Automated Test-Case Generation by Cloning

Mathias Landhäußer, Walter F. Tichy
Test Cloning
A New Way of Test Case Generation

- Similar classes can be tested with similar test cases; e.g.
  - Containers
  - Different list implementations
  - Converters

- Opportunity: Reuse a significant number of test cases

- Opportunity: Oracle can be reused as well
Preparatory Study
Does Cloning Happen in the Wild?

Manually, supported by the plagiarism detection tool JPlag

JPlag highlights source code that is identical or slightly modified

We examined pairs of files that have a similarity score of 50% or more

We counted obvious potential clones only

<table>
<thead>
<tr>
<th>Project</th>
<th>Tests Total</th>
<th>Potential Clones</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>args4j</td>
<td>95</td>
<td>40</td>
<td>42 %</td>
</tr>
<tr>
<td>log4j</td>
<td>583</td>
<td>106</td>
<td>18 %</td>
</tr>
<tr>
<td>collections</td>
<td>1085</td>
<td>61</td>
<td>6 %</td>
</tr>
<tr>
<td>configuration</td>
<td>1481</td>
<td>75</td>
<td>5 %</td>
</tr>
<tr>
<td>email</td>
<td>110</td>
<td>6</td>
<td>5 %</td>
</tr>
<tr>
<td>io</td>
<td>757</td>
<td>28</td>
<td>4 %</td>
</tr>
<tr>
<td>lang3</td>
<td>2098</td>
<td>130</td>
<td>6 %</td>
</tr>
<tr>
<td>primitives</td>
<td>808</td>
<td>102</td>
<td>13 %</td>
</tr>
<tr>
<td><strong>Total/Average</strong></td>
<td><strong>7017</strong></td>
<td><strong>548</strong></td>
<td><strong>8 %</strong></td>
</tr>
</tbody>
</table>
Test Cloning – Step by Step

Component Under Test

```java
public class TConverter {
    public double toFahrenheit(double celsius) {
        return (celsius / 5 * 9) + 32;
    }

    public double toCelsius(double fahrenheit) {
        return (fahrenheit - 32) * 5 / 9;
    }
}
```

Analog | Component Under Test

```java
public class Kilo2PoundConverter {
    public double toKilo(double pound) {
        return pound * 0.45359237;
    }

    public double toPound(double kilo) {
        return kilo * 2.20462262;
    }
}
```

Test Case

```java
public class TConverterTest {
   private static final double TOL = 1e-6;
   @Test
   public double backAndForthTest() {
       TConverter c = new TConverter();
       final int degreeC = 100;
       Assert.assertEquals(degreeC,
         c.toCelsius(c.toFahrenheit(degreeC)),
         TOL);
   }
}
```

Cloned Test Case

```java
public class Kilo2PoundConverterTest {
   private static final double TOL = 1e-6;
   @Test
   public double backAndForthTest() {
       Kilo2PoundConverter c = new Kilo2PoundConverter();
       final int degreeC = 100;
       Assert.assertEquals(degreeC,
         c.toPound(c.toKilo(degreeC)),
         TOL);
   }
}
```

Automation Possible (?)
public class TConverter {

    public double toFahrenheit(double celsius) {
        return (celsius / 5 * 9) + 32;
    }

    public double toCelsius(double fahrenheit) {
        return (fahrenheit - 32) * 5 / 9;
    }
}

public class Kilo2PoundConverter {

    /** @analog TConverter.toFahrenheit(double celsius) */
    public double toKilo(double pound) {
        return pound * 0.45359237;
    }

    /** @analog TConverter.toCelsius(double fahrenheit) */
    public double toPound(double kilo) {
        return kilo * 2.20462262;
    }
}

Identify and mark analogs

- Methods are analogs, if they share an abstract specification
- We use natural language processing to analyze method names

A test case is cloneable if the mapping is complete with respect to the test case; i.e. cloneable tests

- ... only use analog methods
- ... only use instance variables that are identical in the CuT and the model
Deriving Test Cases

List
Add-Operation
Append-Operation
Prepend-Operation
Remove-Operation
GetLength-Operation

javax.swing.JMenuItem
public Component add(Component comp)
public void remove(Component comp)
public int getMenuComponentCount()

Component JMenuItem.add((Component comp)
Appends a component to the end of this menu.
Returns the component added. […]
Parameters:
comp the component to add.
Returns:
the component added.
 […]

void JMenuItem.remove( (Component comp)
Removes the component c from this menu. […]
Parameters:
comp the component to be removed.
 […]

int JMenuItem.getMenuComponentCount()
Returns the number of components on the menu.
Returns:
an integer containing the number of components on the menu

public void listTestCase {
    LinkedList l = new LinkedList();
    Integer i1 = new Integer(1);
    assertEquals(0, l.size());

    c.append(i1);
    assertEquals(1, l.size());

    c.remove(i1);
    assertEquals(0, l.size());

}
Deriving Test Cases

**List**
- Add-Operation
- Append-Operation
- Prepend-Operation
- Remove-Operation
- GetLength-Operation

```java
public void listTestCase {
    LinkedList l = new LinkedList();
    Integer i1 = new Integer(1);
    assertEquals(0, l.size());
    c.append(i1);
    assertEquals(1, l.size());
    c.remove(i1);
    assertEquals(0, l.size());
}
```

```java
javax.swing.JMenu
public Component add(Component comp)
public void remove(Component comp)
public int getMenuComponentCount()
```

```java
public void listTestCase_Cloned() {
    JMenu l = new JMenu();
    Component i1 = new Button("1");
    assertEquals(0, l.getMenuComponentCount());
    f.add(i1);
    assertEquals(1, l.getMenuComponentCount());
    f.remove(i1);
    assertEquals(0, l.getMenuComponentCount());
}
```
How to Automate Analog Detection

- Automatic identification of method analogs via method names

- Split method names into words and find similar methods
  - Naïve assumption: Method names start with verbs (\texttt{add(Book b)})
  - Consider method and parameter names
  - Isolate verbs and retrieve synonyms from WordNet
  - Compute similarity score between model’s methods and CuT’s methods

- Even though the approach considers verbs only, the results are promising
  - Naming conventions help
  - Standard names help (sort, reverse, contains, …)
Cloning Case Study

- Cloning the oracle can lead to failing tests
- The tester has to decide whether the test is wrong or the component under test

```java
test case
@Test
public void convertToCelsiusTest() {
   TConverter converter = new TConverter();
   Assert.assertEquals(0, converter.toCelsius(32), TOLERANCE);
}

cloned test case
@Test
public void convertToCelsiusTest() {
   Kilo2PoundConverter converter = new Kilo2PoundConverter();
   Assert.assertEquals(0, converter.poundToKilo(32), TOLERANCE);
}
```
Cloning Case Study

- Cloning the oracle can lead to failing tests
- The tester has to decide whether the test is wrong or the component under test

```java
@Test
public void removeNullFirst() {
    LinkedList ll = new LinkedList();
    Integer element1 = null;
    Integer element2 = new Integer(2);
    ll.add(element1);
    ll.add(element2);
    assertTrue(ll.contains(element1));
    assertTrue(ll.contains(element2));
    assertEquals(2, ll.size());
    ll.remove(element1);
    assertEquals(1, ll.size());
    assertFalse(ll.contains(element1));
    assertTrue(ll.contains(element2));
}
```

```java
@Test
public void removeNullFirst() {
    JMenu ll = new JMenu();
    JMenuItem element1 = null;
    JMenuItem element2 = new JMenuItem("2");
    ll.add(element1);
    ll.add(element2);
    assertTrue(ll.isMenuComponent(element1));
    assertTrue(ll.isMenuComponent(element2));
    assertEquals(2, ll.getItemCount());
    ll.remove(element1);
    assertEquals(1, ll.getItemCount());
    assertFalse(ll.isMenuComponent(element1));
    assertTrue(ll.isMenuComponent(element2));
}
```
Cloning Case Study

- Cloning the oracle can lead to failing tests
- The tester has to decide whether the test is wrong or the component under test

```java
@Test
good:
public void addAndRemove() {
    LinkedList ll = new LinkedList();
    String e = "asdf1";
    assertEquals(0, ll.size());
    ll.add("asdf1");
    assertEquals(1, ll.size());
    ll.remove("asdf1");
    assertEquals(0, ll.size());
}
```

```java
cloned:
@Test
good:
public void addAndRemove() {
    Library ll = new Library();
    Book e = new Book("asdf1", "asdf1");
    assertEquals(0, ll.getNumberOfBooks());
    ll.addBook(e);
    assertEquals(1); ll.getNumberOfBooks();
    ll.remove(e);
    assertEquals(0, ll.getNumberOfBooks());
    ll.remove(e);
    assertEquals(0, ll.getNumberOfBooks());
}
```
Cloning Case Study

- Test cloning with 5 pairs of classes
  - 5 classes with tests, 5 classes without tests
  - 117 tests available, 85 of which were cloneable

- We generated 90 cloned tests
  - All clones compile
  - ~75% of the cloned tests succeed
  - ~15% of the cloned tests fail due to mismatching oracles
  - ~10% of the cloned tests detect defects
Outlook

- Transcription of test fixtures
- Evaluate if generic tests for design patterns could be transcribed
- Improve analog detection by using more sophisticated natural language processing
- Realistic evaluation on large benchmarks (also: does this approach save work?)
References


