DynaGame: a rule editor for gamified education

Ana Nogueira
Instituto Superior Técnico
Lisbon, Portugal
ana.b.nogueira@tecnico.ulisboa.pt

Amir Hossein Nabizadeh
INESC-ID, University of Lisbon
Lisbon, Portugal
amir.nabizadeh@tecnico.ulisboa.pt

Daniel Gonçalves
INESC-ID, University of Lisbon
Lisbon, Portugal
daniel.goncalves@inesc-id.pt

***Abstract*— Gamification is the usage of game elements in non-game contexts, such as education, where it can increase student's engagement and motivation. Multimedia Content Production (MCP) is a gamified MSc course taking place at Instituto Superior Técnico, resorting to GameCourse, a web application that was solely built for the course. However, there was a need for providing easier and more efficient mechanisms for customizing the gamification elements, ultimately allowing GameCourse to be used in other courses as well. To achieve our goals, the GameRules rule system was integrated with our architecture and a UI was developed to facilitate the rule editing process. This paper presents DynaGame, a user-friendly rule editor that can be easily used by non-programmers to generate and manage existing rules.**

Keywords—gamification, education, blended learning, rule system, rule editor

I. INTRODUCTION

In recent years, the concept of Gamification has garnered attention in multiple contexts due to its benefits in learning, persistence, performance and motivation [1, 2]. One of the contexts where Gamification can be applied and has gathered positive results is education, where new methods have surged in an aim to increase the engagement of students with the class material. In these environments, gamification can depart from the more traditional methods of education and can be leveraged to allow students more control over their learning paths and their learning page.

The Multimedia Content Production (MCP) course at Instituto Superior Técnico, University of Lisbon, is a successful example of a gamified MSc course where positive results have been observed. The MCP course and its gamification process rely on a platform called GameCourse (GC), a web application used for the management of the gamified elements of the course, that recently saw a new rule system component added to its architecture. Previously, the available gamification elements, such as Badges and Experience Points (XP), were awarded by a hard-coded script which allowed no flexibility and was difficult to maintain. The maintenance of the previous system was time-consuming for the teachers and required expert knowledge of the underlying script that was responsible for the attribution of rewards. The addition and integration of a text-based rule system called GameRules has allowed further flexibility and easier customization processes but the platform still lacked a rule editor interface for managing the gamified elements of the course.

We created DynaGame, a rule editor interface for providing and customizing gamification elements in a higher-education context. The rule editor interface leverages the expressivity allowed by the underlying rule system and supplies expert and non-expert users alike with helpful mechanisms to create rules for their gamified courses. To assess our work, we performed a summative evaluation which allowed us to understand the strengths and weaknesses of the developed system.

In the following sections we will first review the available literature on the two areas of research that our work is concerned with: rule systems and rule editors. Next, we provide background on the context in which DynaGame is inserted, followed by an extensive description of the developed rule editor interface. Lastly, we provide an analysis of the work based on the evaluation that was conducted and set forth some conclusions.

II. RELATED WORK

A. Rules and Rule Systems

Rule systems consist software systems that operate over a knowledge base using rules to interpret, process or infer new information. At the center of these systems are rules, usually defined through *if* and *then* clauses, where the if clause is designated as the *antecedent* and the then clause is the *consequent*. The if clause of a rule represents a condition that will trigger the rule, while the then clause refers to the actions that are direct consequences of activating it.

if conditions then actions

Rules can be described in textual representations but there are other means to express these clauses. There are further ways to represent and edit rules based on visual approaches that leverage data-flow techniques, by visually composing the preconditions and actions as flowcharts [3]. Other approaches present rule representation through graphs, for instance, a scenario in which a trained analyst can create rules that represent domain knowledge and use facts to represent data [4]. There is also diversity in the areas where rule-based systems have been successfully applied. Notable uses of rules include areas such as military training [5], the maintenance of medical records [6], but also in clinical diagnosis through the usage of fuzzy sets [7].

B. Rule Editors

The rule editor can be defined as an interface where the users can create the rules that express the logic of the system for inferring new information. Rule editors are conditioned by the underlying representation of the rules that is chosen for a rule system. Programmatically, rules can be stored in similar manners, but we can adopt textual or more graphical

2021 International Conference on Graphics and Interaction (ICGI) | 978-1-6654-8343-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICGI54032.2021.9655275

representations for them. Text-based and programming oriented approaches, usually allow users to directly edit rules through a text box. This was the first model adopted for the structure of our rule system, in which the rules are written programmatically in text files and parsed on execution.

Graphical approaches are an alternative to purely textual ones and can encompass many GUI elements that together increase the overall expressiveness of the system. Ghiani et al. [8] presented a complex rule editor that allows users to customize IoT environments using trigger-action rules. In this model, the rule editing starts with selecting a Trigger or an Action to author an expression. Both of these expressions can be simple or complex, and can contain boolean operations, such as AND or OR. In the editor itself, the concepts used for composing rules are organized hierarchically through contextual dimensions, which limits the amount of items visible in order to reduce the cognitive load for the user.

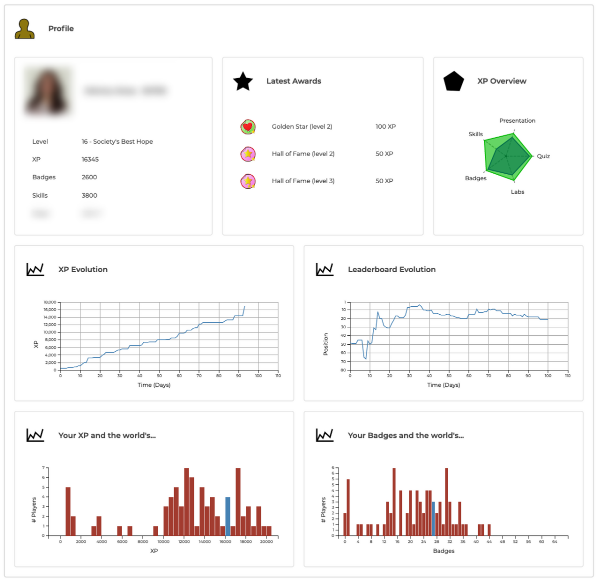
Another system was proposed for a similar context of a Smart Home and Internet of Things (IoT) environment where users are also in control of their environments through compositions of sensors and IoT devices [9]. A rule editor, with both textual and graphical elements was developed for users to manage their devices. In this system, each rule is described symbolically and textually by *if*, *while* and *then* clauses, and each clause consists of three elements: a graphical icon for describing the service type (e.g. motion detector), a service name or location info (e.g. hall), and a description for an event, condition or action (e.g. raise alarm). The rule editor contains drag-and-drop features, graphical icons and clause templates.

Zhang et al. [10] presented a rule editor for clinical applications, where domain experts make use of a rule system to map domain knowledge into computer processable contents using if-then clauses. The rule editor presented has two modes: graphical and textual. In graphical mode, the user is presented with a table of all available rule variables, which can be used to compose new rules using comparison operators and user specified values, by using dropdown menus and input fields. Each clause can be dragged up or down to adjust the order of the rule, and after the completion of the rule fields, the graphical editor can generate a text-based rule expression from the specified clauses. In textual mode, the user is allowed to write the desired rule manually and then validate it, without recourse to the graphical editor. Additionally, the textual mode provides auto-complete and intelliSense functionalities to guide the user.

Regier et al. [11] present a browser-based rule editor for a Clinical Decision System to help manage clinical reminders for new research findings. Previously, the aforementioned reminders were hard-coded and had less functionalities. Their work proposed a new system that has three main advantages: greater control when modifying the reminders; reduced time when making or applying modifications; the usage of rules that could be easily understood by non-programmers. Being domain specific, the rule editor includes a set of primitives that represent commonly used expressions in the clinical context. For instance, a medication primitive that can be configured according to different parameters, such as a medication subset or a medication start date. The method for creating rules relies on preset templates that are adapted to the nature of the primitives.

III. GameCourse

Multimedia Content Production (MCP) is a gamified MSc course which uses a blended learning model of education supported by an online system called GameCourse. This platform is responsible for the attribution of XP and other rewards to the students enrolled in the course in return for the completion of tasks. GameCourse previously relied on a hard-coded script for all its reward attribution, which conditioned the system's flexibility, requiring the expertise of programmers and other users that were familiar with the existing architecture. However, due to the work of multiple professors and MSc students, it is now a modular system in which each of the available modules provides extra functionality and module-specific language. Through its modules, GameCourse provides various gamification rewards such as XP, Badges and Skills, and other components that are central to the user experience, such as Profile Pages or a Leaderboard, as shown on Fig. 1.


Profile

Level	16 - Society's Best Hope
XP	16345
Badges	2600
Skills	3800

Latest Awards

Golden Star (level 2)	100 XP
Hall of Fame (level 2)	50 XP
Hall of Fame (level 3)	50 XP

1. A student's Profile Page on the GameCourse platform.

The reward attribution is now delegated to a previously developed customizable rule system component called GameRules (GR) that was integrated with the GameCourse architecture. Through the usage of text-based user-created rules, the GameRules rule system defines which types of awards are redeemable in the course. On execution, it checks whether target users, have triggered any of the defined award-attributing rules previously established by the administrators. The GameRules system accepts text-based rules, written in Python-like syntax, with a two clause format: the *when* and *then* clauses. The *when* clause is the *antecedent* of the rule, which may cause the *then* clause, or the *consequent*, to be executed. In Fig. 2 we present the structure of a rule accepted by the rule system.

```
rule: Lab Grade
tags: grades
# give grades from lab classes

        when:
                logs = GC.participations.getParticipations(target, "lab grade")
                len(logs) > 0
        then:
                award_grade(target, "Lab", logs)
```

2. Structure of a plain-text version of a GameRules rule.

The addition of the new rule system also included the integration of module-specific language into the syntax of the rules. GameCourse has a built-in Expression Language (EL) that can be used to customize front-end templates for the pages available in the platform. It can also be used, however, to retrieve information from the system's modules, which is a useful functionality to be included in rules as well. Another one of the features included allows the retraction of effects of previous rules. A use case for this particular feature is when work submitted by a student is re-evaluated by a teacher and, as a consequence, the student no longer deserves a previously unlocked reward.

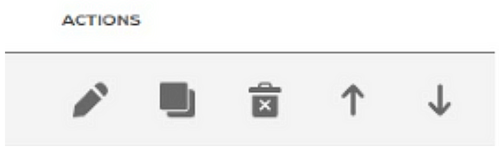
IV. Designing The Rule Editor

A rule editor interface was developed, leveraging the existing GameRules text-based rule-system and integrating into our work into the overall design of the GameCourse system. The rule editor consists of two main screens: a rule listing page and a rule editing page. In the rule listing page, the existing rules for a given course are presented to the user, grouped in collapsible *section* taxonomies. As we present in Fig. 3, each of these named sections has a set of actions associated with it:

- Add rule: adding a new rule to a section;
- Export rules: exporting all the rules in a section into a downloadable plain-text file;
- Move up: moving a section up, causing its precedence within the rule system execution to increase;
- Move down: moving a section down, causing its precedence within the rule system execution to decrease;
- Collapse/Expand: hiding or showing the rules under the section, respectively.

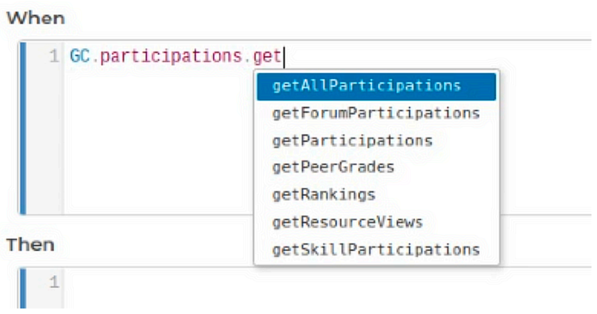
3. Set of icons used for section specific actions.

Each rule available also has its own actions, presented in Fig. 4: each rule can be edited, duplicated, deleted and moved up or down to increase or decrease its precedence, respectively. Additionally, it has a set of attributes, such as a *name, status* (active or inactive), *description and* a set of optional *tags.* We delegate to our rule editing page the mechanisms for editing some of these attributes, as well as having the main editing of the rule clauses that allow us to define functionality for our gamified system.



4. Icons representative of available rule actions.

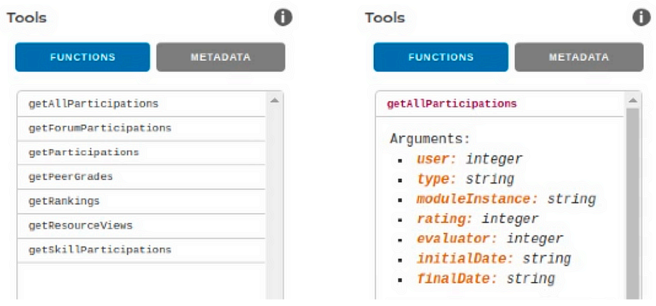
The GameRules system accepts rules with two clauses, as already mentioned. To edit rules with this format, we provide two text boxes in our rule editing page, one for each clause, with autocomplete functionality and automatic suggestions. As the user types code for each clause, function suggestions are given in the right-side of the page and an autocomplete tooltip is presented for easier selection, similar to that of many of the current IDEs.



```
GC.participations.get
getAllParticipations
getForumParticipations
getParticipations
getPeerGrades
getRankings
getResourceViews
getSkillParticipations
```

5. Autocomplete suggestions provided by the rule editor.

For our clause text-boxes we used the CodeMirror [12] framework, since it provides syntax highlighting and the autocomplete functionality we sought. Our rule system uses Python-like syntax for its rules but it also contains tailored expressions specific to modules of the GC environment which are integrated into the rules through a custom language parser. Since it was of our interest to have these custom expressions parsed and suggested in the interface, we tweaked the CodeMirror configuration to detect them as the user types them, and provide suggestions on the go, as seen on Fig. 5.



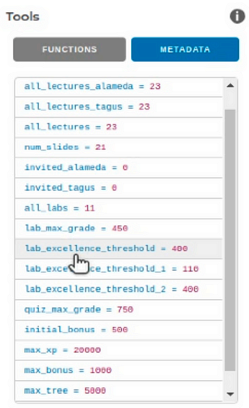
```
getAllParticipations
getForumParticipations
getParticipations
getPeerGrades
getRankings
getResourceViews
getSkillParticipations
```

```
getAllParticipations

Arguments:
 • user: integer
 • type: string
 • moduleInstance: string
 • rating: integer
 • evaluator: integer
 • initialDate: string
 • finalDate: string
```

6. Function suggestions mechanism before and after selecting a function.

When the user finishes typing the name of valid expression or function on the text boxes, a list of arguments and a description of that function/expression is presented on



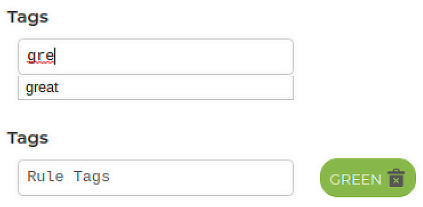
```
all_lectures_alameda = 23
all_lectures_tagus = 23
all_lectures = 23
num_slides = 21
invited_alameda = 0
invited_tagus = 0
all_labs = 11
lab_max_grade = 450
lab_excellence_threshold = 400
lab_exce[illegible]ce_threshold_1 = 110
lab_excellence_threshold_2 = 400
quiz_max_grade = 750
initial_bonus = 500
max_xp = 20000
max_bonus = 1000
max_tree = 5000
```

7. Global rule system metadata variables suggested in the rule editing page.

the right side of the page, as presented on Fig. 6. The descriptions are either obtained through a Python script, for the case of Python defined functions, or by the internal expression dictionary of the GameCourse system.

In addition to listing functions, the rule edit page presents *metadata variables*: variables that are system-wide and whose value is *global* to all the rules of a course. The value of these variables cannot be changed in the rule edit page, but its value is visible in the suggestions box, as is shown in Fig. 7. Clicking on one of these variables will cause it to be added to the clause.

In the rule editing page, we have included the mechanism for adding tags to a rule. Typing on the text box will display a tooltip with suggestions of existing tags, as is shown on the top of Fig. 8. To insert a new tag, the user must press Enter, which will prompt a color-picker to appear for a color to be selected for the new tag. The newly added tag will then be listed next to the text-box with the selected color.

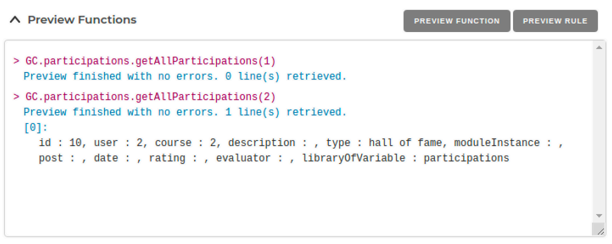


8. Tag functionality during and after creation of a new tag.

In the bottom of the page we implemented a preview section that allows users to preview the outcome of functions they have typed on the clause boxes or, alternatively, to test the created rule in a contained environment. The *rule preview* is implemented by running a minimal version of the rule system which does not track a user's progress in gamification elements, recalculate the XP or log any activity but gathers the obtained results in a separate test database table. In the presence of an error in the syntax of a rule, the user will be informed of the type of error in the preview box so that it can be corrected. Otherwise, the preview script will return a sample of the awards that were attributed on the test database, which represent the results the users will obtain when the rule is ran in the actual gamified environment.

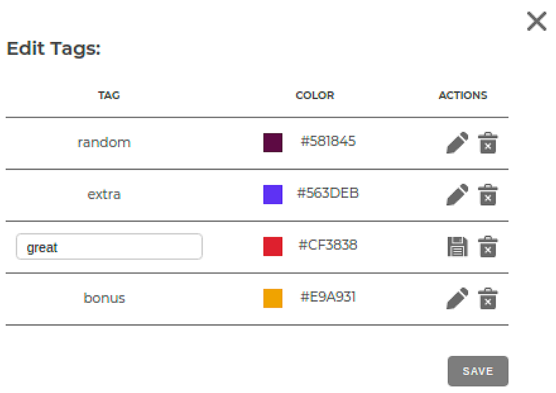
The function preview mechanism, exemplified in Fig. 9, allows users to test the Expression Language functions that are available, which arguments of their choosing. In Fig. 9, we se the *getAllParticipations* function of the *participations* library being tested for the user with the id 2. The result is a detailed list of participations user 2 has made.

In our rule listing page a set of icons is displayed on the top-right of the page. In this menu, the users are able to add new sections, edit existing tags, export and import rules and configure various settings of the rule system. The *Edit Tags* menu, accessible by clicking in the tags icon, gives users the possibility to edit the name and color of tags already on the system. A user must first select a tag to be edited which causes the name of the tag to switch to an editable text box, as we can see on Fig. 10. The color sample box shown in each row of the menu also becomes editable and the color of a tag can be changed by clicking on it. As a consequence, a color picker will appear, where the user can select a new color. The edited tag must then be saved individually. In addition, for the changes to be saved, the user must save all tags as a group as well, using the *Save* button.



```
> GC.participations.getAllParticipations(1)
  Preview finished with no errors. 0 line(s) retrieved.
> GC.participations.getAllParticipations(2)
  Preview finished with no errors. 1 line(s) retrieved.
  [0]:
    id : 10, user : 2, course : 2, description : , type : hall of fame, moduleInstance : ,
    post : , date : , rating : , evaluator : , libraryOfVariable : participations
```

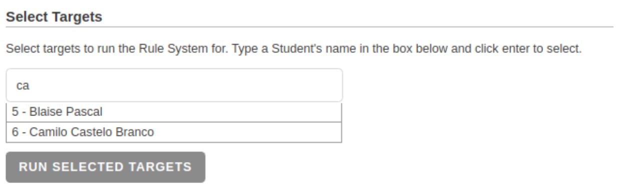
9. The *Function Preview* area allows users to test rules and Expression Language functions.



TAG	COLOR	ACTIONS
random	🟪 #581845	
extra	🟦 #563DEB	
great	🟥 #CF3838	
bonus	🟧 #E9A931	

10. *Edit Tags* menu during a the process of editing a tag.

The aforementioned *Settings* menu is accessible by clicking the wheel icon on the top-right. In this menu we provide rule system related functionality that does not relate directly to rule editing. Clicking the wheel icon causes a modal window to be opened, where various rule system

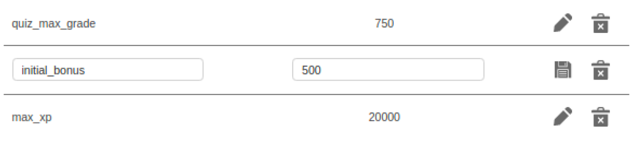


Select targets to run the Rule System for. Type a Student's name in the box below and click enter to select.

ca
5 - Blaise Pascal
6 - Camilo Castelo Branco

11. The *Select Targets* menu for running the rule system allows users to choose which students to run the rules for.

configuration mechanisms are available. Inside this modal, different areas are organized in tabs. Within the first tab, we provide users with a button to run the rule system on demand. GameRules' main use-case relies on it being executed periodically, with that periodicity being dependent on user defined parameters and on the amount of data the system receives. However, during the first semester in which the rule system was used (without the rule editor) it became clear there was a need for it to be manually ran on demand, or for specific users. This functionality was added in the *Settings* menu — we now provide a mechanism for



quiz_max_grade	750	
initial_bonus	500	
max_xp	20000	

12. Metadata Variable editing in the *Rule Settings* modal.

GameRules to be run normally but on demand and a mechanism for the professors to specify which users they want to run the rules for.

The *Select Targets* functionality, as shown of Fig. 11, retrieves all available students registered in GameCourse and suggests them as the user types a name or a user id on the text box. After the targets are selected, the chosen targets will be listed on the right side of the text box. Clicking the button below the box will cause the rule system to be executed in the background for the specified targets only.

In another tab of the *Settings* menu, we implemented an editing menu for the metadata variables that are *global* to the system. Here the user can add, edit or delete variables to be used in rule creation. The editing process is similar to the one already described for the *Edit Tags* menu.

V. EVALUATION

To assess the performance of our work, we performed summative tests with a group of 20 users, which constitutes a varied sample. The tests were performed remotely using video-conferencing software with recording capabilities. Users were asked to perform 11 tasks using the system, during which we gathered relevant metrics such as the number of clicks, errors, the time expended in each task and whether the task was successfully completed.

We gathered some details about our users, such as their age, programming experience and whether they had previous knowledge of the GameCourse system. Some context was then provided regarding the Multimedia Content Production course and the practical application of the rule editor, followed by a limited time period in which the users were allowed to explore the rule editor before performing the tasks.

We devised a set of tasks that would allow us to test different areas of the system.

1) Change the *Initial Bonus* rule so that it awards *300 XP*.
2) Add the tag *Books* with the color Green to the rule "Librarian".
3) Add a new empty Rule called *Library* to the Section "Badges".
4) Open rule *Empty* and tell me the name of the first argument of the function *getAllParticipations* of the *participations* library.
5) Run the Rule System only for targets *Blaise Pascal* and *Camilo Castelo Branco*.
6) Add a new metadata variable to the RuleSystem called *sleep_hours* with the value *2*.
7) Perform the following:
- *a)* Disable Rule *Amphitheater Lover.*
- *b)* Delete Rule *Golden Star.*
- *c)* Duplicate Rule *Artist.*

8) Open rule *Hall of Fame* and preview the rule. If no errors are returned, save the rule.
9) Change the color of tag *great* to red.
10) Create a new rule with the name *Great Prize* that awards a prize of *400 XP* to all students.
11) Create a new rule named *Quiz* that awards XP for participations of the type *quiz grade*.

A time limit was set for each of these tasks — tasks **1**, **4**, **10** and **11** had a maximum time of 2 minutes and 30 seconds, due to its increased difficulty and the remaining tasks had a time limit of 1 minute and 30 seconds.

We came up with this specific set of tasks by fixating on specific elements and functionality we intended to test. Tasks 3, 7 and 8 test the simplest interactions with the system, such as locating rules and understanding other basic actions in which the object of the task is a rule. Tasks 2 and 9 allowed us to specifically test the tag functionality, more precisely, how users are inclined to solve tag related actions such as editing and adding new tags. With Tasks 5 and 6 we intended to understand if users could interpret our request correctly and locate the correct place to solve the task, since these tasks target more *global* functionality of the rule system, as opposed to targeting rules. The remaining tasks, were all chosen for their ability to test the experience of users when editing or interpreting rules. This type of tasks gives us a wide scope of interaction for analysis, since we can look at how users go about when creating new rules, which elements they interacted with, how they interpreted existing code and suggestions and the difficulties they may have found.

The age of our participants ranged from 21 to 63 years old, with the median age of 24 years. 85% or our users had programming experience, either from a professional or academic background.

I. TABLE 1

Task	***Success Rate***	***Avg. completion time***	***Avg. nr. of clicks***
1	70%	1:08	4.85
2	35%	1:20	11.4
3	100%	0:25	3.75
4	60%	1:41	4.35
5	90%	0:42	7.45
6	35%	0:50	8.85
7	100%	0:29	4.75
8	90%	0:33	3.8
9	30%	0:31	8.3
10	20%	2:20	5.9
11	0%	2:29	4.4

Users were asked to rate each task in a scale of 1 *(Very Easy)* to 7 *(Very Difficult)* after they had completed it. It is worth looking into these tasks and analyzing which are the most common pitfalls that users have made, since it will give us a better idea on how to improve them.

We can first look at the tasks our set of participants considered to be the most straight-forward. Tasks **3**, **7** and **8** were considered the easiest tasks, rating in the difficulty scale, on average, with values < 2.0. Users had no major hardship in performing slight alterations to rules, in understanding the *Section* taxonomy or in previewing a rule. The success rate for each of these three tasks was above or equal to 90%. Despite having no difficulty in identifying where to preview a rule, our users were reluctant to spontaneously use that functionality which can indicate that it goes by unnoticed or that its function is unclear.

On a slightly higher difficulty level, we have tasks **1**, **5** and **6**. These were all rated in the average difficulty between 2.0 and 3.0. What all these tasks have in common is that they require some extra knowledge on how the system works.

Task **5** asks the users to "run the rule system" for specific targets. This implies that the user must have a slight understanding that the rule system is the general object on which the rule editor is based and that running it is a course-wide action. Understanding this detail would eventually lead the user to the *Settings menu* accessible from the rule list page, where the task could be completed. In addition, we also noticed that users expected some feedback from this particular feature, which was not provided, given that the rule system runs in the background and might take some time to run. We could, however, improve the interface to provide the user some feedback on whether the request for execution was completed.

An analogous situation occurs with task **6**, in which the user must comprehend that a metadata variable is a global to all rules in the course and that the mechanism for changing it must also be a global one. Once again, the location for completing the rule is within the *Settings menu*. The success rate for this task was only of 35%, which is a very low percentage. We observed that users would not correctly save the variable after adding it to the list. The menu required the user to save the variable individually, and later save the whole set of variables. However, the button for saving all variables was not visible at all times due to the list of variables being too long, and only the users who scrolled to the bottom of the page realized that another *Save* action was required. The remaining would just close the modal window, assuming that they had finished adding a new variable. An alternative mechanism for saving might be worth considering for this menu and similar ones.

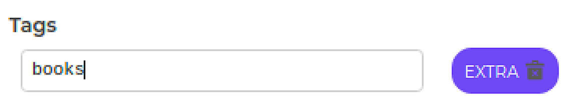
Task **1** asks the users to change the *Initial Bonus* rule already in the system. This task was particularly interesting, since there were two valid methods for completing it. Some users interpreted the available code and changed it on the spot to award 300 XP, as requested, by typing

$$bonus = 300$$

Another fraction of the users interpreted the *Initial Bonus* as being a metadata variable and that changing the bonus should be a task performed on the Settings menu, as shown in Fig. 12.

Next, we can look at tasks **9** and **2** together, since they both concern adding and editing tags. Tasks 2 and 9 have an average reported difficulty of 4.35 and 2.05, respectively.

In task **2** we were able to identify an important issue within the UI that constituted an impediment for the completion of the task. The creation of a new tag inside the rule editing page, as seen on Fig. 13., was achieved by writing the name of the new tag into a text input and then pressing Enter to confirm. This mechanism turned out to be completely counterintuitive to our users, with only 7 of them effectively completing the task.



13. *Tag Creation* issue: the creation of a new tag was not easily understandable by the majority of the users.

We also noticed that most users would first try to open the *Edit Tags* menu to create the new tag, rather than opening the requested rule, so it might be productive to add that functionality in that modal. Exceptionally, some users were more creative in completing the task, by altering the name and color of one of the existing tags in the system. Then they would open the rule editing page and attribute that rule the new tag, which would now be suggested by the system. While this counts as having completed the task, it indicates a lot of misdirection.

In task **9** we report a similar problem to that of task 6, in which the user is required to save twice, first individually and then as a group. Both situations occurred, in which users skipped one on the steps needed for saving the modifications made on a tag. In this task we also observed that it was not very explicit how to change the color of a tag. Users' first impulse would frequently be to click on the hex-code of the current color, instead of the colored square which would prompt the color picker to show up.

Lastly, we analyze the results of the tasks concerning rule creation or code interaction, namely tasks **4**, **10** and **11**. These tasks required extended interaction with the rule editor and the function suggestion mechanism. Consequently, the measurements that were made about errors and clicks are less expressive than a detailed analysis of what the users tried to accomplish when interacting.

One of the indications we provide, but realized is not explicit enough, is the text suggestion to the right hand side that prompts the user to type *GC.* in the When text box to receive suggestions. Many users ignore this initial hint and end up being confused about which course of action to take when creating a rule. The users who overcome this first hurdle, then might run into further obstacles. One behavior that was very regularly observed was that users would try to click the suggestions showing up on the functions box, a functionality that is not provided by our UI. The correct way to follow a suggestion is by using the autocomplete suggestions or by typing it all.

Another obstacle the users faced when writing rules was the uncertainty on how to write an EL expression. Users, very often, were not aware that they must first select the library to which a function belongs and only then choose the function name. It was also not clear sometimes that a "." should be used to separate the library and the function name when using the Expression Language functions, as we exemplified in Fig. 5. The experience of users without programming knowledge was further dampened by the fact they did not know how a function works, namely that its invocation includes an argument list enclosed by parenthesis.

Testing allowed us to extract some data about how the users interact with our work. We can observe in Table 1 that the average number of clicks in task 2, a task which demands the addition of a new tag to an existing rule, is significantly higher when compared to other tasks. From the amount of errors that took place, on average, we realized that the correct path for solving this task was not clear to our users, who repeatedly struggled with completing the task, thus making it clear for us that there is room for improvement of this particular feature. In addition, most users were reluctant to use the Preview mechanisms provided, which might mean that it goes by unnoticed or that its function is unclear. When it comes to rule creation, users without programming experience had more difficulty in completing the tasks which required interaction with the rule clauses. This type of user often times failed to understand how a function works, which signals that further support should be provided by our suggestion system.

VI. Conclusions

Gamification has no doubt become of interest in many contexts, namely in the educational context, where positive results have been observed. Our goal was to improve an existing gamification system currently used in blended learning environments by making it more flexible and easily customizable by programmers and non-programmers.

With this goal in mind, we built DynaGame, a system which encompasses our efforts at creating and integrating a rule editor interface into the GameCourse platform. With DynaGame we provide an interface for users to customize rule-based gamification elements while leveraging the underlying Expression Language of the system. Summative testing has allowed us to evaluate our work and draw conclusions on how target users interact with it. The overall balance is positive, however, the data shows there is still room for improvement.

As future work, we aim to improve the suggestion system in the rule editor to provide more support for users without programming experience, considering the results our evaluation. We suggest the addition of more features that help novice users interact with the system, such as template rules for specific types of rewards or a beginner's guide for rule creation. Another interesting addition would be to have different themes available for the rule editor, while still making sure that the whole system maintains its visual consistency.

ACKNOWLEDGMENT

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.

REFERENCES

1. R. Garris, R. Ahlers and J. E. Driskell, "Games, motivation, and learning: A research and practice model", Simulation & gaming, 2002.
2. G. Barata, S. Gama, J. Jorge and D. Gonçalves, "So Fun It Hurts - Gamifying an Engineering Course", International Conference on Augmented Cognition (pp. 639-648), 2013.
3. M. Mosconi and M. Porta, "A Data-Flow Visual Approach to Symbolic Computing: Implementing a Production-Rule-Based Programming System through a General-Purpose Data-Flow VL", Proceeding 2000 IEEE International Symposium on Visual Languages. IEEE, 2000.
4. M. Nowak, J. Bak and C. Jedrzejek, "Graph-based rule editor", RuleML (2). 2012.
5. Y. Shanliang, F. Yuewen, Z. Peng and H. Kedi, "Implementation of a Rule-based Expert System for Application of Weapon System of Systems", Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC). IEEE, 2013.
6. R. Regier, R. Gurjar and R. A. Rocha, "A Clinical Rule Editor in an Electronic Medical Record setting: Development, Design and Implementation", AMIA Annual Symposium Proceedings. Vol. 2009. American Medical Informatics Association, 2009.
7. M. A. Ghahazi, M. H. Harirchian, M. H. F. Zarandi, S. R. Damirchi-Darasi, "Fuzzy Rule based Expert System for Diagnosis of Multiple Sclerosis", 2014 IEEE Conference on Norbert Wiener in the 21st Century (21CW) (pp. 1-5). IEEE.
8. G. Ghiani, M. Manca, F. Paterno` and C. Santoro, "Personalization of Context-Dependent Applica- tions Through Trigger-Action Rules", ACM Transactions on Computer-Human Interaction (TOCHI) 24.2 (2017): 1-33.
9. T. Tuomisto, A. Haapasalo and K. Hakala, "Simple Rule Editor for the Internet of Things", International Conference on Intelligent Environments, 2014.
10. Y. Zhang, H. Li, H. Duan and Q. Shang, "An Integration Profile of Rule Engines for Clinical Decision Support Systems", 2016 International Conference on Progress in Informatics and Computing (PIC), IEEE, 2016.
11. R. Regier, R. Gurjar and R. A. Rocha, "A Clinical Rule Editor in an Electronic Medical Record setting: Development, Design and Implementation", AMIA Annual Symposium Proceedings. Vol. 2009. American Medical Informatics Association, 2009.
12. CodeMirror, https://codemirror.net/. Last accessed 22 July 2021.