An Empirical Study of Test Generalization in an Open Source Project

Xiaokang Xiang
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
xxiang4@illinois.edu

Brandon Carlson
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
blcrlsn2@illinois.edu

Abstract

Note that this sample paper skeleton serves as a guideline for helping produce a high-quality project report. If you think that you have a better or alternative way for presenting your project results (not fully following the guideline here), you shall contact the instructor.

In the abstract, you shall describe briefly the motivation of test generalization, the name and characteristics of your open source project under test, and the findings of your empirical study.

1. Introduction

You shall describe with more details on the background of parameterized unit tests (PUTs) [3, 4] and Pex [2], the motivation of test generalization (e.g., despite the benefits of PUTs, many existing unit tests are not written in PUTs), how you can generalize conventional unit tests (recall what the instructor taught in class), the name and characteristics of your open source project under test, and the findings of your empirical study.

You shall list the structure of the rest of the paper.

2. Example

You shall give an example conventional unit test (which is preferred to be from your open source project under test) and illustrate the procedure of test generalization with this example.

3. Open Source Project Under Test

You shall give details on what the project is about, and the characteristics of the project's production code base (such as the number of classes, the number of methods, the number of public methods, and the lines of code), and the project's test code base (such as the number of test classes, the number of test methods, and the lines of test code). Note that you may use some existing tools for producing these metrics; use your Google skills to find out these tools for you to use!

If you do not generalize all the conventional test cases for the open source project, you should list the characteristics of the portion that you focus on (e.g., showing the similar metrics listed above for the focused portion). You may give some justification on why you focus on your selected portion instead of other conventional test classes.

4. Benefits of Test Generalization

You shall describe the percentage of conventional unit tests that you can generalize to PUTs and the percentage of convectional unit tests that you cannot.

For those that are amenable to test generalization, you shall list and compare the code coverage achieved by the original conventional unit tests and your generalized PUTs (together with their generated tests by Pex). If you observe some new abnormal behaviors indicating potential faults (such as new uncaught exceptions or unexpected assertion violations), you can also describe them as benefits of the PUTs. When you list code coverage, you can list details (test class by test class or test method by test method) or just list the aggregated statistics for all the test classes or test methods. How much detailed you want get into depends on the page limit and how you want to devote the limited space for showing the best (most interesting) results or findings from your course project.

5. Categorization of Conventional Unit Tests

5.1. Conventional Unit Tests Amenable to Test Generalization

You shall categorize your PUTs generalized from conventional unit tests into the test patterns proposed by de

Halleux and Tillmann [1]. You shall list the statistics of your PUTs falling into each pattern.

You shall also propose new patterns to accommodate those PUTs that you cannot categorize into any of the patterns proposed by de Halleux and Tillmann [1]. You shall describe the definition of these new test patterns and the example PUTs for these patterns. If you run out of space, you can refer the readers to the wiki entry links for the details of your new test patterns. Note that you shall at the same time prepare your wiki entries for your new test patterns no matter whether you describe your new patterns here in details.

You shall list the statistics of your PUTs falling into each of your new pattern being proposed by you.

If you cannot find any conventional unit test to be amenable to test generalization, you can state so (but you are expected to find at least one conventional unit test to be amenable to test generalization).

6. Conventional Unit Tests Not Amenable to Test Generalization

You shall summarize the types of conventional unit tests that are not amenable to test generalization. It would be better if you can categorize them into anti-generalization patterns (if so, you shall also create wiki entries for your anti-generalization patterns).

7. Helper Techniques for Test Generalization

7.1. Factory Methods

You shall summarize the cases where you use factory methods to help Pex to generate better test inputs for your generalized PUTs. Again, it would be better if you can summarize patterns or categories for these cases.

If you find no such cases, you can state so.

7.2. Mock Objects

You shall summarize the cases where you use mock objects to deal with some complications that you face in test generalization. You shall categorize these cases into the mock object patterns proposed by de Halleux and Tillmann [1]. If you cannot categorize them into these patterns, propose new patterns and document them in wiki similar to the preceding guidelines for documenting your new normal PUT patterns.

If you find no such cases, you can state so.

7.3. New Assertions

You shall summarize the cases where you add new assertions to the generalized PUTs in order to improve the PUTs to capture more behaviors or properties.

If you find no such cases, you can state so.

8. Limitations of Pex or PUTs

You shall summarize the limitations of Pex in terms of supporting your test generalization or test generation for your PUTs. You shall also summarize the limitations of PUTs (e.g., some behavior cannot easily be expressed in PUTs and call for new types of tests or specification forms), which could be related to cases for not being amenable to test generalization described earlier.

If you find no such cases, you can state so.

9. Conclusion

Your conclusion can be structured similar to the structure of your abstract. But do not just copy your abstract here.

References

[1] P. de Halleux and N. Tillmann. Parameterized test patterns for effective testing with Pex. Technical report, Microsoft Research Technical Report, Redmond, WA, September 2008.

[2] N. Tillmann and J. de Halleux. Pex-white box test generation for .NET. In *Proc. International Conference on Tests and Proofs*, pages 134–153, 2008.

[3] N. Tillmann and W. Schulte. Parameterized unit tests. In *Proc. joint meetings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 253–262, 2005.

[4] N. Tillmann and W. Schulte. Unit tests reloaded: Parameterized unit testing with symbolic execution. *IEEE Softw.*, 23(4):38–47, 2006.