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.

In order to address this issue, we have developed a tool to improve the readability of automatically written test cases by automatically adding documentation and renaming identifiers. Our rationale is that this improvement in readability will reduce the maintenance effort required associated with the use of automatically generated test cases, and significantly increase their usability.

This Survey

The goal of this survey is to evaluate the proposed tool's usefulness for software developers. Please note that this survey does **not** aim to evaluate the presented automatically written test cases. Instead, the focus is on evaluating the quality of the identifier renaming and the documentation generated by the proposed tool.

This survey starts with a small section regarding basic demographic information. After this, you will be presented with 4 versions of the same automatically generated test:

- a version containing a test case summary generated by the tool
- a version with the test case method name generated by the tool
- a version with variable names generated by the tool
- a version including all of the three modifications above (test summary, renamed methods and renamed variables)

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 **16 questions** and will take **10-15 minutes** to complete. **15** 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											
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

In this section, you will be asked to evaluate the quality of the test case summary generated by the proposed tool.

A). First, we present to you the original automatically generated test case (Listing **A**). The intended class under test is **AsyncFeign** (GitHub Link).

.

```java
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> asyncFeign_Asy
      AsyncFeign<IOException> asyncFeign0 = asyncFeign_As
      Class<Object> class0 = Object.class;
      Target.EmptyTarget<Object> target_EmptyTarget0 = Ta
      // Undeclared exception!
      try {
        asyncFeign0.newInstance((Target<Object>) target_E
        fail("Expecting exception: IllegalArgumentExcepti

      } catch(IllegalArgumentException e) {
         //
         // java.lang.Object is not an interface
         //
         verifyException("java.lang.reflect.Proxy$ProxyCl
      }
  }
```

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

.

```
/**
 * 1. Creates a new AsyncFeign using "asyncFeign0" using
 * 2. Creates a new EmptyTarget "target_EmptyTarget0" wit
 * 3. Expects an IllegalArgumentException when calling me
 *    "asyncFeign0" with "target_EmptyTarget0"
 */
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> asyncFeign_Asy
      AsyncFeign<IOException> asyncFeign0 = asyncFeign_As
      Class<Object> class0 = Object.class;
      Target.EmptyTarget<Object> target_EmptyTarget0 = Ta
      // Undeclared exception!
      try {
        asyncFeign0.newInstance((Target<Object>) target_E
        fail("Expecting exception: IllegalArgumentExcepti

      } catch(IllegalArgumentException e) {
         //
         // java.lang.Object is not an interface
         //
         verifyException("java.lang.reflect.Proxy$ProxyCl
      }
  }
```

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

- ○ Contains no unnecessary information
- ○ Contains some unnecessary information
- ○ Contains mostly unnecessary information

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

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

- ○ Not missing any important information
- ○ Missing some important information
- ○ Missing some very important information

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

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

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

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

Section 2

.

Section 2

In this section, you will be asked to evaluate the quality of the test case name suggested by our tool.

. We present the original test case here for reference

.

```
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> asyncFeign_Asy
      AsyncFeign<IOException> asyncFeign0 = asyncFeign_As
      Class<Object> class0 = Object.class;
      Target.EmptyTarget<Object> target_EmptyTarget0 = Ta
      // Undeclared exception!
      try {
        asyncFeign0.newInstance((Target<Object>) target_E
        fail("Expecting exception: IllegalArgumentExcepti

      } catch(IllegalArgumentException e) {
         //
         // java.lang.Object is not an interface
         //
         verifyException("java.lang.reflect.Proxy$ProxyCl
      }
```

```
    }
```

. For the unit test above, the proposed tool suggests the name **shouldThrowExceptionIfNoType**

3.1. How would you rate the suggested **test case name** 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
○ Does not capture the intent of the test case
○ Is misleading with regard to the intent of the test case

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

3.2. How would you rate the suggested **test case name** on the basis of its naturalness?

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

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

Section 3

.

Section 3

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

. We present the original test case here for reference

.

```
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> asyncFeign_Asy
      AsyncFeign<IOException> asyncFeign0 = asyncFeign_As
      Class<Object> class0 = Object.class;
      Target.EmptyTarget<Object> target_EmptyTarget0 = Ta
      // Undeclared exception!
      try {
        asyncFeign0.newInstance((Target<Object>) target_E
        fail("Expecting exception: IllegalArgumentExcepti

      } catch(IllegalArgumentException e) {
         //
         // java.lang.Object is not an interface
         //
         verifyException("java.lang.reflect.Proxy$ProxyCl
      }
  }
```

. Here is a version of the same test but with variables renamed according to the suggestions made by our tool:

.

```java
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> builder = new
      AsyncFeign<IOException> exception = builder.build()
      Class<Object> task3 = Object.class;
      Target.EmptyTarget<Object> target = Target.EmptyTar
      // Undeclared exception!
      try {
        exception.newInstance((Target<Object>) target);
        fail("Expecting exception: IllegalArgumentExcepti

      } catch(IllegalArgumentException e) {
         //
         // java.lang.Object is not an interface
         //
         verifyException("java.lang.reflect.Proxy$ProxyCl
      }
  }
```

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
builder	○	○	○	○
exception	○	○	○	○
task3	○	○	○	○
target	○	○	○	○

. (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 summaries, method names and variable names generated by our tool.

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

.

```
@Test(timeout = 4000)
  public void test08()  throws Throwable  {
      AsyncFeign.AsyncBuilder<IOException> asyncFeign_Asy
      AsyncFeign<IOException> asyncFeign0 = asyncFeign_As
```

```
        Class<Object> class0 = Object.class;
        Target.EmptyTarget<Object> target_EmptyTarget0 = Ta
        // Undeclared exception!
        try {
          asyncFeign0.newInstance((Target<Object>) target_E
          fail("Expecting exception: IllegalArgumentExcepti

        } catch(IllegalArgumentException e) {
           //
           // java.lang.Object is not an interface
           //
           verifyException("java.lang.reflect.Proxy$ProxyCl
        }
    }
```

. Here is the version of the test case using all the transformations provided by the proposed tool:

.

```
/**
 * 1. Creates a new AsyncFeign using "builder" using a ne
 * 2. Creates a new EmptyTarget "target" with Object.clas
 * 3. Expects an IllegalArgumentException when calling me
 *    "builder" with argument "target".
 */
@Test(timeout = 4000)
public void shouldThrowExceptionIfNoType()  throws Throwa
    AsyncFeign.AsyncBuilder<IOException> builder = new As
    AsyncFeign<IOException> exception = builder.build();
    Class<Object> task3 = Object.class;
    Target.EmptyTarget<Object> target = Target.EmptyTarge
    // Undeclared exception!
    try {
    exception.newInstance((Target<Object>) target);
    fail("Expecting exception: IllegalArgumentException")

    } catch(IllegalArgumentException e) {
        //
        // java.lang.Object is not an interface
        //
        verifyException("java.lang.reflect.Proxy$ProxyCla
    }
}
```

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.