GameCoursePersonal

Ana Gonçalves
Instituto Superior Técnico
Lisboa, Portugal
rita.a.goncalves@tecnico.ulisboa.pt

***Abstract*—Gamification is the application of game design elements in non-gaming contexts. In the context of education, experiments with gamified environments have reported an increase in students' motivation and engagement when compared to non-gamified experiences. GameCourse is the system used in Multimedia Content Production (MCP), a MSc course at Instituto Superior Técnico, to provide students a gamified experience. Still, even using such techniques, an underlying problem persists: education often operates under a "one-size-fits-all" model. To better address the students' individual needs, we turned GameCourse into an adaptive gamification environment, with the implementation of a profiler that divides students into clusters, so that the system can adapt to each student cluster. This also included creating a predictor for the ideal number of clusters, as well as a new type of leaderboard, more suitable for a certain type of students.**

***Keywords*—Gamification; Education; Adaptive Gamification; Player Profiling**

I. Introduction

Over the years, there has been an interest in finding solutions for the different problems associated with education. One of the biggest challenges is knowing how to keep students interested and motivated in learning the courses' materials. As teaching techniques evolved, a promising concept that has been introduced and explored over the previous years is the concept of Gamification.

Gamification is "the integration of game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems, in non-game contexts" [1]. With its growing popularity, gamification has been applied in several different contexts, such as healthcare [2] and fitness [3], as well as education [4]. Previous experiences with Gamification in the field of education have reported an increase in students' motivation and engagement when compared to non-gamified experiences. One of those cases is the Multimedia Content Production (MCP), a MSc course for the Information Systems and Computer Engineering field at Instituto Superior Técnico (IST). This course uses GameCourse, a framework that incorporates game elements such as badges and a leaderboard to motivate students to learn the course materials and reach higher levels in the game, which ultimately translates to their final grade.

Even using novel educational techniques such as gamification, an underlying problem persists in systems such as the GameCourse: education often operates under a "one-size-fits-all" model where the learning process is the same for every student, disregarding their singular needs and preferences both as learners and, in the particular case of gamification, as players. To this effect, Adaptive Gamification started being studied. It is a form of gamification that includes processes for adapting the system based on a player model. With this adaptiveness, the learning process could potentially be more inclusive and effective.

The goal of our work was to provide GameCourse the capabilities to adapt to its users in a meaningful and effective manner. We implemented a way for profiling students using machine learning techniques, based on the work by Nabizadeh et al. [5]. Because we divide students into clusters, we also implemented two ways of discovering the best ideal number of clusters based. Finally, we started exploring what to show to different students by designing and creating a new type of leaderboard.

This work was supported in part by the National Funds through the Fundação para a Ciência e a Tecnologia (FCT) under Project UIDB/50021/2020, and in part by the Project GameCourse, Portugal, under Grant PTDC/CCI-CIF/30754/2017.

II. Related Work

A. Gamification

Gamification is the application of game design elements in non-gaming contexts [1]. It has been applied in several different contexts, such as healthcare [2] and fitness [3]. Another context where gamification has been successfully applied is education [4]. According to Furdu et al. [1], applying gamification in learning environments provides a better learning experience, by combining having fun with learning. It is also a way of providing instant feedback since gamification provides metrics that facilitate knowing how a participant is progressing. Furthermore, it makes for a better learning environment because students can evolve at their own pace, as well as choose the assignments at their own will to obtain enough points to pass the course [1]. Gamification is particularly useful in producing desired behavior change through the formation of habits by reinforcing the reward and emotional response of the users, thus requiring fewer cognitive resources each time the desired activity is reproduced [6].

However, in many cases where the application of gamification fails it is because the processes have been inappropriately gamified. This may be due to the lack of understanding of what gamification is, how it works and, more specifically, how to design gamification experiences that inspire player behavior changes and result in desirable outcomes [6].

B. Player Profiling

In order to implement a system that is capable of adapting to its users in a meaningful and effective manner, we must first know how to categorize them into "player types". These player types work as a synthesis of the users' exhibited behavior towards games. Ferro et al. describe player types as being the same synthetization as personality types, with the difference being the context in which they are applied [7].

While some authors believe that personality traits - which influence how someone experiences and perceives the world - are more relevant than player types [7], others have also studied the impact of learning styles - how someone receives, interacts with, and integrates educational materials - on academic achievements and overall experience of gamification, in comparison to the impact of personality traits [8]. An example of categorization by learning styles is Felder's Index of Learning Styles (ILS) [9], which categorizes students according to their learning preferences on four dimensions.

When it comes to personality traits, the most used model is the Five Factor Model (FFM), also known as the OCEAN model or the Big Five model. As complete as this model is, one of the problems that researches often come across is having a limited time to survey participants. For this reason, briefer measures of the FFM have been developed. One of those measures is the Ten Item Personality Measure (TIPI), a 10-item measure of the FFM [10].

In the realm of player types, the most well-known player typology model is Bartle's player type classification [11]. The latest version of this model divides players into eight categories (Opportunists, Planners, Scientists, Hackers, Networkers, Friends, Griefers, Politicians). However, Bartle's model is bound to a specific game genre and may not work in other contexts such as gamification [12]. Then came one of the most recent contributions in this area: BrainHex. Besides not being related to a specific game genre, this model allows players to have both a main type and subtype, and it includes a list of oppositions to each player type. Additionally, there is an online questionnaire for this model. Not convinced that video game focused player types were a good fit for gamification, Marczeweski [13] developed the User Type Hexad. The idea that "people cannot be broken down into simple individual categories" [13] led him to create a model where the three user types - Intrinsic User, Player User, and Disruptor User - can each be divided into four sub-types.

C. Adaptive Gamification

Personalized gamification can be described as the customization of the game elements, the interaction mechanics, the tasks, or the game rules for each user, according to their preferences [14]. According to Todello et al. [14], personalization can be implemented in two ways: as a customization (also referred to as user-initiated personalization), where the user selects the elements that they wish to use, or as an automatic adaptation (also referred to as system-initiated personalization), where the system takes the initiative to select the elements for each user - with or without some user input in the process.

Giving the users the freedom to add or remove elements is particularly useful when the system does not have any information about a student when they first start using it, as this provides an insight on users' preferences. This is known as a cold-start problem [15]. Nevertheless, it appears that users may not be aware of the gaming features that motivate them the most to use the environment, thus, adaptation should not be based on explicit user choices, but rather on indirect measurements through, for example, interaction traces [12].

One approach is to profile users and have set configurations for each type of player profile. Mora et al. textcolorred[10] first surveyed participants to assign them to the group that would better fit them, according to their Hexad user type. The engine included an assignment function that would link each student to the most fitting group: alpha, beta, delta, or gamma. Each of the four groups had a different Trello dashboard to sign in to, which means that students belonging to the same group would have the same view of the system.

An alternative approach is to calculate preferences using the product of two matrices. Monterrat et al. [16] and Lavoué et al. [12] both tested this approach in a very similar way. To determine a student's preference for each feature (R), they calculate the product between the preference of each BrainHex player type for each feature (i.e. game design element) (A) and the users' score for each BrainHex player type (B) (Fig. 1). Nonetheless, this procedure could be used with other profiling methods, such as profiling based on personality traits. The features shown to each user will then correspond to the features with the highest scores for that user.

	f1	f2	f3
u1	10	00	05
u2	00	06	12
u3	06	03	09
u4	-08	03	02

=

	C	S
u1	10	00
u2	00	12
u3	06	06
u4	-08	06

x

	f1	f2	f3
C	1	0	½
S	0	½	1

Fig. 1. An example of linear model R = B A [16].

Machine learning algorithms can also be useful for classifying students based on their level of participation. Mbabu et al. [17] developed a platform that uses machine learning techniques to adapt to students' learning behavior. Due to the small dataset in this study, K-means was used for clustering students. The system initially does not know which cluster each student belongs to. As the user interacts with the system, the adaptive engine evaluates traits and behavior and passes this data to the classifier. A pre-study was done from logs of an e-learning platform to be the training data for the machine learning algorithms used and four main clusters were identified. Each main cluster was given their own set of game elements that would better fit them.

III. Background

GameCourse is a framework that incorporates various game elements with the goal of motivating students to engage in

course activities. A GameCourseUser with admin privileges is the only user who can create courses, activate and deactivate users, and change admin permissions. A CourseUser represents a GameCourseUser inside a specific course. CourseUsers can have different roles in a specific course, which determines the permissions they have in the course. The default roles are "Student" and "Teacher", but other roles can be created.

GameCourse provides a set of modules that can be disabled or enabled at any time for each course (Fig. 2). Each module brings with it new entities and concepts relevant for the gamified experience like badges, skills, and Experience Points (XP), among others. The resulting vocabulary, referred to as the Expression Language (EL), can be used in expressions when designing views and writing rules.

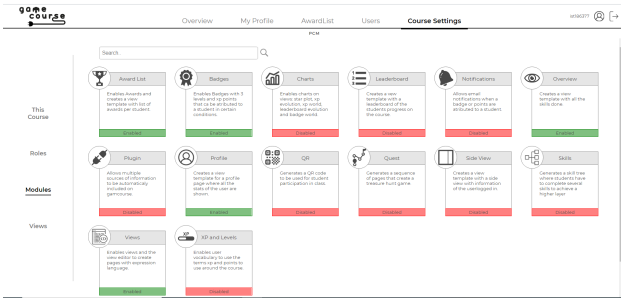

Fig. 2. GameCourse modules.

By enabling the Views module, Teachers can create custom views. These views can then be used to create pages accessible by other CourseUsers (Fig. 3). At the beginning of this work, the views and game elements accessible by each student were the same, although it was already possible to present distinct views to different users, based on their role.

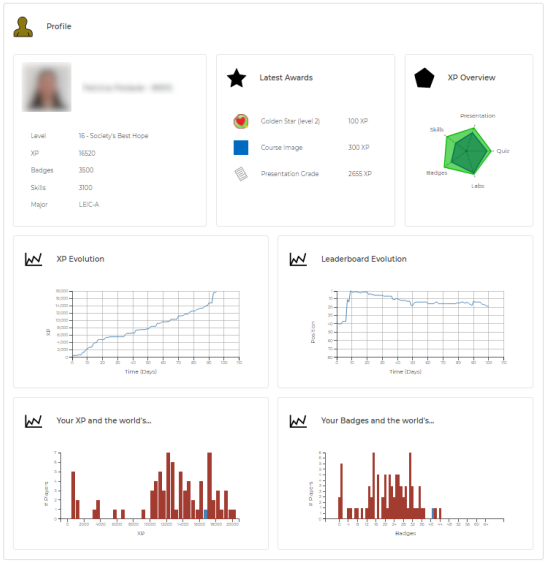

Fig. 3. Student profile page.

Students have many ways of working towards their grade and this is represented by participations, which can then lead to students being awarded XP. All actions that count towards earning XP take place outside of GameCourse. This led to the implementation of the Plugin module that can access, parse, and insert external data into GameCourse's database as participations.

After getting students' participations into the system, GameCourse then transforms these them into awards. Like any game, GameCourse has rules that dictate what conditions must be met in order for the student to get a certain reward. The component that acts as a rule system inside GameCourse is GameRules.

With GameCourse being used in multiple iterations of the MCP course, it was possible to study student behavior and evolution, and compare the results of each year. Barata et al. [18] collected student data from three consecutive terms of MCP. Cluster analysis was performed based on student behavior and performance measures which included, for example, online participation, lecture attendance, and evaluation results. A total of six different clusters were identified, but only four of them were observed in the third iteration of the course. These four clusters were: Achievers, Regular students, Halfhearted students, and Underachievers. Performance-wise, both Achievers and Regular students performed the best, with Halfhearted students performing below average and Underachievers having the worst performance. Participation-wise, the ranking is the same but there is a bigger gap between Achievers and Regular students, with Achievers participating significantly more. As a result of these differences, Achievers and Regular students tend to occupy the top positions of the leaderboard, followed by Halfhearted students and, lastly, the Underachievers. The clusters also differ in which activities they chose to partake in the most.

The latest effort in understanding how different students behave in and interact with gamified environments, with analysis made in the context of MCP while using GameCourse, comes from Nabizadeh et al. [19]. Over a ten-year period, student data was collected and analyzed. Taking into account student performance, they used the elbow technique [20] to identify the optimal number of clusters and determined that number to be four, consistent with previous results. To determine the cluster that each student belonged to, the amount of accumulated XP per day was used as attributes for the cluster analysis later performed using the K-means algorithm [21]. It is important to note that one of the conclusions they came to was that there wasn't any correlation between students' clusters and their BrainHex player types [22].

IV. Development

A. Profiler

The key component when building an adaptive system is the profiler. To implement a system that can adapt to its users in a meaningful and effective manner, it must first know how to categorize them. For our work, we decided that the best solution would be to use the same approach as Nabizadeh et al. [19], developed specifically for MCP.

Because we wanted to automatize the process, we implemented the procedure into a Python script. We refer to the algorithm for determining the cluster each student belongs to as the **profiler**. The profiler requires three arguments to be given: the course ID, the number of clusters, and the minimum cluster size. Both the accumulated XP from the first step and the data that later composes the Main data are represented using dictionaries, and we iterate over the awards and participations from the course identified by the course ID, to collect the necessary data. We then sort the keys in the dictionary containing the data for Main data, which correspond to days. Afterwards, we iterate over each day to transform the data in the dictionary into a Dataframe (from the *pandas* library [23]), that will represent the Main data. The next step is to drop the constant features, using the "VarianceThreshold" feature selector from the *scikit-learn* library [24]. To normalize the resulting Dataframe, we use the "MinMaxScaler" from the same library.

At this point in the execution, the data preprocessing phase has been concluded. The step that follows this phase is to generate the number of clusters (n) that the profiler was given as an input, using the K-Means algorithm, for which we used "KMeans" from the *scikit-learn* library [24]. If any of the clusters doesn't reach the minimum cluster size (given as input), we generate *n - 1* clusters instead. To balance the Main data, we use the "SMOTE" class methods from the *imblearn* library [25], which allow us to perform oversampling using SMOTE. It is then time to detect the most important features in the balanced Main data with the Boruta algorithm. We chose to take advantage of the "BorutaPy" class from the *boruta* library [26] for this step, with a maximum of 2000 iterations. Finally, for the last step, we use the "RandomForestRegressor" class from the *scikit-learn* library [24], with a maximum depth of 10. Once the results are ready, they are written into a file called "`[course ID]` – results.txt". The output written to this file consists in the cluster centre for each cluster and the cluster that each student belongs to.

B. Predictor

We wanted to make this feature as appealing to all courses as possible, and it would be hard to determine the ideal number of clusters for other courses. We anticipated professors wouldn't know which number to choose or even choose a number that would lead to unsuccessful results, so we decided to create a way of advising professors on how many clusters to create. We implemented two options based on the two most well-known methods for predicting the best number of clusters for K-Means. Those two methods are the **Elbow method** [27] (also used by Nabizadeh et al. [5]), and the **Silhouette method** [28].

Like the approach taken for the profiler, we implemented a Python script we'll refer to as the **predictor**. The predictor requires two arguments to be given: the course ID, and a character representing the method we wish to use: "e" for the elbow method and "s" for the silhouette method. The prediction is made based on the XP that each student accumulated thus far, since this is the data that is fed into K-Means in the profiler.

Before explaining how we implemented each of the methods, it is important to explain how these methods work. To determine the ideal number of clusters, it is necessary to calculate a score for each number of clusters, and then decide the best number of clusters based on those scores. This means that we had to choose which numbers we wanted to test for. We decided it didn't make sense to test for less than two clusters (because the students would remain indistinguishable), and for more than ten clusters, due to the contexts in which GameCourse is currently used. Plus, the bigger the array of numbers of clusters we test for, the more overhead we add to the execution. The procedure then becomes similar for both methods: for each number of clusters, we apply the k-means algorithm to the data (using "KMeans" from the *scikit-learn* library [24] as in the profiler), and we then calculate the scores. Each score is stored in a list.

In the case of the elbow method, each score is the sum of squared distances of samples to their closest cluster center and is given by an attribute - inertia_ - from the "KMeans" class. We then create an instance of the "KneeLocator" class from the *kneed* library [29], that receives the list of scores. This class has an attribute – elbow – which indicates the elbow point that corresponds to the number of clusters which has achieved the best score.

For the silhouette method, each score is given by the silhouette coefficient, which we calculate using "silhouette_score" from the *scikit-learn* library [24]. The recommended number of clusters is the one that gets the highest score.

Similarly to the profiler, the predictor writes its output (i.e., the recommended number of clusters) to a file called "`[course ID]` – prediction.txt".

C. Representing Clusters

Users can have multiple roles inside a course, the default ones being "Teacher" and "Student". This is how we individualize a group of users, so that we can give each role the proper permissions and show them the correct views. Since those roles can be assigned or removed at any time, our solution is to represent the different clusters as roles. When a student is assigned to a cluster, besides keeping the "Student" role, they receive the role that represents that cluster. If the student is moved to another cluster, that previous role is removed and a new role is assigned. Because we need to be able to differentiate the "cluster roles" from the other roles, we create them as children of a role called "Profiling" (Fig. 4), the latter being a child of the "Student" role. The views can then be generated with the aspects that match the most specific role possible.

It is important to note that, although the "Achiever", "Regular", "Halfhearted", and "Underachiever" roles are at the same level of the hierarchy, the way we represented them in Fig. 4 was intentional, since their order as children of the "Profiling Role" is important. This order will determine the name of the clusters, based on the cluster centers – the average of

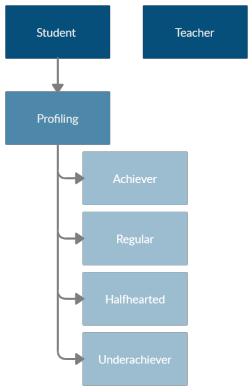

Fig. 4. Role hierarchy diagram.

all points (elements) belonging to each cluster [30]. In this case, the cluster with the highest center will be represented by role "Achiever", the runner up by role "Regular", and so on. Because the names and order of the clusters is determined by the children of the "Profiling" role, professors can choose the names and create as many clusters (roles) as they wish. If these roles are not created prior to running the profiler, the names and order shown in Fig. 4 are used by default. If these names aren't enough, we give the remaining clusters default names: "Cluster1", "Cluster2", and so on.

D. Profiling Module

To encapsulate the new functionalities, we created the Profiling module. Because this is a brand-new module, we had to create a configuration page where the new profiling features can be used. When designing this page, our goal was to provide data regarding students' profiles but, more importantly, to show that data in a meaningful and easily interpretable way.

Once students have been assigned to clusters, it is possible to build a Sankey diagram showing the flow between clusters and how the number of students in each cluster evolved over time (Fig. 5). Because professors can run the profiler multiple times throughout the semester, this diagram is a visual representation of how students are performing in the course at specific points in time, which provides useful feedback for professors. This diagram was implemented using the Highcharts library [31] for Javascript.

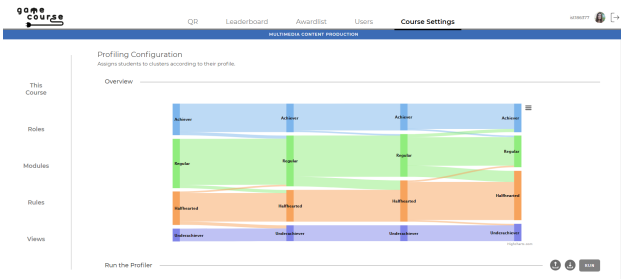

Fig. 5. Module Profiling's configuration page.

The page also contains a section with the tools and settings for running the profiler and the predictor, as well as a table showing the cluster history for each active user (Fig. 6). This table can be sorted by each of the columns.

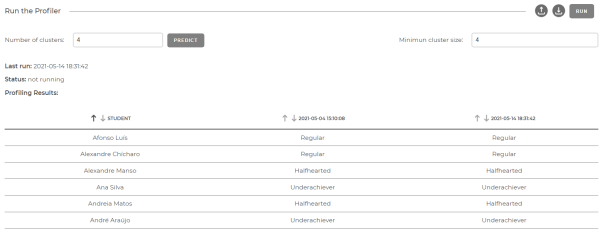

Fig. 6. Partial interface of the Profiling module's configuration page.

Professors can choose the number of clusters and the minimum cluster size in the designated input spaces shown in Fig. 6. The default values that appear on the User Interface (UI) for both of these fields is four, because these were the values determined by Nabizadeh et al. for MCP [5]. The Application Program Interface (API) call to run the profiler makes the profiler start executing in the background and immediately returns so that the UI doesn't freeze waiting for a response. The status will indicate that the profiler is running (Fig. 7), and the "Run" button is replaced by a "Refresh" button to check if results are ready for viewing (Fig. 8).

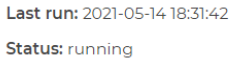

Fig. 7. Status label when the profiler is running.

Fig. 8. Set of buttons when the profiler is running.

Professors can exit the page and even exit the system safely. Once results are ready, a new column is added to the history table, as shown in Fig. 9. Professors can then choose, for each student, between assigning the cluster determined by the profiler or assigning any of the other available clusters. Users can cancel (deletes the results), submit (assigns students to the chosen clusters), and save (saves chosen clusters to an auxiliary database table without submitting) these results. Until the user either cancels or submits, results aren't deleted, and it is not possible to re-run the profiler.

To run the predictor, we give professors the option to choose the method they want to use (Fig. 10). Like the approach we took for the profiler, the predictor runs as a background process so that, in the meantime, the user can freely exit or interact with the system. The output is saved in a file once it's ready, but, while the prediction is underway, the user is not able to re-run the predictor and can only use the "Refresh" button to check if the results are ready (Fig. 11).

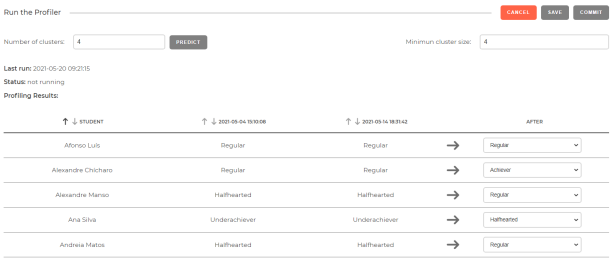

Fig. 9. History table with results from the profiler.

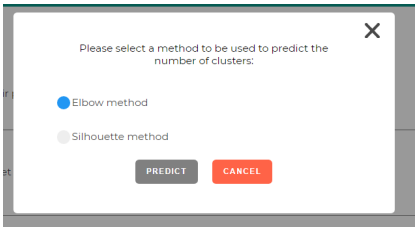

Fig. 10. Predictor method selection.

Once the best number of clusters is determined, the user is informed of the result and must decide if the advised number of clusters should be used for the next time the profiler is ran (Fig. 12). If so, the value on the input box for the number of clusters gets replaced.

E. Towards differentiated leaderboards

As a game design element, a leaderboard consists of "a visual display that ranks players according to their accomplishments" [32]. With leaderboards being one of the most used game design elements in Gamification, they have been the subject of numerous studies over the years. Mixed results on how effective leaderboards are at motivating people have led to trying to understand how different people are affected by this game design element, and how different types of leaderboards may appeal to different people.

It has been concluded that users get different levels of satisfaction depending on the position in which they are placed on the leaderbeard, with optimal positions being the second, fourth, and just above the bottom-three positions (e.g., seventh position if ten users are presented) [33]. This is what originally prompted us to explore different leaderboard designs. There are two main types of leaderboards. The leaderboard used in GameCourse (Fig. 13) is considered an infinite (or absolute) leaderboard, since it displays all users and their scores [34]. As Fig. 13 shows, seeing the positions below sixth requires scrolling down. The second type of leaderboard is the no-disincentive (or relative) leaderboard, where "users only see their rank as compared with the users ranked below and above them" [34]. At first, we thought about creating a different leaderboard design for each cluster we had in MCP. However, we reached the conclusion that, for individual leaderboards, these two designs would suffice. The no-disincentive would be much more suitable for students on the bottom positions (Halfhearted students and Underachivers), and the infinite

Fig. 11. Refresh button when the predictor is running.

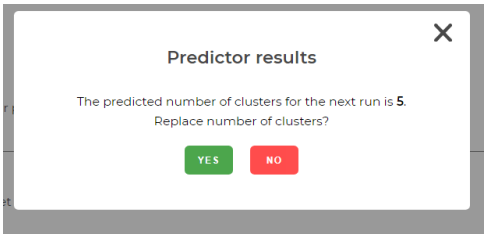

Fig. 12. Predictor results modal.

leaderboard is already suitable for the students in the top positions (Achievers). The reason why we don't advise showing a no-disincentive leaderboard to every student is because students in the top positions may draw more motivation from seeing exactly how many students are behind them and how close they are to the top.

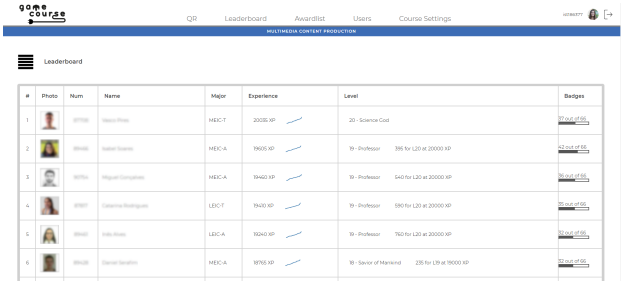

Fig. 13. GameCourse's infinite leaderboard.

Once we decided to provide the option to have no-disincentive leaderboards in GameCourse, we needed to decide the best configuration. The student viewing the leaderboard must appear right in the middle of the leaderboard [35], but we had to decide how many students to show above and below them. We based our decision on the conclusions drawn by Sun et al. [33]. Since the levels of satisfaction are higher for users placing second, fourth, or just above the bottom-three positions, we decided that the best option would be to show the three students above, and three students bellow the viewing student. With this approach, the user is visually always in the fourth position, as well as in the position just above the bottom-three positions. We show the original position next to the students' names, since we don't want to mislead the students into thinking that they are in a position they aren't in. The resulting leaderboard is shown in Fig. 14.

V. Evaluation

We decided to test the performance of the profiler script to find out if we had introduced any unnecessary complexity. The approach that was taken consisted in collecting time samples of how long the profiler took to finish executing

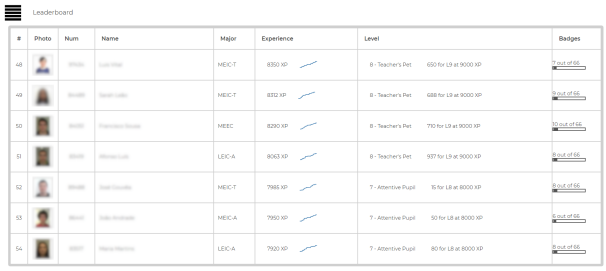

Fig. 14. New no-disincentive leaderboard.

and provide an output. Those time samples were collected for different numbers of students, for different numbers of attributes, and for different numbers of participations, since we wanted to assess how each of these three variables impacted the execution time. For each of these three tests, we ran the profiler twenty times for each value of the variable under test, with both the number of clusters and the minimum cluster size set to four. The values presented below are the average time of each set of samples.

Fig. 15 shows how the execution time varied according to the number of students. We were careful to maintain the number of attributes, since it was inevitable to increase the number of participations and awards due to the profiler ignoring students who don't have participations and awards. Our goal was to simulate the data at the end of the semester for different numbers of students. We can conclude, as expected, that increasing the number of students does increase the execution time.

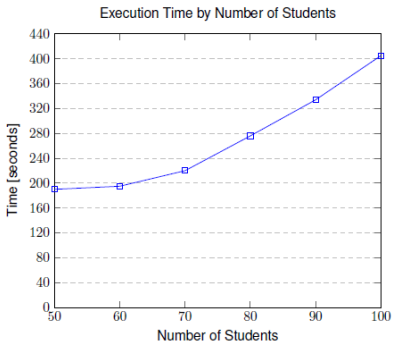

Fig. 15. Profiler execution time by number of students.

According to the documentation for the *scikit-learn* library that was used, k-means' average complexity is given by $\mathcal{O}(k \cdot n \cdot T)$, where k is the number of clusters, n is the number of samples, and T is the number of iterations. However, according to David et al. [36], the worst-case complexity is given by $\mathcal{O}(n^{(k+2/p)})$, where k is the number of clusters, n is the number of samples, and p is the number of features. In our case, the number of features used in k-means is always one (i.e, the accumulated XP), the number of samples corresponds to the number of students, and the number of clusters was kept at four. With this information, we can conclude that the values correspond to what would be expected, and no additional complexity has been introduced when it comes to the manipulation of students, since the time complexity never goes beyond the worst-case complexity (i.e., $\mathcal{O}(n^6)$).

After collecting the time samples for different numbers of students, we set out to test how the number of participations influenced the execution time of the profiler. Fig. 16 shows the results of this experiment, and we can observe that the time increases in a seemingly linear way, as the number of participations increases. Because participations aren't the only attributes taken into account, as explained in IV-A, we were interested in exploring the relation between the number of attributes and the execution time. Because the number of attributes is determined by the number of participations, awards, and days, we varied these values while maintaining the values as realistic as possible. This was achieved by using data from MCP at the end of the semester and deleting the data from entire days until we got the desired number of attributes to conduct each round of tests. The results, presented in Fig. 17, show that an increase in the number of attributes does lead to a longer execution time. There seems to be a higher increase in time when we go from having 1000 attributes to having 1100 attributes.

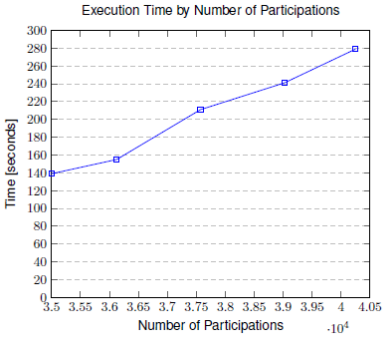

Fig. 16. Profiler execution time by number of participations.

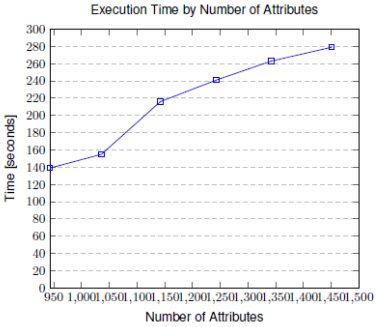

Fig. 17. Profiler execution time by number of attributes.

Both the number of participations and the number attributes appear to have a similar impact on the execution time, which was to be expected since the participations are one of the components that make up the attributes, together with the

awards and the XP accumulated on each day since the first award or participation. The time complexity for the Boruta algorithm $\mathcal{O}(P \cdot N)$, where P and N are, respectively, the number of attributes and objects (in this case, students) [37]. Solé et al. [38] state that the computational complexity for a Random Forest of size T and maximum depth D is $\mathcal{O}(T \cdot D)$. Because the maximum depth is always the same, and we maintained the number of students, we expected the time complexity to be linear. The slope resulting from only varying the number of participations is closest to the slope of a linear time complexity than the slope resulting from varying the number of attributes. Nevertheless, the slope resulting from varying the number of attributes approximates more to that of a linear time complexity, or even a logarithmic time complexity, than that of a super-linear time complexity (e.g., $\mathcal{O}(n^2)$). We can conclude that the observed time complexities are within our expectations.

A. Unit and Integration Tests

To evaluate the correctness of the new Profiling module, we created a test suite using the most popular testing framework for PHP: PHPUnit [39]. We start tests with completely empty database tables and the test classes have the responsibility of both setting up before tests and cleaning up afterwards so that every test starts with a clean environment. Additionally, these tests are completely automatic. With one simple command in the terminal, it is possible to run a single test case or the full test suites. The tests amount to a total of 86 tests. These tests cover both unit testing and integration testing. For integration testing we use a bottom-up integration approach, meaning that we begin by testing the components having the fewest dependencies and follow that order throughout.

VI. Conclusion and Future Work

We stated out this project with the goal of transforming GameCourse into a system with adaptive capabilities. The key to achieving this was creating a profiling script – the profiler –, so that the system could make an informed and effective adaptation to its users. We also focused on providing support to the professors by advising them on the ideal number of clusters to divide students into – the predictor.

This culminated in the creation of the Profiling module. Once we could profile the students, by dividing them into clusters, the next step was to start creating new versions of the existing game elements, better-fitting for the types of students who the former elements weren't ideal for. We took the first step by creating a no-disincentive leaderboard that can be shown to designated users instead of the infinite leaderboard that existed in GameCourse.

To assess whether we had achieved our goals, we performed benchmark tests to evaluate the performance of the system and unit and integration tests to evaluate the correctness of the resulting implementation. In the end, we can confidently say that we've achieved our goal of providing GameCourse the capabilities of profiling and adapting to its users.

Our work has opened the doors to finding creative ways of designing gamified environments tailored to meet students' needs, using GameCourse. We have already taken the first step by creating a new type of leaderboard, but further investigation should be done on what to show to different students based on their cluster. Those differences should be focused on the visual game elements, such as the leaderboard and the profile, and not on game elements that directly influence the final grade such as the badges, skills, and rules, as this could lead to students finding the game unfair. We already know, from previous studies [5] [18], some characteristics of each cluster and what sets them apart from the others, but we have yet to know what they would like to see, for example, on their profile page.

The algorithm we based our profiler on was developed in the context on MCP, but further studies should be conducted to assess how accurate it is for other courses. Other algorithms could be created and integrated into GameCourse, so that there are multiple types of profilers that professors can choose from, like they can already do for the predictor.

As far as the Profiling module configuration page is concerned, it would be helpful, in the future, to show users the progress of the profiler and the predictor when either one of them are running.

Lastly, there should be more communication between the system and the students. There already exists a module – "Notifications" module – that was no longer used nor functional in the start of our work. It was used to notify students of the awards that they had won since the last time they had been active in GameCourse. This could be the starting point to develop notifications that are targeted at different clusters of students, offering suggestions on, for example, what skill to do next, or words of encouragement, like showing how close a student is to getting a badge or placing higher on the leaderboard.

References

[1] I. Furdu, C. Tomozei, and U. Köse, "Pros and cons gamification and gaming in classroom,"Broad Research in Artificial Intelligence and Neuroscience, vol. 8, pp. 56–62, 08 2017

[2] A. Wilson and J. McDonagh, "A gamification model to encourage positive healthcare behaviours in young people with long term conditions,"EAI Endorsed Transactions on Serious Games, vol. 14,p. E3, 05 2014. [Online]. Available: https://doi.org/10.4108/sg.1.2.e3

[3] Z. Zhao, S. Ali Etemad, and A. Arya, "Gamification of exercise and fitness using wearable activity trackers," in Proceedings of the 10th International Symposium on Computer Science in Sports (ISCSS), P. Chung, A. Soltoggio, C. W. Dawson, Q. Meng, and M. Pain, Eds., Springer. Cham: Springer International Publishing, 11 2016, pp. 233–240. [Online]. Available:https://doi.org/10.1007/978-3-319-24560-7_30

[4] A. Mora, E. Planas, and J. Arnedo-Moreno, "Designing game-like activities to engage adult learners in higher education," in Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality, ser. TEEM '16.New York,NY, USA: Association for Computing Machinery, 2016, p. 755–762. [Online]. Available: https://doi.org/10.1145/3012430.3012603

[5] A. H. Nabizadeh, J. Jorge, S. Gama, and D. Gonçalves, "How do students behave in a gamifiedcourse?—a ten-year study,"IEEE Access, vol. 9, pp. 81 008–81 031, 5 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3083238

[6] K. Robson, K. Plangger, J. Kietzmann, I. McCarthy, and L. Pitt, "Is it all a game? understanding the principles of gamification,"Business Horizons, vol. 58, no. 4, pp. 411–420, 04 2015. [Online].Available: https://doi.org/10.1016/j.bushor.2015.03.006

[7] L. Ferro, S. Walz, and S. Greuter, "Towards personalised, gamified systems: An investigation into game design, personality and player typologies," 09 2013.

[8] P. Buckley and E. Doyle, "Individualising gamification: An investigation of the impact of learning styles and personality traits on the efficacy of gamification using a prediction market," Computers & Education, vol. 106, pp. 43–55, 2017. [Online]. Available: https://doi.org/10.1016/j.compedu.2016.11.009

[9] R. Felder and J. Spurlin, "Applications, reliability and validity of the index of learning styles," International Journal of Engineering Education, vol. 21, no. 1, pp. 103–112, 01 2005.

[10] S. D. Gosling, P. J. Rentfrow, and W. B. Swann Jr, "A very brief measure of the big-five personality domains," Journal of Research in personality, vol. 37, no. 6, pp. 504–528, 2003. [Online]. Available: https://doi.org/10.1016/S0092-6566(03)00046-1

[11] R. Bartle, Designing Virtual Worlds. New Riders, 07 2003.

[12] E. Lavoué, B. Monterrat, M. Desmarais, and S. George, "Adaptive gamification for learning environments," IEEE Transactions on Learning Technologies, vol. 12, no. 1, pp. 16–28, 04 2018. [Online]. Available: https://doi.org/10.1109/TLT.2018.2823710

[13] A. Marczewski, User Types HEXAD, 1st ed. CreateSpace Independent Publishing Platform, 10 2015, pp. 65–80.

[14] G. Tondello and L. Nacke, "Validation of user preferences and effects of personalized gamification on task performance," Frontiers in Computer Science, vol. 2, p. 29, 2020. [Online]. Available: https://doi.org/10.3389/fcomp.2020.00029

[15] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in 2010 IEEE International Conference on Data Mining, 2010, pp. 176–185. [Online]. Available: https://doi.org/10.1109/ICDM.2010.129

[16] B. Monterrat, M. Desmarais, E. Lavoué, and S. George, "A player model for adaptive gamification in learning environments," in International conference on artificial intelligence in education. Springer, 06 2015, pp. 297–306. [Online]. Available: https://doi.org/10.1007/978-3-319-19773-930

[17] K. Mbabu, R. Oboko, and S. Kamunya, "An adaptive gamification tool for e-learning platform," Open Journal for Information Technology, vol. 2, no. 2, pp. 41–52, 12 2019. [Online]. Available: https://doi.org/10.32591/coas.ojit.0202.03041m

[18] G. Barata, S. Gama, J. Jorge, and D. Gonçalves, "Studying student differentiation in gamified education: A long-term study," Computers in Human Behavior, vol. 71, pp. 550–585, 06 2017. [Online]. Available: https://doi.org/10.1016/j.chb.2016.08.049

[19] A. H. Nabizadeh, J. Jorge, S. Gama, and D. Gonçalves, "How do students behave in a gamified course?—a ten-year study," IEEE Access, vol. 9, pp. 81 008–81 031, 5 2021. [Online]. Available: https://doi.org/10.1109/ACCESS.2021.3083238

[20] R. Thorndike, "Who belongs in the family?" Psychometrika, vol. 18, no. 4, pp. 267–276, 1953. [Online]. Available: https://doi.org/10.1007/BF02289263

[21] J. Hartigan and M. Wong, "A k-means clustering algorithm," Journal of the Royal Statistical Society: Series C (Applied Statistics), vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: https://doi.org/10.2307/2346830

[22] L. E. Nacke, C. Bateman, and R. L. Mandryk, "Brainhex: Preliminary results from a neurobiological gamer typology survey," in Entertainment Computing – ICEC 2011, J. C. Anacleto, S. Fels, N. Graham, B. Kapralos, M. Saif El-Nasr, and K. Stanley, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 288–293.

[23] "pandas - python data analysis library," accessed: 2021-10-23. [Online]. Available: https://pandas.pydata.org/

[24] "scikit-learn: Machine learning in python," accessed: 2021-10-04. [Online]. Available: https://scikit-learn.org/stable/

[25] "imbalanced-learn documentation," accessed: 2021-10-08. [Online]. Available: https://imbalanced-learn.org/stable/index.html

[26] "Boruta 0.3," accessed: 2021-10-08. [Online]. Available: https://pypi.org/project/Boruta/

[27] R. Thorndike, "Who belongs in the family?" Psychometrika, vol. 18, no. 4, pp. 267–276, 1953. [Online]. Available: https://doi.org/10.1007/BF02289263

[28] L. Kaufman and P. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, 09 2009, vol. 344. [Online]. Available: https://doi.org/10.1002/9780470316801

[29] "kneed 0.7.0," accessed: 2021-10-04. [Online]. Available: https://pypi.org/project/kneed/#description

[30] J. VanderPlas, "In depth: k-means clustering," accessed: 2021-10-04. [Online]. Available: https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html

[31] "Highcharts," accessed: 2021-10-02. [Online]. Available: https://www.highcharts.com

[32] K. Christy and J. Fox, "Leaderboards in a virtual classroom: A test of stereotype threat and social comparison explanations for women's math performance," Computers & Education, vol. 78, pp. 66–77, 2014. [Online]. Available: https://doi.org/10.1016/j.compedu.2014.05.005

[33] E. Sun, B. Jones, S. Traca, and M. Bos, "Leaderboard position psychology: Counterfactual thinking," in Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1217–1222. [Online]. Available: https://doi.org/10.1145/2702613.2732732

[34] M. Ortiz-Rojas, K. Chiluiza, and M. Valcke, "Gamification through leaderboards: An empirical study in engineering education," Computer Applications in Engineering Education, vol. 27, no. 4, pp. 777–788, 2019. [Online]. Available: https://doi.org/10.1002/cae.12116

[35] G. Zichermann and C. Cunningham, Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps, 1st ed. O'Reilly Media, Inc., 2011, pp. 50–51.

[36] D. Arthur and S. Vassilvitskii, "How slow is the k-means method?" in Proceedings of the Twenty-Second Annual Symposium on Computational Geometry, ser. SCG '06. New York, NY, USA: Association for Computing Machinery, 06 2006, p. 144–153. [Online]. Available: https://doi.org/10.1145/1137856.1137880

[37] M. Kursa and W. Rudnicki, "Feature selection with the boruta package," Journal of Statistical Software, vol. 36, no. 11, p. 1–13, 09 2010. [Online]. Available: https://doi.org/10.18637/jss.v036.i11

[38] X. Solé, A. Ramisa, and C. Torras, Evaluation of random forests on large-scale classification problems using a bag-of-visual-words representation. IOS Press, 01 2014, vol. 269, p. 273–276. [Online]. Available: https://doi.org/10.3233/978-1-61499-452-7-273

[39] "Phpunit - the php testing framework," accessed: 2021-10-08. [Online]. Available: https://phpunit.de/