EmpAnADa Project

Christian Lange

C.F.J.Lange@tue.nl

June 4th, 2004

Eindhoven University of Technology,
The Netherlands

Outline

• EmpAnADa introduction
• Part I
  – Completeness and consistency in detail
• Part II
  – Background
    • UML
    • Empirical research
    • Measurement
  – Directions of the project
    • Metrics
    • Refactoring and changes in an artifact
    • Completeness and Consistency
    • Visualization of Metrics
  – EmpAnADa and LaQuSo
EmpAnADa

- Empirical Analysis of Architecture and Design Quality
- Development of quantitative methods to predict attributes of early artifacts, mainly UML models
  - Quality (maintainability, changeability…)
  - Effort estimation
  - Identification of fault-prone elements
- Empirical studies to
  - Validate methods
  - Investigate UML usage and problems in practice

Part I
Empirical Assessment of Completeness in Industrial UML Designs
Overview of Part I

- Completeness
  - Decomposition
  - Consistency
- Techniques Developed
- Case Studies
- Observations
- Conclusions

Why Completeness?

- Interviews and surveys investigating industrial use of UML
  - Uncertainty about precision and accuracy of design metrics for prediction of quality attributes
  - Miscommunication
  - Integration problems
  - Uncertainty about completeness of model
Completeness & Consistency

Problem:
- (Early) models allow several solutions/interpretations (dark grey)
  - Source code describes exactly one solution
- Misinterpretations amplified by (light grey)
  - lack of completeness
  - presence of conflicts
  - UML specific: overlapping diagram types, no formal semantics, degrees of freedom

Goal:
Investigate the magnitude of completeness of UML models in industrial practice

Decomposition of Completeness

Completeness
- Well-Formedness:
  - property of individual diagram
- Consistency and Diagram Completeness:
  - relation between different diagrams
Requirements can be traced across multiple diagram types

- Examples of incompleteness
- Scope of analysis

Relational Meta Model

- Use Case Diagram
  - Use Cases
  - Actors
- Message Sequence Chart
  - Messages
  - Scenarios
  - States
  - State Machines
- Class Diagram
  - Objects
  - Classes
  - Methods
  - Attributes
Examples of Rules

- **Well-formedness**
  - Objects in SD must have a role name
  - Classes must have methods
- **Consistency**
  - Messages in SD must correspond to method in class diagram
- **Completeness**
  - Interaction of classes must be described in SD
  - Methods must be called in SD

Case Studies

- **A1, A2**
  - Image processing
  - Subsystems, same version
  - 168 classes, 853 messages
  - >100 man years
  - Real-world

- **B1, B2**
  - Embedded Controller
  - Subsequent versions, redesign
  - 69 classes, 705 messages
  - >100 man years
  - Real-world

- **C1, C2**
  - Embedded Controller
  - Subsequent version, inspection in between
  - 46 classes, 219 messages
  - 1.5 man years
  - Post-grad students
Results: Well-formedness Rules

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects without name</td>
<td>52.00%</td>
<td>61.58%</td>
<td>91.92%</td>
<td>25.24%</td>
</tr>
<tr>
<td>Classes without methods</td>
<td>60.19%</td>
<td>51.19%</td>
<td>100.00%</td>
<td>20.00%</td>
</tr>
<tr>
<td>Interfaces without methods</td>
<td>8.82%</td>
<td>9.38%</td>
<td>N/A</td>
<td>60.00%</td>
</tr>
<tr>
<td>Abstract classes in MSC</td>
<td>5.56%</td>
<td>0.60%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Public Attributes</td>
<td>67.23%</td>
<td>5.08%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Results: Consistency Rules

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messages Without Name</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.28%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Messages without Method</td>
<td>58.73%</td>
<td>7.62%</td>
<td>100.00%</td>
<td>27.14%</td>
</tr>
<tr>
<td>Messages between unrelated Classes</td>
<td>71.94%</td>
<td>41.03%</td>
<td>77.73%</td>
<td>81.90%</td>
</tr>
</tbody>
</table>
Results: Completeness Rule

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes not called in MSC</td>
<td>61.11%</td>
<td>59.52%</td>
<td>35.29%</td>
<td>42.22%</td>
</tr>
<tr>
<td>Interfaces not called in MSC</td>
<td>100.00%</td>
<td>87.50%</td>
<td>100.00%</td>
<td>70.00%</td>
</tr>
<tr>
<td>Methods not called in MSC</td>
<td>67.65%</td>
<td>77.59%</td>
<td>N/A</td>
<td>40.14%</td>
</tr>
</tbody>
</table>

Observations in Case Studies

- Strong differences in the habits of designers
  - even within a single project
  - miscommunication may be reduced by modeling standards
- Degradation between C1 and C2 wrt. SD completeness: rules helped identify missing SD’s
- Odd results for SD-rules in B2: identified that wrong version of model was in project repository.
- Case C has best scores: off critical-path project (?)
Contributions & Conclusions

- Industrial-strength techniques for quantitative assessment of incompletenesses and inconsistencies.
- Case studies provided insight into use of the UML
  - Absolute number of rule violations quite large
  - Industrial projects move into implementation while there are still many incompletenesses
- Incomplete spots identified by tool were confirmed by designers and led to improvements in the design

Future Work

- Which degree of completeness is needed?
  - for high accuracy and precision of estimates
  - to reduce miscommunication
- Do incompleteness and inconsistencies of the model result in flaws in the implementation?
- Can “Modeling standards” (enforced by tooling) improve completeness?
Part II
Background Topics
&
Future Directions

Background - UML

- Unified Modeling Language
  - Booch, Rumbaugh, Jacobson : 1997
- Purpose: visual modelling of different aspects of the system
  - Higher abstraction level than programming
- Diagrams are views on the model
- 9 diagram types

**Structural**
- Class diagram
- Package diagram
- Object diagram
- Use case diagram
- Deployment diagram

**Behavioral**
- Sequence diagram
- Collaboration diagram
- State diagram
- Activity diagram

- Extensible by stereotypes, tagged values, meta-model changes
- Lack of formal semantics
Background – UML

**Class Diagram**

**Sequence Diagram**

**State Diagram**

**Use Case Diagram**

*Background – Empirical Research*

- SW engineering is cross-disciplinary
  - Technical issues
    - Networks, databases, operating systems…
  - Language issues
    - Syntax, semantics
  - Social issues and psychology
- **Why empirical SW engineering research?**
  - Quantitative studies give objective and statistically significant results regarding
    - Understanding
    - Controlling
    - Predicting
    - Improving
  - Of the SW Development process
- **Empirical strategies**
  - Surveys
  - Experiments
  - Case Studies
Background – Empirical Research Strategies

<table>
<thead>
<tr>
<th>Factor</th>
<th>Survey</th>
<th>Experiment</th>
<th>Case study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution control</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Measurement control</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Investigation cost</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Ease of replication</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

Empirical Research in EmpAnADa

- Survey
  - Web-questionnaire about UML usage
- Experiment
  - Classroom experiment is planned
- Case Studies
  - Always looking for case studies
  - Currently contacts to
    - Philips (PMS, Optical Storage)
    - Siemens Corporate Research
    - LogicaCMG / Thales
    - Oce
    - CGE&Y
    - CBS
Background - Measurement

- "Measurement is a mapping from the empirical world to the formal, relational world" (Fenton)
- Types of metrics
  - Product
    - Lines of Code (LOC), Depth of Inheritance, number of defects...
  - Process
    - Cost per number of errors found, coverage of a test...
  - Resource
    - Team size, experience of staff, age of staff...
- Most product metrics research is about finished products

Measurement in EmpAnADa

- Metrics are objective and automatically collectable
- Implementation metrics
  - Measure the exact product
  - Rather late
- Design Metrics
  - Early feedback in the analysis and design phase
  - Not all implementation metrics can be used
  - Opportunity for new metrics (multi-view)
Measurement in EmpAnADa II

- Relation between design metrics and properties of product
  - Quality properties
    - Maintainability
    - Comprehensibility
    - Changeability
    - ...
  - Defects
    - Finding relationships between metrics and test data, bugreports
    - Metrics to predict fault-prone classes
  - Prediction of process data
    - Development effort
    - Resource usage
- Design metrics as predictions: Accuracy and Precision
  - Depend on
    - Correspondence of model and source code
    - Completeness and Consistency of model

Completeness and Consistency

- Define completeness and consistency for UML models
- Assess the magnitude of violations in industrial practice
- What are the effects of C&C violations for the implementation?
- Which degree of C&C is needed for precise and accurate predictions based on UML models?
Refactoring and Evolution

- Refactoring:
  - "Changing a SW system to improve its structure without altering the external behavior" (Fowler)
  - Most literature considers source code refactorings
  - Approaches:
    - Defining higher level refactorings (architecture, design)
    - Investigating the role of metrics as "design smells"

- Evolution
  - How does the quality of designs change over subsequent versions?

Visualization of Metrics and Artifacts

- For large systems metrics tools produce large amount of data
- Which techniques are useful to present the measurement data in an intuitive way?

- normal perspective
  - original
  - Enriched with colors representing metrics

- reduced perspective
  - Showing only inheritance
  - Showing only dependencies
  - Showing only associations

- extended perspective
  - Showing only classes with metric larger than specific value
  - Class diagram extended with related use cases
Summary of topics

Activity within LaQuSo

- Architecture Repository Group
  - Developing a repository to store SW architectures and designs in different formalisms
Discussion

?