CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Facultad de Informática

TRABAJO FIN DE GRADO

Educational Activities for a Gamified System

Autor: Guiomar Valderrama de la Torre

Director: José Luís Fuertes

MADRID, ENERO DE 2015

Index

List of Figures

List of Tables

Abstract

The objective of this dissertation is to analyze, design, and implement an activity module for a larger educational platform with the use of gamification techniques with the purpose to improve learning, pass rates, and feedback. The project investigates how to better incentivize student learning.

A software requirement specification was delineated to establish the system guidelines and

behavior. Following, a definition of the activities in the module was created. This definition encompassed a detailed description of each activity, together with elements that compose it, available customizations and the involved formulas. The activity high-level design process includes the design of the defined activities by use of the software methodology UWE (UML-based Web Engineering) for their future implementation, modeling requirements, content, navigation and presentation. The low-level design is composed of the database schema and types and the relating EER (Enhanced Entity-Relationship) diagram. After this, the implementation of the designed module began, together with testing in the later stages.

We expect that by using the implemented activity module, students will become more interested in learning, as well as more engaged in the process, resulting in a continuous progress during the course.

Resumen

El objetivo de este trabajo es analizar, diseñar e implementar un módulo de actividades didácticas que formará parte de una plataforma educativa, haciendo uso de técnicas de gamificación con la finalidad de mejorar el aprendizaje, ratio de aprobados y retroalimentación para los alumnos. El proyecto investiga como incentivar mejor el aprendizaje estudiantil.

Se trazó una especificación de requisitos de software para establecer las pautas del sistema y su comportamiento. A continuación, se definieron las actividades del módulo. Esta definición abarca una descripción detallada de cada actividad, junto a los elementos que la componen, las configuraciones disponibles y las formulas involucradas. El proceso de diseño de alto nivel incluye el diseño de las actividades definidas usando la metodología de software UWE (*UML-based Web Engineering*) para su futura implementación, requisitos de modelaje, contenido, navegación y presentación. El diseño de bajo nivel está compuesto por el esquema y tipos de la base de datos y el diagrama de entidad-relación correspondiente. Tras esto se realizó la implementación y pruebas de parte del sistema.

Se espera que usando el módulo de actividades implementado, los estudiantes muestren un mayor interés por aprender, así como estar más involucrados en el proceso, resultando en un progreso más continuo durante el curso.

1. Introduction and Objectives

This dissertation involves the gamified activities that will form part of a wider educational platform for the subject "Procesadores de Lenguajes" imparted in the Universidad Politécnica de Madrid (UPM). The dissertation will seek to analyze and carry out the design and implementation of various activities, bearing the studied advantages of a gamified education. All activities will have an educational purpose, but they will be presented as entertaining to students by offering a system that will keep them incentivized.

To achieve this, the technique of gamification will be used. Gamification can best be defined as the use of game elements and game design techniques in non-game contexts [1]. Some such design includes game elements like points, badges, and leaderboards. In other words, it attempts to make complex and dreary tasks invigorating and entertaining for the users.

This technique has been used in many other projects successfully in areas such as marketing and business. This has made more and more companies and institutes to recently become interested in the field of Gamification. Renowned companies to have favorably applied Gamification in their business include Nike, Foursquare, IBM, and Samsung among others. One of these cases, Samsung Nation [2], included many benefits for users getting involved with the community and in such a way created a strong social platform.

However, we must bear in mind that not all companies that apply Gamification do so successfully. The simple use of points, badges, leaderboards, and other such elements are not enough on their own, and are not always what the system needs. Companies to have failed in the use of Gamification are Zappos [3], which Gamified its platform with no objectives or purpose that is, rewards are given without cause, and Google News [4], which devalued badges in such a way that very simple and base tasks resulted in rewards, thus belittling the worth of any obtained badge.

To properly apply Gamification, it is important to know ones own business. This encompasses grasping the behavior of any target groups we expect will use our system, and using game elements in such a way that they are met with approval from the users. The system should strive to make users feel engaged when they are in and outside the system, such that they do not tire of it while using it and that they feel the need to return to it.

Furthermore, to be sure Gamification is being used correctly, guidelines and frameworks are available which, if followed, should provide sufficient advice. Additionally, analyzing current gamified systems can be useful to determine what works and what does not, keeping in mind that what applies to that specific system may not apply to another.

Lastly, it should be kept in mind that this technique usually requires various iterations and that the design cannot be expected to be impeccable from the start. Regardless, the goal should be to design a complete and functional system by collecting data after each iteration and using it to modify and improve components.

In the case relating to this dissertation, that is, gamification in education, it has been proven to be an effective way to engage students, ease their learning, and improve their performance. Examples that support this can be seen in section 2.1.2. Heeding these aspects, we can adapt them to the needs of this project to build a module for the educational platform.

1.1 Background and context

As mentioned before, this dissertation deals with a section of an ongoing project. To understand our objective, the origin of the larger project should be exposed.

The initial reason to begin the project was actually due to the practical part of the subject. To begin with the project, background data was gathered on the concept of building the educational platform. The performance of the students of the course was analyzed by looking into statistical data collected during three years. This data includes class attendance rates, pass and fail rates, and exercise submission rates for students. Relevant professors were consulted and asked about existing issues within the handling of the subject, such as the complexity of the course, the evaluation methods used, and what was expected from a student who successfully passes the course.

The “Compiladores” subject was a course with a one-year duration, which included the development of a full compiler as a practical exercise. In the mid-90s out of approximately 400 enrolled students only 5% gave in the practical part on June due to the complexity of the exercise involved. This figure was maintained until the 1999/2000 course in which, because of the exhibited difficulty, there was a decision to split the practical assignment into three parts. On the first years in which this measure was taken into effect (from 1999/2000 to 2002/2003) the initial 5% of hand-ins rose up to 37% and on later years it even rose higher than 45%.

An issue that was created by imposing three hand-ins in “Compiladores” is the deliverance of feedback, due to the testing taking a long time. This resulted in students having almost completed the next part before receiving the corrections on the previous one. Given that the result of one of the hand-ins directly affects the next, this caused the students to dedicate additional effort in refining their work. This in turn demotivates them and makes learning harder.

As of now and since the 2010/2011 course, the previous subject has become two different subjects by the name of “Procesadores de Lenguajes”, which deals with the compiler front-

end, and “Traductores de Lenguajes”, which deals with the compiler back-end. After the subject was split, “Procesadores de Lenguajes” presented the same manner of operation as “Compiladores” first exhibited, that is, there was again just one final hand-in for the practical part at the end of the course.

The “Procesadores de Lenguajes” subject is coursed in the third year of the bachelor degree in informatics engineering and the bachelor degree in mathematics and informatics, and is available in both the first and second course semesters. The subject is divided into a theoretical and a practical part, where each must be successfully completed to pass the subject.

Up to the first semester of the 2013/2014 course a total of 463 students have taken the subject. The academic results obtained by these students are not as positive as would be desired. The theoretical part is overcome by 59.8%; the practical by 45.6%; and the whole subject by 43.8%. By further analyzing this data it can be observed that a total of 47.9% of the students do not submit the practical part and thus cannot pass the subject. Excluding the practical part, the pass rate is very high, thus, it is vital to encourage students to do and submit the practical part.

Additionally, the subject is mandatory which means that the student must attain a level of competence in this area that will serve him/her throughout their professional life.

When these problems were presented, the project was born under the thought that the dynamization of the practices would allow for a more agile response and that learning this in an enthusiastic and continuous manner will make later work easier. As the project began to take place, the idea to expand the system and include a larger portion of the course material occurred. This involved the gamification of the platform and several additional modules.

1.2 Scope and Objectives

This dissertation focuses on the design and implementation of several activities for the gamified educational platform. As the activities themselves will also apply Gamification, they will be entertaining to the end user, assuring their engagement through a reward system. Some such rewards are points and badges, which are given once an activity has been successfully completed.

To achieve this, the storage of data resulting from activity fulfillment is a necessity, as is also the design and implementation of an administration module for the management of activities and users. The user interface will be simple initially, not forming part of the core purpose, but can be further detailed in future work. Another goal is to build a highly accessible system so that it is widely available. Web accessibility is based on making information equally available to everyone, regardless of any physical or developmental

abilities or impairments. Accessibility is a necessary part of the system, as certain laws must be followed by public companies, these specific laws are:

- The LSSICE law [5] (ley de los Servicios de la Sociedad de la Información y de Comercio Electrónico) and
- The Ley General de derechos de las personas con discapacidad y de su inclusión social [6]

The main objectives to be held in mind are to:

- Reduce student failure (Objective 1)
- Make learning and applying knowledge more appealing to students (Objective 2)
- Automatize feedback (Objective 3)

A reason for each of these objectives is presented in Table 1.

Goal	**Reason**
Objective 1	A larger percentage than is expected is failing the course. With the proper incentive, the expected pass rate should be reached.
Objective 2	It is important that students feel attracted to the learning and to practice it in a continuous manner so that they may internalize it.
Objective 3	Due to the large number of students enrolled in the subject (over 240 in first semester of 2014/15 course), delivering feedback is a time consuming process. If this is automatized, a better learning experience will be achieved.

Table 1 - Goals and the reasoning behind them

1.3 Achievements

The work carried out has consisted of various tasks. To begin with, some familiarization took place, both with the project and the past work as with the knowledge necessary to further advance it. Succeeding this, the design of activities for the gamified system for educative purposes, as well as the proper modification to existing definitions took place. This design encompassed a detailed description of each activity, together with elements that compose it, available customizations and the involved formulas. Once the activities that are to take a part of the system were well defined, the design process begun. This design includes the design of the defined activities by use of the software methodology UWE [7] (UML-based Web Engineering) for their future implementation, modeling requirements, content, navigation and presentation. In addition to this, the software requirements specification for the project module was done. Continuing the design, the administration module for the control and maintenance of the system and the database, with all adequate data, were designed. Parallel to the design, starting before it was completely done, the task of implementation begun.

1.4 Overview of Dissertation

The contents of what follow in the dissertation will cover the state of the art, development, conclusion and future work, references and appendixes. The state of the art chapter summarizes the current knowledge and other works related to the topic of Gamification and the various technologies that have been used. The development chapter will be the main body of the dissertation, including various subsections dealing with the different tasks performed. The conclusion and future work will deal with what has been achieved, together with an evaluation and a section detailing the limitations in the results and how the work might be further developed. Finally the references will provide the necessary sources mentioned or cited in the dissertation. Additionally, one appendix is added to relay information that is relevant to the rest of the work, though in a supplementary manner.

2. State of the Art

2.1 Gamification

As we have stated before, Gamification is the use of game elements and game design techniques in non-game contexts [1]. This allows for serious tasks to become enjoyable and entertaining.

Gamification should not be confused with other similar types of games that may share its purpose or design. Gamification finds itself related to serious games, games, and gameful design. Figure 1 shows the position of gamification with regards to these other types [8].

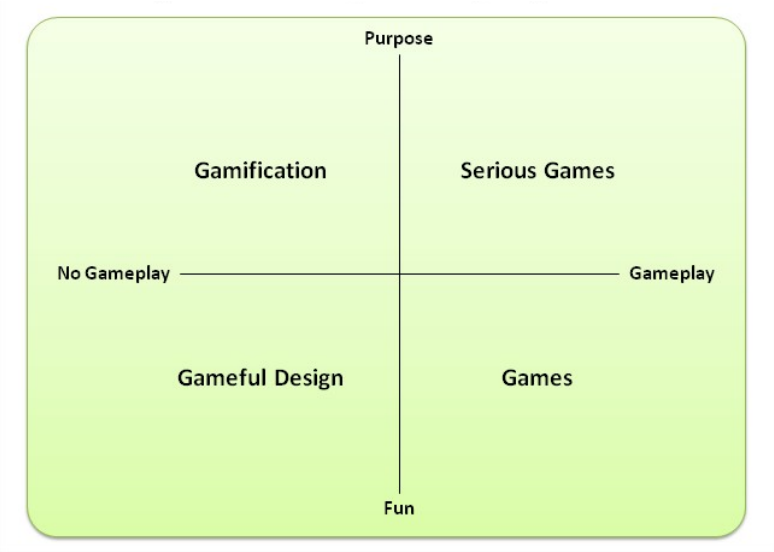

Figure 1 – Types of game thinking and Primary Design Goal

Games are meant explicitly for entertainment and fun. Any activity that is executed only for pleasure and has no other primary purpose is a game.

Gameful Design represents fun but without gameplay. This means it is a game like approach to usability and aesthetics without any addition of game elements. Game design provides a more enjoyable experience regardless of not being essential to the design.

Serious Games are games with an intended purpose different from exclusive entertainment. Although serious games are meant to have a serious purpose, it does not mean that they are not games and that they cannot be fun. Educational games can also be

included in this category as, although they are games, they have a strongly defined educational purpose.

Now that what gamification is and where it stands with regards to similar concepts is clearer, a further and more expansive definition of gamification can take place. Following, categorizations and fundamentals of gamification will be exposed and explained.

Gamification can be sorted into three categories: external, internal, and behavior change [1]. External Gamification aims to make clients engage more with the intended product or business. Internal Gamification focuses on applying these techniques inside the business itself, making employers engage to their responsibilities in an entertaining manner. The purpose of behavioral change, as implied by its name, is to motivate people to change their behavior weather this be regarding their health, manners or other similar aspects.

In Figure 2 we can see how these categories are structured in relation to who the end user is and where the benefit goes [1].

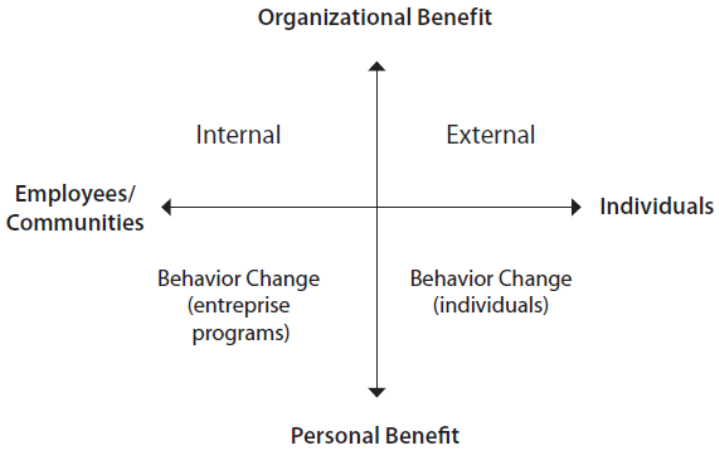

Figure 2 – Categories of Gamification

2.1.1 Gamification Fundamentals

When applying Gamification, certain fundamentals have to be considered for the correct use of this technique. We will analyze in the next sections a classification of the game elements involved and a Gamification design framework with steps to follow.

2.1.1.1 Game Elements

Game elements can be divided in three parts [1]: Dynamics, Mechanics, and Components. These three elements follow a pyramidal structure, in which the higher levels present a greater abstraction. Furthermore, the lower levels are inclusive of the higher ones, that is, they represent them in different terms. Figure 3 displays the three elements in a pyramid.

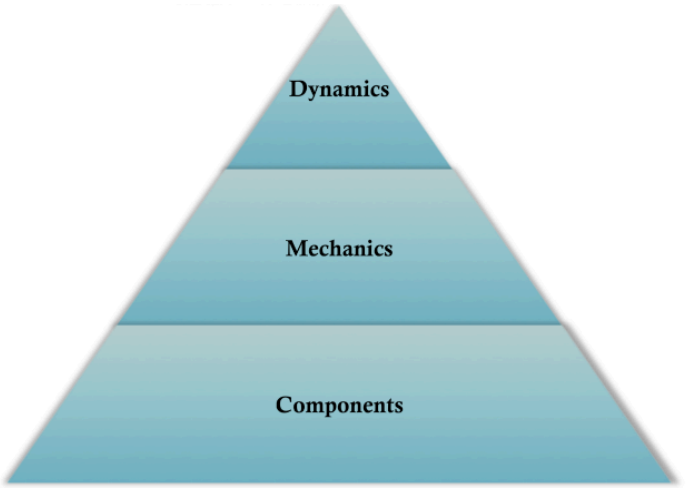

Figure 3 – Pyramid of Elements

2.1.1.1.1 Dynamics

The highest element in the pyramid, dynamics, also shows the greatest level of abstraction. Dynamics are the big picture aspects of the gamified system that have to be considered and managed but that cannot enter directly into the game. An example of this would be relationships.

2.1.1.1.2 Mechanics

Mechanics are the basic processes that drive the action forwards and generate player engagement. Each mechanic represents a way of achieving at least one of the dynamics described in the previous level and thus mechanics are greater in number. A mechanic related to the above example of relationships would be cooperation, as well as competition.

2.1.1.1.3 Components

The lowest level, components, represents specific instances of dynamics and mechanics. Just as each mechanic ties to one or more dynamics, each component ties to one or more higher-level elements. Components that are associated with cooperation or the competition mechanics would be teams or challenges respectively.

2.1.1.2 Design Framework

The Gamification Design Framework [1] defines six steps to be followed in order to make sure that all important aspects of Gamification are covered to a high percentage. The next subsections will explain these steps in detail.

2.1.1.2.1 Define Business Objectives

It is essential to know the purpose of the system and the goals it aims to achieve, the steps shown in Figure 4 state the four parts of the process to define business objectives.

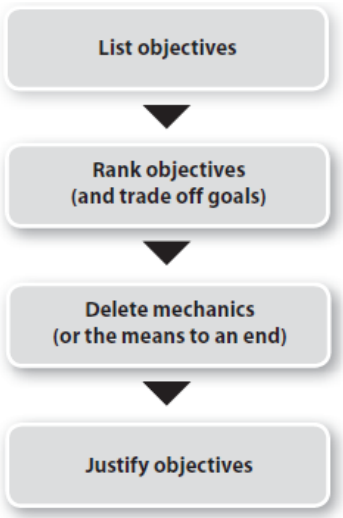

Figure 4 – The Objective Definition Process

In the first step, list objectives, a concrete list of objectives for the system should be made. These objectives should be very specific and precise, avoiding any abstract resolutions. The ranking of objectives takes the previous list and orders the objectives according to their importance in the system. After the ranking, some objectives might have been reconsidered and the decision to delete them must be made. This is specially so if the objective is only a means to another and not an end in itself. Once the final objectives are reached and structured, they must be justified, providing a reason for each of them.

2.1.1.2.2 Delineate Target Behaviors

Once the objective is clear, what is expected of the users (players) and how to measure it should be thought. The anticipated behavior and the metrics to measure it are best considered together. The target behaviors are to be concrete and specific, and work towards the previously define objectives, whether directly or indirectly. When the behaviors are set, the success metrics are developed. These metrics are a translation of the behaviors into measureable results.

2.1.1.2.3 Describe Your Players

The end users are one of the most important aspects to be considered, as the system must cater their needs and wants. To successfully do this, the target perception should be widened, making the system appealing to as many player types as possible. Two basic aspects on which to define players are demographics and psychographics. The

demographics focus on strictly objective information such as age, gender or location, while the psychographics analyses what the player expects to gain from using the system. A common player type model, which can be used to outline the psychographics, defined by Richard Bartle [9], can be seen in Figure 5.

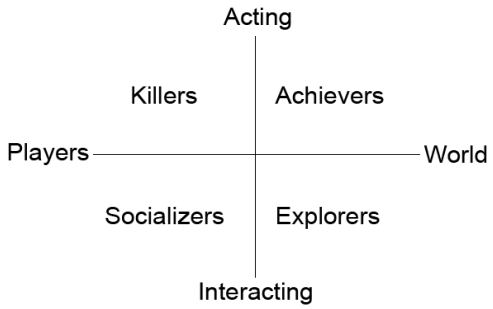

Figure 5 – Bartle's player type model

The player types displayed show how they behave by acting or interacting in relation with other players and the world. **Achievers** find themselves acting with the world, they prefer to progress in the system, gaining rewards that signal success, whether these actually present gameplay benefits or not. Achievers enjoy the prestige they feel they have achieved. **Explorers**, on the other hand, interact with the word, discovering all aspects of the system and taking advantage of them. They can often feel restricted if the system pushes them to act in a determined manner or imposes any restrictions, much preferring a certain level of freedom. **Socializers** favor interacting with other players, focusing on the social characteristics that might be available. To them, the system is just a medium for meeting other people. Finally, **killers** act with other players, relishing competition even if this does not actually bring them any rewards. Not only do they wish to thrive over others, they want to have others fail.

2.1.1.2.4 Devise Activity Cycles

Activity cycles are used to model the action that takes place within the gamified system. Two kinds of cycles take place: engagement loops and progression stairs. The first deal with detail what and why the players do and the system response, and the latter offer a greater perspective on how the player advances in the system.

Engagement loops

Motivation drives players to action, and this action in turn generates feedback as responses from the system. The produced feedback further motivates the player to continue acting with the system. Thus, the engagement loop works as a self-supporting cycle. This relationship can be seen in Figure 6. Feedback is the essential part of this cycle as they produce the main motivation by giving immediate visible results to actions. Most game components can be interpreted as forms of feedback.

Figure 6 – The engagement loop

Progression stairs

While the engagement loop of the gamified system shows the basic process of it, it does not explain how players advance, this is left to the progression stairs. Progression stairs show how the game experience and interaction with the system changes as players move through it. This change can usually be expressed as a journey composed of smaller assignments and long-term goals.

While the difficulty encountered by the player should increase through his progression, a strict linearity should be avoided. Instead, progression stairs should be used. The first stair, onboarding, should be simple and guide players, drawing them into the game. The next stairs should present a higher difficulty rate, each with a steady increase and then a period of relative ease. This rest period allows the players to become used to the new difficulty setting and to experience a satisfying sensation of mastery. These types of stairs often continue in small cycles, having "boss fights" as a final challenge of the level. Boss fights are especially hard challenges that test players thoroughly, making them feel victorious if they do them successfully. All this process is illustrated in Figure 7.

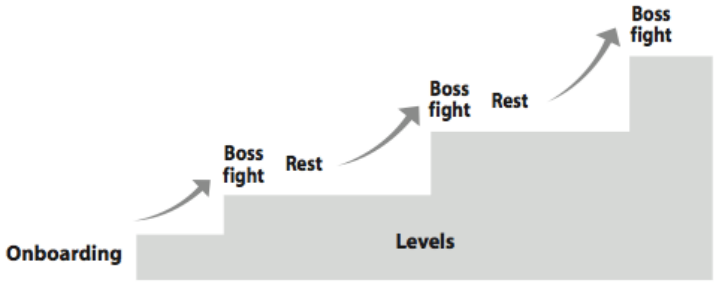

Figure 7 – The progression stairs

2.1.1.2.5 Don't Forget the Fun

Before implementing the gamified system, the design must be revised, especially to make sure that there is fun present in it, as it is a most important feature in gamification. As all

other aspects and their complexities are being considered, it is sometimes easy to overlook the fun. It is advisable to regularly check whether the design looks fun from the outside; else all the effort put in it is lost.

There are many ways to define fun. A popular one is the one held by Nicole Lazzaro, who classifies it in four distinct kinds [10]. **Hard fun** comes from puzzles or challenges that are fun precisely because of the challenge they present, as there is pleasure in overcoming them. **Easy fun** is a kind of casual enjoyment that emphasizes curiosity and does not require excessive effort to obtain. **Serious fun** comes from the excitement of experimenting with different situations. **People fun** is basically social fun, depending on the interaction with others, including competitive interactions.

2.1.1.2.6 Deploy the Appropriate Tools

This is the last step, the implementation stage. Having gone through the previous design steps, the implementation will follow a guideline, knowing the system objective and the intended players. The engagement loops provide a frame of the system. The overall experience of the players will be put together in the development stage. In each stage what to keep and what to exclude will have to be decided, but this is actually a positive thing as it makes the project more feasible. It should also be noted that the system will have to be continuously tested and refined to make sure that it does not deviate from its path.

To apply gamification adequately, a team with a variety of skills will be needed. However, this does not mean that a single person cannot do it; only that expertise in several areas is required. The expertise needed includes: People who understand the business goals of the project; An understanding of the target group of players and the basics of psychology, specially in regard of motivation; Game designers, or people who can do their functions; Analytic experts to make sense of the data generated by the system; Technologists that are able to implement our vision.

Using Gamification does not need any particular technology, but on the other hand, it works perfectly with online systems. Broadly analyzing the details of the system, it is indispensable to have a way to track interactions with the game elements and later integrate the results with the existing system. Two basic options are available for the technical implementation of the gamified system: To build a custom implementation or to use one of the software-as-a-service offerings. The decision of choosing one or the other depends on the business goals of the system and how flexible and customizable we need the system to be.

2.1.2 Gamified Education

The goal of Gamified education is to make the learning process more enjoyable and thus motivate students to engage in activities. To do this, it can combine Gamification techniques with e-learning. E-learning in turn, is the use of computers and IT technologies applied to educational tasks. There are many people who prefer electronic learning to

physical participation in lectures or classrooms. This is because it usually saves time and money and is more readily available. Despite these advantages, e-learning is still learning, and that goes with the associated tiredness or even boredom that comes with it. In fact, a downside of e-learning is that it is usually not as interactive as its physical counterpart, delaying feedback to any question that might occur. The result has been the recent use of Gamification to this e-learning technique, to make it more interesting and engaging. Following we will analyze some cases of Gamification used for education.

2.1.2.1 Class Dojo

ClassDojo [11] is a free classroom tool that helps teachers to improve the behavior of their students in the classrooms quickly and easily. The platform is supported via web and IOS devices (iPhone and iPad). It offers a way for parents, teachers and students to collaborate and manage the classroom behavior.

ClassDojo tracks students' behavior through technology, engaging them in their learning environment. Each student is assigned an avatar and given rewards or consequences from the platform. The avatars are in the shape of cute monsters, some of which can be seen in Figure 8. Furthermore, ClassDojo allows the selection of single or multiple students, classified in groups, to then operate with. This tool includes many game mechanics, such as levels, badges, achievements, avatars and leaderboards.

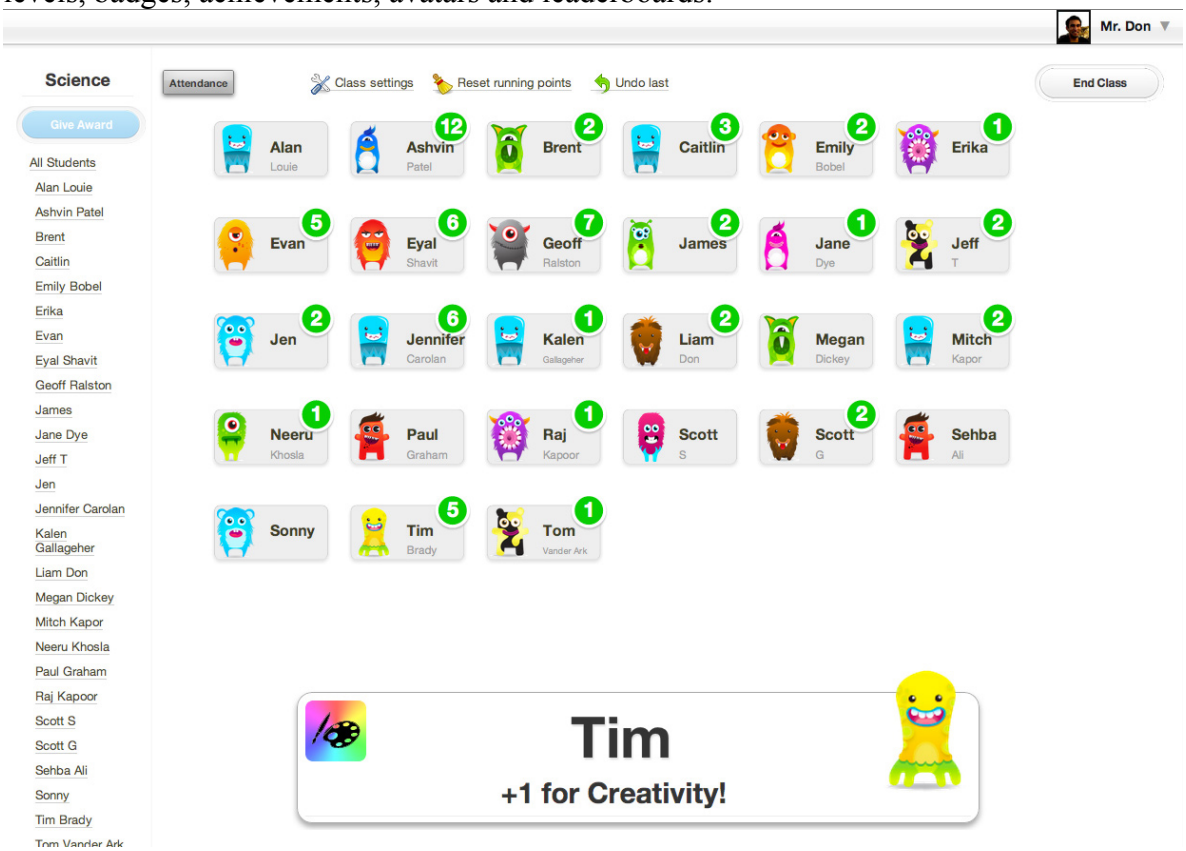

Figure 8 – User avatars in ClassDojo

The system is customizable; so that different teachers may define specific positive and negative behavior lists to track. This permits for a wide and flexible use after some proper planning. Additionally, data on the behaviors recorded can be shared with parents and administrators, making improvement more graphic.

2.1.2.2 Know Re

KnowRe [12] is a cloud-based adaptive secondary math program that provides students with a personalized curriculum to achieve their full potential. The platform aims to help students succeed in math by assessing, personalizing and engaging students with game-like features, attractive graphics and social learning.

This math tool is for use by both teachers and students who want to learn in an easy and quick way. Students can select chapters from a giant map or a list presented as a column to the left of the map, as is seen in Figure 9. Each chapter allows them to practice or challenge their skills.

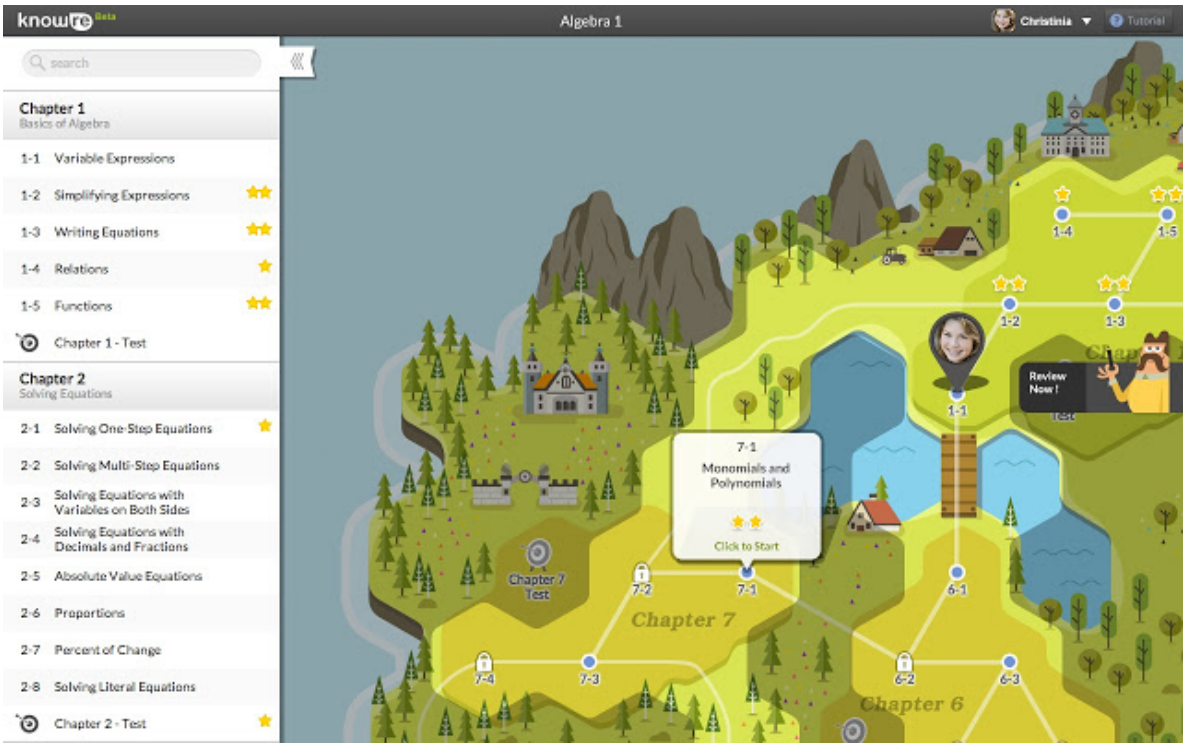

Figure 9 – The chapter map in KnowRe

On the other hand, teachers can start any course by clicking on the map location or the start button. Each chapter records a completion rate, readiness, and learning growth in the form

of stars. When a chapter is completed, teachers can administer tests to check the students' knowledge.

KnowRe attains certain objectives in the platform:

- Maximizing math achievement through adaptive curricula
- Taking personalization to the next level
- Making learning a breeze
- Building skills that lead to long-term success

Activities in KnowRe (as the one in Figure 10) grant coins; the amount of coins rewarded depends on the number of tries the student has to make to answer correctly. For solving the activity in the first try, three coins are awarded; two coins on the second try; one coin if answered correctly in more than two tries; and no coins if it remains unsolved.

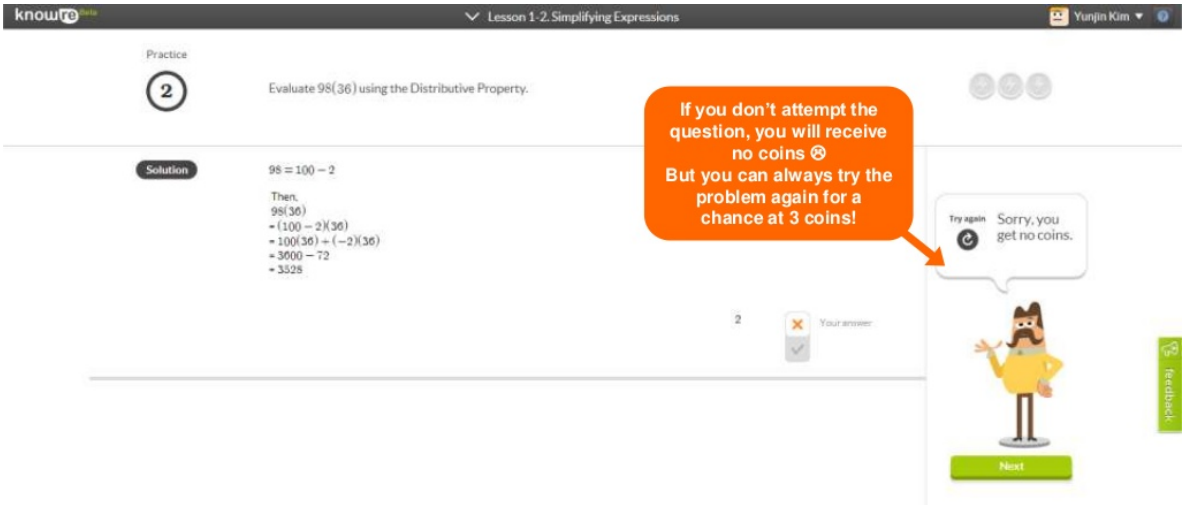

Figure 10 – A question in KnowRe

To show mastery over a chapter, stars are granted. Up to three stars can be obtained when all video lessons in a chapter have been watched, and all questions have been answered correctly on the first try.

2.1.2.3 Zondle

Zondle [13] is a Game Based Learning platform that focuses on delivering game based learning environments for young children, and helps teachers to track student progress and help create personalized learning environments. This allows teachers to focus on how to interact with students and strengthen their weakest areas. The content is delivered in the form of questions and quizzes, and ranks students on their performance later showing this in leaderboards.

More than anything, Zondle focuses on allowing students to practice, review, revise and memorize. As a student enters Zondle, there are three basic steps to take: choose a topic, choose a game, then play and learn from it. Zondle offers a great variety of games, some more interactive than others. Among these games we can find "Chick flick", where the student must fire a chick from a catapult, "Zoo lander", where how an animal lands is controlled by arrow keys, and "Crystal canon", where a canon must be shot taking into account angle and power (Figure 11). Although there are many different games, their essence is basically the same: to answer a question correctly. The standard question types available are: multiple choice, question and answer(s), and true or false.

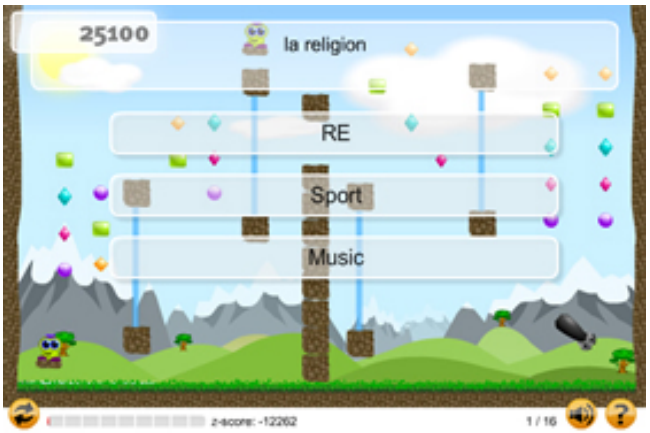

Figure 11 – Zondle crystal canon

Additionally, Zondle provides progress tracking for students to see their own development, and for teachers to track their students. This feature shows the last games played, together with the obtained score and a game play progress graph when viewing one's own progress. For a teacher, this section displays the games set for class, and within each one, the students that have played it, together with their scores and graphs.

Zondle contains many types of activities, however some are clearly games. This situates Zondle in a fine line between being gamified and being a serious game. The example in Figure 11 shows to have gameplay, as is thus not strictly gamification.

2.1.2.4 Socrative

Socrative [14] lets teachers engage and assess their students with educational activities on laptops, tablets (Android and IOS), and smartphones (Android and IOS). Through the use of real time questioning, instant result aggregation and visualization, teachers can gauge the whole class' current level of understanding. Quick questions can be either multiple choice (Figure 12), true or false, or short answer. By checking the answer to these questions, the teacher can easily decide how to continue the class. Socrative saves teachers time so the class can further collaborate, discuss, extend and grow as a community of learners.

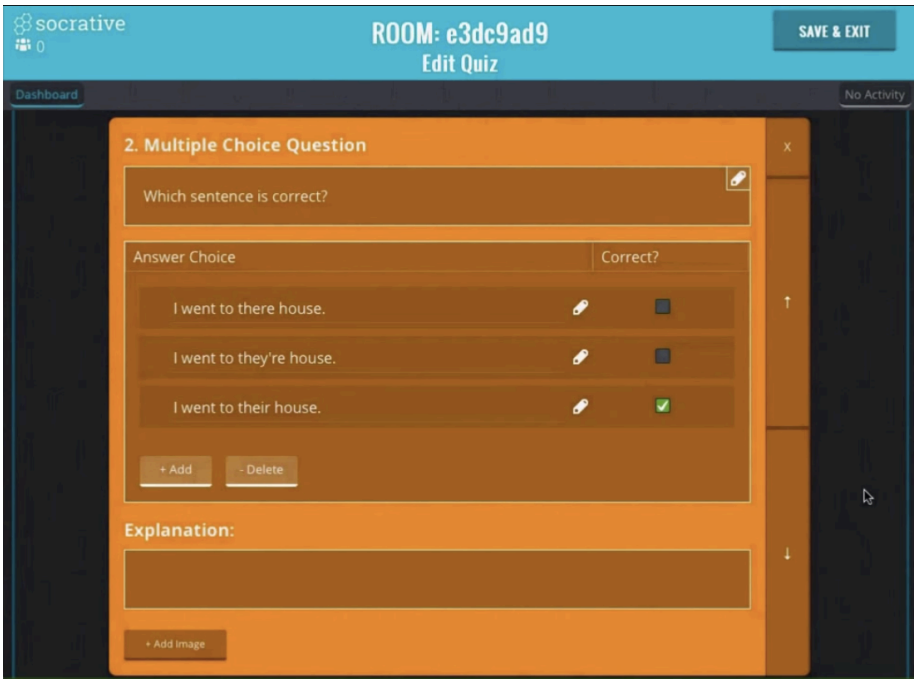

Figure 12 – Socrative multiple-choice question

In a nutshell, Socrative offers:

- Instant feedback
- Personalized content
- Reports
- Compatibility

Figure 13 shows on the left side how a teacher views the answers to a question live, with correct answers in a green background and wrong answers in a red one; on the right side we can see the screen a student sees when answering a quiz, his or her selected answer shown in green.

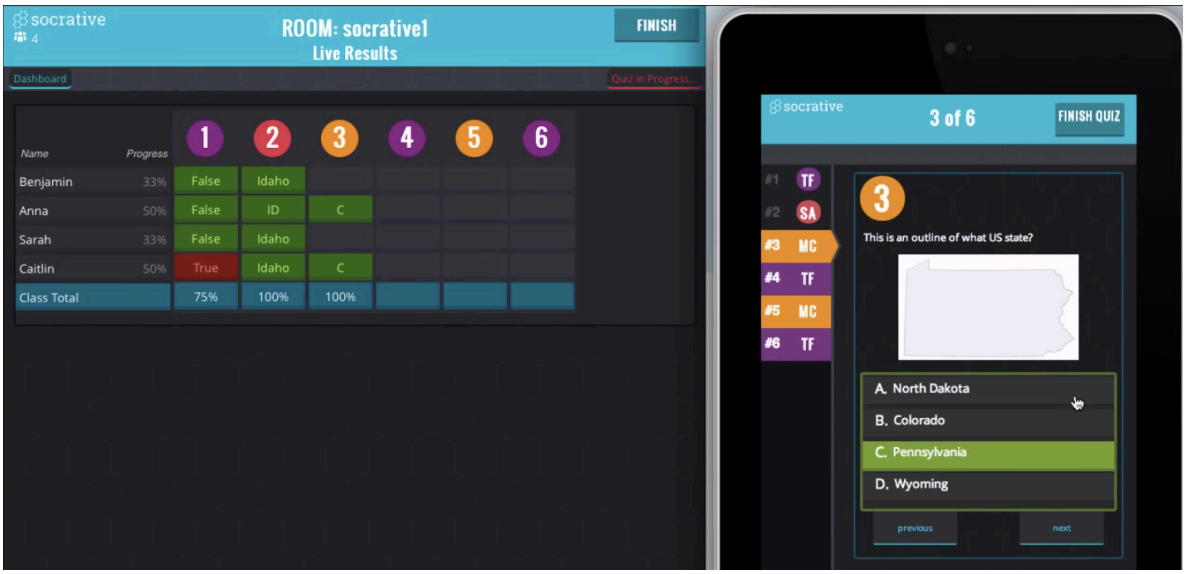

Figure 13 – Socrative live results

2.1.2.5 Khan Academy

Khan Academy is a non-profit online educational website [15] created in 2006 by educator Salman Khan to provide an education that is free and for everyone.

The website features thousands of educational resources, including a personalized learning dashboard, over 100,000 exercise problems, and over 5,000 micro lectures via video tutorials stored on YouTube teaching mathematics, history, healthcare, medicine, finance, physics, general chemistry, biology, astronomy, economics, cosmology, organic chemistry, American civics, art history, macroeconomics, microeconomics, and computer science.

When entering the system, a home page (Figure 14) shows suggested activities, together with a progress bar that shows the current development, and featured programs in the site. A left side bar leads to accomplishments, statistics and the community.

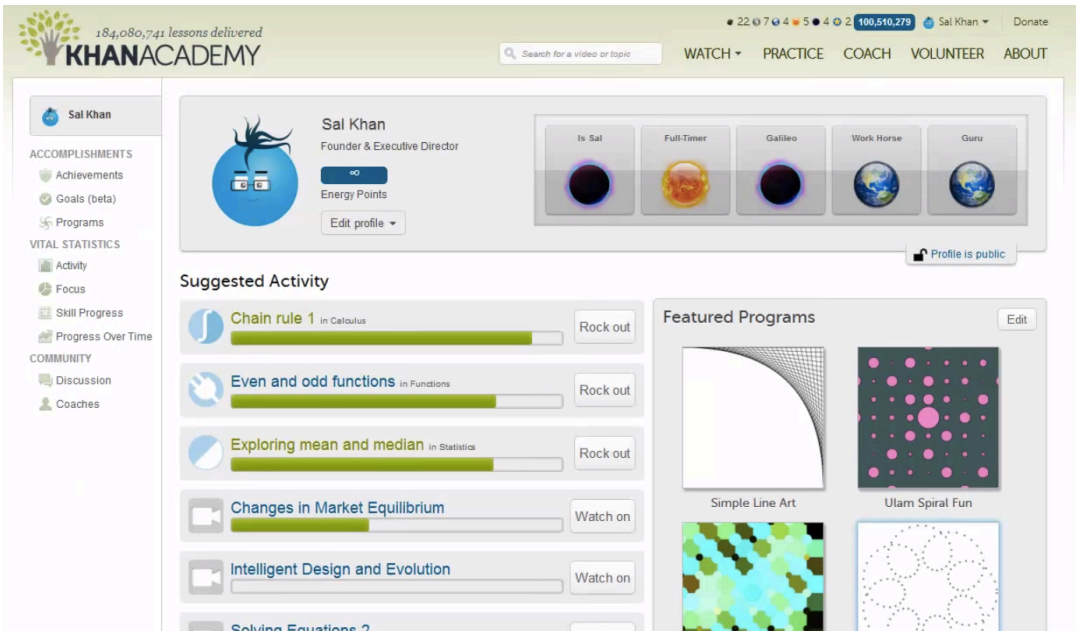

Figure 14 – Khan Academy home page

The two main actions a student can take are to watch and to practice. Watching implies learning new material through videos. Practicing involves honing specific skills. From the practice view, a knowledge map with an astral theme is shown, as can be seen in Figure 15, showing the skills and areas the student has mastered and which he can practice.

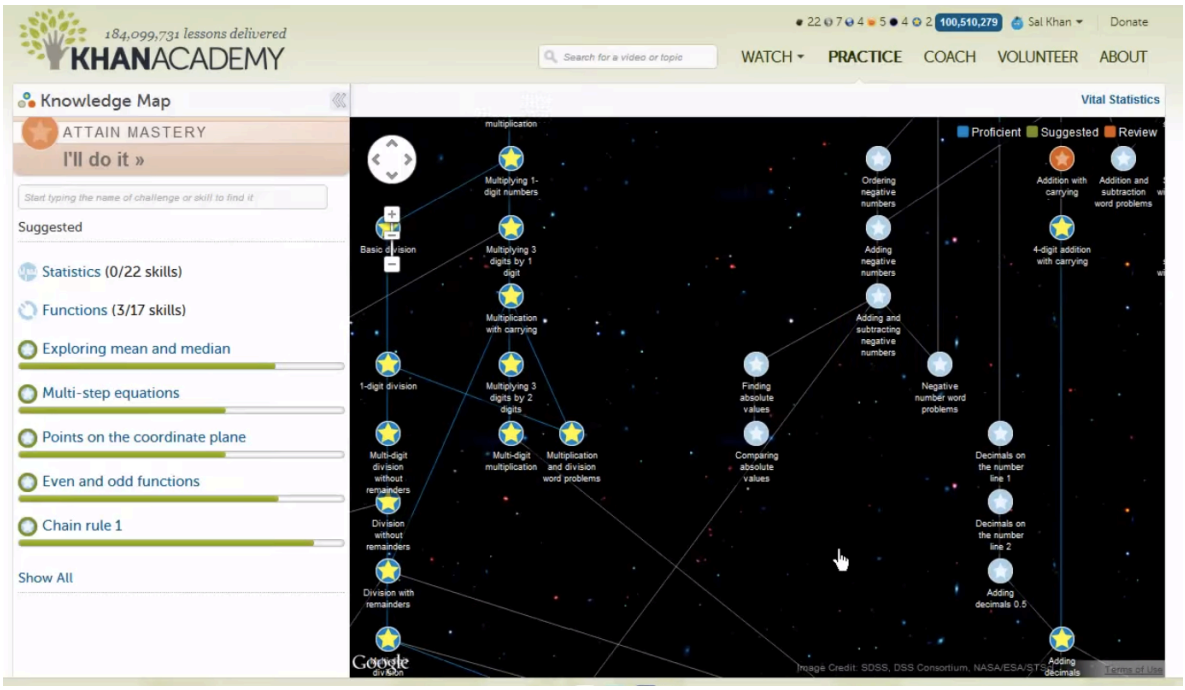

Figure 15 – Khan Academy knowledge map

Once an activity is chosen, the student is faced with one or several exercises related to the selected area of knowledge. Stars on the left become complete as the student successfully answers the questions presented on the right section of the screen. An example of how an activity looks like can be seen in Figure 16.

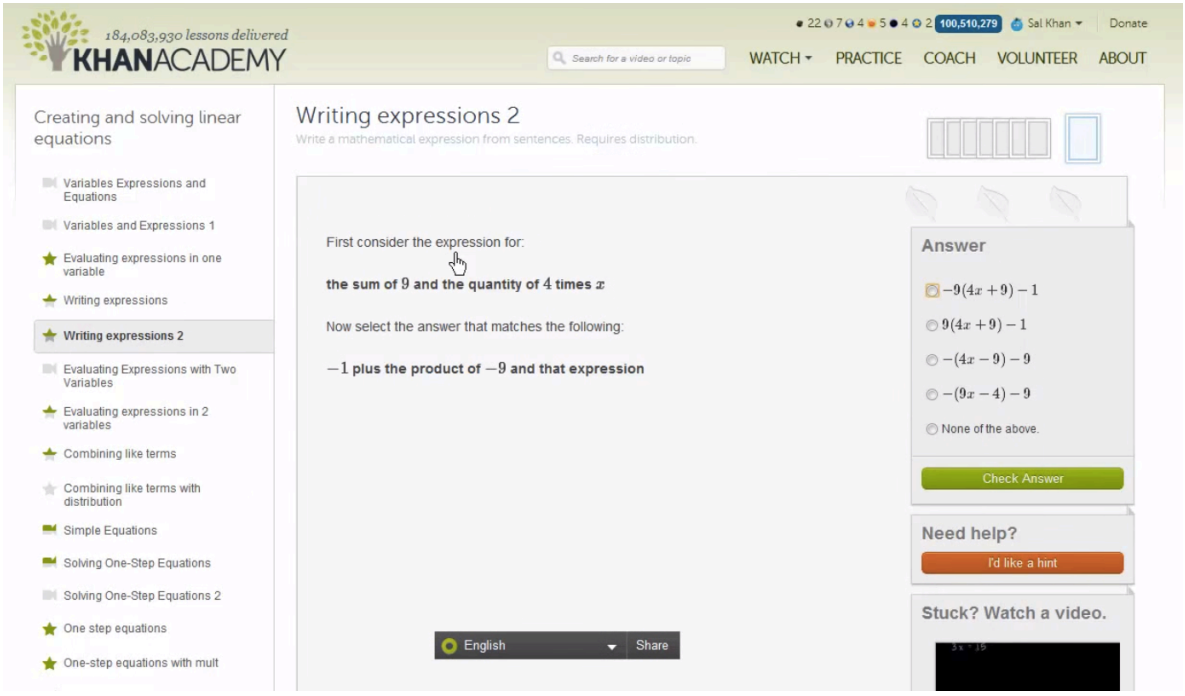

Figure 16 – Khan Academy Writing expressions exercise

Teachers and mentors can access all of their students' data as they can get a summary of class performance as a whole or dive into a particular student's profile to figure out exactly which topics are problematic after checking all the stats and reports associated with the course. Teachers can assign badges to users and student can start earning badges and points for learning.

2.1.2.6 Olds College

Olds College [16] is a college found in Alberta, Canada, and, since September 2013, if offers a compulsory gamified course "The Spirit of Entrepreneurship". Most gamified education is focused on primary and secondary education learning, but Olds College has taken a step to apply gamification on higher education and aims to further use it in the future [17].

Spirit of Entrepreneurship is the first comprehensive course delivered on an iPad app through a video game experience. Students learn to start and run their own lemonade stand empire in a social and mobile environment [18]. The objective of this app is to make students more engaged, have them retain what they learn, have fun, and want to learn more.

Any student who aims to graduate with a certificate, a diploma, or an applied degree must complete the Spirit of Entrepreneurship course. The goal of the course is for students to learn how to start and run a lemonade stand, all with entrepreneur concepts on the

background. The app starts out simply: students buy and sell lemonade stand inventory to other citizens. A simple lemonade stand can be seen in Figure 17, with three lemonade pitchers that can be refilled.

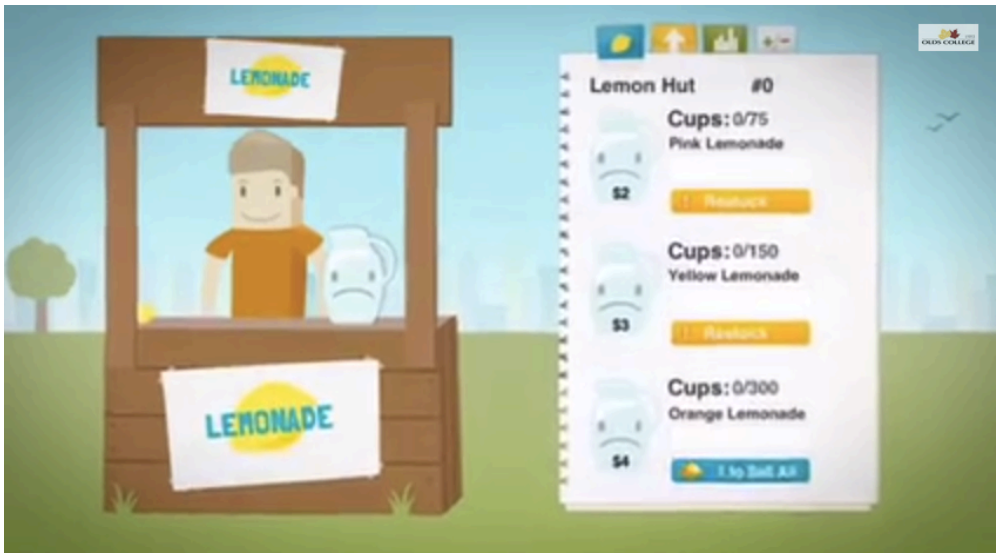

Figure 17 – Spirit of Entrepreneurship lemonade stand

Mini-games and awards encourage students to move through 12 modules of comprehensive content. As they progress through the content, more activities are unlocked in the game. As students become more skilled, they are able to run their stands more effectively and, hopefully, more profitably. Students are required to assess their business environment, make financial decisions, and apply marketing strategies within the game itself. They are also able to review slide notes and interact with each other in real-time [18].

Although this app is defined as gamified by Olds College, we must consider whether it is truly so. If we stop to analyze the Spirit of Entrepreneurship application, we can see it contains an immersive gameplay, which gamification does not include. This categorizes the app as a serious game, as it is in a game context, while gamification applies game design in non-game contexts.

2.2 Software Engineering – UWE

UWE (UML-based Web Engineering) [7] is a software engineering approach for the Web domain aiming to cover the whole life cycle of Web application development. The key aspect that distinguishes UWE is the reliance on standards.

The main focus of the UWE approach [19] is to provide a:

- UML-based domain specific modeling language
- Model-driven methodology

- Tool support for the systematic design
- Tool support for the (semi) automatic generation of Web applications

The characteristic of UWE is the fact to be an approach based on standards which is not limited to the use of the "lingua franca" UML (Unified Modeling Language) but also uses XMI (XML Metadata Interchange) as a model exchange format, MOF (Meta-Object Facility) for meta-modeling, the model-driven principles of the MDA (Model-Driven Architecture) approach, the model transformation language QVT (Query/View/Transformation), and XML (Extensible Markup Language).

The main reasons for using the extension mechanisms of the UML instead of a proprietary modeling techniques is the acceptance of the UML in the development of software systems, flexibility for the definition of a Web domain specific modeling language: a so-called UML profile, and wide visual modeling support by existing UML CASE tools.

UWE uses "pure" UML notation and UML diagram types whenever possible for the analysis and design of Web applications, i.e. without extensions of any type. For the Web specific features, such as nodes and links of the hypertext structure, the UWE profile includes stereotypes, tagged values and constraints defined for the modeling elements. The UWE extension covers navigation, presentation, business processes and adaptation aspects. The UWE notation is defined as a lightweight extension of the UML.

The UWE design approach for Web business processes consists of introducing specific process classes that are part of a separate process model with a defined interface to the navigation model. To model adaptive features of Web applications in a non-invasively way, UWE uses techniques of aspect-oriented modeling (AOM). Following the separation of concerns principle UWE proposes to build an adaptive model for personalized or context-dependent systems and weave the models afterwards.

Computer aided design using the UWE method is possible using MagicUWE, a plugin for MagicDraw.

There are five models used in UWE [20]:
1. Requirements
2. Content
3. Navigation
4. Presentation
5. Process

The **Requirements Model** consists of two parts: use cases of the application and their relationships, and activities describing use cases in detail. Use cases are distinguished by the stereotypes «browsing» and «processing» to indicate whether an application modifies

the persistent data of the application or not. Figure 18 displays the three icons useable in UWE use case diagrams.

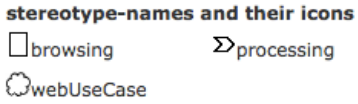

Figure 18 – Use Case icons

As use cases can only capture limited information, each use case can be described more accurately by a process flow. Thus, the actions within a use case as well as the data presented to the user and the required input can be modeled. Figure 19 shows the stereotypes applicable to activities.

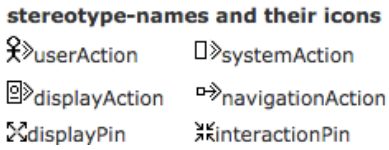

Figure 19 – Activity icons

First, the two stereotypes «user Action» and «system Action» can be used analogously to process flows. The «user Action» stereotype is used to indicate user interactions on the web page initiating a process or responding to an explicit requirement of information. By contrast, «system Action» describes actions, which are executed by the system. Details of the used data structures can be represented by object nodes and action pins. The object node is used to model content classes and the pins their attributes.

During requirements engineering it is usually also useful to specify which data is presented where and when. For the modeling of presentation groups in UWE the stereotype «display Action» is used, while the two action pin stereotypes «display Pin» and «interaction Pin» are used to model the output and input elements. Finally there is the stereotype «navigation Action», which can be used to model the navigation options and the associated presentation elements.

The **Content Model** is a normal UML class diagram; therefore we have to think about the structure of the system. We must consider classes, attributes, methods, and relationships between objects. This model is presented as class boxes, with a top division conformed of the class name, and a bottom division that shows the associated attributes and their respective types. Each box can then be tied via compositions or associations.

The **Navigation Model** shows how the different pages are linked together in a diagram containing nodes and links. Nodes are navigation units connected by links. Nodes can be shown on different pages or on the same page. UWE provides new stereotypes for this

model, which can be seen in Figure 20. For nodes and links the stereotypes «navigationClass» and «navigationLink» are used.

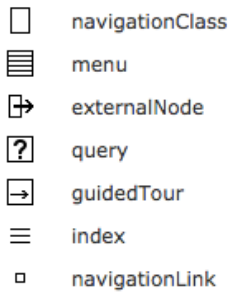

Figure 20 – Navigation icons

The homepage is indicated with the tagged value {isHome}. An additional aim is to have an application where you can access the operations from the Use Case Diagram. For this reason a homepage with connections to many different nodes is needed. In UWE, the «menu» stereotype can be used, if you want to navigate to different classes. A «processLink» stereotype is used for leaving associations, specifically directed ones so that backwards navigation is not possible.

The **Presentation Model** provides an abstract view on the user interface (UI) of a web application. It is based on the navigation model. The presentation model abstracts from concrete aspects of the UI, like the use of colors, fonts, and where the UI elements are placed on the web page; instead, the presentation model describes the basic structure of the user interface, i.e., which UI elements (e.g. text, images, anchors, forms) are used to present the navigation nodes. Also, the UI elements do not represent concrete components of any presentation technology but rather describe what functionality is required at that particular point in the user interface. This could simply mean that a text or image has to be displayed or for example that the user should be enabled to trigger a transition in the navigation model. In the last case, it is clear that an Anchor would be used in the UWE presentation model, but UWE does not define how the anchor should be rendered in the final web application. This could of course be just an anchor element of HTML (<a>), but also a button could serve the purpose.

The basic elements of a presentation model are the presentation classes, which are directly based on nodes from the navigation model, i.e. navigation classes, menus, access primitives, and process classes. Presentation classes can contain other presentation elements. This is accomplished through presentation properties that use the included presentation elements as type. In the case of UI elements, like text or image, the presentation property is associated with a navigation property that contains the content to be rendered.

The inclusion of presentation classes into other presentation classes or pages leads to a tree of presentation classes that are shown together. This means that the links between their corresponding navigation nodes are effectively "followed automatically". On the other hand, if two presentation classes do not belong to the same inclusion tree, then the link between their navigation nodes has to be triggered by user action.

In contrast to presentation classes and pages, a presentation group defines a set of presentation classes that are shown alternatively, depending on navigation. In the sense of the description above, a presentation group creates a set of alternative inclusion trees.

The stereotype names and their icons can be seen in the Figure 21:

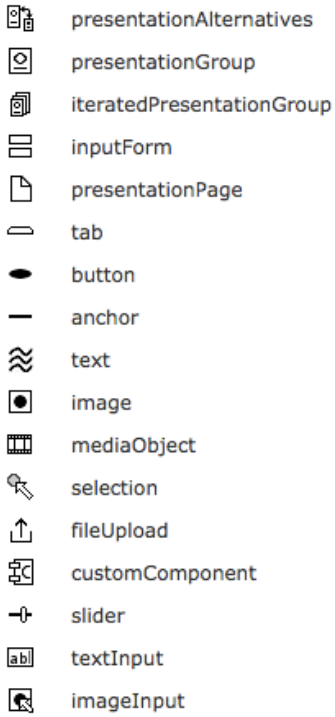

Figure 21 – Presentation Model stereotypes

2.3 Web Accessibility

Web Content Accessibility Guidelines (WCAG) [21] is developed through the World Wide Web Consortium (W3C) process in cooperation with individuals and organizations around the world, with a goal of proving a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally.

The WCAG documents explain how to make web content more accessible to people with disabilities. Web content generally refers to the information in a web page or web application, including:

- Natural information such as text, images, and sounds
- Code or markup that defines structure, presentation, etc.

WCAG is primarily intended for:

- Web content developers (page authors, site designers, etc.)
- Web authoring tool developers
- Web accessibility evaluation tool developers
- Others who want or need a standard for web accessibility

Related resources are intended to meet the needs of many different people, including policy makers, managers, researchers, and others.

The most updated WCAG document is WCAG 2.0 [22], and it is a stable, reference worthy, technical standard. The four primary principles of WCAG 2.0 are:

1. Perceivable: Information and user interface components must be presentable to users in ways they can perceive.
2. Operable: User interface components and navigation must be operable.
3. Understandable: Information and the operation of user interface must be understandable.
4. Robust: Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

These four principles are broken into twelve total guidelines. These guidelines are further divided into success criteria, for a total of sixty-one success criteria. For each guideline testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary, such as in design specification, purchasing, regulation and contractual agreements. In order to meet the needs of different groups and different situations, three levels of conformance are defined: A (lowest), AA, and AAA (highest).

2.4 Web Programing

2.4.1 HTML5

HTML (HyperText Markup Language) is the standard markup language used to describe web pages. HTML5 [23] is the successor of HTML 4, and was completed in October 2014. HTML5 is designed to be cross-platform, regardless of operating system or web browser. It is also designed to be backwards compatible with existing browsers.

HTML5 adds many new features to the standard, and attempts to replace external multimedia extensions such as Flash with open standards. Some of the new functionalities

involve audio support, video support, web storage, drag and drop, scalable vector graphics and geo-location among others [24]. A new feature enables sites to store data larger than a cookie in the client-side, and allows caching files images and data manually. Additionally, offline caching is also provided, permitting viewing when connectivity is lost.

2.4.2 CSS3

CSS (Cascading Style Sheets) is a style sheet language that is used to style and manipulate the look and format of a document written in a markup language. CSS is used as the default style sheet language in most websites to create visually attractive webpages and user interfaces in applications.

CSS3 is the latest CSS standard. The CSS3 specification is still under development by W3C. However, many of its properties have already been implemented in modern browsers. CSS3 is divided into modules, each adding or extending capabilities previously defined in CSS2. CSS3 is backwards compatible with previous CSS versions [25].

2.4.3 PHP

PHP (PHP Hypertext Preprocessor) is a server-side scripting language originally designed for web development, but now also used as a general-purpose programming language. No official standard exists for PHP, however a formal specification is been worked on since 2014. PHP is very easy to use together with HTML code. PHP tends to be processed by a PHP interpreter, which should be implemented as a native module in the web server. The code is interpreted and then executed, generating an output that is sent to the client.

PHP works on the most commonly used platforms (Windows, Linux, Unix, Mac OS X, etc.), is compatible with almost all servers used today (Apache, IIS, etc.) and supports a wide range of databases [26].

2.4.4 SQL - MySQL

SQL (Structured Query Language) is a programming language designed to access and manage information stored in Relational Database Management Systems (RDBMS). SQL includes being able to insert data, query, update and delete, creating and modifying schema, and control data access [27].

SQL became an ANSI (American National Standards Institute) standard in 1986, and an ISO (International Organization for Standardization) standard in the following year. There are different versions of the language, however, to comply with the standard, major commands exist in a similar manner (as SELECT, UPDATE, DELETE, INSERT, WHERE, etc.). Nevertheless, it is not completely portable between different database systems unless properly adjusted.

MySQL is the second most used open-source RDBMS and the most popular database system used with PHP [28]. MySQL can be used for small and large projects and is very fast and reliable. Additionally, it compiles on a large number of platforms.

2.5 Development Environment – PhpStorm

PhpStorm is an Integrated Development Environment (IDE), which aims to ease development [29]. PhpStorm's key features include:

1. **Smart PHP code editor:** An editor that deeply understands the code's structure, supporting PHP 5.3, 5.4, 5.5 and 5.6 for modern and legacy projects. It provides code auto-completion, refactoring, on-the-fly error prevention, supports language mixtures and more.
2. **Code quality analysis:** Hundreds of inspections verify the code as it is typed, analyzing the whole project. PHPDoc support, code (re)arranger, code formatter, PHP Code Sniffer and Mess Detector, quick-fixes and other features help writing neat code that is easy to support.
3. **Development environment:** Many routine tasks can be performed from the IDE with the version control system integration (Git, SVN, etc.), local history, support for remote deployment, remote PHP interpreters, Behat, SQL and databases, command-line tools, Vagrant, Composer, built-in REST Client and SSH Console with remote tools.
4. **Debugging and testing:** Zero-configuration debugging aids the debugging of applications, especially with debugger configuration validation. PHPUnit support allows for the development and running of unit tests from the IDE. Applications can be profiled with Xdebug or Zend Debugger and check aggregated reports in PhpStorm.
5. **HTML/CSS/JavaScript editor:** All popular front-end technologies are supported, including HTML5, CSS, Sass, SCSS, Less, Stylus, Compass, CoffeeScript, TypeScript, ECMAScript Harmony, Jade, Emmet, and JavaScript, with refactorings, debugging and unit-testing. There is also Live Edit to see changes instantly in the browser.
6. **Cross-platform experience:** Allows for the updated PHP integrated development environment on Windows, Mac OS X or Linux with a single license key.

3. Development

3.1 Software Requirements Specification

3.1.1 Introduction

3.1.1.1 Purpose

The purpose of this document is to present a detailed description of the gamified activities which take part of the larger project of a gamified educational platform. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate, and how the system will react to external stimuli. This document is intended for developers and any other members that participate in this project.

3.1.1.2 Scope

This system of activities described is a module for the larger project of a gamified learning site for a specific subject (*Procesadores de Lenguajes*) in the Universidad Politécnica de Madrid.

The system will be designed to increase student participation in the subject. It is our objective to improve the learning process while keeping the user entertained and engaged in the system. In this project, several activities can be setup by professors and done by students. Rewards are given once an activity is completed to encourage the students. A database will be used to store the relevant usage data, which can later be used to examine the success of the system. The system should have a great accessibility by following the WCAG 2.0 (Web Content Accessibility Guidelines).

More specifically, the project aims to increase the student pass rate reducing student failure, automatize feedback and to make learning and applying knowledge more appealing to students. These goals are self-supportive, as faster and more interactive feedback help students feel more encouraged and ready to keep active in their learning, which ultimately leads to a greater possibility of passing the subject.

3.1.1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Active activity	Activity for which the current date stands between its ID and DD.
Activity	Educational exercises of many types that exist in the system, which can be created, viewed, edited or deleted by professors and taken by students.
Activity category	Whether an activity is active, inactive or incomplete.
Administrator	Super user with the ability to register professors and control over the whole system.
Database	Collection of all the information monitored by this system.
Inactive activity	Activity for which the ID is greater than the actual date, or for which the DD is less than the current date.
Incomplete activity	Activity that is either missing information because its creation wasn't completed or because its editing was unfinished.
Player	Student that participates in the system.
Professor	Person with permission to define activities and monitor students.
User	Professor or player.

Table 2 – Definition of terms

Acronym	Meaning
ID	Issue date
DD	Due date
UPM	*Universidad Politécnica de Madrid*
WCAG	Web Content Accessibility Guidelines

Table 3 – Acronyms and their meaning

3.1.1.4 References

The IEEE Recommended Practice for Software Requirements Specifications [30] was consulted.

3.1.1.5 Overview

The next chapter, the Overall Description section, gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter. The third chapter, Requirements Specification, is written primarily for the developers and describes in technical terms the details of the functionality of the product. Both sections of the document describe the same system in its entirety, but are expressed in a different language so as to cater to different audiences.

3.1.2 Overall Description

This section will give an overview of the whole system, explaining it in its context to show how it interacts with other systems and introduce the basic functionality of this system. Constraints and assumptions relevant to the system will also be presented.

3.1.2.1 Product Perspective

This system is part of a greater one, and thus must work together will the larger one. This system will need to communicate with the basic module (of the greater system), which in turn also needs to communicate with an external database to be able to register a user. Further communication with the greater system is necessary to request and return relevant information, such as the number of points a player has.

Since much data is used in this project, a place to store it will be needed. For this, a database will be used. This database will be independent of the database the greater system might use, but they must be somewhat similar in the way they store data.

3.1.2.2 Product Functions

In this system, users will be able to perform different actions depending on their role. If the user is a player, the user will mainly be able to participate in various activities, receiving rewards for each of them. Only certain activities will be available to players, and of these, a certain player might not meet the necessary requirements to take it, thus it would not appear in his activity list.

If the user is a professor, many more options are available. A professor is able to create new activities from scratch or by using an existing activity as a model. He is also able to view, review, or edit any existing activity. This involves being able to see all information regarding that specific activity and then also being able to view how players have interacted with them, as well as changing the scores they have obtained while reviewing. Editing similarly includes seeing all activity information, but additionally permits the professor to change this information. Furthermore, a professor can change ID and DD of any activity, thus making it available or unavailable to players. Finally, a professor can decide to complete delete an activity from the system.

3.1.2.3 User Characteristics

As mentioned in the previous section, there are two main types of users that will interact with the system: professors and players. It should also be mentioned that there will be a system administrator or super user with control over it. As explained before, students can only use the system to participate in activities available to them. Professors, however, while not able to actually participate in activities, have the capability to manage them. The administrator is more similar to the role of professor as he will act as a manager of the whole system.

3.1.2.4 General Constraints

The Internet connection of the users is a constraint for the system given that it is necessary to access it and to obtain and import data to the database. The greater system mentioned must be available and functioning correctly to ensure this system can also function

correctly, as the communication between them is essential. Furthermore, the availability of the external database places a constraint for the functioning of the registration process.

3.1.2.5 Assumptions and Dependencies

An assumption made in the system is that it will be accessed mostly from a laptop or desktop computer, and less or not at all from mobile devices.

3.1.3 Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

3.1.3.1 External Interface Requirements

A link to an external system is the link to the UPM student database to verify the identity of the users. In this way we can be certain that only the intended audience uses the platform. Once this identity is checked, the new user will be added to the system's internal user database as either a student or a professor.

Communicating with the main module interface of this project will also be necessary for management of internal data.

Any further detailing of external interfaces escapes the scope of this project.

3.1.3.2 Functional Requirements

This section describes specific features of the software project.

3.1.3.2.1 User Registration

3.1.3.2.1.1 Requirement #1:
The system interface will present an initial screen whereupon a person can either register or identify himself into the system.

3.1.3.2.1.2 Requirement #2:
The form to fill in to be registered into the system will contain the fields: student ID, registered ID, name, surname, second surname (optional), desired username, password and email. The external student database will be consulted, and the response shown to the user. If the registration is successful, the user will be redirected to the initial screen.

3.1.3.2.1.3 Requirement #3:
The password will be asked for twice and be checked for a match, showing an alert if the match is negative. If a username is already in use, the user must choose a different one. Any other data to be considered in a wrong format will be asked to be corrected before being able to submit a request to register. If that particular student has already created an account, he cannot create a new one.

3.1.3.2.1.4 Requirement #4:
In case of a failed registration due to the student not being found in the external database, the student will be asked to try again or to contact a professor to verify whether the student exists in the external database.

3.1.3.2.1.5 Requirement #5:
A professor must be registered by an administrator of the system. Data required includes: name, surname, username, password and email.

3.1.3.2.1.6 Requirement #6:
Every registered user can be classified through a unique identifier, with students being identified through their student ID and professors and administrators directly through their username.

3.1.3.2.2 Identification

3.1.3.2.2.1 Requirement #7:
Once the user is logged into the system, a logout option will be present in all screens.

3.1.3.2.2.2 Requirement #8:
The identification form requires the student id or username and the password. If any data is incorrect, the user will be prompted to try again and an option for password recovery will be presented. If the data is correct, the user will be redirected to the home screen.

3.1.3.2.2.3 Requirement #9:
Password recovery will be allowed in case a user cannot remember the password (Requirement #8). The password recovery form will require the user to enter the provided email. A link will then be sent via email from which a new password can be set.

3.1.3.2.3 Home screen

3.1.3.2.3.1 Requirement #10:
The home screen will be the first view to appear when a user is identified into the system. It will contain a logout option (Requirement #7), and a link to activities.

3.1.3.2.4 Activities screen

3.1.3.2.4.1 Requirement #11:
The activities screen will contain a list of all available activities. If the user is a player, these activities will be only those that he can participate in. If the user is a professor, all activities will be shown, whether they are active, inactive or incomplete.

3.1.3.2.4.2 Requirement #12:

From this screen, a player can select an activity to participate in. This activity must still be active when it is selected.

3.1.3.2.4.3 Requirement #13:

An activity is defined as available to a player if the current date is between the ID and the DD, and he meets the requirements for the activity. These requirements involve player level, random assignment, and practice group.

3.1.3.2.4.4 Requirement #14:

If the user is a professor, options to add activities, review, view, close, delete, or edit existing ones are shown. An option to view, review and edit will appear near each activity. An option to delete will only be available if the activity is inactive or incomplete, an option to close only if the activity is active, and an option to open only if the activity is inactive. A button to add an activity will be placed in the activities section of this screen.

3.1.3.2.4.5 Requirement #15:

Practical part activities are seen as special activities and do not share the same functionality and operations that the rest of activities do.

3.1.3.2.4.6 Requirement #16:

The list of activities a professor sees can be filtered by different parameters, such as category, ID, DD or points.

3.1.3.2.5 Activity creation

3.1.3.2.5.1 Requirement #17:

Only professors have access to this function. From this screen, an activity can be added to the activity list. First, the professor must decide if he wishes to create a new activity from scratch, or if he wishes to use an existing activity as a model.

3.1.3.2.5.2 Requirement #18:

If professor decides to use a model, he will be taken to a screen where the full list of activities can be seen and filtered. A copy option and a view option will appear near all listed activities, with the exception of practical part correction activities (Requirement #15). The professor should select the copy option of the activity he wishes to use as a model. Once the model is chosen, a screen where all activity information can be seen and edited. All the fields can be changed, with the exception of the activity type. If certain fields are changed, filling new fields with relationship with the one changed may be required.

3.1.3.2.5.3 Requirement #19:

The creation of a brand new activity leads to a form screen from which the type of activity to be created must be selected from: questions with single or multiple correct answers, short answer questions, sequential questions, long answer questions, ranking questions, faulty

questions, cooperative questions, etc. Afterwards, several more fields must be filled in according to the activity type selected.

3.1.3.2.5.4 Requirement #20:

As an activity is being created, any and all progress will be saved so that it appears in the uncompleted activities category until it is finished and saved either to the active or inactive activities category.

3.1.3.2.6 Activity deletion

3.1.3.2.6.1 Requirement #21:

Activities that appear in the inactive or incomplete category can be deleted (Requirement #14). A confirmation will be asked from the professor before deleting. Once the activity is deleted, it will be removed from the system.

3.1.3.2.7 Activity management

3.1.3.2.7.1 Requirement #22:

Activating an inactive activity, via choosing the open option, presents a short form to the professor in which he must enter a new ID and DD, where the default ID will be the current date. If the current date is not between these dates, a warning will be shown, but the action will be allowed.

3.1.3.2.7.2 Requirement #23:

The viewing of an activity (Requirements #15 and #18) leads to a screen where all activity information can be seen. This includes all the fields specified when it was created as well as data according to player action such as number of times the activity has been taken and the average score of those players.

3.1.3.2.7.3 Requirement #24:

The editing of an activity (Requirement #14) leads to a screen where all activity information can be seen and edited. When an activity is being edited, it will automatically be deactivated and temporally placed as an incomplete activity, a warning will be shown to communicate this. Editing involves all the fields specified when it was created with the exception of the activity type, which cannot be modified. If certain fields are changed, filling or editing new fields with relationship with the one changed may be required. Once the editing is finished, the activity is re-enabled and removed from the incomplete category.

3.1.3.2.7.4 Requirement #25:

Closing an activity (Requirement #14) automatically changes the DD of the activity to the current date so that it is no longer active. The closed activity can from then onwards be found in the inactive category.

3.1.3.2.8 Activity selection

3.1.3.2.8.1 Requirement #26:

When a professor chooses to review an activity, he is taken to a revision screen (Requirement #14). From this screen, a professor can not only passively review the answers a player has submitted, he can actively modify the score given if that specific type of activity allows for it. A list of all student answers are displayed in a summarized manner, that is, the player's username appears together with the obtained score and other such relevant data.

3.1.3.2.8.2 Requirement #27:

From the revision screen (Requirement #26), when a professor selects a player's username, he is taken to an individual revision screen for that player. Here, the player's full answer or answers are shown and the score can be modified for all activities.

3.1.3.2.8.3 Requirement #28

When a player selects an activity, he is taken to another screen (Requirement #12). In this screen, he begins to answer the exercise proposed by the activity. When the student finishes the activity, he is notified of his score if immediate feedback is available and awarded certain rewards.

3.1.3.2.8.4 Requirement #29:

Once a player has finished taking an activity, his results can be reviewed by the professor (Requirement #26).

3.1.3.3 Software System Attributes

Software system attributes describe those that must be achieved at a system-wide level rather than at a unit level.

3.1.3.3.1 Performance

Since the system will be accessed through the Internet, the response time of the system to requests will depend on the transmission speed and hardware used by the user. Regardless, response times are expected to be below 4 seconds for any given user at any given time.

3.1.3.3.2 Availability

The system must be available at least during the course. The ability of the registration process to function depends on the availability of the external database necessary to complete the process.

3.1.3.3.3 Security

The system will be developed using techniques that guarantee the privacy of user information. Information considered private includes passwords and, if the user is a player, the activity details of other players.

3.1.3.4 Design Constraints

The system design will not have any specific constraints applied to it so long as it is possible to then follow the WCAG 2.0 (Web Content Accessibility Guidelines) in a manner that the result is an accessible system.

3.1.3.5 Logical Database Requirements

This section specifies the logical requirements for any information that is to be placed into a database.

The system's database will have to be accessed whenever any information regarding activities is necessary. This includes: fetching a list of activities, obtaining that activity's information, obtaining information of player participation data, fetching the attributes of an activity such as its category, and more. The database will also have to be accessed to store information or modify existing information. Some such uses include: Editing information of an activity, creating an activity, participating in an activity and deleting an activity among others.

All data stored in the database must be retained as long as the system continues to operate. Only data expressively set to be deleted will no longer be retained in the database.

3.2 Activity Definition

Activities can be defined as educational exercises of many types, from quizzes to mock exams. This section presents a precise definition of the activities, their components, inner mechanics, etc. The following is a revised and expanded version of previous works [31][32] that included activity definitions.

3.2.1 Common Elements

There are some attributes that apply to most of the listed activities. Following is a list of these attributes with a small description for each:

- **Title**: All activity instances must have a title which somewhat describes its content so that they can be identified by users.
- **Always reward**: For all of the activities a minimum amount of points (or other rewards, such as badges) will be given to players. They might get a score of zero, which affects their final evaluation for the course, but they can still get points and

improve their account in other aspects. In any case the system should alert professors about possible negative effects of these rewards in case they are not choosing them carefully.

- **Obligatory/Optional elements**: For each activity there are some obligatory and optional elements. Optional elements are indicated by the word *Optional* in parentheses after their description. If it is not indicated that an element is optional it should be considered mandatory.
- **Choice min/max**: For questions with multiple choices there is a minimum and maximum number of answers (choices) to show to players. For example, a minimum of 4 and a maximum of 10 indicate that the actual number of answers shown will be between or including these limits.
- **Happy hour**: This is a time range in which students will be awarded more points
  - Two modes can be chosen for this event. One is hidden, which means players are not aware it is happy hour unless they attend something in the system and see the results with increased points. The other one is visible mode, which shows the event to the players. So before attending anything, players would know they will gain more rewards.
  - In case of a visible happy hour, a message will be shown after logging into system.
  - Professors should define the percentage of increased points or chance while defining the event.
  - Professors should define the start and end time of the event.
  - Professors can limit the number of actions a player can do during the happy hour or leave it unlimited.
- **Evaluation system**: The evaluation process is based on the amount of score a player gets or looses.
- **Reward:** Depending on how the player performs in the activity taken, he will get a certain number of points or even obtain a badge. Every time a player does well in an activity, depending on the activity, and how well the student has performed, the player will be closer to obtaining a badge. A specific activity might for example grant a 35% completion towards a badge, or give the full badge in itself.
- **Penalty:** As defined before, players always get points. If they answer a question incorrectly they will get fewer points. Therefore they will not be able to get the maximum points for the activity but they will still get something. The worse a student performs, the fewer points he will get.
- **Issue date:** Professor should define an issue date while creating a question. This date defines when the activity is accessible by players.
- **Due date**: Professor should define a due date while creating a question. This date defines the final date.
- **Visibility**: If the date is between the issue date and the due date, the activity will be visible to students.

- **Maximum amount of betting chips**: Professor can define for each activity some amount of betting chips that a player can get. This should consider the player's level and how well he performed on the activity.

3.2.2 Common parameters and constraints

Some parameters and restrictions are present in all activities and help categorize them. These items are listed and described following:

- **Required level:** Minimum level that the student must have in order to be able to do the activity, if the student's level is lower than the required one, he will not be able to do the activity. A required level of 0 represents an activity open to all students, regardless of their level.
- **Issue date (ID)**: Issue date can be any date that applies to following constraints
  - $ID >= Max\ (beginning\ of\ the\ course, today)$
  - $ID <= Final\ day\ of\ the\ course$
  - Date is in full format: Year-Month-Day Hour:Minute:Second
- **Due date (DD)**: Due date can be any date that applies to following constraints
  - $DD > ID$
  - $DD <= Final\ day\ of\ the\ course$
  - Date is in full format: Year-Month-Day Hour:Minute:Second
- **Time to deadline (TD)**: Number of total seconds remained before reaching the deadline of answering a question
  - $TD = DD - ID$ [$Calculated\ in\ seconds$]
- **Minimum number of points:** Add warning; minimum should be less than the least amount of points you can obtain which is greater than zero.
- **Betting chips:** Different levels get different amount of maximum specified number of betting chips.
  - Number of chips that player will get is as follows:
    - 10%: $0.1 \times Maximum\ amount\ of\ betting\ chips \times activity\ score$
    - 30%: $0.3 \times Maximum\ amount\ of\ betting\ chips \times activity\ score$
    - 50%: $0.5 \times Maximum\ amount\ of\ betting\ chips \times activity\ score$
    - 80%: $0.8 \times Maximum\ amount\ of\ betting\ chips \times activity\ score$
    - 100%: $1.0 \times Maximum\ amount\ of\ betting\ chips \times activity\ score$

3.2.3 Description of each activity

3.2.3.1 Questions with multiple or single correct answers

For this type there is a question defined by the professors (of the same course). The question has a number of answers that are presented with different choices. Depending on the definition of the question, students can select one or more of the choices as well as

leaving them unselected. Finally, when they submitted their answers, they will get a certain score for their correct answers.

Elements

Different elements for this type of activity are:

- **Question**: a simple question which is defined by the professor.
- **Answer**: a set of definitions that are provided as the solution for a defined question. The professor defines both questions and answers. They can be correct solutions or wrong choices. It is the responsibility of the student (player) to distinguish between them.
  - **Correct answer**: Correct answers are the ones that if they show up in available choices, the player should select them to get positive score. Otherwise he can get negative score.
  - **Wrong answer**: Wrong answers are the ones that if they show up in available choices, the player should avoid selecting them to get positive answer. Otherwise he can get negative score. (*Optional*)

  When multiple choices are shown to player, the distribution of the correct and wrong answers will be decided randomly by the system.
- **Correct choice**: A correct answer that has been selected by the player, as well as a wrong answer that is left unselected.
- **Incorrect choice**: A wrong answer that has been selected by the player, as well as a wrong answer that is left unselected.
- **Explanation**: For each correct or incorrect answer, the professor might provide an explanation so after performing the activity the player will know why his answer was right or wrong. (*Optional*)
- **Timer**: As the student starts to answer the question a timer will record the time that particular student takes to finish answering. (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Number of wrong answers (#WA)**: This number does not have any maximum value but it has the following constraint:
  - $\#WA >= 0$
- **Number of correct answers (#CA)**: This number does not have any maximum value but we should consider a minimum for that using the following formula:
  - $\#CA > 0$
- **Number of answers in total (#AT)**: This parameter is defined by sum of all correct and wrong answers defined for the question.

- $\#AT = \#WA + \#CA$

- **Number of correct answers to show in all choices (#CATS)**: This number has a minimum and maximum value defined by the professor while defining the question. When the system wants to show a question to the player, it will create a random number in this defined range which will be #CATS.
  - $\#CATS > 0$
  - $Min(\#CATS) > 1\ defined\ by\ professor$
  - $Max(\#CATS)$ is defined by professor and it is less than or equal to number of available choices.
  - $Min(\#CATS) <= \#CATS <= Max(\#CATS)$
- **Number of wrong answers to show in all choices (#WATS)**: This number has a minimum and maximum value defined by the professor while defining the question. When the system wants to show a question to the player, it will use the randomly generated #CATS to calculate this number using the following formula:
  - $\#WATS = Number\ of\ available\ choices\ to\ select - \#CATS$

  It should also follow these constraints:
  - $Min(\#WATS) > 0\ defined\ by\ professor$
  - *Max(#WATS)* defined by professor and less than number of available choices
  - $Min(\#WATS) <= \#WATS <= Max(\#WATS)$
- **Number of available choices to select (#AC)**: For each question there are a number of answers (choices) shown. This parameter defines how many these choices will be. The professor should define it for each question while creating it.
  - $\#AC = \#WATS + \#CATS$
  - $\#AC >= 4$
- **Number of correct choices by player (#CCBP)**: The system can calculate this number when the player makes his selections. It will be the sum of correct choices of the player.
  - $\#CCBP = \#AC - \#WCBP$
- **Number of wrong choices by player (#WCPB)**: The system can calculate this number when the player makes his selections. It will be the sum of incorrect choices which of the player.
  - $\#WCBP = \#AC - \#CCBP$
- **Amount of penalty score given to player (AOPS)**: Each wrong choice the player chooses will affect his final score. Here we have two approaches:
  - *First:* for each wrong answer the system does not give any score to the player:s
    - $AOPS = 0$
  - *Second:* for each wrong answer not only they get no score but also we penalize them by a percentage (defined by professor) of positive score for a correct answer. In this case the amount of negative score is calculated by following formula:
    - $AOPS = -[Penalty\ Percentage\#AC] \times \#WCBP$

- **Amount of award score given to player (AOAS)**: Each correct choice the player chooses will affect his final score. This amount is calculated by following formula:
  - $AOAS = \#CCBP/\#AC$
- **Total score given to player (TS)**: This is the sum of award and penalty given to the player.
  - $TS = AOAS + AOPS >= 0$
- **Time to Answer (#TA)**: Number of total seconds that a player has to answer a question. If it takes the player less than this amount to finish answering he will be rewarded.
  - $\#TA > 0$
- **Time Taken to Answer (#TTA):** Number of seconds passed from the moment that the player started answering the question.
  - $\#TTA > 0$
- **Time Bonus Points (#TBP):** This is an amount of bonus points given to player according to his timings.
  - **Maximum time bonus points:** This is the maximum amount of bonus points that a player will get it he performs 100% efficient (according to timing) on the activity. The professor defines this number. To keep the competition balanced the following constraint should be taken into account while defining this number.
    - $Maximum\ time\ bonus\ points <= Maximum\ number\ of\ points$
  - **Actual time bonus points (ATBP):**
    - $ATBP = (\#TA - \#TTA) \times \#TA \times Max\ number\ of\ bonus\ points$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:** Professor should define a maximum number of points for a question. The maximum point will be given to players who could get the total score of the question.
    - **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
    - **Actual number of points (APs):** Following formula calculates this parameter.
      - $APS = TS \times [Max\ number\ of\ points + ATBP] + Min\ number\ of\ points$
  - *Betting chips*

3.2.3.2 Short answer questions

In this type of activity, professors (of the same course) will define a question with a short answer. As these short answers might be expressed by different phrases, professors should try to cover all possible correct phrases. The answer provided by the student will be compared to correct phrases and in case of a correct match he will get positive score.

Elements

Different elements for this type of activity are:

- **Question**: a simple question which is defined by the professor.
- **Answer**: a set of keywords provided as the solution for a defined question. Professor defines both question and answers.
- **Correct answer**: one or more keywords provided by student which matches one or more keyword(s) that is provided by professors as correct answer.
- **Wrong answer**: any other input by students which does not match keyword(s) that is provided by professors as correct answer.
- **Explanation**: The professor might provide an explanation so that after performing the activity, the player will know why his answer was right or wrong. (*Optional*)
- **Timer**: As the student starts to answer the question a timer will record the time that particular student takes to finish answering. (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Number of answers (#A)**: This number does not have any maximum value but we should consider a minimum for that using the following formula:
  - $\#CA > 0$
- **Total score given to player (TS)**: Player gets full score in case of writing correct sentence/phrase. Otherwise gets nothing.
  - $TS = 1\ in\ case\ of\ correct\ answer$
  - $TS = 0\ in\ case\ of\ wrong\ answer$
- **Time to Answer (#TA)**: Number of total seconds that a player has to answer a question. If it takes the player less than this amount to finish answering he will be rewarded.
  - $\#TA > 0$
- **Time Taken to Answer (#TTA):** Number of seconds passed from the moment that the player started answering the question.
  - $\#TTA > 0$

- **Time Bonus Points (#TBP):** This an amount of bonus points given to player according to his timings.
  - **Maximum time bonus points:** This is the maximum amount of bonus points that a player will get it he performs 100% efficient (according to timing) on the activity. The professor defines this number. To keep the competition balanced the following constraint should be taken into account while defining this number.
    - $Maximum\ time\ bonus\ points\ <=\ Maximum\ number\ of\ points$
  - **Actual time bonus points (ATBP):**
    - $ATBP = (\#TA - \#TTA)\#TA \times Max\ number\ of\ bonus\ points$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:** Professor should define a maximum number of points for a question. The maximum point will be given to players who could get the total score of the question.
    - **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
    - **Actual number of points (APs):** Following formula calculates this parameter.
      - $APS = TS \times [Max\ number\ of\ points + ATBP] +\ Min\ number\ of\ points$
  - *Betting chips*

3.2.3.3 Sequential questions

This activity is combined of all previously described activities. There are two types of this activity depending on whether the professor defines the sequence manually or it is generated automatically by system. Professors can select one of these two types while creating the activity. When a player accepts to attend a sequential activity and finishes the elements of the sequence successfully he will get rewards. The more elements he can pass successfully the more he will be rewarded. If player fails at one point, he cannot proceed with next elements of sequence. To make this activity more challenging, there is a certain amount of risk involved in the choice of continuing from one element to next one. This means, if you stop at one point you will get the amount of points you have got till that step but if you decide to proceed you will lose a portion of points you have got depending on how proceed rate and risk rate is defined.

Elements

Different elements for this type of activity are:

- **Starting number:** number of points given if the first sequence is overcome. The professor defines this number and, if the professor allows it, a student can choose a number as well.
- **Number of Sequences:** This is the number of items for the activity chain. The professors define this number.
- **Proceed Rate:** This number specifies how much more score a player will get if he proceeds to the next item in sequence.
- **Risk Rate:** This number specifies how much score a player will lose (should invest) if he proceeds to the next item in sequence.

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Starting Number of Points (SNOP):** If a professor allows for it, a student can choose to use the default starting points or bet his own points as starting amount. If not, the SNOP will be the one decided by the professor.
  - $SNOP \leq amount\ defined\ by\ professor$
- **Number of Sequences (#OS):** Here just one constrain should be taken into account.
  - $\#OS > 1$
- **Proceed Rate (PR):**
  - $PR > 1.0$
- **Risk Rate (RR):**
  - $RR = 0$ in case we want player to lose all the points as he proceeds
  - $RR = 1$ in case we do not want player to lose anything in case of failure
  - $0 < RR < 1$ in case we want something in between the two previous cases
  - $RR < PR$
- **Loss Amount After Proceed (LAAP):**
  - $LAAP = \lfloor RR \times Current\ gained\ points \rfloor$
- **Gain Amount After Proceed (GAAP):**
  - $GAAP = \lfloor PR \times Current\ gained\ points \rfloor$

Figure 22 shows how this kind of activity works with a risk rate of 0.5 and a proceed rate of 1.5. In this case the player is not betting any points, so the starting point is defined by the professor.

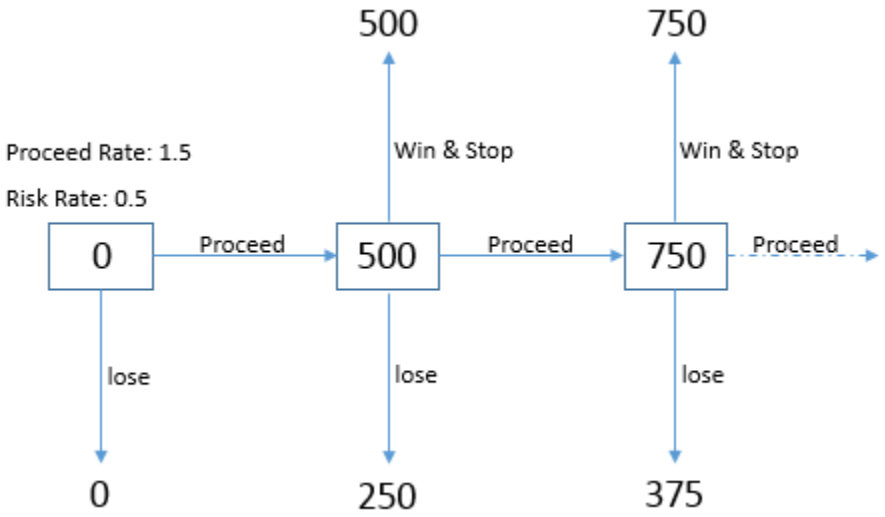

Figure 22 – Mechanism of activity in case professor defines initial point

Figure 23 shows how this kind of activity works with a risk rate of 0.5 and a proceed rate of 1.5. In this case the player is betting some points as initial starting point.

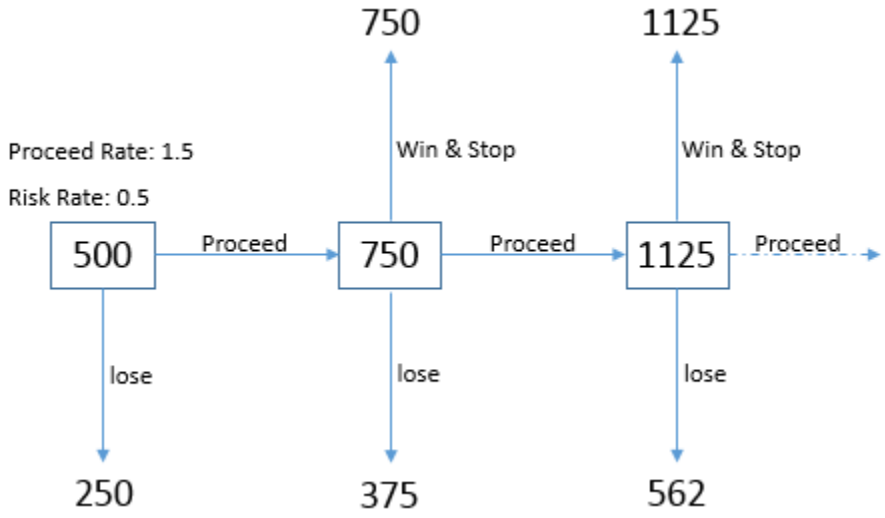

Figure 23 – Mechanism of activity if player bets some initial points

- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:**
      - Start points defined by professor

        - $MNP = SNOP \times (PR)^{\#OS-2}$
    - Start points defined by player (bet)
        - $MNP = SNOP \times (PR)^{\#OS-1}$
- **Minimum points:**
    - Start points defined by professor
        - $MP = 0$
    - Start points defined by player (bet)
        - $MP = SNOP \times RR$
- **Gathered points (GPs):**
    - In case of Start points defined by professor
        - Given N is the number of sequences passed, $GPs(N) = SNOP \times (PR)^{N-2}$
    - In case of Start points defined by player
        - Given N is the number of sequences passed, $GPs(N) = SNOP \times (PR)^{N-1}$
- **Actual number of points (APs):**
    - In case of success
        - Given N is the number of sequences passed, $APs(N) = GPs$
    - In case of failure
        - In case player fails at first step $APs(1) = 0$
        - Given N is the number of sequences passed, $APs(N) = GPs \times RR$ $\quad N \geq 2$
    - Note that this activity does not have any minimum number of points and will not award any betting chips.

3.2.3.4 Long answer questions

In this type of activity, professors will define a question that has a long and descriptive answer. The answer provided by the student will be reviewed by other players or by the professor of the course, and it will then be scored according to their opinion.

Elements

Different elements for this type of activity are:

- **Question**: a simple or complicated question which is defined by the professor
- **Answer**: a long and descriptive answer that will be reviewed by others.
- **Reviewer**: Someone who reviews the answer given by the player. He can be the professor or another player or both.
- **Wrong answer**: An answer that gets no points from reviewers.

- **Review**: The action of grading the answer of a player. If reviewer is another player, the reviewer and review will be anonymous.
- **Additional info**: Reviewers can put some comment explaining their opinion on the answer (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Total score given to player (TS)**: According to the review(s) player will get a score from 0 to 10.
  - $0 \leq TS \leq 10$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:** Professor should define a maximum number of points for a question. The maximum point will be given to players who could get the total score of the question.
    - **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
    - **Actual number of points (APs):** Following formula calculates this parameter
      - $APS = [\frac{TS}{10} \times Max\ number\ of\ points] + Min\ number\ of\ points$

Points that are not mentioned are the same as section 3.2.3.2

Points regarding the matter of reviews are detailed in the following section.

3.2.3.5 Peer reviews

The system may assign reviews to players randomly, or allow them to choose to revise voluntarily. In the former case the player will get a notification informing him that he has to do a review. Players can do this review and get extra rewards (like badges, collectibles and points) or skip it without any penalty.

Elements

Different elements for this type of activity are:

- **Review mode:** The revisions can be automatically assigned, voluntary, or all done by the professor. The review mode allows the professor to choose which modes will be applied and how they will be used.
- **Expiration date:** After a specific time the activity will expire and players cannot attend it. The expiration date should be defined by professor in following format: Day-Month-Year Hour:Minute:Second
  - **Automatically assigned expiration date:** This is the expiration date for automatically assigned reviews.
  - **Voluntary expiration date:** This is the expiration date for voluntary reviews.
- **Follow up question:** Once the revision is finished, one or more questions about the quality of the revised answer will be presented. These questions will have multiple-choice answers to choose from and will allow the professor to understand the score given by that particular student. (Optional)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Defined Points (DPs):** It will be the number of points defined by professor for assigned reviews.
    - **Penalty Points (PPs):** A certain number of points may be subtracted from a player if the professor confirms that the review has a low quality.
    - **Actual number of points (APs):** Following formula calculates this parameter
      - $AP = DP - PP$
    - The points for this activity will be granted once it is deemed as finalized. For this to be so, the final expiration date must have finalized, the set number of required revisions must be reached and no significant variance in the given scores found (SV). If these parameters are not reached, the professor must manually review and finalize it.
- **Number of reviews to be done (NRs):** The professor defines the number of desired reviews per answer.
  - $NR > 0$
- **Max reviews per student (MRs):** This is the maximum number of revisions that a student may do (both mandatory and voluntary).
  - $MR > 0$
- **Mean given scores (MSs):** This is the calculated mean of the scores that players have given an answer.

- $MS = \frac{SS}{NRD}$ where SS is the sum of the scores given and NRD is the number of reviews done.

- **Score Variance (SV):** The variance of the MS is calculated to see if there is any significant discordance between the given scores. If the variance is large, the professor is notified.
  - $SV = 0$ when all students have given the same score

3.2.3.6 Ranking questions

In this type of activity, professors define a question with a number of answers that must be ranked. Depending on the definition of the question, students can select one or more of the choices to rank them, as well as leaving them unselected. Finally, when they submitted their answers, they will get a certain score for their correct answers.

Elements

Different elements for this type of activity are:

- **Question**: A simple question defined by the professor.
- **Answer**: A set of definitions or elements that are provided as the solution for a defined question. The professor defines both questions and answers. They can be correct or wrong solutions. It is the responsibility of the student (player) to distinguish between them.
  - **Correct ranking**: Correct ranking is the order that the player should indicate to get a positive score. Otherwise he can get negative score. An answer can have a higher, lower, or equal rank to another. In the case of equal rankings, the order is interchangeable.

  When multiple choices are shown to player, the distribution of the order of the answers will be decided randomly by the system.
- **Answer tuple:** A recollection of the ranked answers the student has proposed. This can be broken into various sets of different sizes (n-tuple), going from each individual answer to the whole set of answers.
- **Correct order**: A correct order that has been indicated by the player, this is the ranking of one particular ordered answer tuple with regard to another.
- **Incorrect order**: A wrong order that has been indicated by the player, this is the ranking of one particular ordered answer tuple with regard to another.
- **Explanation**: For each answer, the professor might provide an explanation as to why it is so ranked, so after performing the activity the player will know why his answer was right or wrong. (*Optional*)
- **Timer**: As the student starts to answer the question a timer will record the time that particular student takes to finish answering. (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Number of answers in total (#AT)**: This is the number of answers provided by the professor.
- **Number of available choices to select (#AC)**: For each question there are a number of answers (choices) shown. This parameter defines how many these choices will be. The professor should define it for each question while creating it.
  - $\#AC \geq 4$
  - $\#AC \leq \#AT$
- **Number of available choices to rank (#AR)**: Of the available choices shown, the number of them to be ranked can be limited. The professor should define it for each question while creating it.
  - $\#AC \geq \#AR \geq 2$
- **Number of available tuples to select (#ATS)**: For each question there are a number of tuples of answers shown.
  - As $\#AR \geq 2$, at least one pair will always exist: $\#2_{tuples} \geq 1$
  - $1 < n \leq \#AR$, a tuple base must be greater than 1, and can only be as large as the number of available choices to rank.
  - $\#n_{tuples} = \#AR - (n-1)$, the number of tuples for a certain tuple base is equal to the number of available choices to rank subtracted by base minus one.
- **Number of correct tuples chosen by player (#CTCBP)**: The system can calculate this number when the player makes his selections. It will be the sum of correct ordered tuples of the player. There is a count for each n-tuple.
  - $\#CTCBP = \#ATS - \#WTCBP$
- **Number of wrong tuples chosen by player (#WTCPB)**: The system can calculate this number when the player makes his selections. It will be the sum of incorrect choices which of the player. There is a count for each n-tuple.
  - $\#WTCPBP = \#ATS - \#CTCBP$
- **Amount of penalty score given to player (AOPS)**: Each wrong choice the player chooses will affect his final score. Here we have two approaches:
  - *First:* for each wrong answer the system does not give any score to the player or:
    - $AOPS = 0$
  - *Second:* for each wrong answer not only they get no score but also we penalize them by a percentage (defined by professor) of positive score for a correct answer. In this case the amount of negative score is calculated by following formula:
    - $AOPS = -[Penalty\ Percentage\#AT] \times \#WTCBP$

- **Amount of award score given to player (AOAS)**: Each correct tuple the player chooses will affect his final score. The score given depends on how long the tuple is and is calculated by following formula:
  - $AOAS = \frac{\sum(\#AR-(n-1))*\#CTCBP}{\sum(\#AR-(n-1))*\#AT}$
- **Total score given to player (TS)**: This is the sum of award and penalty given to the player.
  - $TS = AOAS + AOPS >= 0$
- **Time to Answer (#TA)**: Number of total seconds that a player has to answer a question. If it takes the player less than this amount to finish answering he will be rewarded.
  - $\#TA > 0$
- **Time Taken to Answer (#TTA):** Number of seconds passed from the moment that the player started answering the question.
  - $\#TTA > 0$
- **Time Bonus Points (#TBP):** This an amount of bonus points given to player according to his timings
  - **Maximum time bonus points:** This is the maximum amount of bonus points that a player will get it he performs 100% efficient (according to timing) on the activity. The professor defines this number. To keep the competition balanced the following constraint should be taken into account while defining this number.
    - $Maximum\ time\ bonus\ points <= Maximum\ number\ of\ points$
  - **Actual time bonus points (ATBP):**
    - $ATBP = (\#TA - \#TTA)\#TA \times Max\ number\ of\ bonus\ points$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:** Professor should define a maximum number of points for a question. The maximum point will be given to players who could get the total score of the question.
    - **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
    - **Actual number of points (APs):** Following formula calculates this parameter
      - $APS = TS \times [Max\ number\ of\ points + ATBP] + Min\ number\ of\ points$
  - *Betting chips*

Examples

An example to illustrate how the score is calculated for this activity can be seen bellow. Figure 24 exposes the setup of the activity.

Question: Put the following numbers in order
#AC = 5
#AR = 5
Correct ranking:
1. 1
2. 2
3. 3
4. 4
5. 5
Points = 150
Min points = 9
There is no timer.
No penalty is considered.

Figure 24 – Initial configuration of the activity

Figure 25 describes how the score is to be calculated.

Number of tuples	Score per tuple	Final score
#1tuples = 5 #2tuples = 4 #3tuples = 3 #4tuples = 2	1tuples = $\sum(5-(1-1)) * \#CTCBP$ 2tuples = $\sum(5-(2-1)) * \#CTCBP$ 3tuples = $\sum(5-(3-1)) * \#CTCBP$ 4tuples = $\sum(5-(4-1)) * \#CTCBP$ 5tuples = $\sum(5-(5-1)) * \#CTCBP$	$AOAS = \frac{\Sigma Score\ per\ tuple}{11}$ $TS = AOAS$ $AP = TS * 150 + 9$

Figure 25 – Analysis of elements necessary to find the score

Figure 26 shows how two different answers are scored.

Answer by student a:

1. 1
2. 3
3. 4
4. 2
5. 5

1tuple #CTCBP = 2;
2tuple #CTCBP = 3;
3tuple #CTCBP = 1;
4tuple #CTCBP = 0;
5tuple #CTCBP = 0;

$$AOAS = \left(\frac{2*5+3*4+1*3+0*2+0*1}{5*5+4*4+3*3+2*2+1*1}\right)$$

$$AOAS = \frac{5}{11}$$

$$AP = \frac{5}{11} * 150 + 9 = 77.18 \approx 77$$

Answer by student b:

1. 1
2. 2
3. 3
4. 5
5. 4

1tuple #CTCBP = 3;
2tuple #CTCBP = 3;
3tuple #CTCBP = 2;
4tuple #CTCBP = 1;
5tuple #CTCBP = 0;

$$AOAS = \left(\frac{3*5+3*4+2*3+1*2+0*1}{5*5+4*4+3*3+2*2+1*1}\right)$$

$$AOAS = \frac{7}{11}$$

$$AP = \frac{7}{11} * 150 + 9 = 104.45 \approx 104$$

Figure 26 – Scores for two students with different answers

3.2.3.7 Faulty questions

For this type there is a question defined by the professors. The question has an error in its statement, and a number of possible corrections are presented. Once the student has answered what is wrong, a set of answers to the new question is presented. Finally, when they submit their answers, they will get a certain score for their correct answers.

Elements

Different elements for this type of activity are:

- **Question**: a simple question with an error which is defined by the professor
- **Statement Correction:** a set of options is provided as the correction to the error in the question.
  - **Right correction:** Right corrections are the ones that grant the player a positive score should they be selected. Otherwise he can get negative score.
  - **Wrong correction:** Wrong corrections are the ones that the player should avoid to get a positive answer. Otherwise he can get negative score. (*Optional*)
- **Question Answer**: a set of definitions that are provided as the solution for a defined question. The answers shown depend on the statement correction previously

chosen. The professor defines both questions and answers. They can be correct solutions or wrong choices. It is the responsibility of the student (player) to distinguish between them.
  - **Correct answer**: Correct answers are the ones that if they show up in available choices, the player should select them to get positive score. Otherwise he can get negative score.
  - **Wrong answer**: Wrong answers are the ones that if they show up in available choices, the player should avoid selecting them to get positive answer. Otherwise he can get negative score. (*Optional*)

  When multiple choices are shown to player, the distribution of the answers will be decided randomly by the system.
- **Correct choice**: A correct answer or correction that has been selected by the player, as well as a wrong answer or correction that is left unselected.
- **Incorrect choice**: A wrong answer or correction that has been selected by the player, as well as a wrong answer or correction that is left unselected.
- **Explanation**: For each correct or incorrect answer and correction, the professor might provide an explanation so after performing the activity the player will know why his answer was right or wrong. (*Optional*)
- **Timer**: As the student starts to answer the question a timer will record the time that particular student takes to finish answering. (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Number of wrong corrections (#WC)**: This number does not have any maximum value but it has the following constraint:
  - $\#WC >= 0$
- **Number of right corrections (#RC)**: This number does not have any maximum value but we should consider a minimum for that using the following formula:
  - $\#RC > 0$
- **Number of corrections in total (#CT)**: This parameter is defined by sum of all right and wrong corrections defined for the question.
  - $\#CT = \#WC + \#RC$
- **Number of right corrections to show in all choices (#RCTS)**: This number has a minimum and maximum value defined by the professor while defining the question. When the system wants to show a question to the player, it will create a random number in this defined range which will be #RCTS.
  - $\#RCTS > 0$
  - $Min(\#RCTS) > 1$ defined by professor
  - $Max(\#RCTS)$ is defined by professor and it is less than or equal to number of available choices.

- $Min(\#RCTS) <= \#RCTS <= Max(\#RCTS)$

- **Number of wrong corrections to show in all choices (#WCTS)**: This number has a minimum and maximum value defined by the professor while defining the question. When the system wants to show a question to the player, it will use the randomly generated #RCTS to calculate this number using the following formula:
  - $\#WCTS = Number\ of\ available\ choices\ to\ select - \#RCTS$

  It should also follow these constraints:
  - $Min(\#WCTS) > 0$ defined by professor
  - $Max(\#WCTS) < \#PC$ defined by professor
  - $Min(\#WCTS) <= \#WCTS <= Max(\#WCTS)$
- **Number of possible corrections to select (#PC)**: For each question there are a number of corrections (choices) shown. This parameter defines how many these choices will be. The professor should define it for each question while creating it.
  - $\#PC = \#WCTS + \#RCTS$
  - $\#PC >= 4$
- **Right correction by player (RCBP):** The player can choose only one correction out of #PC.
  - $RCBP = 1$ if a right correction is chosen
- **Wrong correction by player (WCBP):** The player can choose only one correction out of #PC.
  - $WCBP = 1$ if a wrong correction is chosen
- **Number of wrong answers (#WA)**: This number does not have any maximum value but it has the following constraint:
  - $\#WA >= 0$
- **Number of correct answers (#CA)**: This number does not have any maximum value but we should consider a minimum for that using the following formula:
  - $\#CA > 0$
- **Number of answers in total (#AT)**: This parameter is defined by sum of all correct and wrong answers defined for the question.
  - $\#AT = \#WA + \#CA$
- **Number of correct answers to show in all choices (#CATS)**: This number has a minimum and maximum value defined by the professor while defining the question. When the system wants to show a question to the player, it will create a random number in this defined range which will be #CATS.
  - $\#CATS > 0$
  - $Min(\#CATS) > 1$ defined by professor
  - $Max(\#CATS)$ is defined by professor and it is less than or equal to number of available choices.
  - $Min(\#CATS) <= \#CATS <= Max(\#CATS)$
- **Number of wrong answers to show in all choices (#WATS)**: This number has a minimum and maximum value defined by the professor while defining the question.

When the system wants to show a question to the player, it will use the randomly generated #CATS to calculate this number using the following formula:

- $\#WATS = Number\ of\ available\ choices\ to\ select - \#CATS$

It should also follow these constraints:

- $Min(\#WATS) > 0$ defined by professor
- $Max(\#WATS) < \#AC$ defined by professor
- $Min(\#WATS) <= \#WATS <= Max(\#WATS)$

- **Number of available choices to select (#AC)**: For each question there are a number of answers (choices) shown. This parameter defines how many these choices will be. The professor should define it for each question while creating it.
  - $\#AC = \#WATS + \#CATS$
  - $\#AC >= 4$
- **Number of correct choices by player (#CCBP)**: The system can calculate this number when the player makes his selections. It will be the sum of correct choices of the player.
  - $\#CCBP = \#AC - \#WCBP$
- **Number of wrong choices by player (#WCPB)**: The system can calculate this number when the player makes his selections. It will be the sum of incorrect choices which of the player.
  - $\#WCBP = \#AC - \#CCBP$
- **Amount of penalty score given to player (AOPS)**: Each wrong choice the player chooses will affect his final score. Here we have two approaches:
  - *First:* for each wrong answer the system does not give any score to the player, or
    - $AOPS = 0$
  - *Second:* for each wrong answer not only they get no score but also we penalize them by a percentage (defined by professor) of positive score for a correct answer. In this case the amount of negative score is calculated by following formula:
    - $AOPS = -[Penalty\ Percentage\#AC]\times\#WCBP - \#PC\times WCBP$
- **Amount of award score given to player (AOAS)**: Each correct choice the player chooses will affect his final score. This amount is calculated by following formula:
    - $AOAS = \#CCBP\#AC + \#PC\times RCBP$
- **Total score given to player (TS)**: This is the sum of award and penalty given to the player.
  - $TS = AOAS + AOPS >= 0$
- **Time to Answer (#TA)**: Number of total seconds that a player has to answer a question. If it takes the player less than this amount to finish answering he will be rewarded.
  - $\#TA > 0$
- **Time Taken to Answer (#TTA):** Number of seconds passed from the moment that the player started answering the question.

- $\#TTA > 0$
- **Time Bonus Points (#TBP):** This an amount of bonus points given to player according to his timings
  - **Maximum time bonus points:** This is the maximum amount of bonus points that a player will get it he performs 100% efficient (according to timing) on the activity. The professor defines this number. To keep the competition balanced the following constraint should be taken into account while defining this number.
    - $Maximum\ time\ bonus\ points\ <=\ Maximum\ number\ of\ points$
  - **Actual time bonus points (ATBP):**
    - $ATBP = (\#TA - \#TTA)\#TA \times Max\ number\ of\ bonus\ points$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of points:** Professor should define a maximum number of points for a question. The maximum point will be given to players who could get the total score of the question.
    - **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
    - **Actual number of points (APs):** Following formula calculates this parameter
      - $APS = TS \times [Max\ number\ of\ points + ATBP] + Min\ number\ of\ points$
  - *Betting chips*

3.2.3.8 Cooperative questions

This activity can follow the format of previously described questions, either questions with single or multiple answers, short answer questions, or long answer questions. For this type of activity, a defined number of students will be given part of a question that they must solve. Every player receives the solution reached by the others, and must be able to use this to find the more complex solution to the whole problem.

Elements

Different elements for this type of activity are:

- **Partial solution:** answer to the incomplete question that is given to each player.
- **Final solution:** answer reached by each participating student by using the partial solutions of the others.
- **Number of participating students:** number of students that will cooperate to find the final solution.

- **Participating students:** players that will take part in the activity together. The professor can decide to let the system automatically assign this in a random manner, or to group participants by their team. If a team were not to have enough members to participate, they could be grouped with other team or be discarded as participants depending on the professor's decision. Otherwise, if a team has more members than the number of participating students set for the activity, the participating members are randomly selected from within the group.
- **Timer**: A timer will record the time a particular student takes to finish answering. The timer will begin running when the student enters the first question, and will continue to run when he is logged into the system until he sends the final solution. (*Optional*)

Parameters, Relations and Constraints

Different parameters for this activity are as follows:

- **Time to Answer (#TA)**: Number of total seconds that a player has to answer a question. If it takes the player less than this amount to finish answering he will be rewarded.
  - $\#TA > 0$
- **Time Taken to Answer (#TTA):** Number of seconds passed from the moment that the player started answering the question.
  - $\#TTA > 0$
- **Time Bonus Points (#TBP):** This an amount of bonus points given to player according to his timings
  - **Maximum time bonus points:** This is the maximum amount of bonus points that a player will get it he performs 100% efficient (according to timing) on the activity. The professor defines this number. To keep the competition balanced the following constraint should be taken into account while defining this number.
    - $Maximum\ time\ bonus\ points <= Maximum\ number\ of\ points$
  - **Actual time bonus points (ATBP):**
    - $ATBP = (\#TA - \#TTA)\#TA \times Max\ number\ of\ bonus\ points$
- **Type of components which will be used as reward for the players**:
  - *Points*:
    - **Maximum number of partial points:** Professor should define a maximum number of points for the partial question. The maximum point will be given to players who could get the total score of the question.
    - **Maximum number of final points:** Professor should define a maximum number of points for the final question. The maximum point will be given to players who could get the total score of the question.

- **Minimum number of points:** Professor should define a minimum amount of points. This value will be used for people who get zero score for the question.
- **Partial points (PPs):** Following formula calculates this parameter
  - $PP = TS \times Max\ number\ of\ partial\ points$
  - TS is calculated according to the type of activity being followed
- **Final points (FPs):** Following formula calculates this parameter
  - $FP = TS \times [Max\ number\ of\ final\ points + ATBP]$
  - TS is calculated according to the type of activity being followed
- **Actual number of points (APs):** Following formula calculates this parameter
  - $AP = PP + FP + Min\ number\ of\ points$

Points that are not mentioned are the same as section 3.2.3.1 , 3.2.3.2 or 3.2.3.4 according to the activity format followed.

3.3 Activity Design

The activity design was carried out using UWE (UML-based Web Engineering) [7]. Five different models were used, each formed by one or more diagrams. Each model considers each user type individually. Administrators and professors are similar enough in their design to be considered in the same model; players, however, are contemplated separately. Compared to players, professors have many more and complex functions. In some cases, the player models were simple enough to be explained textually and not require a diagram to be understood, specially given they can be seen as a simplification of the equivalent professor diagram.

3.3.1 Requirements Model

3.3.1.1 Use Case

Figure 27 shows the main use case of the application and its relationships with a professor user. The actions allowed for a professor include: filtering, viewing, adding, editing, deleting, closing, opening, and reviewing.

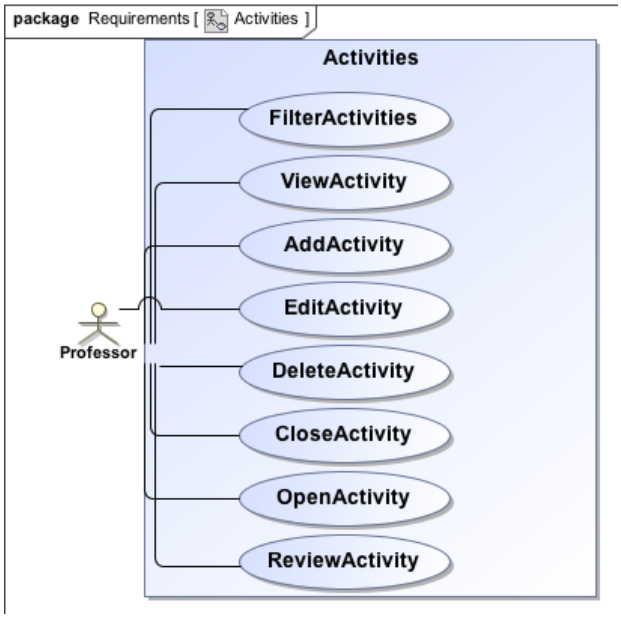

Figure 27 – Professor activity use case

If we consider this same use case having a player as a user, the possible actions are filtering, which is not exactly the same action though it bears the same name, and doing, which allows a student to take a specific activity. Figure 28 illustrates the player activity use case.

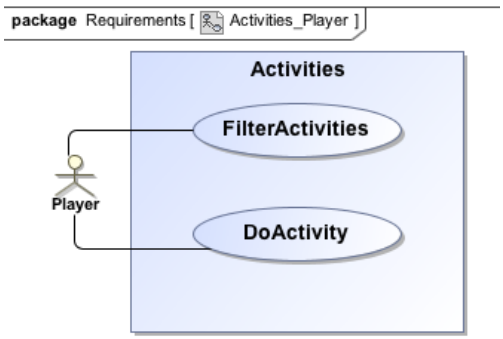

Figure 28 – Player activity use case

Each action for every user is described in detail in the following section.

3.3.1.2 Use Case Detail

The actions here described were explained in section 3.1.3.2 .

3.3.1.2.1 Professor

3.3.1.2.1.1 Filter

The filter action displays a form with various fields to look for. After this, the resulting filtered activities are shown in a list. From here, many navigation actions are available. Figure 29 presents these actions in further detail.

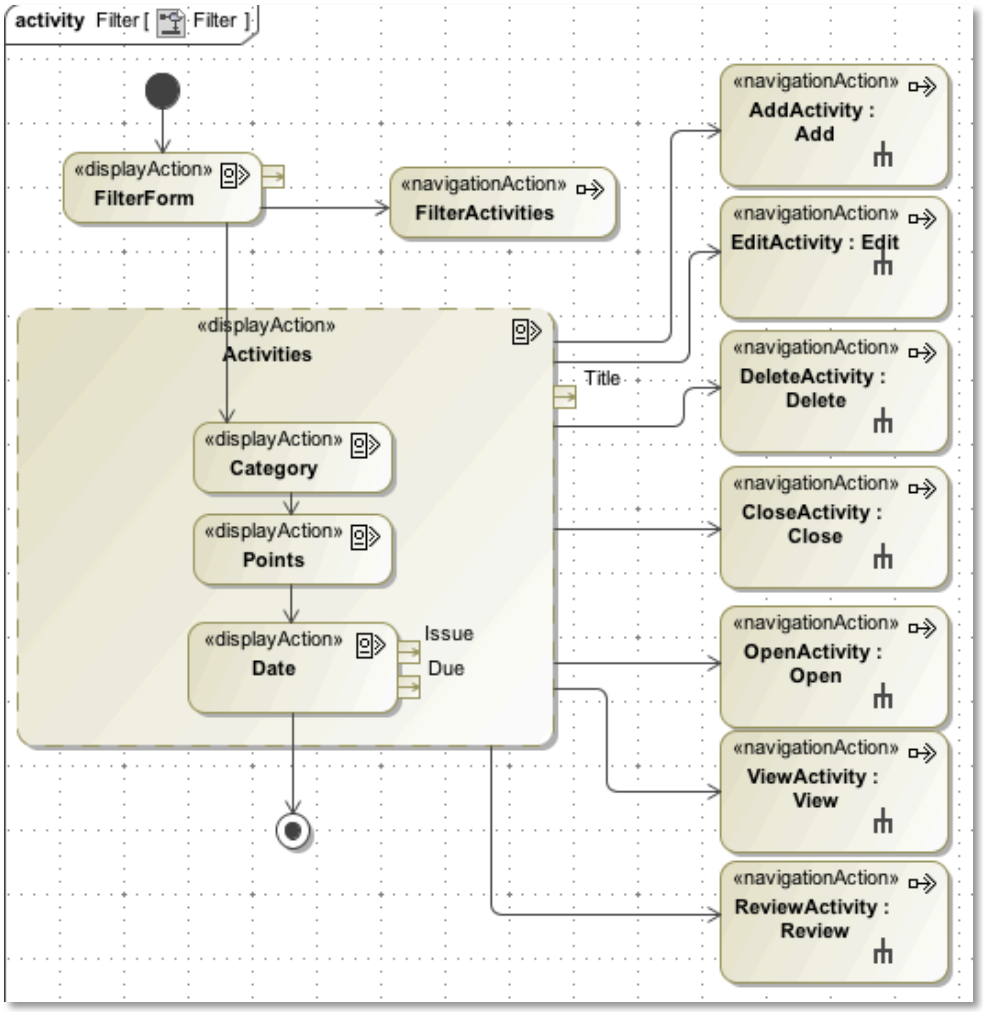

Figure 29 – Filter action

3.3.1.2.1.2 View

Figure 30 shows that viewing displays two different sets of information, the activity information on one side, and some summarized player data on the other.

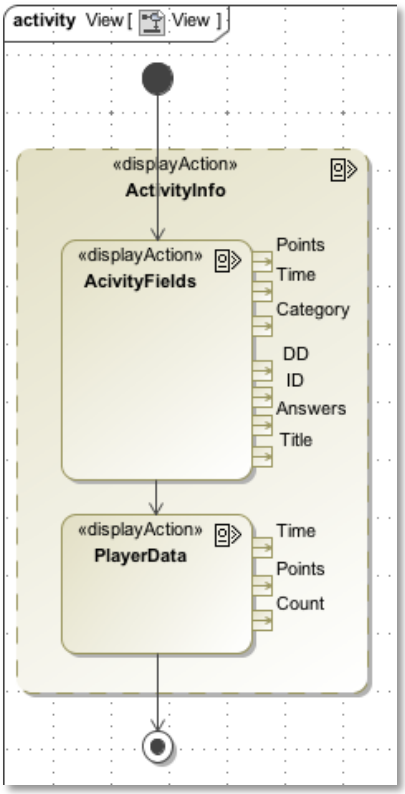

Figure 30 – View action

3.3.1.2.1.3 Add

Adding an activity takes place over the course of various steps. This action first branches out once the user decides to base the new activity off an existing one or to create a new one from scratch. If the former is selected, a list of activities is displayed from which the professor can either View or Copy an activity. If the latter is chosen, the type of activity to be created must be input following by a form with all necessary activity fields. Finally the new activity is saved by the system. Consulting the diagram in Figure 31 this action may be followed more easily.

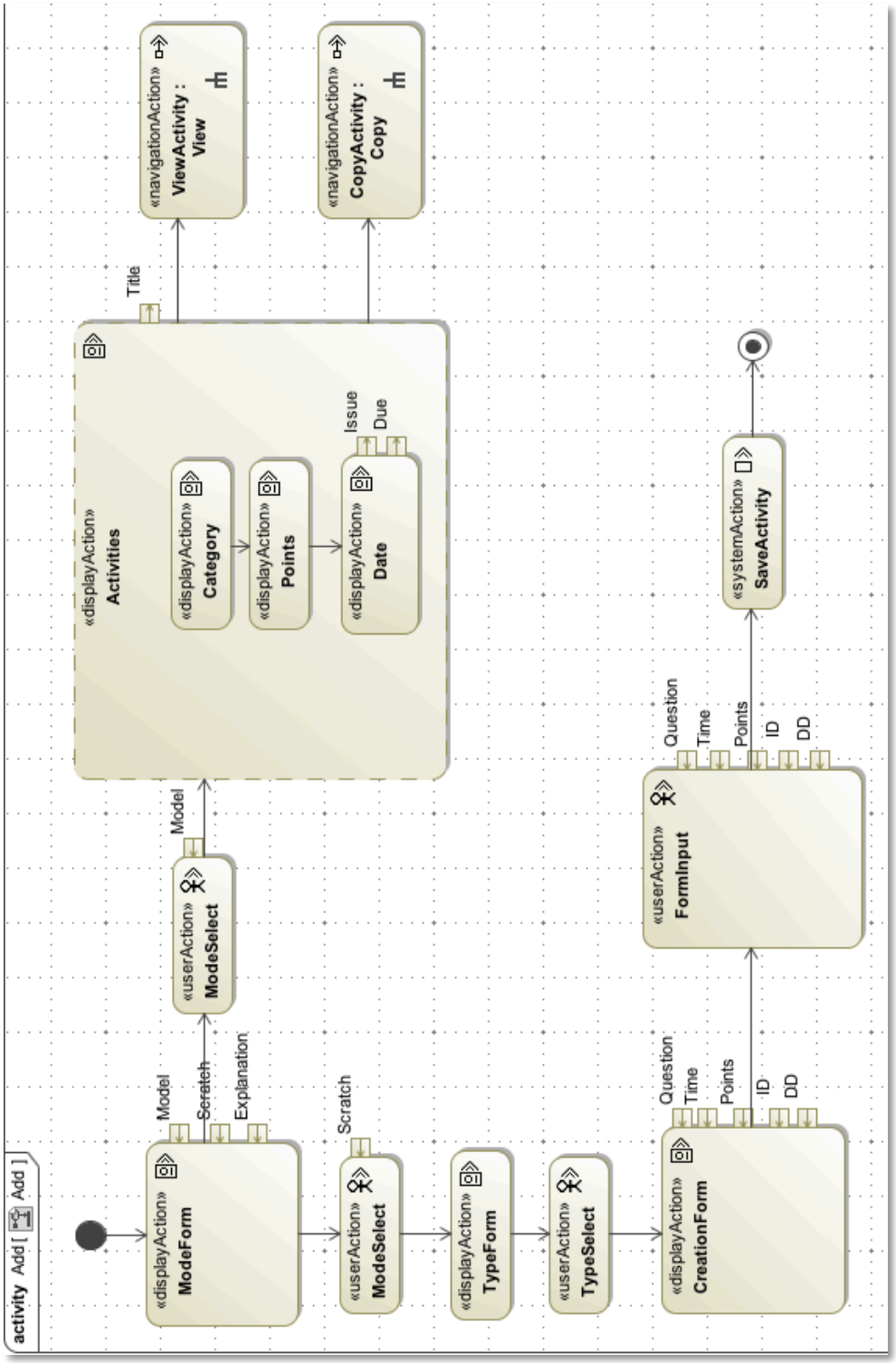

Figure 31 – Add action

3.3.1.2.1.4 Copy

The copy action is presented in three parts. The first is a display of the fields of the selected activity, the next allows for user edit in such fields, and the last is a system action that saves the changes that have been made. Figure 32 shows these three steps and the fields that an activity might have. The Copy action is accessed through the Add action, by choosing to create an activity from a model.

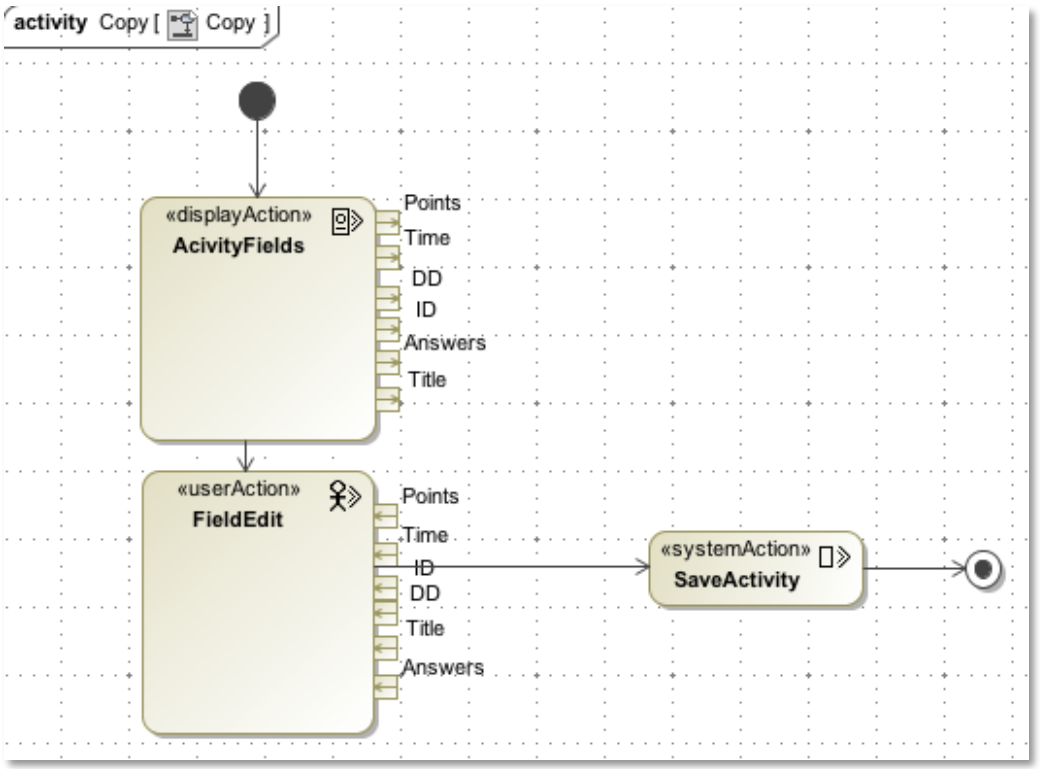

Figure 32 – Copy action

3.3.1.2.1.5 Edit

The action of editing starts with an automatic system action that deactivates the activity so that no incongruences happen while modifying it. After that, the activity fields are displayed so that the professor can then edit whichever he wants. Finally, the system saves these changes. Figure 33 shows a diagram of this action.

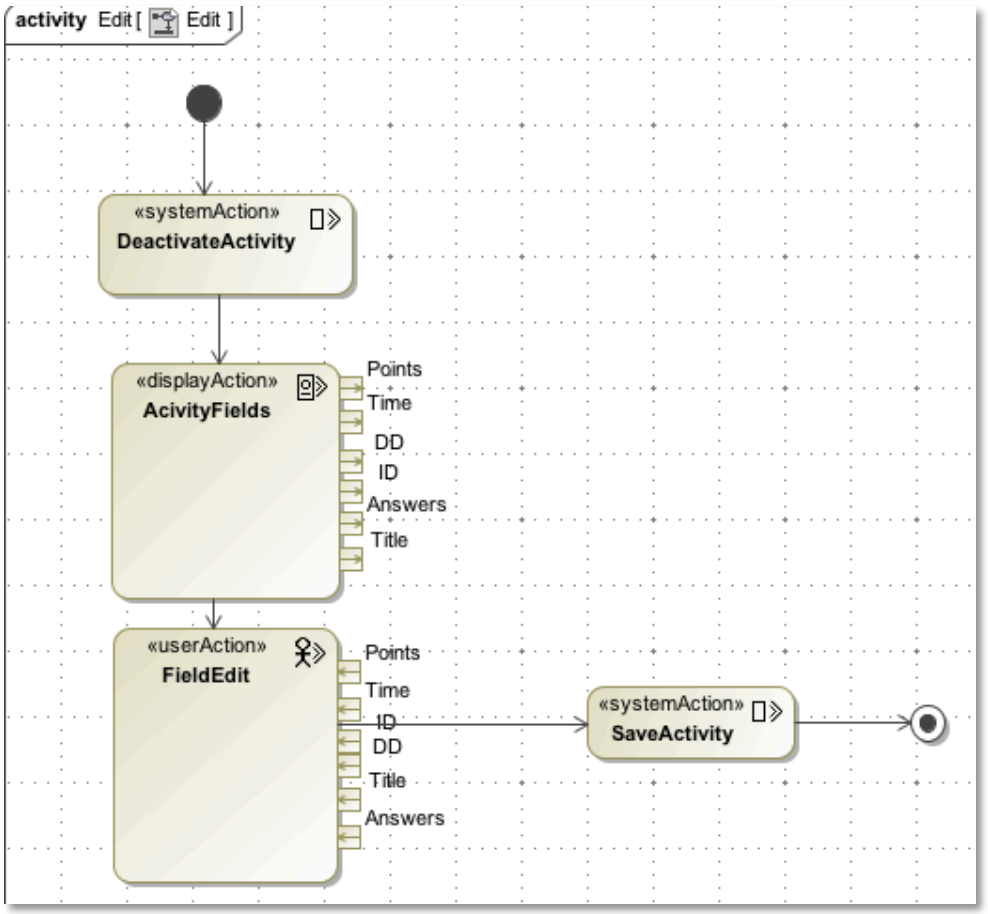

Figure 33 – Edit action

3.3.1.2.1.6 Delete

This action shows a confirmation message for the deletion and, if accepted, deletes the activity via system action. Figure 34 illustrates this.

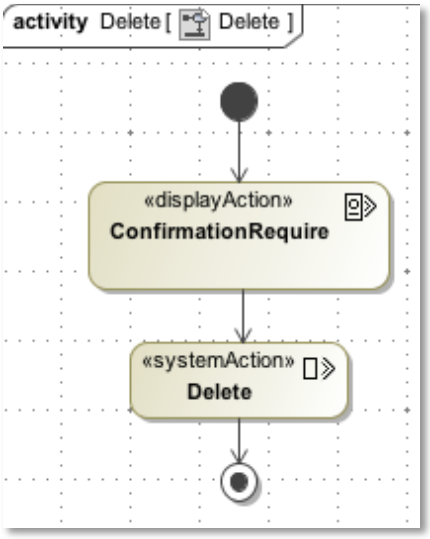

Figure 34 – Delete action

3.3.1.2.1.7 Close

Close is a simple and automatic action, the system takes care of changing the due date of the relevant activity, as seen in Figure 35, and is then ended.

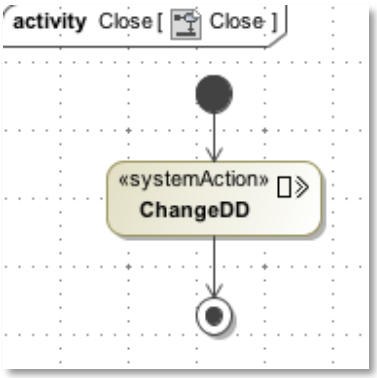

Figure 35 – Close action

3.3.1.2.1.8 Open

The opening of an activity begins by displaying the issue and due dates, which can then be modified by the user. A warning might appear if the issue date is greater than the current date. Either way, the activity modification is saved by the system as can be seen in Figure 36.

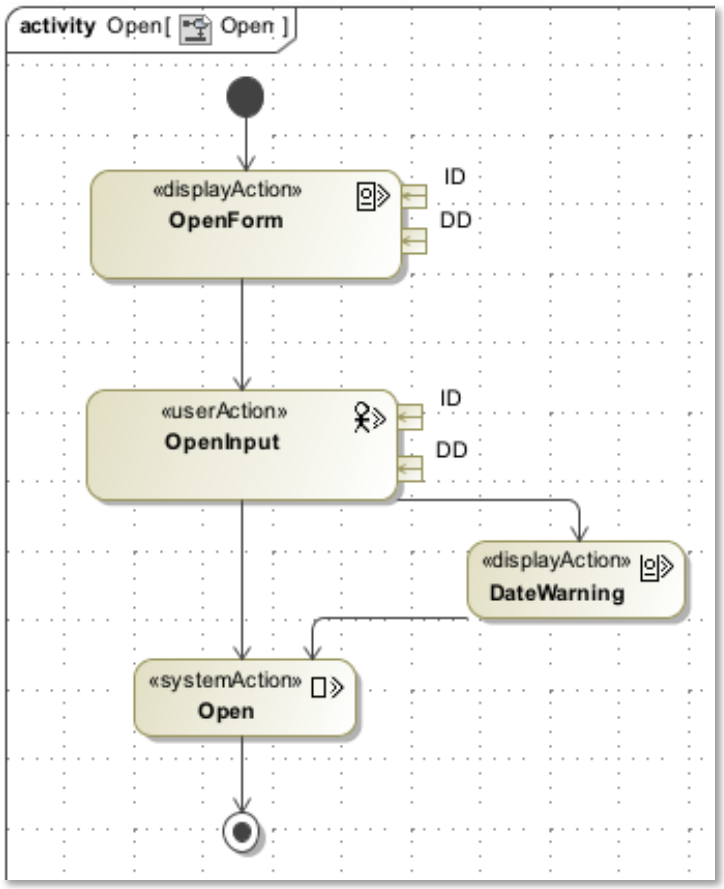

Figure 36 – Open action

3.3.1.2.1.9 Review

When reviewing an activity, a list with all players and some summarized data is presented. A player can then be selected for further reviewing, displaying more data, such as the given answers to the specific activity. From here, the professor can edit the score given to the player. Once this is done, the system saves any changes. Figure 37 illustrates the process.

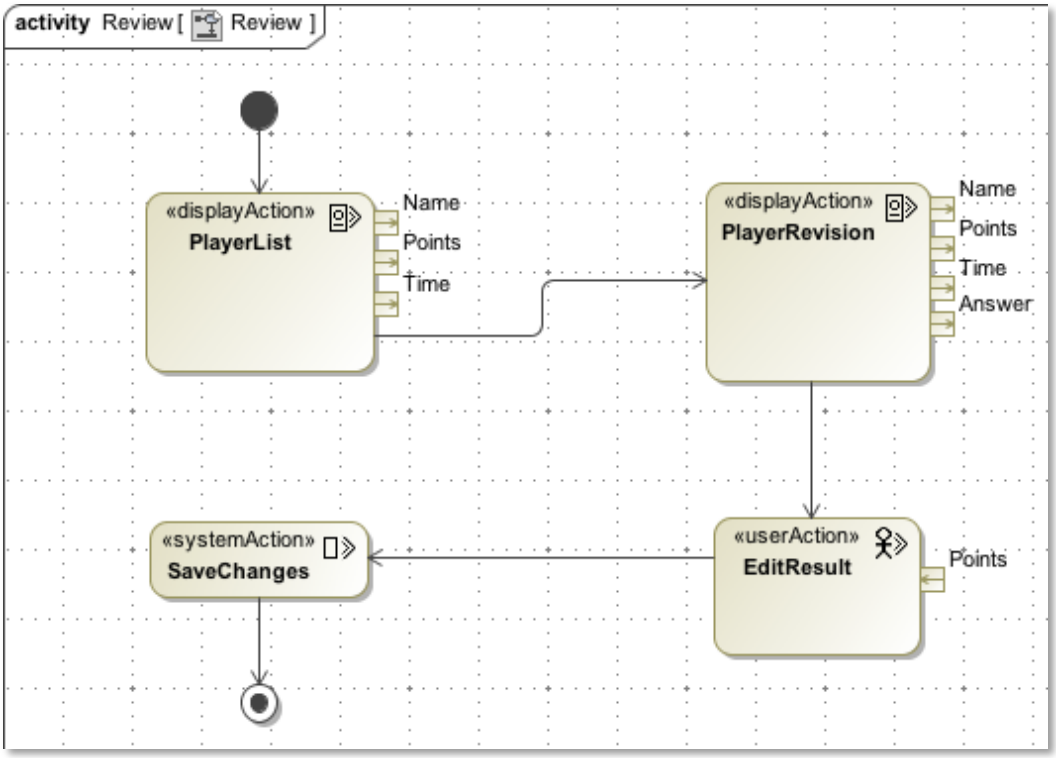

Figure 37 – Review action

3.3.1.2.2 Player

3.3.1.2.2.1 Filter

The player filter action works similarly to the professor one. The main difference is there are fewer fields to filter with, and just one navigation action to choose from. These differences can be seen in Figure 38.

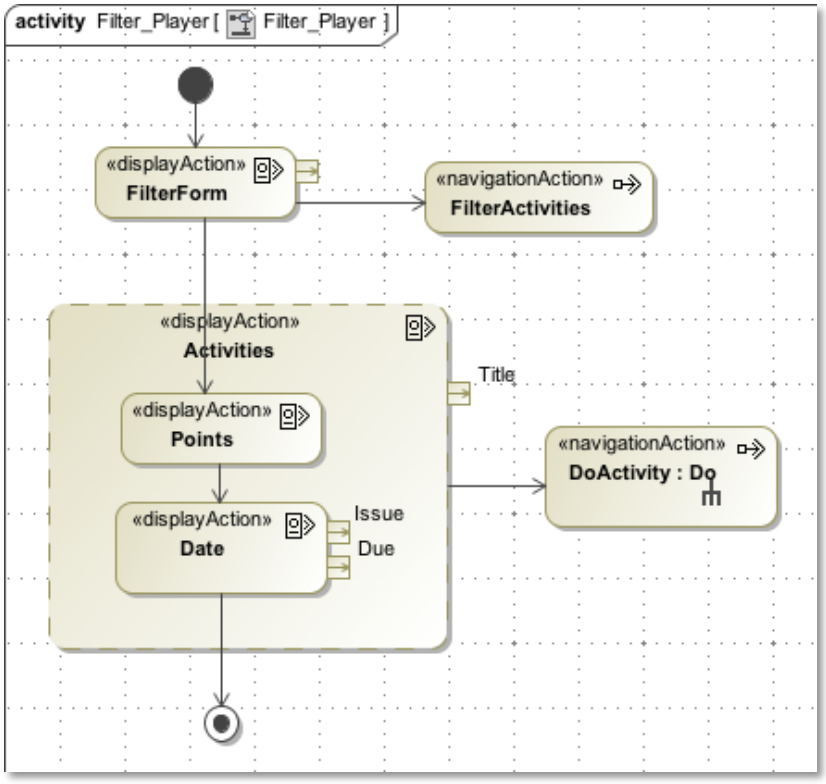

Figure 38 – Player filter action

3.3.1.2.2.2 Do

This is the main player action. The activity is shown with all relevant fields such as question and possible answers. The player then takes action to resolve the problem presented. The system then corrects the solution given, displays the feedback, and saves the answer. Figure 39 properly illustrates this action.

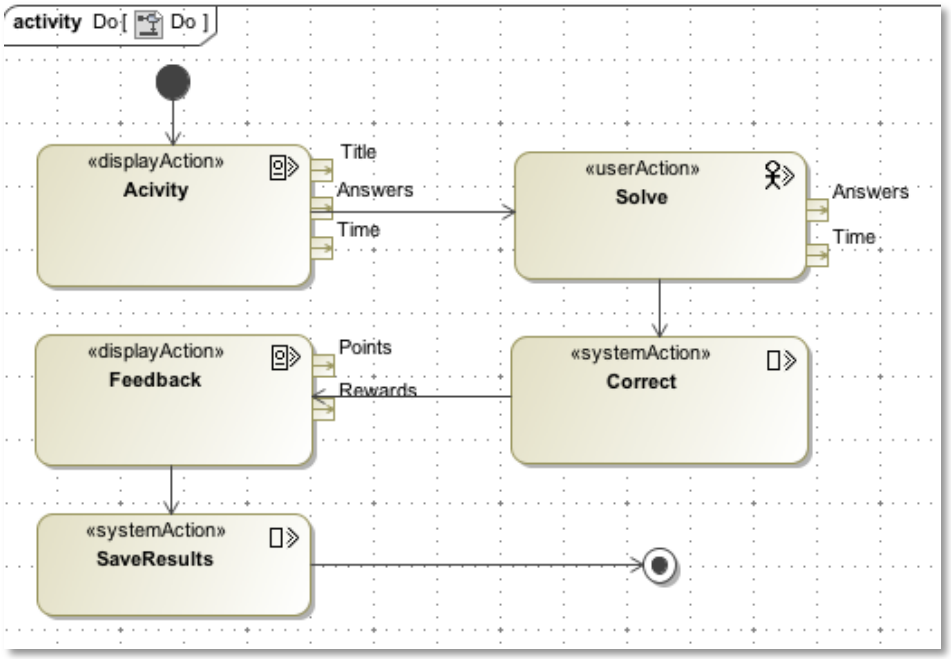

Figure 39 – Do action

It must be noted that this action represents a generalization of how activities are. There are activities that do not exactly follow this diagram, requiring manual correction. Thus in this case, after the player solves the activity, the system saves the results and a new and different system action takes care of the correction and eventual feedback.

3.3.2 Content Model

In Figure 40 we can see the classes and their respective attributes in detail.

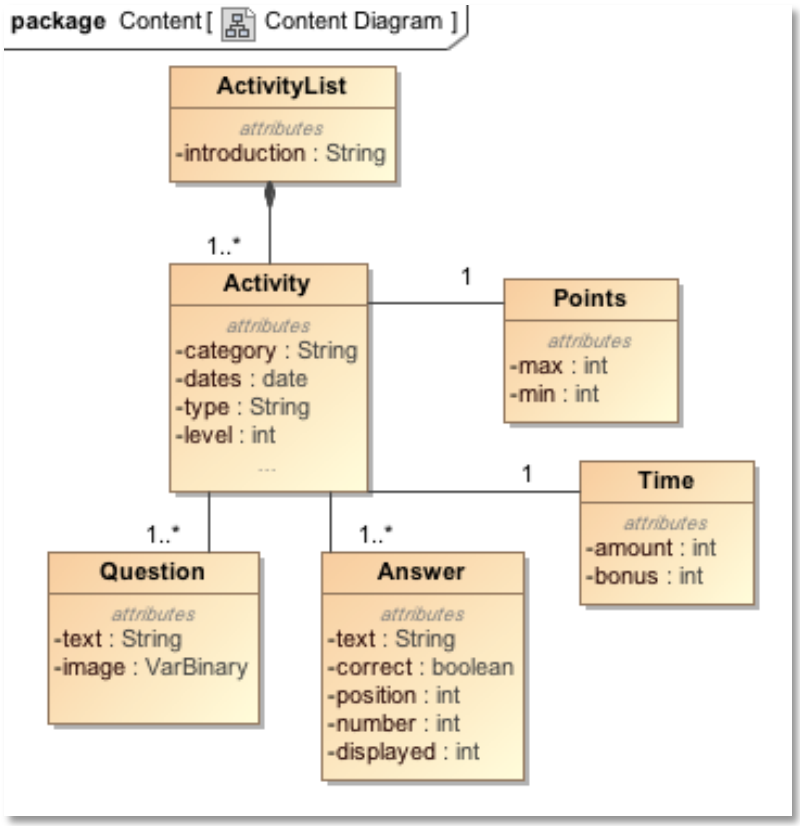

Figure 40 – Content Model

The ActivityList class is found at the top, and is composed by one or many Activity classes. Each Activity then has a Point and Time class, and one or many Questions and Answers.

3.3.3 Navigation Model

Navigation is different for each user. However, both navigations begin from an Initial page, which then leads to the Home page via Login. For a user to Login he must have previously gone through the Register. Once logged in, a Logout takes the user back to the Initial page. The Home page contains a Menu from which the ActivityList can be accessed. The ActivityList can have a Filter run through it that takes the user back to a filtered ActivityList. The professor navigation model then offers separate process links to many process classes. The whole navigation diagram can be seen in Figure 41.

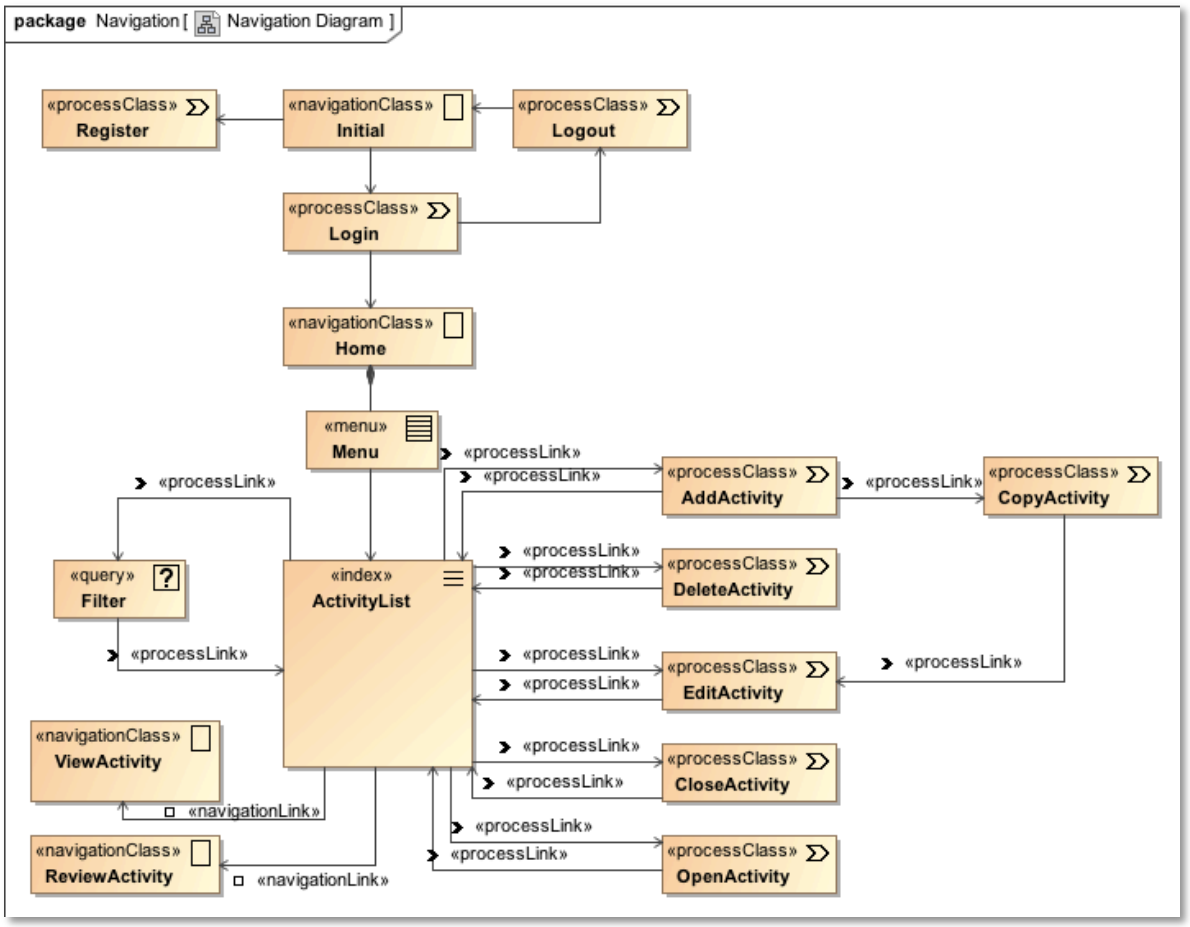

Figure 41 – Professor navigation model

The player navigation model, in turn, offers just one other navigation class, as can be seen in Figure 42.

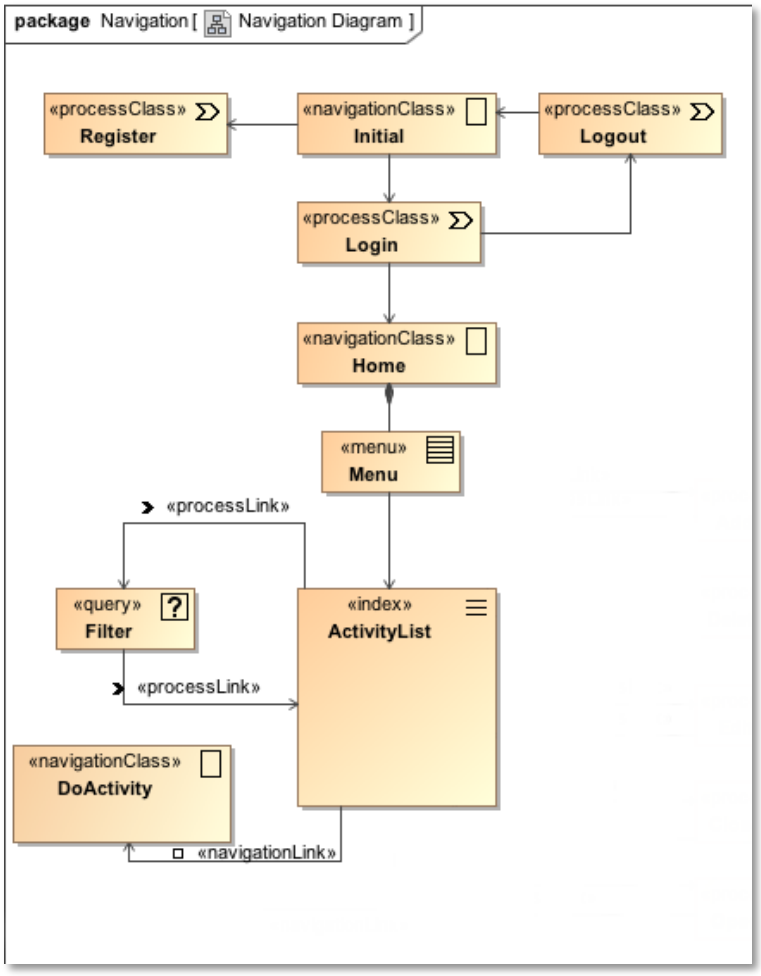

Figure 42 – Player navigation model

3.3.4 Presentation Model

The presentation model gives an idea of which part of the webpage navigation and process classes belong to. It is important to remember that this model abstracts from specific aspects of the interface.

3.3.4.1.1 Professor

The professor presentation model is exhibited starting from an Activity home page. This page presents a header from which other parts of the platform can be accessed, a form for filtering activities, a button to add activities and an alternating view below. This alternating view shows ActivityList, a list of the existing activities, as default. The respective elements of each part can be seen in the diagram detailed in Figure 43.

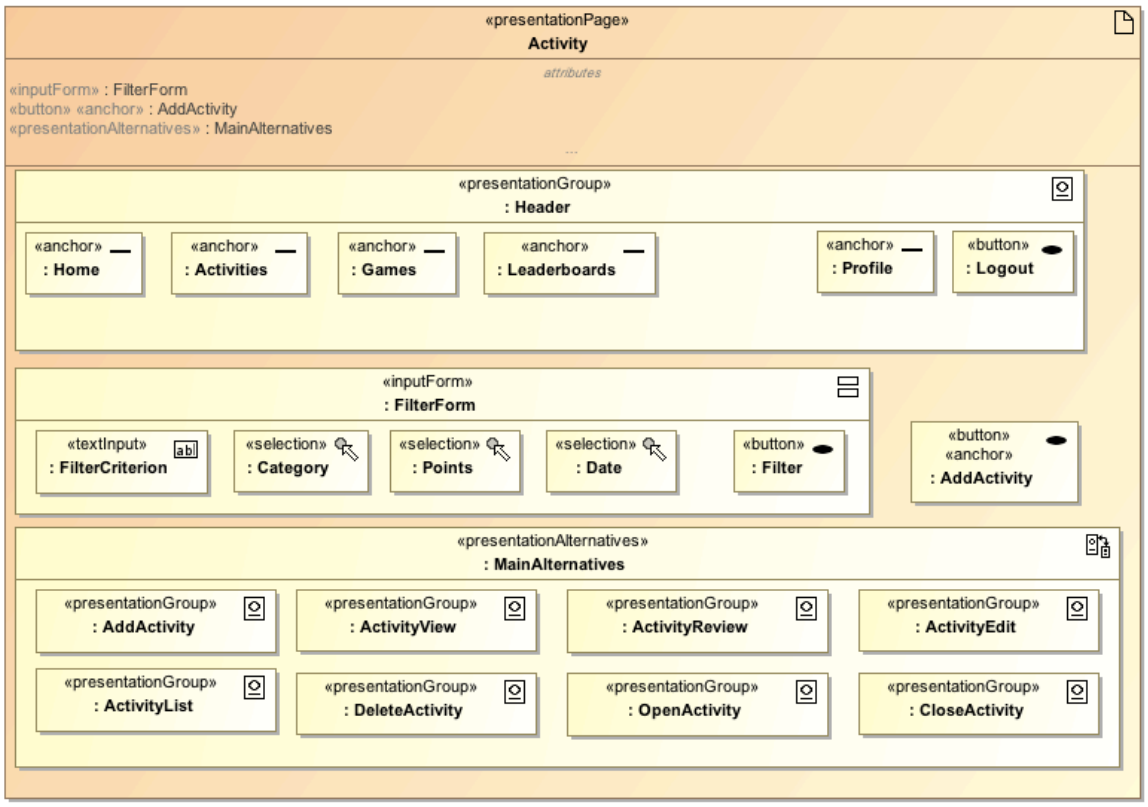

Figure 43 – Professor Activity Home

The default activity list mentioned, displays activities and some of their key fields under labeled columns such as category, points or dates. Beside each activity, buttons to view, review and edit can be found. Then, depending on the specific activity's category, an alternating view enabling open, close and delete buttons is shown. The ActivityList view is expanded in Figure 44. Note that most buttons also double as anchors.

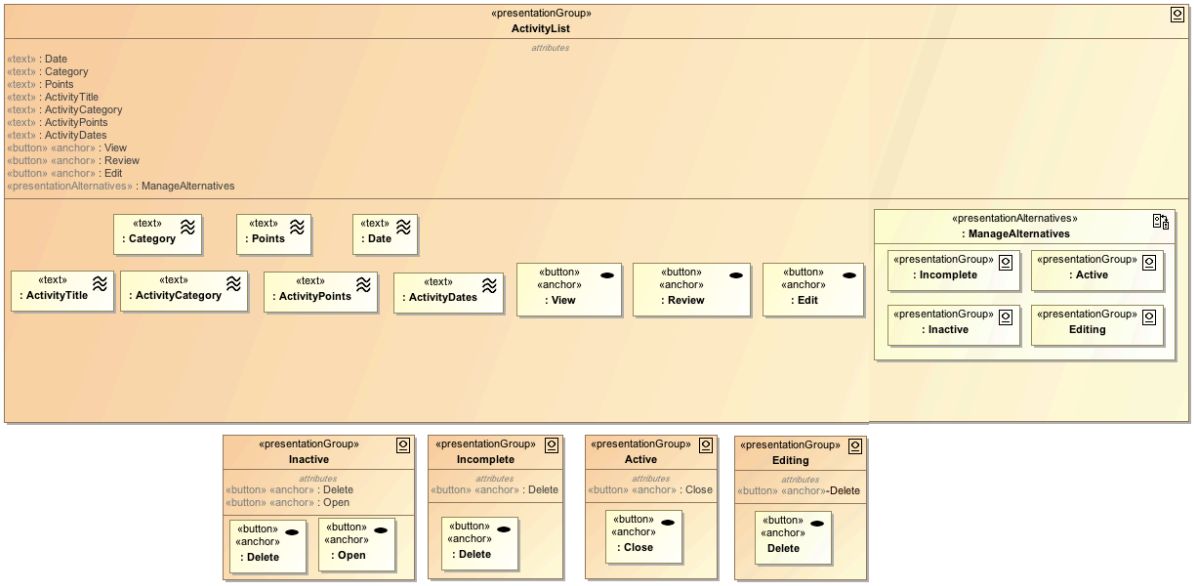

Figure 44 – Professor Activity List

The message and view details of open, close and delete can be seen in Figure 45. CloseActivity simply relays a message, while DeleteActivity also offers to either agree or disagree with its message. Then OpenActivity is the more complex as it allows for text input and then may relay an additional message in case of error or warning.

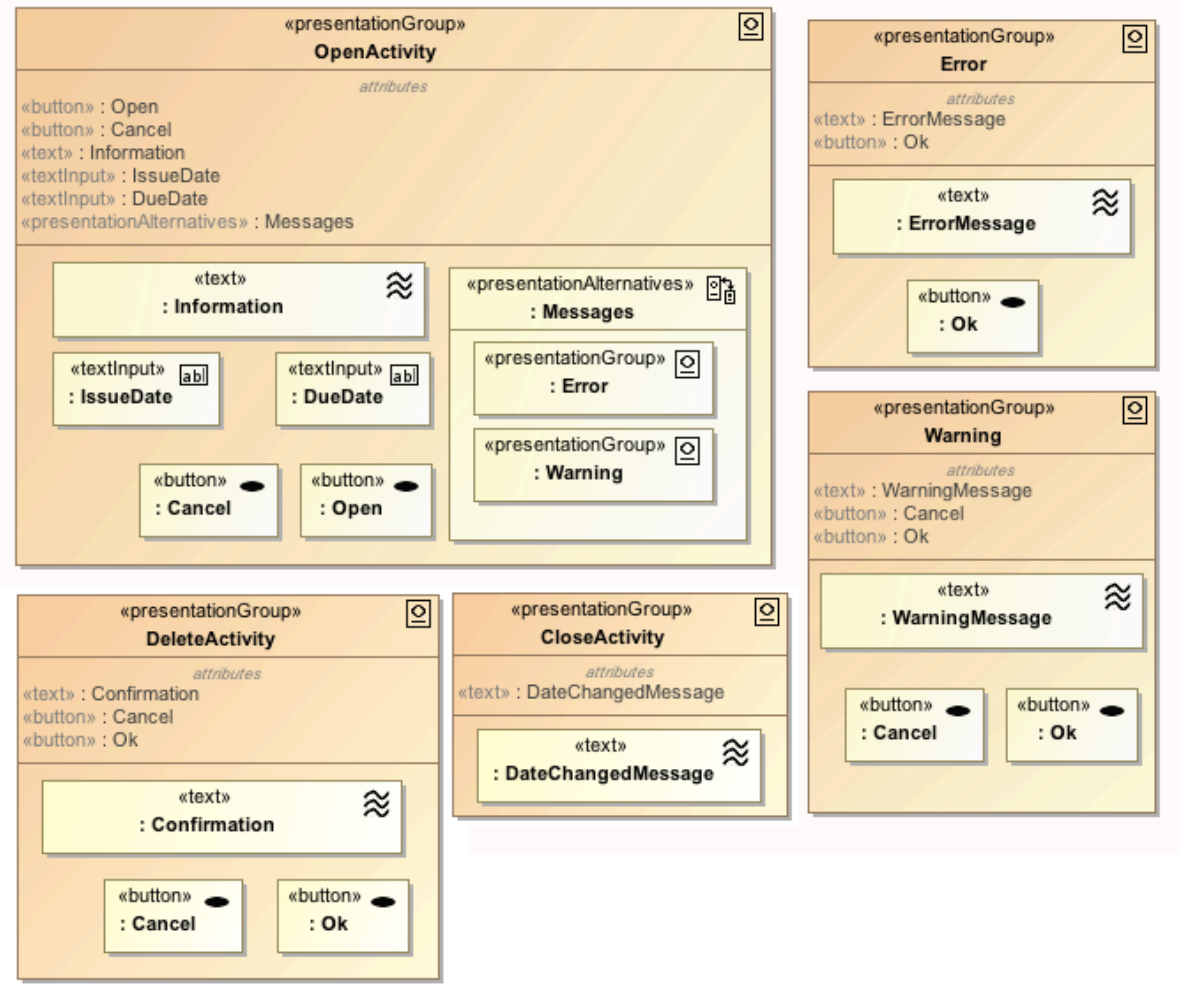

Figure 45 – Professor Delete, Open, and Close

ActivityView simply displays in text the fields relating to a specific activity. An example of such fields can be studied in Figure 46.

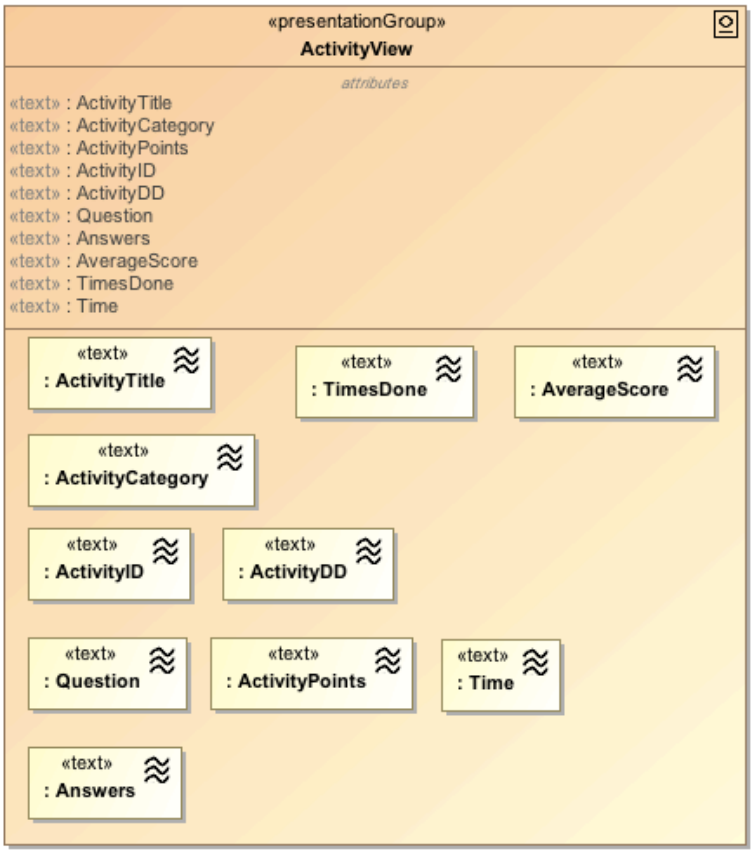

Figure 46 – Professor Activity View

ActivityReview shows text regarding general activity information and then offers an alternative view of a general revision of all players, which is the default, or a revision for an individual player. Figure 47 indicates the aforementioned view with more text fields in the alternating views. Let it be noted that in the SpecificPlayer view the score can be either displayed text or input text depending on whether that activity allows for the overriding of scores.

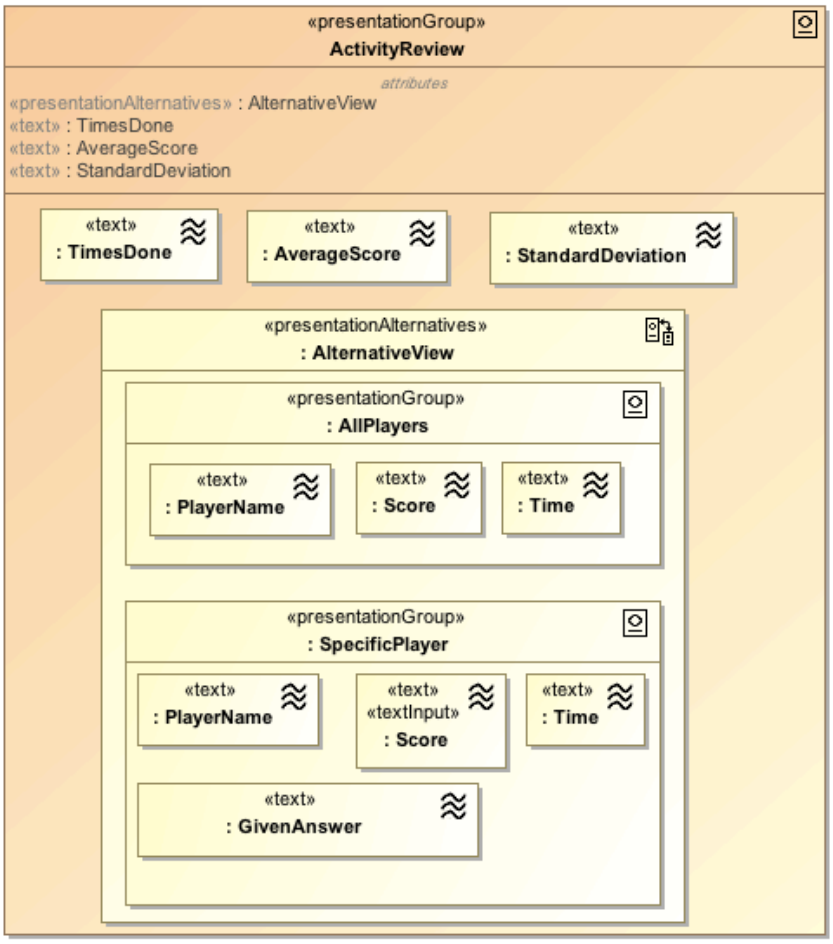

Figure 47 – Professor Activity Review

As can be seen in Figure 48, ActivityEdit much resembles ActivityView. The main difference is that in editing text input is expected where simple text display was found before.

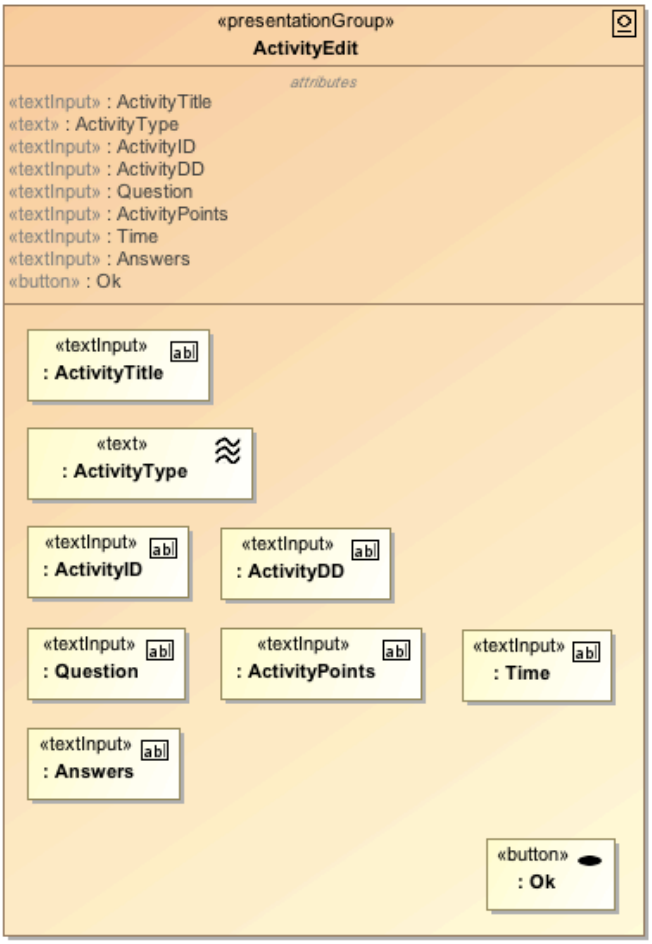

Figure 48 – Professor Activity Edit

Adding an activity is a more complex process and thus is presented in progressive views. The default alternative view is that of CreationHome, which allows for the professor to choose to create the activity from an existing one or from scratch. In the Model choice, the view presents an activity filter much like the one found in the Home screen and a list of activities like that in ActivityList, with the difference of having a Copy button. These first views of AddActivity can be seen in Figure 49.

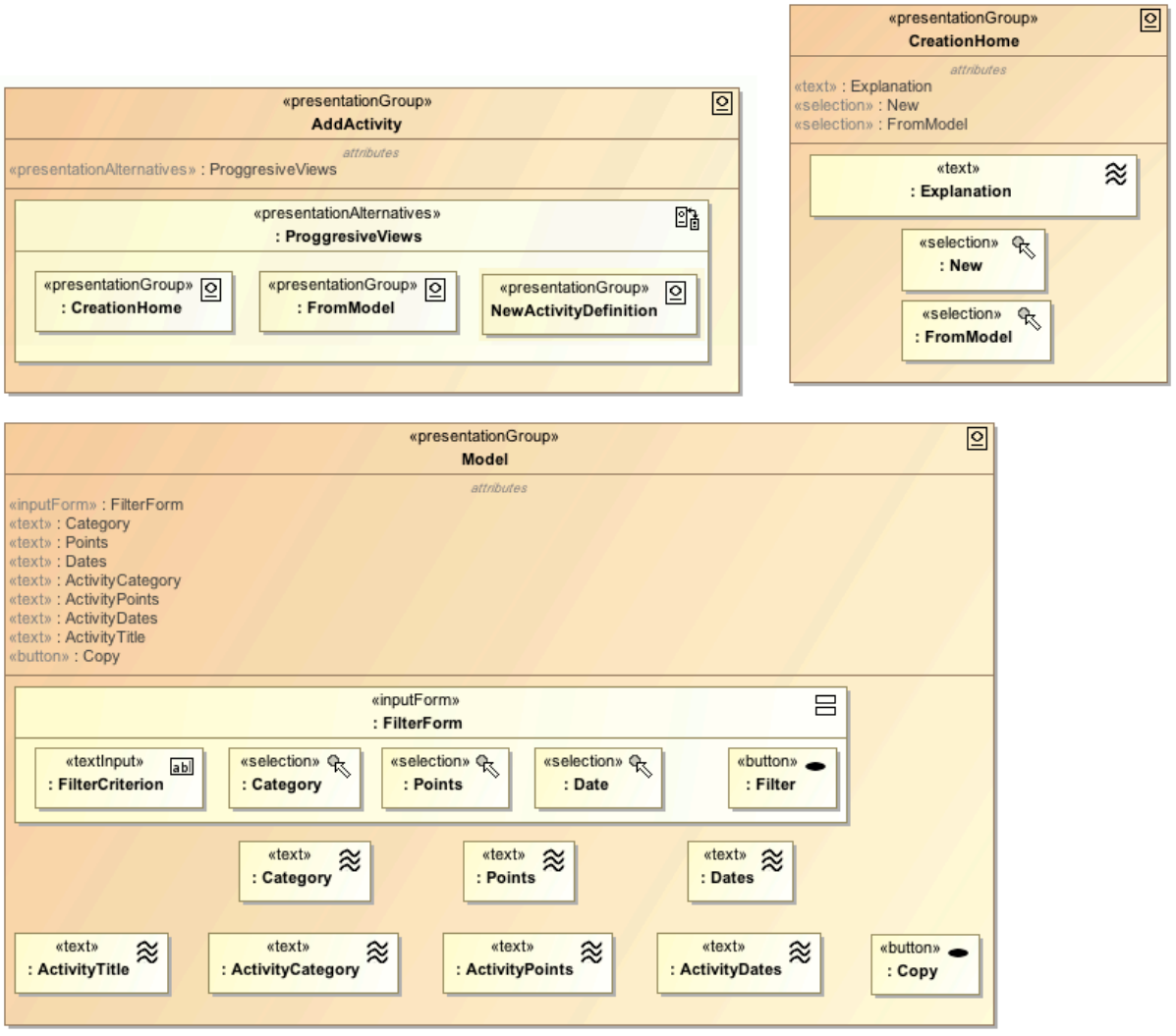

Figure 49 – Professor Adding activity home and from model

Figure 50 displays the rest of the activity adding views, which branch from choosing to create a new activity. TypeDefinition simply presents an explanatory text and an activity type selection. In BasicDefinition an extensive explanation of the fields of that activity type is presented, together with a button that will take the professor to the edit view for that new activity.

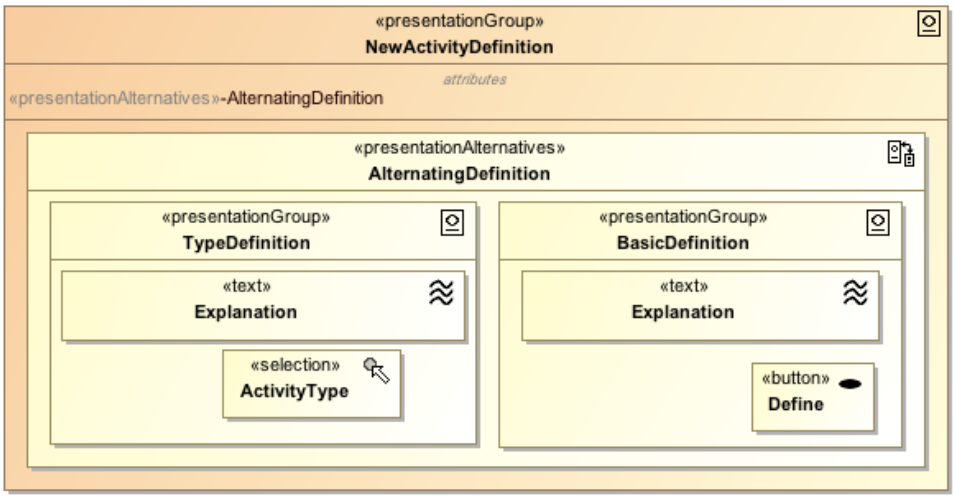

Figure 50 – Professor Adding a new activity

3.3.4.1.2 Player

The player activity home page is very like the professor activity home page. The differences, as observed in Figure 51, are fewer filtering options and just two main alternatives, ActivityList still being the default.

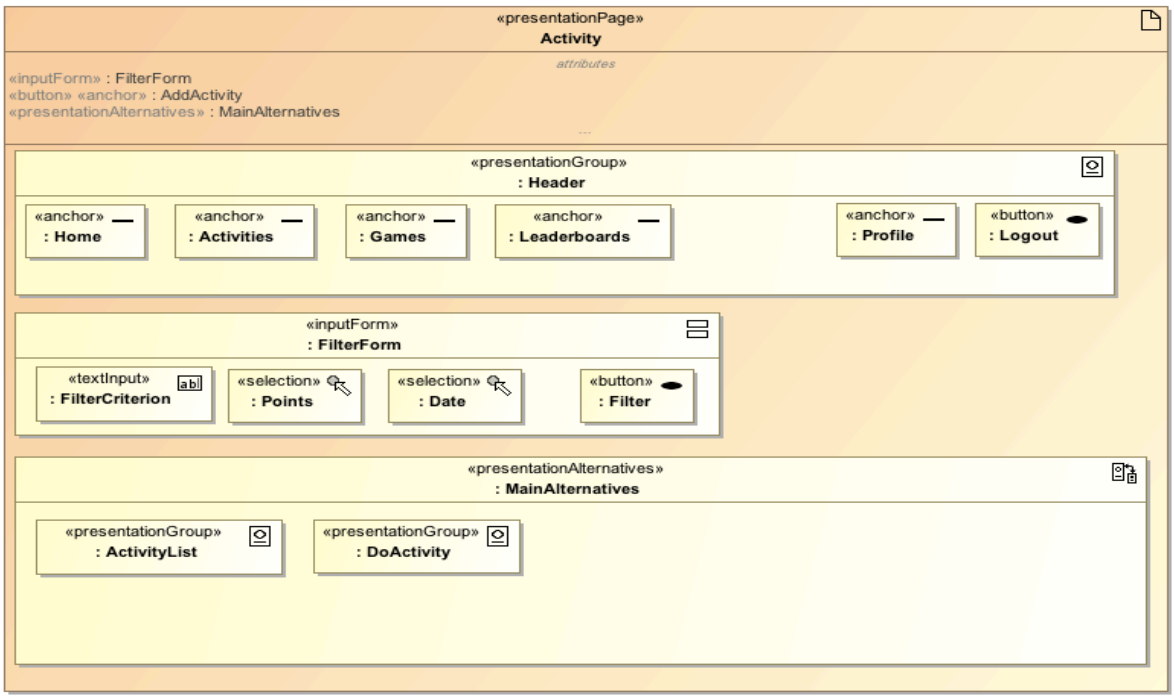

Figure 51 – Player Activity Home

Again, the player ActivityList view is a simplification of the professor's. There are no management options, and the category column is absent as only active and available ones will be displayed, as can be seen in Figure 52.

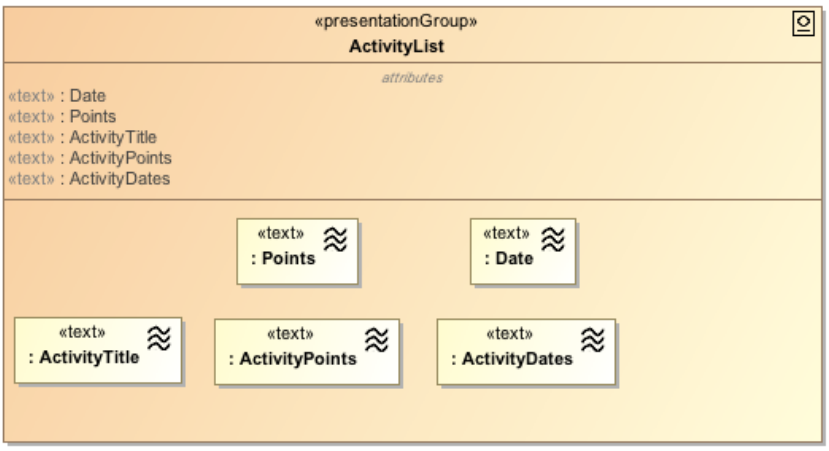

Figure 52 – Player Activity List

The new presentation group, DoActivity, displays the necessary activity information as text and then expects player text input, selection, or the like. These views can be seen in more detail in Figure 53.

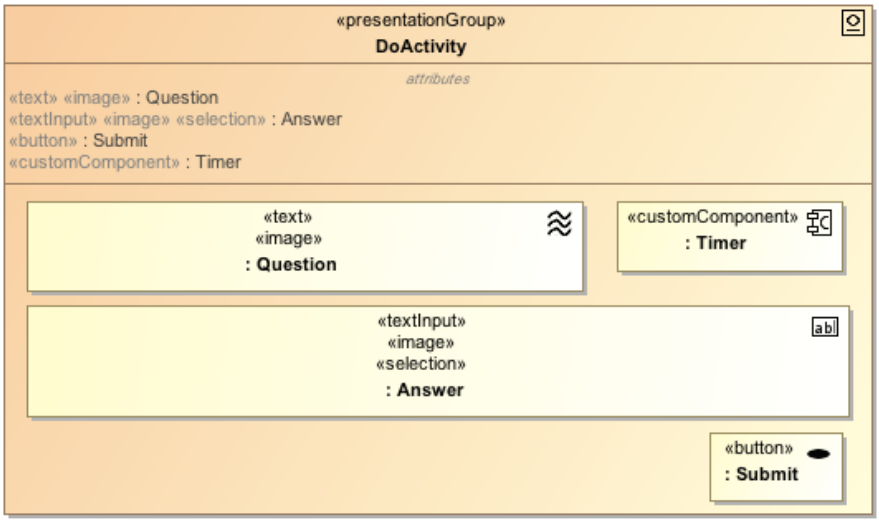

Figure 53 – Player Do Activity

3.4 Database

The database system used in the project is MySQL; and the tool used for its management, MySQL Workbench. With this, an entity-relationship diagram was designed to specify the structure and types for each table. Figure 54 shows the main tables created for the basic functioning of the activity module, detailing the field names and types. Tables for only two activity types are included in the diagram (3.2.3.1 questions with multiple or single correct answers and 3.2.3.2 short answer questions). The structure of the tables not included will be discussed below. It is important to remember that this is not the final database for use in the system; this database must be later merged with others created for other modules of the platform. Additionally, some field type lengths, such as that of answers, may be later changed if seen fitting for the better use of the module.

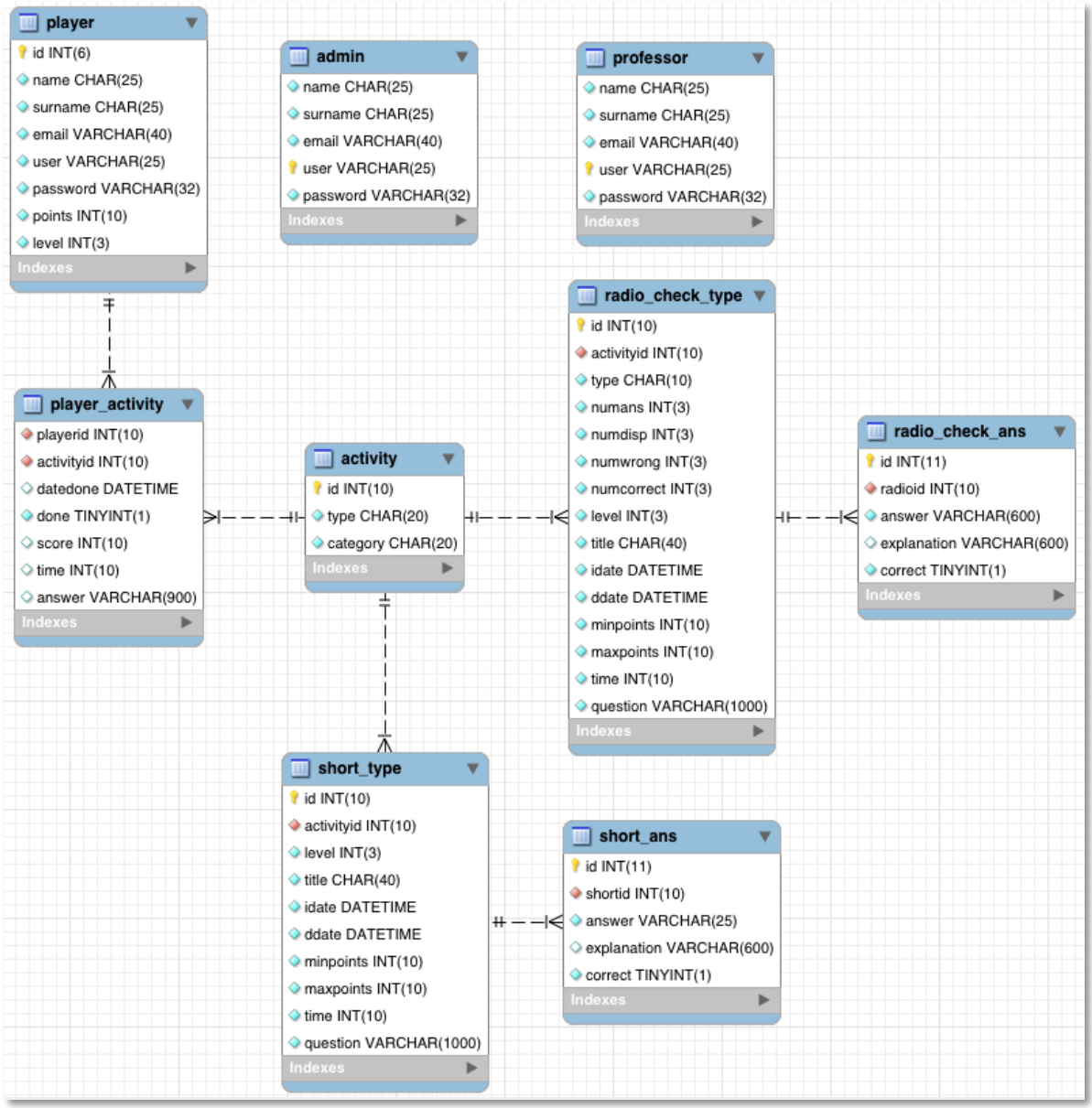

Figure 54 – EER diagram

3.4.1 Player, admin and professor

The three user types share many fields and are quite similar, in fact, the admin and professor tables contain the same information. The player table consists of an id, which is given by the student's university id number and is the primary key, and other identifying data such as name and surname, email, a user name, which must be unique, and password to access the system, and system data points and level. The admin and professor tables are

very much the same, but the id is not required, the user field is used as primary key instead, and points and level are not included.

3.4.2 Activity and player_activity

The table ‘activity’ contains basic and general information of all activities that exist within the system. It only contains an auto incremented id, the type of the activity, and its category (active, inactive, editing or incomplete). Note that although the system expects the category to be one of the four mentioned, the database does not itself set any restrictions of the sort. The type specified in this table tells us to what table type that activity will be linked to. Player_activity is a junction table, containing a reference to a player and an activity through their respective ids. Additionally, this table stores information relevant to how a player has interacted with an activity. The fields include whether the activity has been done, when it was done, the score obtained, the time taken and the selected answers. All fields but the ids and done default to null.

3.4.3 Activity type and answers

For each type of activity, there are two tables: its type table, with information of the activity itself, and the answer table, with information about how to solve it. In Figure 54, two activities are elaborated as radio_check_type, with radio_check_ans, and short_type with short_ans. Each activity type has different fields fitting to its needs however, some will appear frequently.

In the activity type table we can find an id, an activity id that links it with the activity table, the required level for players to partake in the activity, a title, a question, time for completion, minimum and maximum points, and issue and due dates. Some other fields that may be necessary for certain activity types include the total number of answers, the number of answers to be displayed, number of right and wrong answers, review mode, risk and proceed rates.

The answer table varies greatly according to the activity type. Fields that will appear regardless are an id and an activity type id that points to the specific activity type instance. As can be seen in Figure 54, some activity types have similar answer tables; both answer tables displayed share answer, correct, and explanation fields. Other fields for answer tables would be rank, comment, correction, and correct correction.

3.5 Implementation

The implementation of the activities made use of HTML5, CSS3 and PHP5.4 languages through the development environment of PHPStorm. The project was organized into several folders to ease the understanding of each file. As can be seen in Figure 55, there are four main folders in the root, as well as three files. These three files manage the entrance to the module and are temporal in the system, as they are outside the scope of the project. The activities folder is where the main functionality and views of the module are, as all activity related components are found there. As its name suggests, images are saved in the images folder. Meanwhile, scripts folder contains simple scripts and functions for use in other files. Lastly, shared folder stores basic files (which are not scripts) common in other files.

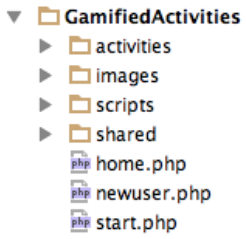

Figure 55 – Project structure

An expanded view of the project structure is shown in Figure 56. Here we can see the files contained in each of the mentioned folders, as well as the manage subfolder within activities. This subfolder separates professor and administrator functionalities from the players'.

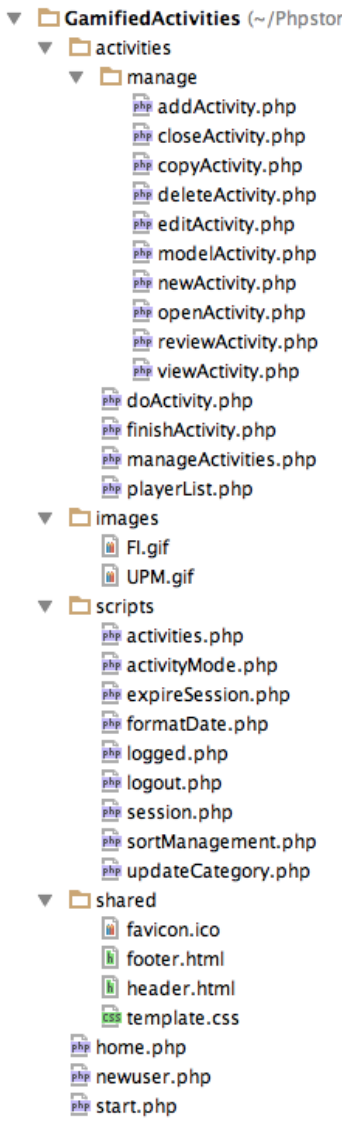

Figure 56 – Expanded project structure

It is easy to deduce what many of the files deal with by their name; especially those that are directly named after use cases (seen in section 3.3.1.2). Regardless, a brief explanation will be provided in Table 4, including additional information for more complex functionalities.

File	Purpose
addActivity.php	View of activity addition, with mode selection (new or model).
closeActivity.php	Deactivation of a selected activity, change of category to inactive.
copyActivity.php	View of activity copied, and editing of its data.
deleteActivity.php	Deletion of a selected activity pending confirmation.
editActivity.php	Editing of activity data, affecting activity type and answer tables in the database.

File	Purpose
modelActivity.php	View of activities available to use as model, activity filter, and view and copy options.
newActivity.php	View of new activity creation, type selection and information.
openActivity.php	View of activation form for a selected activity, with date input.
reviewActivity.php	View of a specific activity's review information. Statistical data, player summary information, and specific player data.
viewActivity.php	View of a specific activity information and short statistical data.
doActivity.php	View of activity to solve, with changes according to activity type.
finishActivity.php	View of activity corrections, corrects activity, checks parameters, and saves data to player and player_activity tables in the database.
manageActivities.php	Administrator and professor activity list view, management options dependent on category, and activity filter.
playerList.php	View of player activity list, showing only activities that meet certain criteria, access to do them; and an activity filter.
FI.gif	ETSIInf logo image.
UPM.gif	UPM logo image.
activities.php	Sorts the activity list depending on the user type and redirects.
activityMode.php	Sorts the activity creation mode into new and model and redirects.
expireSession.php	Defines a function that controls session expiring.
formatDate.php	Defines functions to format and validate datetime and date.
logged.php	Checks login data and logs user if correct.
logout.php	Unsets session parameters and redirects to the home page.
session.php	Restricts access if user is not logged in.
sortManagement.php	Decides where to redirect depending on the management choice selected.
updateCategory.php	Changes activities from active to inactive category or vice versa according to date.
favicon.ico	Site icon image.
footer.html	HTML footer used in view files.
header.html	HTML header used in view files.
template.css	CSS template for use in all view files.
home.php	View of home page, allows for login and new user creation.
newuser.php	Receives form information, checks data, if correct adds a new user to the player and player_activity tables in the database.
start.php	View of start page once logged into the system.

Table 4 – Files and their purpose

3.6 Testing

Two types of testing were designed and performed for the implementation. The first is integration testing, which took place as the implementation advanced, and examined several test cases independently through the use of mock data. The second type is system testing, and did not take place until the basic implementation on the module an integration testing was complete.

3.6.1 Integration Testing

Several simple test cases were created and executed to assess the correct behavior of the code. Depending on the result, one or more iteration of a same integration test was necessary to ascertain compliance with the desired functionality. Table 5 specifies the

behavior expected on a normal and correct use of the functionality. In defines the necessary data for the functionality in an origin: item format, where if the type is directly specified in the origin it is because it is a direct user input. Out refers to the anticipated output. As the aspects are being tested in isolation, stunt data was inserted where necessary. Whatever data was artificially input into the system with the purpose of testing was later removed from the database.

Functionality	Description	In	Out	Database state
User login	Logs a user into the system.	String: username String: password	User session created.	No change.
User creation	Create a new player.	String: id String: name String: surname String: email String: username String: password	Player user session created.	New player and player_activity table rows.
User logout	Log the user out.	None.	User session destroyed.	No change.
Activity list redirect	Redirects the user to its activity list.	Session: user type	Pertaining activity list view.	No change.
Activity list display	Displays the pertaining activities.	Session: user type Session: username Database: player columns Database: player_activity columns	Manageable or doable activity list view.	No change.
Activity filter	Shows activities that meet certain criteria.	Session: user type String: title match Option: activity type Option: category Integer: minpoints Integer: maxpoints String: issue date String: due date	Activity list view that meets the searched criteria.	No change.
Do activity	Display the activity to be done according to activity type.	POST: type POST: typeid Database: activity type rows Database: activity answer rows Database: player_activity rows	View of necessary activity information and answer according to type.	player_activity row updated to show activity as done.
Finish activity	Evaluates the activity, shows corrections and stores results.	Session: activity performance information Database: activity type rows Database: activity answer rows	Activity result and corrections.	player_activity and player updated to store results.

Functionality	Description	In	Out	Database state
Open activity	Changes an activity's category from inactive to active.	GET: type GET: typeid Database: activity type rows	Text notification.	Activity type and activity updated to reflect date and category changes.
Close activity	Changes an activity's category from active to inactive.	GET: type GET: typeid Database: activity type rows	Text notification.	Activity type and activity updated to reflect date and category changes.
Delete activity	Delete an activity from the system and ask confirmation.	GET: type GET: typeid Database: activityid	Text notification.	Deleted rows from activity, activity type, activity answers, player_activity.
View activity	Displays all information about a specified activity.	GET: type GET: typeid Database: activity type rows Database: activity answer rows Database: player_activity rows	View of the selected activity's information.	None.
Review activity	Display statistical information, player performance summary and player performance detail of an activity.	GET: type GET: typeid Database: activity type rows Database: activity answer rows Database: player_activity rows	View of the selected activity's information.	None.
Edit activity	Allow changing information of an activity, as well as adding new data.	GET: type GET: typeid Database: activity rows Database: activity type rows Database: activity answer rows Database: player rows	Edited data is displayed together with all other activity information.	Activity, activity type and activity answer updated to reflect changes. New row in answer if added. New player_activity rows if activity category was incomplete. Category set to appropriate.
Add activity redirect	Redirects to activity new or model mode.	POST: mode	View of selected mode page.	None.
New activity	Create a new activity of a selected type with stunt data.	None.	Text information on activity types and specific type data.	New row added for activity and activity type.

Functionality	Description	In	Out	Database state
Model activity list	Displays a list of complete activities able to be copied and viewed.	None.	View of activities that can be copied.	None.
Copy activity	An activity is added using the information of another as stub data. This information can then be edited.	GET: activity type GET: typeid Database: activity rows Database: activity type rows Database: activity answer rows Database: player rows	Stub or edited activity information is displayed.	New row added for activity, activity type, activity answer and player_activity. Activity, activity type and activity answer updated to reflect changes. Category set to appropriate.
Format datetime	A date is analyzed and returned in proper form.	Parameter: datetime	Datetime formatted and verified.	None.
Format date	A date is analyzed and returned in proper form.	Parameter: date	Date formatted and verified.	None.
Session expiration	The session expires after a set time goes by and more time is added when action takes place.	Parameter: minutes to add	Session is extended by default and parameter minutes. Or user is logged out after the set time goes by.	None.

Table 5 – Test cases for unit testing

Once a functionality has proved to work when used as expected, the integration test is expanded to cover unexpected behavior and incorrect input cases. Examples of unexpected behavior would be backwards navigation or attempting to access a view without the proper authorization, an incorrect input would be introducing a string where an integer is expected, or an issue date later than a due date. After the larger integration test is passed, the code is reviewed for legibility and efficiency.

3.6.2 System Testing

The objective of the system testing was to find if the specification and its purpose was met. For system testing, all the integrated software components are considered input and tested as a whole. Use of stubs or drivers was avoided at this stage as by this stage all necessary data could be generated and made available for system testing.

Some use cases used for testing are described in Table 6. Note that, as opposed to the test cases for integration testing, the test cases for system testing focus on finding defects within the system as a whole.

Description	Units tested	Expected result
User logs in, and then logs out.	User login User logout	The user is no longer logged and the session is destroyed.
A professor adds a new activity. A player then does this activity.	Add activity redirect New activity Edit activity Format datetime Activity redirect Activity list display Do activity Finish activity	An activity is created and player performance data is stored.
A professor creates an activity from a model and then edits its information.	Add activity redirect Model activity list Copy activity Format datetime Edit activity	An activity with the introduced information is created and then updated.
A player does an existing activity. A professor then reviews his performance.	Activity redirect Activity list display Do activity Finish activity Review activity	Player performance information is generated and can be reviewed.
A professor creates a new activity; he then copies this activity to add a new activity.	Add activity redirect New activity Edit activity Format datetime Activity redirect Activity list display Model activity list Copy activity	A new activity and a copy of this activity are created.
A professor opens an activity, then closes it, and finally deletes it.	Activity redirect Activity list display Open activity Close activity Delete activity	The activity changes category to active, then inactive and is then removed from the system.
A player enters his activity list and filters by a certain criteria.	Activity redirect Activity list display Activity filter Format date	A list of activities that meet the criteria is displayed.

Description	Units tested	Expected result
A professor edits an activity, leaves the view without saving progress and filters his activity list by editing category.	Activity redirect Activity list display Activity edit Activity filter	A list of editing category activities, the edited one included.

Table 6 – Test cases for integration testing

More varied test cases were devised and performed to test the system. If a test case fails, the individual integration tests related to those units are rerun to verify their correct functioning, then the logs and code are reviewed to identify the problem.

3.6.3 Additional Testing

After the final period of code review, validation against the SRS (section 3.1) was done to confirm the module implemented all the requirements specified. A full verification of the implementation against the activity design did not take place, given that only a subset of the designed activities was implemented.

Further testing of the module, such as with more extensive system testing (graphical user interface testing, stress testing, etc.), was deemed unnecessary and wasteful given the module is yet to be integrated with the rest of the system and will most likely have to be adapted to it, rendering any additional tests obsolete.

4. Conclusions

In this project we have been able to establish the baseline for the gamified system and fully designed the activity module. By doing this we have exposed how very much this system can help to achieve our objectives. Gamification is still quite a new concept, however, it has become a field for which many companies have shown interest recently.

Although the simple use of gamification techniques does not automatically guarantee success, if applied properly by following the guidelines and frameworks adapted to our objectives and public, it can result in achieving the business objectives proposed.

Nevertheless, we should not limit out thinking to the application of gamification techniques, it is vital that we also store and analyze the data gathered from using the system. With this information we can examine the way the users interact with the system to obtain feedback on how to improve the system.

As stated in section 1.2 the objectives held were to analyze, design, and implement the activity module in order to improve learning, pass rates, and feedback as well as investigating how to incentivize student learning. All these objectives have been successfully met, but the results are yet to be tested in a working environment.

The outcome of this work included the implementation of the core activity module as well as some activity types. Having this, both students and professor will be able to start using the platform for learning and teaching respectively, from which we will be able to observe the results.

The results we expect to obtain in a short term as the users adapt to the new system are:
1. A 10% DAU/MAU (daily/monthly average users) rate
2. A 30% participation rate in activities on average
3. A new activity added at least once every three weeks

As the module and entire system is further developed and the design polished, and the users become more accustomed to using it, we foresee an improvement in the previous measures:
1. Increase of DAU/MAU rate to 40%
2. Average activity participation rate up to 60%
3. New activities added in a weekly basis

5. Future Work

As mentioned earlier, more activity types were designed than were implemented. As a result, the implementation of the types not included must take place. A guide on how to approach this can be found in apendix 7.1

The activity module must first be integrated with the rest of the system, after this, it can be used by students and professors alike. As they use the platform, different statistical data is stored. This information is then analyzed to discover usage, participation, and performance among other rates that evaluate the success of the system.

No design is perfect, the one developed in this project is no exception and as such it may be subject to additions or modifications as usage proves it lacking or insufficient. Some such ideas to expand and improve the design are presented in Table 7.

Idea	Description	Change
Retry activities	Be able to redo activities up to a certain number of times with some penalty.	Additional columns must be added to activity or activity type tables. Pertaining logic must be implemented in activity list, do activity, and finish activity.
Activity hints	A player can ask for a hint while doing an activity in exchange for receiving less reward.	Additional columns must be added to the activity type table. Functionality must be added when doing and finishing the activity.
Dynamic timer	Display a real time timer for activities with time limit.	The timer text must be substituted with a changing animation of the elapsed time.
Encapsulate review data	Categorize review data according to the time it was collected. In this way a professor will be able to compare player performance in different semesters.	Time periods must be defined to then be able to implement a function that sorts the activity instances.
Penalization customization	Allow professors to designate different penalization percentages and even penalize more certain answers.	Additional columns must be added on activity type and activity answer tables. Score calculation logic must be changed when correcting an activity instance.
Keep author record	Store the username of the professor that created and last managed an activity.	Additional columns must be added to the activity table.
Further reviewing	Professors will be able to modify obtained scores and add comments on player performance.	A comment column must be added to player_activity. Information update for player score and comment must be implemented when reviewing.

Idea	Description	Change
HTML syntax	Allow professors to set whether they wish to be able to write using HTML syntax when adding or modifying activities.	Professor input must be treated according to chosen setting.
Activities with videos	An activity where a video forms part of the question and thus must be watched.	This can be implemented as a new activity type, or added to other types with video data columns in the activity type table.
Activities with images	An activity where images can appear in the question or answers.	This can be implemented as a new activity type, or added to other types with image data columns in the activity type and answer tables.
Gentler sequential activities	A new type of activity similar to the sequential activity type but allowing full runs even with mistakes by lessening the rewards and even adding penalties while progressing.	The new activity must be fully designed before being implemented. However, it will follow a structure similar to the original sequential type (section 3.2.3.3).

Table 7 – Future design work

As mentioned earlier in the dissertation (section 1.2), the user interface achieved is quite simple as it was not a main objective of this Final Year Project. Therefore, a more appealing and usable interface design and implementation is pending and future work (in fact, it is planned for this design to start next February).

A goal for the farther future is the development of a mobile app. The idea for the mobile app is not to make it a replica of the web based system, but to have it serve a different purpose. The mobile app would allow for the easy activation of activities and a concise way to do them so that the professor can review the player performance in real time. With this, class lessons would become much more interactive and entertaining for students, as well as allow the professor to quickly assess in which direction the lesson should head. This idea comes from the Socrative system referenced in section 2.1.2.4 .

When a more complete system implementation is achieved, it could also be interesting to use it as a base model for other courses in order to better the learning experience of more students.

6. References

[1] K. Werbach and D. Hunter, “For The Win”, Philadelphia: Wharton Digital Press, 2012

[2] “Samsung Nation”, 2014. [Online]. Available: http://www.samsung.com/us/samsungnation/

[3] “Zappos”, 2014. [Online]. Available: http://www.zappos.com/

[4] “Google news”, 2014. [Online]. Available. https://news.google.com/

[5] Jefatura del Estado de España, “Ley de los Servicios de la Sociedad de la Información y de Comercio Electrónico”, 2002. [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758

[6] Ministerio de sanidad, servicios sociales e igualdad, “Ley General de derechos de las personas con discapacidad y de su inclusión social”, 2013. [Online]. Available: http://www.boe.es/boe/dias/2013/12/03/pdfs/BOE-A-2013-12632.pdf

[7] UWE, “UWE – UML-based Web Engineering”, 2014. [Online]. Available: http://uwe.pst.ifi.lmu.de/

[8] A. Marczewski, "What's the difference between Gamification and Serious Games", 2013. [Online]. Available: http://gamasutra.com/blogs/AndrzejMarczewski/20130311/188218/Whats_the_difference_between_Gamification_and_Serious_Games.php?

[9] R. Bartle, “Hearts, Clubs, Diamonds, Spades: Players Who Suit Muds”, 2012. [Online]. Available: http://mud.co.uk/richard/hcds.htm

[10] N. Lazzaro, “Why We Play Games: Four Keys to More Emotion Without Story”, 2004. [Online]. Available: http://www.xeodesign.com/xeodesign_whyweplaygames.pdf

[11] “ClassDojo”, 2011. [Online]. Available: https://www.classdojo.com/

[12] “KnowRe”, 2013. [Online]. Available: http://www.knowre.com/

[13] “Zondle”, 2011. [Online]. Available: https://www.zondle.com/publicPagesv2/

[14] “Socrative”, 2014. [Online]. Available: http://www.socrative.com/

[15] A. Khan, “Khan Academy”, 2014. [Online]. Available: https://www.khanacademy.org/

[16] Olds College, “Olds College: Alberta”, 1913. [Online]. Available: http://www.oldscollege.ca/

[17] Olds College, “Connect Your Passion”, 2013. [Online]. Available: http://www.oldscollege.ca/about-us/Initiatives/connect-your-passion/index

[18] Olds College, “Spirit of Entrepreneurship”, 2013. [Online]. Available: http://www.oldscollege.ca/about-us/Initiatives/spirit-of-entrepreneurship/index

[19] UWE, “UWE - About UWE”, 2014. [Online]. Available: http://uwe.pst.ifi.lmu.de/aboutUwe.html

[20] UWE, “UWE – Tutorial”, 2014. [Online]. Available: http://uwe.pst.ifi.lmu.de/teachingTutorial.html

[21] WCAG, “WCAG Overview”, 2013. [Online]. Available: http://www.w3.org/WAI/intro/wcag
[22] B. Caldwell, M. Cooper, L. Guarino, G. Vanderheiden, “Web Content Accessibility Guidelines (WCAG) 2.0”, December 2008. [Online]. Available: http://www.w3.org/TR/WCAG20/
[23] I. Hickson, R. Berjon, S. Faulkner, T. Leithead, E. Doyle, E. O'Connor, S. Pfeiffer, “HTML5”, 28 October 2014. [Online]. Available http://www.w3.org/TR/html5/
[24] W3Schools, “HTML5 Introduction”, 2014. [Online]. Available: http://www.w3schools.com/html/html5_intro.asp
[25] W3Schools, “CSS Introduction”, 2014. [Online]. Available: http://www.w3schools.com/css/css3_intro.asp
[26] W3Schools, “PHP Introduction”, 2014. [Online]. Available: http://www.w3schools.com/php/php_intro.asp
[27] W3Schools, “SQL Introduction”, 2014. [Online]. Available: http://www.w3schools.com/sql/sql_intro.asp
[28] W3Schools, “PHP: MySQL Database”, 2014. [Online]. Available: http://www.w3schools.com/php/php_mysql_intro.asp
[29] JetBrains, “PHP IDE: JetBrains PHPStorm”, 2014. [Online]. Available: https://www.jetbrains.com/phpstorm/
[30] IEEE Recommended Practice for Software Requirements Specifications, IEEE Standard 830-1998, 1998.
[31] M. Valipoor, "Gamification Techniques for Students Developing Practical Exercises". Master Thesis, ETSIInf, UPM, 2014.
[32] G. Valderrama, "Gamification, Entregas Online". Practicum, ETSIInf, UPM, 2014.

7. Appendix

7.1 Future Implementation Guide

This section details the steps to take and files to work with when adding a new activity type or functionality to the module. For a general understanding of what each file in the project does refer to Table 4 in section 3.5 When adding new files follow the folder organization exposed on the aforementioned section.

7.1.1 New Activity Types

The core structure of the interaction between activities and the system is already implemented, but this must be adapted when a new activity type is added. Certain files have been written considering no more than one activity type. In these instances, the logic pertaining to this activity type must be abstracted and enclosed in a case, allowing for the new consideration to be placed in a different case.

7.1.1.1 Database

The first important aspect to consider is the database tables necessary for this type. As explained in section 3.4.3, the most intuitive design which works for most activity types (detailed in section 3.2.3) is having two separate tables: one for the basic activity type information and a different one for the answer related data.

The columns that the new database tables will have must be thoroughly considered taking into account the information necessary for the activity type. Some fields are required in all activity types in order for some functionality to work correctly. To know what information must be available read section 3.2.2. The specific required information for each activity type can be found in section 3.2.3.

7.1.1.2 Code

Once the database tables are well thought, the implementation of the new activity type can take place. Table 8 shows which files have a relation with activity types and must be modified and monitored to guarantee the correct inclusion of a new activity type.

File	Reason
closeActivity.php	Table names to update differ according to type.
copyActivity.php	The fields affected when copying and editing the activity differ according to type.
deleteActivity.php	Table names to update differ according to type.
editActivity.php	The fields affected when editing the type table and the answer table differ according to type.
modelActivity.php	All activity type tables must be iterated through to display them (except incomplete) and match them to filter parameters.
newActivity.php	The information to add to the new activity stub will depend on the type.

File	Reason
openActivity.php	Table names to update differ according to type.
reviewActivity.php	The specific activity review information and specific player data will depend on type.
viewActivity.php	The specific activity information will depend on type.
doActivity.php	View and logic will be different according to activity type.
finishActivity.php	Correction logic and display differ according to activity type.
manageActivities.php	All activity type tables must be iterated through to display them and match them to filter parameters.
playerList.php	All activity type tables must be iterated through to find which are available and match them to filter parameters.
updateCategory.php	All activity type tables must be iterated through to check dates.

Table 8 – Files that interact with activity types

Knowing what aspect of the file is related to activity types, it is easier to see how it must be modified to allow for new activity types to be considered. Some files require the same treatment and can be grouped when considering the new code to be added. Additionally, there are modifications that are more complex than others, namely those that require new logic.

To consider a new activity type when closing, opening, or deleting an activity, all that must be done is include a case where the activity type name is examined and then assign the appropriate table name to it.

For views with activity lists (modelActivity.php, manageActivities.php, playerList.php), simply abstracting the type table name and assigning it through a case should be enough to gather all necessary data. In these views only information all activity types must have is fetched, thus making the behavior independent from the activity type.

Similarly, in updateCategory.php, it would be enough to abstract the type and assign the table name through a case as this requires even less data that the lists, looking only for issue and due dates.

copyActivity.php and editActivity.php can also be grouped, as they behave very similarly. These files need closer attention, as all activity type and answer table fields are displayed and able to be modified. This means that not only must the table names be well differentiated, but that each case will have its own query to select and update the related fields. Furthermore, the visual structure of the view might benefit from changes for specific activity types.

Although the modifications for reviewActivity.php are more extensive than those of viewActivity.php, both must be consulted for the specific fields of the corresponding activity type. As in the situation above, the table names must be analyzed and assigned

through a case that will then also define the fields to be consulted. While view requires only activity type table data, review also needs answer table data.

As in the two cases just above, in newActivity.php, exclusive type fields different in each activity type is necessary. The new activity type being created is filtered and a customized insert query sent for each case.

Lastly, the most complicated implementation is that of doActivity.php and finishActivity.php. Here is where all the designed logic must be added. The same case processing as in previous cases is required to make a clear separation in activity types. In these files, it would be advisable to refer to a new file for each type case for a more clear understanding of the different logic applied in each event.

7.1.2 New Functionalities

When adding new functionalities or modifying an existing one, it is important to know where this functionality exists or should be placed. If the functionality is already present, the modification should take place within the same file where it is found. When necessary, a new file can be created to externalize the functionality or expand it. A new functionality should be first evaluated to decide whether it should be implemented in a new file or not. This will depend on how large and or complicated it is and on how many places it will be necessary to use it. If the new functionality is simple and will only be used once, implementing it directly on the affected file is appropriate.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
	Fecha/Hora	Wed Jan 07 19:48:07 CET 2015
	Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
	Numero de Serie	630
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)