Gamification based Learning Environment for Computer Science Students

Imre Zsigmond, Maria Iuliana Bocicor and Arthur-Jozsef Molnar
Faculty of Mathematics and Computer Science, Babeș-Bolyai University,
1 Mihail Kogălniceanu, RO-400084 Cluj-Napoca, Romania

Keywords: Gamification, Digital Badges, Static Analysis, Learning Environment, System Design.

Abstract: In the present paper we propose an integrated system that acts as a gamification-driven learning environment for computer science students. Gamification elements have been successfully applied in many fields, resulting in increased involvement of individuals and improved outcomes. Our idea is to employ a well-known aspect of gamification - awarding badges, to students who solve their assignments while observing best practices. The system is deployed as a standalone server having a web front-end through which students submit assignment source code. The system checks submissions for plagiarism using Stanford's MOSS, and statically analyzes it via SonarQube, where a custom set of rules is applied. Finally, the program is executed in a sandboxed environment with input/output redirection and a number of predefined test cases. Badges are awarded based on the results of static and dynamic analyses. Components of the proposed system were previously evaluated within several University computer science courses and their positive impact was noted by both students and teaching staff.

1 INTRODUCTION

Our increased reliance on technology during the past decades has left a market impact on all facets of life. Education is no exception and e-learning systems have proven their worth in this domain, as they promote accessibility, personalised learning, adaptability as well as many other advantages. Computer science is one such field where online learning environments are most beneficial and convenient. They provide effective tools for students, who can work on and upload assignments remotely, can receive instant feedback, as well as for teaching staff, through automated verification, detection of plagiarism and grading.

Gamification is defined by the use of game design elements in non game contexts and is getting increased attention and use in e-learning environments (Deterding et al., 2011). While the definition allows for various implementations, in practice the most common game mechanics are points, badges and leaderboards (Raftopoulos et al., 2015). Game mechanics are methods users can invoke to interact with the game state in order to produce gameplay (Sicart, 2008).

Originally, badges were coat of arms subsets that denoted outstanding skill or merit (Halavais, 2012). Other instances of historical inspiration were medals, trophies or ranks. In the modern context of gamification, badges are used to the same effect, with the qualifier that they tend to be available online, contain information on how they were achieved, and have a representational image and title (Gibson et al., 2015). A list of all achievable badges can usually be found in the application that uses them, and they tend to be different enough to be categorized into distinct types. From a user experience or system design perspective, badges:

- Record user progress.
- Encourage alternative behaviors defined by the designer.
- Highlight specific user experience elements that are not covered by tutorials.
- Hint at advanced features or additional content.
- Serve as important metrics on application use.

As an illustration of the use and usefulness of digital badges consider the work of (Majuri and Hamari, 2018). Out of the 128 gamification papers analyzed by the authors, 47 implemented some kind of digital badges and 91 reported results. Out of the 39 papers that detailed the use of badges 64.10% reported positive, 30.77% reported mixed and only 5.13% reported

Zsigmond, I., Bocicor, M. and Molnar, A.
Gamification based Learning Environment for Computer Science Students.
DOI: 10.5220/0009579305560563
In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2020)*, pages 556-563
ISBN: 978-989-758-421-3

negative results. Studied papers tended not to report on specific badge usage, and did not have a clear description of why almost a third of them have ended up with mixed results. We keep this issue in consideration by tracking how badges are awarded and their impact on student involvement and the learning process.

Our approach extends previous work presented in (Zsigmond, 2019) in order to create a generalized badge awarding system to be used within University-level courses on computer programming and software engineering. The system must provide sufficient customization to support the results of up-to-date research in the field. It must be extensible in order to cover relevant coursework within a 3-year undergraduate program. Adding and configuring new courses, assignment types and gamification elements must be well supported.

2 BACKGROUND

2.1 Gamification

In most gamification literature, badges appear as a secondary consideration. In our case, they represent the main gamification element of the proposed platform. The main proposed innovation stems from its integration within the platform, together with the usage of both static and dynamic code analyses for awarding them.

Existing literature shows that the applications of gamification are not limited to e-learning. (Johnson et al., 2016) analyze 19 papers on gamification in health and well-being, and find that 59% report positive, and 41% report mixed results. Behaviors relating to physical health were overwhelmingly positive while those focused on cognition were positive and mixed. Studies that investigated badge use in education tended to emphasize improved interaction with the material and friendly competition with one's peers.

A computer engineering master level course was gamified in (Barata et al., 2013), with the result of increased forum engagement between 511% to 845% compared to the previous year. A data structures course was gamified with the use of badges in (Fresno et al., 2018). Authors found positive effects on trial and error as well as task submission behavior. A freshman's orientation application was created in (Fitz-Walter et al., 2011), with badges being the motivator to attend events, befriend new people and explore the campus. Authors report positive attitude towards the activities in the surveys.

High school students participated in a gamified online vocabulary study group in a work described by (Abrams and Walsh, 2014). Authors reported increased time spent by students on the site and an increased number of words learnt. (Cavusoglu et al., 2015) argued that badges are valuable in stimulating voluntary participation. The authors of (Anderson et al., 2013) provide empirical data to show that users not only value badges, but modify their behavior in an attempt to earn them. In (Huynh et al., 2016), authors conducted data analysis on Duolingo's[1] milestones, and found statistically significant correlation between user progress and badges. In a subsequent analysis, the same authors investigated badges and winning streaks, noting that advanced users found them less interesting than beginners. (Abramovich et al., 2013) supported these findings, as they discovered the existence of a connection between the students prior knowledge and how they valued badges.

Some existing works also reported mixed or predominantly negative results. (De-Marcos et al., 2014) reports that a cooperation based social network outcompeted a competition based gamification group. (Ruipérez-Valiente et al., 2017) implemented their own metric for badge usage on their engineering course, and after grouping students by performance found no overall clear correlation between badge use and success. (Hamari, 2013) added badges to a purely utilitarian trading service and found no behavioral change in users, concluding that badges would be more suitable within a more hedonistic usage context.

2.2 Static Code Analysis

Source code is meant to be computer interpreted. Well understood best practices and increased processing power lead to the development of software tools that enforce coding standards. Many of them also attempt to find bugs before and during program execution. Tools such as FindBugs[2] and Programming Mistake Detector[3] support multiple languages and cover an extensive list of source code issues by parsing the program's abstract syntax tree. Such tools are integrated within most IDEs in the form of source code linters such as SonarLint[4]. From an educational perspective, these tools are preferable as they provide students with immediate feedback and encourage early development of good coding practices.

One of the most widely used such tools is the

[1]Duolingo language learning platform - https://www.duolingo.com/

[2]http://findbugs.sourceforge.net/

[3]https://pmd.github.io/

[4]https://www.sonarlint.org/

SonarQube platform. Freely available in the form of a *Community Edition*[5] that supports 15 popular programming languages, SonarQube is available as a multi-platform server application accessible through a web interface. A command-line tool is available for scanning project source code and uploading it to the server for analysis. SonarQube checks source code against an extensive list of possible errors, security issues, code duplication, and proposes possible refactorings. The platform also provides an evaluation of key software product characteristics such as Maintainability, Reliability and Security.

In addition to its open-source nature, there were two important reasons for selecting SonarQube as our platform's static analysis tool. First, source code analysis is implemented using a plugin system. Plugins can be created in order to cover new languages or to improve the analysis of already supported languages by implementing new rules and guidelines. This allows us to customize the set of analysis checks at both course and assignment level. As an example, it allows us to ensure that user interaction is confined to a user interface module, and that program functions communicate using parameters and return calls, as opposed to global variables. Second, SonarQube includes the required tools to automate source code analysis and result extraction, which allows us to seamlessly integrate it within our platform. We assign high importance to both these requirements in order to ensure the proposed platform covers the maximum number of computer programming and software engineering courses.

3 GAMIFICATION BASED LEARNING ENVIRONMENT

3.1 Gamification Study

This work is a continuation of (Zsigmond, 2019). The platform was already prototyped within a successful experiment in the spring semester of the 2018/2019 school year. Technical details are available in Section 3.4. This work further develops the presented solution by incorporating a mechanism to employ both static and dynamic code analysis in order to award digital badges. Code analysis is described in Sections 3.2 and 3.4, while the awarding algorithm is detailed in Section 3.3.

The popularity of digital badges can easily be linked with the ubiquity of video games (Hilliard, 2013), where they are used in most genres. A welcome side effect of this usage is that we can be reasonably confident that both students and the teaching staff have already become familiar with the concept. One of the leading theories on why badges work is social comparison theory (Festinger, 1954). It emerges from people comparing their digital social status, in the form of earned badges, to other people. Alternatively it is used as a benchmark for themselves. More recent and specific theories on the subject, influenced by the previous one, are social influence theory and that of planned behavior (Ajzen, 1991).

We aim to create 2 groups of badges: achievement and style. Badges in the achievement group target performance and mastery. Their aim is to be non trivial, optional and worth extra points toward the final grade. For example, a typical badge in this category would be awarded for extending an assignment that requires persistent storage to text files, to also work with JSON files. Badges in the style group target minor achievements or desired behavior. Their aim is for students to try small deviations from the optimal study plan as well as to reinforce desired behaviors. A typical badge in this category would be awarded for completing a specific assignment on time. Badges from both categories fall under the types defined in (Facey-Shaw et al., 2018): motivation and engagement, awareness and behavior change, and recognition of achievement.

We aim to carry out an experimental evaluation of our approach by enrolling all 14 student formations[6] who take programming courses. Student formations will be assigned randomly into one of 4 groups, 3 experimental and 1 control, as summarized in Table 1. The experimental question is how does student behavior change in relation to being exposed to either, both, or none of the aforementioned badge groups. Data will be collected about their behavior on the site, their grades, as well as a Likert scale survey to infer details we cannot deduce otherwise.

Table 1: Number of student formations in each experimental group.

		Achievement badges	
		With	Without
Style badges	With	3	3
	Without	3	5

3.2 Static Code Analysis

As detailed in Section 2.2 and shown in Figure 2, our platform integrates a *SonarQube Community Edition*

[5]https://www.sonarsource.com/plans-and-pricing/community/

[6]Approximately 200 students in total.

server instance for static code analysis. The community version supports most popular programming languages with the notable exception of C/C++, which is covered through a third party open-source plugin (Sonar Open Community, 2020). Assignment source code is extracted server-side and imported into SonarQube automatically. The server configuration employed for analysis can be configured at both programming language and assignment level, allowing different rules to be enforced within the same course, depending on the current assignment. The server can be accessed through the same rich API that is used by its web front-end. This provides programmatic access to project and rule configurations, as well as detailed information regarding rule violations, code smell information and technical debt calculated using the SQALE method (Letouzey and Ilkiewicz, 2012).

All submitted assignments undergo static analysis. The result represents one of the inputs for the badge awarding algorithm, as shown in the examples within Table 2.

3.3 Awarding Algorithm

Defining an awarding system involves identifying and recording the achievements, as well as their trigger conditions. A good set of achievements is not straightforward to define, as it must take into account course objectives, the students' difficulties, challenges and ambitions. Important criteria in defining achievements are represented by clearly stating the achievement's requirements, in order to drive a good understanding of what needs to be done to gain the associated badge. Although a complete set of achievements is not yet available, Table 2 illustrates a few relevant examples, based on our preliminary studies. The requirement represents the condition that must be fulfilled in order to get the badge. The reward represents what can be obtained together with the badge, provided the requirement is satisfied; not all achievements have a reward. The value is a numerical amount that can be gained with the badge, while the accumulated values may lead to some new reward for the student.

Available achievements and their associated badges can be consulted by students at any time. Students can also consult already earned badges. Achievements are defined using clear, straightforward language. This makes it easy for students to understand them, and allows new achievements to be added in a straightforward manner. Once the solution to an assignment is uploaded, students are awarded badges according to achievement requirements.

The awarding scheme described represents the

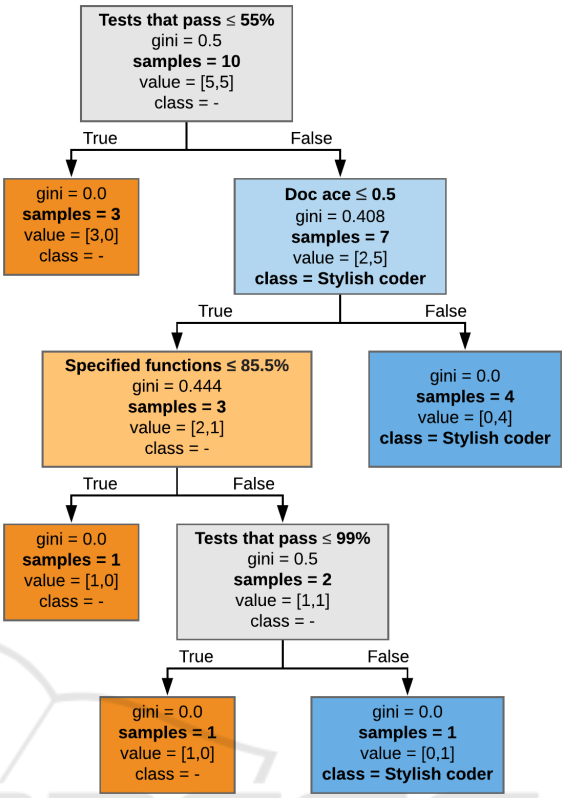

Figure 1: Example of decision tree, built for a mock data set. Each node contains the test attribute, the number of samples and the majority class of samples associated to that node. Blue nodes indicate that all samples are classified as *Stylish coder*, while light-blue indicates that the majority of samples are in this category. The same is true for orange and light-orange, with the difference that the samples are not awarded the *Stylish coder* badge. Grey nodes indicate that the two categories are equally represented (equal number of *Stylish coder* and non-*Stylish coder* samples).

first phase in the implementation of our long-term plans. The next phase is to define and employ more complex, progressive meta achievements, which recognise student achievements in a certain sub-field such as testing, documentation, or coding style. This will also be linked with other accomplishments of the student that did not merit issuing a standalone badge. This second, more complex achievement type employs more elaborate rules for which we expect a more complex server-side implementation. For this reason, instead of directly defining a set of intricate conditionals, we propose a more generalised approach, in which starting from an initial set of manually assigned progressive and meta-badges, the system will be able to learn the rules by which these are assigned and to further automatically award badges to new users, according to learnt criteria. For this task we propose using decision trees

Table 2: Achievement examples with associated badges.

Id	Requirement	Badge title	Reward	Value	Category
1	Function specifications for $> 90\%$ of functions	Doc ace	-	5	style
2	Five bonus activities	Rising star	One less requirement for the next assignment	100	achievement
3	Percentage of code coverage $\geq 95\%$	Test master	Bonus points for the final grade	-	achievement
4	Among the first 10% of students to submit the assignment	Road Runner	-	50	achievement
5	All assignments on time	Time manager	-	-	style
6	Functions communicate without using global variables	Rightful transferer	-	10	style
7	No functions or classes above the maximum allowed complexity level	Complexity guru	-	30	achievement

(Quinlan, 1986; Quinlan, 2014), classification models that are incrementally constructed from input data and that can also be linearized into rules of type $IF\ condition_1\ and\ condition_2\ and\ \cdots\ and\ condition_n\ THEN\ result$, if needed.

The system described in Section 3.4, enriched with the badge awarding and static code analysis components will be put into operation during the spring semester of the current academic year[7]. Collected and anonymised data, including the students' achievements and received badges will be curated in a data set where each instance represents a student. Each instance is characterised by the same set of attributes: recorded realisations and received badges. These will be available since the system keeps a complete history of students' delivered assignments and the results of their evaluation. Some features are numeric, such as the number of assignments completed on time, or the percentage of code coverage. Others are boolean in nature such as being among the fist 10% of students to submit an assignment, or already having a certain badge. The number of features is not yet fixed, but it will be the sum of all recorded realisations and all badges that we defined in the system during the first phase. This is the input data for the decision tree. Its creation also requires the target variable, whose values will be represented by the new set of progressive, meta-achievement badges. Initially these will be devised and manually assigned by the teaching staff, who understand the importance and context of the students' realisations as well as the gamification mechanics. Once these outputs are available, the decision tree will be generated and further validated by means of the *DecisionTreeClassifier* from scikit-learn (Pedregosa et al., 2011), which uses an optimised version of the CART algorithm (Breiman, 2017). The tree thus trained is intended to be used for the following generations, to automatically award progressive and meta-badges. This constitutes the second phase of our proposed awarding system. Nonetheless, this second phase is not the final one. As the system is dynamic, more and more data will be recorded each semester and new elements, including achievements will be added. Therefore the classifier will have to be retrained periodically using the new data available.

For better illustrating our idea, we built a mock data set with 10 instances and 14 attributes (for each feature we specify in brackets whether it is boolean - B - represented by of 0 or 1, or numeric - N): *Activities attended* (N), *Assignments on time* (N), *Time manager Badge* (B), *Specified functions (%)* (N), *Doc ace Badge* (B), *Documentation delivered* (B), *Code coverage (%)* (N), *Test master Badge* (B), *Number of bonus activities* (N), *Rising star Badge* (B), *Tests that pass (%)* (N), *All tests pass from first submission* (B), *Among the first 10% of students who complete all assignments* (B), *Layered architecture* (B). The output is represented by a meta-achievement which indicates whether the student's coding style is proper or not. It is influenced by aspects such as the presence of function specifications and tests, program structure, as well as other elements extracted using static code analysis. For the current example we decided to use a binary output variable, whether the coding style is good or not, with the associated *Stylish coder* meta-badge. In reality, the data set will include at least 200 students together with their assignment data. More meta-badges will be created, leading to the problem of multi-class classification. Figure 1 shows the created decision tree, using the mock data set. Its most relevant attribute is placed at the root and as one moves further down the tree the splitting measure helps in deciding which is the best split at every node. This is computed using a splitting measure such as information gain or the Gini impurity index. Each node contains its associated Gini index (Raileanu and Stof-

[7]Second semester of 2019/2020.

fel, 2004), the number of samples associated to that node, a "value" that represents how many of the samples sorted into that node fall into each of the two categories, as well as the majority class of samples associated to that node.

3.4 System Design

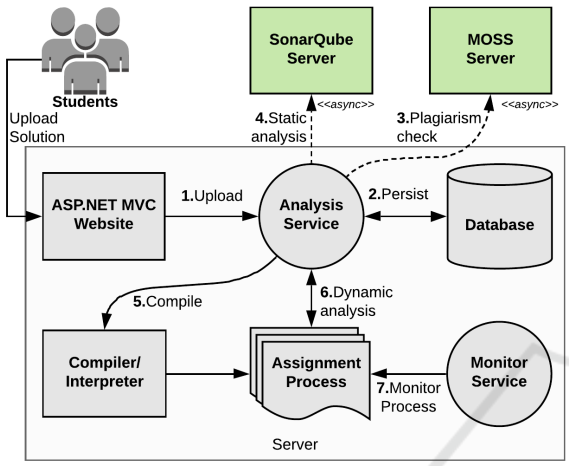

Figure 2: High level system architecture.

Technical design was driven by the requirement that the system is available over the web, so that students can access it at their convenience. The need for this tool to be usable in many courses led to framework independent compiler/interpreter calls, and external program execution, as well as a pluginable architecture. Anti-plagiarism requirements led to using the freely available MOSS server[8], that integrates code similarity detection for many programming languages. The integration of the badge-awarding system required extracting the results of source code analysis, for which SonarQube provides the required APIs. Figure 2 illustrates a high level view of the solution.

For a typical use case a student accesses the site, where they see their list of assignments and pick one they want to attempt. The site displays personalized versions of the text for each assignment based on what gamification settings are enabled for the respective course. After coding a solution they upload it. On the server side, verification and pre-processing start on the code, followed by compilation and execution, or direct interpretation depending on the programming language. We ran our semester long experiment with C++, and we also prototyped C# and Python. The next step is testing. Three tests are run in parallel, depending on server settings: correctness, coding style and plagiarism. Coding style and plagiarism tests involve asynchronous API calls to the SonarQube and MOSS servers, respectively. In our case, testing assignment correctness implies using separate threads with redirection of the application's standard input and output streams. Each assignment has a number of predefined test cases. For each of them, the server launches a new instance of the generated executable to ensure a clean run. An external service monitors and terminates runaway processes. Results are saved to the database and displayed to the student together with any badges they earned throughout the process. Figure 3 provides a simplified illustration of this process.

With regard to static analysis, we carried out an initial evaluation using the source code for several first year programming assignments. We used SonarQube's default rules for Python. Our empirical observation is that many students write methods having very high cyclomatic complexity, combined with superfluous use of branching statements. In addition, we found methods having varying return types that depend on execution code path, which leads to defects difficult to identify for beginners in dynamically typed languages.

In parallel to static analysis, the server also verifies submission correctness. Requirements for each assignment clearly specify input/output requirements. As an example, one assignment requirement was to catalogue old maps. This had to be achieved using the "`add`" and "`display`" console commands, specified to take the following form:

```
add␣mapNumber,␣state,␣type,␣yearsOfStorage
display
```

As part of one test case, the server writes to standard input "`add 1234, used, geographic, 20`", followed by "`display`". If the regular expression "`.*(1234)*.*used.*geographic.*20`" matched the standard output, the test case was considered successful.

4 CONCLUDING REMARKS

Gamification has proven to be a beneficial tactic in various areas of activity besides video games. Gamification elements applied in learning environments often provide better learning experiences, increased motivation and can trigger behavioural changes. To promote these benefits among students studying computer science, a gamification based learning system was first implemented during the spring semester of the 2018/2019 academic year.

[8]https://theory.stanford.edu/ aiken/moss/

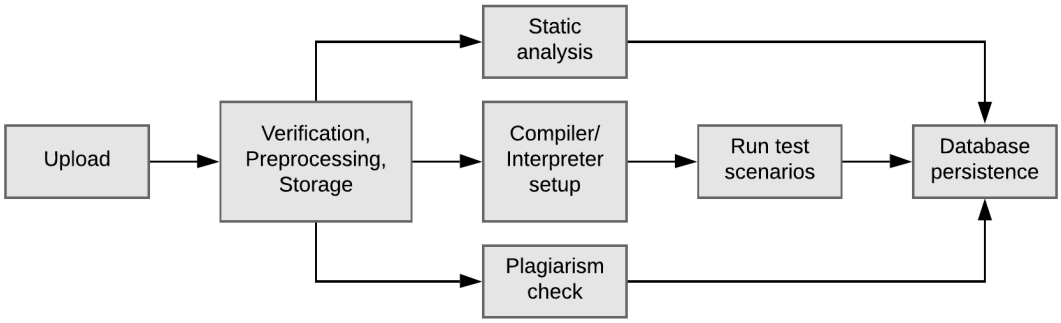

Figure 3: Solution workflow.

The current paper extends the learning system by adding two significant components. The first one is the static analysis component. We consider it essential for having an unbiased measurement of student assignment source code quality, as well as freeing up teaching staff by helping students avoid many common coding mistakes. The availability of measurements allowed us to implement the badge-driven achievement system, which represents the second innovative component of the system.

Awarded badges will belong to one of two groups: achievement, which target mastery, and style, which target minor achievements or desired behaviour. A number of badges have already been defined; however, the system allows the introduction of new badges based on the students' difficulties, challenges and ambitions. The badge awarding algorithm consists of two phases. The first phase is based on simple rules that define the conditions in which the badge and its associated reward and value can be achieved. The second phase includes progressive and meta-badges and relies on a decision tree to learn more complex awarding rules.

The static code analysis component allows our system to statically analyse submitted code and provide near real-time reports on rule violations, code smell information and accumulated technical debt. Furthermore, the results of these analyses will be integrated with the badge awarding algorithm to reward students who follow good programming practices and write clean code.

The improved system will be evaluated during the spring semester of the 2019/2020 academic year to provide an answer to our experimental question: how do badges influence student behaviour and performance? These will be analysed, taking into account the badge groups students were exposed to (or lack of exposure, for one of the groups), as well as their grades, conduct and feedback. The badge awarding algorithm will be further refined according to the output analysis. As further work is concerned, we consider applying data mining on collected feedback in order to enhance the system and its user experience. We also intend to add support for more programming platforms, as well as new gamification techniques such as quizzes, challenges and new measures for progress.

REFERENCES

Abramovich, S., Schunn, C., and Higashi, R. M. (2013). Are badges useful in education?: It depends upon the type of badge and expertise of learner. *Educational Technology Research and Development*, 61(2):217–232.

Abrams, S. S. and Walsh, S. (2014). Gamified vocabulary. *Journal of Adolescent & Adult Literacy*, 58(1):49–58.

Ajzen, I. (1991). The theory of planned behavior. *Organizational behavior and human decision processes*, 50(2):179–211.

Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. (2013). Steering user behavior with badges. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 95–106. ACM.

Barata, G., Gama, S., Jorge, J., and Gonçalves, D. (2013). Engaging engineering students with gamification. In *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, pages 1–8. IEEE.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Cavusoglu, H., Li, Z., and Huang, K.-W. (2015). Can gamification motivate volun-tary contributions?: the case of Stackoverflow Q&A community. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, pages 171–174. ACM.

De-Marcos, L., Domínguez, A., Saenz-de Navarrete, J., and Pagés, C. (2014). An empirical study comparing gamification and social networking on e-learning. *Computers & Education*, 75:82–91.

Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: Defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning*

future media environments, volume 11, pages 9 – 15. ACM.

Facey-Shaw, L., Specht, M., van Rosmalen, P., Brner, D., and Bartley-Bryan, J. (2018). Educational functions and design of badge systems: A conceptual literature review. *IEEE Transactions on Learning Technologies*, 11(4):536 – 544.

Festinger, L. (1954). A theory of social comparison processes. *Human relations*, 7(2):117–140.

Fitz-Walter, Z., Tjondronegoro, D., and Wyeth, P. (2011). Orientation passport: Using gamification to engage university students. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, OzCHI '11, pages 122 – 125, New York, NY, USA. ACM.

Fresno, J., Ortega-Arranz, H., Ortega-Arranz, A., Gonzalez-Escribano, A., and Llanos, D. R. (2018). Applying gamification in a parallel programming course. In *Gamification in Education: Breakthroughs in Research and Practice*, pages 278–302. IGI Global.

Gibson, D., Ostashewski, N., Flintoff, K., Grant, S., and Knight, E. (2015). Digital badges in education. *Education and Information Technologies*, 20(2):403 – 410.

Halavais, A. M. (2012). A genealogy of badges: Inherited meaning and monstrous moral hybrids. *Information, Communication & Society*, 15(3):354 – 373.

Hamari, J. (2013). Transforming homo economicus into homo ludens: A field experiment on gamification in a utilitarian peer-to-peer trading service. *Electronic Commerce Research and Applications*, 12(4):236 – 245. Social Commerce- Part 2.

Hilliard, K. (2013). Activision badges - the original gaming achievement (2013, october 23). Retrieved October, 2019, from Game Informer: https://www.gameinformer.com/b/features/archive/2013/10/26/activision-badges-the-original-gaming-achievement.aspx.

Huynh, D., Zuo, L., and Iida, H. (2016). Analyzing gamification of "Duolingo" with focus on its course structure. In *International Conference on Games and Learning Alliance*, pages 268–277. Springer.

Johnson, D., Deterding, S., Kuhn, K.-A., Staneva, A., Stoyanov, S., and Hides, L. (2016). Gamification for health and wellbeing: A systematic review of the literature. *Internet interventions*, 6:89–106.

Letouzey, J.-L. and Ilkiewicz, M. (2012). Managing technical debt with the sqale method. *IEEE Softw.*, 29(6):44–51.

Majuri, Jenni, K. J. and Hamari, J. (2018). Gamification of education and learning: A review of empirical literature. In *Proceedings of the 2nd International GamiFIN Conference, GamiFIN 2018*. CEUR-WS.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Raftopoulos, M., Walz, S., and Greuter, S. (2015). How enterprises play: Towards a taxonomy for enterprise gamification. In *Conference: Diversity of Play: Games–Cultures-Identities. DiGRA. Recuperado de https://goo. gl/3PD4f9*.

Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.

Ruipérez-Valiente, J. A., Muñoz-Merino, P. J., and Delgado Kloos, C. (2017). Detecting and clustering students by their gamification behavior with badges: A case study in engineering education. *International Journal of Engineering Education*, 33(2-B):816 – 830.

Sicart, M. (2008). Defining game mechanics. *Game Studies*, 8(2).

Sonar Open Community (2020). C++ plugin.

Zsigmond, I. (2019). Automation and gamification of computer science study. *Studia Universitatis Babeș-Bolyai Informatica*, 64(2):96–105.