MORPHOSIS
A Case Study on Lightweight Architecture Sustainability Analysis
Large-scale Industrial Control Systems
Challenge: Software Architecture Erosion

Circles = components, Colors = subsystems,
Size = lines of code, Arrows = dependencies
MORPHOSIS
Multi-perspective SW Architecture Analysis Method

Abstract Development Process

1. Requirements Specification
2. Architecture Design Phase
3. Software Architecture Description
4. Implementation Phase
5. Source Code

MORPHOSIS

Evolution Scenario Analysis
- Scenario evaluation report

Architecture Enforcement
- Automatically checked code rules

Architecture-level Metrics Tracking
- Periodic metric report

Iterate
- Must comply to

[...]
Evolution Scenario Analysis

Method

1. Literature search: sustainability evaluation of software architectures

Selected method: (enhanced) ALMA

2. Top Down Elicitation: 8 interviews with domain experts

3. Bottom Up Elicitation: 3-month job rotation, document analysis

List of 31 generic evolution scenarios

4. Artifact analysis: Models, code, docs; additional interviews

7 refined, prioritized evolution scenarios

7 evolution scenarios most relevant for SUS


© ABB August 24, 2012 | Slide 4
Evolution Scenario Analysis

Results

Status:
- mid 2011
- mid 2012

Year when the evolution scenario becomes relevant

Occurrence Probability

Scenario 1: 2013 (In research)
Scenario 2: 2016 (Project started)
Scenario 3: 2017 (Unlikely)
Scenario 4: 2016 (In research)
Scenario 5: 2013 (Unlikely)
Scenario 6: Discarded
Scenario 7: In research
Architectural Enforcement
Derived Module Dependency Rules from UML Tool

- Goal: Automatic checking of allowed dependencies
  - Derived dependency rules from UML layer diagrams
  - Created name mapping from modules in UML to code
  - Constructed CQL rule for each module

Enterprise Architect UML model

NDepend CQL rule
(Code Query Language)
### Architecture-level Metric Tracking

**Example: Module Interaction Stability**

- Characterizes software according to the principle of Maximization of Stand-Alone Extensibility
- Promotes the use of stable modules in lower layers

<table>
<thead>
<tr>
<th>Layer 4</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 3</td>
<td>0.5</td>
<td>0.66</td>
</tr>
<tr>
<td>Layer 2</td>
<td>0.5</td>
<td>0.6</td>
</tr>
<tr>
<td>Layer 1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Instability of a module

\[
I(m) = \frac{|\text{fanout}(m)|}{|\text{fanin}(m)| + |\text{fanout}(m)|}
\]

- Modules that depend on m

\[
SD(m) = \{m_i \in \text{fanout}(m) \mid I(m_i) > I(m) \land L(m) \geq L(m_i)\}
\]

- Set of stable dependencies to lower layers

\[
MISI(m) = \frac{|SD(m)|}{|\text{fanout}(m)|}
\]

\[
= 1 \text{ when } \text{fanout}(m) = \emptyset,
\]

\[
MISI(S) = \frac{1}{M} \sum_{i=1}^{M} MISI(m_i).
\]


© ABB August 24, 2012 | Slide 8
MORPHOSIS
Lessons learned

- Perceived good cost / benefit ratio
  - Low analysis overhead, high automation
  - However: benefits are not quantified yet
- Quantifying the costs for evolution scenarios is hard
  - Impact prediction difficult if code not available
- Externalizing and prioritizing evolution scenarios provides focus to plan mitigation measures
- Architecture enforcement raises developer awareness
  - Higher regard for the architecture description
- High developer interest in metrics
  - Desire to improve quality, less concerned about being judged
MORPHOSIS

Conclusions

- Evolution Scenario Analysis
  - Provided detailed description template
- Architecture Enforcement
  - Integrated rules from UML model into build process
- Architecture-level Metrics Tracking
  - Automated tracking of novel architecture metrics

- Future Work
  - Re-evaluate / add evolution scenarios
  - Conduct a longitudinal study correlating metrics to actual maintenance costs
Power and productivity
for a better world™
Architecture Enforcement
Tool Support: NDepend (C#), CppDepend (C++)

Custom queries on the source code with SQL-like language

Metric visualization (box = class, size = LOC)

Code Metrics

Code & design rules

Query results

Classes violating rules

© ABB August 24, 2012 | Slide 12
MORPHOSIS Discussion

- Lightweight method?
  - Scenario analysis w/o workshops
  - Automated metrics reporting
- In case of conflicting stakeholders: better run ATAM
- Substantial architecture redesign unlikely
  - Method aims at selective improvements
- Future Work
  - Re-evaluate evolution scenarios
  - Conduct a longitudinal study correlating metrics to actual maintenance costs