Educational Software Engineering: Where Software Engineering, Education, and Gaming Meet

Tao Xie
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
xie@csc.ncsu.edu

Nikolai Tillmann
Microsoft Research
One Microsoft Way
Redmond, WA, USA
nikolait@microsoft.com

Jonathan de Halleux
Microsoft Research
One Microsoft Way
Redmond, WA, USA
jhalleux@microsoft.com

***Abstract*—We define and advocate the subfield of educational software engineering (i.e., software engineering for education), which develops software engineering technologies (e.g., software testing and analysis, software analytics) for general educational tasks, going beyond educational tasks for software engineering. In this subfield, gaming technologies often play an important role together with software engineering technologies. We expect that researchers in educational software engineering would be among key players in the education domain and in the coming age of Massive Open Online Courses (MOOCs). Educational software engineering can and will contribute significant solutions to address various critical challenges in education especially MOOCs such as automatic grading, intelligent tutoring, problem generation, and plagiarism detection. In this position paper, we define educational software engineering and illustrate Pex for Fun (in short as Pex4Fun), one of our recent examples on leveraging software engineering and gaming technologies to address educational tasks on teaching and learning programming and software engineering skills.**

I. Introduction

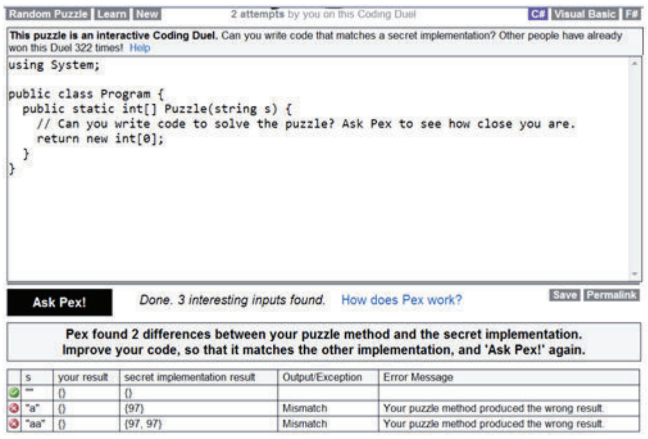
Among various subfields of software engineering, software engineering education [14] has been an important one, focusing on education topics for software engineering (e.g., how to better teach and train software engineering skills). Typically research work on software engineering education does not appear in research tracks of major software engineering conferences but appear in their education tracks or conferences with focus on software engineering education. For example, the International Conference on Software Engineering (ICSE)[1] typically has a track on Software Engineering Education. The ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) has also recently included a co-located Educator's Symposium[2]. The Conference on Software Engineering Education and Training (CSEE&T)[3] has focused on software engineering education and training since 1987. Indeed, research work on software engineering education sometimes also appears in venues in computer science education such as the SIGCSE Technical Symposium (SIGCSE)[4] and the Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)[5].

In this position paper, we define and advocate the subfield of educational software engineering (i.e., software engineering for education) within software engineering research. This subfield develops software engineering technologies (e.g., software testing and analysis [7], software analytics [23], [24]) for general educational tasks, going beyond educational tasks for software engineering. For example, general educational tasks can even be on teaching maths [1], [6], [15]. As an analogy, data mining for software engineering [21] (also called mining software repositories [8]) leverages data mining technologies (which typically come from the data mining community) to address tasks in software engineering, whereas educational software engineering leverages software engineering technologies (which typically come from the software engineering community) to address tasks in education. In addition, in the solution space, gaming technologies often play an important role together with software engineering technologies.

We expect that researchers in educational software engineering would be among key players in the education domain and in the coming age of Massive Open Online Courses (MOOCs) [3], [11], which have recently gained high popularity among various universities and even in global societies. Educational software engineering can and will contribute significant solutions to address various critical challenges in education especially MOOCs such as automatic grading [16], [19], intelligent tutoring [12], problem generation [1], [6], [15], and plagiarism detection [10], [13].

To provide a concrete example of educational software engineering, in this position paper, we illustrate Pex for Fun [19] (in short as Pex4Fun), one of our recent examples on leveraging software engineering and gaming technologies to address educational tasks on teaching and learning programming and software engineering skills. In particular, our illustration of Pex4Fun focuses on the gaming (Section II), social dynamics (Section III), educational usage (Section IV), and software engineering technologies (Section V), being four common aspects of a typical project on educational software engineering.

[1]http//www.icse-conferences.org/
[2]http://www.splashcon.org/history/
[3]http://conferences.computer.org/cseet/
[4]http://www.sigcse.org/events/symposia
[5]http://www.sigcse.org/events/iticse



This puzzle is an interactive Coding Duel. Can you write code that matches a secret implementation? Other people have already won this Duel 322 times! Help

```
using System;

public class Program {
  public static int[] Puzzle(string s) {
    // Can you write code to solve the puzzle? Ask Pex to see how close you are.
    return new int[0];
  }
}
```

Done. 3 interesting inputs found.

Pex found 2 differences between your puzzle method and the secret implementation. Improve your code, so that it matches the other implementation, and 'Ask Pex!' again.

	s	your result	secret implementation result	Output/Exception	Error Message
✓	""	{}	{}		
✗	"a"	{}	{97}	Mismatch	Your puzzle method produced the wrong result.
✗	"aa"	{}	{97, 97}	Mismatch	Your puzzle method produced the wrong result.

Fig. 1. The user interface of the Pex4Fun website

II. Gaming in Pex4Fun

Pex4Fun [19] (http://www.pexforfun.com/) is an interactive-gaming-based teaching and learning platform for .NET programming languages such as C#, Visual Basic, and F#. Figure 1 shows a screen snapshot of the user interface of the Pex4Fun website, which shows an example coding duel under solving by a player. It is a browser-based teaching and learning environment with target users as teachers, students, and even software practitioners, etc.

The core type of Pex4Fun games is a *coding duel* where the player has to solve a particular programming problem. A coding duel created by a game creator (who could be any user of Pex4Fun) consists of two methods with the same method signature and return type[6]. One of these two methods is the secret (golden) implementation, which is not visible to the player. The other is the player implementation, which is visible to the player and can be an empty implementation or a faulty implementation of the secret implementation. The player implementation can include optional comments to give the player some hints in order to reduce the difficulty level of gaming.

After a player selects a coding-duel game to play, the player's winning goal is to modify the player implementation (visible to the player, shown in the upper part of Figure 1) to make its behavior (in terms of the method inputs and results) to be the same as the secret implementation (not visible to the player). Apparently, without any feedback or help, the player has no way to guess how the secret implementation would behave. The player can get some feedback by clicking the button "Ask Pex" (shown in the middle-left part of Figure 1) to request the following two types of feedback: (1) under what sample method input(s) the player implementation and the secret implementation have the same method result and (2) under what sample method input(s) the player implementation and the secret implementation have different method results. Example feedback is shown in the table near the bottom of Figure 1. In the table, the first line prefixed with a green circle indicates the first type of feedback, and the second and third lines prefixed with a red circle indicate the second type of feedback.

[6]The method signature of a coding duel must have at least one input parameter. The return type of a coding duel must not be void.

As described in Section V, Pex4Fun leverages the underlying test-generation engine called Pex [18], [22] to generate such feedback and determine whether the player wins the game: the player wins the game if the test-generation engine cannot generate any method input to cause the player implementation and the secret implementation to have different method results.

The design of coding-duel games and the gaming platform follows a number of design principles [19]. For example, the games need to be interactive and the interactions need to be iterative and involve multiple rounds. The feedback given to the player should be adaptive and personalized to the modifications made by the player on the player implementation. The games should have a clear winning criterion. There should be no or few opportunities for the player to cheat the games (e.g., by adding very complicated code portions in the player implementation to pose difficulties for the underlying test-generation engine).

III. Social Dynamics in Pex4Fun

To add more fun to Pex4Fun, we have developed a number of features related to social dynamics, making games in Pex4Fun a type of social games. For example, Pex4Fun allows a player to learn what coding duels other people were already able to win (or not). For a given coding duel opened by a player, the description text box above the working area shows some statistic such as "Can you write code that matches a secret implementation? Other people have already won this Duel 322 times!", as shown in Figure 1.

Ranking of players and coding duels. Initially, when only a relatively small number of coding duels were provided by us in Pex4Fun, we provided a mechanism of earning medals to encourage users to play coding duels. After signing in, a user could earn virtual medals for winning coding duels. The user got the first medal for winning any five of the coding duels that were built into Pex4Fun. The user got the second medal for winning another 20 of the built-in coding duels.

Furthermore, a user can click the "Community" link on the Pex4Fun main page to see how the user's coding duel skills compare to other users. In the community area (http://www.pexforfun.com/Community.aspx), there are two ranked lists of all users (one based on the number of points earned by a user and the other one based on the number of coding duels won by a user), as well as coding duels that other users have published. A user can earn points by winning a coding duel, rating a coding duel that the user won, registering in a course, creating a coding duel that somebody else attempts to win, creating a coding duel that somebody else wins, etc. Note that a user can rate any coding duel that the user wins as "Fun", "Boring", or "Fishy". All ratings are shared with the community.

Live feeds. A player can click the "Live Feed" link on the Pex4Fun main page to see what coding duels other players are winning (or not) right now (http://www.pexforfun.com/

Livefeed.aspx). Maybe someone else is trying to win a coding duel that the player has created or the player is also trying to win.

Social dynamics in Pex4Fun share similar motivations as other recent gamification examples in software engineering. For example, Stack Overflow badges[7] have been used to provide incentives for Stack Overflow users to ask or answer questions there. Through asking or answering questions, a user earns reputation points. For example, 10 reputation points are given to a user when the user's answer to a question receives an "up" vote. In addition, a user can earn three ranks of badge: bronze, silver, and gold badges. Bronze badges are given to users who often help teach users on how to use the system. Silver badges are given to users who post very insightful questions and answers, and show dedication to moderate and improve the Stack Overflow contents. Gold badges are given to users who demonstrate outstanding dedication or achievement. Such badges earned by a user appear on the user's profile and in the user's user card. Along a similar spirit, early 2012, Microsoft added a new plug-in to the Microsoft Visual Studio to allow software developers to unlock achievements[8], receive badges, and increase their ranking on a leaderboard based on the program code that they have written.

IV. Educational Usage of Pex4Fun

The game type of coding duels within Pex4Fun is flexible enough to allow game creators to create various games to target a range of skills such as skills of programming, program understanding, induction, debugging, problem solving, testing, and specification writing, with different difficulty levels of gaming. In addition, Pex4Fun is an open platform: any one around the world can create coding duels for others to play besides playing existing coding duels themselves. Teachers can create virtual classrooms in the form of courses by customizing existing learning materials and games or creating new materials and games. Teachers can also enjoy the benefits of automated grading of exercises assigned to students. Pex4Fun has provided a number of open virtual courses including learning materials along with games used to reinforce students' learning (http://www.pexforfun.com/Page.aspx#learn/courses).

Pex4Fun was adopted as a major platform for assignments in a graduate software engineering course. A coding-duel contest was held at a major software engineering conference (ICSE 2011) for engaging conference attendees to solve coding duels in a dynamic social contest. Pex4Fun has been gaining high popularity in the community: since it was released to the public in June 2010, the number of clicks of the "Ask Pex!" button (indicating the attempts made by users to solve games at Pex4Fun) has reached over one million (1,135,931) as of March 3, 2013.

Various Pex4Fun users posted their comments on the Internet to express their enthusiasm and interest (even addiction) to Pex4Fun. Here we included some examples. "PEX4fun could become a better FizzBuzz than FizzBuzz.", "it really got me *excited*. The part that got me most is about spreading interest in/teaching CS: I do think that it's REALLY great for teaching | learning!" "Frankly this is my favorite game. I used to love the first person shooters and the satisfaction of blowing away a whole team of Noobies playing Rainbow Six, but this is far more fun.", "Teaching, learning - isn't this really the same, in the end? In fact, for me personally, it's really about leveraging curiosity, be it mine or someone else's - at best both! And PexForFun (+ all the stuff behind) is a great, promising platform for this: you got riddles, you got competition, you get feedback that makes you think ahead...". "I'm afraid I'll have to constrain myself to spend just an hour or so a day on this really exciting stuff, as I'm really stuffed with work"."PexForFun improves greatly over projecteuler w.r.t. how proposed solutions are verified; in fact what it adds is that you don't just get a 'nope' but something more articulate, something you can build on. That's what I think is really great and exciting - let's push it even further now!"

V. Software Engineering Technologies in Pex4Fun

Behind the scenes on the server in the cloud, the Pex4Fun website uses dynamic symbolic execution [5] implemented by Pex [18], [22], in order to determine the game progress of the player and to compute customized feedback [17]. Pex is an automatic white-box test generation tool for .NET. It has been integrated into Microsoft Visual Studio as an add-in. Besides being used in industry, Pex was also used in classroom teaching at different universities [20].

In particular, dynamic symbolic execution (DSE) [5] is a variation of symbolic execution [2], [9] and leverages runtime information from concrete executions. DSE is often conducted in iterations to systematically increase code coverage such as block or branch coverage. In each iteration, DSE executes the program under test with a test input, which can be a default or randomly generated input in the first iteration or an input generated in one of the previous iterations. During the execution of the program under test, DSE performs symbolic execution in parallel to collecting symbolic constraints on program inputs obtained from predicates in branch statements along the execution. Then DSE flips a branching node in the executed path to construct a new path that shares the prefix to the node with the executed path, but then deviates and takes a different branch. Finally, DSE relies on a constraint solver to compute a satisfying assignment (if possible), which forms a new test input whose execution will follow the flipped path.

VI. Discussion

Educational software engineering is closely related to the field of educational games [4] (i.e., games for education), with example conferences such as the Games+Learning+Society Conference[9] and example initiatives such as the MacArthur Digital Media and Learning initiative[10]. The field of educational games typically focuses on gaming technologies for

[7]http://stackoverflow.com/badges
[8]http://channel9.msdn.com/achievements/visualstudio
[9]http://www.gameslearningsociety.org/conference/
[10]http://www.macfound.org/programs/learning/

supporting educational purposes, whereas educational software engineering typically focuses on software engineering technologies for supporting educational purposes. In the context of Pex4Fun, the field of educational games would focus more on the aspect of gaming (Section II) whereas the field of educational software engineering would focus more on the aspect of software engineering technologies (Section V). Note that educational software engineering deals with not only educational games but also other educational tools not being games.

In addition, it is reasonable to consider that software engineering for developing educational games or generally educational tools (such as software quality assurance for educational-game software) would be part of educational software engineering. In other words, educational software engineering is not limited to software engineering technologies as infrastructure support for educational tools (as exemplified by Pex4Fun), and can also include software engineering tools or processes to assist the development of educational tools.

We advocate educational software engineering to be within software engineering research, and to contribute to software engineering research in three example ways. First, when targeting at educational tasks, researchers may be able to leverage or develop software engineering technologies (to be effective for such tasks), which generally may not be effective or mature enough to deal with tasks related to software industry. An example case would be developing program-synthesis technologies for educational tasks [1], [6], [15]. Another example case would be developing test-generation technologies for Pex4Fun, since secret implementations created for coding duels tend to be simpler than real-world code implementations. Second, targeting at educational tasks may pose unique requirements for software engineering technologies. For example, test generation for software engineering tasks such as achieving code coverage aims at generating and reporting test inputs that can achieve new code coverage. However, test generation for Pex4Fun aims at generating and reporting test inputs that can serve as feedback to achieve effective learning purposes. Third, some educational tasks (such as intelligent tutoring [12] and problem generation [1], [6], [15]) call for creation of new software engineering technologies, which may not exist in traditional software engineering (because there are no counterparts in the software engineering domain for such tasks in the education domain).

VII. Conclusion

In this position paper, we have defined and advocated educational software engineering as an emerging subfield of software engineering. Educational software engineering develops software engineering technologies for general educational tasks. In this subfield, gaming technologies often play an important role together with software engineering technologies. We have presented Pex4Fun, one of our recent examples on leveraging software engineering and gaming technologies for teaching and learning programming and software engineering skills. Pex4Fun can be also used in the context of Massive Open Online Courses (MOOCs) to address issues such as automatic grading.

Acknowledgment

We thank reviewers for their valuable feedback. Tao Xie's work is supported in part by NSF grants CCF-0845272, CCF-0915400, CNS-0958235, CNS-1160603, an NSA Science of Security Lablet grant, a NIST grant, a Microsoft Research Software Engineering Innovation Foundation Award, and NSF of China No. 61228203.

References

[1] E. Andersen, S. Gulwani, and Z. Popovic. A trace-based framework for analyzing and synthesizing educational progressions. In *Proc. CHI*, 2013.

[2] L. A. Clarke. A system to generate test data and symbolically execute programs. *IEEE Trans. Softw. Eng.*, 2(3):215–222, 1976.

[3] A. Fox and D. Patterson. Crossing the software education chasm. *Communications of the ACM*, 55(5):44–49, 2012.

[4] J. P. Gee. *What Video Games Have to Teach Us About Learning and Literacy.* Palgrave Macmillan, 2007.

[5] P. Godefroid, N. Klarlund, and K. Sen. DART: Directed automated random testing. In *Proc. PLDI*, pages 213–223, 2005.

[6] S. Gulwani, V. Korthikanti, and A. Tiwari. Synthesizing geometry constructions. In *Proc. PLDI*, pages 50–61, 2011.

[7] M. J. Harrold. Testing: a roadmap. In *Proc. FOSE*, pages 61–72, 2000.

[8] A. E. Hassan. The road ahead for mining software repositories. In *Proc. FoSM*, 2008.

[9] J. C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, 1976.

[10] C. Liu, C. Chen, J. Han, and P. S. Yu. GPLAG: detection of software plagiarism by program dependence graph analysis. In *Proc. KDD*, pages 872–881, 2006.

[11] K. Masters. A brief guide to understanding MOOCs. *The Internet Journal of Medical Education*, 1, 2011.

[12] T. Murray. Authoring intelligent tutoring systems: an analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 1(10):98–129, 1999.

[13] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proc. SIGMOD*, pages 76–85, 2003.

[14] M. Shaw. Software engineering education: a roadmap. In *Proc. FOSE*, pages 371–380, 2000.

[15] R. Singh, S. Gulwani, and S. Rajamani. Automatically generating algebra problems. In *Proc. AAAI*, 2012.

[16] R. Singh, S. Gulwani, and A. Solar-Lezama. Automated feedback generation for introductory programming assignments. In *Proc. PLDI*, 2013.

[17] K. Taneja and T. Xie. DiffGen: Automated regression unit-test generation. In *Proc. ASE*, pages 407–410, 2008.

[18] N. Tillmann and J. de Halleux. Pex-white box test generation for .NET. In *Proc. TAP*, pages 134–153, 2008.

[19] N. Tillmann, J. D. Halleux, T. Xie, S. Gulwani, and J. Bishop. Teaching and learning programming and software engineering via interactive gaming. In *Proc. ICSE, Software Engineering Education (SEE)*, 2013.

[20] T. Xie, J. de Halleux, N. Tillmann, and W. Schulte. Teaching and training developer-testing techniques and tool support. In *Proc. SPLASH, Educators' and Trainers' Symposium*, pages 175–182, 2010.

[21] T. Xie, S. Thummalapenta, D. Lo, and C. Liu. Data mining for software engineering. *IEEE Computer*, 42(8):35–42, August 2009.

[22] T. Xie, N. Tillmann, P. de Halleux, and W. Schulte. Fitness-guided path exploration in dynamic symbolic execution. In *Proc. DSN*, pages 359–368, 2009.

[23] D. Zhang, Y. Dang, J.-G. Lou, S. Han, H. Zhang, and T. Xie. Software analytics as a learning case in practice: Approaches and experiences. In *Proc. MALETS*, 2011.

[24] D. Zhang and T. Xie. Software analytics in practice: Mini tutorial. In *Proc. ICSE, Software Engineering in Practice, Mini Tutorial*, page 997, 2012.