Crowdsourcing Code and Process via Code Hunt

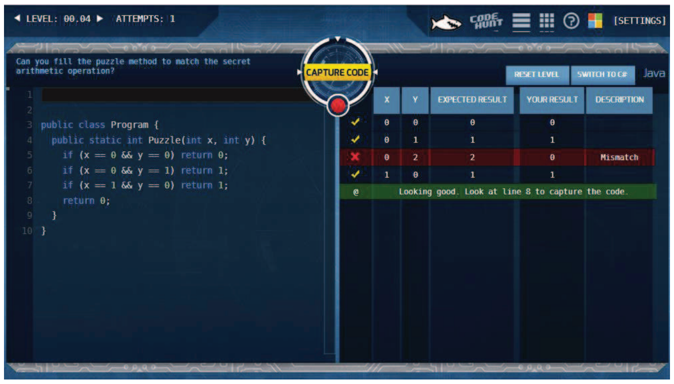
Tao Xie*, Judith Bishop†, R. Nigel Horspool‡, Nikolai Tillmann†, and Jonathan de Halleux†
*University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
†Microsoft Research, Redmond, WA 98052, USA
‡University of Victoria, Victoria, BC V8W 2Y2, Canada
Email: taoxie@illinois.edu, {nikolait,jhalleux,jbishop}@microsoft.com, nigelh@cs.uvic.ca

***Abstract*—Crowdsourcing programming relies on active participation. One way to get such participation is through an engaging game. Code Hunt (https://www.codehunt.com/) from Microsoft Research is a web-based serious gaming platform with the potential to be leveraged as a crowdsourcing system. In Code Hunt, players create programs by re-engineering against a changing set of test cases. The game has been played by over 100,000 players in world-wide contests, and to practice coding skills. The vast collected data of code modified by players and the process taken to succeed could be used by others for software construction, teaching, or learning. In this position paper, we discuss these existing crowdsourcing activities in Code Hunt and a future game type for crowdsourcing.**

I. Introduction

In recent years, crowdsourcing systems [3] have increasingly gained popularity for solving various problems. In the software engineering domain, crowdsourcing has also been used to solve programming, verification, and design problems. However, most crowdsourcing systems for software engineering are research prototypes and are typically deployed and evaluated on a relatively small scale of crowd with typically not more than 20 individuals [4], [5], [2]. Given the nature of the crowdsourcing domain and often diverse behavioral or capability characteristics of individuals, there is a strong need for a crowdsourcing system deployed in real usage with a large number of users and contributed solutions.

In this position paper, we propose that Code Hunt (https://www.codehunt.com/) [1] from Microsoft Research can serve as such an example of a crowdsourcing system for the research community. Code Hunt is a web-based serious gaming platform where players write code to advance through levels. Since its release in early 2014, Code Hunt has attracted over 100,000 users and accumulated their contributed solutions when playing coding duel games. In April 2014, Code Hunt was used at a very large competition called Beauty of Programming in the Greater China Region. In three rounds, 2,353 students scored in the game, with an average 55.7% puzzles solved across this large number. Code Hunt is being offered for more competitions, as ongoing efforts. The Code Hunt Challenge being run as part of Microsoft Imagine Cup (https://www.imaginecup.com/codehunt) has hundreds of participants per month from all around the world. Code Hunt's predecessor Pex4Fun [8] also gained 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 1.6 millions as of January 2015.



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

```
 1
 2
 3 public class Program {
 4   public static int Puzzle(int x, int y) {
 5     if (x == 0 && y == 0) return 0;
 6     if (x == 0 && y == 1) return 1;
 7     if (x == 1 && y == 0) return 1;
 8     return 0;
 9   }
10 }
```

	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.				

Fig. 1. User interface of game playing in Code Hunt

II. Code Hunt

The main game type in Code Hunt is a coding duel [8], represented as two code segments - a secret solution and one with a matching signature, but empty in content. To solve a coding duel, a player iteratively modifies the empty segment to match the functional behaviors of the secret code segment. Code Hunt relies on test cases generated by a state-of-the-art white box test generation tool (Pex [6], [7]) to characterize the functional behaviors of the secret code segment. These test cases are presented to the player as clues. The game aspect in Code Hunt is to recognize a pattern from the test cases, and to re-engineer the code to meet the expected behaviors.

The back end of Code Hunt runs in the cloud on Windows Azure, and games in Code Hunt can be played by players via any modern Internet browser. It supports writing code in both C# and Java when playing games. To play games, a player walks through a series of sectors, each of which further contains a series of levels. In a level, the player modifies the given code to solve a coding duel. Figure 1 shows the user interface of game playing in Code Hunt. When solving a coding duel, the player iteratively modifies the given code (displayed on the left hand side of the user interface) to match the functional behaviors of a secret code segment (not shown in the user interface). After the player clicks the button "Capture Code" (displayed on the top-middle part of the user interface), Code Hunt relies on the test cases generated by a state-of-the-art test generation tool (Pex) to characterize the functional behaviors of the secret code segment along with the code modified by the player. To guide the player to modify the given code, sample generated test cases are displayed (on the right hand side of the user interface) to the player to report same or different sample functional behaviors between the secret code segment and the code modified by the player. The game aspect in Code Hunt is essentially re-engineering from sample expected behaviors observed from the generated test cases.

III. Crowdsourcing Code and Process

Code Hunt can be leveraged as a crowdsourcing system for specific problems in software construction. In particular, by designing a coding duel, the duel designer can leverage the crowd to provide alternative implementations sharing the same functional behaviors as the given code segment (i.e., the secret code segment in the coding duel). After the crowd successfully contributes solutions for the coding duel, the duel designer can leverage tool automation to analyze these solutions and select the most desirable one(s) to use according to some metrics. For example, developers can use tool automation to select alternative implementations (of the same functional behaviors) that are more efficient (judged with an execution-cost metric) or of higher design quality (judged with a design quality metric) than the developers' existing code segment. Note that when designing a coding duel, the duel designer has the freedom of adding comments (which can include full or partial specification description of the functionalities implemented by the secret code segment) in the code given to the players; adding such comments may facilitate code crowdsourcing.

In addition, by designing a coding duel, the duel designer can leverage the crowd to contribute problem/duel-solving processes. Such problem-solving processes consist of every single attempt (code version) made by a player, being recorded in the Code Hunt platform. The problem-solving processes can be used in various ways, e.g., in educational software engineering [4], [9]. The teachers who assign the coding duel to their students in class can have a good understanding of common struggles or mistakes to be made by students ahead of time. The students in class solving the coding duel can also benefit from such problem-solving processes indirectly by leveraging hints produced by a hint generation engine that mines these problem-solving processes.

We next illustrate four key challenges that Code Hunt as a crowdsourcing system addresses [3]:

- *How to recruit contributors.* The crowdsourcing feature is piggybacked on Code Hunt, which itself is not explicitly a crowdsourcing system. So there is no need to recruit contributors but just rely on the contributors to Code Hunt. Code Hunt has been used in 15 programming contests. In such contests, contributors make contributions in order to compete. Code Hunt has also been used by players who would like to challenge themselves and improve their learning while having some fun. This usage is in the default zone of Code Hunt and has had over 90,000 players who start the graded duels.
- *What they can do.* The players of Code Hunt play a coding duel to modify the given code to match the functional behaviors of the secret code segment.
- *How to combine their contributions.* The collaborations among contributors are implicit. When code is crowdsourced, the contributions by various players for a coding duel are evaluated to select the most desirable one(s) with respect to the target metric. When processes are crowdsourced, problem-solving processes are mined all together. Some approaches such as program boosting [2] can also be used to blend these crowd-contributed duel solutions to produce even more-desirable solutions.
- *How to manage abuse.* Code Hunt leverages Pex [6], [7] as the checking engine to assure that the crowd-contributed duel solutions indeed have the same functional behaviors as the secret code segment. So there is little chance of being abused, such as contributing a solution that has different functional behaviors (e.g., seeded with code-level security vulnerabilities) than the secret code segment.

IV. Future Plan

In the future, we plan to explore a new game type to allow explicit collaboration and competition among players. In particular, instead of keeping the secret code segment secret, we can make the (secret) code segment visible to the players. Then the goal of the players is to modify the given code segment to achieve more desirable metric values than other players' solutions according to the specified metric (such as a design quality metric). The "best" player solution can be even displayed to the players. Pex is still used to make sure that crowd-contributed solutions have the same functional behaviors as the given code segment. In a current coding duel, if secret code segments are too complex, it is possible that few players can reverse-engineer the functional behaviors of the secret code segments. With this new game type, more-complex code segments can be used in the gaming.

Acknowledgments. Tao Xie's work is supported in part by a Microsoft Research Award, NSF grants CNS-1434582, CNS-1439481, CCF-1349666, CCF-1409423, CCF-1434590, and CCF-1434596.

References

[1] J. Bishop, N. Horspool, T. Xie, N. Tillmann, and J. de Halleux. Code Hunt: Experience with coding contests at scale. In *Proc. ICSE, JSEET*, 2015.

[2] R. A. Cochran, L. DAntoni, B. Livshits, D. Molnar, and M. Veanes. Program boosting: Program synthesis via crowd-sourcing. In *Proc. POPL*, pages 677–688, 2015.

[3] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, Apr. 2011.

[4] B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer. What would other programmers do: Suggesting solutions to error messages. In *Proc. CHI*, pages 1019–1028, 2010.

[5] T. D. LaToza, W. B. Towne, C. M. Adriano, and A. van der Hoek. Microtask programming: Building software with a crowd. In *Proc. UIST*, pages 43–54, 2014.

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

[7] N. Tillmann, J. de Halleux, and T. Xie. Transferring an automated test generation tool to practice: From Pex to Fakes and Code Digger. In *Proc. ASE, Experience Papers*, pages 385–396, 2014.

[8] N. Tillmann, J. de Halleux, T. Xie, S. Gulwani, and J. Bishop. Teaching and learning programming and software engineering via interactive gaming. In *Proc. ICSE SEE*, pages 1117–1126, 2013.

[9] T. Xie, N. Tillmann, J. de Halleux, and J. Bishop. Educational software engineering: Where software engineering, education, and gaming meet. In *Computer Games and Software Engineering*. Taylor & Francis Group, 2015.