Effects of Competitive Coding Games on Novice Programmers

Fischer, Konrad *
konrad.fischer@campus.lmu.de

Vaupel, Sarah *
vaupel.sarah@campus.lmu.de

* *Ludwig-Maximilians-Universität München*
Munich, Germany

Abstract

Technology Enhanced Learning (TEL) can be used to improve the learning experience of aspiring programmers. Additionally, gamification elements can be used to increase students' motivation. This research paper focuses on a tool utilizing both TEL and gamification elements: A coding platform that provides challenges for users to solve in order to improve their programming skills. Well-known coding platforms include *Codewars* and *CodinGame*, the usage of which had a positive effect on undergraduate students' motivation [1]. Inspired by the aforementioned approaches, this research paper describes a coding platform with more focus on competition and its evaluation in the context of tertiary programming education.

On most coding platforms, users solve coding challenges in a playful manner, but have no way to interact with each other while solving a programming challenge. Adding the ability to play against another user in real-time could further enhance a user's learning experience and motivation. In this research paper, we seek to answer the question whether this notion of competition can increase motivation to solve programming tasks.

We implemented the coding platform *SuperDevBros* to investigate this question. The main purpose of this platform is to provide one-on-one programming challenges for two users to compete against each other in real-time.

Two user studies were conducted in order to evaluate the effects of real-time competition on users' self-assessment, enjoyment, and motivation. In addition to subjective feedback from questionnaires, the participants' objective performance while using the platform was recorded.

After the user studies were completed, data was gathered and evaluated from the users' interactions with the system and their subjective feedback. Some of this data was collected at multiple phases of the study to observe changes in the participants' attitudes triggered by the competitive mode.

Of particular interest was the comparison of the participants' objective performance and their perceptions, i.e. regarding their self-assessment, enjoyment and motivation.

The results of this study suggest that this kind of competitive environment in programming leads to learners assessing themselves more distinctively, while their enjoyment decreases. Furthermore, learners' motivation drops in competition with others, but remains on a high level. Learners who already achieved a relatively high level of programming knowledge are generally more inclined to enjoy competitive gamification components in programming.

Despite of the limited generalizability of the results, we conclude that competition in programming education does not necessarily have solely positive effects on learners' experiences. The inclusion of competition in tertiary computer science education might prove to be useful, but further research needs to be conducted to identify the best way of integrating it into educational concepts.

Contents

1 Introduction

This article touches on the subject of coding platforms, a tool which utilizes elements of *Technology Enhanced Learning* and *gamification* to enhance the learning experience of aspiring programmers. We contribute to this topic by introducing a coding platform with additional focus on competition, and its evaluation in the context of tertiary programming education.

Research has found indications that motivation is beneficial for learning [14, 18]. Gamification elements can also contribute positively to the students' learning results and experience [16], thus increasing the chances of successfully mastering a new field. Gamification "refers to the use of design elements characteristic for games in non-game contexts" (abbreviated from [15, p.5]).

Technology Enhanced Learning (TEL) describes the application of technology to enhance learning and teaching. One possible application is the use of technology to provide additional motivation during the learning process of students. This might include adding gamification elements to the learning experience, or providing automated constructive feedback to the users of a learning system.
An example of an online platform that utilizes TEL is *Codewars*[1]. In *Codewars*, "players" solve small programming exercises; they are given a task and develop a solution. They can test their solution against a set of unit tests until their code satisfies all given tests.

However, on *Codewars* players do not interact with each other while solving challenges. Such interaction might be desirable though, as it might motivate users to give their best to find solutions for the tasks they are presented with. Real-time competition could be a particularly effective form of such interaction, as it gives the player immediate feedback on how their performance compares to that of others.
However, users may respond to competitive environments in various ways. Some users might feel challenged by this setting, which might raise their willingness to learn. On the other hand, others might feel stressed or even scared of failure, thus decreasing their motivation.

In this research paper, we seek to answer the question whether the notion of competition is able to increase motivation while solving programming challenges. Inspired by *Codewars*, we have implemented the speed coding platform *SuperDevBros*. The main purpose of our platform is to provide programming challenges that can be solved by two users at the same time, where the user who solves the challenge first "wins". To let the users experience direct competition against each other, both users share the same interface, where a user can see the opponent's progression. An additional game element was employed by adding "jokers", which can lead to a temporary advantage over the opponent.

We conducted a user study using our platform, where the participants were presented with both a singleplayer and a multiplayer mode of a set of small programming challenges. Our goal was to evaluate to what extend the addition of competitive game elements motivated users to solve code challenges. The evaluation of the study showed that, against expectations, the introduction of a competitive component did not have a clear positive effect on the users' self-assessment, enjoyment or motivation. The results suggest that the users' self-assessment shifted towards extremes when being confronted with a competitive environment, while the users' enjoyment seems to follow an opposite trend. Furthermore, the motivation of the users dropped in competition with others, but nevertheless remained on a relatively high level. This leads to the conclusion that competition in programming education does not necessarily have solely positive effects on leaners' experiences.

This research paper summarizes our approach on competitive gameplay in a live coding platform

[1] `https://www.codewars.com/`

and the users' experiences with it.
In Section 2, we first give an overview over research articles that motivated our approach. We also describe the main features of *Codewars* and related platforms, which served us as reference platforms for *SuperDevBros*. An in-depth description of our platform, its features and technical details are provided in Section 3. In Section 4, we describe the user studies we conducted. We then present the results and our interpretation of them in Section 4. To conclude, we summarize our findings and give an overview of features we envision as future work in Section 5.

2 Related Work

This section discusses related work. First, previous research on motivation in programming education is described. Second, an overview of conducted research on gamification in education is given. This is followed by a report on similar approaches to coding platforms.

2.1 Previous Research on Motivation in Programming Education

Numerous publications form the previous research conducted on motivation in programming education. Two publications are discussed in greater detail: Tony Jenkins' thesis because it provides an in-depth qualitative analysis of students' motivation, and Law's, Lee's and Yu's study because they investigate the effects of an e-learning system of motivation.

2.1.1 Tony Jenkins: The Motivation of Students of Programming (2001)

Jenkins investigated in their master thesis [20] the motivation of students during their programming courses.

A first study group of potentially over 400 students at two universities was surveyed at three important points in their programming courses, mainly focusing on their reasons why they chose to study an IT degree and on the development of their attitudes towards their degree course (and about the programming in particular) throughout their first year. This survey consisted of three questionnaires that the study group answered before, halfway through and at the end of one of their programming modules. All of the three surveys included questions about the students' attitude towards their studies, where they were asked to identify a main motivator among a set of motivators. The most frequently given answer to the students' attitude was their "own satisfaction", which was the most *intrinsic* (i.e. deriving from interest in the subject) motivator given. The percentage of students who reported their own satisfaction as their main motivator rose from 48.98 % at the beginning up to 55.73 % halfway through, and then decreased to 48.46 % at the end of the module. The second most frequent answer was the students' will to "get a good job", although the relative frequency fell from 47.81 % at the beginning down to 37.02 % halfway through the module, and afterwards rose slightly to 38.94 % after the module. Getting a good job was the most *extrinsic* (i.e. reward-driven) motivator of the set of motivators given in the questionnaires. A noteworthy fact is that the number of students who reported a so-called *null* motivation ("I just want to pass") rose from almost zero (0.29 %) at the beginning to 4.20 % halfway through the module, and even increased further to 5.77 %. This possibly corresponds to a group of students becoming increasingly disillusioned with their course and no longer having any positive motivation to succeed.

A second group was studied on a weekly basis through their first programming course at one university, focusing on their aspirations, struggles and experiences. The programming course consisted only of coursework, with no formal examination but four assessed programming exercises that the students had to work on alone. This study was more focused on individuals than the first study, investigating the development of selected students that form two subgroups: students with prior experience in programming, and novices. 9 experienced programmers and 15 novice programmers were interviewed.
The only common experience that the author of the study was able to gather from students with

prior experience is that they all passed the course in the end. Reported individual experiences in this subgroup were highly mixed, ranging from perceiving the course as highly enjoyable to hardly bearing to remember it. The language used by some experienced participants of the course was emotive, with some expressing strong negative emotions connected to their experience with learning programming. However, all interviewed students with prior experience seemed to look back on the course with a constant theme of achievement and challenge, even ones who reportedly struggled with learning programming.
Among the subgroup of novices who participated in the same programming course, the majority stated to have had a largely negative experience, with few students looking forward to their second semester's programming in a positive way. Just as the experienced group, all of the novices interviewed passed the course. A recurring theme among the novice subgroup was initial enthusiasm, followed by its gradual erosion until the course was perceived as wholly negative at the end of the semester. Apart from three students (a fifth of the subgroup of 15 novices) who reported more positive experiences, most of the novices expressed their negative experiences with learning programming in an emotive way, similar to the subgroup of students with prior experience.
Another noteworthy point is that Jenkins reported a large impact of assessments and coursework on the students' experience with the programming course. The students seemingly felt "driven by the assessment, making it appear to them to *be* the module rather than simply one aspect of it". They also reported an immense workload due to the assessments, and felt that their work was not adequately reflected in their results, which had a negative effect on their motivation. [20]

2.1.2 Kris M.Y. Law, Victor C.S. Lee, Y.T. Yu: Learning Motivation in E-Learning Facilitated Computer Programming Courses (2010)

Law, Lee, and Yu conducted a study in [21] on factors influencing students' motivation to learn programming. Their study included the evaluation of the effects of an e-learning system on motivation. They concluded that their e-learning system *PASS* was helping to increase students' efficacy, which they define as "what a person believes he or she can do in a particular learning task" [21, p.220]. The authors' system uses test cases provided by the teacher of a course to examine students' solutions for such a task. The tests were executed upon submission of a solution by a student and the student was shown the results. According to the authors, some of the advantages of their system are that there is no time spent waiting for a human to review students' solutions and that students can work on problems from anywhere. Within the six factors of motivation the authors investigated "'individual attitude and expectation', 'clear direction', and 'reward and recognition'" [21, p.226] were affecting students' motivation the most. They also discovered that the factors "'individual attitude and expectation', 'challenging goals', and 'social pressure and competition'" [21, p.226] were positively influencing efficacy in a significant way. Another insight gained in the study was that a specific way of teaching, which included the teacher evaluating solutions in real-time and presenting good programs that students had submitted during the class to the audience, led to participants feeling a small amount of "peer pressure and competition" [21, p.227]. Students wanted to submit good solutions that would then be shown to their peers which represents an acknowledgement of their work. The authors highlighted that this way of conducting a course is enabled by their system *PASS*. One possible extension of their system the authors considered is a real-time component that shows students the advance of their peers towards a solution for a task. With the addition of such a component, the authors hope to add "an element of 'peer pressure and competition'" [21, p.227] which could make students more keen to surpass other students. Another extension of their system Law, Lee, and Yu considered is to allow students to add test cases to the system which it could then use to check other students' solutions.

2.2 Previous Research on Gamification in Education

In a systematic mapping study in [16] conducted by Dicheva, Dichev, Agre and Angelova, the authors investigate to which educational contexts gamification has been applied to, and which

game elements have been used to gamify educational systems. In the aforementioned study, the authors systematically reviewed existing work on gamification in education, with the intention of mapping out topics rather than synthesizing study results. 34 papers presenting empirical studies from major scientific databases were analyzed and classified along the following dimensions:

- *Gamification Design Principles.* The classification of previous articles showed that the most widely used gamification design principles in education were visual status, social engagement (e.g. individual and team competitions), freedom of choice (e.g. what type of challenges to complete, choice of specific challenges to complete, controlling the order and/or speed of completing the challenges), freedom to fail (e.g. allowing students to revise and re-submit assignments), and rapid feedback.
- *Game Mechanics.* Dicheva et al. discovered that the most popular game mechanics used in education were points, badges, and leaderboards.
- *Context: Type of Application.* Comparing the contexts where gamification had been previously applied showed that the majority of reported case studies were on gamification of blended learning courses.
- *Context: Education Level.* According to the results of [16], two out of 34 analyzed papers considered gamification for the K12 education, while the remaining 32 papers targeted higher education and training.
- *Context: Academic Subject.* A majority of the 34 papers focused on the gamification of Computer Science or IT courses.
- *Implementation.* The implementation of gamification in previous research reportedly varied, but the authors were able to identify five categories:
  - No e-learning platform or other software used, e.g. gamification only through teacher efforts and a leaderboard.
  - Manually collecting data on student performance and processing it with a computer program.
  - Supporting gamification elements in existing Learning Management Systems (LMS) in the form of a plugin.
  - Third party software to support some aspect of gamification.
  - Software for supporting gamification implemented as standalone applications.
- *Reported Results.* The majority of the 34 papers reported encouraging results of the experiments, such as significantly higher engagement of students, increased attendance, participation, and material downloads, positive effects on the quantity of students' contributions/answers (without a corresponding reduction in their quality), and increased passing ratios and participation in voluntary activities.

Dicheva et al. concluded their study with the observation that many previous publications on gamification had merely described some game mechanisms and dynamics and had re-iterated their possible use in educational context, while true empirical research on the effectiveness of incorporating game elements in learning environments was reportedly still scarce. They added to their review that most previously conducted empirical studies had not included a proper evaluation, and that the true effect of employing game elements in learning environments remained to be demonstrated in practice. [16]

2.3 Similar Approaches to Coding Platforms

This section reports similar approaches to coding platforms, mainly *Codewars* and *CodinGame*. Both acted as inspiration for the coding platform which is evaluated in this article.

2.3.1 Codewars

Codewars[2] is a platform which allows users to train their programming skills using challenges from a number of programming languages. Learners use an editor in their browser to craft a solution to a challenge. These challenges are created by other users and then reviewed by the community before they are made available to all players [7]. *Codewars* validates the user's solution to a challenge using tests that are executed on their server [4]. One difference between *Codewars* and our platform is that *Codewars*' challenges always have two sets of unit tests. The user can execute tests from the first set at any time to check whether they are on the right track to solving the challenge, and they will be shown the result of those tests. The second set of tests will only be executed when the user attempts to submit the challenge and they will not be shown the output of these tests [5]. After the user has completed a challenge on *Codewars*, they will be shown the solutions other users have come up with, which allows them to learn new ways of solving tasks that they themselves did not think of [4]. Users can upvote the solutions of other users regarding "best practices" and "cleverness" [8]. Each challenge also has a dedicated web page where users can discuss it's content and seek help if they are having trouble solving it [4]. Furthermore, *Codewars* includes a ranking system [6, 4]. A user's rank is displayed on their profile. Apart from that, *Codewars* also has a separate honor system which is also partly based on the users skill level but mostly rewards contributions to the community like creating challenges [4, 6]. Honor ranks unlock new privileges for users [9].

2.3.2 CodinGame

CodinGame[3] is another platform offering coding challenges for users to improve their programming skills. Like *Codewars*, *CodinGame* offers an interface the user can use from any browser and uses a set of tests to verify that the user's solution is correct [17, 12]. Like on *Codewars*, users can discuss individual challenges on the *CodinGame* website and view other players' solutions after solving a challenge. Solutions can be discussed and voted on, and there is a user level displayed on their profile which the user can increase by solving challenges [10, 13]. One unique feature of *CodinGame* is that all challenges are related to games in some way. Precisely, there is a simple visual representation of a game in the browser based interface. The challenge the user solves represents a part of the logic of that game and each time the user tests their solution, the game is executed with the logic built by the user and therefore gives them visual feedback [1, 17]. Using this integration of games, *CodinGame* hopes to improve the users' learning experience [17]. Another feature of *CodinGame* is that the platform includes a multiplayer mode called "Clash of Code". In this mode, the user faces off against multiple opponents in matches of 5 to 15 minutes. There are multiple types of these matches. One requires the user to be the first to submit code that passes a set of tests, which is the same principle our platform uses. Another type rewards the user finding the shortest solution that passes all tests, and in the third type of matches there is no challenge description given but only a number of test cases and the players have to guess what their task is [2]. Apart from developers who want to improve their programming skills, *CodinGame*'s second target group are companies that can utilize the platform's challenges to screen applicants to jobs and use the multiplayer coding battles for team-internal coding competitions [11].

Butt conducted a study in [1] on game-based learning with *Condingame*. The participants were undergraduate students that had already studied software engineering or computing for one term or longer. They were asked to work on 3 - 5 of *Codingame*'s challenges for 90 minutes and afterwards their opinions were collected. The results of the study showed that most of the subjets were interested in getting to know game-based learning and had a positive opinion on getting more chances to use such an approach in the future. Not as many students believed that game-based learning could replace conventional learning techniques. The authors suggest combining game-based learning with traditional teaching methods might result in a positive learning environment

[2] `https://www.codewars.com/`

[3] `https://www.codingame.com/`

for the students. [1]

2.3.3 Other Platforms

While *CodinGame* and *Codewars* are in our opinion the most well-known platforms that allow users to practice their coding skills using challenges, there are a number of other platforms that offer this or a similar service, some of which are mentioned here. *Coderbyte*[4] and *CodeAbbey*[5] provide challenges for the users to solve using an IDE in their browser similar to *Codewars*, and also allow for discussion of the solutions of a challenge [3, 23]. *Exercism*[6] also provides challenges for the users to solve but no environment in the browser to do so. After users solve a task, they upload the solutions and can then discuss them with a mentor provided by *Exercism* [19]. *Topcoder*[7] allows companies to publish projects on the platform; programmers then construct solutions for the companies. The best solution is financially rewarded [24].

3 Implementation

Inspired by the previously described approaches of existing coding platforms such as *Codewars* or *CodinGame*, and in order to put the claims of previous research to the test, we developed a coding platform that adds a notion of competition to the solving of programming tasks. On this platform, the users can not only "play" challenges for themselves, but also directly compete against other users in the form of a one-on-one multiplayer mode.

In this section, we present the implementation of this platform and its main features. We begin with the *Lobby Page*, the landing page of the platform. Afterwards, we give a detailed overview over the *Game Page*, the main component. This section concludes with an explanation of a feature which enables users to submit and review new challenges.

3.1 The Lobby Page: Creating and Joining Game Sessions

To access the main app, one has to first create an account. After login, the user will be redirected to the Lobby Page, which can be seen in Figure 1.

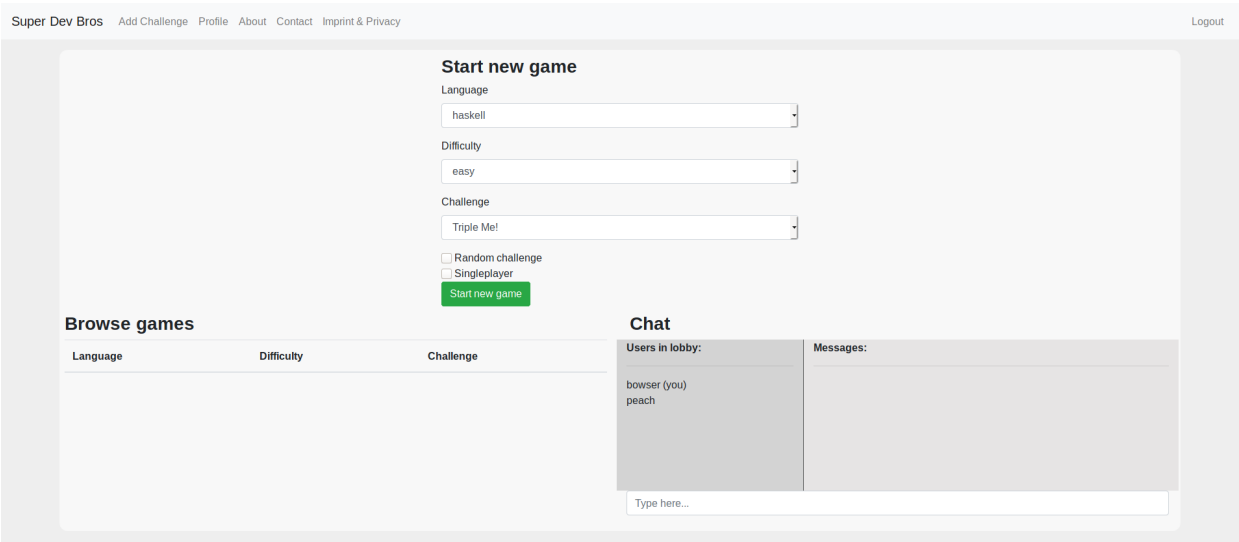


Figure 1: *The Lobby Page.*

[4] https://coderbyte.com/
[5] https://www.codeabbey.com/
[6] https://exercism.io/
[7] https://www.topcoder.com/

From the Lobby Page, users can create and join game sessions. A new game session can be created using the uppermost panel on this page. Using this interface, one can select the desired programming language and difficulty level of the challenge. It is distinguished between easy, medium and hard challenges (the difficulty level of a challenge is chosen by the challenge creator, who decides upon the complexity of a task for the respective programming language). The user can also choose to select a particular challenge from the list of available challenges for this language and difficulty. Optionally, the user can tick a box to be presented with a random challenge with the selected language and difficulty.
One can also choose whether to play the challenge in singleplayer or in multiplayer mode. If the singleplayer mode is selected, the user will start the game immediately. If the multiplayer mode is chosen, the user (host) will then be directed to a loading screen until a second user (guest) joins the newly created game session.

The second half of the Lobby Page consists of an interface for joining existing game sessions and a minimalistic chat interface. Every open game session that has a hosting user and no second user (guest) is listed together with information about its selected language, difficulty and (possibly) the challenge title. If a user decides to join a session, the host user and the guest user will be directed to the Game Page.

3.2 The Game Page: Playing Games

After a game has started, the player will be presented with the Game Page, of which the multiplayer version can be seen in Figure 2. The most important components of the Game Page are

- the two code editors
- the challenge description tab
- the test output tab
- the button for submitting the current solution for testing
- the joker buttons (blue)
- the joker status bar (above the two editors)

As shown in Figure 2, for multiplayer games there will be two code editors beside each other. The user "bowser" serves as our default opponent in this paper. Above the two editors are the challenge description and test output tabs, as well as a simple log that shows a list of what has happened in the game.
If the current game is a singleplayer game, there will only be one code editor in the bottom row; the one of the opponent, and all components related to jokers will be absent.

3.2.1 The Code Editors

The player's code editor, as seen in Figure 2 in the bottom left and enlarged in Subfigure 3a, is a conventional code editor which includes syntax highlighting for the programming languages supported by the system. The two buttons below the editor allow the player to submit their code for testing, and to reset the content of the editor to the initial state when the game was started. The player can submit their code as often as they like and will get feedback either in the form of the test outputs if they fail, or in the form of the Game Statistics Page (see Section 3.2.4) if they win the game by submitting a correct solution.

The opponent's code editor, as seen in Figure 2 in the bottom right and enlarged in Subfigure 3b, shows an obfuscated version of the opponent's code with each character replaced by special characters. The username of the opponent is shown above the editor. This editor is synchronized so that it always displays an approximation of the current length of the opponent's code. This emphasizes the real-time character of the system by showing the player their opponent's current and

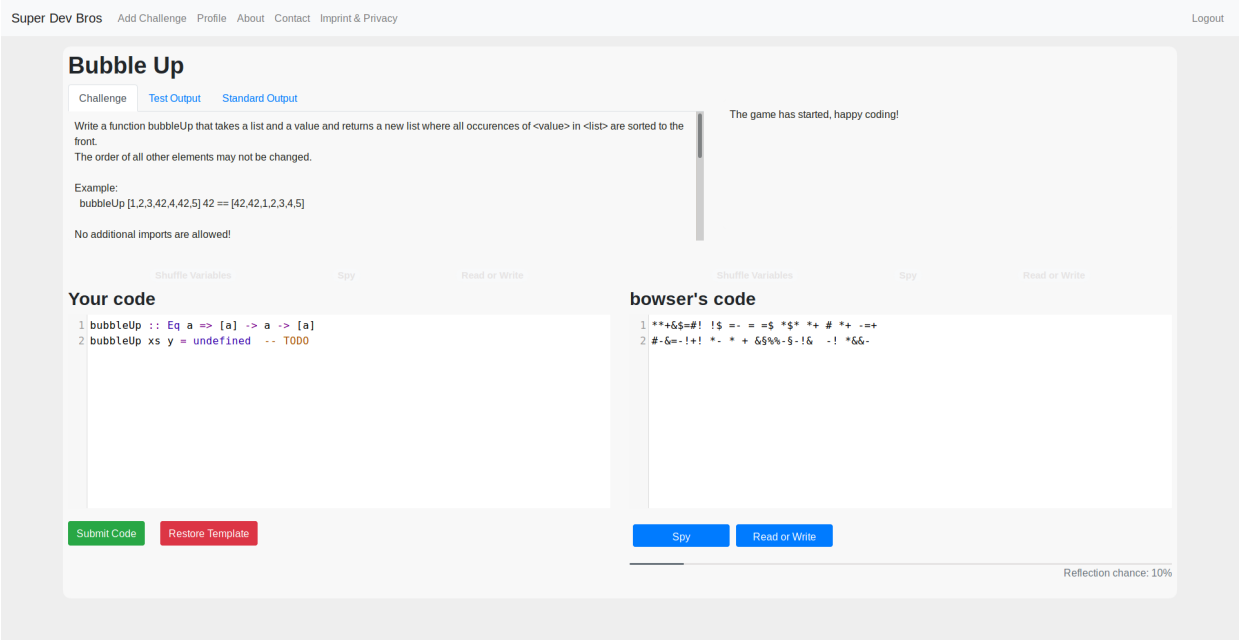
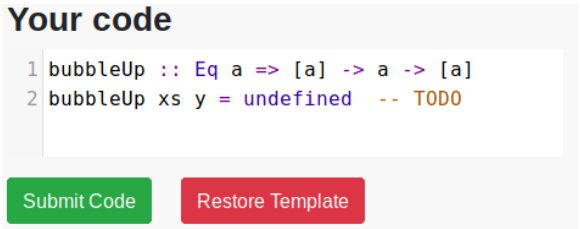
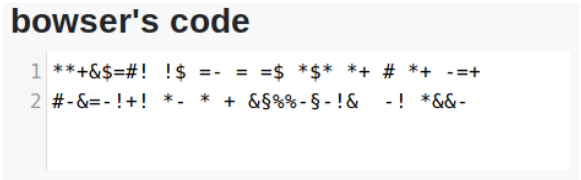


Figure 2: *The Game Page, with one's own code editor on the left and the opponent's editor on the right side.*



(a) *The player's code editor, with buttons for submitting the current code for testing and to restore the default template (to start again).*



(b) *The opponent's code editor, where the code of the opponent is visible, but replaced with random characters unless the spy joker is active.*

Figure 3: *A more detailed view on the code editors from Figure 2.*

continuously updated progress. The user can temporarily gain access to an un-obfuscated version of the opponent's code using the spy joker, which will be explained in more detail in Section 3.2.2.

3.2.2 The Joker System

The joker system was added to the platform to emphasize the game character of the application and to increase the enjoyment that users get while solving challenges. It allows the players to gain a temporary advantage by either hampering the opponent's progress or by accessing additional information. The system includes three kinds of joker:

- *Shuffle Variables Joker*: Upon activation, this joker will swap the identifiers of a number of randomly selected variables in the opponents' code, thereby making it harder for the other player to read their own code and continue working on it. This does not change the functionality of the opponent's code and will not introduce syntactical or semantical errors. (This joker type is currently only available for *JavaScript*.)
- *Spy Joker*: While this joker is active, the code in the opponent's editor is temporarily visible in un-obscured form. Therefore, the player will be able to gain additional information about their opponent's progress and potentially about a possible solution for the challenge.

- *Read or Write Joker*: The effect of this joker is that the text color of the editor of a player's opponent is temporarily changed to the same color as the background. Therefore the opponent can either read their code (by selecting it), or continue working on it but not both at the same time while this joker is active.

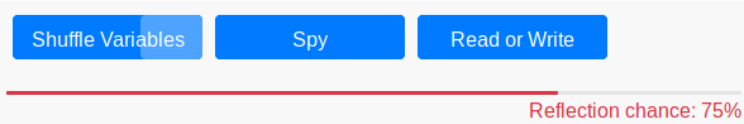


Figure 4: *The joker interface.*

The user can trigger a joker using the joker buttons (in the bottom right corner of Figure 2 and enlarged in Figure 4). After a joker has been triggered, its use will be blocked for a certain amount of time, indicated by the button's color being gradually "refilled", representing a loading bar. The joker is available again when the color of the whole button is restored to dark blue. When a user trigger a joker, there is a chance that the joker is not applied on the opponent but instead affects the player themselves. This "reflection chance" is shown to the user with the help of the colored bar below the joker buttons. The reflection mechanism was introduced to make it less appealing to the players to trigger jokers excessively, thereby annoying the opponent instead of productively working on the task. Another aspect was to introduce a small tactical component to the platform and thereby making it feel more like a game instead of a learning system.

3.2.3 Submitting Code for Testing

When players click the button "Submit Code" in the lower left of their own code editor (as can be seen in Figure 3a), their code is sent to the server to be evaluated. The server will execute a set of unit tests, which were specified when the challenge was created, on the player's code. It uses the external software *Coconut 2* for test execution, which was developed in the context of a bachelor thesis at the *Research and Teaching Unit for Programming and Modelling Languages* of the department of informatics at the *LMU Munich* [22]. *Coconut 2* provides an application program interface (API) that allows other applications to submit code in various languages for execution and then get back the results. For our purpose, we slightly modified *Coconut 2* to be able to run tests for *JavaScript* and *Haskell* code. The testing frameworks we used on *Coconut 2* were *Jest*[8] (for *JavaScript*) and *HUnit*[9] (for *Haskell*).

Upon completion of the tests, the server will get the results back from the *Coconut 2* instance and send messages to the clients depending on the results. If all tests passed, the winner is informed about their victory, and the opponent is informed about their defeat. If some of the tests failed, the respective player will be shown the output of the tests in the tab "Test Output" (as seen below the challenge title in Figure 2) and will therefore receive a hint on which parts of their program they have to improve.

Any user can submit code for testing as often as they wish. Furthermore, no negative consequences arise from a submission that did not pass all unit tests.

3.2.4 Winning a Game: Game Statistics

Upon the regular end of a game (a game that ended as a result of one player's code passing all tests), both the winner and the loser are redirected to a page showing statistics on the match. Game statistics are shown in the form of diagrams, including the frequency of keystrokes and the number and type of jokers played by both players. Additionally, the page also shows both players the state of their code at the time one of them won the challenge. The winner is also shown the

[8] `https://jestjs.io/`
[9] `http://hackage.haskell.org/package/HUnit`

code of the loser, to be able to compare it with their own, but the loser is not shown the winner's code so that they can compete in another match with the same challenge without already knowing a correct solution to the task. Not showing the loser the winner's code could increase their desire to play the same challenge again, in order to solve it and access the solution(s).

3.2.5 Viewing Solutions to a Challenge

If a user has won a particular challenge at least once, they have gained access to the Solution Page of this challenge. The contents of this page can be seen in Figure 5.

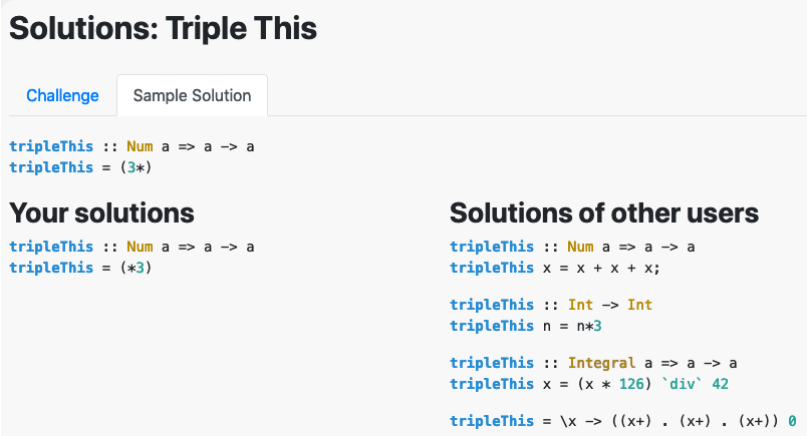


Figure 5: *The Solution Page.*

The Solution Page shows, besides basic information such as title and description of the challenge, all solutions that have been saved in the database. This includes the sample solution (i.e. the solution the author of the challenge submitted upon creation of the challenge) at the top of the page, the user's own solutions to the left (i.e. all solutions that the user submitted which led to them winning one session of this challenge), as well as all solutions that other (anonymous) users submitted to the right.

This approach, which is also implemented in existing coding platforms such as *Codewars* or *CodinGame*, allows all users of the platform to access and contribute to a collective pool of solutions to a particular task or challenge.

3.3 Submitting and Reviewing New Challenges

The system includes the possibility for users to create their own challenges, which can then be solved on the platform. This feature aims to enable a continuous growth in the number of challenges on the platform without the involvement of the operator of the platform (which in the context of a university could be the teacher of a lecture) having to create such challenges themselves. Users would then have a wider selection of challenges to complete and improve their programming skills. After a challenge has been submitted, it has to be approved by a user with the permission to review challenges first. This process is meant to prevent the submission of inappropriately easy challenges (which users only upload with the intention to later solve them themselves to easily gain points), and of too hard challenges e.g. with confusing descriptions. The Challenge Submission Page asks the user among others to specify

- the title of the challenge
- a description
- the programming language
- the name of the main function of the challenge

- the difficulty (easy, medium, hard)
- blacklisted words
- a template that will be the starting value of the user's editor
- an example solution
- tests that will be used to check whether a solution is correct

The specified name of the main function of the challenge is used by the system to execute the specified tests. From the perspective of a user trying to solve a challenge, this will be the function they have to provide an implementation for. The function with this name will be called with different inputs and its outputs will be compared to the expected outputs.

Every time a solution attempt is submitted by a user, the server will check whether the code of that solution contains one of the blacklisted words the challenge's creator specified and reject the solution without executing the challenge's tests if that is the case. Only code is checked for blacklisted words, comments are ignored. The main purposes of the blacklist feature are to prevent users from importing specific parts of a programming language's standard library and preventing them from calling specific functions. Both of these actions can make a challenge too easy or pose security risks.

The example solution specified by the user will be checked for correctness using the specified tests. If it does not pass those tests, the new challenge is rejected.

The tests the user has to specify have to be provided in the form of *JavaScript* or *Haskell* code that utilizes the respective testing framework provided for that language. It is planned to implement a simpler way to specify tests in the form of just the inputs and expected outputs of the main function of the challenge. These input-output-pairs would then have been converted to test code that uses the testing frameworks by the system.

4 Evaluation

The previous Section consisted of a presentation of the main features of our competitive coding platform. To evaluate the effects of the addition of one-on-one challenges on the users' self-assessment, enjoyment and motivation, two user studies were conducted among undergraduate students.

In this Section, we present an evaluation of these two user studies. Firstly, a description of the scope and the execution of the studies is provided. After touching on limitations to validity and generalizability, a statistical analysis of the results follows. The Section is concluded with our interpretation of this data.

4.1 Scope: Computer Science Students in Their First/Second Year

For the evaluation of the system, we focused on two groups of users. The first group consisted of 7 students taking part in a practical course where they developed a game in groups using the programming language *JavaScript*. The second user group consisted of 16 students attending a lecture on functional programming and modelling, which utilized the programming language *Haskell*. Usually, students who are taking this course have neither been introduced to concepts of functional programming nor to the *Haskell* language before. The entire study was conducted at the *LMU Munich*.

Out of the 24 participants, 7 studied Media Informatics, 14 studied Computer Science as their major subject, and 3 studied Computer Science as their minor subject. 12 of the students were in their first year of studies, 6 in their second, 4 in their third, and 2 students chose not to disclose this information.

21 of the participants did not have any prior experience with coding platforms and 3 did, specifically with *Codewars*, *Codecademy*, *SoloLearn*, and *Exercism*.

The mentioned groups of students were chosen as participants for the study because they represent the main target group of the system: programmers who want to learn a new language or new techniques in a programming language they already know. The goal of the system is helping members of this group to learn more efficiently by providing an engaging way of interacting with programming in the form of competitive and solo challenges. The additional motivation we hope to provide with the highly interactive tasks and immediate feedback might be of particular importance during the first stages of the learning process.
Furthermore, we chose students from courses focusing on two different languages to evaluate whether the system is able to motivate students regardless of the programming language or paradigm they are being taught.

4.2 Method

The conducted user studies were designed to collect two sorts of feedback from the participants: subjective feedback from questionnaires, and objective feedback from the participants' interaction with the system. These two sorts of feedback were collected in turn in order to introduce the participants to the system's features step by step, and to evaluate how the participants' attitudes changed by using the platform.

Both user studies were similarly structured (see Figure 6): After a brief introduction to the study, the participants were asked to complete a pre-questionnaire. The pre-questionnaire consisted of questions regarding basic demographical information, and the participants' general attitudes towards programming and competition. After that, the participants were first told to try solving challenges on their own using our system (*singleplayer phase*), which was followed by a second questionnaire investigating their experience with the singleplayer games they played. The participants were then instructed to compete against other participants in multiplayer challenges (*multiplayer phase*), again followed by a questionnaire investigating their experience with the previously played games. During the fourth and last phase of the study, the participants were free to choose whether they wanted to solve challenges on their own or in competition with other users. After this so-called *freeplay phase*, the participants were asked to fill out a final post-questionnaire, in which they were asked about their general experience with the system and their attitudes towards the competitive aspects of the platform.

The participants were guided through the study by announcements of the supervisor and by slides shown on a projector. During the study, certain interactions of the participants with the system were logged. During all of the three phases, the participants were free to choose challenges of any of the three offered difficulties ("easy", "medium", "hard") although the supervisor recommended to start with one particular challenge of the lowest difficulty ("Triple This"). In the *JavaScript* study, participants were able to choose from five "easy" challenges, three "medium", and one "hard" challenge. In the *Haskell* study, five "easy" challenges, four "medium", and three "hard" challenges were provided. The varying distribution of challenge difficulties was due to some tasks being easier to solve using one programming language than the other, and some of the challenges being too hard to solve for the scope of the study in one of the examined programming languages.

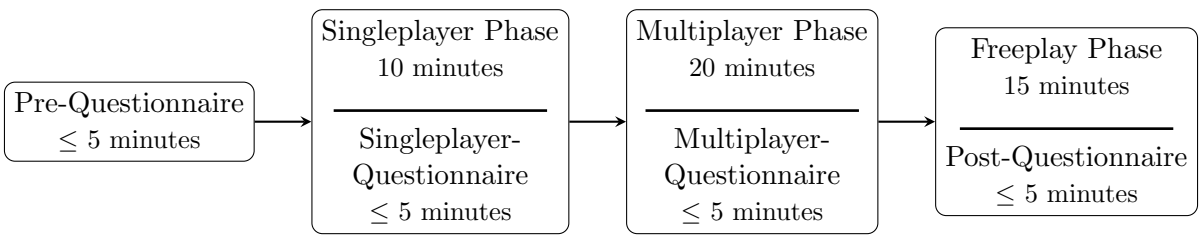


Figure 6: *Structure of the conducted user studies.*

4.3 Limitations to Validity and Generalizability

Before we continue with the results in Section 4.4, this section provides a description of issues that we experienced during the study and that might limit the validity and/or the generalizability of the results.

Among other types of information that were recorded while the participants interacted with the system are the number of multiplayer games won and the number of games lost per participant. Because the logs this information was extracted from are incomplete, there are cases where the loser of a multiplayer game cannot be determined. Section 4.4 contains results that utilize the ratio of multiplayer games won per participant. This ratio is computed using the previously mentioned data. Because this data is error-prone, the true ratio of multiplayer games won per participant could differ from the reported ratios. The true ratios are most likely to be smaller than the ratios that are stated in the following section.

Another technical issue is that the number of jokers used was not logged during the first study. This led to the decision to exclude this objective data regarding the joker usage from Section 4.4.6 from the interpretation, and instead focus on subjective feedback on the jokers.

Apart from issues regarding the underlying data of the results, caution needs to be exercised when generalizing the results. One threat to generalizability is the potentially biased selection of the study participants. The participants share a similar background: all of them were students of computer science or related subjects at the same university. Any biases that arise from this group of subjects therefor also apply to our selection of participants. Additionally, the participants could have expected tasks more similar to the assessments of the respective courses, since the study was advertised in sessions of these courses. Another possible source of bias is the participation in the study being on a voluntary basis (though reward was given in the form of a lottery).

Furthermore, generalizing the results from studies utilizing the two programming languages *JavaScript* and *Haskell* to other languages could be difficult. Efforts were made to mitigate this by using languages of two different paradigms.

Another concern is the limited time range of our study. Some participants could have had problems with getting used to our platform, given the limited time per study phase (as can be seen in Figure 6).

Finally, some unforeseen technical issues during the studies could have negatively influenced the affected the participants' opinion of the platform.

4.4 Results

In the following section, a statistical analysis of the results of the user studies is provided. The focus lies on a quantitative analysis of numeric questionnaire answers and certain (logged) interactions with the system, but individual free-text feedback that was received via the questionnaires is also included. We begin with an overview over the distribution of selected challenge properties (singleplayer/multiplayer, phase of the user study, ...). This is followed by insights into the (subjective) questionnaire answers, and a comparison of those answers to (objective) data that was collected during the user studies. We conclude with an interpretation of the results.

4.4.1 Choice of Challenge Mode

Table 1 shows the number of game sessions per study and phase. In the singleplayer and multiplayer phases, the participants were instructed to start singleplayer/multiplayer sessions only, while they were free to start sessions of any mode they wished during the freeplay phase.

Table 1: *The number of game sessions the participants opened per study and phase. Specifically in case of the freeplay phases of our two studies, the ratio of multiplayer (MP) sessions over all opened sessions is included.*

Phase	**First study**	**Second study**	**Total**
Singleplayer Phase	8	30	38
Multiplayer Phase	7	34	41
Freeplay Phase	20 (20.00 % MP)	57 (21.05 % MP)	77 (20.78 % MP)
Total	35 (31.43 % MP)	121 (38.02 % MP)	156 (36.54 % MP)

The numbers of sessions differ per phase. In the first study, 22.86 % of all sessions took place during the singleplayer phase, 20.00 % of sessions were opened during the multiplayer phase, and 57.14 % of sessions during the freeplay phase. In the second study, 24.79 % of sessions were opened during the singleplayer phase, 28.10 % during the multiplayer phase, and 47.11 % during the freeplay phase.

4.4.2 Choice of Challenge Difficulty

Table 2: *Difficulties of all sessions that were played and ended regularly.*

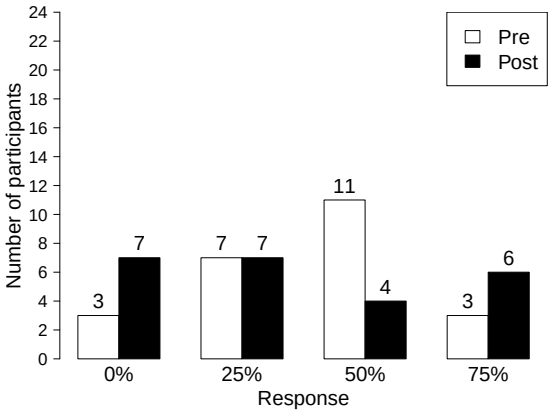
Phase	**Easy**	**Medium**	**Hard**
Singleplayer phase	73.91 % (17)	21.74 % (5)	4.35 % (1)
Multiplayer phase	51.28 % (20)	43.59 % (17)	5.13 % (2)
Freeplay phase	57.45 % (27)	34.04 % (16)	8.51 % (4)
Total	58.72 % (64)	34.86 % (38)	6.42 % (7)

Table 2 shows the frequencies of the difficulties of the challenges users solved. Note that all data regarding the difficulty of challenges played is incomplete as only the difficulty of games that ended regularly can be determined (i.e. with one of the players winning) due to technical problems.

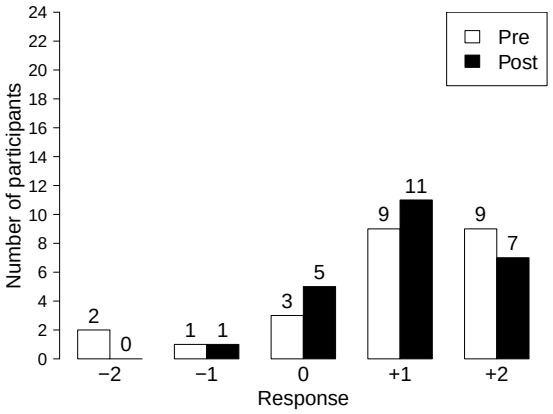
4.4.3 Comparison of Answers to Questions That Were Asked Twice (Trend Questions)

A number of questions was deliberately asked twice, either once in the pre-questionnaire (i.e. before using our platform) and once again in the post-questionnaire (i.e. after using our platform) or once in the singleplayer questionnaire and once in the multiplayer questionnaire. The aim of these so-called *trend questions* is to gain insights about the impact of our platform on the participants' assessments by comparing the distribution of the participants' answers after the later phase to their answers after the earlier phase.

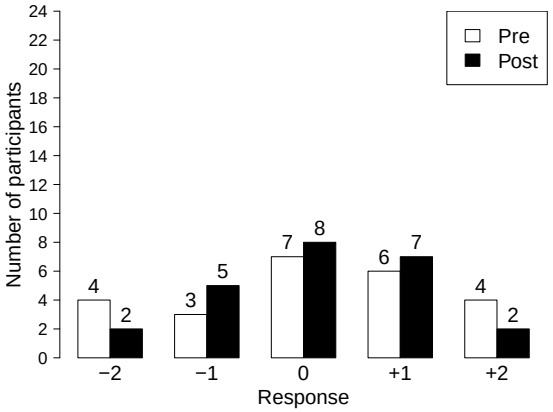
Figure 7 shows the distribution of responses, grouped by the phase at which the questions were asked. The white bars represent the number of responses in the pre-/singleplayer-questionnaire, and the black bars show the number of votes in the post-/multiplayer-questionnaire. The responses of all 24 participants from both studies are included.



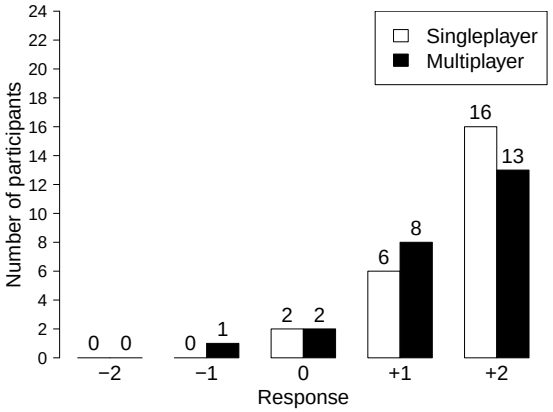
(a) *I assess my skills as better than _ of the group*



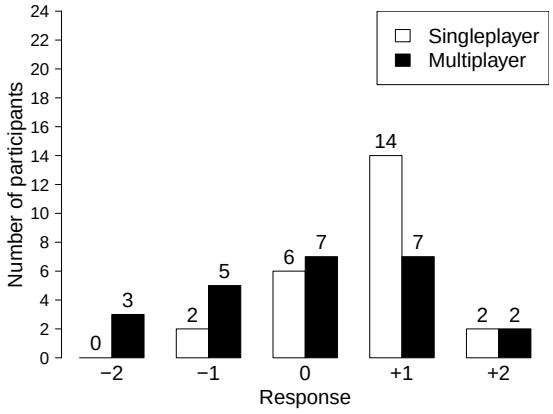
(b) *I can easily motivate myself to solve programming tasks*



(c) *I like to challenge my skills in competition with others*



(d) *I was motivated to solve the programming tasks*



(e) *I felt calm and confident while programming*

Figure 7: *Barplots displaying distributions of the participants' answers to every question that was asked twice. Possible answers range from "strongly disagree" (= −2) to "strongly agree" (= +2). The white bars represent the number of responses in the pre-/singleplayer-questionnaire, the black bars show the number of votes in the post-/multiplayer-questionnaire.*

"I assess my skills as better than _ of the group" In the question whose answer distributions are shown in Subfigure 7a, the participants were asked to compare their perceived programming skill level to the average skill level of the rest of their group. As can be seen in Subfigure 7a, the numbers of participants assessing their skills as both better than 75 % of the group and better than no other group members increased. Furthermore, before being confronted with our platform, 10 participants rated their programming skills on the lower half of the spectrum, while 14 participants rated themselves on the upper half. After using the platform, this distribution switched: 14 participants saw themselves in the lower half of the spectrum, and 10 participants in the upper half.

"I can easily motivate myself to solve programming tasks" The participants were also asked how easy it was for them to motivate themselves to solve programming tasks in general. As Subfigure 7b shows, there was a slight shift of the responses towards an easier self-motivation at the end of the study compared to its beginning.

"I like to challenge my skills in competition with others" Another question the participants had to answer was whether they liked to compete with others in order to test their skills. When comparing the participants' answers from the pre- and post-questionnaires, a shift of opinions towards the center of the scale can be observed, as can be seen in Subfigure 7c.

The previous trend questions were asked once in the pre- and once in the post-questionnaire. However, the next to be evaluated trend questions were asked once in the singleplayer- and once in the multiplayer-questionnaire to directly compare the participants' feedback on the singleplayer and multiplayer modes.

"I was motivated to solve the programming tasks" With this question, the participants were asked to rate their motivation to solve the particular programming tasks they were presented with. The results, as shown in Subfigure 7d, show that there was some decrease in motivation among the participants after being confronted with the multiplayer mode.

"I felt calm and confident while programming" Another value that the participants were told to evaluate was their level of calm and confidence, once before and once after the multiplayer phase. Decreased levels of calm and confidence can be observed in Subfigure 7e.

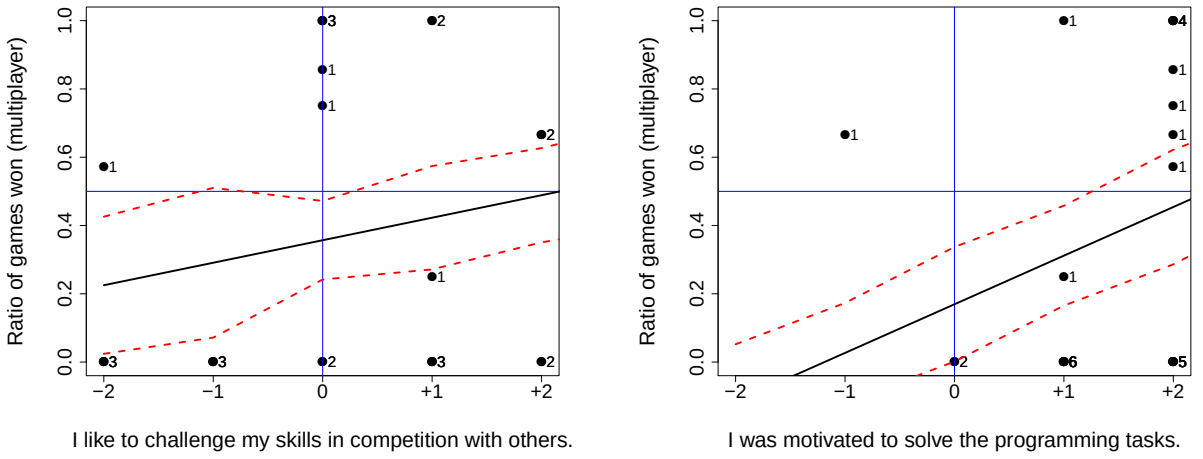
The average answer for each question and phase can be seen in Table 3. The most prominent change (-0.67) represents how calm and confident the participants felt after being confronted with the multiplayer feature compared to before. The participants' motivation to solve the programming tasks decreased by 0.20 after the addition of the competitive element, which is the second most prominent change. The average assessment of participants' skills decreased by 3 %. Apart from this, the averages varied only slightly between the phases of the study.
Another noteworthy detail is that the participants perceived the presentation of the tasks as less clear and comprehensible (-0.16) after playing multiplayer challenges, even though the tasks themselves remained the same.

Table 3: *Comparison of the response mean values before and after using the platform (Pre vs. Post) and before and after confronting the participants with the multiplayer feature (Singleplayer vs. Multiplayer).*

Question	**Pre**	**Post**	**Change**
I assess my skills as better than _ of the group.	39 %	36 %	−3 %
I can easily motivate myself to solve programming tasks.	+0.92	+1.00	+0.08
I like to challenge my skills in competition with others.	+0.12	+0.08	−0.04
Question	**Singleplayer**	**Multiplayer**	**Change**
I was motivated to solve the programming tasks.	+1.58	+1.38	−0.20
I felt calm and confident while programming.	+0.67	0.00	−0.67
The presentation of the tasks was clear and comprehensible.	+1.33	+1.17	−0.16

4.4.4 Comparison of Questionnaire Answers and Performance

In this section, the participants' (subjective) questionnaire answers are compared to their (objective) performance when interacting with the system.
In Figure 8, the participants' reported desire for competition and motivation to solve the programming tasks are compared to their performance against others.



(a) *Ratio of games won over desire for competition*

(b) *Ratio of games won over motivation solve the programming tasks*

Figure 8: *Scatter plots showing selected answers of the participants on the x-axis, and the ratio of multiplayer games won on the y-axis, as well as a regression line (thick, black line) using a linear model together with the respective 89 % confidence interval margins (red, punctuated lines). The data points can be separated into four quadrants, as indicated by the thin, blue lines.*

Ratio of games won over desire for competition Subfigure 8a shows the participants' readiness to challenge their skills in competition with others in relation to their ratio of multiplayer games won.

One aspect that needs to be mentioned is that 13 participants of varying subjective desire for competition did not win any multiplayer games, while 11 participants (also of varying subjective desire for competition) won at least one multiplayer game. Not to win any multiplayer game in this context does not necessarily mean that all of the started games were lost, but that all of the started games were either lost or had to be aborted (e.g. due to limited time per phase).

Another noteworthy observation is the distribution of the quadrant sizes. These four quadrants can be described as follows (neutral answers (= 0) are included both in the left and in the right quadrant):

- Top left: Participants that do not like to compete with others, but won more than 50 % of the multiplayer games they played.
- Top right: Participants that like to compete with others and won more than 50 % of the multiplayer games they played.
- Bottom left: Participants that do not like to compete with others and won less than 50 % of the multiplayer games they played.
- Bottom right: Participants that like to compete with others, but won less than 50 % of the multiplayer games they played.

On one hand, there are as many participants in the bottom left quadrant as in the bottom right quadrant, i.e. among the participants that won less than 50 % of the multiplayer games, their desire to compete with others spreads across the entire spectrum. On the other hand, there is only one participant in the top left quadrant, but four participants in the top right quadrant. This means that, among the participants that won more than 50 % of the multiplayer games, there are more participants that like to compete with others than participants that dislike it.

Ratio of games won over motivation to solve the programming tasks The participants' motivation to solve the programming tasks in relation to their ratio of multiplayer games won is shown in Subfigure 8b.

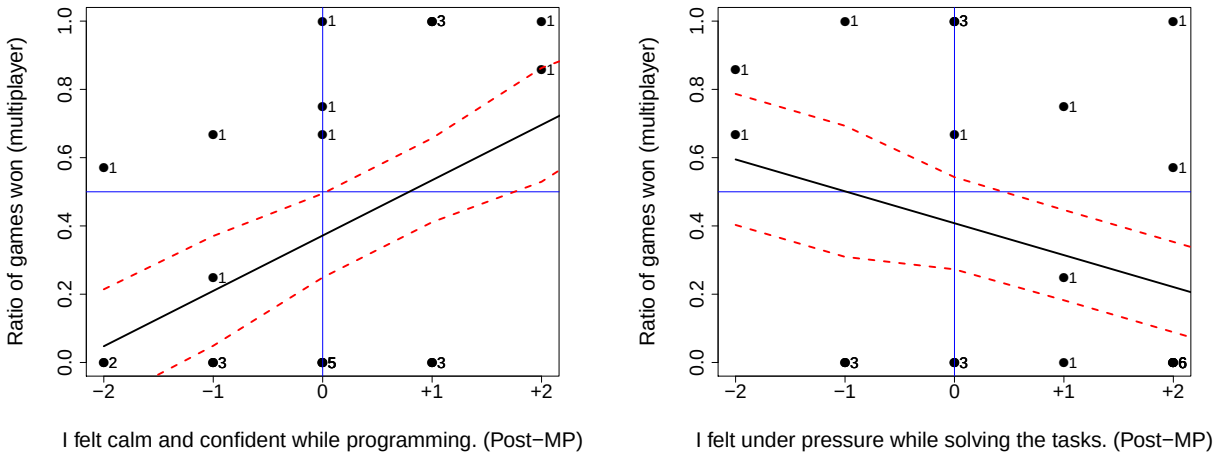
This plot can again be separated into four quadrants:

- Top left: Participants that did not feel motivated to solve the programming tasks, but won more than 50 % of the multiplayer games they played.
- Top right: Participants that felt motivated to solve the programming tasks and won more than 50 % of the multiplayer games they played.
- Bottom left: Participants that did not feel motivated to solve the programming tasks and won less than 50 % of the multiplayer games they played.
- Bottom right: Participants that felt motivated to solve the programming tasks, but won less than 50 % of the multiplayer games they played.

In Subfigure 8b, the top left quadrant has only one member while the top right quadrant has nine. This indicates that, among the participants that won more than 50 % of the multiplayer games, there are nine times more participants that rate their motivation as high as participants that rate it as low. The bottom left quadrant is empty (excluding neutral answers), while the bottom-right quadrant has 12 members.

Additionally, the average motivation among the participants that won more than 50 % is approximately +1.60, while the average motivation among those that won less than 50 % is approximately +1.21.

In addition to the participants' desire for competition and their motivation, we also investigate their perceived levels of calm and confidence, and their perceived levels of pressure while programming in our competitive environment. Figure 9 shows the participants' answers on their perceived calm and confidence, as well as their perceived pressure versus the ratio of multiplayer games won.



(a) *Ratio of games won over perceived calm and confidence*

(b) *Ratio of games won over perceived pressure*

Figure 9: *Scatter plots that show the participants' answers to questions regarding calm and confidence as well as pressure on the x-axis and the ratio of multiplayer games won on the y-axis, as well as a regression line (thick, black line) using a linear model together with the respective 89 % confidence interval margins (red, punctuated lines). The data points can be separated into four quadrants, as indicated by the thin, blue lines.*

The vertical distribution of data points (i.e. the distribution of ratio of multiplayer games won) remains the same, as it is the same dimension as in the previous plots. However, there are some noteworthy insights regarding the answer distributions of participants that won more than 50 % versus those of participants than won less than 50 %.

Ratio of games won over perceived calm and confidence Subfigure 9a shows the ratio of the participants' multiplayer games won over their perceived calm and confidence. Among the participants that won more than 50 % of the multiplayer games they played, a positive correlation between perceived calm and confidence and ratio of games won can be observed, while a majority of participants that won less than 50 % of the multiplayer games they played either reported neutral or negative levels of perceived calm and confidence.

Ratio of games won over perceived pressure At the end of the multiplayer phase, we also asked the participants to rate their levels of perceived pressure while solving the tasks. Their ratio of multiplayer games won over this rating is shown in Subfigure 9b. The most frequent data point consists of six participants that reported a very high perceived pressure (+2) and did not win any multiplayer game (i.e. a ratio of 0.0). This means that among all participants that did not win any multiplayer games (13 participants), approximately 46 % stated that they felt under very high pressure while solving the tasks.

4.4.5 Participants' Feedback Regarding Enjoyment of Different Modes

Apart from perceived values such as the participants' self-assessment, motivation and pressure, the effects the newly introduced competitive environment had on the participants' levels of enjoyment are also investigated. Enjoyment as an intrinsic motivator plays an important role in improving the users' overall experience.

To be able to compare the participants' enjoyment during the non-competitive (singleplayer) and the competitive (multiplayer) phase, the participants were asked to answer two questions that address this issue in the post-questionnaire. One question aimed at how much fun they had while solving the tasks alone, while the other question had them rate how much fun they had while

solving the tasks in competition with others. The participants' answers to these two questions are compared in Figure 10.

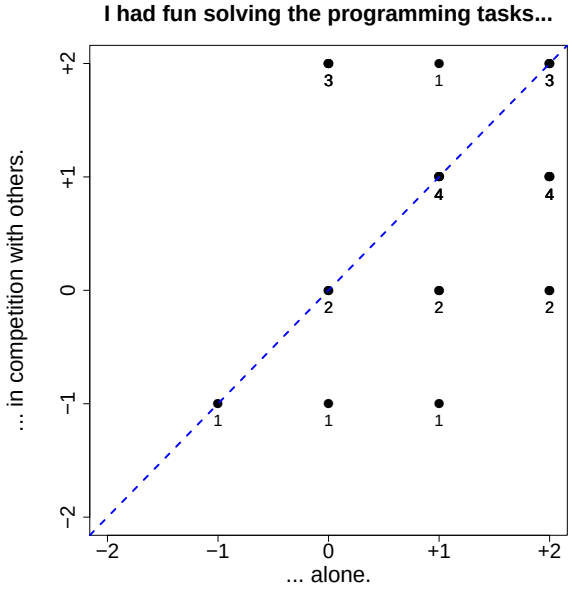


Figure 10: *The participants' enjoyment gained from solving the programming tasks in competition with others (i.e. in multiplayer mode) over that gained from solving them alone (i.e. in singleplayer mode).*

The dotted line in Figure 10 separates the scatter plot into two regions: data points in the region above the dotted line are participants that had more fun solving the tasks in competition with others, while data points below the dotted line stand for participants that had more fun solving them alone. Data points on the line represent equal enjoyment of both modes.
10 participants stated that they had more fun solving the tasks alone, 10 participants reported equal levels of enjoyment, and four participants reportedly enjoyed solving the programming tasks in competition with others more.

4.4.6 Joker Usage

As described in Section 3.2.2, a joker system that adds a game element to our platform was included. In this section, the effects of using the joker system on the participants' performance are evaluated. Figure 11 shows every participant's ratio of games won over their number of jokers used during multiplayer sessions.
As can be seen in Figure 11, the most frequent data points consists of nine participants that neither used jokers nor won any multiplayer game. In contrast to this observation, the participant that used the joker system the most (with a total number of 14 jokers used) won more than 60 % of the multiplayer sessions they played.
However, a majority of the participants only sparsely interacted with the joker system, with 20 out of 24 participants ($\approx 83.3\,\%$ of the participants) having used less than five jokers in total.

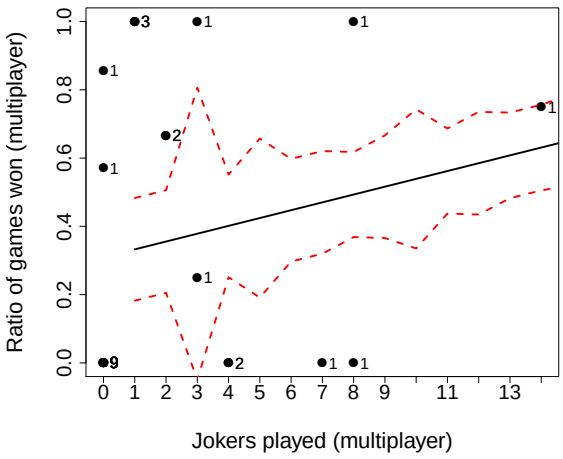


Figure 11: *Scatter plot displaying the (overall) number of jokers played during multiplayer games on the x-axis, and the ratio of multiplayer games won on the y-axis, as well as a regression line (black line) using a linear model together with the respective* 89 % *confidence interval margins (red, punctuated lines).*

4.4.7 Individual Feedback from the Participants

In a number of open questions in the questionnaire, the participants were asked to provide either general feedback on the platform or justify their answers to numeric questions.

General Feedback Participants were presented with a row of questions, asking for general feedback on the platform after each phase. They either described problems they had while participating in the user study, provided feedback on the features of the platform, or suggested new features.

Out of the overall 30 issues reported by participants, four were of technical nature. One participant wished for more game elements to be included into the platform, and 16 participants gave negative feedback on their user experience (UX), e.g. that their code was not saved after they lost a multiplayer game or usability issues regarding the code editor. Five participants requested features that are known from Integrated Development Environments (IDEs), like for example autocompletion, to be included into the platform. Furthermore, one participant gave positive feedback on the user interface, and three participants mentioned challenge-specific problems.

Participants also submitted a number of wishes and improvements. One participant suggested to see an opponent's code after the end of a match, which is something that was purposefully excluded to encourage participants that lost a game to try the same challenge again. Two other suggestions consisted of a wish to continue working after an opponent had won the game. Furthermore, one participant gave positive feedback on the statistics shown after a game's end. Apart from that, one participant mentioned that the system encourages players to produce bad code, without elaborating on the reasons, one participant called attention to ways to cheat our system, and four participants reported that they were either confused about how the jokers worked, experienced problems when using them, or wished for them to be removed from the platform. One participant reported that they liked the absence of IDE features. Another participant requested more social functionalities, specifically the ability to invite friends to a game.

Type of Challenges When asked to justify their reply to a question asking whether they liked the type of challenges they were presented with, one participant mentioned that they did not understand how the jokers worked.

Ability to Use the Platform as Desired When asked whether they were able to use the platform as desired 19 participants answered positively and five negatively. The justifications for

the negative answers included one wish for IDE functionalities to be included and two complaints about technical problems.

4.5 Interpretation

In this section, we interpret the results gathered in Section 4.4.
Because of the generally low amount of willingness to interact with the joker system during the study (see Section 4.4.6), and because of technical issues regarding this feature (see Section 4.4.7) during the study, we will not further interpret the sparse data that could be collected from the participants' joker usage.

4.5.1 Usability Issues

Results of the evaluation suggest that there were some issues regarding the users' usability experience when using the system. Users rarely used the jokers, which were created with the intention of making the gameplay of our system more enjoyable. This low usage and the responses to open questions that asked for the removal of the jokers or mentioned issues with using them suggest that our platform suffered from general technical and usability-related issues. Furthermore multiple participants mentioned that they had trouble navigating to parts of the app they wanted to reach. This is another usability-related problem.

4.5.2 Competition and Its Effects on Self-Assessment, Fun, and Motivation

Effect on Self-Assessment After the experiment, more students reported either very high or very low self-assessments, while fewer students reported medium self-assessments (see Figure 7a). One possible explanation of this "polarisation effect" is that the average participant did not get a chance to evaluate their skills before being confronted with our system. Without a point of reference, they might have been inclined to give a "neutral" answer. However, after competing with others, they might have been driven to more distinctive answers depending on their (subjective) impression of how well they performed. This can be seen as a positive effect, as participants might know their true skill levels better after using the platform. Knowing one's true skill level is important feedback on where one can still improve and where one has already achieved some level of understanding. On the other hand, the slight decrease of the average participants' skill level (see Table 3) is a cause for discussion. This decrease might be caused by the high number of participants that won a minority of the multiplayer games compared to the low number of participants that won a majority of the multiplayer games they played.

Effect on Fun Regarding Subfigure 7d, the participants' opinion whether they like to compete with others seemed to shift towards the center of the scale after using our platform. In Subfigure 8a a slight positive effect of desire for competition on the participants' performance (i.e. ratio of games won) can be observed. Additionally, Figure 10 shows that a majority of the participants enjoyed solving the tasks alone more than while competing with others.
A different observation that matches the previously described one is the high number of participants that reported high levels of pressure under competitive conditions (see Figure 9b). One could argue that the participants experienced this pressure because they had trouble understanding their assignment, but this conflicts with our observation from Table 3 that they perceived the tasks as clear and comprehensible.

Effect on Motivation Instead of experiencing a positive form of competitive pressure, participants seem to have felt pressurized in a negative way. However, an initial average response of +1.58 when participants were asked for their motivation (see Subfigure 7d, see also Table 3) indicates that the motivation to solve the tasks was high to begin with, and remained on a high level even after decreasing by 0.20 to an average of +1.38 in the multiplayer phase.

5 Future Work and Conclusion

Finally, this section touches on future work which could be done on the platform, and a conclusion of this article.

5.1 Future Work

As users repeatedly mentioned UX issues they faced when working with the system, one of the first improvements made to our application could be to resolve some of these problems. The most frequent usability concern was that the navigation on our platform was perceived as confusing: Some users reported problems with locating certain pages from their current page. Therefore, making the navigation more clear would be the first aspect of the user interface that could be improved.

There are a number of features that could be added to our system. One of these is a rating system for challenges and their solutions, e.g. similar to the one offered by *Codewars*. In a simple version, this would allow the users to express their general opinion on the quality of a challenge. A more advanced rating system could allow users to rate specific aspects of the challenge, e.g. its educational value and difficulty. Other ways of providing feedback that could be added are the possibility to rate solutions (which facilitates the ranking of solutions by a majority vote of the users), and the possibility to leave comments on the solutions of other users to discuss them and e.g. suggest improvements.

Apart from that, the approval process for submitted challenges could be improved to allow all users to participate. Currently, only users with a special role can approve challenges. One possible way to allow all users to participate could be to let users play unapproved challenges, which would be marked as such, and then allow them to vote on whether the challenge should be approved.

Furthermore, more game elements could be added to the system. Game elements that could be added to the platform include

- A notion of experience points, which users gather by solving challenges. Experience points could be used to allow the users to rise in ranks, providing an extrinsic motivator for them to solve more challenges.
- Badges that are awarded for fulfilling certain requirements, like solving a number of challenges using a specific language. Badges could be displayed on the users' profile page.

The suggested improvements would then have to be evaluated in further user studies.

5.2 Conclusion

The goal of this research paper was to study the effects of competitive environments on novice programmers. To achieve this goal, a coding platform was implemented, which differs from existing platforms in a way that it adds real-time competition to solving programming tasks. This platform was then evaluated in two user studies among undergraduate students of computer science in the first years of their study program. The results suggest that the users' self-assessment shifted towards extremes when being confronted with a competitive environment, while the users' enjoyment seems to follow an opposite trend. Furthermore, the motivation of the users dropped in competition with others, but nevertheless remained on a relatively high level.

Despite of the limited generalizability of the results, we conclude that competition in programming education does not necessarily have solely positive effects on leaners' experiences. The inclusion of competition in learning environments might prove to be useful, but it is not a silver bullet to increase the learners' motivation.

References

[1] Prins Butt. Students' Perceptions of Game-Based Learning using CodinGame. page 151, 7 2016. International conference on ICT in Education, ICICTE ; Conference date: 07-07-2016 Through 09-07-2016.

[2] [CG]Thibaud. Clash of Code: Challenge your Friends to Short Coding Battles. `https://www.codingame.com/blog/clash-of-code-time-has-come-for-clash/`. Downloaded 10.2.2020.

[3] Coderbyte. Coderbyte. `https://coderbyte.com/`. Accessed 17.2.2020.

[4] Codewars. About Codewars. `https://github.com/Codewars/codewars.com/wiki/About-Codewars`. Downloaded 10.2.2020.

[5] Codewars. Codewars. `https://www.codewars.com/`. Downloaded 10.2.2020.

[6] Codewars. Honor & Ranks. `https://github.com/Codewars/codewars.com/wiki/Honor-&-Ranks`. Downloaded 10.2.2020.

[7] Codewars. Kata. `https://github.com/Codewars/codewars.com/wiki/Kata`. Downloaded 10.2.2020.

[8] Codewars. Kata Solutions & Voting. `https://github.com/Codewars/codewars.com/wiki/Kata-Solutions-&-Voting`. Downloaded 10.2.2020.

[9] Codewars. Privileges. `https://github.com/Codewars/codewars.com/wiki/Privileges`. Downloaded 10.2.2020.

[10] CodinGame. CodinGame. `https://www.codingame.com/`. Accessed 10.2.2020.

[11] CodinGame. CodinGame For Work. `https://www.codingame.com/work/`. Downloaded 10.2.2020.

[12] CodinGame. FAQ. `https://www.codingame.com/faq`. Downloaded 10.2.2020.

[13] CodinGame. Profile. `https://www.codingame.com/profile/`. Downloaded 10.2.2020.

[14] Edward L. Deci, Robert J. Vallerand, Luc G. Pelletier, and Richard M. Ryan. Motivation and Education: The Self-Determination Perspective. *Educational Psychologist*, 26(3-4):325–346, jun 1991.

[15] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, page 9–15, New York, NY, USA, 2011. Association for Computing Machinery.

[16] Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. Gamification in Education: A Systematic Mapping Study. *Journal of Educational Technology & Society*, 18(3):75–88, 2015.

[17] Romain Dillet. With CodinGame, Learning To Code Becomes A Game. `https://techcrunch.com/2015/11/11/with-codingame-learning-to-code-becomes-a-game/`. Downloaded 10.2.2020.

[18] Zoltán Dörnyei. Motivation in second and foreign language learning. *Language Teaching*, 31(3):117–135, jul 1998.

[19] Exercism. Exercism. `https://exercism.io/`. Accessed 17.2.2020.

[20] Tony Jenkins. The motivation of students of programming. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, page 53–56, New York, NY, USA, 2001. Association for Computing Machinery.

[21] Kris M.Y. Law, Victor C.S. Lee, and Y.T. Yu. Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1):218 – 228, 2010.

[22] Elisabeth Lempa. Coconut 2: Concurrently virtualising user code compilation. Bachelorarbeit/bachelor thesis at LMU Munich, 2018.

[23] Rodion Gorkovenko. CodeAbbey. `https://www.codeabbey.com/`. Accessed 17.2.2020.

[24] Topcoder. FAQs. `https://www.topcoder.com/faqs/`. Downloaded 17.2.2020.