



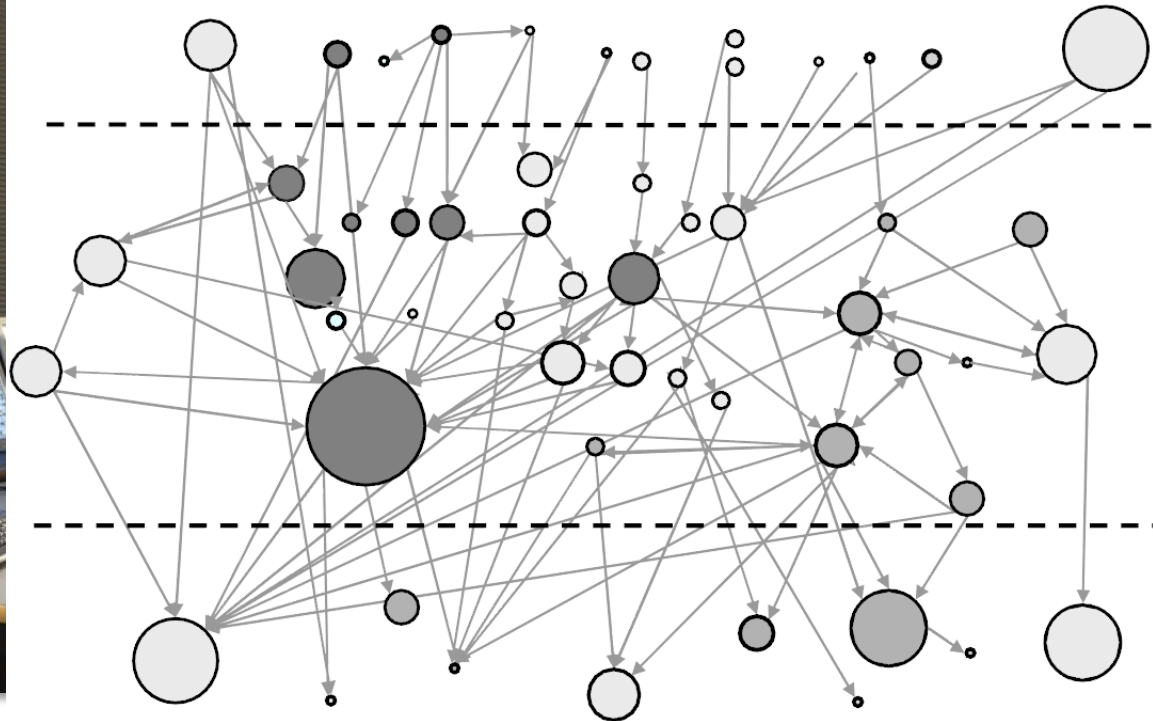
Heiko Koziolk, Dominik Domis, Thomas Goldschmidt, Philipp Vorst, Roland Weiss
ABB Corporate Research, Ladenburg, Germany

MORPHOSIS

A Case Study on Lightweight Architecture Sustainability Analysis

Large-scale Industrial Control Systems

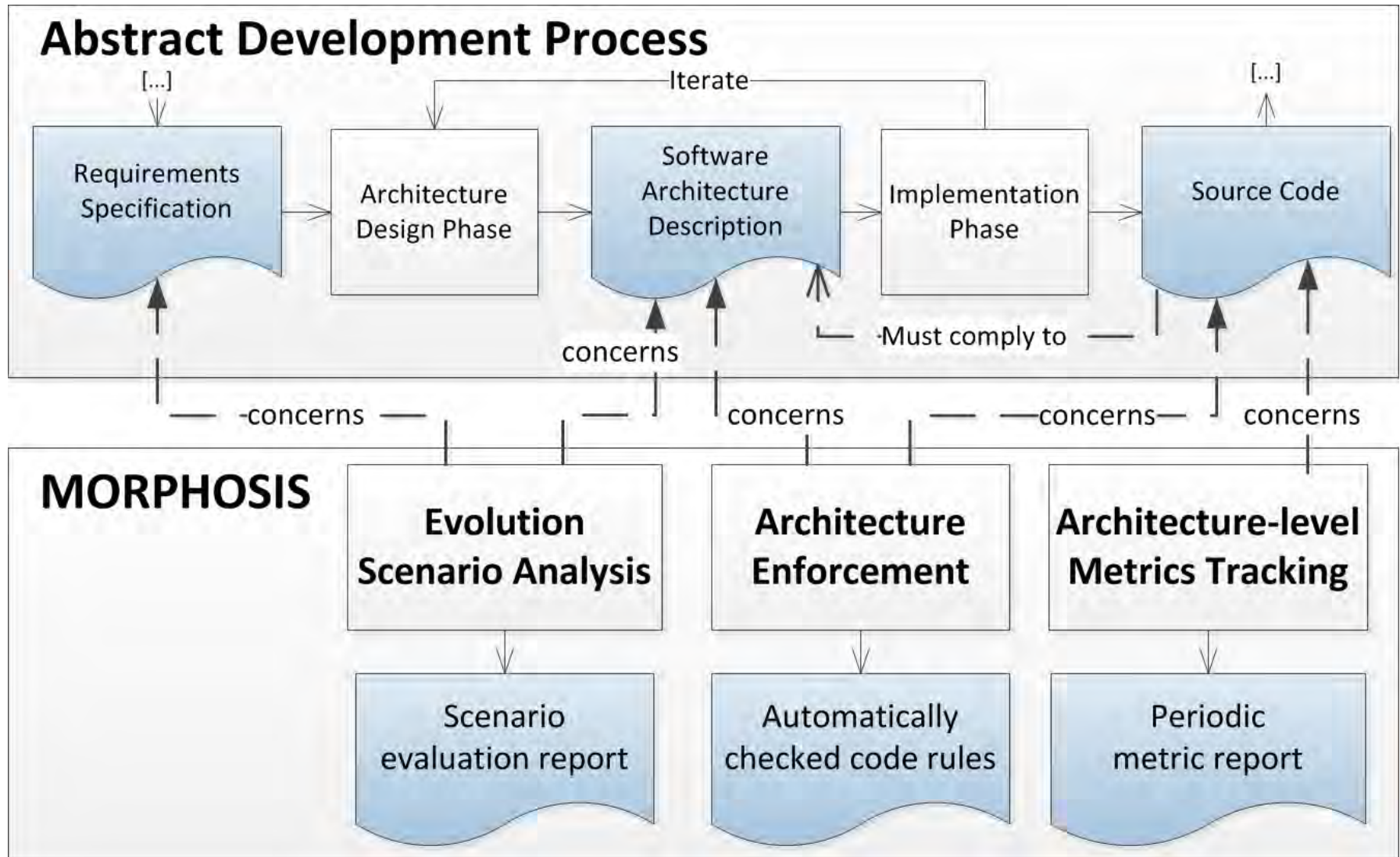
Challenge: Software Architecture Erosion



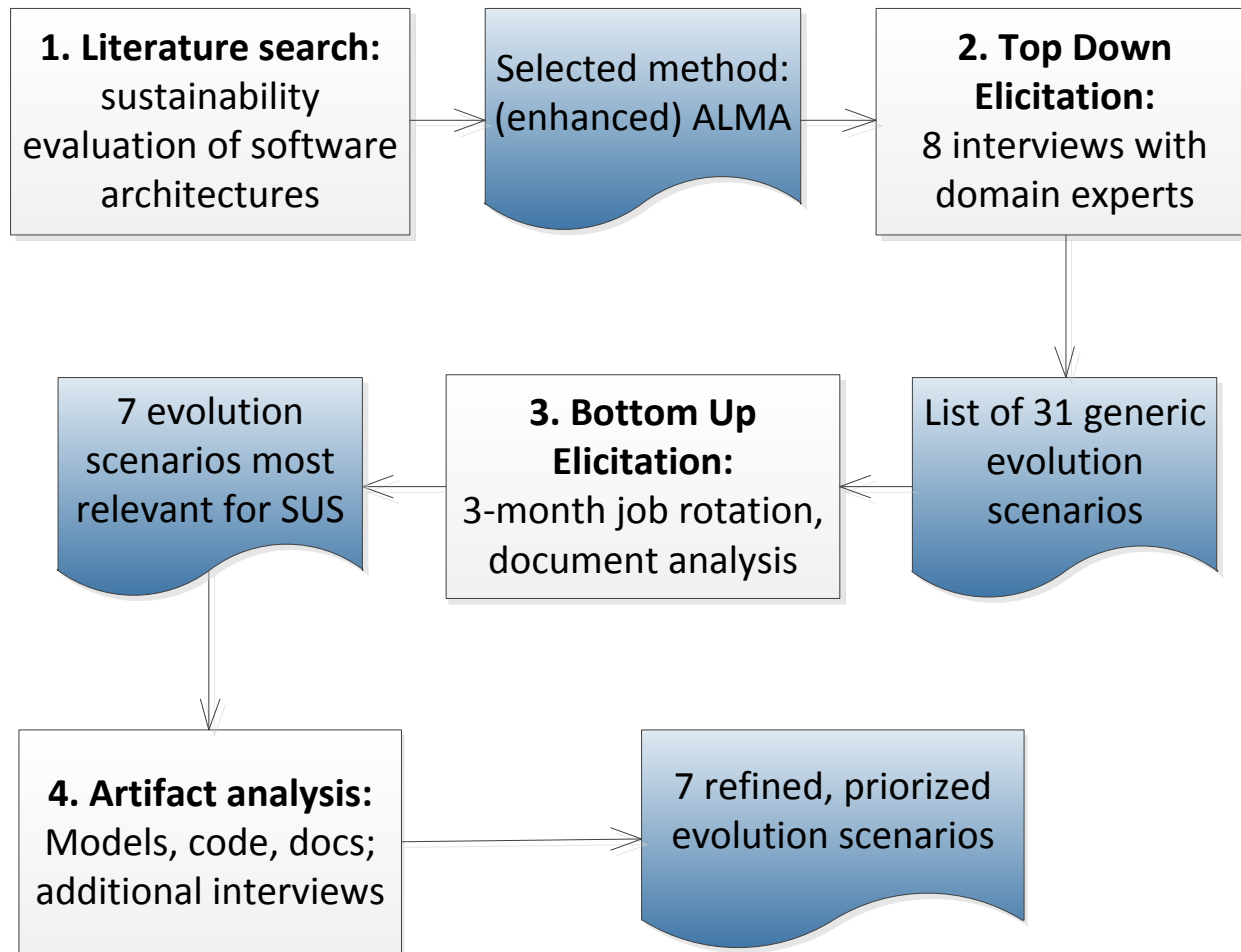
Circles = components, Colors = subsystems,
Size = lines of code, Arrows = dependencies

MORPHOSIS

Multi-perspective SW Architecture Analysis Method



Evolution Scenario Analysis Method



P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures: methods and case studies*. Addison-Wesley, Reading, MA, 2002.

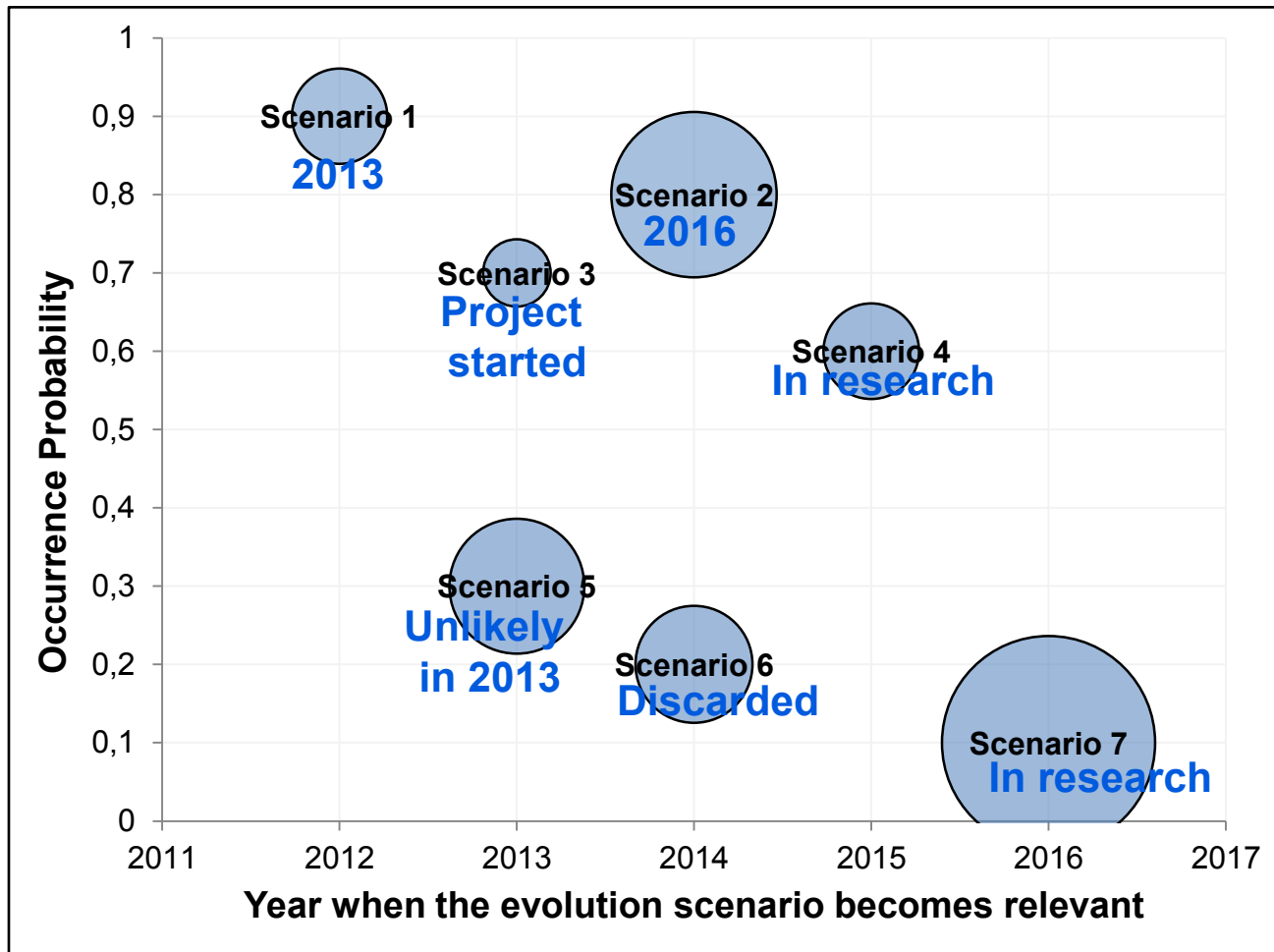
H. Koziolok *Sustainability evaluation of software architectures: A systematic review*. Proc. QoSA'11, p. 3-12, ACM, June 2011.

P. Bengtsson, N. Lassing, J. Bosch, and H. van Vliet, *Architecture-level modifiability analysis (ALMA)* Journal of Systems and Software, vol. 69, no. 1-2, pp. 129-147, 2004

Evolution Scenario Analysis Results

Status:

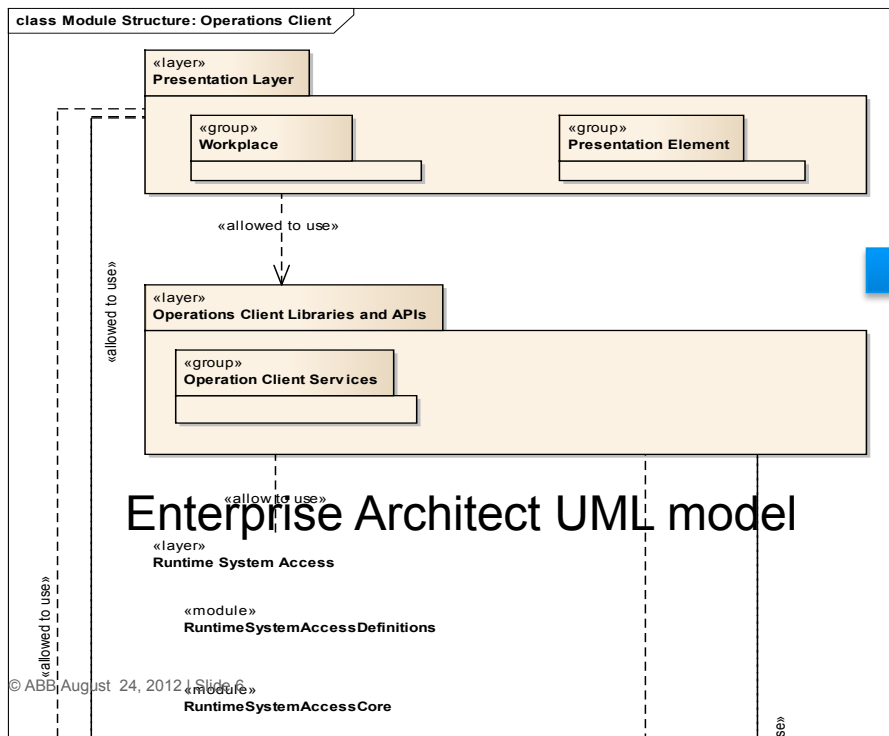
- mid 2011
- mid 2012



Architectural Enforcement

Derived Module Dependency Rules from UML Tool

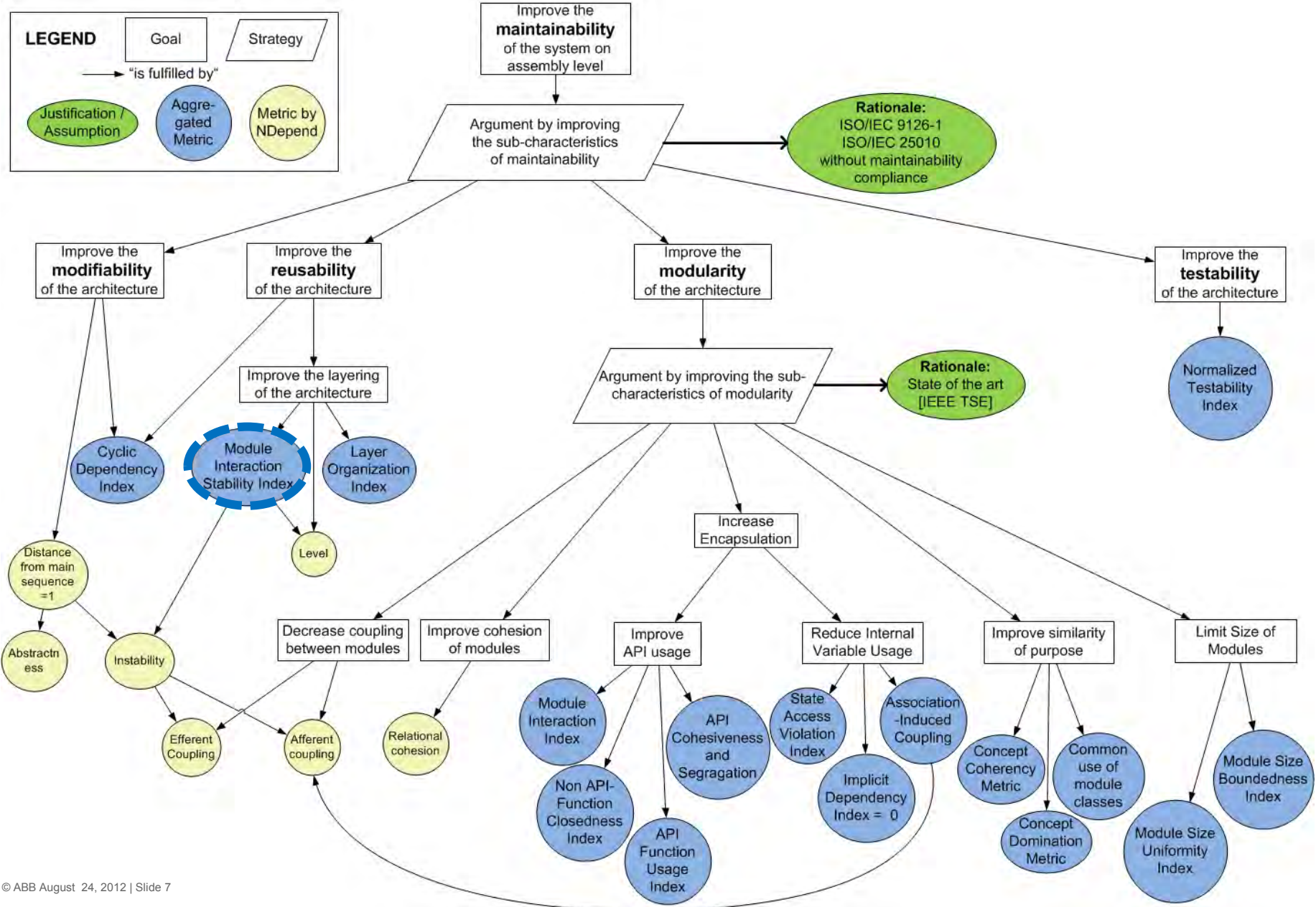
- Goal: Automatic checking of allowed dependencies
 - Derived dependency rules from UML layer diagrams
 - Created name mapping from modules in UML to code
 - Constructed CQL rule for each module



```
// «Name» Presentation Layer Dependencies «/Name»
WARN IF Count > 0 IN
SELECT ASSEMBLIES WHERE IsDirectlyUsedBy "ASSEMBLY::OperationsClient.Workplace"
AND !{
  {
    NameIs OperationsClient.Workplace" OR
    NameIs OperationsClient.WorkplaceApplication" OR
    NameIs OperationsClient.WorkplaceCore" OR
    NameIs OperationsClient.WorkplaceOverlap" OR
    NameIs OperationsClient.WorkplacePresentation" OR
    NameIs OperationsClient.WorkplaceScreen" OR
    NameIs OperationsClient.WorkplaceTools"
  }
  OR
  {
    NameIs OperationsClient.GraphicsViewer" OR
    NameIs OperationsClient.ProcessGraphics.SystemAccessAdapter" OR
    NameIs "GraphicsKernel" OR
    NameIs ProcessGraphics.SystemAccess" OR
    NameIs ProcessGraphics.Types" OR
    NameIs SystemAccess" OR
  }
}
//other elements
```

NDepend CQL rule
(Code Query Language)

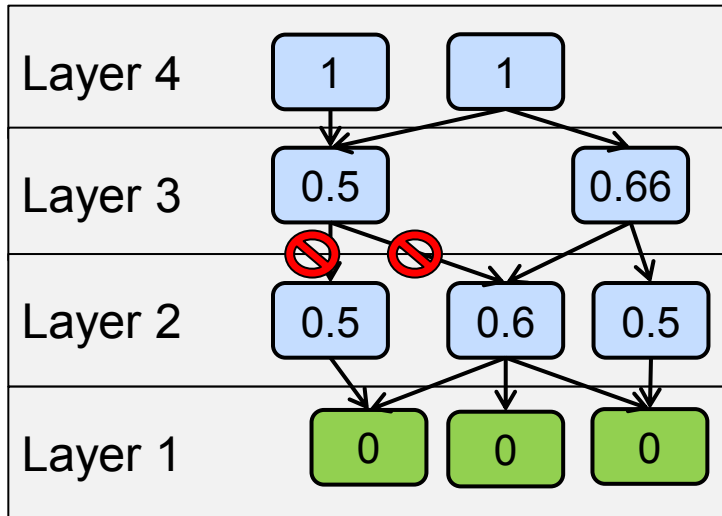
Architecture-level Metric Tracking



Architecture-level Metric Tracking

Example: Module Interaction Stability

S. Sarkar, G. M. Rama, and A. C. Kak,
 "API-based and information-theoretic
 metrics for measuring the quality of
 software modularization,"
 IEEE Trans. Softw. Eng., vol. 33,
 pp. 14–32, January 2007.



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

Instability of a module

$$\mathcal{I}(m) = \frac{|fanout(m)|}{|fanin(m)| + |fanout(m)|}$$

Modules m depends on

Modules that depend on m

$$SD(m) = \{m_i \in fanout(m) \mid \mathcal{I}(m) > \mathcal{I}(m_i) \ \& \ \mathcal{L}(m) \geq \mathcal{L}(m_i)\}$$

Set of stable dependencies to lower layers

$$MISI(m) = \frac{|SD(m)|}{|fanout(m)|}$$

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

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

For all modules

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

The screenshot displays the Visual NDepend application interface. At the top left, a 'Design' window shows a custom SQL-like query: `WARN IF Count > 0 IN SELECT NAMESPACES WHERE NbTypes < 5 ORDER BY NbTypes ASC`. Below this, a table lists '50 namespaces matched' with columns for namespace names and the number of types. The central area features a 'Metrics' visualization of a dependency graph where boxes represent classes and their size is proportional to LOC. A 'Code Metrics' panel on the right shows statistics: # IL Instructions: 116 501, # lines of code (LOC): 13 884, # lines of comment: 31 646, Percentage Comment: 63%, # Assemblies: 24, # Namespaces: 109, # Types: 1 594, # Methods: 8 089. At the bottom, a 'CQL Query Explorer' window lists various rules such as 'Code Quality', 'Design', 'Code Diff', 'Code Coverage', and 'Dead Code'. A table below this lists specific rule violations with columns for 'Active', '#Items', and the rule description, such as 'Assembly should not contain namespaces dependency cycles' (6 items) and 'Type should not have too many responsibilities (Efferent Coupling)' (45 items).

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

