Towards an Online Programming Platform Complementing Software Engineering Education

Niels Gandraß, Torge Hinrichs, Axel Schmolitzky
Hamburg University of Applied Sciences
Hamburg, Germany
{niels.gandrass, torge.hinrichs, axel.schmolitzky}@haw-hamburg.de

***Abstract*—Existing online programming platforms provide many opportunities for individuals to develop and improve their programming skills. Using descriptive assignments that address different skill levels these platforms target both beginners and experienced programmers. We used one such platform featuring advanced exercises in a project course during the fifth semester of our bachelor degrees in computer science and were surprised by its success and the high motivation of the participants. Nonetheless, we also observed deficits of the platform with regard to software engineering aspects. In this paper we identify requirements for an online programming platform that complements software engineering education and match existing platforms against these proposed requirements.**

***Index Terms*—education, online assessment, online programming platform, programming, software engineering education, team programming**

I. Introduction

> "It was observed that game based learning act as a good alternative over regular lab assignments and rote learning. Through competitions and coding challenges, students tend to develop interest and therefore are expected to learn more quickly and think more creatively. Hence, digital game based learning should be incorporated in our education system as it offers unique structure to compliment traditional strategies, infusing teaching with energy, spark innovative thinking and provide diversity in teaching methods." — Papastergiou, 2009 [1]

A key aspect of software engineering is programming [2]. In any software development team at least one person actively has to produce source code and all team members are required to have an understanding of its complexity. The ability to program is necessary to follow technical details as well as to judge the feasibility of complex systems. Developers who are strong in both understanding and producing source code will get productive faster in new projects or while analysing complex scenarios [3]. Therefore, the ability to program is the most important skill that software engineering students need to acquire. The better the programming education in a curriculum, the more promising the software engineering education, which later builds upon it, will be.

The individual ability to program can be trained with the help of an *Online Programming Platform* (OPP). We define an OPP as a web-based platform that offers several *assignments* of varying complexity, a possibility to *enter code* that solves an assignment, and an *automatic feedback mechanism* for the proposed solutions. Such platforms can be used by students who want to gain more experience in programming.

Currently, a growing number of OPPs can be observed. Some of these aim primarily at *programming beginners* who are just starting to learn the very basics of coding; in the following we refer to such platforms as *Introductory OPPs*. Others (we call them *Professional OPPs*) aim at recruiting *experienced programmers* who program professionally, have substantial experience in more than one programming language and who still like to solve complex problems in their spare time. Such professional programmers typically know how important maintainable code and a solid software architecture is. We miss something in between (an *intermediate OPP*) for what we call *advanced programmers*, loosely classified as having at least one year of programming experience in one language, in the middle of a computer science curriculum. Students at this stage are just starting to abstract from concrete mechanisms of one programming language, ideally through comparison with another language. They get in contact with design patterns, they learn that an algorithm can be implemented in different ways (some of them even unnoticed wrong), struggle to write readable code, start breaking down larger programs into modules, and designing their interfaces. An intermediate OPP should support all of this with appropriate assignments. Such a platform should explicitly address the code quality of the submitted solutions and it also should support programming in teams.

In this paper we start by describing the benefits and shortcomings we observed while using an OPP within a bachelor project at the Hamburg University of Applied Sciences (HAW Hamburg). Taking these findings as a baseline, we then derive requirements for an OPP that incorporates software engineering aspects and offers support for collaborative team work to be used in the computer science education at universities. Lastly, already available platforms are analysed and compared against our derived requirements.

II. Background: The PHATAI Project

The Bachelor in Computer Science (BCS) degrees at the HAW Hamburg traditionally have a strong emphasis on programming, most modules in the curricula rely on the programming abilities of their participants. But still students in higher

semesters experience difficulties to model abstract or complex problems in source code.

In the fifth semester of all HAW Hamburg BCS curricula, students have to attend a project module (we call the formal module BCS Project or just *Project* in the following), in which they are supposed to work independently in a team and solve a larger problem, using software engineering techniques imparted during the third and fourth semester. The subjects of the projects depend on individual offers by staff members of the department and vary every semester. Students may choose from this range of offers and only a short list of the initially offered projects (typically the most voted ones) can take place. The Project awards 9 ECTS Credit Points (CPs) that are typically translated to a weekly workload of 12 hours for the participants; only three of these hours are contact time with the lecturers.

In one of these projects, during the summer term of 2019, we conducted an experiment [4]: the participants had to solve problems of increasing complexity on the OPP *CodinGame* [5], first simple quizzes as individuals to become acquainted with the platform, later, in teams of four, three of the much more complex *arena games*. Before we can describe our project concept in detail, we have to describe some core concepts of CodinGame.

A. CodinGame: An Online Programming Platform

CodinGame is an OPP with a focus on recruiting programmers, thus aiming primarily but not exclusively at experienced coders. It offers a large variety of programming assignments, ranging from simple ones with just conditions and loops, up to complex scenarios that require, for example, graph algorithms or machine learning. It further offers a wide variety of assignment conduction types, from single person exercises over programming battles in small groups up to arena games where the success of an implementation can be ranked in world-wide leagues. For most assignments the success of a solution is measured with unit tests, whereas the solutions of the arena games battle against a large number (typically around 100) of other solutions in the same league to get their score and rank. Beginners of an arena game always start in the *Wood League* and can climb up through the *Bronze*, *Silver* and *Gold League* into the *League of Legends*. In some arena games, the complexity of the assignment increases in higher leagues through the introduction of additional rules.

One key feature of CodinGame is its use of standard-IO (i.e. the console): each assignment describes the input format a proposed solution will get in textual form via standard input and specifies the (syntax of the) output the solution has to produce textually via standard output. This simple unification concept allows the platform to support a large variety of programming languages for the same assignment; most OPPs are based on this general principle. At CodinGame a user can choose from up to 27 languages.

Another key feature of CodinGame is the visualization of solutions. Many assignments, especially the arena games, offer an animated, two-dimensional view of the programmed

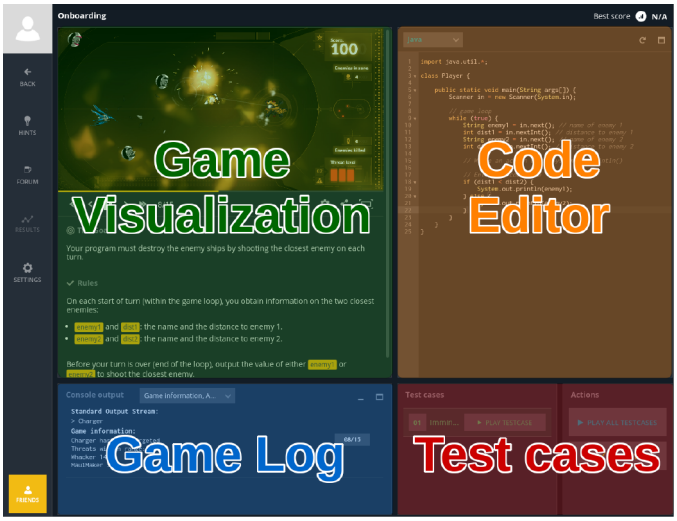

Figure 1. CodinGame's web-based user interface

entities, as shown in figure 1. Programmers can see how their solutions perform; such a visualization of object interaction is also a well-received key concept of the programming environment Greenfoot [6]. Additionally, interactive controls allow pausing and continuing the execution, stepping forward or backward, rewinding to the beginning or fast-forwarding to the end of a battle. This gives the programmer fine-grained control over the execution and allows to analyse the effects of an improvement or the tactics of the opponent.

A user of the platform can run a proposed solution against a subset of the test cases of an assignment, which includes the compilation of the source code. If the code compiles and stands these tests, the user then can submit the code and it will be run against all test cases. In arena games, as was said before, the solution will also battle against other solutions. Thus, especially for the arena games, a programmer typically goes through multiple program-submit cycles, over a longer period of time (from hours over days up to even weeks).

We chose this platform for our project because one of us had substantial personal experience with it.

B. Integration Into a Classroom Teaching Scenario

For our new version of a BCS Project we coined the acronym PHATAI, for "Program Hard And Talk About It". The "program hard" part can easily be served simply by using any OPP; but we wanted the participants to not only improve their individual programming skills, but also their ability to talk about their design process and design decisions with the goal of *giving useful information to others*, i.e. their ability to act as team players and benefit from it.

After a warm-up phase of two weeks with individual coding of CodinGame quizzes, we split the participants up into three teams of four and kept these teams for the rest of the semester. The teams went through three iterations, each with the same basic structure - *Coding, Presentation of Results, Improve Coding, Short Presentation of Improvements* - as shown in figure 2.

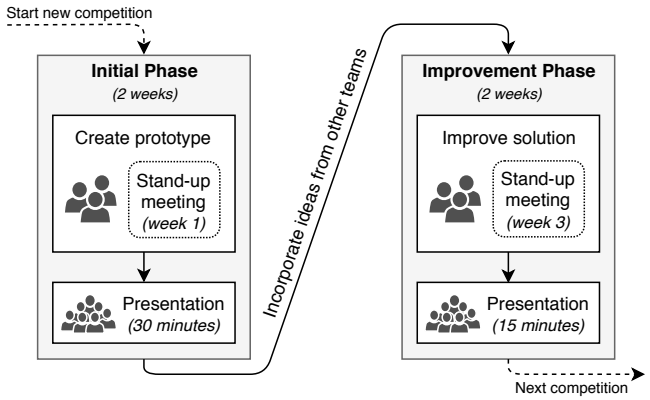

Figure 2. A challenge iteration in the PHATAI project

The initial coding phase lasted two weeks. After the first week, the lecturers held an interim stand-up meeting with each team separately to get an impression of the progress. After the two weeks, in a plenary session each team had to give a 30 minutes presentation of the design and the concepts (algorithms, data structures, ...) of its solution. Each team then got another two weeks to improve its solution (again with interim stand-up meetings), ideally also based on the information gathered during the talks. Finally, a 15 minutes presentation had to be given by each group on the improvements. Even though CodinGame offers 27 programming languages, we decided that Java had to be used exclusively in all coding challenges. This allowed for a better comparison of the developed solutions.

One highly motivating effect during such an iteration was that each solution was automatically ranked by the platform; a higher rank typically indicated a better solution. But if the final rank after four weeks were the only criterion for success, the project would have been all about coding only. Therefore we defined success differently: After the presentations, all participants had to mark the *usefulness* of the talks of the other groups. After three iterations of being graded by peers, the team with the highest "reputation" won a small prize (a set of *Rocket Books*, sponsored by a friendly company located in Hamburg).

Because there is no support for team programming on CodinGame, the teams had to organize their collaboration on their own; the lecturers just required the use of a GitLab instance where they were granted access to the code repositories. This allowed additional plenary sessions in which the quality of the code could be discussed.

At the end of the semester, each group had given a total of six presentations. The quality of the presentations clearly improved over the time of the project: The presented UML diagrams and code snippets became more readable; the students became more confident and their talks more focused. Furthermore, we observed that the quality of the team work improved: at the start of the first arena game we observed mainly "lonesome hacking" and almost no team work; at the end of the third arena game iteration the students made fair use of the git repository, started building architectures that also allowed parallel work on different components and tried to comply with established coding conventions.

C. Lessons Learned

The overall success of the project was a surprise. Programming on a platform with attractive visualizations of the solutions in combination with competitive rankings turned out as highly motivating for the students. The project gave the participants the time to delve deeply into coding and problem solving, so that they even liked to give talks about it.

But there were also some downsides. One of the arena games was taken off the platform in the middle of its iteration; the lecturers had no control over it. The platform offers no support for team work: neither does it support the use of a repository nor does it encourage to split an implementation into suitable modules since the editor allows just one compilation unit. The quality of the code is completely irrelevant - the ranking depends solely upon the effective textual output of a solution.

III. Identifying Platform Requirements

One outcome of the PHATAI project, as described in the previous section, is that we will repeat it in a similar form in the summer term 2020. Some aspects will be improved, but we will again stick with CodinGame for that pass. Nonetheless, we see a clear demand for an OPP that offers improved support for aspects relevant to software engineering education. Therefore, we identified basic requirements for such an OPP that would better suit our needs. These requirements are based on our experiences gathered from the PHATAI project as well as from the usage of Moodle[1] based online Java programming assignments [7] in introductory programming education (first and second semester).

A. Stakeholder Analysis

The first step towards requirements is an analysis of the stakeholders including their individual goals and needs. Each stakeholder has a different point of view and pursues different objectives working with the platform. While the number of stakeholders is potentially larger, the following sections focus on the most important ones. We identified four main roles, namely: *Student*, *Lecturer*, *Developer*, and *Administrator*. In the following we describe their primary goals as well as their behaviour during usage and operation of the aspired platform.

1) Student: The role Student denotes users of the platform, i.e. persons spending a considerable amount of their time on solving provided assignments to improve their skills. As users could also be interested lecturers or other non-students having access, this role has to be distinguished from the formal student status at universities. We still decided to name this role Student, as students are the main target group of the platform.

Students are participants in any given course that is available via the OPP. If using the platform within a university course

[1]Moodle project website: https://moodle.org/ (visited on 10/18/2019)

is mandatory for students, they then can be further categorized into two subsets. The first consists of students that are highly motivated and that aim at learning new skills and improving their existing ones. In contrast, the second set contains students whose primary goal is to pass the course or who lack extra time to spend improving their skills due to the workload of other courses or for personal reasons.

2) Lecturer: A lecturer wants to utilize such a platform to improve the quality of his course by giving students an opportunity to practice and repeat lessons, or even allowing for an optional deeper dive into a certain topic. In our context every person that provides coherent sets of exercises for the OPP falls into this category. From a software engineering lecturers point of view it is desirable to shift the focus away from sole technical details towards a more abstract, method or concept based approach incorporating software engineering aspects.

3) Developer: An OPP should be designed for continuous improvement and development. Developers for an OPP can be divided into *platform developers* and *assignment developers*. The primary goal of a platform developer is to implement new features and fix bugs that were found during operation of the platform. A platform developer also provides the ecosystem for assignment developers that design and implement new assignments for the platform. Ideally it is (technically) less demanding to develop a new assignment than to develop the platform itself. Such a distinction has been successful for example in the Greenfoot programming environment [6].

4) Administrator: Administrators are responsible for the operation and maintenance of the platform. This includes deployment of the OPP software as well as the hardware and other resources required to run it, e.g. servers or network bandwidth. Their key objective is to provide the needed resources and keep the required amount of maintenance effort low.

B. Deriving Requirements

We used the observed benefits and shortcomings, identified in section II, as a baseline for our requirements. These include both a selection of important features and functional requirements. In the following, they are categorized according to their appropriate stakeholders, as identified above. The requirements mainly source from our own experience as lecturers, students and admin staff, but we also conducted informal interviews with other lecturers and students to reflect our ideas.

1) Students: The platform shall incite the students to solve the provided assignments by providing interesting/challenging assignments that are either realistic or entertaining or, ideally, both (STD-01). Visual feedback mechanisms shall help to understand the assignments and also guide the workflow (STD-02). A simple and lightweight on-boarding (registration and initial orientation) is required for students in order to minimize the entrance barrier (STD-03). Furthermore, the platform shall support both pseudonymous (e.g. trough local user accounts) and personal (e.g. trough university LDAP / single sign-on) authentication mechanisms to allow students to complete exercises without the possibility to link negative results to their real identity (STD-04). This can be especially relevant for students of the second identified group, which are still feeling insecure about programming.

From a students perspective the platform has to be easy to understand and use during the work on assignments. Therefore, no complicated setup or configuration should be required. In addition, the development environment shall be supportive, through functionality such as syntax highlighting, autocompletion and “search and replace” as known by existing developer tools (STD-05). This is particularly important for students of the first group, which already have programming knowledge and used professional development environments before. It shall also be possible for a student to get hints or additional assistance, when stuck during an assignment, in order to primarily support programming beginners (i.e. the second student group) and keep up their motivation (STD-06). The solution progress shall also be visible at any time (STD-07). In case of syntax errors, the editor shall assist the student in solving the issue (STD-08). Evaluation of the current solution and generating feedback accordingly shall be possible at any time and be quickly available (STD-09).

Another central requirement is that students should be able to work in teams of variable sizes (STD-10). Accordingly, it is required that the platform can handle solutions build from multiple files coming from various input buffers (STD-11). Moreover, the integration of a version control system (e.g. Git), that enables features such as file revising, branching, code collaboration, and rollback, is a desired feature (STD-12). Furthermore, it shall be possible for teams to compare their solutions against each other in the form of code reviews and course-wide competitions (STD-13).

2) Lecturers: The platform shall support lecturers in a way that they can utilize it within their courses while generating as little work overhead as possible (LCR-01). Its primary objective shall be the incorporation of practical programming assignments that are also able to integrate software engineering concepts or different methods and approaches (LCR-02).

A lecturer should be able to manage course members and teams as well as the assignments relevant for the course. Therefore, adding, modifying and deleting assignments is required (LCR-03). Available assignments shall be exchangeable between lecturers (LCR-04). In addition, it shall be possible to provide additional. Course-specific information for an assignment (hints, bonus material, links, book recommendations on specific topics) (LCR-05). In this context a lecturer shall be able to set limitations on the workflow, e.g. to restrict the order in which the assignments may be solved or to set timing constraints (LCR-06).

Different types of assignments shall be available. These shall include at least the following (LCR-07):

- *Solo Assignments*
  - One student solves the assignment alone.
- *Competitive Assignments*
  - Solutions of multiple students are compared against each other or put into a competitive scenario.
- *Collaborative Assignments*
  - Multiple students work together to create a common solution.

Orthogonally, assignments can be categorized into *Language-Independent Assignments* (LIAs) and *Language-Specific Assignments* (LSAs), and both types should be supported by the platform (LCR-08). This includes that LIAs must be agnostic to the used programming language and therefore grading of a proposed solution can only be done using black-box tests via Standard-IO. In contrast, LSAs can define various constraints on the source code of a solution that have to be respected while solving an assignment; this includes the mandatory use of a specific programming language and the mandatory use or the exclusion of a specific language construct. The latter implies the need for a sophisticated source code analysis tool support that allows to define the necessary white-box tests.

One of the key benefits of such a platform should be that it allows transparency for the lecturer as well as the students. The evaluation criteria should be publicly visible for all participants. The passing criteria should be independent of the lecturer and the learning outcome becomes comparable. This general requirement dictates more specific ones. A lecturer shall be able to define evaluation criteria and weigh them (LCR-09). For example, it shall be possible to reward "clean code" or a "smart" solution. Another aspect of this requirement is that the lecturer shall be able to trace the progress of each student on the assignment (LCR-10).

3) Developer: The requirements of platform developers focus on developing, improving and shipping the platform. The platform developer team has to be able to roll out a new version or parts of the platform seamlessly without any interaction by the user (DVL-02). It must also be possible to run and test individual parts of the platform separately and independent of each other in local test environments or through CI-pipelines (DVL-03).

Platform developers furthermore provide an ecosystem for assignment developers who shall be able to create and maintain available assignments without the need to get in touch with core components of the platform (DVL-04). Created assignments shall be releasable on the platform without modifying the platform code (DVL-05).

4) Administrators: Administrators define requirements that are primarily important for the operation and back-end design of such a platform. For them it is important that the OPP is self-contained and every execution of untrusted source code is encapsulated inside a sandbox environment, therefore running independently of the actual platform itself (ADM-01). This is required so that the platform can not be easily taken over by a malicious user.

Submitted programs and source code assignments also need to be run independent of each other without any possibility of interaction besides well-defined APIs given by the exercise (ADM-02). Furthermore, the (repeated) execution of a submitted solution must not depend on the state of previous executions. Consequently, proper environment initialization and clean-up is required (ADM-03).

Resources that are available to executed assignments shall be limited and it must be possible to stop the execution of an evaluation or grading task at any given time (ADM-04). For the sake of resource efficiency, a flawed execution shall be detected and stopped as fast as possible (*early stopping*) (ADM-05). This includes pre-execution checks (e.g. static code analysis) as well as runtime checks (e.g. interdependent test cases).

IV. Analysis of Existing platforms

After the identification of our requirements for an advanced OPP we systematically analysed existing platforms. We did this with two intentions:

1) To find out how far these platforms already fulfil our requirements.
2) To get more inspiration and insight on how such platforms can be designed.

Because not all platforms that have been described in publications are openly available, we could only take a closer look at OPPs that are publicly accessible. We further selected the platforms according to their popularity across different scoring websites [8], [9] and took different distribution platforms like mobile app or websites into consideration. We ended up with seven platforms that we devided roughly into two groups: The Introductory OPPs LightBot [10], Edabit [11], and CodeMonkey [12] that are designed for programming beginners; and the Professional OPPs CodinGame [5], CoderByte [13], HackerRank [14], and CodeWars [15] that target professional programmers. An overview of the analysed platforms and some of their properties is given in table I. For the sake of readability and compactness, we merged the identified requirements in Section III into the following criteria:

Number of Supported Languages: This metric provides a fairly good first impression of the capabilities of a platform.
Recruiting: Whether the main business case of the platform is recruiting.
Competition: Whether solutions by users can compete against each other.
Visual Feedback: Whether some form of graphical feedback to the user is provided that helps to understand the assignment and supports debugging of the written source code.
Version Control: Whether an external version control system can be connected to the platform.
Team-play: Whether the platform supports group assignments in any form.
Supportive Editor: Whether the editor of the platform offers IDE-like features such as syntax highlighting, auto completion

and search & replace.

Code Quality Feedback: Whether the automatic feedback mechanism takes also the quality of the source code of the solution into account, not just its functionality.

Table I
COMPARISON OF SELECTED PLATFORMS

	Platform	L	R	C	F	V	T	S	Q
Intr.	CodeMonkey	1	×	∼	✓	×	×	✓	×
	Edabit	8	×	×	✓	×	×	×	×
	LightBot	1	×	×	✓	×	×	×	×
Prof.	CoderByte	11	✓	∼	×	×	×	✓	×
	CodeWars	20	✓	✓	×	×	×	✓	×
	CodinGame	27	✓	✓	✓	×	×	✓	×
	HackerRank	23	✓	✓	×	×	×	✓	×

L: # supported languages R: Recruiting C: Competition
F: Visual feedback V: Version control T: Team-play
S: Supportive editor Q: Code Quality feedback

×: No ∼: Partial ✓: Yes

Most available OPPs have in common that they provide easy to use code editors with usability features such as auto-completion and syntax error highlighting.

A. Introductory Platforms

Introductory OPPs mainly support only one programming language and typically provide instantaneous and pleasing visual feedback to the user, as exemplary shown in figure 3. They offer a low entrance barrier that aims at keeping novice programmers motivated. Assignments range from control flow handling to basic algorithmic thinking.

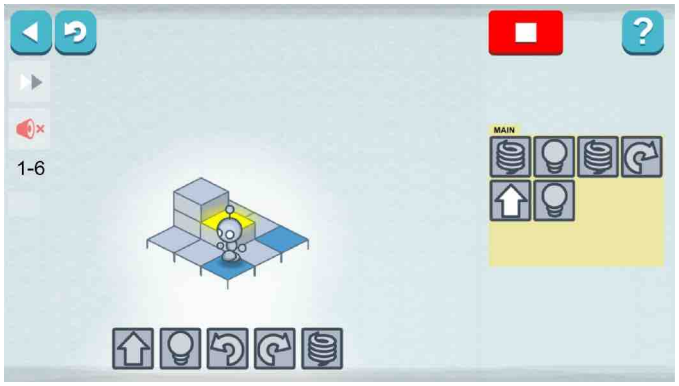

Figure 3. LightBot: user interface with predefined actions

Figure 3 shows the LightBot UI. The goal is to navigate the robot to all blue tiles and transform them into yellow tiles by performing only the actions provided at the bottom of the screen. In the example in Figure 3, the robot can move forward, transform a blue tile into a yellow one with the bulb action, turn left or right, and jump. All actions of a solution need to fit into the yellow area on the right side of the screen. All solutions are evaluated in isolation and never compared to or put into competition with other solutions.

Comparing this platform to the requirements in III shows that only some of the points match. The platform is easy to use (STD-01, STD-03) and provides a visualization of the solution and the problem (STD-02). If the user continuously answers wrong, some help text is displayed (STD-05). Due to the simplicity of the platform other requirements are not applicable. For example, it is not possible to produce source code with syntax error, because the language consists of predefined building blocks only. The platform is a standalone application run in a browser or on a mobile device. There is no interaction with others or the server once the application is downloaded and run. All requirements regarding the infrastructure or development are not applicable.

B. Professional Platforms

The professional OPPs are frequently backed and utilized by large companies to identify potential software developers for recruiting. They allow for solutions using various programming languages, but are often limited to console based output. The only professional OPP that provides pleasing visual feedback is CodinGame, as shown in figure 1. The complexity of available assignments ranges from simple ones that can easily be completed within 5 minutes to more advanced challenges including agent behaviour modelling and various algorithmic problems. Furthermore, some platforms also offer exercises targeting specific skills such as regular expressions, for example found at the HackerRank platform. Professional OPPs usually incorporate some sort of competition among users. This can be found in the form of global score-boards, user levels, badges, achievement systems, and skill certifications. On top of that, some platforms that feature AI programming exercises allow the solutions of multiple users to compete against each other and rank them in a platform wide league system. Moreover, platforms feature regular programming challenges users can take part in and compete against each other. This strong inter-user competition is utilized on most professional OPPs to keep advanced programmers motivated.

The professional platforms are based on basic gamification aspects. The users generally enjoy solving the assignments (STD-01). They are web-based and thus run in a browser (STD-03). CodinGame provides a graphical feedback mechanisms (STD-02), the others are only text based. Feedback on syntax errors is provided by the compiler error shown in a console output (STD-07). This is not supportive but at least shows the error. Assisting features of the editors are minimal (CodinGame and HackerRank provide code completion (STD-05) and syntax highlighting, some others only only highlighting) or not existing. None of the platforms supports teamwork in any form (STD-09) or a real version control system such as git (STD-11), splitting a solution into several text files is also not possible (STD-10). The platforms offer ways to post a solution into a forum and discuss different approaches (STD-12). On all professional platforms it is possible to create and manage own assignments and support during the creation process is offered (LCR-01). Additional information can be provided in the detail description field of each assignment

(LCR-05). The assignments are available for all users so they are exchangeable with others (LCR-04). Other aspects like restricting certain language features, clean code, different assignment types or forcing specific software engineering methods is not possible (LCR-02, LCR-06, LCR-07, LCR-10). Other requirements for lecturers are not applicable. From an administrative and development point of view the platforms are commercially used, so the internal features and processes are not publicly available. But the execution of a solution has limited use of resources and execution time (ADM-05).

C. Further Platforms

During our analysis we discovered further platforms that are documented in scientific publications but are not openly available to the public. Therefore, these could not be included into our detailed analysis. Nonetheless, the platforms depicted below feature novel approaches that we considered noteworthy and inspirational during our research.

1) EasyHPC: Zou et al. [16] created a platform to give students a hand-on experience in high performance computing. The most important feature of this platform is that they are using one of the strongest supercomputers worldwide (Rank 4, June 2019) [17] for their calculations. The platform shall be able to support four forms of education: online courses, quiz database, interactive community and a virtual laboratory. Online courses are lectures provided by a university or a lecturer. The quiz database provides questions for an according topic. The community shall enable students to discuss their solutions in a forum. The virtual laboratory provides a virtual desktop environment, which can be predefined by teachers to allow for an easy start into an exercise. In the paper, design choices and requirements of students and lecturers are presented. Students can select online courses offered by lecturers and complete both assignments and small tasks step-by-step. Teachers can publish new courses, update class materials and deliver course assignments. One unique feature of this platform is that it is possible to configure high-performance computing environments and upload programs running in parallel.

Regarding our requirements, this platform is not enough focused on teaching and supporting software engineering. However, the design process presented in the paper should be taken into consideration in the design of new platforms.

2) BotZone: BotZone was presented in 2018 by Zhou et al. [18] of Peking University. The platform focuses on studying AI and machine learning. The goal is to design and build a multi-agent game AI platform that is capable of evaluating different implementations of game AI by applying them to various scenarios. The implementations compete against each other and achieve ranks in an Elo ranking system. In this way the best approach of the submitted implementations can be evaluated. Students are able to write their solutions in different languages, such as C/C++, Java, C# JavaScript, or Python. The main focus of the platform is the competitive aspect. Therefore, the structure and architecture is optimized to achieve this goal. Detailed technical information on the pipeline or the execution of the different tasks are discussed vaguely. The paper also includes a survey of BotZone in practice. The platform was used in an AI learning course and the authors showed that the use of BotZone was fun and also stressful for a large part of the students.

BotZone is an excellent example of a platform that focuses on competing solutions. In the design of any new platform that is supposed to incorporate competition aspects the lessons learned from this work should be considered.

3) Kodr: The paper of Amr Draz et al. [19] presents a modular learning platform that combines several features from different other platforms to teach Python programming. Kodr helps students to debug their code by using a visualization of the written code. Students are able to review and replay each step the program performs. It also visualizes the memory and stack usage. In addition, the paper features two surveys on how to evaluate the use of the platform and its different exercise types.

The evaluation method of Kodr can be helpful in examining the learning behaviour of students in different assignments.

D. Summary

Even though a lot of inspiration could be found during this analysis, none of the platforms examined fulfils all our requirements; and we also see no extension path for any of the platforms to suit the needs for an OPP complementing software engineering education. No platform supports collaborative team work or incorporates version control systems (VCS), such as Git or SVN. The lack of such functionality prevents the improvement of many valuable skills a software developer is required to have, such as conducting code reviews and managing team workflow. Moreover, none of the OPPs take execution performance, general code quality or software architecture into account when judging a proposed assignment solution. The test verdicts solely depend on whether the submitted programs solve the assignments or not.

V. Related Work

Our requirements seem to lighten up a blind spot in the platform landscape, but several research projects had at least partly similar goals or tackled problems that also result from our requirements.

A. Automated Assessment in a Programming Tools Course

José Luis Fernández Alemán [20] used an automated assessment approach on a programming tools course, which teaches the use of established tools like *make*, *gcov*, *gbd* or *gcc* using C as the given programming language. The assessments include testing, debugging, deployment and versioning tasks. Students are able to access the assignments trough a web based platform and solve the problems on their own. The solution is then send to an ”Online Judge” that compiles and executes the program with predefined input cases. The results get analysed and a feedback is generated for the student notifying if the program is correct or not.

This is a very basic approach but is underlined with a case study of 46 participants showing that this approach lead to

better results comparing to the learning outcome participants who were trained in a traditional way. This paper shows that even basic assistance in programming education helps to improve the learning outcome for students.

B. The BOSS Online Submission and Assessment System

Joy et al. [21] use a more complex approach for an online assessment system. They identified three fundamental components for an assessment. The first being the correctness of the solution. This is defined by the specification of the assignment and is checked by black and white box tests. The second component is the style, which they sub-classify in two parts, language independent and language specific characteristics. Independent features are for example efficiency of the program or the choice of an algorithm used. Among others, Language specific features are the code structure or the use of external libraries. The last component, authenticity, describes tasks of organizational nature such as managing the users, verifying the identity of users and checking for plagiarism. The paper shows which requirements can be derived from different user-groups interacting with the platform for example the staff and the students. These requirements define different views of the system, which dictate the basic architecture and the individual components inside the system.

In comparison to the requirements presented in this work, some are in common and the lessons learned can be reused, for others, such as teamwork or version control and management, new approaches have to be developed. The detection of plagiarism, a core requirement in the BOSS system, is deliberately excluded from our requirements. We think that the focus on platforms that provide helpful, non-mandatory additional resources for learners reliefs from many problems formal grading systems have to face.

C. Towards Practical Programming Exercises and Automated Assessment in Massive Open Online Courses

Staubitz et al. [22] state several benefits and shortcomings of different approaches of designing an online programming platform for massive user use. They concluded that they need a flexible solution that can handle different scenarios based on the expertise of the course. Programming beginners prefer a web based editor where no set up is required. Advanced users favour development tools used on their own machine. The presented platform consists of an automated testing suite that can check the given solution with a number of different tests on unit and acceptance level. This paper conquers issues we also identified as relevant in our requirements, such as scaling the application and modifying it on demand, depending on the course and the experience of the students. The results can help to improve strategies dealing with those issues in future systems.

D. Giving Automated Feedback About Student Code Identifiers: a Method Based on the Description of Programming Problem

Nascimento et al. [23] presented a system that can help programmers to improve the readability of their programs. The system analyses student assignments and generate automated feedback on the used identifier names. A survey shows that this can help to improve the code quality of the assignments and enhances the readability of the code in contrast to manual reviews by lecturers. The first step is a normalization of the assignment description and extract a reference vocabulary from it. This step contains normalization, tokenization and stemming techniques. In the second phase, the identifier names in the source code are estimated in a mapping to the words in the vocabulary. Words that can not be mapped to a vocabulary entry are considered to be inappropriate and a warning is created to signal the students to reconsider the identifier name. In a case study also presented in the paper this leads to an accuracy of 75% compared with a manual assessment by the lecturers.

This paper presents useful techniques and methods for performing analyses of non-functional characteristics of student source code. The approaches used for normalization of source code and the deduction of code quality can be used in future systems.

VI. Conclusion

Online programming platforms provide vast opportunities for individuals to develop and improve their programming skills. In this paper, we showed that many platforms feature novel approaches that provide users with modern and highly motivational game-based programming assignments. But none of the available platforms addresses essential software engineering aspects such as code quality, maintainability, adequate program architecture, team based programming, or code reviews.

Based on our experience gathered from successfully using the CodinGame platform within a bachelor project, we proposed requirements for an OPP that better complements software engineering education. We assume that an OPP, when developed with respect to these requirements, can significantly raise the quality of teaching and be beneficial for both students and lecturers.

In future work we aim at creating a platform prototype which is based on the proposed requirements and at evaluating it with students in software engineering courses. Gathered feedback will be used to refine the requirements and improve the created platform. A long term goal is to establish such an online programming platform in the BCS degrees at the HAW Hamburg as an assistance in all programming-related modules.

References

[1] M. Papastergiou, "Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation", *Computers & Education*, vol. 52, no. 1, pp. 1–12, 2009, ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2008.06.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131508000845.

[2] A. Schmolitzky, "Zahlen, Beobachtungen und Fragen zur Programmierlehre", in *Tagungsband 15. Workshop "Software Engineering im Unterricht der Hochschulen"*, 2017, pp. 83–90. [Online]. Available: http://ceur-ws.org/Vol-1790/paper10.pdf.

[3] D. I. K. Sjoeberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N. .-.-. Liborg, and A. C. Rekdal, "A Survey of Controlled Experiments in Software Engineering", *IEEE Transactions on Software Engineering*, vol. 31, no. 9, pp. 733–753, Sep. 2005. DOI: 10.1109/TSE.2005.97.

[4] T. Hinrichs and A. Schmolitzky, "Einbindung einer Online-Programmierplattform in die Präsenzlehre – ein Erfahrungsbericht", in *Proceedings of the Fourth Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2019), Essen, Germany, October 8-9, 2019*, Gesellschaft für Informatik e.V., 2019. DOI: 10.18420/abp2019-8.

[5] CodinGame, [Online]. Available: https://www.codingame.com/ (visited on 10/15/2019).

[6] M. Kölling, "The Greenfoot Programming Environment", *Trans. Comput. Educ.*, 14:1–14:21, Nov. 2010, ISSN: 1946-6226. [Online]. Available: http://doi.acm.org/10.1145/1868358.1868361.

[7] N. Gandraß and A. Schmolitzky, "Automatisierte Bewertung von Java-Programmieraufgaben im Rahmen einer Moodle E-Learning Plattform", in *Proceedings of the Fourth Workshop "Automatische Bewertung von Programmieraufgaben" (ABP 2019), Essen, Germany, October 8-9, 2019*, Gesellschaft für Informatik e.V., 2019. DOI: 10.18420/abp2019-1.

[8] The 10 most popular coding challenge websites for 2020, [Online]. Available: https://www.freecodecamp.org/news/the-10-most-popular-coding-challenge-websites-of-2016-fb8a5672d22f/ (visited on 10/29/2019).

[9] The 10 Best Coding Challenge Websites for 2018, [Online]. Available: https://medium.com/coderbyte/the-10-best-coding-challenge-websites-for-2018-12b57645b654 (visited on 10/29/2019).

[10] LightBot, [Online]. Available: https://lightbot.com/ (visited on 10/15/2019).

[11] Edabit, [Online]. Available: https://https://edabit.com/ (visited on 10/15/2019).

[12] CodeMonkey - Coding for Kids, [Online]. Available: https://www.codemonkey.com/ (visited on 10/15/2019).

[13] Coderbyte, [Online]. Available: https://coderbyte.com/ (visited on 10/15/2019).

[14] HackerRank, [Online]. Available: https://www.hackerrank.com/ (visited on 10/15/2019).

[15] CodeWars, [Online]. Available: https://www.codewars.com/ (visited on 10/15/2019).

[16] Z. Zou, Y. Zhang, J. Li, X. Hei, Y. Du, and D. Wu, "EasyHPC: An Online Programming Platform for Learning High Performance Computing", in *2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Dec. 2017, pp. 432–435. DOI: 10.1109/TALE.2017.8252374.

[17] TOP500 List - June 2019 Supercomputers, [Online]. Available: https://www.top500.org/list/2019/06/ (visited on 10/29/2019).

[18] H. Zhou, H. Zhang, Y. Zhou, X. Wang, and W. Li, "Botzone: An Online Multi-agent Competitive Platform for AI Education", in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018, Larnaca, Cyprus: ACM, 2018, pp. 33–38, ISBN: 978-1-4503-5707-4. [Online]. Available: http://doi.acm.org/10.1145/3197091.3197099.

[19] A. Draz, S. Abdennadher, and Y. Abdelrahman, "Kodr: A Customizable Learning Platform for Computer Science Education", in *Adaptive and Adaptable Learning*, K. Verbert, M. Sharples, and T. Klobučar, Eds., Cham: Springer International Publishing, 2016, pp. 579–582, ISBN: 978-3-319-45153-4.

[20] J. L. Fernandez Aleman, "Automated Assessment in a Programming Tools Course", *IEEE Transactions on Education*, vol. 54, no. 4, pp. 576–581, Nov. 2011, ISSN: 1557-9638. DOI: 10.1109/TE.2010.2098442.

[21] M. Joy, N. Griffiths, and R. Boyatt, "The Boss Online Submission and Assessment System", *J. Educ. Resour. Comput.*, vol. 5, no. 3, Sep. 2005, ISSN: 1531-4278. DOI: 10.1145/1163405.1163407. [Online]. Available: http://doi.acm.org/10.1145/1163405.1163407.

[22] T. Staubitz, H. Klement, J. Renz, R. Teusner, and C. Meinel, "Towards practical programming exercises and automated assessment in Massive Open Online Courses", in *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Dec. 2015, pp. 23–30. DOI: 10.1109/TALE.2015.7386010.

[23] D. S. Marcos Nascimento Eliane Araújo, "Giving Automated Feedback About Student Code Identifiers: a Method Based on the Description of Programming Problem", in *Anais do SBIE 2019 (Proceedings of the SBIE 2019)*, 2019. DOI: 10.5753/cbie.sbie.2019.537.