A Platform Fed by Software Industry Problems to Learn Software Development

Luiz Ferreira
Departamento de Ciências da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil
Email: luizpcf@dcc.ufmg.br

Eduardo Figueiredo
Departamento de Ciências da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil
Email: figueiredo@dcc.ufmg.br

***Abstract*—Nowadays, simply knowing the theoretical aspects of software engineering is not sufficient to remain competitive. People must also learn to apply their knowledge in new domains and practical situations to solve complex software development problems. Professionals holding different jobs in software industry must be creative and flexible problem solvers. This requires the ability to apply experience and practical knowledge to address novel problems. Consequently, learning to think critically, to analyze the problem description, and to synthesize information to solve software development problems in groups are crucial skills for successful participation in our modern and highly competitive software market. This work then proposes the development of a Web-based platform, named CodeChallenge, to teach people how to solve complex problems in software development by programming solutions for actual problems. The learning experience is going to be enhanced by students exchanging experience with peers in an environment that mimics a social network. From the business model point of view, it is possible to charge in a long term our industry partners based on their problems solved by our students.**

I. Introduction

In a society where many people cannot afford paying for education, it is important to create ways to facilitate access to education. With respect to Software Engineering, a field of study that always needs creative and flexible problem solvers, it is of upmost importance to create alternatives to teach not only theoretical aspects of software development, but also how to solve complex software development problems. In addition, creating a learning environment that favors critical thinking and helps people to boost their careers is not only relevant but also important and necessary.

The idea of learning from real-world problems is not new. For instance, there is web approaches to teach people new languages by making them translate documents and books [1]. The basic idea is simple in this case. Somebody who needs a document translation uploads it to a Web repository. That document then gets presented to students in a Web interface, and students can translate it in order to practice the language they are learning. When the document is fully translated, it is returned to the original content owner who, depending on the type of document they uploaded, pays for the translation. Similar approach has been applied to other areas. However, as far as we are concerned, there is anything like that to teach software engineering.

There exists today a large amount of high quality online courses that teach theoretical aspects of software engineering [2][3]. There are also initiatives to teach software development based on games and simulators [4]. However, unlike these approaches, we aim both to teach people how to program and to generate value from this produced code. The main idea is to create exercises based in actual problems from our industry partners. That is, specific problems that are part of the requirements of actual software systems are posted in a repository. Then, students are challenged to solve those specific problems and, as a result, deliver useful code to that system owner. In fact, several students are expected to solve the same problem. Therefore, we plan to add a ranking of best solutions based on users' recommendations in a social network environment. A similar ranking mechanism also applies to students based on their number of correct solutions to open problems. In summary, our general goal of the master dissertation is to teach software development based on actual industry problems to a large amount of people without costs for students.

II. CodeChallenge

In this work, we expect to create a platform of courses related to software engineering called CodeChallenge. This platform is composed of several courses aiming to teach software development in different programming languages, such as C#, Java, Ruby, JavaScript, and Python and different subjects of software engineering . The exercises of all courses are going to be designed as an implementation of real life problems, like Minimum Viable Product (MVP) or prototypes for startups and software products for small business. From the business model side, we foresee solving real life complex problems by breaking them into smaller problems which are solved by students. After all sub-problems have been solved, CodeChallenge aggregates one to another creating the solution for the original problem. Software Engineering courses will be available for free and, as a result, we believe that we are able to achieve a widely audience.

Figure 1 presents an overview of CodeChallenge. This figure shows that the proposed learning platform includes two types of users: (industry) partners and students. Our partners submit a real life problem of software development to CodeChallenge. Each open problem is then displayed for our students as an exercise that they can solve, based in what problems they already solved. CodeChallenge will enable students to solve problems by developing source code in its Web front end or offline in the desirable IDE. In the latter

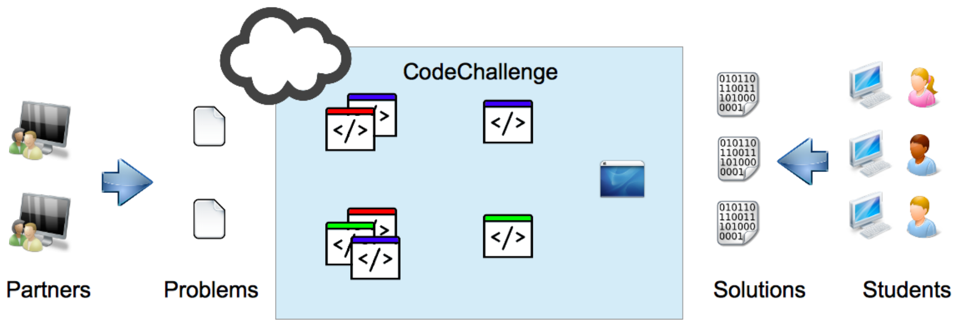

Fig. 1. CodeChallenge Architecture

case, students have to upload the source code with the problem solution. The verification of the quality of solutions is going to be performed in three steps. First, the learning platform is going to execute automated tests and those who pass in all tests are marked as likely correct. The platform will allow users to create more tests to guarantee that the solution proposed will solve the partner problem. After that, each likely correct solutions are going to be syntactically analyzed (e.g., to check code style and patterns) and quantitatively evaluated by source code metrics [5]. This automated analysis may approve or not the solution and will help the users to rank the better solution. Finally, users of the platform, i.e. students and partners, are able to qualitatively evaluate the quality of the approved solutions.

After a solution is solved by an user, CodeChallenge will allow and encourage users to do code reviews and improve code of others users. This will give us two benefits. First, will increase students skills helping them to understand how they can make better codes. Second, it will improve the overall solution that each partner will receive, improving code quality and maintenance.

This can introduce a new business model to massive open online courses [6] by developing a model based on creating small products to early stage companies. This approach will allow entrepreneurs to test their hypothesis faster at low cost.

The work has two milestone. First, it is expected to develop a teaching platform to allow worldwide students to practice their program skills in real world problems. This first milestone is planned for 2014. Second, both the learning and programming platforms are going to be integrated and evaluated based on empirical research methods, such as surveys and experiments [7].

This work can be divided in 3 phases. 1st Term: Development of a prototype module that allow students to do software engineering exercises and verifying automatically if the exercise is correct in the cloud. 2nd Term: Development of a module of the course to teach software engineering based on real life problems provided by our industry partners. 3rd Term: Design empirical studies to evaluate the proposed platform.

The first prototype version of CodeChallenge is going to be evaluated both in a controlled experiment with students in our university and by applying a survey with worldwide users [7]. We aim first to analyze if the platform is able to create an environment where students can write source code to solve exercises and verifies its correctness automatically. In a second step, CodeChallenge will be evaluated by analyzing whether it is capable of solving actual problems from industry partners and delivered value for them. Detailed evaluations strategies are expected to be designed during the work execution.

III. Conclusion and Expected Results

This project aims to help students older than 15 years old, who cannot pay a large amount for learning how to program or for improving their software development skills. CodeChallenge focuses on people either who are not ready yet to start their careers or have interests in learning new technologies. We believe that this project can impact the IT industry worldwide, by providing skilled manpower to an ever growing software market and by helping poor people to grow and enhance their programming skills.

We expect that at the end of the work, in December of 2015, Luiz Ferreira will be able to defend his master dissertation under the supervision of Eduardo Figueiredo.

IV. Acknowledgements

This work was partially supported by CNPq (grant Universal 485907/2013-5) and FAPEMIG (grants APQ-02532-12 and PPM-00382-14).

References

[1] L. von Ahn, “Duolingo: learn a language for free while helping to translate the web,” in *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 2013, pp. 1–2.

[2] E. Figueiredo, J. Pereira, L. Garcia, and L. Lourdes, “On the evaluation of an open software engineering course,” *44th Annual Frontiers in Education (FIE) Conference*.

[3] D. C. Schmidt and Z. McCormick, “Creating and teaching a mooc on pattern-oriented software architecture for concurrent and networked software,” in *Proceedings of the WaveFront Forum at the SPLASH 2013 conference*, 2013.

[4] N. Tillmann, J. De Halleux, T. Xie, S. Gulwani, and J. Bishop, “Teaching and learning programming and software engineering via interactive gaming,” in *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013, pp. 1117–1126.

[5] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, 1994.

[6] C. Dellarocas and M. Van Alstyne, “Money models for moocs,” *Communications of the ACM*, vol. 56, no. 8, pp. 25–28, 2013.

[7] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer, 2012.