Project: Test-Splitter

Oguz Demir - University of Texas at Austin
oguz@utexas.edu

Motivation

Testing is the most widely used methodology for validating quality of software

Testing often requires a lot of manual effort

Testing can miss many bugs due to inadequate test suites

Automated testing can reduce the cost of testing and increase its effectiveness

Automate generation of unit tests (which test a few methods/classes at a time)

- Unit tests are useful in many contexts
  - Fast to execute and re-execute
    - Help in detecting regression failures
  - Easy to reason about
    - Help in locating and removing failures

Overview: Test-Splitter

Idea: Automate generation of suites of unit tests using existing system tests

- Leverage the wide availability of system tests and get more value from them

Goals – Long-term

- Generate tests for units that have no or very few unit tests
  - Speed up regression testing
    - When a small change is made, a small unit test can run instead of a large system test
  - Enhance debugging
- Improve generated tests, e.g., increase test execution performance
- Increase code coverage of system under test
- Support test maintenance, e.g., update tests when codebase evolves

Approach: Test-Splitter

Input: system test *s*

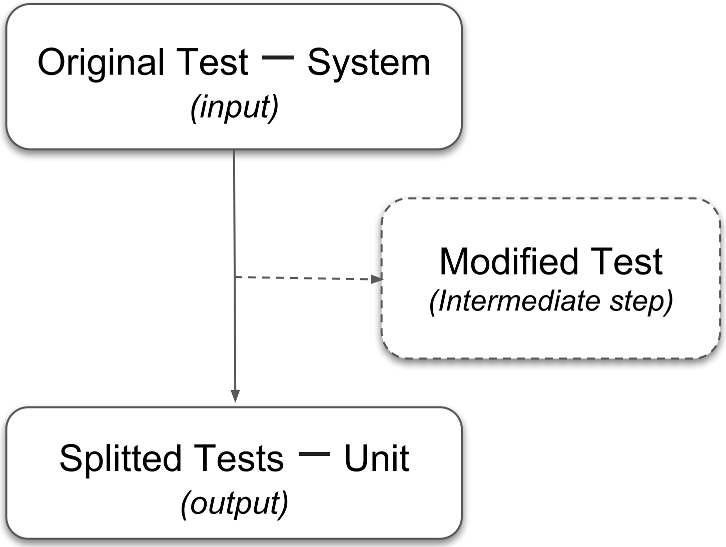
Output: unit tests $u_1, \ldots, u_k$

2-step procedure

- Instrument s with respect to split criteria *c*
    - Create instrumented system test *s*'
    - Create code for unit tests $u_1, \ldots, u_k$
- Run *s*'

The unit tests are ready for execution once the procedure terminates

Many split criteria can be defined, e.g., split at assertions or at method invocations



Example from JFreeChart Library

Original test:

```
@Test
public void testGetSubTaskCount() {
        Task t1 = new Task("T", new Date(100), new Date(200));
        assertEquals(0, t1.getSubtaskCount());
        t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
        assertEquals(1, t1.getSubtaskCount());
        Task s2 = new Task("S2", new Date(111), new Date(120));
        t1.addSubtask(s2);
        assertEquals(2, t1.getSubtaskCount());
        t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
        assertEquals(3, t1.getSubtaskCount());
        t1.removeSubtask(s2);
        assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Splitted tests:

"TaskTest_testGetSubTaskCount" is changed to **"testName"** for visibility.

```java
@Test
public void generatedU1() {
    Task t1 = new Task("T", new Date(100), new Date(200));
    assertEquals(0, t1.getSubtaskCount());
}

@Test
public void generatedU2() {
    Task t1 = (Task) ObjectRecorder.readObject("testName",1);
    // Split Point: 1
    t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
    assertEquals(1, t1.getSubtaskCount());
}

@Test
public void generatedU3() {
    Task t1 = (Task) ObjectRecorder.readObject("testName",2);
    // Split Point: 2
    Task s2 = new Task("S2", new Date(111), new Date(120));
    t1.addSubtask(s2);
    assertEquals(2, t1.getSubtaskCount());
}
```

```java
@Test
public void generatedU4() {
    Task s2 = (Task)ObjectRecorder.readObject("testName",3);
    Task t1 = (Task)ObjectRecorder.readObject("testName" , 3);
    // Split Point: 3
    t1.addSubtask(new Task("S3",new Date(121),new Date(130)));
    assertEquals(3, t1.getSubtaskCount());
}

@Test
public void generatedU5() {
    Task s2 = (Task)ObjectRecorder.readObject("testName" , 4);
    Task t1 = (Task)ObjectRecorder.readObject("testName" , 4);
    // Split Point: 4
    t1.removeSubtask(s2);
    assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Modified test which will record the snapshots:

"TaskTest_testGetSubTaskCount" is changed to **"testName"** for visibility.

```
@Test
public void testGetSubTaskCount() {
    Task t1 = new Task("T", new Date(100), new Date(200));
    assertEquals(0, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", t1, 1);
    ObjectRecorder.finalizeWriting("testName", 1);
    // Split Point: 1
    t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
    assertEquals(1, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", t1, 2);
    ObjectRecorder.finalizeWriting("testName", 2);
    // Split Point: 2
    Task s2 = new Task("S2", new Date(111), new Date(120));
    t1.addSubtask(s2);
    assertEquals(2, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", s2, 3);
    ObjectRecorder.writeObject("testName", t1, 3);
    ObjectRecorder.finalizeWriting("testName", 3);
    // Split Point: 3
    t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
    assertEquals(3, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", s2, 4);
    ObjectRecorder.writeObject("testName", t1, 4);
    ObjectRecorder.finalizeWriting("testName", 4);
    // Split Point: 4
    t1.removeSubtask(s2);
    assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Snapshots taken:

```
@Test
public void testGetSubTaskCount() {
    Task t1 = new Task("T",new Date(100),new Date(200));
    assertEquals(0, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", t1, 1);
    ObjectRecorder.finalizeWriting("testName", 1);
    // Split Point: 1
    t1.addSubtask(new Task("S1",new Date(100),new Date(110)));
    assertEquals(1, t1.getSubtaskCount());
    ObjectRecorder.writeObject("testName", t1, 2);
    ObjectRecorder.finalizeWriting("testName", 2);
    // Split Point: 2
    ..
    ..
    ..
}
```

```
<org.jfree.data.gantt.Task>
  <description>T</description>
  <duration class="org.jfree.data.time.SimpleTimePeriod">
    <start>100</start>
    <end>200</end>
  </duration>
  <subtasks/>
</org.jfree.data.gantt.Task>
```

```
<org.jfree.data.gantt.Task>
  <description>T</description>
  <duration class="org.jfree.data.time.SimpleTimePeriod">
    <start>100</start>
    <end>200</end>
  </duration>
  <subtasks>
    <org.jfree.data.gantt.Task>
      <description>S1</description>
      <duration class="org.jfree.data.time.SimpleTimePeriod">
        <start>100</start>
        <end>110</end>
      </duration>
      <subtasks/>
    </org.jfree.data.gantt.Task>
  </subtasks>
</org.jfree.data.gantt.Task>
```

Example from Commons-Codec Library

Original test:

```
@Test
public void testMd2HexLength() {
    String hashMe = "this is some string that is longer than 32 characters";
    String hash = DigestUtils.md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
    hashMe = "length < 32";
    hash = DigestUtils.md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
}
```

Example from Commons-Codec Library

Splitted tests:

"DigestUtilsTest_testMd2HexLength" is changed to **"testName"** for visibility.

```
@Test
public void generatedU1() {
    String hashMe = "this is some string that is longer than 32 characters";
    String hash = DigestUtils.md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
}

@Test
public void generatedU2() {
    String hash = (String) ObjectRecorder.readObject( "testName", 1);
    String hashMe = (String) ObjectRecorder.readObject( "testName", 1);
    // Split Point: 1
    hashMe = "length < 32";
    hash = DigestUtils.md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
}
```

Example from Commons-Codec Library

Modified test which will record the snapshots:

```
@Test
public void testMd2HexLength() {
   String hashMe = "this is some string that is longer than 32
characters";
   String hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
   ObjectRecorder.writeObject( "testName", hash, 1);
   ObjectRecorder.writeObject( "testName", hashMe, 1);
   ObjectRecorder.finalizeWriting( "testName", 1);
   // Split Point: 1
   hashMe = "length < 32";
   hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
}
```

"DigestUtilsTest_testMd2HexLength " is changed to **"testName"** for visibility.

Current Results with 5 Different Open-Source Projects

	Original #Tests	**#Tests after Split**	**Total size of Snapshots**	**#Snapshots**
JFreeChart	2176	6620	212.1 MB	4444
Commons-lang	4417	6856	23.1 MB	2175
commons-codec	887	1208	2.6 MB	312
commons-net	287	292	20 KB	6
commons-io	1349	1648	971 KB	167

- All modified tests are executed to generate all snapshots.
- Split criteria: split at assertion blocks

Results for Commons-Lang

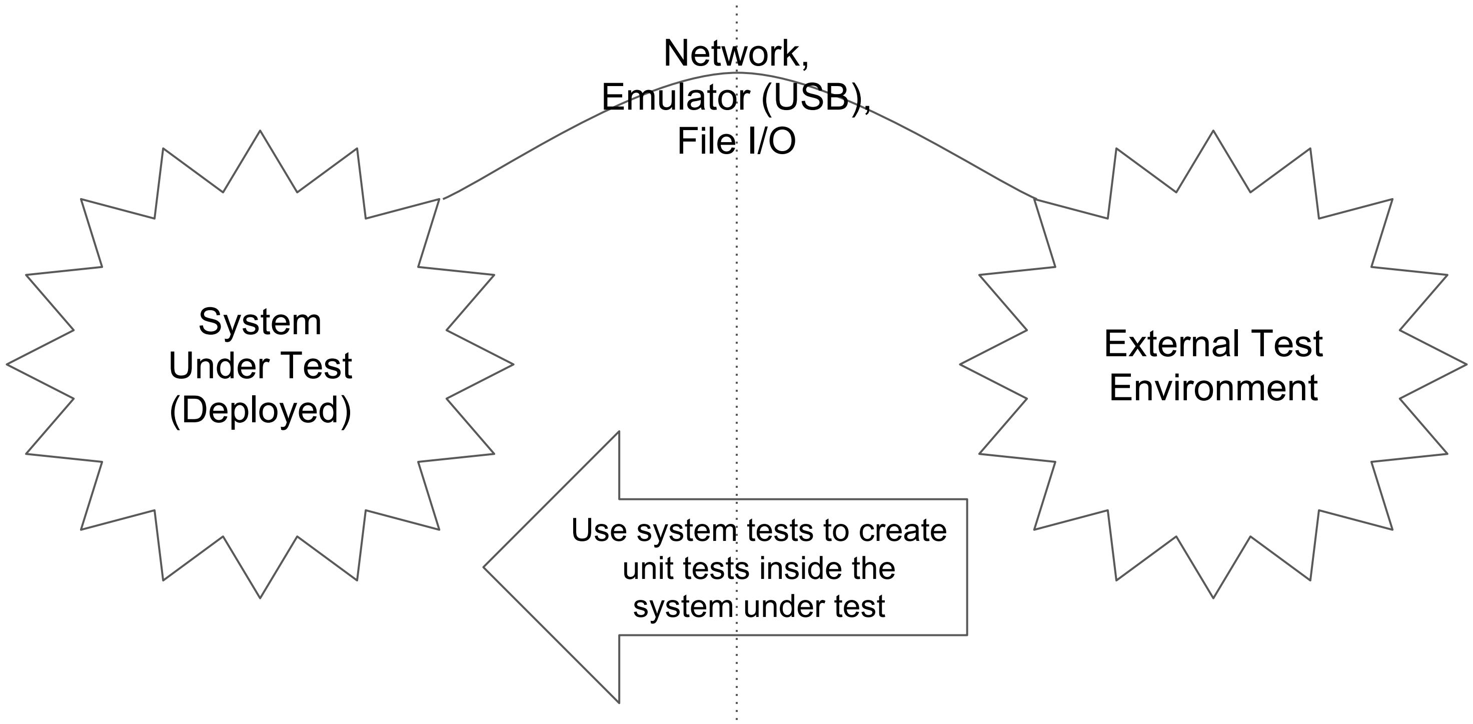
Module Name	Original Tests	Generated Tests	Errors	Failures	Elapsed Time	Time For Snapshots	Original Elapsed Time
org.apache.commons.lang3.reflect	149	256	2	0	0.296	0.268	0.138
org.apache.commons.lang3.concurrent	162	165	0	0	1.355	1.297	1.406
org.apache.commons.lang3.exception	76	99	0	0	0.055	0.051	0.022
org.apache.commons.lang3.text	291	764	1	0	0.269	0.244	0.218
org.apache.commons.lang3.event	18	34	0	1	0.034	0.012	0.031
org.apache.commons.lang3.math	140	304	0	0	0.081	0.072	0.061
org.apache.commons.lang3.mutable	163	195	0	0	0.021	0.019	0.009
org.apache.commons.lang3.text.translate	21	34	0	0	0.002	0.003	0.002
org.apache.commons.lang3.time	1450	1650	2	0	9.882	9.975	9.366
org.apache.commons.lang3.builder	425	684	2	0	2.156	2.289	2.128
org.apache.commons.lang3	1617	2598	5	0	0.563	0.599	0.745
org.apache.commons.lang3.tuple	60	77	0	0	0.013	0.009	0.016
	4572	6860	12	1	14.727	14.838	14.142

Optimization Ideas: Simplification

```
@Test
public void generatedU4() {
  Task s2 = (Task) ObjectRecorder.readObject("testName" , 3);
  Task t1 = (Task) ObjectRecorder.readObject("testName" , 3);
  // Split Point: 3
  t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
  assertEquals(3, t1.getSubtaskCount());
}
```

- Task s2 object is written to disk in snapshot generation and read back from disk for executing generatedU4.
- However, s2 object is not used during the execution of generatedU4.
- Therefore, this object can be omitted from both snapshot file and method generatedU4.

Feature Ideas For Creating Unit Tests



Using Assertions In Source Code to Create Tests

```
public void someMethod() {
      …
      assert …
      t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
      assert …
      …
}
```

```
Carved test: addSubtask
```

Combinations to Increase Coverage

```
@Test
public void test1() {
      …
      someMethod(“ss” , -1 , 2.00, ‘a’);
      …
}
```

```
@Test
public void test2() {
      …
      someMethod(“aa” , 11 , 1.90, ‘x’);
      …
}
```

```
@Test
public void test1’() {
      …
      someMethod(“aa” , -1 , 1.90, ‘a’);
      …
}
```

```
@Test
public void test2’() {
      …
      someMethod(“ss” , 11 , 2.00, ‘x’);
      …
}
```