Generating Mutated Strings for Automated Tests

Author: Swastik Agarwal
Supervisors: Carolin Brandt, Andy Zaidman
CSE 3000 Research Project
30 June 2021

TU Delft

1.Background

- **TestCube:** Creates test generation tools that developers love to use.
- **Problem:** TestCube mutates strings that are random and hard to understand.

2.Research Question

How can you design, implement and evaluate an extension to test amplification to generate easier to understand strings?

Subquestions:

1. What techniques are being used to mutate strings at the moment?
2. How to generate easier to understand Strings to extend test amplification?
3. How much does the test comprehension improve with easier to understand string inputs?

3.Approach

Output from TestCube's Implementation

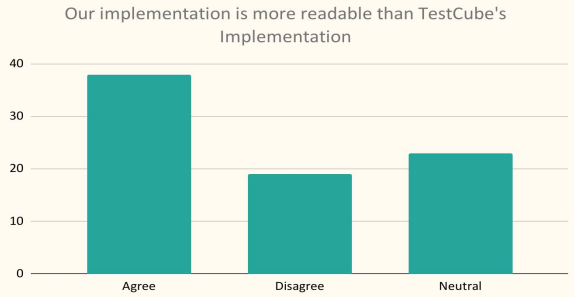
```
@Test
public void eq_mg52_assSep170() throws Exception {
    // MethodAdderOnExistingObjectsAmplifier: added method on existing object
    String __DSPOT_text_2 = "X(r!Fs2l>UgIvC=TU&zg";
    String h = "<p>Hello<p>there<p>world";
    Document doc = Jsoup.parse(h);
    doc.select("p").eq(1).text();
    doc.select("p").get(1).text();
    // MethodAdderOnExistingObjectsAmplifier: added method on existing object
    // AssertionGenerator: create local variable with return value of invocation
    Element o_eq_mg52__12 = doc.text(__DSPOT_text_2);
    // AssertionGenerator: add assertion
    Assertions.assertTrue(((Document) (o_eq_mg52__12)).hasText());
}
```

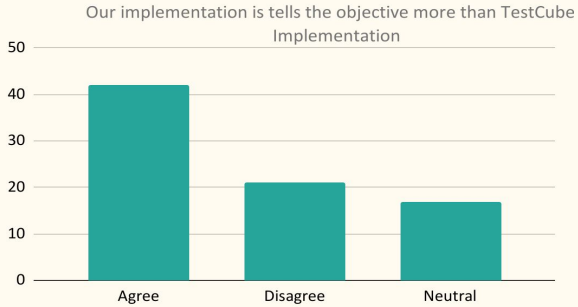
Separating alphabets and special characters and using readable words in the string.

```
@Test
public void eq_mg52_assSep110() throws Exception {
    // MethodAdderOnExistingObjectsAmplifier: added method on existing object
    String __DSPOT_text_2 = "TestChar_!/|]^)-&**\n";
    String h = "<p>Hello<p>there<p>world";
    Document doc = Jsoup.parse(h);
    doc.select("p").eq(1).text();
    doc.select("p").get(1).text();
    // MethodAdderOnExistingObjectsAmplifier: added method on existing object
    // AssertionGenerator: create local variable with return value of invocation
    Element o_eq_mg52__12 = doc.text(__DSPOT_text_2);
    // AssertionGenerator: add assertion
    Assertions.assertFalse(((Document) (o_eq_mg52__12)).hasParent());
}
```

Output from our implementation

4.Results





5.Conclusion

- Using readable words in strings makes randomly generated strings more readable.
- Adding more readable words in the readable word bank will increase readability.