Search Based Software Engineering for Variability Management

Roberto E. Lopez-Herrejon
Systems Engineering and Automation Institute
Johannes Kepler University Linz, Austria
Project Context

- **FWF Lise Meitner Fellowship**
  - Named after distinguished austrian nuclear fission scientist
  - Sponsored by the Austrian Science Fund (FWF)

- **Fellowship duration: 2 years**
  - Start date: **August 2012**

- **Personnel involved**
  - Research fellow
  - Academic host: Alexander Egyed
For this talk …

- Give an overview of the project
  - Goals
  - Novel contributions

- Brief description of …
  - Early results
  - Ongoing and upcoming work
Background – Feature Models

- **Feature models**
  - de facto standard to model variability
  - denote sets of "valid" feature combinations
Background – Software Evolution

- **Reverse Engineering**
  - Process of analyzing a software system to identify its components and their relationships with the goal of creating a higher level abstraction of them

- **Software Evolution**
  - Process of progressive changes to the software artifacts or their properties
Background – Consistency Checking

- **Consistency checking**
  - Verifies that artifacts adhere to *consistency rules* that describe the semantic relationships among elements

- **Example. UML consistency rule**
  - Message action must be defined as an operation in receiver's class.
Project Big Picture

Reverse Engineering

VARIABILITY

Software Evolution

Consistency Checking

SBSE
Relation with workshop …?

Change Impact Analysis

Testing

Reverse Engineering

Software Evolution

Consistency Checking
Problems Addressed
Problem 1

Fixing inconsistencies in the presence of variability
Problem 1 – Example

How can this instance be fixed?
- Define store in CD
- Define store in Record
- Define store in TV and Mobile
- Define store in Play
- Define store in VOD

What if …
- Incorrect message name – start
- Incorrect target
- Incorrect feature ascription
- Instances overlap

...
Problem 2

- Reverse Engineering of Variability
  - Most common scenario from products variants to a SPL

Search Based Variability Mining
Problem 3

- Variability Evolution
  - Adding new features, new members of the product family, modifications

Feature Model

Search Based Feature Oriented Refactoring
Early Results

Reverse Engineering Feature Models from Product Configurations (SBSE 2012)

Collaboration with University of Seville

David Benavides, Jose Galindo, Sergio Segura, Jose Parejo
Problem Pictorial View

Feature Model

Reverse Engineering

non trivial
error prone
non unique
non optimal

Feature Sets

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>...</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>...</td>
<td>✓</td>
</tr>
</tbody>
</table>

... ... ... ... ... ...

✓ ✓ ✓ ...
ETHOM
Structural Encoding Example
ETHOM
CTC Encoding Example

![Diagram of CTC Encoding Example]

- VOD
  - 0 Play
  - 1 Record
  - 2 Display
  - 9 OS
  - 12 PPV
    - 3 TV
    - 6 Mobile
      - 4 Aerial
      - 5 Cable
    - 7 Std
    - 8 Smart
      - 10 Kernel
      - 11 Advanced

Excludes: E,4,12
Requires: R,5,12, R,8,12, R,8,11
Crossover — One point
(1) Feature Diagram

crossover point
Crossover — One point
(1) Result

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,2</td>
<td>Or,1</td>
<td>M,0</td>
<td>Or,0</td>
<td>Alt,0</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,0</td>
<td>Or,0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,0</td>
<td>Or,0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>
```

\[ \text{Smiley face} \]

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt,0</td>
<td>Alt,0</td>
<td>Alt,0</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt,0</td>
</tr>
</tbody>
</table>

\[ \text{Sad face} \]

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,2</td>
<td>Or,1</td>
<td>Op,0</td>
<td>Or,0</td>
<td>Op,3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,0</td>
<td>Or,0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Op,0</td>
<td>Or,0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>
```

\[ \text{Sad face} \]
Crossover — One point
(2) Cross-Tree Constraints
Crossover — One point
(2) Result

E,3,6  R,2,6
Experimental Setting
Evaluation

- Case studies
  - 59 feature models from SPLOT repository
  - No. products 1...896
  - No. features 9 ... 27

- Executions
  - Initial populations for each feature model were the same for the fitness functions being analysed
  - 10 runs for each feature model for each fitness function
  - 16 cores at 2.40 GHz, 25GB RAM, Cent OS, Java 1.6
Pictorial View of a Generation

individuals = feature models that denote tables of feature sets

$\text{fm}_1 \quad \text{fm}_2 \quad \text{fm}_3 \quad \ldots \quad \text{fm}_n$

- full containment
- perfect fit
- desired feature sets
Relaxed Fitness Function

- **Relaxed Fitness Function – maximized**
  \[
  \text{FFRelaxed}(\text{sfs}, \text{fm}) = | \{ \text{fs} : \text{sfs} \mid \text{validFor}(\text{fs}, \text{fm}) \} |
  \]
  
  - sfs = set of desired feature sets
  - \( \text{fm} \) = feature model to evaluate
  - \( \text{fs} \) = a feature set

- **Auxiliary function** validFor
  - checks if a feature set is valid in a FM
  - computed with FAMA using propositional logic

- **Maximizes containment of desired feature sets**
FFRelaxed Results (1)

Histogram Generations for Complete Containment

- 94.64% runs reached maximum
- 5 generations on avg. for reaching maximum

maximum not reached in 25 generations
maximum = cardinality of the desired feature sets

$\text{fm}_i$  $\text{fm}_j$  $\text{fm}_n$

desired feature sets

both reached maximum

surplus
FFRelaxed Results (3)

Surplus(sfs, fm) = \#products(fm) - |sfs| \times 100

\frac{|sfs|}{|sfs|}

- avg. 2401.24% more feature sets
- no feature model with \leq |sfs|
Ongoing work

- Analyzing other fitness functions
  - Finer comparison granularity

- Comparison with local search approaches
  - Extension to HeuristicLab platform

- Studying variability-aware chromosome operators
  - Crossover and mutation

- Extensions to feature model encodings
  - Based on genetic programming
Stay Tuned!

http://www.sea.uni-linz.ac.at/sbse4vm/
Acknowledgements

C2MV2
Consistency and Composition for Managing Variability in Multi-View Systems

Questions?