e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

# Improving the Readability of Automated Unit Test Case Generation

Dr. C. Shoba Bindu , Dr. K. K. Baseer , Ms. A. Sushma Deepthi[3]

[1]CSE, JNTUA, Ananthapuramu, Andhra Pradesh, INDIA
shobabindhu@gmail.com
[2]IT, SVEC, Tirupati, Andhra Pradesh, INDIA
drkkbaseer@gmail.com
[3]CSE, JNTUA, Ananthapuramu, Andhra Pradesh, INDIA
avvaru.sushmadeepthi@gmail.com

## ABSTRACT

*Automated unit testing tools are mainly used for to reduce the cost and time of testing activities. It will produce a large number of unittests. One of the automated unit testing tool is EVOSUITE that can be automatically generates test cases. Adeveloper would need to read and understand these tests cases throughout software development and evolution. However, generated tests are more difficulttoread and understand. Unreadable tests are difficult to maintain and time-consuming process to the developers. In this paper an approach, coined Evosuite Enhancer is used to overcome this problem by adding comments to the unit tests and replacing the method name, with the aim of improving the readability of generated tests.To evaluate the improvement of readability, a controlled experiment is performed with 50 students and takes a readability scores and post-test questionnaires from them. The results from that experiment users prefer this approach and easy to read and understand the testcases.*

**Categories and Subject Descriptors**. D.2.5 [Software Engineering]: Testing and Debugging – Testing Tools.

**Keywords.** Software Testing, Automated unit test case generation, Controlled experiment, Readability.

## 1.INTRODUCTION

Software testing is an important activity of software development life cycle(SDLC) and in particular software quality assurance. However, it is expensive and testing process will consume 50% of whole project effort [3, 6] and programmers spending a one fourth of their work time on developer testing. Several automated unit testing tools have been proposed andit will produce a large number of unit tests.

Evosuite[8] is a tool that automatically generates test suites for java programs that achieve high code coverage and provide assertions. automatically generated test cases are helpful to the developers by the following two main reasons those are (i)to reduce the cost of testing processand(ii) to achieve high code coverage. A program can be developed by various people and test it. The resulting tests are read and understand by various developers throughout software development process. However, generated test cases are hard to comprehend and difficult to maintain[5]. so, readability is an important factor to optimize in the perspective of the automated unit test case generation[7].

Unit tests which are automatically generated by a tool will take more time to understand the tests .if a test is understood by only until unless it can be read. If code is not readable, it will be more difficult to perform any tasks that require understand it. Unreadable tests are difficult to maintain and understand and more time-consuming.

For example, consider the unit test test0 in Figure 1, which was automatically generated for the target class Event. From a birds'-eye view, the code of the unit test is short and simple: it contains a constructor, calling getEvent method and assertions. However it is difficult to tell, without reading the contents of the target class, 1) what is the behavior class under test, 2) whether the generated assertions are correct.

```
public class Event{
 @Test
  public void test01()  throws Throwable
{
     Event event0 = new Event();
     String string0 = event0.getEvent();
   assertEquals("", string0);
 }
}
```

**Figure.1: Motivating Example**

To solve this problemin this paper an approach, coined EvoSuiteEnhancer. By applying this approach to the testcases the resulting testcases of this approach are helpful to developers for better understanding of the code under test (CUT). This leads to the first research question:

RQ1:Does New Approach Improve Readability Of Test Cases?

Evosuiteenhancer will generate new testcases that are read and understand by developers and

RQ2: Time Spent On Understanding Test Cases?

By applying this approach to the testcases whether time taken to understand this test cases are more or less to understand.

The contributions of this paper are summarized as follows:

- Evosuite enhancer a novel approach it generates test cases with addition of comments to the testcases and replaces a test method by a number with its corresponding method name.
- A controlled experiment involving 50 human participants to investigate whether the approach may generate more readable test cases.

## 2. THE EVOSUITE ENHANCER APPROACH

This section describes the Evosuite Enhancer approach

### 2.1 Approach Overview

Figure 2 depicts the proposed EvoSuite Enhancer approach;it is an improvement to the Evosuite, which is designed to generate automatically test cases with addition of two things. Those are 1) adding comments to the each test case, and 2) replacing the test method number to method name. The work flow of this approach is shown in the figure2; it consists of the following 3 steps. Those are i) Test Case Generation ii)Evosuite Enhancer, and iii)Generated Test cases with improved readability.In step1 namely Test case Generation- test cases are generated automatically by using Evosuite [8]. In step2, Evosuite Enhancerwill take the input as generated test cases from previous step and in this it will done two tasks. One task is adding comments to the test cases and one more task is renaming test method number to method name. And finally in the last step the tests are generated with adding of two tasks mentioned in the previous step and adding to the original test suite. An example of test cases generated by EvoSuite Enhancer for the test case showed in Figure 1, which tests the java class Event in Figure 3.

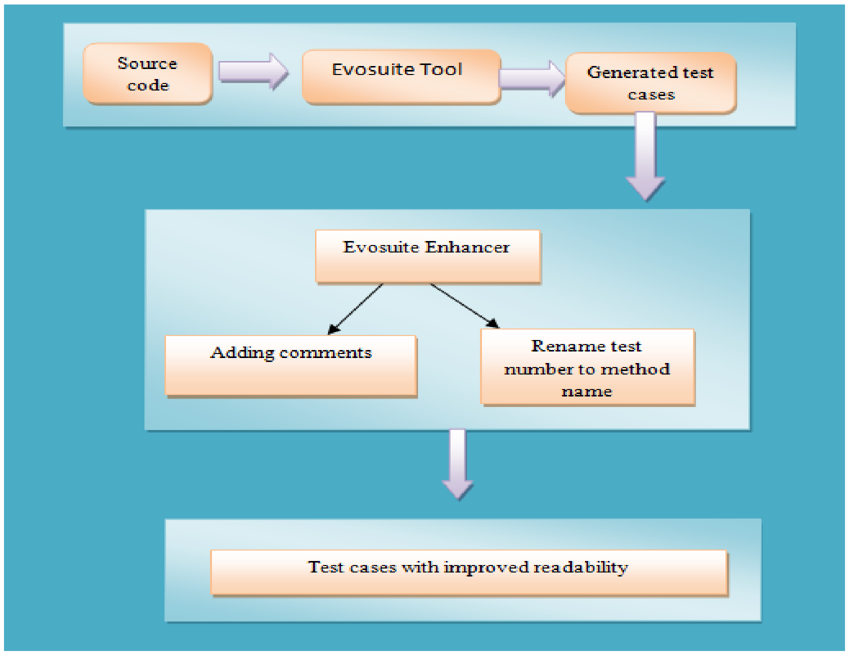


**Figure 2: Overview of Evosuite Enhancer**

### 2.2 Test Case Generation

In this step, 3 tasks are performed one by one as it will show in Figure 2. First task is a source code of java and then code is given to the Evosuite tool and finally tool generates the test cases.The source code is written in java having statements, classes, functions and many more. It will take it as input to the Evosuite tool and generates test cases.

Researchers have been proposed various techniques of automatically generating tests based on input assource code of a programunder test based on different searching techniques, such as symbolic execution [4], genetic algorithms [8], and so on. Among them, one is selected that is EVOSUITE [8], is a tool that automatically produces Junit test cases with assertions for classes written in java code. It uses a genetic algorithm to evolve candidate test suites according to the selected coverage criteria where the search is guided by the fitness function [8], which considers all the test targets like statements, branches, functions, etc., at the same time.

Evosuite generates a set of test cases for a given source code, the generated test cases consists of methods and constructors, assertions. Which are read an interpreted by the developers however these tests are

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

difficult to read.for this to improve readability of test cases an approach Evosuite Enhancer is proposed.

### 2.3 Evosuite Enhancer

In this step, the generated files are collected from the Evosuite for a given source code. Evosuite Enhancer will done two main tasks to improve the readability of the test cases. Those are i) adding comments before the declaration of the test class, and ii) rename the test method number to the method name.

Evosuite generated test cases for a piece of code it will be shown in Figure1. The functionality of the code isdon’t know until unless the code is read. To know the functionality of the code quickly and understand the test cases easily, comments are added to the test cases. For example, in Figure1, for an Event class the test case is generated, comments are added to that test case for which method the test case is for. It will be shown in Figure 3; the test case is for getEvent.

```
-----------------------------------------------------
-----------------------------------------------------
--------

//Empty String Test Case for getEvent

public class Event {
@Test (timeout = 4000)
public void getEvent_EmptyString () throwsThrowable {
    Event event0 = new Event ();
    String string0 = event0.getEvent ();
    assertEquals ("", string0);
}
}

-----------------------------------------------------
-----------------------------------------------------
---------
```

**Figure 3 : Example of comments and method name added generated by Evosuite Enhancer for a JUnit test method exercising the class Event.java**

The second task is renaming method test number to method name. The generated test cases for a given program there will be lot of methods and that methods name will be given like that namely test0(), test1(),test2() and soon shown in Figure1. For which method the test case is for don’t know until to read the total contents of the code. For this rename a test method number ( test0( ) )

to a valid useful method name like getEvent( ) it will shown in Figure 3. For a String it have to check empty string or non empty string and for an integer, it have +ve, -ve, and zero values. In Figure 3 the test method name is renamed to the**getEvent_EmptyString()** . By doing this it will be clearly known for which method the test case is for without reading the total contents of the code.

## 2.4 Test cases with improved Readability

Evosuite Enhancer performs two tasks and it will be the output of this step. The modifications are done in the previous step that will be collected and modified to the original test suite. The new test suite will be more readable and easy to understand. It will take less time to read so reduce the effort and cost of time to developers.

## 3.EXPERIMENT METHODOLOGY

The goal of this experiment is to investigate how easy to understand the generated tests before and after an approach is used. This section describes the experimental setup and procedure for the experiment in detail, following previous reporting guidelines for empirical software engineering research[12]. the purpose of this study was to investigate how easy to understand the generated tests by developers and compare the resulting tests before the approach and after the approach is applied. in this study participants ask to implement a java class and run evosuite and obtain results and other with an approach .

### 3.1 The Automated Testing tool:EVOSUITE

The automated unit testing tool used in this study is EVOSUITE[8], which automatically generates JUnit test suites for a given java class. It requires input as a java byte code of the class under test, along with its dependencies. Evosuite generates test suites with the aim of maximizing code coverage, minimizing the number of unit tests and optimizing their readability. The generated tests include assertions that capture the current behavior of the implementation. Evosuite can be used as a command-line tool for large scale experimentation,. However for this experiment The Evosuite intellij plug-in was used it allows developers to generate a Junit test Suite for any class by right-clicking the class name and selecting the Run Evosuite option.

### 3.2 ResearchQuestions

**Rq1: Does New Approach Improve Readability Of Test Cases?**First goal is to verify whether developers are able to easy to read and understand test cases, improved by adding comments before the declaration of the test class.

**Rq2: Time Spent On Understanding Test Cases?**The aim of this question is to verify whether the generated test cases with this approach, developers spending more or less time on understanding the test cases.

### 3.3 Object Selection

The task given to the participants of the experiment was to implement a java class with its test suite. Java classes are extracted from java open-source projects and participants test the selected classes from open source projects.selection of classes is done based on the following criteria. classes should have the following requirements.

- itcontains the source code as 40-70 non commenting source statements(NCSS).
- classes are testable by EVOSUITE with at least 80% code coverage.
- The participants have good programming skills howeverit will be simple, coded and tested in less than one hour.
- classes should be no inner classes and few dependencies.

### 3.4 Participant Selection

Selection of the participants for this experiment by sending email invitations to computer science graduate students. students of all these levels have at least basic understanding and experience in programming java, testing with JUnit and using Intellij. A total of 50 students were recruited to take part in the experiment. according to background survey, all participants had previously used the intellij idea and the Junit testing Framework, and had at least two years of programming experience.

### 3.5 Experiment Procedure

A controlled experiment is started with a 30 minute tutorial session, which included a live demo on a small example of how the experiment would proceed. participants then practiced on a simplified version of the main task: they implemented a class with a single method that is supposed to return the integer addition of its arguments. Interacted with the participants to make sure they all had a good understanding of java, JUnit, Evosuite and their task.

Each participant received two tasks:

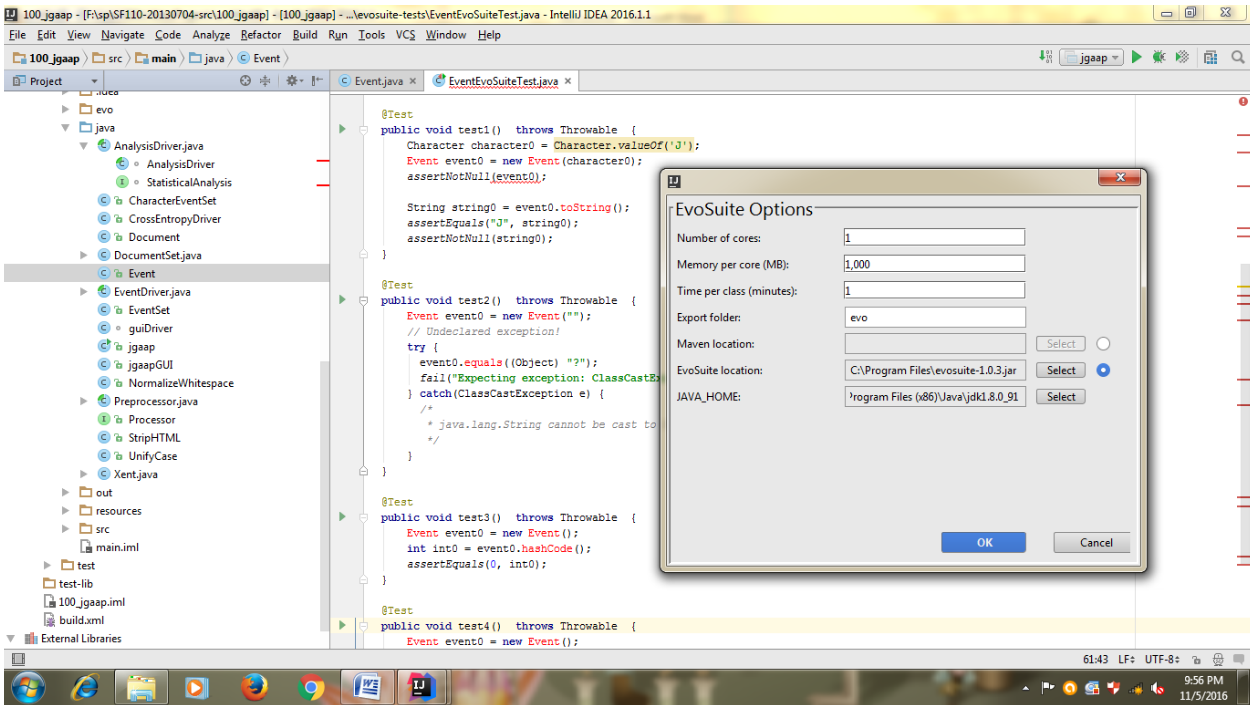
i) one task consisted of a java class to test together with the corresponding generated test cases without comments.

ii)the second task included one java class to test and corresponding generated test cases improved with the comments generated by an Evosuite Enhancer.

And fill the post-test Questionnaires.

the experimental infrastructure consists of

- sun JDK 1.8
- Intellij idea
- Junit
- the evosuite intellij plug-in(Figure 4)
- an intellij workspace consisting of only the target project with the class under test opened in the editor.



```java
@Test
public void test1()  throws Throwable  {
    Character character0 = Character.valueOf('J');
    Event event0 = new Event(character0);
    assertNotNull(event0);

    String string0 = event0.toString();
    assertEquals("J", string0);
    assertNotNull(string0);
}

@Test
public void test2()  throws Throwable  {
    Event event0 = new Event("");
    // Undeclared exception!
    try {
      event0.equals((Object) "?");
      fail("Expecting exception: ClassCastEx
    } catch(ClassCastException e) {
      /*
       * java.lang.String cannot be cast to
       */
    }
}

@Test
public void test3()  throws Throwable  {
    Event event0 = new Event();
    int int0 = event0.hashCode();
    assertEquals(0, int0);
}

@Test
public void test4()  throws Throwable  {
    Event event0 = new Event();
```

EvoSuite Options

| | | |
|---|---|---|
| Number of cores: | 1 | |
| Memory per core (MB): | 1,000 | |
| Time per class (minutes): | 1 | |
| Export folder: | evo | |
| Maven location: | | Select ○ |
| EvoSuite location: | C:\Program Files\evosuite-1.0.3.jar | Select [✓ ●] |
| JAVA_HOME: | ›rogram Files (x86)\Java\jdk1.8.0_91 | Select |

**Figure 4: EvoSuite can be used as a command line tool or as an intellij plug-in, producing coverage test suites for java classes fully automatically.**

## 3.5 Readability Scoring

Readability is considered as a key factor in the context of automated test case generation.Prior to their participation, instructed to the participants to rate the score for readability of test case. For that participants should select near 5 for more readable and a number near 1 for less readable, with a score of 3 indicating neutrality.

### 3.6 Experiment Results

#### Rq1:Effects on resulting tests cases

In order to determine the readability of test cases are improved for that they compare both the testcases one with evosuite and another with EvoSuite Enhancer. They observe the major things from the test cases. In the experiment, for a Event class Evosuite generated more no. of test cases and test methods like test0(),test1() and soon. Initially participants were not able to understand quickly for which method the test case is generated until only they will read entire code. And next, By adding comments to the test cases before declaration of the test class for example in Event class the comments are added like this for a test method as // **EmptyString Test Case for getEvent** and renaming test method number to useful method name like **getEvent_EmptyString()**so thatthey are easily read and understand quickly compared to that Evosuite. And participants were given the score for readability of the test cases[9]. The score range is from 0 to 5,0for low readable and 5 for more readable 3 for neutral. The results will be shown in Figure 5. The average score for before an approach is 2.88 and after approach is 3.6.

#### RQ2: Effects on time spent on understanding

Participants said that takes less time to understand the test casescompare to Evosuite because no need to read entire code by seeingthe test case they identify easily by adding comments to the test cases. Time taken to understand the test cases is depends on the capability of the students. In this experiment participants were given 40% rating totake more time to understand the test cases.The results of this is shown in table 1.

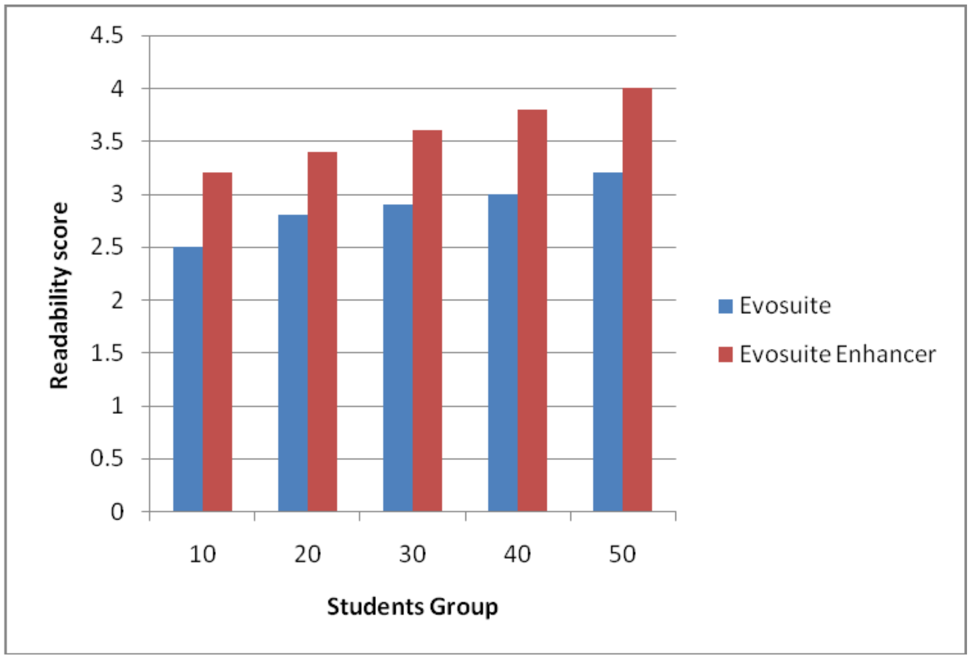


**FIGURE 5: Student Group vs. Readability Score**

**3.7 Post-test Questionnaires.** Table 1 shows the results to the questions from the exit survey. Participants were answered those questions the results will be given below.

**Table 1 Post-Test Questionnaires**

| **Sno** | **Questions** | **Percentage of ratings** |
|---|---|---|
| 1 | Adding comments to the tests leads to better results | 87 |
| 2 | Without comments, tests are difficult to read and understand | 80 |
| 3 | Had enough time to finish task with new approach | 40 |
| 4 | Is easy to read and understand | 78 |

| | | |
|---|---|---|
| 5 | Is somewhat readable and understandable | 80 |
| 6 | Is hard to read and understand | 10 |
| 7 | By adding test method name to useful valid name leads to better understanding | 85 |

The analysis is summarized in Table 1. The results highlight that i) participants were given 87% of ratings by adding comments to the test leads to better understanding ii)by renaming method names to valid method names gives more understanding the test cases.

## 4. RELATED WORK

Readability of code is of main concern for developers[13]. Code is readable and more maintainable; code that is more readable today easier to read, understands , and maintain at a later date. buse and weimer[2] introduced a metric for code readability based on human judgments. they collected human annotation data for code snippets and trained a classifier based on those scores.

Saff and Ernst[9] found statically significant evidence that student developers were more creative on programming tasks if they were provided with the extra feedback of regression tests continuously running in the background and notifications pop up as soon as new error was discovered.

in previous work[10] the effect of using EVOSUITE for testing only. in their work Results showed that automated unit test generation can support testers in producing better test suites but does not help in finding more faults .

G. Fraser [1] in their work a controlled experiment is done with students in that 63% most of the participants are said to add comments to generated tests because generated tests are difficult to read and understand. in this paper an approach Evosuite Enhancer is used that will add comments to the generated tests and renaming test number to a valid method name.

To improve the readability of the generated tests. Daka et al.[5] proposed a domain-specific model to incorporate human judgments to guide automated unit test generation. Afshan et al.[11] proposed a technique to incorporated a natural language model into search-based test input generation to generate more readable strings.

S. Panichella et al. in their paper an approach TestDescriber[7] is proposed. it will generate summaries to the test cases. Summarization will be in 4 levels class level, Test method, Branch covered, and Fine-grained statements. The results of this approach and experiment results in that participant mentioned feedback to improve the summaries, those are to reduce Redundant information from test to test and useless naming of test methods.

In this paper, shows that comments represent an important element for complementing and improving the readability of the test cases. Evosuite Enhancer will do the renaming of the method names to the useful names and add comments before the test class. So that it is possible to take a quick look of what is actually being tested by that test case and know the which methods of the class are tested.

## 5. CONCLUSIONS

Automated unit testing tools generate test cases these test cases read and interpreted by many developers. If tests are unreadable then it may be more difficult to read and understand and it will take more time. To overcome this problem in this paper a controlled experiment is conducted with students and one with evosuite and another with an approach evosuite enhancer.The experiment results shows that by adding comments to the tests before the declaration of the class and renaming test number to method name, developers to see at a glance what is actually being tested by that test case and which methods of the class are tested.So that developers are able to read and understand test cases in less time. Participants given readability score to test cases and finally the test cases are more readable. However, this approach is to increase readability of unittests is still quite limited in the scope of its changes to its test appearance.

## 6. REFERENCES

**[1]** G. Fraser, M. Staats, P. McMinn, A. Arcuri, and F. Padberg, "Does automated white-box test generation really help software testers?" in ACM Int. Symposium on software Testing and Analysis(ISSTA), 2013, pp. 291-301.

**[2]** R.P.L. Buse and W.R. Weimer. Learning a Metric for Code Readability. IEEE Transactions on Software Engineering, 36(4):546-558, 2010.

**[3]** F. P. J.Brooks. The MythicalMan-Month. Addison- Wesley, 1975.

**[4]** C.Cadar,V.Ganesh,P.M.Pawlowski ,D.L.Dill,and D.R.Engler. Exe: Automatically generating inputs of death. InProceedingsofthe Conferenceon Computer and CommunicationsSecurity(CCS),pages 322–335. ACM, 2006.

**[5]** E. Daka, J.Campos, G. Fraser, J. Dorn, andW.Weimer.Modelingreadabilitytoimprove unittests. In Proceedingsofthe 10th JointMeeting ofthe European SoftwareEngineering Conference and the ACM SIGSOFTSymposium onthe FoundationsofSoftware Engineering(ESEC/FSE). ACM, 2015. Toappear.

**[6]** S.Haiduc, J.Aponte,L.Moreno, and A.Marcus. On theuseofautomatedtextsummarizationtechniquesfor summarizingsource code. InProceedingsofthe Inter-nationalWorking Conferenceon Reverse Engineering (WCRE),pages35–44.IEEE,2010.

**[7]** S Panichella, A Panichella, M Beller, A Zaidman H C Gall, "The Impact of test cases summaries on bug fixing performance: an empirical investigation", in Proc. 38th International Conference on Software Engineering (ICSE), 2016 , pp. 547-558.

**[8]** G.Fraserand A.Arcuri. Whole testsuite generation.IEEE Trans.Software Eng., 39(2):276–291, 2013.

**[9]** D. Saffand M. D Ernst, "An experimental evaluation of continuous testing during development," in SCM Int. Symposium on Software Testing and Analysis(ISSTA), 2004, pp.76-85.

**[10]** J. M. Rojas, G. Fraser, and A. Arcuri, "Automated Unit Test Generation during Software Development: A Controlled Experiment and Think-Aloud Observations," in Proceedings of the 2015 International Symposium on Software Testing and Analysis, 2015, pp. 338-349.

**[11]** S. Afshan, P. McMinn, and M. Stevenson. Evolving readable stringtestinputsusing a naturallanguage model toreduce humanoracle cost. In

ProceedingsInternationalConferenceon Software Testing,Verifi- cationand Validation(ICST), pages 352–361. IEEE,2013.

**[12]** A.Orso and G.Rothermel, "Software Testing: A Research Travelogue(2000-2014),", in ACM Future of Software Engineering(FOSE), 2014, pp.117-132.

**[13]** Posnett, D., Hindle, A., Devanbu, P.: A Simpler Model of Software Readability. In:Working Conference on Mining Software Repositories (MSR). pp. 73-82 (2011).