Extraction and Improvement of Conditionally Compiled Product Line Code

Bo Zhang
Software Engineering Research Group (AGSE)
University of Kaiserslautern, Kaiserslautern, Germany

Abstract
Conditional Compilation (CC) is one of the most widely used variation mechanisms in the development of software product lines (SPLs). However, a problem in SPL maintenance is that conditionally compiled code blocks are often overly scattered, nested, and tangled, which makes the code difficult to understand. Moreover, if variant code is evolved independently of the corresponding variability model, there is a risk that the two may become inconsistent. As a countermeasure, this paper proposes a maintenance process, consisting of variability model extraction and variability implementation improvement. The variability extraction is conducted automatically from preprocessor code with tool support.

1. SPL Code Maintenance

- Variability Extraction
  - Macro constants are considered as product configuration.
  - Each #ifdef block is considered as a variation point.

- Variability Improvement
  - Variability defect detection
  - Variability refactoring

2. Variability Extraction

- Extracting Product Configurations
  - Based on parsing constant definition (i.e., #defines)
  - Constant metrics
    - Cardinality: number of defined values of the constant
    - # Blocks: number of #ifdef blocks that use the constant
    - AvgLevel: avg. nesting level of relevant #ifdefs
    - AvgLength: avg. blocks length of relevant #ifdefs

<table>
<thead>
<tr>
<th>Constant</th>
<th>Cardinality</th>
<th># Blocks</th>
<th>AvgLevel</th>
<th>AvgLength</th>
</tr>
</thead>
<tbody>
<tr>
<td>portCRITIC</td>
<td>2</td>
<td>5</td>
<td>0.2</td>
<td>7.6</td>
</tr>
<tr>
<td>portPRES</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>portBYTE1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>portSTACK</td>
<td>2</td>
<td>8</td>
<td>0.75</td>
<td>12.75</td>
</tr>
<tr>
<td>portALIGN</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>portUSING</td>
<td>2</td>
<td>8</td>
<td>0.38</td>
<td>15.88</td>
</tr>
<tr>
<td>pushm_x_x</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>portPOINT</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>portBYTE2</td>
<td>4</td>
<td>7</td>
<td>0.57</td>
<td>3.14</td>
</tr>
<tr>
<td>configKER</td>
<td>3</td>
<td>7</td>
<td>0.29</td>
<td>2</td>
</tr>
<tr>
<td>portNUM_C</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

3. Variability Improvement

- Variability Defect Detection
  - Detecting obsolete variants (e.g., dead #ifdefs)

- Variability Refactoring
  - Optimizing #ifdef code to improve understandability / maintainability
  - Migrating Conditional Compilation (CC) to other variability mechanisms (e.g., Module Replacement)

4. Acknowledgement
This work is within the MOTION project of "Innovationszentrum Applied System Modeling", which is sponsored by the German state of Rhineland-Palatinate. See http://www.applied-system-modeling.de/.