Similarity-Based Prioritization in Software Product-Line Testing

Mustafa Al-Hajjaji, Thomas Thüm, Jens Meinicke, Malte Lochau, Gunter Saake

(1) Otto-von-Guericke-Universität Magdeburg
(2) TUD – Technische Universität Darmstadt

FOSD Meeting 2014, Schloss Dagstuhl
Motivation

- Testing a SPL is a difficult task
  - Explosion of possible products $2^n$; where $n$ the feature number.

- Reduce the time to detect a defect?

Feature model of Mobile Phone SPL

```
c1 = {MobilePhone, Calls, Screen, Colour, Media, MP3, GPS}
c2 = {MobilePhone, Calls, Screen_High_resolution, Media, Camera}
c3 = {MobilePhone, Calls, Screen_High_resolution}
c4 = {MobilePhone, Calls, Screen, Basic}
c5 = {MobilePhone, Calls, Screen_High_resolution, GPS}
c6 = {MobilePhone, Calls, Screen, Basic, Media, MP3}
c7 = {MobilePhone, Calls, Screen, Colour}
c8 = {MobilePhone, Calls, Screen_High_resolution, Media, MP3}
c9 = {MobilePhone, Calls, Screen, Colour, GPS}
c10 = {MobilePhone, Calls, Screen, High_resolution, Media, MP3, GPS}
c11 = {MobilePhone, Calls, Screen, High_resolution, Media, MP3, Camera}
c12 = {MobilePhone, Calls, Screen_High_resolution, Media, Camera, GPS}
c13 = {MobilePhone, Calls, Screen_High_resolution, Media, MP3, Camera}
```
Defect Features

- Unit tests may find defect inside a single feature
  - n test suites required for a product line with n features.

Feature model of Mobile Phone SPL
Interaction defects

- 2-wise interaction defect
  - Reproducible by including 2 specific features

Feature model of Mobile Phone SPL
Interaction defect

- 3-wise interaction defect
  - Reproducible by including 3 specific features

Feature model of *Mobile Phone* SPL
Motivation

- Kuhn et al. (2004)
  - Pairwise interaction → 70% of defects,
  - 3-wise interaction → 95% of defects,
  - 6-wise interaction → almost all the defects.

- Sampling algorithms
  - CASA (Garvin et al. 2011),
  - Chvatal (Chvatal 1979),
  - And ICPL (Johansen et al. 2012)
Similarity-based Prioritization

1. **Defect?**
   - Yes: Fix & start over
   - No: Select the next product

2. **Test**

3. **Generate all valid configurations**

4. **Generate samples**

5. **List of configurations**

6. **Generate**

7. **List of sorted configurations**

8. **Prioritize**

9. **Sampling**

10. **Select the next product**
Similarity-based Prioritization

- Dissimilar test cases are likely to detect more defects than the similar ones!! (Hemmati et al. 2010)

- Hamming distance

\[
d(c_i, c_j, F) = 1 - \frac{|c_i \cap c_j| + |(F' \setminus c_i) \cap (F \setminus c_j)|}{|F|}
\]

where \( c_i \) and \( c_j \) are configurations and \( F \) is the set of all features in a SPL.
Similarity-based Prioritization

- The configuration with the maximum number of selected features
  - Covers most defects in individual features
  - Selection of the next configuration with large distance
  - Common in the Linux community (a.k.a. allyesconfig) (Dietrich et al., 2012)
Configuration Creation in FeatureIDE
Evaluation

• Two case studies:
  – Mobile Phone SPL (10 features)
  – Smart Home SPL (60 features)

• We simulate defects
  – Caused by single features.
  – Occurring because of pairwise interactions
Potential defects

- We simulate five kinds of potential defects
  - $C_i = \{c | c \in SPL \land f1 \in C\}$
    - E.g., division by zero
  - $C_i = \{c | c \in SPL \land f1 \notin C\}$
    - E.g., $f1$ initializes a variable, $F1$ is removed.
  - $C_i = \{c | c \in SPL \land f1, f2 \in C\}$
    - E.g., one feature calls a method in another feature and the retrieved value is wrong
  - $C_i = \{c | c \in SPL \land f1 \in C \land f2 \notin C\}$
    - E.g., one feature calls a method from a feature that is not selected
  - $C_i = \{c | c \in SPL \land f1, f2 \notin C\}$
    - E.g., $f1$ and $f2$ initializes a variable, $f1$ and $f2$ are removed
Evaluation

The number of configurations to detect all defects; D- default order of each algorithm, P- similarity-based prioritization approach.

<table>
<thead>
<tr>
<th>SPL</th>
<th>Sampling algorithm</th>
<th>Default order</th>
<th>Similarity-based prioritization</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Phone SPL</td>
<td>ICPL</td>
<td>1.5</td>
<td>1.3</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>Chvatal</td>
<td>1.7</td>
<td>1.3</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td>CASA</td>
<td>1.7</td>
<td>1.2</td>
<td>29%</td>
</tr>
<tr>
<td>Smart Home SPL</td>
<td>ICPL</td>
<td>1.08</td>
<td>1.08</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Chvatal</td>
<td>1.80</td>
<td>1.50</td>
<td>17%</td>
</tr>
<tr>
<td></td>
<td>CASA</td>
<td>1.90</td>
<td>1.60</td>
<td>16%</td>
</tr>
</tbody>
</table>

Average number of configurations to detect a defects
Evaluation

![Graph for Mobile Phone SPL](image1)

![Graph for Smart Home SPL](image2)

Mobile Phone SPL  
Smart Home SPL

Average number of configurations to detect a defect for 200 random orders
## Evaluation

<table>
<thead>
<tr>
<th>SPL</th>
<th>algorithm</th>
<th>Sampling</th>
<th>Similarity-based prior.</th>
<th>Percentage of prior. compared to sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Mobile Phone SPL</em></td>
<td>ICPL</td>
<td>175ms</td>
<td>1ms</td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td>Chvatal</td>
<td>245.1ms</td>
<td>1ms</td>
<td>0.4%</td>
</tr>
<tr>
<td></td>
<td>CASA</td>
<td>528.6ms</td>
<td>1ms</td>
<td>0.2%</td>
</tr>
<tr>
<td><em>Smart Home SPL</em></td>
<td>ICPL</td>
<td>1929.5ms</td>
<td>21.3ms</td>
<td>1.1%</td>
</tr>
<tr>
<td></td>
<td>Chvatal</td>
<td>31900.7ms</td>
<td>20ms</td>
<td>0.1%</td>
</tr>
<tr>
<td></td>
<td>CASA</td>
<td>641702.5ms</td>
<td>23.1ms</td>
<td>0.004%</td>
</tr>
</tbody>
</table>

Average execution time of the sampling algorithms and similarity-based prioritization.
Conclusion

• The rate of early defect detection of similarity-based prioritization is better than
  – Random,
  – CASA order,
  – And Chvatal order
• Better or at least equal to default order of ICPL algorithm.
• ICPL is better than
  – The default order of CASA,
  – And Chvatal algorithms.
Future Work

- Other criteria to be included in our prioritization approach (Multi-objectives).
- Other sampling algorithms such as,
  - AETG,
  - IPOG,
  - and MoSo-PoLiTe
- Use real test cases
Thank you for your attention.
References: