*: There is a simulated environment that can be investigated
- *Goals*: There is an objective that must be achieved
- *Interaction*: Actions change the state of play and have consequences
- *Outcomes*: Performance can be measured (scoring, success/defeat)
- *People*: Other individuals take part in the activity
- *Rules*: It is bounded by artificial constraints
- *Safety*: The activity has no consequences in the real world.

Therefore, serious games and gamification are design processes based on integrating properties from this list. The more properties that are taken into account, the more "game-like" an activity becomes. However, it is also very important to point out that, even though both gamification and serious games share their root in game design, they are conceptually different and have different goals.

On the one hand, serious games take the role of a discrete resource in a course, providing the students with understanding on a topic by directly transmitting knowledge or training skills, just like a book, a lecture or a hands-on lab. Therefore, serious games are learning resources that, by virtue of being consumed or interacted with, directly affect the student's learning. Briefly summarizing, one might say that, up to some point, serious games can be considered interactive simulations which include game-like properties. Their main advantage are their high levels of interactivity, capability of immediate feedback and sandbox nature. The student's decisions and actions play a fundamental role in how the resource is experienced, in such a manner that parts of the experience might even be different for each student, but outcomes within the game do not affect other aspects of the course. These characteristics benefit students by motivating experimentation and re-consumption of the game as a learning resource itself until the topic is mastered (ie, playing again and again).

On the other hand, gamification serves as a tool for course structure or content delivery, in such a manner that students become engaged with the topic by interacting with its new game-like properties. Between all these properties, probably the most important one in gamification would be "outcomes." Thus, the final product is not a discrete learning resource that is incorporated to the course, but a new version of the course itself. In this regard, game-like properties and mechanics are integrated into the kind of actions a student can usually perform during the course (eg, consume learning resources, deliver exercises, help fellow learners, etc.), or in the aesthetics or look and feel of course content, in such a manner that following the course becomes a motivating experience itself. A good gamification design will also slowly build a sense of accomplishment and progress for the student.

κPAX mixes both techniques to provide a set of learning resources (serious games) and then motivates students to consume them collaboratively, while also providing useful feedback on skill mastery (gamification). However, it must be noted that, as a platform, it is not directly concerned with the educational structure of such content beyond the purely technological aspects (ie, integration and deployment). Content itself is left up to the developers and educators. Therefore, the focus during the platform design process fell into the gamification aspects and how to integrate games as easily pluggable elements. How each serious game should be designed and created is out with of the scope of this project.

In order to better understand how gamification can be effectively used, we provide a brief background about how gamification works from a motivational standpoint and some relevant examples of existing platforms that rely on gamification to engage their user base in similar scenarios.

2.1 ON THE BASIS OF GAMIFICATION

Gamification pursues engagement. In order to achieve this goal, it relies on well-known game design techniques to shape user behavior through engagement and motivation. The psychology of motivation has raised several theories with different levels of complexity, even though, in the end, they all translate to the physiological level. When you are motivated, the brain emits the chemical signal dopamine, as anticipation to pleasure. Even though gamification feeds from different sources in the psychological and motivational literature, the self-determination theory (SDT) (Deci and Ryan, 1991) and Flow (Csikszentmihalyi and Csikzentmihaly, 1991) are its main foundations, and enough to understand its context. On one hand, in SDT, it is stated that there are three aspects which govern human motivation: autonomy, competency and relatedness. First of all, people are motivated when they are in control of their own choices and can create their own strategies. Second, the task at hand, even though challenging, should be perceived to be within reach of one's own skills. Finally, people are also motivated when their actions have an impact on their immediate environment, especially at the social level (eg, achieving recognition, helping other people, etc.). According to Flow, the optimal experience also requires that the difficulty of a task and the our skill mastery is perfectly balanced all the time. At the beginning, tasks should be very easy. As our skills grow, tasks should slowly become more challenging. An imbalance will lead to frustration (the task is too difficult) or boredom (too easy), and therefore, a lack of engagement.

In this regard, effective gamification requires that the different aspects of human motivation are taken into account when importing game properties and mechanics to a nongame context, such as in education. The mechanics need not be complex, but nevertheless need to account for motivational aspects. This is a delicate process that should never be performed in an *ad hoc* manner, but with the help of a gamification framework, understood as a formal process or guideline. Given the increasing interest in the topic, several such frameworks have been developed in recent years ranging from those fully defined to a loose list of pieces of advice with a few examples tacked on. For a more detailed comparison of them, a comprehensive literature review can be found in Mora et al. (2015). Even though the specifics may be a bit different between them, they usually agree on three basic steps: defining the expected behaviors, identifying player types and deploying the appropriate game mechanics, for the given player types.

By defining expected behaviors, the purpose of gamifying a process is set. Gamification operates at a motivational level, and, therefore, whenever it is applied, it is because the designer desires to shape the participant's behavior, so that they will perform some specific actions. For instance, in a learning environment, it could be attending class, delivering exercises or solving problems at class, among others. Usually, when deciding on the expected behaviors, the designer also decides a set of metrics that will help him or her to evaluate the degree of success.

The player type is the link that translates pure motivational theory into game design and gamification. At this stage, game players are classified into roles, each one considered to be motivated by different game mechanics. Again, different approaches exist, but the most well known, and the basis for *κPAX*, would be Bartle's (1996). The main roles are established by classifying players according to two axes: players vs world, and interacting vs acting. This results in four possible roles. The so-called "Killers" like competition against other players. "Socializers" enjoy having a sense of community with other players, as well as recognition. "Achievers" work towards a clear goal or obtaining some reward. And finally, "Explorers" like experimenting with the game rules and narrative. Usually, players do not belong to a single type, but have a "Bartle Quotient": they belong to all categories at once, each at a different degree. Given a player type classification, a gamified system may be designed so it will cater to a single type, by relying on a very focused set of mechanics, or to different types at once, by using a broader array.

The choice of game mechanics judges as suitable for the chosen types of player could be considered the core of the gamification process, as well as the most creative one. Again, just as with the term "game," there are different approaches to defining "game mechanic." We will go with Sicart's "methods invoked by agents, designed for interaction with the game state" (Sicart, 2008). There are many types of mechanics that can be extracted from the study of games. As far as the application of gamification in education is concerned, however, we can assess the most important ones from a representative survey in Nah et al. (2014). They are listed as:

- *Points*: A basic numeric performance measurement. It need not be tied to course evaluation (pass/fail)
- *Badges*: A token of recognition used a status symbols
- *Leaderboards*: Competitive ranking which displays each participant's position
- *Levels*: Indicator of overall skill mastery, attained by accumulating points
- *Storytelling*: The course encompasses a story or narrative
- *Customization*: Ability to customize some aspect, such as an avatar
- *Unlockables*: Content that is not accessible until the player has completed some task.

This is just a short summary extracted from some use cases in learning. There are many kinds of mechanics that could be extracted by studying games. The limit is in the designer's imagination, and, therefore, it is not possible to provide an exhaustive review in this section. Nevertheless, the most widely used mechanic by a very large margin, in all contexts, learning or not, is some variation of the Points, Badges and Leaderboards trio, often shortened as "PBL" (not to be confused with "Project Based Learning" in a learning context). Briefly summarized, whenever a participant acts according to the desired behavior, he or she is awarded either some points or a badge that can be used as a symbol of status (proving to others that he or she was able to do it). Points are used to rank all students in a public leaderboard, the one with the most points being at the top.

This simple combination is good enough to provide some satisfaction to each one of Bartle's player types: competition (Killer), social status (Socializer), concrete achievements and clear sense of progress (Achiever) and a sense of real system interaction with feedback (Explorer). More complex mechanics, such as the inclusion of storytelling or narrative also exist, like in Villagrasa and Duran (2013), but the effort required to properly design and deploy this kind of approach makes it less appealing. Nevertheless, other game mechanics can be found in the literature, but often included on top of a PBL system. For instance, using points as a form of virtual currency.

2.2 PLATFORMS BASED ON GAMIFICATION

Once the basis of gamification has been discussed, it is time to take a brief look at some relevant existing platforms that are conceptually similar to *κPAX*. Namely, those able to integrate a broad set of educational resources while at the same time relying on gamification techniques to encourage users to consume them. In addition, even though not directly related to education, we think it is worth mentioning some examples of platforms which use gamification to persuade users to consume video games. It is important to note that, in this context, we define "consume" not as a synonym of "buying," but as experiencing, experimenting, tinkering, etc., with the video game content. This is an aspect very relevant to educational resources such as serious games, and, therefore, up to a point, what *κPAX* is about.

Most mainstream platforms that act as a gamified learning management system (LMS) tend to be quite focused on corporate training. All of them incorporate the typical LMS resources: virtual classrooms with a calendar or scheduler and some method of web content delivery. However, resources tend to be simple ones, such as downloadable documents or HMTL, not complex ones such as virtual labs or simulations. They all also provide a highly polished graphic style. Given that, from an educative standpoint, they all share very similar basic functionalities; in this section we just focus on the gamification design aspects and chosen mechanics.

Between the most current platforms, Academy LMS (Growth Engineering, 2015) is a popular one. According to the creators, the system may also accommodate higher education, going beyond small-scale corporate training. Its application of gamification is based on three very basic game mechanics: points, badges and levels. As students interact with the system, they accumulate points, that allow them to gain levels, as a status symbol and feedback mechanism. In fact, the level number acts as a course progress indicator. Other very similar platforms can be found in Accord, ExpertusOne and Axonify (Accord, 2009, Expertus, 2013, Axonify Inc., 2014). In these cases, the points and badges dichotomy is expanded by incorporating leaderboards, thus fully embracing the model. Axonify also incorporates some other more social-network-oriented game techniques, such as "coaches," virtual avatars that will comment on the student's performance, and teambuilding.

With regard to platforms developed by the academia, and not the enterprise, the most relevant one would be Gradecraft (Holman et al., 2013). It is a project at a much lower scale, but directly designed with higher education in mind. Even though this platform was created for a specific subject, videogames and learning, it is implemented as a generic LMS, which could easily fit any other course

Table 13.1 Gamified Platforms

Name	Context	Game Mechanics
Academy LMS	Learning	Points, Badges, Levels
Accord	Learning	Points, Badges, Leaderboards
ExpertusOne	Learning	Points, Badges, Leaderboards
Axonify	Learning	Points, Badges, Leaderboards, Mentoring, Teambuilding
Gradecraft	Learning	Progress Bar, Badges
Steam	Gaming	Points, Badges, Leaderboards, Levels, Trading, Crafting
Gree	Gaming	Points, Badges, Leaderboards

subject. Gradecraft has a great focus on learning analytics, and, therefore, its main mechanics are related to providing video-game-like feedback: progress bars and badges based on assignment completion.

Moving to platforms that use gamification as a tool to encourage users to consume computer games, probably the most well know one is Steam (Valve Inc., 2003). Truth be told, this platform was the most influential during *κPAX*'s conceptual design process. Steam's gamification mechanics are based, again, on the PBL model. An API is provided for game developers so that they are able to update player high score leaderboards and award special badges when players complete a particular action in the game (for instance, successfully completing a stage). Developers also have the choice of letting their games drop trading cards during gameplay, which can be later exchanged with other players, or crafted into special badges. Finally, buying games and crafting badges award experience points to players, which allows them to increase their "player level." High level players have more options when customizing their user profiles, as well as a higher chance of getting some bonus trading cards each day.

A similar, but more simplistic approach, can be found in Gree (Tanaka, 2004), a Japanese social networking service focused on mobile games and applications. Gree offers an API too, which allows the development of over the platform providing leaderboards and badges to players. In contrast with Steam, this process is much easier and straightforward from a developer standpoint. Additionally, the platform allows the possibility of trading in-game items in order to customize the player experience. Some other minor game networking services exist, which are spread across various mobile developers, but they are conceptually very similar to Gree.

The main properties of the reviewed platforms are summarized in Table 13.1.

3 *κPAX* CONCEPTUAL DESIGN

Our main goal was to design and implement an online learning tool that exclusively relied on serious games as an educational resource, complemented with gamification as the engagement and feedback mechanism. The main idea would be to provide a set of core services that allow the community to publish the games.

The first step in the development was the conceptual design of the system; the description of how this process should work and meet its requirements. The basic capabilities were defined at an abstract level, before analyzing technological approaches. First, we started with the requirements definition and gamification mechanics design. From here, it was possible to create an initial mock up system that would provide the basis for the current one.

3.1 SYSTEM REQUIREMENTS

The approach for the system requirements definition process was twofold. In a first iteration, the minimum core expected functionalities were identified. We started with a minimalist approach, but having already in mind that *κPAX* would be modular and could be later easily extended by developers who wanted to improve it. In a second iteration, we then tried to collect the list of maximum possible future functionalities which would require the minimum number of changes in the core when incorporated. The result was a set of requirements which would allow a long-term incremental development, even when the system was already deployed, around a very stable core.

The list of *κPAX*'s core requirements follow:

1. The system must be open to and accessible from any operating system (Windows, Mac/iOS, Linux and variants such as Android, etc.) or device (mobile, tablets, videogame consoles, etc.).
2. There are three kinds of users: First of all, developers, who publish games as learning resources. It is possible that, in the real world, a developer may create a game at the behest of another entity (such as educators), but in the system we consider anyone who makes a game available within the platform and takes care of its management a "developer". Second, learners, who consume them. Finally, the administrators, who manage the system and approve game submissions, among others.
3. Developers can publish games, making them available to the platform's user base. This process should be as simple as possible.
4. Learners must be able to search for games related to a specific skill and freely play them.
5. Learners have a personal profile they can edit.
6. The system must provide feedback mechanisms on the evolution of learner skills, as they progresses through different games. As the system considers that a user becomes an expert in a subject/skill, labels on the levels of competence are incorporated in the personal profile.
7. *κPAX* should allow both on-line and off-line playing. It should be possible to download the game, play and reconnect to synchronize high scores, achievements, etc.

The role of *κPAX* admins is to guarantee that the games, as learning resources, will aim to promote learning (this includes theory, skills, etc.) and take advantage of the platform to facilitate distribution. Therefore, the administrators should comprise the educators at the institution deploying *κPAX* (in our case, lecturers at UOC). Developers request publication of their games via the platform interface, providing a description and some basic information about the game (topic, related competencies, genre, etc.). Each submission is then reviewed by the administrators, who then decide whether to approve it and incorporate the game into the platform, or not. Which characteristics must be taken into account in order to approve a submission are up to the admins. *κPAX* just provides the technological means to manage this process. Some of the technological requirements will be discussed in Section 4.3.2.

Games should offer incentives and awards due to partial successes, and some recognition can also exist for specific tasks, such as completing a game level, a set of games, achieving a set of skills, etc. Overcoming levels/stages should be directly related to the achievement of competencies. However, it is worth mentioning that these details of game design are beyond the scope of the system definition, and up to the developers. Again, *κPAX* just behaves as the distribution channel. Nevertheless, games must be able to relay on the system to track the learning progress and offer a recognition scheme (see core requirement 2).

From the basic requirements, we also defined a set of possible functionalities that, even though not considered necessary for the initial versions of the platform, would be taken into account when designing the system core as probable additions:

- The system should support the potential for a game to be paused or saved and recovered, enabling it to be played at any time. Furthermore, it should be possible to resume games from any platform, upon synchronization with the previous machine.
- The system may incorporate a "teaching" mode that provides the player with digital resources that could be learning support.
- The system should allow interaction between users and groups: challenges, partnerships, groupings, etc. There are games that can be played individually or in groups. Competitions or "leagues" can be promoted, or simply teams of like minded players. Thus, players can join teams, according to the offers/demands of participants, for a specific game/league. In addition, it is possible to make regular tournaments, weekend leagues, etc., proposed by administrators or by users themselves.
- Players should know the connection status of their friends, for example, to allow requests for help if the game allows it.
- It should be possible to compare/correlate goals or achievements among friends and/or the community. Users should also be able to publish this information on games, results, etc., in other social networks (eg, Twitter, Facebook, etc.).

3.2 GAMIFICATION DESIGN

The design of the gamification elements in *κPAX* needed to be included at the very initial stages of the project, since this would have big implications in the system development, acting as the structure that gives cohesion to the whole. Furthermore, offering a feedback mechanism is considered a very important part of the platform (see Section 3.1). Gamification was chosen as the means to fulfill this requirement. In order to create the gamified approach, we reviewed current gamification frameworks and followed the steps broadly described in Section 2.1: defining expected behaviors, identifying player types and, finally, choosing game mechanics that motivate the player types.

The expected behavior of users in *κPAX* is quite straightforward: that they play the different serious games that developers make available through the platform. Even though the use of gamification to inspire this behavior may seem redundant with the use of serious games as an educational resource, it must be taken into account that a serious game is not unquestionably engaging just by virtue of being a video game. The goal of serious games is, first and foremost, learning, not entertainment, even though being fun is a highly desirable property. Nevertheless, even when a game has been correctly designed to be enjoyable, different people may like different kinds of games. For instance, some people may like triva games, but dislike logic puzzles. A student will not be directly engaged in a game from a disliked genre. In addition, as a user becomes used to a game and the novelty wears off, interest wanes. Given that playing the game has an educational purpose and is not a choice completely left up to students, there are cases where gamification may help, motivating the users into playing a broad set of instructional games.

κPAX does not focus on a single player type, and its gamification design tries to cater to each of Bartle's roles. Therefore, we opted for the tried and tested PBL approach which many platforms already integrate (see Section 2.2), providing mechanics that can be adapted to all player types. Using this system, we also wanted to keep the gamification mechanics simple and straightforward, since the core focus of the platform is learning through serious gaming. Nevertheless, we have included some twists of our own. In this regard, Fig. 13.1 summarizes how the gamification mechanics work in *κPAX*.

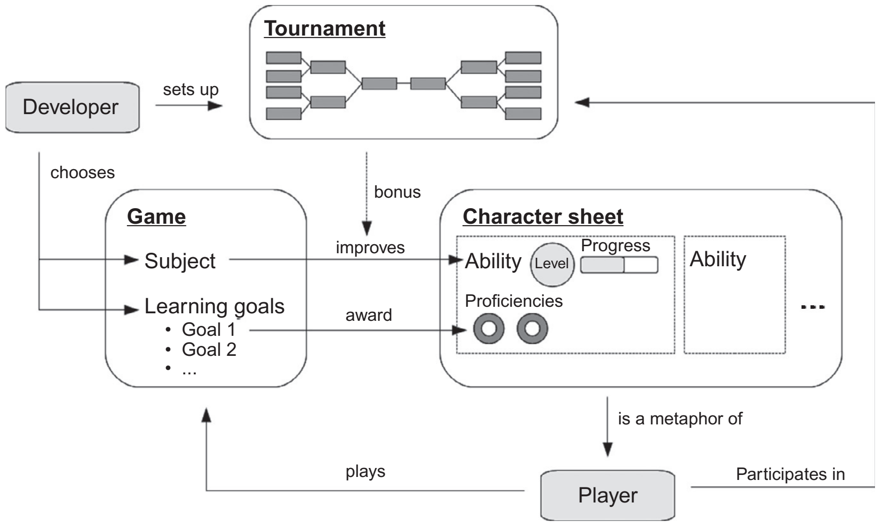


FIGURE 13.1

κPAX gamification mechanics summary.

First and foremost, gamification is used as a method of personal feedback, heavily inspired by tabletop role-playing games (TRPG), and especially Advanced Dungeons&Dragons (AD&D) [Cook (1989)], in format and nomenclature. In this regard, each user has a *character sheet*, or profile, that acts as a metaphor for the learning progress. The character sheet contains a set of *ability scores*, each one representing the student's degree of mastery on different general subjects (eg, math, language, history, etc.). Abilities slowly increase as users interact with games within that subject. However, the higher the current ability score, the longer it takes to increase it. In addition, each ability score has a linked set of *proficiencies*, discrete indicators of specific skills students have demonstrated (eg, solving a very difficult equation, knowing some amount of vocabulary, etc.). Each proficiency can be earned up to three times (bronze, silver, and gold levels).

When a developer creates a game that will be included in the platform, a subject of study and a small (2–3) set of goals, that players should accomplish, are chosen. Every time a user plays a game, the impact on the character sheet is twofold: on the one hand, progress towards increasing an ability score increases according to user performance in the game (for instance, in the final score). A user may play the game many times, with no limit, in order to successively increase the ability score on the subject. On the other hand, any time one of the game's learning goals is achieved, a proficiency is awarded. Therefore, a user's ability score and linked proficiency list may be influenced by performance in different games on the same subject.

In addition to all the inputs and outputs related to the character sheet, the system also incorporates the functionality to create tournaments in different formats (league, single-elimination, etc.). This goes one step ahead of current approaches, which exclusively rely on course-long leaderboards. *κPAX* is able to deploy much more flexible competition scenarios. Given these game mechanics, the system may engage "Killers" (tournaments), "Socializers" (status by level), "Achievers" (proficiencies) and "Explorers" (playing different games to increase abilities).

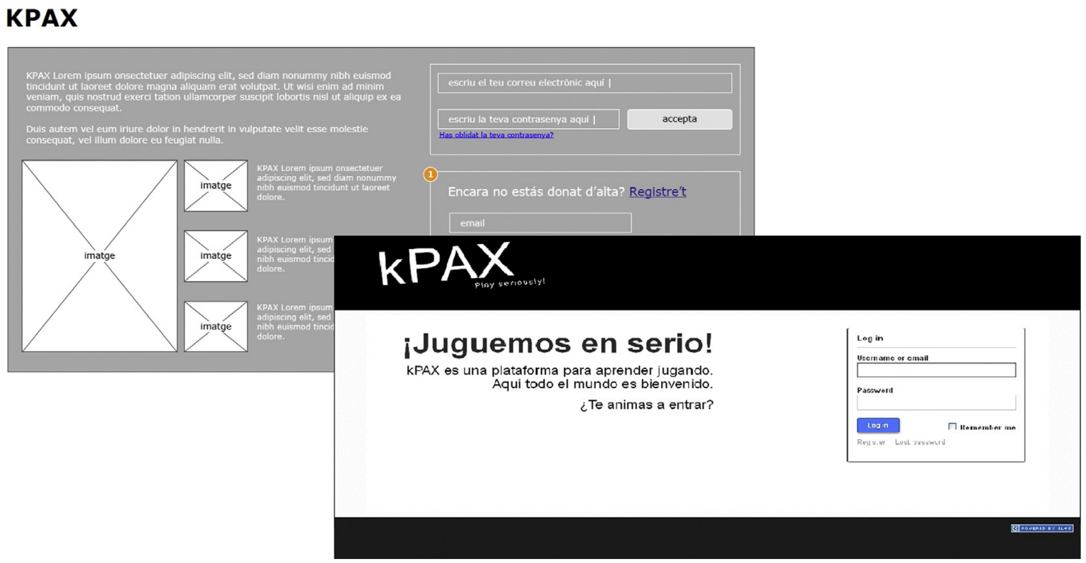
Finally, it is important to point out that we never explicitly present the gamified system as a game itself, but just as a form of performance feedback. In fact, the use of PBL seems almost natural, since students are used to being graded on their performance. Therefore, the system can be presented as an effective method for tracking their progress toward subject mastery. This approach transmits a sense of purpose and meaning to the system.

3.3 WEBSITE AND INTERFACE DESIGN

Once the basic system requirements and the gamification mechanics have been established, it is possible to proceed to the initial design of the system interface and web view. Initially, the design included the minimum parts of the system, the goal being a first release with basic features which would act as a wireframe prototype. From this version, the system could easily grow, step by step, by slowly adding new capabilities, or creating a more complex interface, that could be easily tested. This first release contained the following parts, using a very simple design:

- The presentation page (portal), which was very simple and provided validation of existing users and registration of new ones
- Log in capabilities, relying on a local user database
- The activity tab, visible once logged in, where internal communication between a users connected as friends is possible. Here it is possible to find the user's published results, achievements, messages, challenges, etc.
- The games tab, which allows a user to view and search through all available games in *κPAX*. Different searches can be performed, changing several criteria
- For each game in the platform, a page containing all its related information (ie, game properties tab). This page includes both static data and specific space for future dynamic data (eg, current best results, the competitions and challenges set on it, etc.)
- A network tab that allows users to search for friends and chat with them
- Finally, the user profile tab, which contains all the personal information and progress in the system (the "character sheet", see Section 3.2)

In successive iterations, the web pages have been evolving in complexity beyond the prototype stage. Additionally, as functionalities are added as plugins, new interfaces become necessary. Examples of such evolution are shown in Figs. 13.2 and 13.3.



KPAX

KPAX Lorem ipsum onsectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla.

KPAX Lorem ipsum onsectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore.

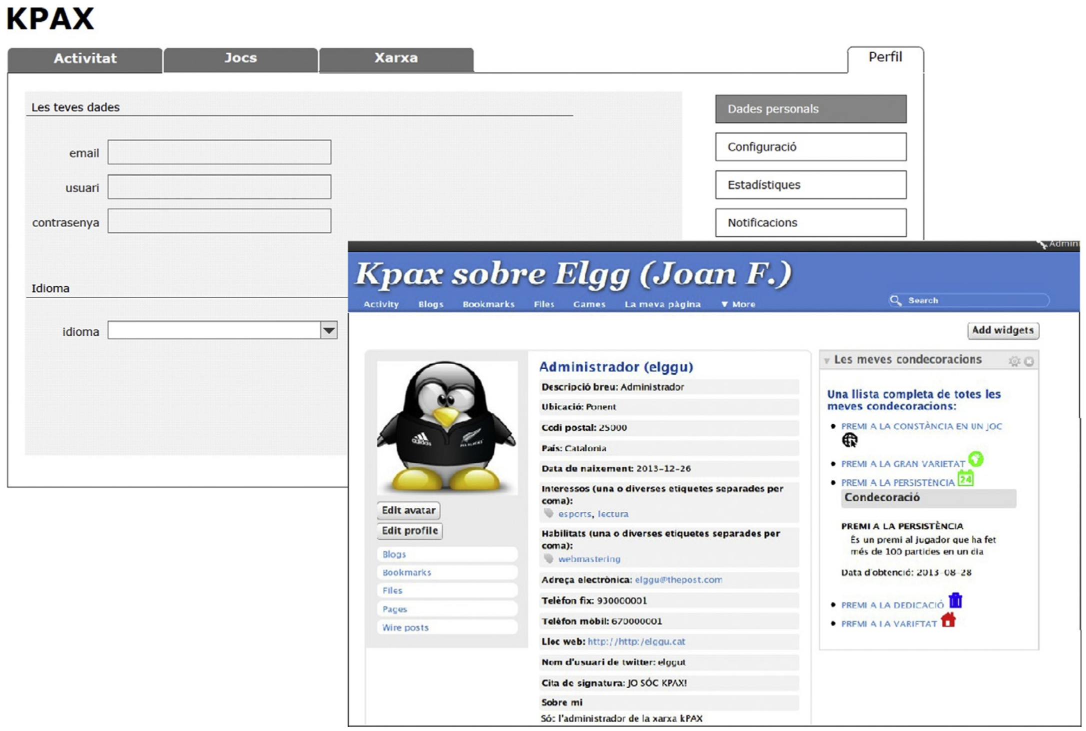
imatge
imatge
imatge
imatge
escriu el teu correu electrònic aquí |
escriu la teva contrasenya aquí |
accepta
Has oblidat la teva contrasenya?
Encara no estás donat d'alta? Registre't
email
kPAX
Play seriously!

¡Juguemos en serio!

kPAX es una plataforma para aprender jugando.
Aquí todo el mundo es bienvenido.
¿Te animas a entrar?

FIGURE 13.2

Evolution of interface: main page.



KPAX

Les teves dades

email

usuari

contrasenya

Idioma

idioma

Kpax sobre Elgg (Joan F.)

Administrador (elggu)

Descripció breu: Administrador

Ubicació: Ponent

Codi postal: 25000

País: Catalonia

Data de naixement: 2013-12-26

Interessos (una o diverses etiquetes separades per coma): esports, lectura

Habilitats (una o diverses etiquetes separades per coma): webmastering

Adreça electrònica: elggu@thepost.com

Telèfon fix: 930000001

Telèfon mòbil: 670000001

Lloc web: http://http:/elggu.cat

Nom d'usuari de twitter: elggut

Cita de signatura: JO SÓC KPAX!

Sobre mi

Só: l'administrador de la xarxa kPAX

Les meves condecoracions

Una llista completa de totes les meves condecoracions:

- PREMI A LA CONSTÀNCIA EN UN JOC
- PREMI A LA GRAN VARIETAT
- PREMI A LA PERSISTÈNCIA

Condecoració

PREMI A LA PERSISTÈNCIA

És un premi al jugador que ha fet més de 100 partides en un dia

Data d'obtenció: 2013-08-28

- PREMI A LA DEDICACIÓ
- PREMI A LA VARIETAT

FIGURE 13.3

Evolution of interface: game properties.

4 κ*PAX* ARCHITECTURAL DESIGN

From a technological standpoint, in order to simplify the system's architecture we decided to base it on W3C[4] standards, also based on CSS3 and Javascript, which should be exportable to different platforms. It should be accessible both from client applications and via web browsers. After a study into possible open social network platforms, we finally chose *elgg*[5] as a basis to implement κ*PAX*.

κ*PAX* has got a "core" architecture based on services, which in turn has a library that implements these services and allows the development of games and external applications that interact with the core. The services is available for use in any possible target platform, with an easily extensible data model to include future expansions. This keeps the inner workings of κ*PAX* completely transparent to developers. The platform includes the services, forming a set of independent logical modules for users and groups, security, registers, publication, network, games and resources management. Fig. 13.4 shows a diagram of the platform core, depicting how it works.

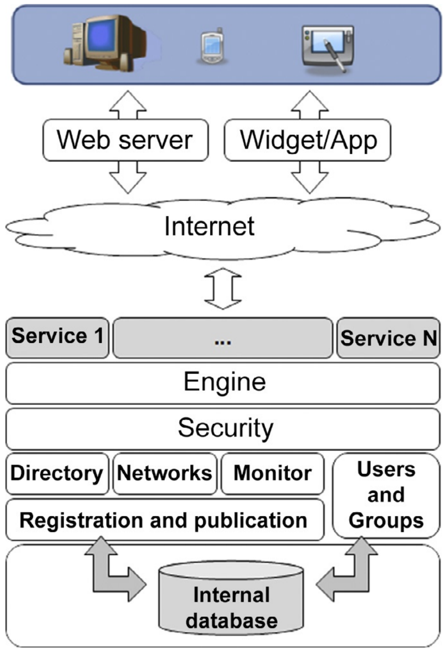


FIGURE 13.4

Platform schema.

The internal services engine is based on *Apache*/*Tomcat*, *Axis* or any other applications server which that allows communication mechanisms, such as sockets and/or XML and channels that allow secure encrypted transmissions (for everything concerning users, games and social networks).

[4]http://www.w3.org/.
[5]http://elgg.org/.

Developed games will be connected with the core by means of a set of calls to services provided by *κPAX*. These services and the corresponding documentation are available online for those developers wanting to add games or extend the platform.

4.1 SERVICES DEFINITION

Services make the internal *κPAX* code transparent to external developers. Thus, they only need to know which services to call and which parameters to use. Services allow mainly to validate a user or a game (validation is used to avoid fakes, inappropriate content or games that are not considered to be educative), to inform *κPAX* about a new game, a new result of a game, a new badge/competence level for a player, etc. The rest of the work is done by the platform itself. From a technical viewpoint, these are implemented in Java, use hibernate to access the *κPAX* (mySQL) database, and pass information using JSON. Finally, as a part of the platform architecture, all are served by a jBoss[6] server.

In general, these are the services the platform should implement (at the moment it does not support all of them):

- New users and games registration
- Access to dynamic information (eg, activities, users, social activity, etc.)
- Games interaction (services provided to the games information pass and retrieving)
- Social networks, bridge or publication in external networks
- Authorization (eg, of users to form groups, or participate in games, or join, etc.)
- Specific services for developers, including management and publication of games, monitoring their activity, etc.
- Services for statistics, platform global monitoring, etc.
- Resources (services to access and present internal and external resources available for the games)
- Visualization, information processing services for multiple platforms
- Security (for users, games, internal communications, etc.)
- Internal services for modules intercommunication
- Groups-users management (eg, user authentication, group membership, etc.).

In the platform's current state, a number of services from the previous list have been implemented, but many of them have not been integrated as yet. As an example, the main basic services for a typical scenario are introduced below. They are presented in a tentative chronological order to show how an external user and developer might use them.

A serious game has just been completed, so it is registered in *κPAX*. To publish it the developer fills in and submits all the required information. This implies that some services need to be called, but they are already in the *κPAX* core code and are kept invisible to the developer. Once the *κPAX* administrators validate and accept the game, players can see it in the games tab. As an accepted game, it is stored in the internal database. The service definition that allows adding a game is as follows:

```
+ addGame(idSession, name, gameInfo):idGame
```

On the other hand, players are also registered using some of the following services, depending on whether they are local *κPAX* users or use an external network (ie, Realm) password (eg, UOC, Facebook,

[6]http://www.jboss.org/.

twitter, etc.). Thus, when a new user registers, a service stores the information in the database. Notice that even when the player does not register as a *κPAX* user, an internal logical user is added to the database.

```
+ addUser(login, pwd, realm):idUser
```

From this moment, the player can log onto the platform (or external apps, webs, etc. connected to *κPAX*) triggering the corresponding services and receiving a session id. Then, the player can look for a specific game under a certain criteria, see all the information about a few games in order to choose one, etc.

```
+ login(login, pwd):idSession
+ listGamesByCategory(session, idCategory):listGames
+ listGames(session):listGames
+ getGame(session,gameId):gameInfo
```

After browsing and finding our developer's game, if the player is playing it (for instance, on a smartphone), this will send the results to the platform at the end of the game (or later in case of being offline).

```
+ syncStateGame(gameId, userId, state):Bool
```

This approach makes it really simple to keep all the information on the games, the players and the results of each player's games in the database. Furthermore, it only leaves to developers the easy task of adding a few calls to their code in order to keep the platform up to date with regard to a player's achievements.

4.2 DATABASE DEFINITION

Although elgg contains a database itself, we have decided to create a specific one for *κPAX*. We find this clearer both for us and for external developers, and it avoids complete system breakdowns.

The *κPAX* database stores all the necessary information for the proper functioning of the platform. A basic structure with the minimum information has been designed and implemented. The structure of this basic database can be seen in Fig. 13.5. Notice that the idea is to be as simple as possible in a way that it is easy to understand, so that it may be complemented and modified (if necessary) by new modules' developers. The database, together with the elgg part of the platform, is served by an XAMPP[7] server.

As said in the *κPAX* architecture section, *κPAX*'s database can only be accessed by the services provided, for both reading and writing information. Thus, the allowed operations are limited and new capabilities' developments may require the inclusion of modifications, or even new tables in the database, and the inclusion of new services.

The main tables appearing in the basic initial version of the *κPAX* database are:

Users. The system has a set of users, who can access the platform via their usernames and passwords, or through external validation by other social networks. In that case, the necessary information about the connection to the external service is stored in the *Realm* table (see below). In that case, as said before, it is not mandatory for a user to have an account in the *κPAX* system, therefore, the *login* and *password* fields of the *User* table are empty. Anyway, a player will always have a logical *κPAX* user, even when a local account does not exist. Authentication is very important in order to associate each call made to the system with a corresponding user (ie, assign *idUser* to calls made to services).

[7]https://www.apachefriends.org/index.html.

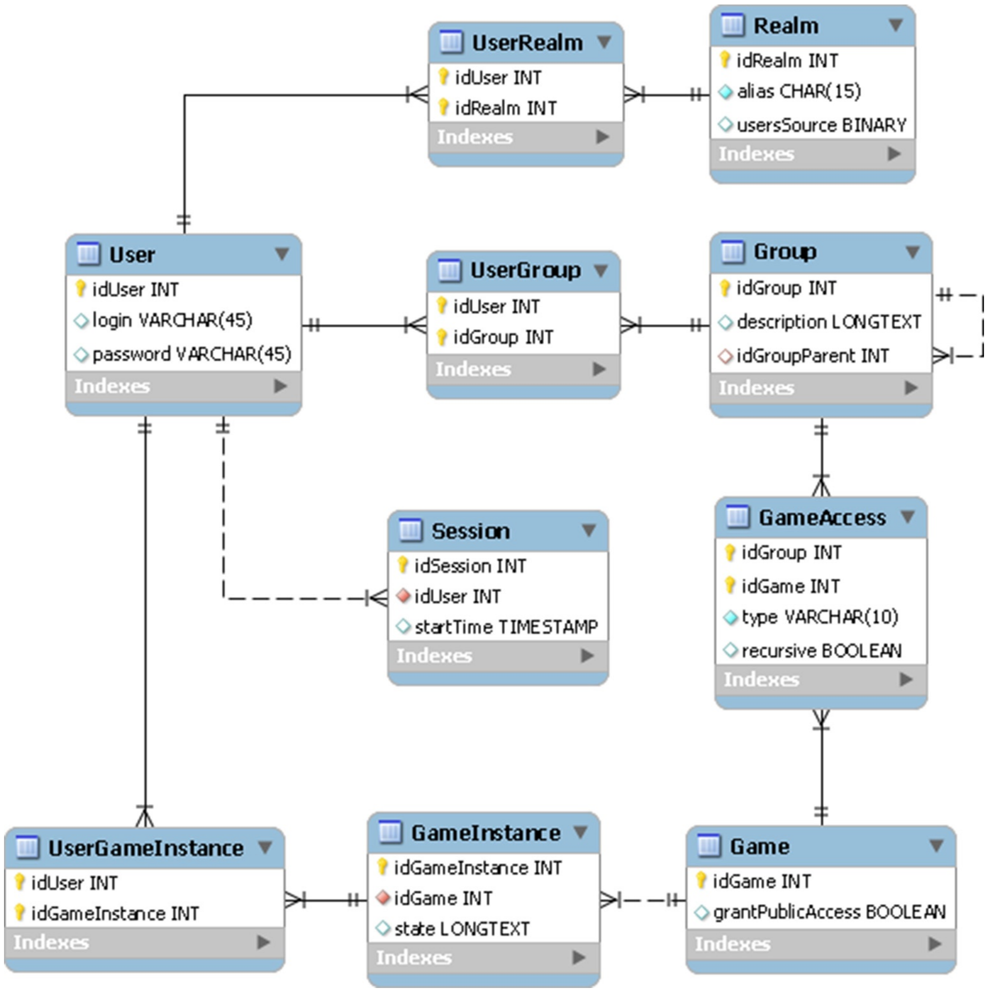


FIGURE 13.5

Platform database design.

Groups. To manage access to games, users can join groups. All groups must be registered in the *Group* table. Its structure is hierarchical and there are various root groups. Each group can only be part of another group (indicated in the attribute idGroupParent), although not necessarily be there. Root groups are limited to special system groups.

Users can belong to as many groups as necessary. Their affiliation to a group is in the *UserGroup* table. Attributes can be added to this table in order to specify types of membership, giving certain permissions within the group to some users (eg, manage users, create subgroups, etc.).

Games. Given a game, which is accessible through the *κPAX* platform it must be properly registered in the system. Only authorized games will appear in player searches. Thus, any developer can create an entry in the *Game* table, but it has to be accepted by the system administrators. Each game has a unique *idGame* within the system. In the *Game* table, all the necessary information to describe every game and even common properties can be added. For example, the field *grantPublicAccess* indicates that any user from *κPAX* can play this game (unless it is not specifically stated otherwise in a game later). Other properties might be, for instance: whether a game can be played by one or multiple players, whether it is collaborative or not, etc.

Games access. Since some games may be used for assessment purposes in academic environments, it is possible to restrict the access to certain users. For this reason, access rights can be assigned to the games for different groups. The table *GameAccess* contains information on access to games by different groups. To make the structure as flexible as possible, it contains not only the allowed accesses, but denied accesses to a game by certain groups (ie, blacklists). This information is encoded in the *type* attribute of the table *GameAccess* (values allow or deny).

Any order of refusal should outweigh more than the access ones to prevent someone who has been explicitly denied access to play the game. To know whether a user can play a game, we should firstly verify that no group for this user has got the access barred. In such a case, access will not be allowed. Since the structure of groups is hierarchical, a parameter *recursive* is defined in the *GameAccess* table. It indicates that the authorization must be made within the group tree. That means that if a group at a higher level of the hierarchy in the user's groups has access to some information, that will also apply.

Session. When a user logs in, a session associated with the connection is created. These sessions are recorded in the *Session* table, which contains the *idUser*, and are limited to a period of time (the *startTime* attribute has been defined with the creation date and time).

Game Instances. In order to play, users need to join an instance of a particular game, which should contain all the state information of that game. This information is stored in table *GameInstance*. Every instance has an unique identifier *idGameInstance* within the system. Since each game will have its proper information, a field has been created where the state of the game instance is stored using XML syntax, with the internal format and attributes decided by the game developers. In games allowing several users or collaborative gaming, this XML keeps the information needed to manage such a feature. Players who are playing a game are recorded in the table *UserGameInstance*.

Realm. As commented above, players in *κPAX* do not need to register on the platform and can be validated by third party networks. The *Realm* table contains a list of these networks validating those *κPAX*'s users.

4.3 SECURITY

Security is a key step in the architectural design process of any system, and should no longer be considered an add-on at a later stage of development. Even at the cost of some overhead, a minimum security baseline must be kept in order to ensure some degree of protection against attackers who want to intrude or disrupt the system. The security design model of *κPAX*'s architecture focuses on guaranteeing two basic capabilities.

- *User authentication*: To ensure that only authorized users can log into the system and they can only access and modify their own information.
- *Data integrity and trustworthiness*: To ensure that any information exchange between the platform and applications is correct and the result of an actual outcome from user actions.

In this subsection, we briefly explain how they are achieved.

4.3.1 User authentication

Being a multi-user system, ensuring that only legitimate users are able to log in, and constraining them to their account space, is a very basic functionality. As far as we are aware, every similar platform has some degree of user authentication. However, in addition to the ability to authenticate users with local accounts, *κPAX* is able to authenticate them through trusted third parties, such as UOCs virtual campus, or some social networks. Therefore, any user who already has an account with any of the trusted third parties need not have an additional local account, easing account management from the user's standpoint. In fact, the use of trusted third parties is the default behavior, since it is expected that every *κPAX* user has, at least, a UOC virtual Campus account. Local accounts should the used only as a last resort, for very special cases.

This part of the security model relies on a plugin that we implemented, based on OAuth 2.0 protocol (Hammer-Lahav and Hardt, 2011) integration for trusted third party authentication. Initially, only integration with UOC's OAuth 2.0 authorization server is supported, through simple Representational state transfer (REST) calls, based on the HTTP protocol. In order to ensure that only authorized applications may invoke the service, incoming REST calls require a client ID and secret API key provided by UOC beforehand. Otherwise, incoming REST calls are rejected. Therefore, *κPAX* has been assigned a client ID and secret key. Using this protocol, it is also possible to rely on UOC to obtain some additional information about the user, such as his or her username, e-mail address or role (student, instructor, etc.), even though the main goal is just user authentication.

The authentication process is summarized in Fig. 13.6. The process is initiated when a user connects via the browser to the login page, selecting authentication via his UOC account (a). Then, a message exchange begins between *κPAX* and the UOC OAuth 2.0 authorization server, using the secret API key to secure all messages. First, *κPAX* uses the client ID to identify itself and requests that UOC authenticates the user (b). Then, the user's browser is redirected to a UOC login page, where he or she can enter his or her username and password to authenticate him- or her self (c). It must be noted that this

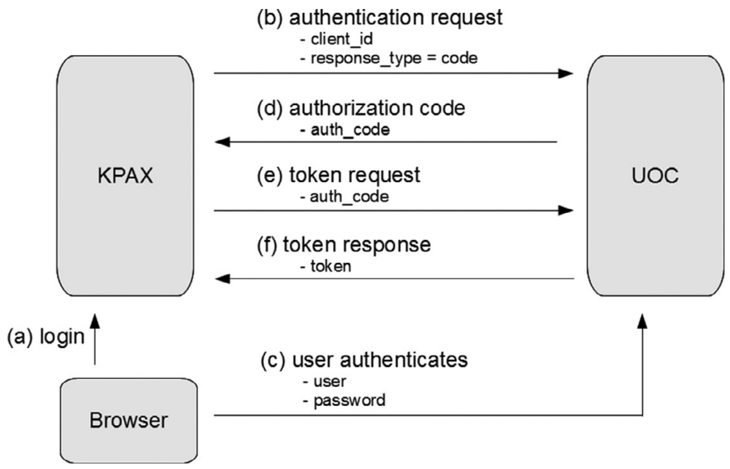


FIGURE 13.6

Authentication using OAuth 2.0.

communication is direct between the user and UOC, and, therefore, *κPAX* never has access to this sensitive information. After the user has been satisfactorily identified, a brief OAuth 2.0 message exchange follows (d) and (e). At the end, *κPAX* obtains a token that acts as proof of correct user authentication, and can also be used to access some of the user's data stored at UOC.

4.3.2 Integrity and trustworthiness

The integrity and trustworthiness security model of *κPAX* must take into account two kinds of adversaries. First of all, from an integrity standpoint, it must ensure that any information sent by a user cannot be modified in transit by an attacker. Second, from a trustworthiness standpoint, it must be ensured that a *legitimate user* is unable to send false information to the system in order to gain some advantage (ie, cheating). For instance, trying to force an update with an invalid score, higher than the one actually obtained by playing a game normally. This is a sometimes overlooked (or even outright ignored) problem. For instance, the highly popular game distribution platform Steam (see Section 2.2) takes into account the former, but does away with any kind of control for the latter. In contrast, Gree does take it into account. We consider that this security mechanism is very important in order to ensure the platform's reputation.

This part of the security model relies on the digital signature of data by each application, prior to transmitting. The fact that applications must have signed data requires some management on the part of the developers during the game registration process. But from the user's standpoint, it is completely invisible. The process description follows:

1. As part of the game registration process, it is required that the developer creates an RSA key pair: SK_{app} (the secret key) and PK_{app} (the public key). Such a key pair will be specific to that game and the only one, and it can be easily generated using open tools such as OpenSSL The OpenSSL Project, 1999). A step-by-step tutorial is provided to the developers.
2. The developer uploads PK_{app} to the system, which is stored in the database associated to that particular game. This upload process requires no extra security and can be done in plain text.
3. The developer integrates SK_{app} into the application's code. SK_{app} will be used to sign any service request to the platform. A timestamp is also added to any request prior to signature, countering replay attacks.
4. The platform validates any data received by applications using the associated PK_{app}, which is locally stored in the database and easily accessible.
5. At any time, developers may update SK_{app} and PK_{app} to new values, from their developer accounts. It must be noted, that when the update is finished the current instances of the application will stop working, since they contain the former private key in their code. Therefore, this method should be used, for instance, only when upgrading the application to a new version. This will also force users to upgrade.

The advantages of using a digital signature algorithm, instead of other simpler algorithms as an alternative, such as HMAC (Krawczyk et al., 1997), are twofold: first of all, no system manager can cheat, since SK_{app} is always in the developer's hand; and second, intruding the developer's *κPAX*'s account will also attain nothing, since the only retrievable information would be PK_{app}, which is not enough to execute the signature algorithm. The only point of failure is the developer, the only entity storing SK_{app}. Attackers could force an update with a new SK_{app} and PK_{app} of their choice. However, such an update would be easily detected, since all legitimate game instances would stop working after the change.

The main vulnerability of this approach is a reverse engineering code. SK_{app} should be somehow obfuscated to avoid this. Therefore, the method will avoid casual attackers, but not necessarily dedicated ones. Nevertheless, the technological level required for this kind of attack is not trivial, and the developer can always update the keys and the application. Highly sophisticated methods that could make the task much more difficult to attackers (but, unfortunately, which are never 100% effective) do exist (Subramanya and Yi, 2006), but we consider that, for this kind of platform, they are not really worth the costs (deployment complexity, monetary).

5 CONCLUSIONS

This paper presents our experiences in the design and development of a gamified platform to support learning through serious games. The project, born in the UOC (a fully on-line university), tries to take advantage of serious videogames as a resource to train a broad set of proficiencies. The learning process is framed inside a gamified social network in order to increase the engagement of players. Although the product has not been tested in a real environment, a first release is already completed and ready to be used. In the next semesters, it will be used to assess its efficiency within the UOC campus in a set of chosen subjects.

5.1 TECHNICAL DEVELOPMENT

In order to have a platform from an initial stage, we have made a global design and implemented only the core of it, ready for evolution. Thus, we have presented in this paper the main ideas and experiences of the complete design process, and the technology behind all the work that has been finished (and included in the release). Notice that *κPAX* is continuously being extended by final project students and external developers. The project has been developed under an open access license, making it accessible to any developer interested in participating in the project.

The design of *κPAX* takes into account three main aspects: a set of system requirements depicting our idea of the platform, the use of gamification and the interface design. These conceptual decisions imply taking into account four architectural focuses: the definition of the database, the services, the security, and the integrity and trustworthiness.

From a practical view, the core of the platform has been developed as an extension to elgg, an open source social networking engine, and mySQL as the database technology. These run on a XAMPP server. Furthermore, a set of services which mainly access the database are programmed in Java, and run in a jBoss server. Other technologies used in the project are JSON, hibernate, PHP, OAuth 2.0, and maven. An advantage of using elgg is that components can be easily added by developing new plugins. Thus, we have created the core development of the platform with a limited amount of capabilities and generated a first release. From this point, a number of new plugins have been developed and we are currently working on the integration, which is no easy task, since developers apply different changes to the database and create services.

The current situation is: after 4 years we have a first release ready to be tested. This is the result of a two months work on the part of a professional developer and around 15 students' final degree projects (250 h each). From now on, the results of every assessment will give us hints as to how to continue improving it. In order to do that we need specific games for the chosen skills. At the moment we have a trivia-like game, which will be used to prepare some of the lessons of chosen subjects. Furthermore, we will be working in two or more games to work with other learning aims. Altogether, *κPAX* should become a valuable new resource that will make students' learning a bit more engaging.

5.2 ACADEMIC IMPACT

From an academic point of view *κPAX* is mainly a tool for teachers who detect "gamers" among their students. It allows the inclusion of games and gamification in any education level subjects to promote work on competences and skills. The aim of gamification, in this case, is to engage students' and maintain motivation for learning. Educators using *κPAX* for their lectures need to choose the appropriate game (or games) for the competences/skills to work and set up the subject, including *κPAX* resources (eg, by generating a league on a game "X" and designing how the outcome can be included in the subject's global results). Feedback can be given to students both in the platform (where players can see their profile improving with every play) and in the classroom, depending on how the teacher has designed the whole activity. Thus, for the former, the feedback is automatic and for the later, it depends on the subject design. The teacher, as a creator of the specific competition regarding a game, can access some statistics and history to see who is more/less active, when and how much students are playing, what there achievements are at any given time, etc. This information can help students in the development of the learning process.

κPAX offers a set of services to help developers to include their serious games into the platform. The included games an be produced by anybody with the ability to develop (including from software companies to single developers). The use of services makes the integration of new games very easy, and involves only adding a few calls to *κPAX* in the game's source code. The game's quality cannot be ensured by construction, but the teachers and players will play and grade them. However, administrators perform a first filter by only accepting those games which are considered educative, and accomplish certain ethic terms. should an inappropriate game pass this first control, users can always denounce the contents and administrators will remove it from the available games in the platform.

5.3 FUTURE WORK

As of the current version, all the data collection is entirely geared towards the gamification engine, in order to generate the skill progression feedback via the character sheet. However, such a large amount of information could also be used from a learning analytics standpoint, in order to gauge other important information, such as which topics raise the most interest, which ones could be considered more difficult (learners consistently get poor scores), which topics are under- or over-represented, etc. But this would be a direction for a future version of the platform, or an auxiliary subsystem, under the umbrella of a spin-off project.

Further work includes analyzing the current feedback from the different tests in order to start refining the platform as well as including new plugins, so it can be upgraded from its current first release to a new version.

Finally, right now, developers and administrators act in a completely independent manner during the submission process. This is the nature of *κPAX* as a solely technological platform. However, we could consider going beyond the pure technological aspect and move towards a service-based approach in order to better coordinate the efforts of developers and administrators, or even other educators who just use published games, but do not directly participate in the platform. For instance, this could be carried out by creating special training sessions for developers as to how to create serious games, or by providing incentives for games in some of the under-represented skills/topics.

ACKNOWLEDGMENTS

This work was partly funded by the Spanish Government through the project TIN2013-45303-P "ICT-FLAG" (Enhancing ICT education through Formative assessment, Learning Analytics and Gamification). Moreover, we would also like to thank the following collaborators to the *κPAX* project, even though they are not paper authors: Àgata Lapedriza, Xavier Bar, César Crcoles, Josep Jorba, David Masip, Juan Francisco Sánchez, Francesc Santanach and Alicia Valls.

Last but not least, we thank a number of students who have been developing new functionalities as a part of their degree final projects. Part of their work (eg, Fernández, 2013; Farrerons i Herrero, 2014) has already been integrated into the platform and the rest will be in the near future.

REFERENCES

Abt, C., 1970. Serious games. Viking press, New York.
Accord, 2009. Accord LMS. http://www.accordlms.com.
Axonify Inc., 2014. Axonify. http://www.axonify.com.
Bartle, R., 1996. Hearts, clubs, diamonds, spades: Players who suit MUDs. J. MUD Res. 1 (1), 19.
Brumels, K., Blasius, T., 2008. Comparison of efficacy between traditional and video game based balance programs. Clin. Kines. J. Am. Kinesiother. Assoc. 62 (4), 26–31.
Cook, D., 1989. Advanced Dungeons & Dragons, second ed. TSR Inc., Lake Geneva, WI.
Csikszentmihalyi, M., Csikzentmihaly, M., 1991. Flow: The Psychology of Optimal Experience, vol. 41 Harper Perennial, New York.
Deterding, S., Khaled, R., Nacke, L., Dixon, D., 2011. Gamification: toward a definition. In: Chi 2011, pp. 12–15.
Deci, E.L., Ryan, R.M., 1991. A motivational approach to self: integration in personality. Perspect. Motiv. 38, 237.
Expertus, 2013. Expertus one. http://www.expertus.com.
Farrerons i Herrero, J., 2014. Perfil complet d'usuari per a la xarxa kpax.
Fernández, S.M., 2013. Tema per elgg. Master thesis, Universitat Oberta de Catalunya.
Growth Engineering, 2015. Academy LMS. http://www.growthengineering.co.uk/academy-lms.
Hamari, J., Koivisto, J., Sarsa, H., 2014. Does gamification work? — a literature review of empirical studies on gamification. In: 47th Hawaii International Conference on System Sciences (HICSS), pp. 3025–3034.
Hammer-Lahav, D.E., Hardt, D., 2011. The OAuth2.0 Authorization Protocol.
Holman, C., Aguilar, S., Fishman, B., 2013. Gradecraft: what can we learn from a game-inspired learning management system? In: Proceedings of the Third International Conference on Learning Analytics and Knowledge. ACM, pp. 260–264.
Huizinga, J., Hull, R., 1949. Homo ludens. A study of the play-element in culture [trans. by rfc hull.].
Krawczyk, H., Bellare, M., Canetti, R., 1997. HMAC: Keyed-Hashing for Message Authentication.
Mora, A., Riera, D., Gonzlez, C., Arnedo-Moreno, J., 2015. A literature review of gamification design frameworks. In: Seventh International Conference on Virtual Worlds and Games for Serious Applications: VS-Games, pp. 100–107.
Nah, F.F.-H., Zeng, Q., Telaprolu, V.R., Ayyappa, A.P., Eschenbrenner, B., 2014. Gamification of education: a review of literature. Yamamoto, S. (Ed.), HCI in Business, Springer, Heidelberg, pp. 401–409.

Pelling, N., 2011. The (short) prehistory of gamification. Funding Startups (& Other Impossibilities), Haettu. https://nanodome.wordpress.com/2011/08/09/the-short-prehistory-of-gamification/.

Prensky, M., 2001. Digital natives, digital immigrants part 1. On the Horizon 9 (5), 1–6.

Sicart, M., 2008. Defining game mechanics. Game Stud. 8 (2), 1–14.

Subramanya, S., Yi, B., 2006. Digital rights management. IEEE Potentials 25 (2), 31–34.

Tanaka, Y., 2004. Gree. http://product.gree.net.

The OpenSSL Project, 1999. OpenSSL: The Open Source toolkit for SSL/TLS.

Valve Inc, 2003. Steam. http://store.steampowered.com.

Villagrasa, S., Duran, J., 2013. Gamification for learning 3d computer graphics arts. In: Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality. ACM, pp. 429–433.

Whitton, N., 2009. Learning with digital games: A practical guide to engaging students in higher education. Routledge, New York.