JUnitMe2.0 : JUnit tests generation with Alloy and CodeModel

Auteurs :
Salla DIAGNE
Anis TELLO

16th December 2015

Table of contents

Introduction

"Every program is guilty until proven innocent"

During the development phase, developers spend lots of time to write tests corresponding to code they have written in the program.

A program with high code coverage has been more thoroughly tested and has a lower chance of containing software bugs than a program with low code coverage.[1]

But what if developers didn't have to spend hours writing tests to have a good code coverage? What if there was a magic stick that can generate a bunch of unit tests?

Since Java is one of the most used programming languages in the world, we have decided to find a solution that would generate automatically Java unit tests. That would save time for developers, and would give the time and the energy to focus on working on business layer.

This project is an extended version of an application developed by Valentin Lefils and Quentin Marrecau [2] [3] . The first version of the application has treated the same problems we are facing, but only on small example of Java programs.

In this project, we present our tool: JUnitMe2.0. Our tool can generate automatically Java unit tests for any Java open source application. From a description of specification, our tool generate all instances that covere the data specifications, then generate the unit tests corresponding to these instances.

The rest of this report is organized as follows : Section 1.1 provides motivating examples and the goals of this work. Section 1.2 describes an overview of our tool: JUnitMe. Section 1.3 describes the algorithms. Section 1.4 explains the implementation and the architecture, section 1.5 provides an example of a use case and expected results. Section 2.1 evaluates our tool from a complexity point of view. Section 2.2 evaluates our tool from a performance point of view. Section 2.3 evaluates the ease of use of our tool. Finally, Section 3. concludes this report.

Chapter 1

Technical work

1.1 Goal

Project goal is to generate a big amount of tests for a Java program using Alloy analyser.

1.2 Overview

1.3 Algorithm

1.4 Implementation & architecture

using **Spoon** Java library to analyze and trasform source code, **Alloy** a language and tool for relational models, **Alloy Analyzer** a solver that takes the constraints of a model and finds structures that satisfy them and **CodeModel** a Java library for code generators.

1.5 Utilisation

Chapter 2

Evaluation

2.1 Complexity

2.2 Performance

2.3 Ease of use

Conclusion

Bibliography

[1] Wikipedia. Code coverage. https://en.wikipedia.org/wiki/Code_coverage.

[2] Valentin Lefils and Quentin Marrecau. Génération de tests junit avec alloy. http://static.monperrus.net/iagl/2014/rendu3-alloy-test-data-generation/01_Junit_Generation_Marrecau_Lefils/Rapport.pdf, 2015.

[3] Valentin Lefils and Quentin Marrecau. Junitme (github). https://github.com/user4me/JunitMe.