Increasing Student Engagement in Software Engineering with Gamification

Swapneel Sheth, Jonathan Bell and Gail Kaiser
Department of Computer Science,Columbia University, New York, NY, USA
{swapneel,jbell,kaiser}@cs.columbia.edu

Abstract

Gamification, or the use of game elements in non-game contexts, has become an increasingly popular approach to increasing end-user engagement in many contexts, including employee productivity, sales, recycling, and education. Our preliminary work has shown that gamification can be used to boost student engagement and learning in basic software testing. We seek to expand our gamified software engineering approach to motivate other software engineering best practices. We propose to build a game layer on top of traditional continuous integration technologies to increase student engagement in development, documentation, bug reporting, and test coverage. This poster describes to our approach and presents some early results showing feasability.

1 Introduction

Students in undergraduate computer science courses are often averse to testing their code, despite numerous attempts to bring testing to early parts of the curriculum and the many proven benefits to learning testing habits early. To attack this problem, we created a prototype gameful platform called *HALO*, or "Highly Addictive sociAlly Optimized software engineering" [1, 2]. *HALO* uses MMORPG (Massively Multiplayer Online Role Playing Game) motifs to create an engaging and collaborative development environment. *HALO* makes the software development process and, in particular, the testing process more fun and social by using themes and mechanics from popular computer games. We propose to apply similar game mechanics to a continuous integration environment to increase student engagement and learning throughout the entire software development lifecycle and present our initial ideas.

2 Background

Much work has been done to create games to teach software engineering concepts [3–5]. Using games for teaching computer science is one example of the broader field of "games with a purpose," or "serious games" — games that are designed for a primary purpose other than entertainment. Serious games have also been developed for other educational purposes, from learning languages to healthcare education. They have even been used to solve complex computational challenges such as protein folding, with Foldit, or encourage exercise, with WiiFit.

Serious games are intended to be experienced primarily as games. Although they may have a serious purpose, the experience of playing them is, generally speaking, playful. Gamified systems, on the other hand, take design elements from games and adapt them for serious purposes. Gamified systems are therefore appropriate in contexts where games are not, and helpful with problems that games cannot address.

Our *HALO* system is not a game, but rather utilizes game mechanics to make software testing more fun and engaging. For instance, we provide students with social rewards, including titles — prefixes or suffixes for players' names — and levels, both of which showcase players' successes in the game world. We also hide testing behind a short story and a series of quests, for which students receive experience points upon completion. Students who are successful in *HALO* are recognized as their points accumulate, causing them to "level up."

We conducted a pilot study in two undergraduate Columbia Computer Science courses: COMS W1007 "Introduction to Object Oriented Programming and Design", a CS2-level course, and COMS W3134 "Data Structures," offering *HALO* as an optional addition to the programming assignments. The sample tasks supplied via *HALO* were typical edge cases that students may not have been aware of while coding.

We found that students who chose to use *HALO* showed greater improvement in the course. One student mentioned that *HALO* gives her "peace of mind when [she's] submitting her assignment, because [she] has tested her program very throughly." Another student said: "This makes it easier to make sure that our programs are doing what they are supposed to." We are in the process of analyzing quantitative results, but the preliminary data is promising: between the first assignment (where no students used *HALO*) and the second assignment (where all students were encouraged to try *HALO*), students who used *HALO* showed a statistically significant improvement in performance ($p < .04$).

However, a key limitation to our approach thus far is that it is restricted to teaching software testing, and only at the introductory level. We seek to expand our studies to be applicable to general software development practices, and describe our approach below.

3 Engaging Students in Continuous Integration

HALO encourages students to test their code with a series of quests, created by the instructor and tailored to the programming assignment. This requirement adds an overhead to the instructor to deploy *HALO* with each assignment, and cannot work in instances where students design their own projects. We seek to remove instructor overhead, and expand *HALO* to support collaborative group projects by directly tracking student metrics. For instance, one could track the frequency of commits, and provide students with a sort of competition to commit more often [6].

Instead of just focusing on version control, we will build our approach into a Continuous Integration (CI) development environment, an approach to software development that highlights quality control. In CI, all code is maintained in a central repository, with an automated build and testing process. At regular intervals (e.g. after commits), the application is automatically built and tested. The CI process provides automated insight directly into many quality metrics that can be used as input to a game-like environment.

We will extract metrics such as code coverage, build successes, bug introduction, and test failures from the development environment and create a dashboard to highlight these metrics to students. Using this system, students will be recognized both for their overall successes in each area, and their individual improvements. With such an approach, we hope to engage and encourage both students who are performing well, and those in need of improvement. Students will be able to compete both with their teammates (within their group), and with other students in the course. The poster will also include detailed mockups of the proposed environment.

References

[1] S. Sheth, J. Bell, and G. Kaiser, "HALO (Highly Addictive, sociaLly Optimized) Software Engineering," in *Proceeding of the 1st international workshop on Games and software engineering*, GAS '11, (New York, NY, USA), pp. 29–32, ACM, 2011.

[2] J. Bell, S. Sheth, and G. Kaiser, "Secret ninja testing with halo software engineering," in *Proceedings of the 4th international workshop on Social software engineering*, SSE '11, (New York, NY, USA), pp. 43–47, ACM, 2011.

[3] S. Elbaum, S. Person, J. Dokulil, and M. Jorde, "Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable," in *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, (Washington, DC, USA), pp. 688–697, IEEE Computer Society, 2007.

[4] A. Nickel and T. Barnes, "Games for cs education: computer-supported collaborative learning and multiplayer games," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, (New York, NY, USA), pp. 274–276, ACM, 2010.

[5] M. Eagle and T. Barnes, "Experimental evaluation of an educational game for improved learning in introductory computing," *SIGCSE Bull.*, vol. 41, pp. 321–325, March 2009.

[6] L. Singer and K. Schneider, "It was a bit of a race: Gamification of version control," in *Games and Software Engineering (GAS), 2012 2nd International Workshop on*, pp. 5 –8, june 2012.