# Pex4Fun: Teaching and Learning Computer Science via Social Gaming

Nikolai Tillmann, Jonathan de Halleux
Microsoft Research

Tao Xie
North Carolina State University
Email: {nikolait,jhalleux}@microsoft.com, xie@csc.ncsu.edu

## Abstract

*Pex4Fun (`http://www.pexforfun.com/`) from Microsoft Research is a web-based serious gaming environment for teaching computer science. Pex4Fun can be used to teach and learn computer programming at many levels, from high school all the way through graduate courses. With Pex4Fun, a student edits code in any browser – with Intellisense – and Pex4Fun executes it and analyzes it in the cloud. Pex4Fun connects teachers, curriculum authors, and students in a unique social experience, tracking and streaming progress updates in real time. In particular, Pex4Fun finds interesting and unexpected input values that help students understand what their code is actually doing. The real fun starts with coding duels where students write code to implement a teacher's specification. Pex4Fun finds any discrepancies in behavior between the student's code and the specification.*

*This tutorial equips participants with skills and knowledge of using Pex4Fun in teaching and learning, such as solving puzzles, solving coding duels, exploring course materials in feature courses, creating and teaching a course, creating and publishing coding duels, and learning advanced topics behind Pex4Fun.*

## 1 Tutorial Overview

Pex4Fun (`http://www.pexforfun.com/`) from Microsoft Research brings programming with fun to a user's web browser. In a game-like way, a user can write, compile, and run code in order to learn programming concepts, practice coding skills, and analyze the behavior of code interactively. This tutorial on Pex4Fun is given using an interactive style with slide handouts, slide presentation, demos, and discussions. Throughout the tutorial, tutorial participants will carry out hands-on and interactive exercises on the Pex4Fun web site, under the guidance of the tutors. Through the tutorial, the following skills and knowledge of using Pex4Fun in teaching and learning are expected to be gained by the tutorial participants.

**Solve puzzles.** The main Puzzle method used in Pex4Fun can take parameters and return values. In order to run such a puzzle method, someone must provide argument values. Click "Ask Pex!", and then Pex [5], the underlying test-generation engine, automatically finds interesting argument values by analyzing the code from the puzzle method.

**Solve coding duels.** A coding duel is an interactive puzzle. In a coding duel, your task is to implement the Puzzle method to have exactly the same behavior as another secret Puzzle method (e.g., a teacher's specification). To start with a simple coding duel, click an example coding duel from the web site. You see a dummy implementation that does not do much. Click "Ask Pex!" to see how it differs from the secret implementation. Compare your result to the secret implementation result. Analyze the differences and change the code to match the secret implementation result for all input values. Click "Ask Pex!" again. Repeat this process until you win the coding duel. When you have won the duel, you could try other coding duels. Pex for fun can track your progress if you sign in.

**Explore course materials in feature courses.** The current feature courses include C# for fun (for C# learners), Parameterized Unit Testing [7, 6] (for developer testing learners [9, 8]), Code Contracts [1] in .NET (for specification learners).

**Create and teach a course.** Pex4Fun can be used to build interesting, engaging, demanding classes on mathematics, algorithms, programming languages, or problem solving in general. Teachers can use an integrated wiki to author class materials built upon puzzles and coding duels. In particular, a teacher combines existing pages into a course. The pages might have been written by the teacher or by any other author. The teacher invites students to the course by sharing a registration link with them. A course can have multiple teachers. Any user can become a student by registering for a course through the registration link. The student can then work through the pages that are part of the course. To pass the course, the student completes exercises in the form of coding duels. A student can unregister from a course at any time.

**Create and publish coding duels.** Five steps are needed to create and publish coding duels. Step 1: sign in, so that Pex4Fun can maintain coding duels for you. Step 2: write a specification starting from a puzzle template where you can write the specification as a Puzzle method that takes inputs and produces an output. Step 3: create the coding duel by clicking a button "Turn This Puzzle Into A Coding Duel" (appearing after you click "Ask Pex!"). Step 4: edit visible program text next by clicking the coding duel Permalink URL, which opens the coding duel, and by filling in a slightly more useful outline of the implementation (with optional comments) that somebody else will eventually complete. Step 5: publish after you finish editing the visible Puzzle method text by clicking the "Publish" button.

**Learn advanced topics behind Pex4Fun.** Pex4Fun was started in the Research in Software Engineering (RiSE) group at Microsoft Research, where research on software modeling, verification, and testing has been conducted for over a decade. One very promising testing technique, which draws on advances in software verification and automated theorem proving, is dynamic symbolic execution [3, 5], where a program is executed multiple times with different concrete inputs, while the taken execution paths are monitored at the instruction level, with a symbolic representation constructed for the conditions checked by the branching statements. A constraint solver [2] computes new concrete test inputs that will exercise different execution paths. This technique is effective for finding software defects, whose diverting execution paths are triggered by either implicit runtime checks or explicit sanity checks in the code. Coding duels (as well as puzzles) in Pex4Fun leverage this technique [4]: given a function $f(x)$ from the student and a function $g(x)$ from the teacher (the secret program), this technique explores the following meta program h: $h(x) := Assert(f(x) == g(x))$. By thoroughly analyzing this program with this technique of dynamic symbolic execution, Pex4Fun generates a test suite that is customized to both programs. Every time the student submits a new program, Pex4Fun generates a new test suite, showing any behavior mismatches to the secret program, or, if there are no mismatches, indicating that the student won the coding duel.

## 2 Post-Tutorial Activities or Support

After the tutorial, tutorial participants can discuss Pex4Fun on the MSDN Forums for Pex (`http://social.msdn.microsoft.com/Forums/en-US/pex/`), where the participants can also post their Permalinks (permanent URL links) to share them with other people.

Teacher participants are encouraged and supported to create their courses under the Pex4Fun web site. In particular, by customizing teaching modules from the web site, a teacher can create entire courses, combining explanatory text, example code, and coding duels as exercises. Students can register to such a course, and then the teacher gets informed about the progress of the students, including a table that shows which exercises a student completed – in effect automatically grading the course assignments.

## References

[1] M. Barnett, M. Fähndrich, P. de Halleux, F. Logozzo, and N. Tillmann. Exploiting the synergy between automated-test-generation and programming-by-contract. In *Proc. ICSE Companion*, pages 401–402, 2009.

[2] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. TACAS*, pages 337–340, 2008.

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

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

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

[6] N. Tillmann, P. de Halleux, and T. Xie. Parameterized unit testing: Theory and practice. In *Proc. ICSE 2010 Companion, Tutorial*, pages 483–484, 2010.

[7] N. Tillmann and W. Schulte. Parameterized unit tests. In *Proc. ESEC/FSE*, pages 253–262, 2005.

[8] 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.

[9] T. Xie, N. Tillmann, J. de Halleux, and W. Schulte. Future of developer testing: Building quality in code. In *Proc. FoSER*, pages 415–420, 2010.