S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 181

Serious Games for Software Refactoring

Thorsten Haendler[1], Gustaf Neumann[2]

Abstract:

This summary refers to the paper *Serious Refactoring Games* published as a full research paper in the proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019) [HN19].

Software design issues can severely impede software development and maintenance. Thus, it is important for the success of software projects that developers are aware of bad smells in code artifacts and improve their skills to reduce these issues via refactoring. However, software refactoring is a complex activity and involves multiple tasks and aspects. Therefore, imparting competences for identifying bad smells and refactoring code efficiently is challenging for software engineering education and training. The approaches proposed for teaching software refactoring in recent years mostly concentrate on small and artificial tasks and fall short in terms of higher level competences, such as analysis and evaluation. In this paper, we investigate the possibilities and challenges of designing serious games for software refactoring on real-world code artifacts. In particular, we propose a game design, where students can compete either against a predefined benchmark (technical debt) or against each other. In addition, we describe a lightweight architecture as the technical foundation for the game design that integrates pre-existing analysis tools such as test frameworks and software-quality analyzers. Finally, we provide an exemplary game scenario to illustrate the application of serious games in a learning setting.

Keywords: Serious Games; Refactoring; Software Engineering Education and Training.

Motivation and Overview

Software systems become increasingly complex while being maintained and extended over years. Issues in software design and architecture (such as bad smells) can negatively affect a systems' maintainability and extensibility. Removing these issues via refactoring is often neglected in practice due to the perceived difficulties, risks or lack of adequate tool support. Refactoring efficiently demands software developers competent and also motivated, which poses challenges for software engineering education and training. Existing approaches for training and teaching software refactoring fall short in addressing higher-level competences (e.g., analysis and evaluation), by mostly focusing on artificial tasks and small code examples.

[1] Institute for Information Systems and New Media, Vienna University of Economics and Business (WU), Austria
thorsten.haendler@wu.ac.at
[2] Institute for Information Systems and New Media, Vienna University of Economics and Business (WU), Austria
gustaf.neumann@wu.ac.at

doi:10.18420/se2019-58

Serious games represent a promising way to impart higher-level competences by simulating (or providing) real-world conditions, while including motivational (and fun) aspects.

In [HN19], we propose a game design and architecture for serious gaming in software refactoring applicable to real-world artifacts and under authentic conditions. For this purpose, we integrate development tools such as quality analyzers (measuring a system's technical debt as a score representing the person-hours to fix the debt items) as well as regression tests (ensuring the correct system behavior). The game design includes single-player and multi-player games modes (see Fig. 1), where players can either compete against a pre-defined benchmark (*technical debt score*) or against each other.

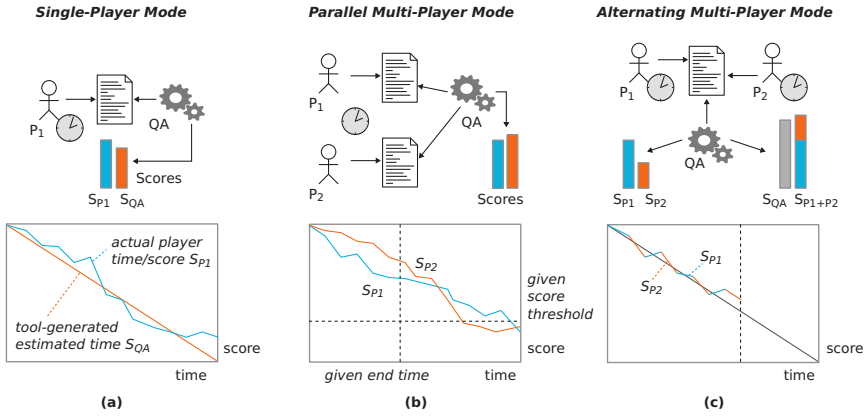

Fig. 1: Exemplary refactoring game modes with game constellations (top) and corresponding leader-boards with score-point progression (bottom) [HN19].

In addition, a lightweight game architecture is specified as a technical foundation for the game design. A game-play scenario illustrates the applicability. During game play, the players can demonstrate and consolidate the competences of identifying refactoring candidates and performing refactoring steps (*application*), analyzing code base and software design (*analysis*) as well as comparing refactoring options and strategies (*evaluation*).

The proposed approach is generic in the sense that it can be applied to any smell type (e.g., in source code, design/architecture, tests) provided that quality analyzers with corresponding smell metrics are available. We also believe that these refactoring games are not limited to training purposes, but could also be used for improving the quality of industry software. The code base can be analyzed and improved by a multitude of players, while each successful move is financially rewarded, similar to micro-task crowd-sourcing.

References

[HN19] Haendler, Thorsten; Neumann, Gustaf: Serious Refactoring Games. In: Proc. of the 52nd Hawaii International Conference on System Sciences (HICSS 2019), Software Engineering Education and Training Track, Hawaii, USA. 2019.