Universitatea Politehnica Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Studying the Evolution of Static Methods and their Effect on Class Testability

Cosmin Marsavina
University Politehnica of Timisoara, Romania
cosmin.marsavina@upt.ro

05.11.2020

Outline

Universitatea Politehnica Timişoara

- Introduction
- Methodology
- Analysis
- Discussion
- Conclusions
- Questions

Introduction

Testability

- Testing is one of the most important activities that occur during software development
- In order for this activity to go smoothly the production classes need to be highly testable
- We want to identify specific design flaws that have a negative impact on testability

Introduction

Static Methods

- Static methods are generally perceived as one of the main causes for reduced testability

- We argue that not all the static methods which are utilized in complex systems are detrimental to this software quality aspect (for example, the ones that are part of utility classes)

Introduction

Research Questions

RQ1. How are static methods used in complex software systems?

RQ2. Do all static methods affect testability in a negative manner?

Introduction

Contributions

- A process through which static methods can be identified and categorized that also examines their evolution
- An empirical study that includes 6 open-source projects in which we try to establish whether or not the usage of static methods has a negative impact on software testability

Methodology

Detecting and Categorizing Static Methods

- Static methods are identified by getting all the methods from a class and filtering out the ones without the static keyword

- We also categorize static methods based on:

  1. the types of the classes they are part of: singletons, utility classes, and the rest of the production classes that have at least one static method

  2. whether or not they modify state: utilize mutable state or only operate on their parameters

Methodology

Static Method Evolution

- After categorizing the static methods we want to study how each type has evolved over the lifespan of a system

- We use Git to collect the historical data needed for this analysis:

  1. we retrieve a project's source code from the corresponding repository
  2. we iterate over its commits and compute the differences between the current version and the previous one
  3. we also record data related to the different types of static methods that appear and the production classes that utilize them

Methodology

Quantifying Class Testability

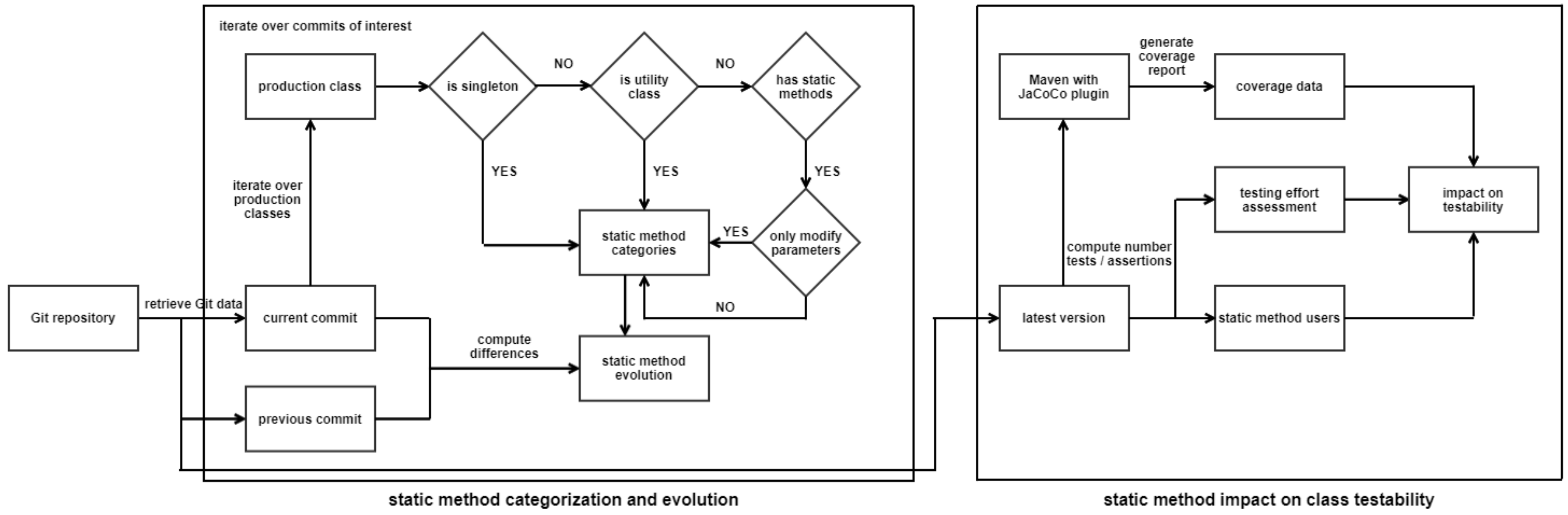
- Testability is assessed based on the testing effort that was put into a class

- This includes the line coverage obtained for the production class and the two measurements for its corresponding test class, namely:
  1. the number of lines of code for the respective class
  2. the number of assertions that are made in its test methods

- We want to establish which types of static methods make testing more difficult

Methodology

Implementation



Analysis

System Selection

- 6 open-source systems that are commonly used throughout the literature were selected based on 4 criteria; the projects needed to have:
  1. a significant number of production classes
  2. corresponding test classes that contain a large number of tests
  3. a Git repository with a considerable amount of commits
  4. a Maven project structure

- Even though all the projects meet the criteria they are fairly different from one another in terms of size and complexity, development practices, or testing effort

Analysis

Categories of Static Methods

- Different types of static methods are more common depending on a project’s characteristics

  - for example, the Commons libraries and jFreeChart have a higher percentage of utility classes compared to the other projects

  - while for 2 of the systems, BCEL and Lang, there are more classes with static methods that modify state, the other 4 have significantly more methods that only operate on their parameters

- With the exception of Geode, the Singleton pattern is rarely used in the other projects

Analysis

Evolution of Static Methods

- If instances from a category are present in the first version of a system then their percentage is generally higher than the one for the last version considered

- The maximum values for the percentage of classes that use methods of a specific type occur at the beginning of the development process

Analysis

Impact on Class Testability

- Not all static methods have a negative effect on the testability of the classes that utilize them
- Classes that use either utility classes or static methods which only operate on parameters are not tested less compared to the rest of the code
- In contrast, classes that utilize stateful singletons or static methods which modify state appear to be more difficult to test

Discussion

First Research Question

- We found that a large number of classes have static methods and the percentage of production classes that utilize them is quite high
- Different categories of static methods appear more frequently depending on the specific type of a system
- In terms of evolution, static methods are being used less in later years than at the beginning of the development process

Discussion

Second Research Question

- The results prove that not all types of static methods negatively impact class testability

- While the usage of static methods that modify state or are from stateful singletons causes a production class to be tested less, there are other categories of methods (e.g., from utility classes) for which this is not the case

Discussion

Threats to Validity

- To avoid **internal threats** we:
  1. tested the detection strategies against a number of small systems
  2. verified if the differences between commits were computed correctly
  3. manually checked the metrics calculated for quantifying testability

- We tried to mitigate **external threats** by selecting projects with different characteristics, based on a set of well-established criteria

Conclusions

Main Findings

- We showed that static methods are present in the production classes and widely used throughout the source code

- We have successfully proven that only some categories of static methods hinder the testing process

Conclusions

Future Work

- Improving the empirical study by analyzing more projects (both open-source and commercial ones)
- Refining the method through which we quantify class testability
- Identifying other design flaws that reduce the testability of a production class

Questions

Universitatea Politehnica Timişoara