Karlsruhe Institute of Technology

Automated Test-Case Generation by Cloning

Mathias Landhäußer, Walter F. Tichy

IPD Tichy, Department of Informatics

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

Project	Tests Total	Potential Clones	%
args4j	95	40	42 %
log4j	583	106	18 %
collections	1085	61	6 %
configuration	1481	75	5 %
email	110	6	5 %
io	757	28	4 %
lang3	2098	130	6 %
primitives	808	102	13 %
Total/Average	**7017**	**548**	**8 %**

Karlsruhe Institute of Technology

Test Cloning – Step by Step

component under test model

```
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

```
public class Kilo2PoundConverter {

  public double toKilo(double pound) {
    return pound * 0.45359237;
  }

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

automation possible (?)

test case

```
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

```
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!

Analogies Made Visible

```
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

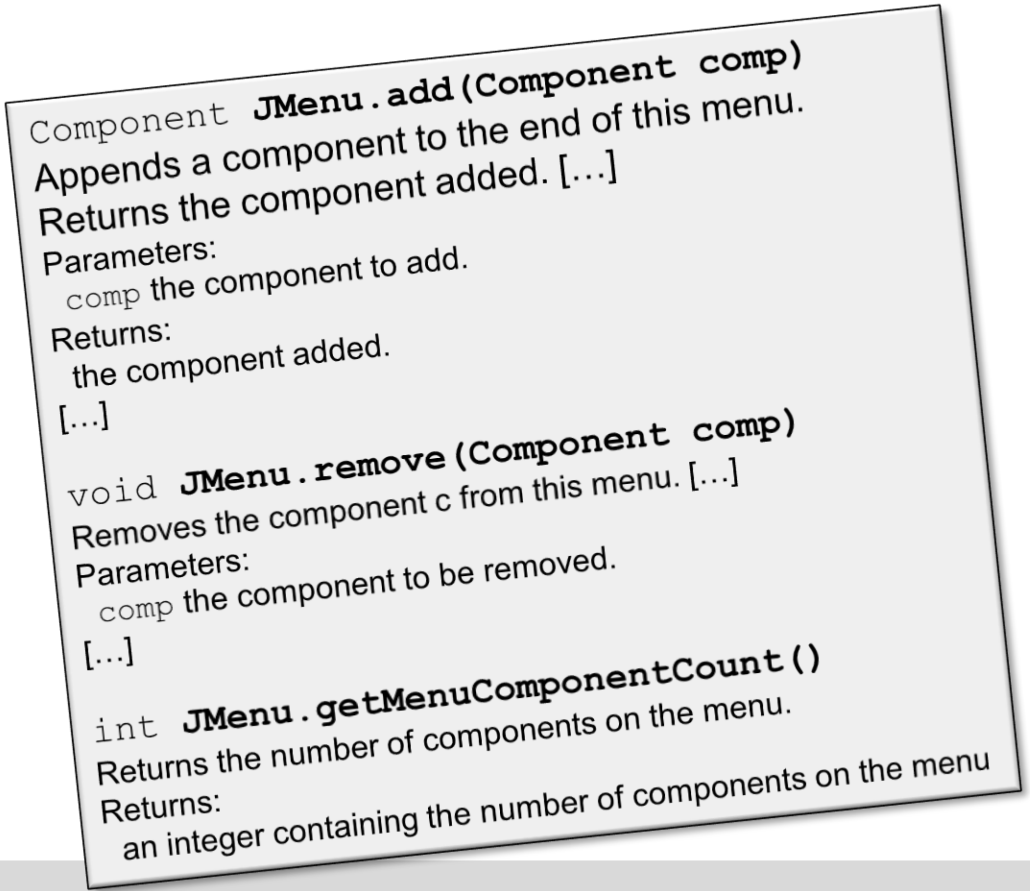
List
Add-Operation
  Append-Operation
  Prepend-Operation
Remove-Operation
GetLength-Operation

```
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());
}
```

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



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

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

Deriving Test Cases

List

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

```
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());
}
```

javax.swing.JMenu

public Component add(Component comp)
public void remove(Component comp)
public int getMenuComponentCount()

```
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 (**`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

test case

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

cloned test case

```
@Test
public void convertToCelciusTest() {
    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

test case

```
@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));
}
```

cloned test case

```
@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

test case

```
@Test
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());
}
```

cloned test case

```
@Test
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());
}
```

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

- [1] L. Prechelt, M. Philippsen, and G. Malpohl, “Finding plagiarisms among a set of programs with JPlag,”Journal of Universal Computer Science, vol. 8, no. 11, pp. 1016–1038, 2002.
- [2] C. K. Roy, J. R. Cordy, and R. Koschke, “Comparison and evaluation of code clone detection techniques and tools: A qualitative approach,” Science of Computer Programming – Special Issue on Program Comprehension (ICPC 2008), vol. 74, no. 7, pp. 470–495, 2009.
- [3] W. E. Howden, Software Testing and Validation Techniques, 2nd ed. New York: IEEE Computer Society Press, Jun. 1981, ch. Introduction to the Theory of Testing, pp. 16–19.
- [4] E. J. Weyuker, “On testing non-testable programs,” The Computer Journal, vol. 25, no. 4, pp. 465–470, 1982.
- [5] Z. Q. Zhou, D. H. Huang, T. H. Tse, Z. Yang, H. Huang, and T. Y. Chen, “Metamorphic testing and its applications,” in Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004), 2004, pp. 346–351.
- [6] T. Y. Chen, T. H. Tse, and Z. Zhou, “Fault-based testing in the absence of an oracle,” in Int. Computer Software and Applications Conf., 2001, pp. 172–178.
- [7] J. V. Gesser, “Javaparser – Java 1.5 Parser and AST,” http://code.google.com/p/javaparser/, accessed: 03/21/2012.
- [8] Eclipse Modeling Framework, http://eclipse.org/emf/, accessed: 03/21/2012.
- [9] P. D. Stotts, M. Lindsey, and A. Antley, “An informal formal method for systematic JUnit test case generation.” inXP/Agile Universe, ser. LNCS, D. Wells and L. A. Williams, Eds., vol. 2418. Springer, 2002, pp. 131–143.
- [10] C. Fellbaum, Ed., WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press, 1998.
- [11] Cycorp Inc., “ResearchCyc,” http://research.cyc.com/, ac-cessed: 03/21/2012.
- [12] E. Enslen, E. Hill, L. Pollock, and K. Vijay-Shanker, “Mining source code to automatically split identifiers for software analysis,” in6th IEEE Int. Working Conf. on Mining Software Repositories, 2009, MSR ’09, May 2009, pp. 71–80.