Block 5

.

Enhancing the Usability and Integration of Automatic Test Generation Tools

Background

There exist several tools that automatically generate unit test cases for source code. These tools can generate test cases in a resource efficient manner and augment existing manually written test cases. However, automatically written test cases tend to have poor readability due to obscure identifier names and a lack of documentation, which can incur a significant maintenance effort once these tests are integrated into a codebase.

Researchers have developed several tools and techniques to address this issue. In this survey, we will evaluate these approaches.

This Survey

The goal of this survey is to evaluate several research tools developed to enhance the readability of automatically written tests. Please note that this survey does **not** aim to evaluate the presented automatically written test cases.

This survey starts with a small section regarding basic demographic information. After this, you will be presented with several versions of automatically generated tests across 4 sections. You will be asked to evaluate each version on certain criteria. A version of the automatically generated test case in its original form will also be presented to you for reference.

The survey contains **17 questions** and will take **10-15 minutes** to complete. **16** of these questions are multiple choice. There are also optional feedback forms intended to allow you to elaborate on your answers if appropriate.

Thank you for taking the time to participate in this survey!

Default Question Block

Q1.1. What is your primary profession?

- ○ Software Developer
- ○ Student (Undergraduate)
- ○ Student (Graduate)
- ○ Academia (Professor or Researcher)

Q1.2. How many years of experience do you have with Java (in years)?

	0	6	12	18	24	30	36	42	48	54	60
Industry											

	0	6	12	18	24	30	36	42	48	54	60
Academic											

Q1.3. Do you use automatic test case generation frameworks, such as Evosuite?

◯ Yes

◯ No

Q1.4. Do you use such automatic test case generation tools in projects you develop?

◯ Yes

◯ No

Q1.5. Please elaborate on which tools you use. If you don't use any test case generation tools, why not?

Section 1

.

Section 1: Summaries

In this section, you will be asked to evaluate the quality of the test case summaries generated by two different tools.

. First, we present to you the original automatically generated test case

.

```
@Test
public void test3() throws Throwable {
    Rational rational0 = new Rational(1L, 3215L);
    Rational rational1 = rational0.abs();
    assertEquals(1L, rational0.numerator);
    assertEquals(3215L, rational0.denominator);
    assertEquals(3.11041E-4F, rational1.floatValue(), 0.0
}
```

A). The following is the same unit test but along with the summary generated by the first tool (Listing **A**)

.

```
/**
 * 1. Creates a 2 Rational objects, "rational0" and "rati
 * 2. Checks the numerator and denominator of "rational0"
 *    float value of "rational1"
 **/
@Test
public void test3() throws Throwable {
    Rational rational0 = new Rational(1L, 3215L);
    Rational rational1 = rational0.abs();
    assertEquals(1L, rational0.numerator);
    assertEquals(3215L, rational0.denominator);
    assertEquals(3.11041E-4F, rational1.floatValue(), 0.0
}
```

B). The following is the same unit test but along with the summary generated by the second tool (Listing **B**)

.

```
/**
 * OVERVIEW: The test case "test3" covers around 6.0% (lo
 * statements in "Rational"
 **/
@Test
public void test3() throws Throwable {
    // The test case instantiates a "Rational" with numer
    // and denominator equal to 3215L.
    // The execution of this constructor implicitly cover
    // conditions:
    // - the condition " denominator equals to 0L" is FAL
    Rational rational0 = new Rational(1L, 3215L);
    // The test case declares an object of the class "Rat
    // is equal to the absolute value of "rational0"
    Rational rational1 = rational0.abs();
    // Then, it tests:
    // 1) whether the numerator of rational0 is equal to
    assertEquals(1L, rational0.numerator);
    // 2) whether the denominator of rational0 is equal t
    assertEquals(3215L, rational0.denominator);
    // 2) whether the float value of "rational1" is equal
    // with delta equal to 0.01F;
    assertEquals(3.11041E-4F, rational1.floatValue(), 0.0
}
```

Q2.1. How would you rate the **conciseness** of the two summaries?

	Contains no unnecessary information	Contains some unnecessary information	Contains mostly unnecessary information
Listing **A**	○	○	○
Listing **B**	○	○	○

. (Optional) Please elaborate on your answer above if appropriate.

Q2.2. How would you rate the **content** of the two summaries provided?

	Not missing any important information	Missing some important information	Missing some very important information
Listing **A**	○	○	○
Listing **B**	○	○	○

. (Optional) Please elaborate on your answer above if appropriate.

Q2.3. How would you rate the **readability** of the two summaries provided?

	Easy to read and understand	Somewhat easy to read and understand	Difficult to read and understand
Listing **A**	○	○	○
Listing **B**	○	○	○

. (Optional) Please elaborate on your answer above if appropriate.

Q2.4. Considering conciseness, content and readability, which summary do you prefer to use?

- ○ Summary in Listing **A**
- ○ Summary in Listing **B**
- ○ Neither

. (Optional) Feel free to leave any comments for the two proposed summaries here.

Section 2

.

Section 2

In this section, you will be asked to evaluate the quality of the test case name suggested by two different tools.

. We present the original test case here for reference

.

```
@Test
public void test8()  throws Throwable  {
    CharPosition charPosition0 = new CharPosition(346,
    VWordPosition vWordPosition0 = new VWordPosition(34
    assertNotNull(vWordPosition0);

    boolean boolean0 = vWordPosition0.equals((WordPosit
    assertEquals(false, boolean0);
    assertEquals("vertical(346;346,346)", vWordPosition
}
```

3.1. How would you rate each of the **two suggested test case names** on the basis of its ability to convey the intent of the test case?

	Fully captures the intent of the test case	Mostly captures the intent of the test case	Somewhat captures the intent of the test case	Doe not captu the inter of th test case
shouldReturnEqualsOtherObjectsWithoutType	○	○	○	○
testEquals	○	○	○	○

. (Optional) Please elaborate on your answer above if appropriate.

3.2. How would you rate the two suggested test case names on the basis of their **naturalness**?

	Easy to read and to understand	Somewhat easy to read and understand	Difficult to read and understand

	Easy to read and to understand	Somewhat easy to read and understand	Difficult to read and understand
shouldReturnEqualsOtherObjectsWithoutType	○	○	○
testEquals	○	○	○

. (Optional) Please elaborate on your answer above if appropriate.

Q3.3. Which test case name do you prefer?

- ○ **shouldReturnEqualsOtherObjectsWithoutType**
- ○ **testEquals**
- ○ Neither

. (Optional) Feel free to leave any comments for the two proposed test case names here.

Section 3

.

Section 3

In this section, you will be asked to evaluate the quality of the variable names suggested by a tool.

. We present the original test case here for reference

.

```
@Test(timeout = 4000)
```

```
public void test03()  throws Throwable  {
    MultivaluedHashMap<List<String>, List<String>> multiv
    LinkedList<String> linkedList0 = new LinkedList<Strin
    multivaluedHashMap0.putSingle(linkedList0, linkedList
    List<String> list0 = multivaluedHashMap0.getFirst(lin
    assertEquals(0, list0.size());
}
```

. Here is a version of the same test but with variables renamed based on the suggestions of the tool.

.

```
@Test(timeout = 4000)
public void testGetFirst()  throws Throwable  {
    MultivaluedHashMap<List<String>, List<String>> map =
    LinkedList<String> value = new LinkedList<String>();
    map.putSingle(value, value);
    List<String> result = value.getFirst(keys);
    assertEquals(0, result.size());
}
```

Q4.1. How would you rate the **variable names** used for the code snippet above?

	Fully conveys the intended usage of the variable	Somewhat conveys the intended usage of the variable	Does not convey the intedend usage of the variable	Does not convey the intended usage of the variable, and is misleading
map	○	○	○	○
value	○	○	○	○
result	○	○	○	○

. (Optional) If you think that the above variable names need improvements, please suggest more appropriate names.

Section 4

. **Section 4**

In this last section of the survey, you will be presented with a version of the test case that includes the generated summaries, method names and variable names.

A). We present the original version of the test case here for reference:

.

```
@Test(timeout = 4000)
public void test040()  throws Throwable  {
    KeycloakUriBuilder keycloakUriBuilder0 = KeycloakUriB
    HashMap<String, Integer> hashMap0 = new HashMap<Strin
    URI uRI0 = keycloakUriBuilder0.buildFromEncodedMap(ha
    assertEquals("x", uRI0.getRawPath());
```

```java
}
```

B). Here is the version of the test case using all the transformations:

.

```java
/**
 * 1. Creates a new KeyCloakUriBuilder "uri" from path
 * 2. Creates a new  HashMap and uses it to create a new
 *    method "buildFromEncodedMap" of "uri"
 * 3. Checks if the raw path of "result" equals "x"
 */
@Test(timeout = 4000)
public void testEncodedPath()  throws Throwable  {
    KeycloakUriBuilder uri = KeycloakUriBuilder.fromPath(
    HashMap<String, Integer> map = new HashMap<String, In
    URI result = uri.buildFromEncodedMap(map);
    assertEquals("x", result.getRawPath());
}
```

Q5.1. How would you rate the overall **improvement in readability** in the code snippet 4-B (enhanced using the proposed tool) over code snippet 4-A (original automatically generated test)?

◯ Significant increase in readability
◯ Minor increase in readability
◯ No change in readability
◯ Minor decrease in readability

- ◯ Significant decrease in readability

. (Optional) Please elaborate on your answer above if appropriate.

Q5.2. If you were to use automatically generated unit tests, how likely are you to also use the proposed tool to transform the generated test cases?

- ◯ Extremely likely
- ◯ Somewhat likely
- ◯ Neither likely nor unlikely
- ◯ Somewhat unlikely
- ◯ Extremely unlikely

. (Optional) Please elaborate on your answer above if appropriate.

Q5.3. With the existence of the proposed tool, how likely are you to utilize automatically written tests in projects you work on?

- ◯ Extremely likely
- ◯ Somewhat likely
- ◯ Neither likely nor unlikely
- ◯ Somewhat unlikely
- ◯ Extremely unlikely

. (Optional) Please elaborate on your answer above if appropriate.

Q5.4. Please rank the features of the tool that you found useful (if any) in the context of automatically generated test cases, in order of relative importance.

Items	Useful
Test Case Summaries	
Variable Renaming	
Method Renaming	

Not Useful

. (Optional) Please elaborate on your answer above if appropriate.

Block 4

. (Optional) Feel free to leave any final comments here.

. We appreciate you taking the time to participate in our survey. Please go to the next page to finalize and submit the survey.