Making Software Product Line Evolution Safer

Felype Santiago
Advisor: Paulo Borba
Co-advisor: Rohit Gheyi
Software Product Lines (SPL)
Software Product Lines (SPL)

public class Game {
    public class Connection {
        public class Startup {
            public class Bluetooth extends Multiplayer {
                public class Multiplayer {
                    public class Internet {
                        public int limit() {
                            return 4;
                        }
                    }
                }
            }
        }
    }
}

<table>
<thead>
<tr>
<th>Costs</th>
<th>Quality</th>
<th>Time to market</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Feature Model

Mandatory
Optional
Xor
Or
Feature Model (FM)

• Valid configurations
  – SPL products

Diagram:
- Game
  - Multiplayer
    - Internet
    - Bluetooth
  - Startup
    - Connection
Configuration Knowledge (CK)

- Game
  - Multiplayer
  - Startup
    - Internet
    - Bluetooth
      - Connection

<table>
<thead>
<tr>
<th>Feature Expression</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game ^ Startup</td>
<td>Game.java, Startup.java</td>
</tr>
<tr>
<td>Multiplayer</td>
<td>Multiplayer.java</td>
</tr>
<tr>
<td>Internet</td>
<td>Internet.java, Connection.java</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Bluetooth.java</td>
</tr>
</tbody>
</table>
Safe Composition Verification

Well-formed (WF) products

```
public class Game {
    public class Multiplayer {
        public class Internet
            extends Multiplayer{
                public int limit() {
                    return 0;
                }
            }
    }
```
Theory of SPL Refinement

- Safe evolution
SPL evolution

• Changes manually applied
• Eclipse, Netbeans, ...
  – Only code assets
Dependencies
Dependencies

- \{\text{Game, Startup, Multiplayer, Bluetooth}\}
Behavioral Changes
Problem

• Changes manually applied may introduce errors
  – Invalid products
  – Behavioral changes

• Risks to quality and productivity
Proposed Solutions

• Approaches to check product line evolutions
• Optimizations and approximations
All Product Pairs (APP)

1. Checking well-formedness
2. Mapping corresponding products
3. Generating target products with their corresponding source products
4. If they do not match, checking if other target product matches

FM: Feature model
Code: Code assets
CK: Configuration knowledge
COB: Compatible observable behavior

Configuration leads to well formed product
Concrete product
APP – Exemplo 1

1 - Checking well-formedness
2 - Mapping corresponding products
3 - Generating target products with their corresponding source products
4 - If they do not match, checking if other target product matches

{Game, Multiplayer, Startup, Bluetooth}
APP – Exemplo 2

1 - Checking well-formedness
2 - Mapping corresponding products
3 - Generating target products with their corresponding source products
4 - If they do not match, checking if other target product matches
Restrictions

1 - Checking well-formedness

2 - Mapping corresponding products

3 - Generating target products with their corresponding source products

4 - If they do not match, checking if other target product matches
All Products (AP)

1. Checking well-formedness
2. Mapping corresponding products
3. Generating target products with their corresponding source products
4. If they do not match, checking if other target product matches

FM: Feature model
Code: Code assets
CK: Configuration knowledge
COB: Compatible observable behavior

Configuration leads to well-formed product
Concrete product
Optimized Approaches

- Bypass assets and products checking
Impacted Products (IP)

• Check only products that contain changed assets

1 - Checking well-formedness
2 - Mapping corresponding products
3 - Generating target products with their corresponding source products
4 - If they do not match, checking if other target product matches
Impacted Products (IP)

```
public class Connection {
}

public class Internet extends Multiplayer{
    private Connection con;
    public int limit(long l) {
        return 10;
    }
    public int test() {
        return limit(0);
    }
}
```

```
public class Connection {
    public int limit(long l) {
        return 10;
    }
}
```

```
public class Internet extends Multiplayer{
    private Connection con;
    public int test() {
        return con.limit(0);
    }
}
```

<table>
<thead>
<tr>
<th>Feature Expression</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game ^ Startup</td>
<td>Game.java, Startup.java</td>
</tr>
<tr>
<td>Multiplayer</td>
<td>Multiplayer.java</td>
</tr>
<tr>
<td>Internet</td>
<td>Internet.java, Connection.java</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Bluetooth.java</td>
</tr>
</tbody>
</table>

Changed assets
Impacted Classes (IC)

- Check only changed classes
- Generates subproducts

1. Checking well-formedness
2. Mapping corresponding products
3. Generating target products with their corresponding source products
4. If they do not match, checking if other target product matches
public class Connection {
    public int limit(long l) {
        return 10;
    }
}

public class Internet extends Multiplayer{
    private Connection con;
    public int test() {
        return con.limit(0);
    }
}
Impacted Classes (IC)

```
public class Connection {
}

public class Internet extends Multiplayer{
    private Connection con;
    public int limit(long l) {
        return 10;
    }
    public int test() {
        return limit(0);
    }
}
```

```
public class Connection {
    public int limit(long l) {
        return 10;
    }
}
```

```
public class Internet extends Multiplayer{
    private Connection con;
    public int test() {
        return con.limit(0);
    }
}
```

### Feature Expression
- Game
- Multiplayer
- Startup
- Internet
- Bluetooth
- Connection

### Assets

<table>
<thead>
<tr>
<th>Feature Expression</th>
<th>Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game &amp; Startup</td>
<td>Game.java, Startup.java</td>
</tr>
<tr>
<td>Multiplayer</td>
<td>Multiplayer.java</td>
</tr>
<tr>
<td>Internet</td>
<td>Internet.java, Connection.java</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Bluetooth.java</td>
</tr>
</tbody>
</table>

Changed assets
Implementation

• Alloy
  – Possible product configurations
  – WF verification
• Java
  – FM and CK refinement
  – Mapping of likely corresponding products
• Soot
  – Class dependences analysis
• SafeRefactor – Randoop
  – Product refinement – Unit tests generation
SafeRefactor
Randoop

public class Internet {
    public int limit() {
        return 0;
    }
    public int time() {
        return 0;
    }
}

public class Internet {
    public int limit() {
        return 10;
    }
    public int time() {
        return 0;
    }
}

public void test() {
    Internet var0 = new Internet();
    int var1 = var0.limit();
    assertTrue(var1 == 0);
}
Feature Model
<?xml version="1.0"?>
<configurationModel>
  <configuration>
    <expression>MobileMedia</expression>
    <required/></required>
    <provided>BaseThread.java,ControllerInterface.java</provided>
  </configuration>

  <configuration>
    <expression>CopyPhoto</expression>
    <required>ControllerInterface.java</required>
    <provided>PhotoViewController.java</provided>
  </configuration>
</configurationModel>
Evaluation

• Product lines ranging up 32 KLOC
  – TaRGeT
  – Mobile Media
  – Game (Motivating examples)
## Subjects

<table>
<thead>
<tr>
<th>Subject</th>
<th>SPL</th>
<th>KLOC</th>
<th>Features</th>
<th>Products</th>
<th>Changed Artifacts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motivating Example 1</td>
<td>0.03 ➔ 0.03</td>
<td>6 ➔ 6</td>
<td>8 ➔ 8</td>
<td>Code</td>
<td>Invalid product configurations</td>
</tr>
<tr>
<td>2</td>
<td>Motivating Example 2</td>
<td>0.03 ➔ 0.03</td>
<td>6 ➔ 6</td>
<td>8 ➔ 8</td>
<td>Code</td>
<td>Behavioral change in 1 class</td>
</tr>
<tr>
<td>3</td>
<td>MobileMedia v3’</td>
<td>1.36 ➔ 1.37</td>
<td>8 ➔ 8</td>
<td>9 ➔ 9</td>
<td>Code</td>
<td>Behavioral change in 2 classes</td>
</tr>
<tr>
<td>4</td>
<td>MobileMedia v4’</td>
<td>1.56 ➔ 1.56</td>
<td>9 ➔ 9</td>
<td>21 ➔ 21</td>
<td>Code</td>
<td>Behavioral change in 2 classes</td>
</tr>
<tr>
<td>5</td>
<td>MobileMedia v4’’</td>
<td>1.56 ➔ 1.56</td>
<td>9 ➔ 9</td>
<td>21 ➔ 21</td>
<td>Code</td>
<td>Behavioral change in 5 classes</td>
</tr>
</tbody>
</table>

### Catalog of defective refinements

<table>
<thead>
<tr>
<th>Subject</th>
<th>SPL</th>
<th>KLOC</th>
<th>Features</th>
<th>Products</th>
<th>Changed Artifacts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>TaRGeT</td>
<td>32.54 ➔ 32.54</td>
<td>22 ➔ 22</td>
<td>1008 ➔ 1512</td>
<td>FM</td>
<td>Replace alternative</td>
</tr>
<tr>
<td>7</td>
<td>TaRGeT</td>
<td>20.79 ➔ 20.76</td>
<td>12 ➔ 12</td>
<td>12 ➔ 12</td>
<td>Code</td>
<td>Removal of warnings</td>
</tr>
<tr>
<td>8</td>
<td>TaRGeT</td>
<td>14.64 ➔ 14.82</td>
<td>9 ➔ 9</td>
<td>1 ➔ 1</td>
<td>FM, CK, code</td>
<td>Extract feature Output</td>
</tr>
<tr>
<td>9</td>
<td>TaRGeT</td>
<td>14.82 ➔ 15.17</td>
<td>9 ➔ 10</td>
<td>1 ➔ 2</td>
<td>FM, CK, code</td>
<td>Add feature TC4 Output</td>
</tr>
<tr>
<td>10</td>
<td>TaRGeT</td>
<td>28.13 ➔ 28.37</td>
<td>13 ➔ 14</td>
<td>24 ➔ 36</td>
<td>FM, CK, code</td>
<td>Add feature HTML Output</td>
</tr>
<tr>
<td>11</td>
<td>TaRGeT</td>
<td>28.67 ➔ 29.07</td>
<td>16 ➔ 17</td>
<td>48 ➔ 60</td>
<td>FM, CK, code</td>
<td>Add feature XML TestLink Output</td>
</tr>
<tr>
<td>12</td>
<td>TaRGeT</td>
<td>29.76 ➔ 30.34</td>
<td>18 ➔ 19</td>
<td>140 ➔ 168</td>
<td>FM, CK, code</td>
<td>Add feature STD Output</td>
</tr>
<tr>
<td>14</td>
<td>TaRGeT</td>
<td>18.63 ➔ 22.04</td>
<td>12 ➔ 13</td>
<td>12 ➔ 24</td>
<td>FM, CK, code</td>
<td>Add feature CNL</td>
</tr>
<tr>
<td>15</td>
<td>TaRGeT</td>
<td>15.30 ➔ 20.21</td>
<td>10 ➔ 11</td>
<td>2 ➔ 4</td>
<td>FM, CK, code</td>
<td>Add feature CM</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Subject</th>
<th>Time (min)</th>
<th>Generated Products</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>APP</td>
<td>AP</td>
<td>IP</td>
</tr>
<tr>
<td>1</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>0.16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0.86</td>
<td>0.89</td>
<td>0.87</td>
</tr>
<tr>
<td>4</td>
<td>2.92</td>
<td>0.18</td>
<td>1.63</td>
</tr>
<tr>
<td>5</td>
<td>1.32</td>
<td>0.38</td>
<td>1.36</td>
</tr>
<tr>
<td>6</td>
<td>29.16</td>
<td>30.44</td>
<td>13.31</td>
</tr>
<tr>
<td>7</td>
<td>0.17</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>1.93</td>
<td>1.93</td>
<td>0.05</td>
</tr>
<tr>
<td>9</td>
<td>61.19</td>
<td>63.42</td>
<td>0.10</td>
</tr>
<tr>
<td>10</td>
<td>11.97</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>29.27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>82.21</td>
<td>80.93</td>
<td>0.17</td>
</tr>
<tr>
<td>13</td>
<td>30.82</td>
<td>28.53</td>
<td>0.07</td>
</tr>
<tr>
<td>14</td>
<td>0.18</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Analysis of Results

• False negatives - The tool answers “SPL not refined”
  – Different set of public methods (APP)
  – Methods tested under different conditions that they are normally exposed in the line

• False-positives - The tool answers “SPL is refined”
  – Low coverage of the tests generated by Randoop
  – Changes only in GUI or output files

• False-positives e false-negatives
  – Outputs may be non-deterministic in concurrent scenarios
Random Commits

• 35 pairs of TaRGeT’s Commits
  – from: random number
  – to: from+1

• Manual analysis
Results

TOTAL TIME (MEAN OF 5 EXECUTIONS)

Time (minutes)

Random Pairs
## Results

<table>
<thead>
<tr>
<th>Approach</th>
<th>Hits</th>
<th>False positives</th>
<th>False negatives</th>
<th>Time (media)</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP</td>
<td>31</td>
<td>2</td>
<td>1</td>
<td>69.30</td>
</tr>
<tr>
<td>AP</td>
<td>31</td>
<td>3</td>
<td>0</td>
<td>72.56</td>
</tr>
<tr>
<td>IP</td>
<td>31</td>
<td>3</td>
<td>0</td>
<td>30.36</td>
</tr>
<tr>
<td>IC</td>
<td>31</td>
<td>3</td>
<td>0</td>
<td>14.27</td>
</tr>
</tbody>
</table>
Changes in UI components

```java
@override
protected void createFormContent(IMangedForm managedForm)
{
    ScrolledForm form = managedForm.getForm();
    FormToolkit toolkit = managedForm.getToolkit();
    form.setText("Test Comparison");

    GridLayout layout = new GridLayout();
    layout.setColumns = 2;
    form.getBody().setLayout(layout);

    this.createComparisonResultsSection(form, toolkit);
    this.createCompareTestCasesSection(form, toolkit);
}

/** *
 * Creates the Comparison Results Section, with two tables. The first represents *
 * test cases. The second represents all the old test cases and their similar *
 * test case in the previous table. *
 * @param form The form widget managed by this form. *
 * @param toolkit The toolkit used by this form. *
 */
private void createComparisonResultsSection( final ScrolledForm form, FormToolkit toolkit )
{
    this.comparisonResultsSection = TCGUtil.createSection(form, toolkit, Section.TITLE_BAR | Section.EXPANDED);
    Composite client = TCGUtil.createComposite(toolkit, this.comparisonResultsSection);
    client.setLayoutData(new GridData(GridData.VERTICAL_ALIGN_TOP, GridData.FILL_HORIZONTAL));
    client.setLayoutData(new GridData(GridData.VERTICAL_ALIGN_TOP, GridData.FILL_HORIZONTAL));
    GridLayout layout = (GridLayout) client.getLayout();
    layout.setSpacing = 8;
    layout.setSpacing = 0;
    this.comparisonResultsSection.setText("Comparison Results");
}
```
Construct Validity

• Open world assumption
```java
class Alfa {
    public int methAlfa(int timeout) {
        if (timeout < 0)
            return 0;
        else
            return timeout;
    }
}
```
Internal Validity

- The same machine for all executions
- Manual analysis of Commits
External Validity

- Selection of subjects
Final Remarks

• Four approaches to check evolutions
• Optimized approaches with significant time reduction
• Errors detected earlier
Related Work

- Using regression testing to analyze the impact of changes to variability models on products. (Heider et al.)
- Reducing combinatorics in testing product lines. (Kim et al.)
- Improving the Testing and Testability of Software Product Lines (IIsis et al.)
Future Work

• Evaluation
  – More contexts
  – Formulas

• Optimizations
  – Incremental compilation
  – Parallelism

• Eclipse plugin
Making Software Product Line Evolution Safer

Felype Santiago
Advisor: Paulo Borba
Co-advisor: Rohit Gheyi
class Alfa {
  protected int methAlfa() {
    return 1;
  }
}

class Beta extends Alfa {
  public int methBeta() {
    return super.methAlfa();
  }
}

class Alfa {
  protected int methAlfa() {
    return 0;
  }
}

class Beta extends Alfa {
  public int methBeta() {
    return super.methAlfa();
  }
}

Class Alfa doesn’t use methAlfa()
Currently, Randoop checks for a small set of contracts:

- **Equals to null**: o.equals(null) should return false
- **Reflexivity of equality**: o.equals(o) should return true
- **Symmetry of equality**: o1.equals(o2) implies o2.equals(o1)
- **Equals-hashcode**: If o1.equals(o2)==true, then o1.hashCode() == o2.hashCode()
- **No null pointer exceptions**: No NullPointerException is thrown if no null inputs are used in a test.