CHAPTER 5

Educational Software Engineering

Where Software Engineering, Education, and Gaming Meet

Tao Xie, Nikolai Tillmann,
Jonathan de Halleux, and Judith Bishop

CONTENTS

5.1 INTRODUCTION

Among various subfields of software engineering, software engineering education [1] has been an important one, focusing on educational 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 appears in their education tracks or conferences with focus on software engineering education. For example, the International Conference on Software Engineering (ICSE; http://www.icse-conferences.org) typically has a track on software engineering education. The ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications has also recently included a colocated Educator's Symposium (http://www.splashcon.org/history). The Conference on Software Engineering Education and Training (http://conferences.computer.org/cseet) has focused on software engineering education and training since 1987. Indeed, research work on software engineering education sometimes also appears in meetings on computer science education, such as the SIGCSE Technical Symposium (http://www.sigcse.org/events/symposia) and the Annual Conference on Innovation and Technology in Computer Science Education (http://www.sigcse.org/events/iticse).

In this chapter, we define and advocate the subfield of educational software engineering (i.e., software engineering for education) within the domain of software engineering research. This subfield develops software engineering technologies (e.g., software testing and analysis [2], software analytics [3,4]) for general educational tasks, going beyond educational tasks for software engineering. For example, general educational tasks can even be on teaching math [5–7]. As an analogy, data mining for software engineering [8] (also called mining software repositories [9]) 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 the key players in the education domain and in the coming age of massively open online courses (MOOCs) [10,11], 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 [12,13], intelligent tutoring [14], problem generation [5–7], and plagiarism detection [15,16].

To provide a concrete example of educational software engineering, in this chapter,* we first lay out the background on online programming exercise systems by describing a number of existing representative systems. As a concrete example of gamificating an online programming exercise system, we illustrate Pex for Fun [13] (Pex4Fun, for short), which leverages 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 its underlying software engineering technologies (Section 5.3.1), its gaming (Section 5.3.2), social dynamics (Section 5.3.3), and educational usage (Section 5.3.4), which are the four common aspects of a typical project on educational software engineering. We also describe Code Hunt (Section 5.3.5), which is a recent gaming platform evolved from Pex4Fun.

5.2 BACKGROUND: ONLINE PROGRAMMING EXERCISE SYSTEMS

5.2.1 CodingBat

CodingBat (http://codingbat.com/), created by Nick Parlante, is an online platform for providing a set of programming exercises in Java and Python. Note that some of the exercises in CodingBat are equipped with hint text and/or the solution code. We next use an example (http://codingbat.com/prob/p146974) to illustrate how a student can solve an exercise problem in CodingBat. In this example, the web page for the exercise problem includes a short natural language problem statement: "Given an array of scores, return true if each score is equal or greater than the one before.

* This chapter significantly extends a previous short article [31] presented in the *3rd International Workshop on Games and Software Engineering* (2013). This new extension in this chapter primarily includes surveying-related online programming exercise systems (Section 5.2); restructuring and enriching the description of Pex4Fun as an example for gamificating an online programming exercise system, adding the description of Code Hunt (Section 5.3.5), a recent gaming platform evolved from Pex4Fun; and discussing additional more recent related work. AQ 1

The array will be length 2 or more." In addition, the web page also includes a table for showing a small number of sample expected input/output pairs:

```
scoresIncreasing({1, 3, 4}) → true
scoresIncreasing({1, 3, 2}) → false
scoresIncreasing({1, 1, 4}) → true
```

Then the code editor in the middle of the web page includes an empty method body for `public boolean scoresIncreasing(int[] scores)` (note that when a problem creator creates a problem, the method, such as `scoresIncreasing` in the problem, should return a value). A student who tries to solve the problem is expected to fill in code in the empty method body to solve the given programming problem. After the student fills in code and clicks the Go button, CodingBat displays compilation issues, if any, that are encountered when the code is compiled; otherwise, CodingBat runs a predefined set of test cases (prepared by the problem creator) against the code and reports these test cases being labeled as failing test cases or passing test cases. Note that these test cases are reported in the form of the preceding example expected input/output pairs. Based on the feedback (i.e., the reported test cases and their failing/passing statuses), the student can further attempt to improve his or her code to make all test cases as passing test cases.

Because the predefined set of test cases is visible to the student, the student can "fool" CodingBat by writing code that includes a conditional statement for each reported test case so that the conditional in the conditional statement checks whether the method arguments are the same as the input in a reported test case and then the true branch of the conditional statement simply returns the expected output in the reported test case. Apparently, the code written in this way to overfit the reported test cases is not the correct code for the exercise problem. However, CodingBat would still report All Correct because all the predefined test cases are indeed passing test cases.

CodingBat allows the student to view the progress graph for a problem, showing the problem-solving history (e.g., the percentage of passing test cases and percentage of failing test cases) for each version of the code written and submitted by the student for the problem over time. The student can also view graphs from some random users just for fun. A student can earn *code badges* by solving problems across all of the fundamental sections (which include very common code patterns that often come up at coding, such as problems related to strings, arrays, and logic). For example, a student earns a five-star badge when the student solves three problems in each fundamental section. The student can share his or her account with a "teacher" account, from which the associated teacher can view the problem-solving statistics of the student along with the code written by the student for each problem. However, the teacher is not suggested to use CodingBat as a grading platform of exams or homework assignments, but is suggested to leverage CodingBat as a practice platform for students.

5.2.2 CloudCoder

CloudCoder [17] (http://cloudcoder.org/) and *CodeWrite* [18] (http://codewrite.cs.auckland.ac.nz/) are two systems closely related to CodingBat. CloudCoder is an open-source web-based programming exercise system with exercises in C/C++, Java, Python, and Ruby. CloudCoder provides similar mechanisms as CodingBat's for students to solve problems based on the testing results against a predefined set of test cases. However, CodeWrite allows a student to construct an exercise problem along with the test cases for the problem so that other peer students can solve the exercise problem (in the same way as solving an exercise problem in CodingBat or CloudCoder). Note that in CodingBat or CloudCoder, only teachers or platform providers (not students) are supposed to construct exercises.

5.2.3 Practice-It

Practice-It (http://practiceit.cs.washington.edu/) is an online platform for students to practice solving Java programming problems. Many of the problems were drawn from the University of Washington's introductory programming courses. A student can select a problem from the list of problems organized by chapter topics of a programming textbook or section topics of the University of Washington's introductory programming courses. Once granted permission by the platform administrators, users of the platform can also create and upload a problem. If the problem creator defines some constraints for the problem, Practice-It first checks the student's code against such constraints and reports constraint violation errors such as the following: Error: Your solution doesn't meet one or more constraints required by this problem. Your solution must use a static method. (must appear 2 times in your solution). If the student's code does not compile, Practice-It reports repairing hints based on the compilation errors. After the student's code compiles, Practice-It runs a predefined set of test cases (prepared by the problem creator) against the student's code. Then Practice-It reports a table that includes testing information for each test case: the test input, expected output, actual output (produced by the student's code), and result (pass or fail). When the expected output is different from the actual output, the result is fail; the different outcomes of the actual output and the expected output are also reported.

5.2.4 CodeLab

CodeLab (http://www.turingscraft.com/), providing paid access of full exercises to students, is a web-based programming exercise system with exercises in Python, Java, C++, C, JavaScript, C#, VB, and SQL. CodeLab provides short exercises, each of which typically focuses on a programming concept or language construct. Different from other related systems, CodeLab does not report any explicit test cases (i.e., input/output pairs) to a student after the student submits code for a given exercise problem. Instead, CodeLab informs the student whether his or her submitted code is correct (the correctness judgment of the code seems to be based on running a predefined set of test cases, without considering code elegance or efficiency). If incorrect, CodeLab additionally informs the student of repairing hints such as likely locations of faulty code portions and hint sentences, for example, You almost certainly should be using:/. These repairing hints seem to be identified based on syntactic comparison of the submitted code and the solution code. CodeLab provides no feedback to the student in terms of specific correct (or incorrect) input/output behaviors of the submitted code. CodeLab organizes exercises in a sequence related to a programming concept or language construct (typically 3–10 exercises in a sequence). The exercises included in a sequence are of gradually increasing sophistication. The teacher of a class is suggested to allocate 5%–10% of a student's class grade to be correct completion of the CodeLab exercises. Besides leveraging the existing exercises in CodeLab, a teacher can create additional exercises in CodeLab for his or her class to use.

5.2.5 Codecademy

Codecademy (http://www.codecademy.com/) is an online interactive platform that offers free programming classes in Python, PHP, jQuery, JavaScript, Ruby, HTML and CSS. In a browser, on the left-hand side, a student is provided short texts that illustrate a programming knowledge point and instructions for the student to carry out a related programming exercise in the online code editor displayed in the middle of the browser.

The instructions also include a hint portion that can be viewable only after the student clicks Stuck? Get a hint After the student finishes writing code in the code editor following the instructions and then clicks the bottom Save and Submit Code, Codecademy assesses the written code against the instructions (based on checking the outputs of the code against the predefined outputs of the exercise); if the code is incorrect, Codecademy provides a simple hint sentence to the student, such as Did you include two console.log()s in your code? (This hint is based on syntactic differences of the student's code and the solution code.)

AQ 2

Note that, different from programming exercise systems (such as CodingBat) in which code written by students needs to be in the form of a method with non-void return, CodeLab and Codecademy allow code written by students to be just one or multiple lines of code.

5.2.6 BetterProgrammers

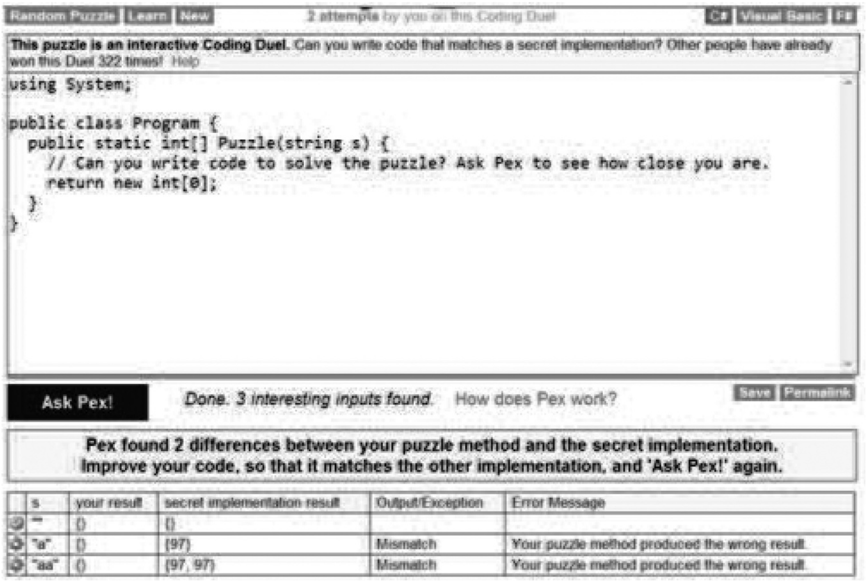
BetterProgrammers (http://www.betterprogrammer.com/) is an online platform for Java programmers to solve a sequence of programming tasks with increasing complexities. Instead of focusing on training, the platform focuses on assessing and certifying programmers so that companies can leverage such certification information in interviewing and hiring programmer candidates. The top 50 programmers ranked by the platform are posted on the front page of the platform website. The platform does not provide rich code editor but just requests programmers to copy the code skeleton embedded with the task description as code comments (from the simple code editor in the platform) to the programmers' favorite Java IDE, finish the programming task, copy the completed code for the task back to the simple code editor in the platform, and submit the completed code. It is unclear how BetterProgrammers checks the correctness of the submitted code. For each programming task, the recommended time and maximum time for task completion are listed along with the elapsed task time in real time.

5.2.7 Discussion

Software engineering technologies underlying the existing online programming exercise systems [19] are typically simple. For example, a simple testing technique (i.e., running a predefined set of test cases against the code submitted by a student to check the code's correctness) is commonly used in these existing systems. Some systems such as Practice-It and CodeLab seem to use lightweight static program analysis to check the code submitted by students against some predefined constraints or against the correct solution code to give the students repairing hints. Gamification does not play an explicit role in designing these existing systems. Some systems provide some social dynamics: a user can view progress graphs from some random users (CodingBat), a user can earn code badges (CodingBat), a user can construct exercise problems for other users to solve (CodeWrite), and users are ranked (BetterProgrammers). All these systems place heavy emphasis on their educational value.

5.3 PEX4FUN: GAMIFICATION OF AN ONLINE PROGRAMMING EXERCISE SYSTEM

In this section, we present Pex4Fun [13] (http://www.pexforfun.com/), an example of gamificating an online programming exercise system based on software engineering technologies. In particular, Pex4Fun is an interactive gaming-based teaching and learning platform for .NET programming languages such as C#, Visual Basic, and F#. Figure 5.1 shows a screen snapshot of the user interface of the Pex4Fun website, which includes an example coding duel under solving by a player. It is a browser-based teaching AQ 3 and learning environment [20] with target users as teachers, students, and



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!

```
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.

FIGURE 5.1 The user interface of the Pex4Fun website.

even software practitioners. In particular, in a coding duel game, a major game type in Pex4Fun, a student needs to write the code to implement the functionalities of a secret specification (in the form of sample solution code not visible to the student). Based on an automated test generation tool, Pex4Fun finds any discrepancies in behavior between the student's code and the secret specification. Such discrepancies are given as feedback to the student to guide how to fix the student's code to match the behavior of the secret specification.

We next illustrate Pex4Fun by focusing on its underlying software engineering technologies (Section 5.3.1), its gaming (Section 5.3.2), social dynamics (Section 5.3.3), and educational usage (Section 5.3.4), which are the four common aspects of a typical project on educational software engineering. Finally, we describe Code Hunt (Section 5.3.5), which is a recent gaming platform evolved from Pex4Fun.

5.3.1 Software Engineering Technologies Underlying Pex4Fun

Behind the scenes on the server in the cloud, the Pex4Fun website leverages dynamic symbolic execution (DSE) [21] implemented by Pex [22,23], in order to (1) determine whether the code submitted by a student is correct, (2) check the code's correctness and game progress of the player, and (3) compute customized feedback [24]. 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 the industry, Pex was also used in classroom teaching at different universities [25]. AQ 4

In particular, DSE [21] is a variation of symbolic execution [26,27] and leverages run-time 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.

5.3.2 Gaming in Pex4Fun

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.* 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 5.1) 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 5.1) 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. An example feedback is shown
AQ 5 in the table near the bottom of Figure 5.1. In the table, the first line prefixed indicates the first type of feedback, and the second and third lines indicate the second type of feedback.

As described in Section 5.3.1, Pex4Fun leverages the underlying test generation engine called Pex [22,23] 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 [13]. 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 method signature of a coding duel must have at least one input parameter. The return type of a coding duel must not be void.

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).

5.3.3 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 statistics such as the following: Can you write code that matches a secret implementation? Other people have already won this Duel 322 times! (see Figure 5.1).

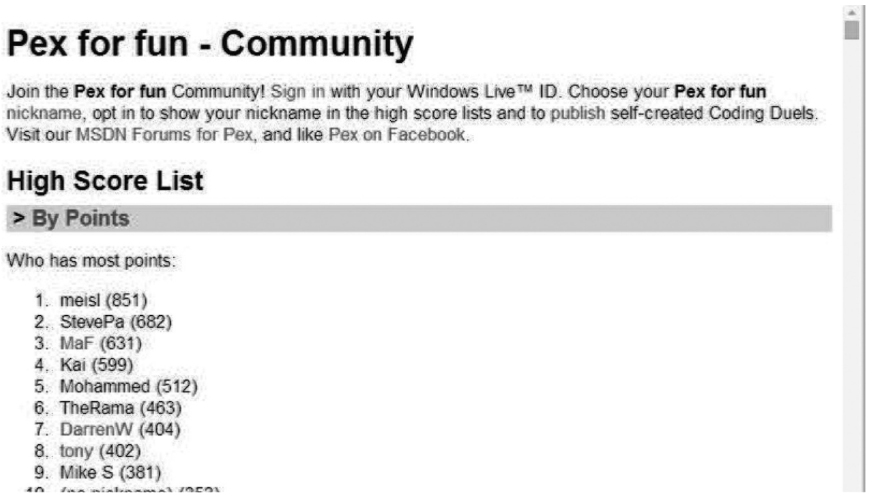
5.3.3.1 Ranking of Players and Coding Duels

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 with 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. Figure 5.2 shows the ranked list of all users based on their earned points. 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, and so on. 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.

5.3.3.2 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



Pex for fun - Community

Join the **Pex for fun** Community! Sign in with your Windows Live™ ID. Choose your **Pex for fun** nickname, opt in to show your nickname in the high score lists and to publish self-created Coding Duels. Visit our MSDN Forums for Pex, and like Pex on Facebook.

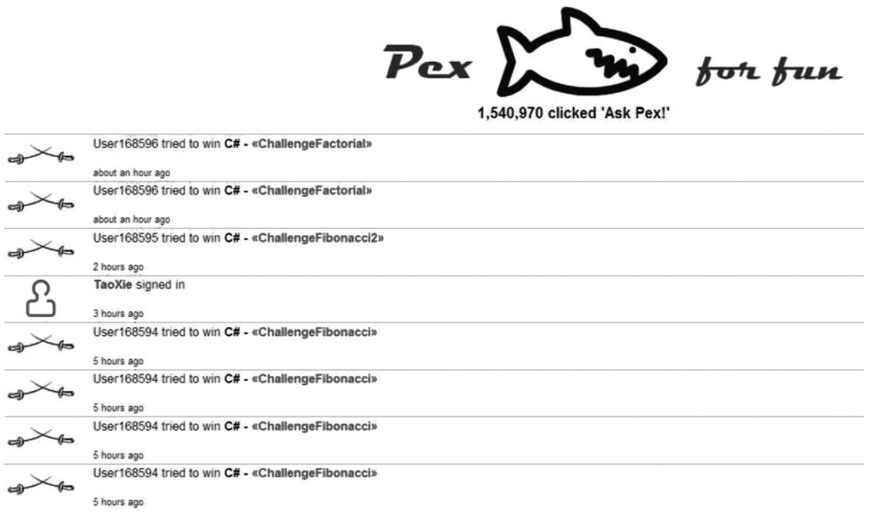
High Score List

> By Points

Who has most points:

1. meisl (851)
2. StevePa (682)
3. MaF (631)
4. Kai (599)
5. Mohammed (512)
6. TheRama (463)
7. DarrenW (404)
8. tony (402)
9. Mike S (381)

AQ 6 FIGURE 5.2 User ranking in Pex4Fun.



User168596 tried to win **C# - «ChallengeFactorial»**
about an hour ago

User168596 tried to win **C# - «ChallengeFactorial»**
about an hour ago

User168595 tried to win **C# - «ChallengeFibonacci2»**
2 hours ago

TaoXie signed in
3 hours ago

User168594 tried to win **C# - «ChallengeFibonacci»**
5 hours ago

User168594 tried to win **C# - «ChallengeFibonacci»**
5 hours ago

User168594 tried to win **C# - «ChallengeFibonacci»**
5 hours ago

User168594 tried to win **C# - «ChallengeFibonacci»**
5 hours ago

AQ 7 FIGURE 5.3 User ranking in Pex4Fun.

a coding duel that the player has created or the player is also trying to win. Figure 5.3 shows an example screen snapshot of the live feed.

Social dynamics in Pex4Fun share similar motivations as other recent gamification examples in software engineering. For example, Stack Overflow badges (http://stackoverflow.com/badges) 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 his or her 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 other 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 (http://channel9.msdn.com/achievements/visualstudio), receive badges, and increase their ranking on a leaderboard based on the program code that they have written.

5.3.4 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: anyone around the world can create coding duels for others to play besides playing the existing coding duels themselves. Teachers can create virtual classrooms in the form of courses by customizing the 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 that include 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 (2011) for engaging conference attendees to solve coding duels in a dynamic social contest. Pex4Fun has been gaining high popularity in the community: Because 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 1.5 million (1,540,970) as of July 21, 2014.

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!

5.3.5 Code Hunt

Code Hunt [28,29] is a recent gaming platform evolved from Pex4Fun. Code Hunt includes more gaming aspects to offer more engaging experiences to the player. Figure 5.4 shows the main page of the Code Hunt website. The gaming platform also has sounds and a leaderboard to keep the player engaged.

With coding duels as the game type, Code Hunt organizes games in a series of worlds, sectors, and levels, which become increasingly challenging. Figure 5.5 shows the example list of sectors available for the player to choose from. In each level, a coding duel game is played by the player, who has to discover a secret code fragment and write code (in Java or C#) for it. Figure 5.6 shows an example coding duel for the player to play, with the player's code shown on the left-hand side of the figure. After the player clicks the Capture Code button (shown near the middle-top part of the figure, with the same effect of clicking the Ask Pex! button in Pex4Fun),

FIGURE 5.4 (**See color insert**.) The main page of the Code Hunt website.

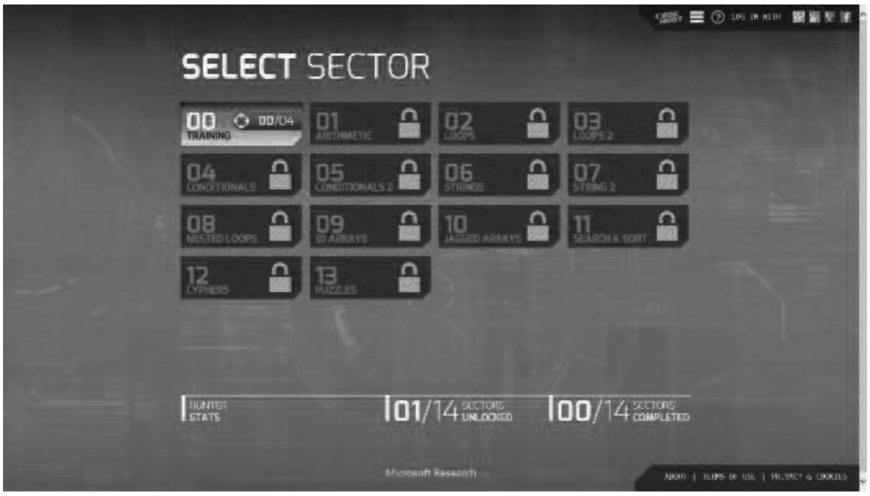
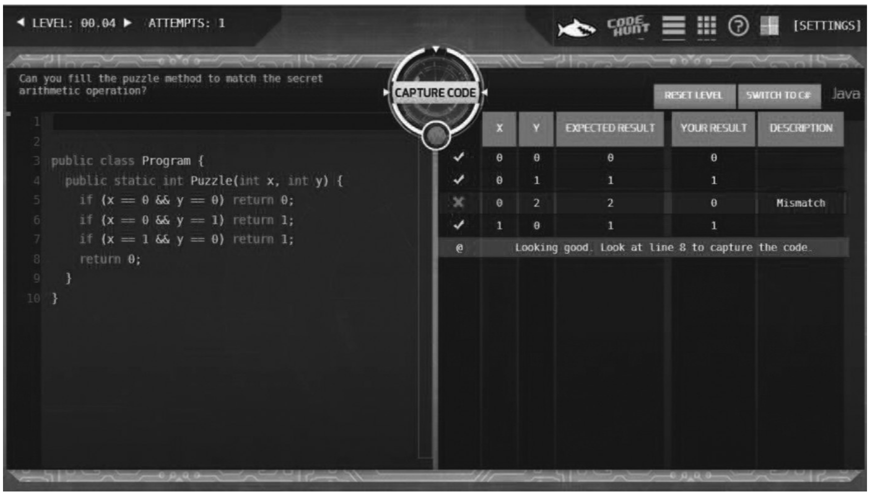


FIGURE 5.5 Example sectors in Code Hunt.

the underlying Pex tool gives customized progress feedback to the player via the generated test cases, as displayed in the table on the right-hand side of Figure 5.6. In addition, Code Hunt might hint a user to focus on a particular line of code, as shown in the last line of the table on the right-hand side of the figure. When the player's code achieves the same result as the secret implementation, Code Hunt flashes Captured Code and provides a score to the player based on how good the code was. Other improvements



Can you fill the puzzle method to match the secret arithmetic operation?

```


public class Program {
  public static int Puzzle(int x, int y) {
    if (x == 0 && y == 0) return 0;
    if (x == 0 && y == 1) return 1;
    if (x == 1 && y == 0) return 1;
    return 0;
  }
}
```

	X	Y	EXPECTED RESULT	YOUR RESULT	DESCRIPTION
✓	0	0	0	0	
✓	0	1	1	1	
✗	0	2	2	0	Mismatch
✓	1	0	1	1	
@	Looking good. Look at line 8 to capture the code.				

FIGURE 5.6 **(See color insert.)** An example coding duel in Code Hunt.

for Code Hunt beyond Pex4Fun are that Code Hunt offers Java as a supported language (via a source code translator) and it runs on Microsoft Azure, making it scalable to a large number of simultaneous users.

5.4 DISCUSSION

Educational software engineering is closely related to the field of educational games [30] (i.e., games for education), with example conferences such as the Games+Learning+Society Conference (http://www.gameslearningsociety.org/conference) and example initiatives such as the MacArthur Digital Media and Learning initiative (http://www.macfound.org/programs/learning). The field of educational games typically focuses on gaming technologies for supporting educational purposes, whereas educational software engineering typically focuses on software engineering technologies for supporting educational purposes. In the context of Pex4Fun and Code Hunt, the field of educational games would focus more on the aspect of gaming (Section 5.3) whereas the field of educational software engineering would focus more on the aspect of software engineering technologies (Section 5.3.1). 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 Code Hunt) 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:

1. 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 [5–7]. Another example case would be developing test generation technologies for Pex4Fun and Code Hunt, because secret implementations created for coding duels tend to be simpler than real-world code implementations.
2. 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 and Code Hunt aims at generating and reporting test inputs that can serve as feedback to achieve effective learning purposes.
3. Some educational tasks (such as intelligent tutoring [14] and problem generation [5–7]) 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).

5.5 CONCLUSION

In this chapter, 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 (along with Code Hunt), one of our recent examples on leveraging software engineering and gaming technologies for teaching and learning programming and software engineering skills. Pex4Fun and Code Hunt can also be used in the context of MOOCs to address issues such as automatic grading.

ACKNOWLEDGMENT

We thank the reviewers for their valuable feedback. Tao Xie's work is supported in part by a Microsoft Research Award, NSF grants CCF-1349666, CNS-1434582, CCF-1434596, CCF-1434590, and CNS-1439481, and the NSF of China, number 61228203.

AQ 8

AQ 9

AQ 10

REFERENCES

1. Mary Shaw. Software engineering education: A roadmap. In *Proceedings of FOSE*, pp. 371–380, 2000.
2. Mary Jean Harrold. Testing: A roadmap. In *Proceedings of FOSE*, pp. 61–72, 2000.
3. Dongmei Zhang, Yingnong Dang, Jian-Guang Lou, Shi Han, Haidong Zhang, and Tao Xie. Software analytics as a learning case in practice: Approaches and experiences. In *Proceedings of MALETS*, pp. 55–58, 2011.
4. Dongmei Zhang, Shi Han, Yingnong Dang, Jian-Guang Lou, Haidong Zhang, and Tao Xie. Software analytics in practice. Special Issue, *IEEE Software*, 5(30):30–37, 2013.
5. Erik Andersen, Sumit Gulwani, and Zoran Popovic. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of CHI*, pp. 773–782, 2013.
6. Sumit Gulwani, Vijay Korthikanti, and Ashish Tiwari. Synthesizing geometry constructions. In *Proceedings of PLDI*, pp. 50–61, 2011.
7. Rohit Singh, Sumit Gulwani, and Sriram Rajamani. Automatically generating algebra problems. In *Proceedings of AAAI*, 2012.
8. Tao Xie, Suresh Thummalapenta, David Lo, and Chao Liu. Data mining for software engineering. *IEEE Computer*, 42(8):35–42, 2009.
9. Ahmed E. Hassan. The road ahead for mining software repositories. In *Proceedings of FoSM*, pp. 48–57, 2008.
10. Armando Fox and David Patterson. Crossing the software education chasm. *Communications of the ACM*, 55(5):44–49, 2012.
11. AQ 11 Ken Masters. A brief guide to understanding MOOCs. *The Internet Journal of Medical Education*, 1, 2011.
12. Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. Automated feedback generation for introductory programming assignments. In *Proceedings of PLDI*, pp. 15–26, 2013.
13. Nikolai Tillmann, Jonathan De Halleux, Tao Xie, Sumit Gulwani, and Judith Bishop. Teaching and learning programming and software engineering via interactive gaming. In *Proceedings of ICSE, Software Engineering Education (SEE)*, pp. 1117–1126, 2013.

14. Tom 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.
15. Chao Liu, Chen Chen, Jiawei Han, and Philip S. Yu. GPLAG: Detection of software plagiarism by program dependence graph analysis. In *Proceedings of KDD*, pp. 872–881, 2006.
16. Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of SIGMOD*, pp. 76–85, 2003.
17. Andrei Papancea, Jaime Spacco, and David Hovemeyer. An open platform for managing short programming exercises. In *Proceedings of ICER*, pp. 47–52, 2013.
18. Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. CodeWrite: Supporting student-driven practice of Java. In *Proceedings of SIGCSE*, pp. 471–476, 2011.
19. Qianxiang Wang, Wenxin Li, and Tao Xie. Educational programming systems for learning at scale. In *Proceedings of Learning at Scale*, pp. 177–178, 2014.
20. Judith Bishop, Jonathan de Halleux, Nikolai Tillmann, Nigel Horspool, Don Syme, and Tao Xie. Browser-based software for technology transfer. In *Proceedings of SAICSIT, Industry Oriented Paper*, pp. 338–340, 2011.
21. Patrice Godefroid, Nils Klarlund, and Koushik Sen. DART: Directed automated random testing. In *Proceedings of PLDI*, pp. 213–223, 2005.
22. Nikolai Tillmann and Jonathan de Halleux. Pex—White box test generation for .NET. In *Proceedings of TAP*, pp. 134–153, 2008.
23. Tao Xie, Nikolai Tillmann, Peli de Halleux, and Wolfram Schulte. Fitness-guided path exploration in dynamic symbolic execution. In *Proceedings of DSN*, pp. 359–368, 2009.
24. Kunal Taneja and Tao Xie. DiffGen: Automated regression unit-test generation. In *Proceedings of ASE*, pp. 407–410, 2008.
25. Tao Xie, Jonathan de Halleux, Nikolai Tillmann, and Wolfram Schulte. Teaching and training developer-testing techniques and tool support. In *Proceedings of SPLASH, Educators' and Trainers' Symposium*, pp. 175–182, 2010.
26. Lori A. Clarke. A system to generate test data and symbolically execute programs. *IEEE Transactions on Software Engineering*, 2(3):215–222, 1976.
27. James C. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.
28. Nikolai Tillmann, Judith Bishop, Nigel Horspool, Daniel Perelman, and Tao Xie. Code Hunt—Searching for secret code for fun. In *Proceedings of SBST*, 2014.
29. Nikolai Tillmann, Jonathan de Halleux, Tao Xie, and Judith Bishop. Code Hunt: Gamifying teaching and learning of computer science at scale. In *Proceedings of Learning at Scale*, pp. 221–222, 2014.
30. James Paul Gee. *What Video Games Have to Teach Us about Learning and Literacy?* Palgrave Macmillan, New York, 2007.
31. Tao Xie, Nikolai Tillmann, and Jonathan de Halleux. Educational software engineering: Where software engineering, education, and gaming meet. In *Proceedings of GAS*, pp. 36–39, 2013.

Author Query Sheet

Chapter No: 5

Query No.	Queries	Response
AQ 1	The URLs in the "Notes" section have been moved to the text.	
AQ 2	Should "bottom Save and Submit Code" be "Save and Submit Code at the lower half of the interface"? Readers would find helpful screen shots of all interfaces and tools discussed; please consider providing.	
AQ 3	"under solving by a player" not clear.	
AQ 4	"in order to determine (1) whether code submitted by a student is correct to check the code's correctnessthe game progress of the player and to compute customized feedback" has been changed to "in order to (1) determine whether the code submitted by a student is correct, (2) check the code's correctness and game progress of the player, and (3) compute customized feedback"; please confirm.	
AQ 5	I deleted color references to Figure 5.1 as the figure is to be typeset grayscale. Please confirm.	
AQ 6	The text is cut at the bottom of Figure 5.2. Please check.	
AQ 7	Please confirm if the caption of Figure 5.3 can be changed as "Example screenshot of the Live Feed".	
AQ 8	The section head **Bibliography** is changed to **References**, since all references are cited in text. Please confirm.	
AQ 9	References are renumbered sequentially based on order of occurrence in citations. Please confirm.	

AQ 10	Please provide location of proceedings for references 1, 2, 3, 5, 6, 7, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 28, 29, 31.	
AQ 11	Please provide page range for reference 11.	