Removing no longer used statements in amplified test cases

TU Delft

30/06/2021

Author: Wessel Oosterbroek / w.oosterbroek@student.tudelft.nl / Supervisor: Carolin Brandt / Responsible Professor: Andy Zaidman

CSE3000

① Goal

```java
public void booleanAttributesAreEmptyStringValues_assSep5() throws Exception {
    Document doc = Jsoup.parse("<div hidden>");
    Attributes attributes = new Attributes();
    Attribute first = doc.iterator().next();
    first.getKey();
    first.setKey(doc.toString());
    Assertions.assertFalse(((Collection) ((Document) (doc)).getAllElements())
        ).isEmpty());
}
```

Listing 1: An amplified test case created with the TestCube Intellij plugin.

```java
public void booleanAttributesAreEmptyStringValues_assSep5() throws Exception {
    Document doc = Jsoup.parse("<div hidden>");
    Assertions.assertFalse(((Collection) ((Document) (doc)).getAllElements())
        ).isEmpty());
}
```

Listing 2: An amplified test case created with the TestCube Intellij plugin, with unused statements removed.

② Redundant Statements

- Unused variables
- Statements from old assertions
- Statements that (indirectly) interact with the variable(s) being tested

For this project: Statements that do not have any influence on mutation coverage.

We need to detect and delete these statements

③ Detecting Redundant Statements

Slicing:

- Statements that are not in slice → redundant

Taint analysis:

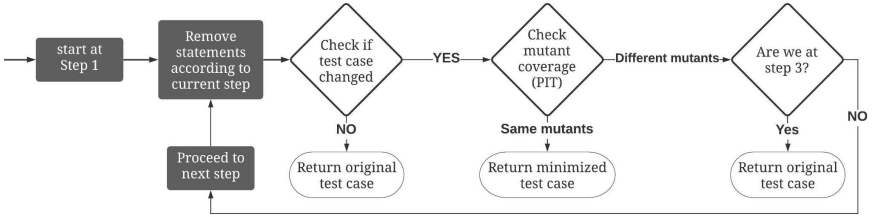
- Unused inputs → redundant

Code Analysis:

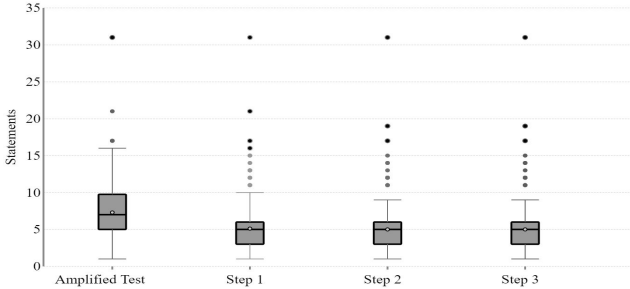
- Use analysis on the test case to remove redundant statements.
- Implemented algorithm using 3 steps:
  - Remove all statements, except those needed for compilation.
  - Remove all statements that do not interact with statements needed for compilation, as well as those that only do when being declared.
  - Remove all statements that do not interact with statements needed for compilation.

Verify using mutation testing → ✔

④ Implementation



⑤ Evaluation Results



⑥ Conclusion & Future Work

- Significant number of redundant statements are removed.
- Both dynamic slicing and taint analysis are interesting options that could be investigated in the future work.
- Try/Catch (code blocks)