A Blended Learning with Gamification Approach for Teaching Programming Courses in Higher Education

Goran Dambić*, Tomislav Keščec*, and Danijel Kučak*
* Algebra University College/Software Engineering, Zagreb, Croatia
goran.dambic@algebra.hr, tomislav.kescpec@algebra.hr, danijel.kucak@algebra.hr

Abstract - This paper is a result of three different needs that became apparent and unavoidable during the COVID 19 pandemic at our institution. Further exploration revealed that for each of the needs, there is a huge amount of research papers, thus validating the approach taken by this paper. First need is for blended learning. Old school approach where teacher instructs from a cathedra that worked for centuries should be replaced either completely or partially with online learning content, due to new natural media for learning – Internet. Second need is a gamification. Many research papers show that newer generations of students lose their attention and motivation for learning more quickly than previous, especially if the topic is difficult. To counteract this, many gamification efforts have been explored. Lastly, programming is difficult for students to learn, so new teaching techniques are always welcome to keep students motivated. With COVID 19 situation where live teaching becomes impossible and where many institutions were forced to migrate online, these needs became paramount very quickly. Based on said, this paper proposes a blended learning component that complements existing methods of online teaching and that also contains gamification elements. A way of transferring curriculum into actionable items of proposed system is defined, as well as a system itself. Authors hope that this approach will lead students to better understanding of programming topics, lower dropout rates and higher satisfaction in learning.

Keywords – blended learning; gamification; learn programming; programming courses; COVID 19; higher education

INTRODUCTION
In the spring of 2020, almost all higher education institutions in Croatia were forced to switch to online teaching, due to the worldwide COVID 19 pandemic. For paradigm change this big, institutions did not have years or months to prepare, but only days, maybe a couple of weeks. This change has affected students, teachers, and administration staff alike, leaving them to cope with new circumstances as best as they can.

Before this change happened, all software engineering courses at the authors’ institution were being held face-to-face in the classroom, with almost no online components. Each course was composed of lectures, held for larger groups of students in large classrooms, and lab exercises, held for smaller groups of students in specialized laboratories. Once face-to-face classes become impossible due to epidemiological measures, roughly the same course composition was transferred to online platforms.

Seemingly unrelated to this, there is a consensus among many researchers ([1], [2], [3]) that programming courses, especially introductory ones, are not easy to master because programming requires a set of specific skills: to understand abstract concepts, to learn the syntax of an artificial language and to transform real-world actions and information into appropriate programming instructions and data [1]. On top of that, a student wanting to learn to program must be a power user of an operating system and must be proficient in a set of development environments and tools. Therefore, it is no surprise that programming courses almost without exceptions experience high dropout rates [1] [4]. As these dropouts cause dissatisfaction for students and financial difficulties for institutions, numerous researches have been conducted to analyze, predict and prevent them [3], [4], [5], [7].

When the first wave of the pandemic was over and all courses were successfully transferred to online platforms, a question of the impact of a purely online approach to dropout rates has been raised. According to [6], teaching online takes more time and effort than teaching face-to-face courses and students need more discipline to succeed in online courses. The latter was confirmed by a survey taken among students of all undergraduate and graduate programs. Students said that most important online learning challenges include not having to go to the university campus, not participating in the social interaction with teachers and other students, and the high availability of distractions in the form of web-browsing resources, video-streaming services, and computer games, all available at only one click away. With computer courses being difficult by themselves and with all distractions becoming available when classes went online, it became apparent that to keep students interested and motivated and to prevent an increase in dropouts, a blended learning approach had to be designed and implemented.

As stated in [8] and [9], blended learning can be defined as the thoughtful integration of different online and face-to-face teaching methods, with five available components to choose from: face-to-face or online teacher-led, face-to-face or online collaboration, and online self-paced. This paper suggests a design and implementation of an online self-paced component for
programming courses. This would allow students to choose the time most appropriate for their learning, allow them to learn at their own desired speed, and provide them with the flexibility to learn in any location [8]. This blended learning approach would hopefully enhance students’ learning engagement and experience [10].

In the last decade, gamification has become an important technique in many areas [11]. For example, studies have shown that applying gamification elements in websites engaged users [13]. It comes as no surprise that gamification can be effective in education and especially in programming courses [11], [12]. At the same time that authors of this work decided to envision a blended learning component for online self-paced learning, the decision was made that this component should have gamification components built into it. This was done with the purpose to engage students more, to present the learning material in a more approachable way, and to allow students to increase their motivation and interest, hopefully resulting in better grades, success rate, and lower dropout rate.

**METHODOLOGY**

This paper describes the approach taken to design and implement a self-paced learning component for blended learning. In addition, this paper describes the rationale behind converting the learning content of one programming course to content appropriate for use in this new learning component. To create this new component, the following requirements were defined:

- New component must be context agnostic – it must be capable of supporting many courses, both in programming area and other.
- New component must be available to students in modern formats they are accustomed to, mainly in mobile interfaces.
- New component must organize learning in small chunks, appropriate for learning experience lasting no more than several minutes.
- New component must use gamification elements.

The fact that new component must be context agnostic comes from the facts that blended learning approach was necessary for all courses in software engineering study program, both programming and other (computer architecture, physics, mathematics, computer networks, etc.) and that creating specialized components for each course type would be impractical and unrealistic.

As mobile phones become a part of every aspect of our lives, students prefer to consume their learning content via mobile interfaces [14]. With this important fact in mind, the proposed system was built with a mobile-first approach. The choice was made not to create native mobile applications, but to create a regular web application with a user interface and user experience designed first for mobile resolutions and then for regular screen sizes. This way, both students using mobile phones and those using laptops were satisfied and were able to enjoy learning content in their favorite interface.

When shaping learning content into consumable units, it is important to take into account the duration of each unit. Although many pieces of research have proven that there is no fixed duration after which students’ attention starts to decline ([15], [16]), the decision has been made to keep the duration of each unit under 10 minutes, based on rich prior experience in e-learning for adults. When a certain topic was too complex or too broad to fit in 10 minutes, it was split into two or more shorter units. The goal was to produce short and easy-to-consume chunks that students can consume without losing attention and in-between other activities.

As suggested in [17] and [18], when gamification consumers perceive gamification elements as informational, it may give them feelings of competence and achievement and hence it might enhance intrinsic motivation and promote performance gains. Gamification elements most cited are points, levels, and leaderboards, so all of these elements must be included in the blended learning component proposed in this paper.

A. Design and the implementation of the blended learning self-paced component

To satisfy the aforementioned requirements, a proposed system was designed as a web application with a mobile-first user interface and experiences. The system uses national AAI@EduHr infrastructure [19] for user authentication and allows only users enrolled in the specified institution to sign in. After they have signed in, users can see a list of available courses, a list of available topics with their progress for each of the courses, and a leaderboard where other students’ progress from the same course is shown. Fig. 1 shows a part of the list of available topics for the Data Structures and algorithms course, together with user progress (current level and amount of units required to advance to the next level).

Once a topic is selected, the next self-paced lesson is started (Fig. 2). Each lesson has multiple items and each item can either give some information to the student or can require a student to solve a problem. Once all items in a lesson have been completed, a lesson is considered completed and the student is awarded points. The student is taken back to a list of topics that immediately reflects newly accumulated points and progress. If student switches to a leaderboard, her or his points are used to determine a new position among other students.

Figure 1. A screenshot of a Data Structures and Algorithms course with top three topics and user current level and completion percentage
B. Converting existing learning content for blended learning self-paced component

Once the workflow of the new system was finished, it became necessary to define a way to fit the existing learning content of a certain course in this workflow. The first step was to take each learning outcome and break it into smaller parts called topics, with each topic roughly equal to the content of one week worth of lectures and lab exercises (for example, one defined topic was called “Binary trees basics”, other was “Heaps and priority queues”, etc.) This step defined the exact amount of learning material that needed further decomposition. That decomposition was done by defining multiple levels for each topic: the student starts each topic with level 1 and progresses to level 5. Once level 5 is achieved, that topic is considered completed. After completion, the student can choose to practice that topic more. In each practice run of the completed topic, the student gets 10 random questions from the topic. The number of repetitions is not limited.

To progress from level $n$ to level $n + 1$, the student has to complete a variable amount of lessons, with current percentages of completed lessons at the current level also shown to the student for gamification purposes. Each lesson is made from multiple items. Each item can either give some smaller amount of information to the student or require the student to solve a problem. When solving a problem, a problem statement is shown (with possibly one or more images) and the student is expected to answer a question. The way to answer the question depends on the question type.

In the first question type, the student is presented with two or more possible answers and the student must choose the correct one by inserting it into the slot (only one answer is correct). This question type is most suitable for code completion types of questions, for example:

**Question:** Complete the following implementation of the destructor in .cpp file:

```cpp
[empty slot] Osoba();
```

**Answers:** Osoba::~ (correct answer), Osoba::, ~Osoba::, Osoba:::

In the third question type, the student is presented with a problem statement that has multiple empty slots. The student is also presented with two or more answers: some of them must be used in correct empty slots, and some of them should not be used at all. This question type is most suitable for larger code completion types of questions, for example:

**Question:** Implement an include guard for .h file:

```cpp
[empty slot 1]
[empty slot 2]
class MojaKlasa { _; }
[empty slot 3]
```

**Answers:**

```cpp
#ifdef _MOJAKLASA_H_ (use in slot 1)
define _MOJAKLASA_H_ (use in slot 2)
#endif (use in slot 3)
#ifdef _MOJAKLASA_H_ (not used)
#undef _MOJAKLASA_H_ (not used)
#else (not used)
```

One important aspect of this proposed method of converting existing learning content for blended learning self-paced component is to determine key messages that need to be delivered to students and then to use different types of questions to deliver messages from different angles to make sure that students understand them.

**RESULTS**

The proposed system was built and ready to use right before the start of the summer semester in 2021, with the intention to test its behavior and acceptance with one course, that being Data structures and algorithms. The transformation of existing learning content for the blended learning self-paced component was done for the half part of the semester successfully and everything was ready for the students to engage and use the system.

For the first half of the semester, a set of seven topics was defined. In these topics, there were 35 levels defined with 65 lessons. For lessons, 321 items were created. For items that were defined as questions, 1,340 correct and wrong answers were provided. Table 1 shows the distribution of questions items per question type:
### Table 1. Number of Created Questions per Question Type

<table>
<thead>
<tr>
<th>Question type</th>
<th>Number of questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>219</td>
</tr>
<tr>
<td>Type 2</td>
<td>28</td>
</tr>
<tr>
<td>Type 3</td>
<td>74</td>
</tr>
</tbody>
</table>

Out of 157 enrolled students, until half of the semester, 54 of them have participated in the new self-paced component. Fig. 3 shows students’ activity expressed in the number of completed lessons in each week, starting from week 11 of 2021 (first week of the semester), and all the way to week 19 (beginning of midterm exams at the half of the semester).

Fig. 4 shows students’ activity per hour in the day (for example, the x value of 13 represents the time between 13:00 and 13:59).

Out of 157 enrolled students, 46 of them were taking the class second or third time. Out of 872 activities in total, 330 activities belonged to such students. Their distribution per week is shown in Fig. 5.

### DISCUSSION

Usage statistics show that new component available through the mobile interface is used at least once by 54 out of 157 students, which is 34%. This relatively low number might be explained by the following factors:

- There was no promotion of the new system other than telling students about it in the lectures.
- Usage of the system was completely optional and did not contribute any points to the final grade or passing the course.
- The system was in its early beta version so some students might be reluctant to engage in a system that has yet to reach full maturity.

Fig. 3 displays a huge amount of activities at the beginning, which might be attributed to the new system that students haven’t seen before. After this initial enthusiasm, activities dropped to more realistic levels and remained there right up to the midterms. The reason for that might be students’ bigger engagement in preparation for midterm exams, but to confirm this assumption, another research should be done with values from the whole semester. If there were to be found an increase in the activities after the midterms, the assumption would be confirmed.

Another interesting observation can be made: although students taking the class second or third time make 29% of the student population enrolled in the course, their share of all activities is 38%. This might be explained by the fact that these students are familiar with the course difficulty and are ready to invest more time to complete it successfully this time. If this insight might be shared with students taking the course for the first time, hopefully, it would yield a better passing rate. Also, this shows that investing more efforts in advertising the system to such students might be beneficial for their passing the course.

When students’ activities per hour in the day are looked at, it is clear that many activities take place in hours that are not usually considered regular studying hours. This proves that the online availability of this system allows each student to learn at her or his own pace and that further improvements of this approach are the correct way to go.

The results of this paper show that students have accepted the new system and are willing to use it. By including the system in official course activities, the number of users might significantly increase. Although the fact that students like and are willing to use the system is encouraging, it does not tell anything about the system’s effectiveness and usefulness in helping students achieve better grades, in increasing the passing rate, or in lowering total dropout rates. One very empirical way of exploring the system’s effects might be to analyze the success of a generation of students without the system with the success of a generation of students that have used the system. Another thing that has not been answered in this paper is the effect of gamification on the system’s adoption. Another research might be done where one group of students have the same system without visual gamification elements (levels, points, and leaderboard) and compare...
their activities and results against another group that has. Further improvements might also lie in the direction of adding more question types and adding multimedia elements like short video lectures.

REFERENCES


[19] AAI@EduHr, Autentifikacijska i autorizacijska infrastruktura sustava znanosti i visokog obrazovanja u Republici Hrvatskoj, URL: www.aaiedu.hr, 2011.