Using Service Utilization Metrics to Assess and Improve Product Line Architectures

Ebru Dincel
Center of Software Engineering
University of Southern California
(in collaboration with Andre van der Hoek and Nenad Medvidovic)
Software Measurement

Assessment Techniques

- **Subjective**
  - Questionnaire based
  - Scenario based
  - Experience based

- **Objective**
  - Modeling
  - Simulation
Product Line Architectures

- Shared core
- Optionality
- Variability
- Optional variability
- Hierarchy
Service Utilization Metrics

- Structural quality assessment for a PLA
  - Absolute values for problem detection
  - Relative values for alternative assessment
- Based on service utilization of provided and required services
- Name/interface/behavior/protocol matching
Basic Building Blocks: PSU & RSU

- PSU (Provided Service Utilization)
  \[ PSU_x = \frac{P_{actual}}{P_{total}} \]

- RSU (Required Service Utilization)
  \[ RSU_x = \frac{R_{actual}}{R_{total}} \]
Example

- \( PSU_B = \frac{1}{3}, RSU_B = \frac{1}{3} \)
- \( PSU_B = 1, RSU_B = \frac{2}{3} \)

=> Component b is a better fit in architecture (b)
Compound Building Blocks:
CPSU & CRSU

\[
\text{CPSU} = \frac{\sum_{i=1}^{n} P_i^{\text{actual}}}{\sum_{i=1}^{n} P_i^{\text{total}}} \quad \quad \text{CRSU} = \frac{\sum_{i=1}^{n} R_i^{\text{actual}}}{\sum_{i=1}^{n} R_i^{\text{total}}}
\]
Example

- CPSU = 1/2, CRSU = 1/2
- CPSU = 1, CRSU = 5/6

<table>
<thead>
<tr>
<th>CPSU</th>
<th>CRSU</th>
<th>indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>↑</td>
<td>Cohesion</td>
</tr>
<tr>
<td>↑</td>
<td>↓</td>
<td>Imbalance, self insufficiency</td>
</tr>
<tr>
<td>↓</td>
<td>↑</td>
<td>Imbalance, redundancy</td>
</tr>
<tr>
<td>↓</td>
<td>↓</td>
<td>Disaster</td>
</tr>
</tbody>
</table>
Variability

- Statistical Analysis:
  - span (narrow) & average (high)

- Analysis of rest of the architecture is essential
PLA evaluation

- Case study
  - Limited core assets
  - High variation, less reusability
  - COTS dependent configurations

- Structural Assessment

- Potential Improvement
Improvements

- Redefining the component interfaces
- Splitting a component
  - *MediaUtilities*: display/process image
- Splitting PLA into two subdomains
  - image storage, image manipulation
- Results: higher service utilization metrics values, and controlled variation
Discussion

+ simple, relative, unbiased, incremental
+ Architecture Description Language (ADL) independent
+ more than assessment of structural quality
+ statistical means for PLA assessment extension

- iterative
- needs human judgment
- architectural pruning
Future Work

- Quality attributes: architectural vulnerability, schedulability, etc..
- Architectural pruning
- Collective PLA evaluation
- Case study