Architecture Model Reconstruction
Towards Change Scenario Evaluation

Heiko Koziolek, Emanuel Kolb, Jens Doppelhamer • Industrial Software Systems, ABB Corporate Research
Introduction
Software Quality Prediction

Goals for ABB

- Understand consequences that system changes have on quality attributes
  - Without costly try-and-error
  - Including existing/legacy software
- Understand trade-offs between quality attributes during evolution
Introduction: Q-ImPrESS¹
Software Quality Prediction

- Performance: Response time
- Reliability: Prob. of failure
- Maintainability: Cost

Legend:
- Tool-supported manual action
- Tool-supported automatic action
- Development iterations

Best Evolution Alternative Implementation

Applied at:
ABB ERICSSON Ericsson Nikola Tesla

¹Quality Impact Prediction for Evolving Service-oriented Software
Q-ImPrESS Meta Model – Sample Models

Static

Behaviour

Resources

Annotations

Allocation

Usage
Reverse Engineering
Architecture Model Extraction

Problem:
- Suitable abstraction of the code base
- Suited for quality predictions
- Higher level components not explicit in the code
Q-ImPrESS
Reverse Engineering – Overall Workflow

2 Structural Investigation of Software Systems
3 Generalized Abstract Syntax Tree
4 Software Model eXtractor
5 Service Architecture Model

© ABB Group
April 30, 2010 | Slide 6
Model Extraction
SoMoX Overview

- Underlying component concept:
  - Explicit interfaces
  - Composite components
- Target model defined within the Q-ImPrESS meta-model
SoMoX
Component Identification

1. Identify initial components
2. Create candidate composites
   - Classes with their required and provided interfaces
   - Based on OO metrics, compute score for candidates

[done]
SoMoX
Candidate Composites

- Identify initial components
- Create candidate composites

OO Metrics used to evaluate candidates
- Distance from the main sequence
- Coupling and
  - Name resemblance
  - Interface violation
- + X

Metrics are combined
Metrics are weighted
- System/technology specific
Metrics are computed pair wise
- Subsequent clustering
Case studies
Overview

<table>
<thead>
<tr>
<th>Java</th>
<th>LoC</th>
<th># Files</th>
<th>Total file size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoCoME</td>
<td>5k</td>
<td>125</td>
<td>600kB</td>
</tr>
<tr>
<td>Ant</td>
<td>200k</td>
<td>1690</td>
<td>10MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C++</th>
<th>LoC</th>
<th># Files</th>
<th>Total file size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial I</td>
<td>10k</td>
<td>15</td>
<td>150 KB</td>
</tr>
<tr>
<td>Industrial II</td>
<td>50k</td>
<td>58</td>
<td>750 KB</td>
</tr>
<tr>
<td>Industrial III</td>
<td>150k</td>
<td>65</td>
<td>2.4 MB</td>
</tr>
</tbody>
</table>
Java: CoCoME
Examply implementation of a SOA

Stats

<table>
<thead>
<tr>
<th>Sources</th>
<th>600kB, 5kLoC, 58 files</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAST</td>
<td>6MB (30s)</td>
</tr>
<tr>
<td>SAM Rep.</td>
<td>6kB (5s) 30PC, 18CC</td>
</tr>
</tbody>
</table>
Service Effect Specifications (SEFFs)

Component Behavior

XY.GetNewSample()

Controller. ReadData()

SharedMemory. StoreData()
Industrial Case Studies
Performance Prediction

Annotated Behavioural Description

Queueing Network Simulation

Performance Measurement

Performance Prediction (e.g. Response Time Distribution)
## Case studies
### Summing up the results

<table>
<thead>
<tr>
<th>Java</th>
<th>LoC</th>
<th>Size</th>
<th>SISSy</th>
<th>SoMox</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoCoME</td>
<td>5k</td>
<td>600KB</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Ant</td>
<td>200k</td>
<td>10MB</td>
<td>++</td>
<td>(++</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C++</th>
<th>LoC</th>
<th>Size</th>
<th>SISSy</th>
<th>SoMox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial I</td>
<td>10k</td>
<td>150kB</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Industrial II</td>
<td>50k</td>
<td>750kB</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Industrial III</td>
<td>150k</td>
<td>2,4MB</td>
<td>✖</td>
<td></td>
</tr>
</tbody>
</table>

++: very good; +: good; ●: ok; -: bad; --: very bad; ✖: No results
Q-ImPrESS
Reverse Engineering – Conclusion

- Promising approach to reverse engineering for model based software quality prediction
- Support for C++ incomplete
  - Real world dialects (MS C++) problematic
  - No precompiled header support
=> Performance issues
Power and productivity for a better world™
Q-ImPrESS
Further information

- http://www.q-impress.eu
- http://jira.ow2.org/browse/QIMPRESS
- http://sdqweb.ipd.kit.edu/wiki/SoMoX
- http://sissy.fzi.de/
- http://sdqweb.ipd.kit.edu/wiki/Palladio_Component_Model