Gamification-based e-learning Platform for Computer Programming Education

Conference Paper · June 2013

CITATIONS
23

READS
467

2 authors, including:

Jakub Swacha
University of Szczecin
71 PUBLICATIONS **284** CITATIONS

SEE PROFILE

X World Conference on Computers in Education
July 2-5, 2013; Toruń, Poland

Gamification-based e-learning Platform for Computer Programming Education

Jakub Swacha, *jakubs@uoo.univ.szczecin.pl*
Institute of Information Technology in Management, University of Szczecin,
Mickiewicza 64, 71-101 Szczecin, Poland

Paweł Baszuro, *pbaszuro@acm.org*
Consulgo, Gorzów Wielkopolski, Poland

Abstract
In this paper a novel design of an e-learning platform is proposed, aimed at programming education for groups of students. The platform is conceived as open-source software and its main technological design goal is to reduce its requirements on server-side resources. On the educational level it makes use of a number of gamification concepts, which are expected to increase students' engagement in learning by not only awarding individual work but also stimulating students, on different occasions, for both immediate rivalry and team work.

Keywords
computer programming education, e-learning platform, gamification

INTRODUCTION
Gamification can be defined as "the application of gaming metaphors to real life tasks to influence behaviour, improve motivation and enhance engagement" (Marczewski, 2012, p. 4). As education is an area in which improved motivation and enhanced engagement is certainly needed, not surprisingly it is also an area of various gamification attempts (Lee & Hammer, 2011).

Teaching computer programming is considered to be difficult and often ineffective (Robins et al., 2003). For this reason, various approaches were proposed and tried with lack of convincing success (Dehnadi & Bornat, 2006). They included using games as the subject of programming to augment the involvement of students, either at an initial (Leutenegger & Edgington, 2007) or later (Swacha, 2010) stage of course. Gamification, however, offers a whole new set of opportunities to make the students more involved, such as clear goals with variety of ways to approach them, curiosity, systems of challenges, constraints and achievements (cf. Table 1 in Deterding et al., 2011).

These opportunities can be exploited even in a traditional teaching process (Kumar & Khurana, 2012). However, it is the online learning and teaching that should be of primary concern now, due to increasing demand for education and training in the information age (Oncu & Cakir, 2011).

Online learning requires a platform, "a set of interactive online services that provide learners with access to information, tools and resources to support educational delivery and management through the Internet" (Ghirardini, 2011, p. 118). There are well-known online learning platforms that employ gamification, such as curatr.co.uk (Betts, 2010) and new ones keep being developed (Gåsland, 2011), also at least several from over fifty gamification platforms listed by Herger (2012) could be used

in educational scenarios. Nevertheless, a dedicated platform, with specific technical capabilities, such as an ability to automatically verify that the source code entered by students online is syntactically and semantically correct, can yield superior educational results in computer programming courses (Fernandez, 2011).

In this paper we define key design goals and propose a model of an e-learning platform aimed at computer programming education, which includes (1) structure of the platform, (2) concept of gamification-driven course progress, (3) basic platform functionalities, and (4) technology of implementation. A new e-learning platform is being developed, which will eventually serve as a proof of concept for the design proposed here.

RELATED WORK

Although computer programming courses are often offered using universal online learning platforms, in this section we shall only investigate environments which are specialized for this task, i.e., technically capable of checking and running program code entered by the student.

Currently, one of the most popular types of systems supporting the learning of computer programming is automatic assessment platform, often referred to as an online judge (Kosowski et al., 2008). Such a system usually consists of a repository of programming problems, which can be searched by various criteria, a judge system that performs the automatic assessment of the submitted solutions, and a user management system that tracks the history of submissions for each user and generates user rankings. Program correctness and efficiency are the standard criteria of assessment, yet some systems offer additional options, such as coding style evaluation, design pattern recognition or plagiarism detection (Ihantola, 2011, pp. 23-25). Although the automatic assessment platforms are very helpful in verification of programming skills, teaching programming cannot be based on them alone, as they lack capabilities to effectively provide instructional materials to the students and communicate with them.

One way of solving this problem is to compose online judges into existing learning management systems (LMS). Such solutions were proposed, among others, for such popular e-learning systems as Claroline (Georgouli & Guerreiro, 2011) or Moodle (del Alamo et al., 2012; Verdú et al., 2012). A more far-reaching approach is taken by Amelung et al. (2011) who propose automatic assessment as a service which can be integrated with any LMS. As a result, teachers can use programming exercises, that can be automatically assessed, in combination with their existing courses, and in environments they are accustomed to.

Another approach is to design a new learning management system, specialized for teaching programming, with embedded code execution and assessment capabilities. Good examples of such systems are those described by Moon et al. (2008), Kasyanov & Kasyanova (2009), or Combéfis & le Clément de Saint-Marcq (2012).

The systems mentioned so far perform code execution server-side; recently, a number of solutions appeared that perform code execution within the browser environment to reduce server requirements and security risks. They work by translating the source code client-side, so that it can be run in Java Virtual Machine (Helminen & Malmi, 2010), Silverlight environment (Foord, 2012) or any JavaScript-capable browser (Guo, 2013). This technology allows to embed interactive programming exercises on websites or in electronic textbooks (see, e.g., Miller & Ranum, 2012). The client-side code execution can be combined with server-side software to track the students' progress, as in, e.g., Code Avengers (2012), a programming e-learning platform for high schools.

Although one can find rudimentary gamification elements in online judges, such as registering the number of correct and incorrect submissions, and ranking users according to the former, as well as hosting contests with due date for submissions, none of the platforms discussed so far in this section is actually gamified.

Currently, to the best knowledge of these authors, Codeacademy is the only e-learning platform specialized for computer programming, designed with gamification in mind (Ryzac, 2012). The programming courses there are organized into sections, consisting of sequences of, possibly interconnected, exercises, which in turn consist of an educational text introducing the topic, instructions that tell a student what to do, and the actual interactive exercise to be completed. Students earn one point for completing each exercise, completion of a lesson is registered as an achievement. Also, the maximum number of points earned in one day and the maximum number of days in a streak the particular student logged in are registered. Students are rewarded with badges for attaining specific number of points, lengths of streaks, or completing certain lessons or courses, which they can share on Facebook.

Codeacademy has no student management capabilities of LMS, the course format is somewhat limited, and its being a cloud-based solution brings risk of losing time and effort spent on preparing courses, as its license may change (currently it is free for both students and teachers), its functionalities may evolve in undesirable direction, or, in the worst case, it may be closed down.

KEY DESIGN GOALS

The proposed platform supports teaching computer programming in e-learning and blended learning courses; an active teacher's involvement is assumed. The platform is a stand-alone solution – not an extension module for an existing e-learning system. The platform provides core functionalities of LMS, in particular, it allows teachers to distribute electronic learning materials to the students, track students' progress, and communicate with individual students and groups of them.

The platform employs gamification mechanisms (Knewton, 2012), such as: progression (approaching the success incrementally, with current progress measured with levels and points), achievements (public recognition of work completion), appointments (rewarding time-keeping), collaboration (on solving exercises), epic meaning (given by using specific terminology), and cascading information theory – a continuous releasing of information, in a form of bonuses (unexpected rewards), countdown (time limits for challenges, contests and quests), discovery (course elements to be uncovered by students themselves), loss aversion (awarding regular activity), infinite play (more exercises available as the student progresses), as well as synthesis (requiring more than one type of activity to progress). The mechanisms are expected to increase students' motivation and enhance engagement throughout the course progress. They also encourage them to participate in various activities that are supposed to extend their knowledge (by learning from instructional materials and external sources), professional skills (by practicing programming exercises), and social competencies, such as collaboration (by awarding solving problems in groups of students), communication (by awarding merit-based discussion on forum and tutoring other students) and confidence in own skills (by stimulating direct competition with other students).

The platform assumes an active role of the teacher, as an instructor, mentor and judge. The platform allows submissions to be assessed both automatically (in terms of correctness and efficiency) and by teacher, in terms of quality – see, e.g., work by Skupas & Dagiene (2010) for justification of semi-automatic program evaluation.

STRUCTURE OF THE PLATFORM

The platform consists of four sections:

- Account Management, which handles access rights to courses and their elements, and tracks students' progress, submissions, and achievements;
- Course Management, through which publishing instructional materials, and tasks and exercises, as well as holding contests is accomplished;

- Training Area, where instructional materials can be viewed, embedded exercises and quizzes solved, and task results submitted;
- Competition Area, where voluntary and contest exercises can be solved and other students challenged;
- Forum, where students and teachers may communicate.

In the nomenclature used in this paper, quiz is a sequence of questions verifying student's knowledge, exercise is a problem that student has to solve by writing (or correcting, or improving) a program, and task is anything else that student has to do (e.g., finding some information, or preparing specific document or diagram).

CONCEPT OF COURSE PROGRESS

The overall progress of a student in a course is denoted by: (1) the set of completed training areas, (2) the number of earned points, and (3) the attained level.

The training areas are closely related to lessons, encompassing instructional materials for a specific topic. In synchronic courses, the consecutive training areas are revealed no sooner than a lecture for specific topic has been given.

There are various ways for a student to earn points. Each point earned is attached to one or more of the following five domains:

- Learning, points in this domain (shown in cyan) can be earned by attending or viewing lectures and completing interactive lessons, including reading certain materials and passing embedded exercises, tasks, and quizzes;
- Practicing, points in this domain (shown in green) can be earned by completing quests and solving voluntary exercises;
- Competition, points in this domain (shown in red) can be earned by challenging other students (or accepting challenges) and taking part in contests (especially with success), as well as revealing incorrect answers on the forum;
- Collaboration, points in this domain (shown in yellow) can be earned when working in groups when solving exercises and taking part in contests, as well as getting highly positive rating for a post on the forum;
- Tutoring, points in this domain (shown in blue) can be earned by publishing own exercises, getting highly positive rating for them, and answering questions on the forum.

The points in respective domains can be spent to perform specific actions:

- points from the Learning domain can be spent to complete a teaching area;
- points from any domain can be spent to post a question on the forum, challenge other student, decline a rematch, or take part in a contest;
- points from at least three different domains can be spent to level-up.

Moreover, points may be lost for failing quests or after getting highly negative rating for a post on the forum. The points that were spent or lost become out-of-domain points (shown in gray), but still count to the total number of student's points. When the total number of points passes specific threshold, a student is allowed to level-up.

Exercises

Each exercise has: unique identification number and name, purpose (embedded, voluntary, only for contests, only for quests), type (for groups or individual students), difficulty level, point reward, bonus (extra points in a domain other than Practicing), special bonus (usually given for the first student to solve it, e.g., free level-up, instant completion of specific area, or other achievement), list of relevant training areas, prerequisites (completing specified other exercises), context description, task description, special requirements, required output description, hints, initial code, correctness checking code, execution time limit, memory usage limit, submission history, rating (initially neutral).

Students can only solve exercises whose difficulty level is not higher than the student's level plus two, and that are not relevant to training areas that were not yet revealed to the student.

Quests

Quest is an exercise, task or a combination of those chosen by a teacher for a specific student. Quests are offered in private communication.

Contests

Contests can be held by teachers only. Each contest has a unique name, type (for groups or individual students), submission deadline, set of exercises, and rewards for a specified number of top-finishing contestants. Submissions are evaluated on correctness, running time and memory, length, submission time, but can be additionally evaluated on quality by a teacher or instructor.

Challenges

Students can challenge only students whose level is not lower than their own minus one. Challenging students choose challenge type (sink-or-swim or time-trial), difficulty level and submission deadline. Challenged student can accept challenge, postpone it, or decline, but declining a challenge from a previously beaten opponent (rematch) costs points. An exercise is chosen randomly by the system, such that is accessible to both competitors. A sink-or-swim challenge is won by a student who submitted a correct solution in due time, only if the opponent failed to do so. If they both succeeded, there is a draw. In a time-trial challenge, the student who submits a correct solution faster than the other wins. If they do it at the same moment (no more than 30 seconds of delay), there is a draw.

The winner receives points from the competition domain proportional to the loser's level. Besides, competitors receive usual awards for solving the exercise, regardless of the competition result.

Additional gamification mechanisms

Students earn badges by completing certain tasks (e.g., requiring knowledge beyond training materials), solving certain exercises first, attaining certain levels in specific amount of time, achieving successes in challenges, contests and being active on the forum.

Students' course statistics is recorded and accessible via student's profile and shown in various rankings. It includes: total points (with history); points in respective domains; level; list of badges; progress in respective training areas; submitted and correct exercise solutions; challenges given, accepted, and won; contest history; forum posts; longest streaks of: days being logged in, correct exercise submissions, challenges won and not lost.

End of course

End of course is a date set by the teacher for a specific group of students. The final grade for the course is the level that a given student attained until that day. It is the teacher's decision whether students can continue using the platform after that date.

BASIC PLATFORM FUNCTIONALITIES

The platform's basic functionalities can be organized in the following groups:

- administering: users, courses (materials and participants), the forum;
- asking teacher for: quality review of a submission, judgment;
- communicating: by web chat (for users online), in the forum, by mail;
- editing and testing: solution source code;

- posting on the forum: questions, answers to questions, answers to answers (improvements or revealing errors), other discussion;
- publishing: lectures, interactive lessons, quizzes, tasks, exercises;
- rating: instructional materials, exercises, posts in the forum;
- searching for: exercises, students, posts in the forum;
- setting up: challenges (for students), contests and quests (for teachers);
- submitting solutions for: exercises, tasks, quizzes;
- viewing: lectures, interactive lessons, exercises, tasks, quizzes, student profiles, student rankings, exercise rankings, topics and posts in the forum.

Specific functions are only available for privileged users: administrators and teachers (within their own courses).

Student group management

Multiple groups of students can attend the same course in the same time with different training area revealing dates and contests. Each group of students can be either open or closed. Open group members see rankings and challenge students from all open groups; closed group members can only see rankings and challenge students from their own group.

IMPLEMENTATION TECHNOLOGY

The platform is conceived as a fat-client web application with a possibility of hosting server-side components in educational institution's own network.

Three crucial components can be identified in the architecture of the system: user interface, program execution and validation environment, and back-end (see Figure 1).

From the users' point of view, the platform can be perceived as a website, containing a collection of web pages. Each web page is in fact a rich internet application (RIA) written in HTML, JavaScript and CSS, with components and functionalities appropriate to its type.

Theoretically, the platform can be used for teaching any programming language. In order to ensure that, a virtual-machine-based approach is used, in which web browser's JavaScript engine simulates other languages' execution engines. The first time a specific student's source code is executed within the platform, his/her web browser automatically downloads an appropriate virtual machine from the course server. Note that the virtual machine is implemented in pure JavaScript, therefore, no special plug-ins have to be installed on the client machine.

A common application programming interface (API) is provided to be used within the virtual machine. For any given exercise, the API can be limited by a teacher to concentrate students' attention on areas specific to the exercise.

By default, program evaluation is done client-side to reduce burden on the server and allow for immediate response. Programs evaluated as correct are sent to the server along with results, to let the teacher check for cheating and optional plagiarism detection. Server-side execution, in a sandboxed environment, is also available for, e.g., usage in contests, where the programs need to be run in the same environment for execution time comparisons.

The chosen implementation technology lets the platform to be used from any software or hardware platform, provided JavaScript-capable web browser is available. This way, it becomes available to users of both traditional personal computers, and a wide range of modern mobile devices, regardless of their processor type and operating system.

The back-end handles storage, access control and searching for all types of content and metadata used by the platform. It also provides a base for the platform's communication capabilities.

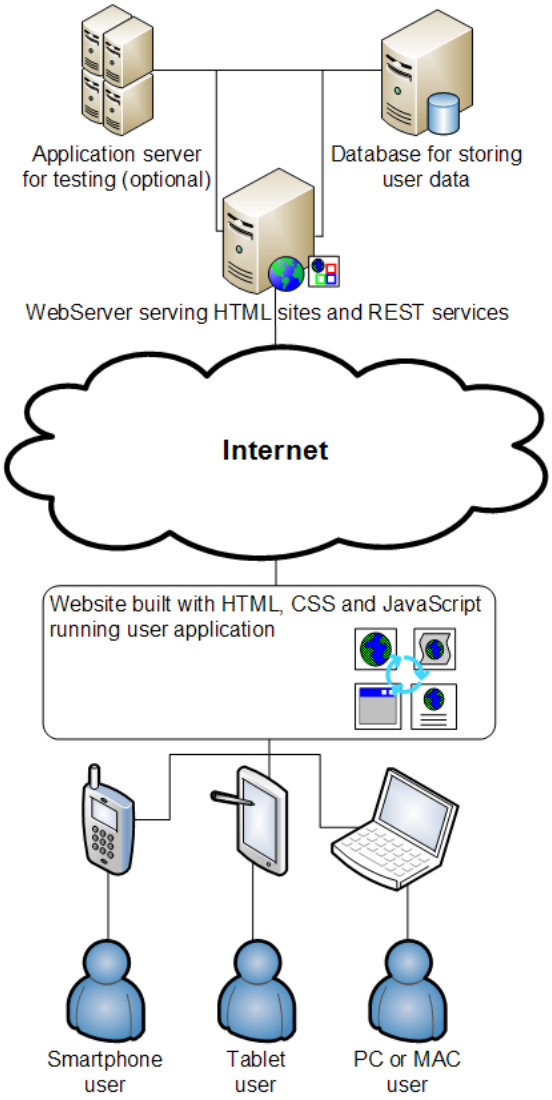

Figure 1: Overview of the platform architecture

An open-source implementation of the platform is currently under development. The Client is built using Google Web Toolkit framework. The communication between client and server is made using REST services and JSON data format. Server-side services are written in PHP, with MySQL used for data storage. The only exception to this rule is the server side evaluation of programs, which currently requires Java Runtime Environment. The system is being developed using continuous software development model, where each set of changes is published on a dedicated web site (http://www.orey.edu.pl) at certain time periods, just after acceptance tests.

CONCLUSION

The platform described above takes a modern concept of a learning management system specialized for computer programming with client-side code evaluation environment, and enhances it with an unprecedented level of gamification mechanisms. These mechanisms aim not only at improving students' motivation and engagement, but also encourage them to participate in various activities, they could otherwise neglect, like learning from instructional materials, practicing coding, taking part in programming competitions, tutoring and collaborating with other students. This way the students are better prepared to both cooperation and work under pres-

sure (of time or competitor), which should be advantageous in their professional careers.

The next step of our research will be to test how the platform works in a real educational environment.

REFERENCES

Amelung, M., Krieger, K., & Rosner, D. (2011). E-Assessment as a Service. *IEEE Transactions on Learning Technologies* 4(2), 162-174.

Betts, B. (2010, August 16). Social Is Not An Option. *Learning Solutions Magazine*. Retrieved from http://www.learningsolutionsmag.com/articles/506/social-is-not-an-option/pageall.

Code Avengers. (2012). Retrieved from http://www.codeavengers.com/.

Combéfis, S., & le Clément de Saint-Marcq, V. (2012). Teaching Programming and Algorithm Design with Pythia, a Web-Based Learning Platform, *Olympiads in Informatics* 6, 31-43.

Dehnadi, S., & Bornat, R. (2006). The camel has two humps. In *Little PPIG 2006*. Coventry,UK:Coventry University. Retrieved from http://www.cs.kent.ac.uk/dept_info/seminars/2005_06/paper1.pdf.

del Alamo, J. M., Alonso, A., Cortés, M. (2012). Automated Grading and Feedback Providing Assignments Management Module. In *ICERI2012 Proceedings* (pp. 3784-3793). Madrid, Spain: IATED.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference* (pp. 9-15). New York, NY, USA: ACM.

Fernandez, J. L. (2011). Automated assessment in a programming tools course. *IEEE Transactions on Education*, 54(4), 576-581.

Foord, M. (2012). *Try Python: Interactive Python Tutorial in the Browser*. Retrieved from http://www.trypython.org/.

Gåsland, M. M. (2011). *Game Mechanics based E-Learning*. Trondheim, Norway: Norwegian University of Science and Technology. Retrieved form http://daim.idi.ntnu.no/masteroppgave?id=6099.

Georgouli, K., & Guerreiro, P. (2011). Integrating an Automatic Judge into an Open Source LMS. *International Journal on E-Learning*, 10(1), 27-42.

Ghirardini, B. (2011). *E-learning methodologies*. Rome, Italy: Food and Agriculture Organization of the United Nations.

Guo, P. J. (2013). Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *Proceedings of the technical symposium on Computer science education*. Denver, CO, USA: ACM. Preprint retrieved from http://pgbovine.net/projects/pubs/guo-sigcse-preprint_2012-11-13.pdf.

Herger, M. (2012). *Gamification Platform Matrix*. Retrieved from http://enterprise-gamification.com/index.php/en/resources/platforms on November, 30, 2012.

Helminen, J., & Malmi, L. (2010). *Jype - a program visualization and programming exercise tool for Python*. In Proceedings of the 5th International Symposium on Software Visualization (pp. 153-162). New York, NY, USA: ACM.

Ihantola, P. (2011) *Automated Assessment of Programming Assignments: Visual Feedback, Assignment Mobility, and Assessment of Students' Testing Skills*. Espoo, Finland: Aalto University.

Kasyanov, V. N., & Kasyanova, E. V. (2009). A Web-Based System for Distance Learning of Programming. *Lecture Notes in Electrical Engineering* 27, 453-462.

Kosowski, A., Małafiejski, M., & Noiński T. (2008). Application of an online judge & contester in academic tuition. *Lecture Notes in Computer Science* 4823, 343-354.

Kumar, B., & Khurana, P. (2012). Gamification in education - learn computer programming with fun. *International Journal of Computers and Distributed Systems*, 2(1), 46-53.

Knewton (2012). *The Gamification of Education*. Retrieved from http://www.knewton.com/gamification-education/.
Lee, J. J., & Hammer, J. (2011). Gamification in Education: What, How, Why Bother? *Academic Exchange Quarterly*, 15(2), 146-151.
Leutenegger, S., & Edgington, J. (2007). A games first approach to teaching introductory programming. *ACM SIGCSE Bulletin* 39(1), 115-118.
Marczewski, A. (2012). *Gamification: A Simple Introduction*. Raleigh, NC:Lulu.
Miller, B., & Ranum, D. (2012). *Problem Solving with Algorithms and Data Structures*. Decorah, IA, USA: Runestone Interactive. Retrieved from http://interactivepython.org/courselib/static/pythonds/.
Moon, I., Han, S., Choi, K., Kim, D., Jeon, Ch., Lee, S., & Jeon, H. (2008). Virtual Education System for the C Programming Language. *Lecture Notes in Computer Science* 5145, 196-207.
Oncu, S., & Cakir, H. (2011). Research in online learning environments: Priorities and methodologies. *Computers & Education* 57, 1098-1108.
Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education* 13(2), 137-172.
Ryzac (2012). *Codeacademy*. Retrieved from http://www.codecademy.com/.
Skupas, B., & Dagiene, V. (2010). Observations from semi-automatic testing of program codes in the high school student maturity exam. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (pp. 31-36). New York, NY, USA: ACM.
Swacha, J. (2010). New concepts for teaching computer programming to future Information Technology engineers. In *Perspective technologies and methods in MEMS design* (pp. 188-191). Lviv, Ukraine:Lviv Politechnic National University.
Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J.P., de Castro, J.P., & Queirós, R. (2012). A distributed system for learning programming on-line. *Computers & Education* 58, 1-10.

Biographies

Jakub Swacha, PhD, is the head of Software Engineering Section of the Institute of Information Technology in Management at the University of Szczecin, Poland. He is an author of a Python textbook in Polish, and a co-author of the Rey educational programming language. Among his current scientific interests are novel methods of teaching programming and gamification.

Paweł Baszuro, MSc, is a freelancer, software developer and research enthusiast. He is particularly interested in developing programming languages and environments tailored to the needs of teaching varied students' groups. He is the main author of the Rey educational programming language. His latest projects concentrate on using gamification and social networking in teaching programming.