

EmpAnADa

Empirical Analysis of
Architecture and Design Quality
<http://www.win.tue.nl/~clange/empanada/>

BENEVOL 2004

Ir. Christian Lange
Dr. Michel Chaudron

Technische Universiteit Eindhoven
System Architecture and Networking Group

Project Outline

- EmpAnADa: Empirical Analysis of Architecture and Design Qualities
- Focus on UML architecture and design models
- Developing quantitative techniques for early assessment
- Empirical Validation
 - Industrial case studies
 - Controlled experiments
- Gaining insight into the use (and problems) of the UML
 - Interviews
 - surveys

Goal of our Research

Develop practical techniques
that help improve UML designs

- What to improve? How to identify flaws in (parts of) design?
 - General design heuristics (smells)
 - Coding conventions
 - Benchmarks
- Which properties of an implementation we predict based on properties of the UML design?
 - high coupling *leads to* large test & maintenance effort,
 - deep inheritance *leads to* error proneness
 - ...

Side-effect: effort estimation & progress monitoring

Why Completeness?

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

Problem with Ambiguity of UML

Early models allow several solutions/interpretations

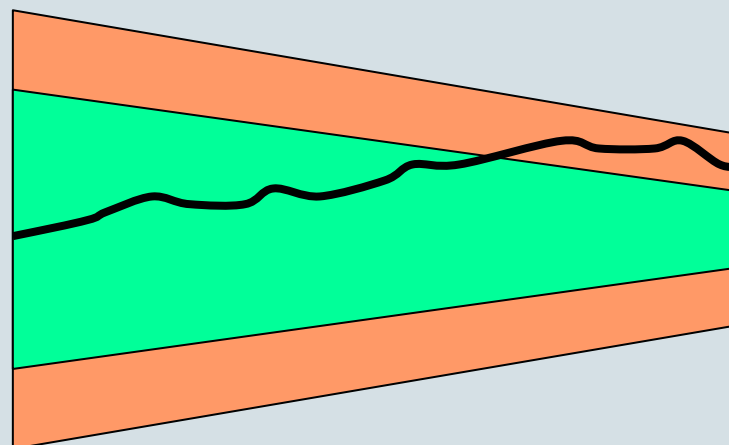
-Source code describes exactly one solution

Misinterpretations amplified by

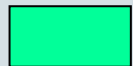
-lack of completeness

-presence of conflicts

-UML specific: overlapping diagram types, no formal semantics, degrees of freedom



Intended meaning
of design



Misinterpretations
of design



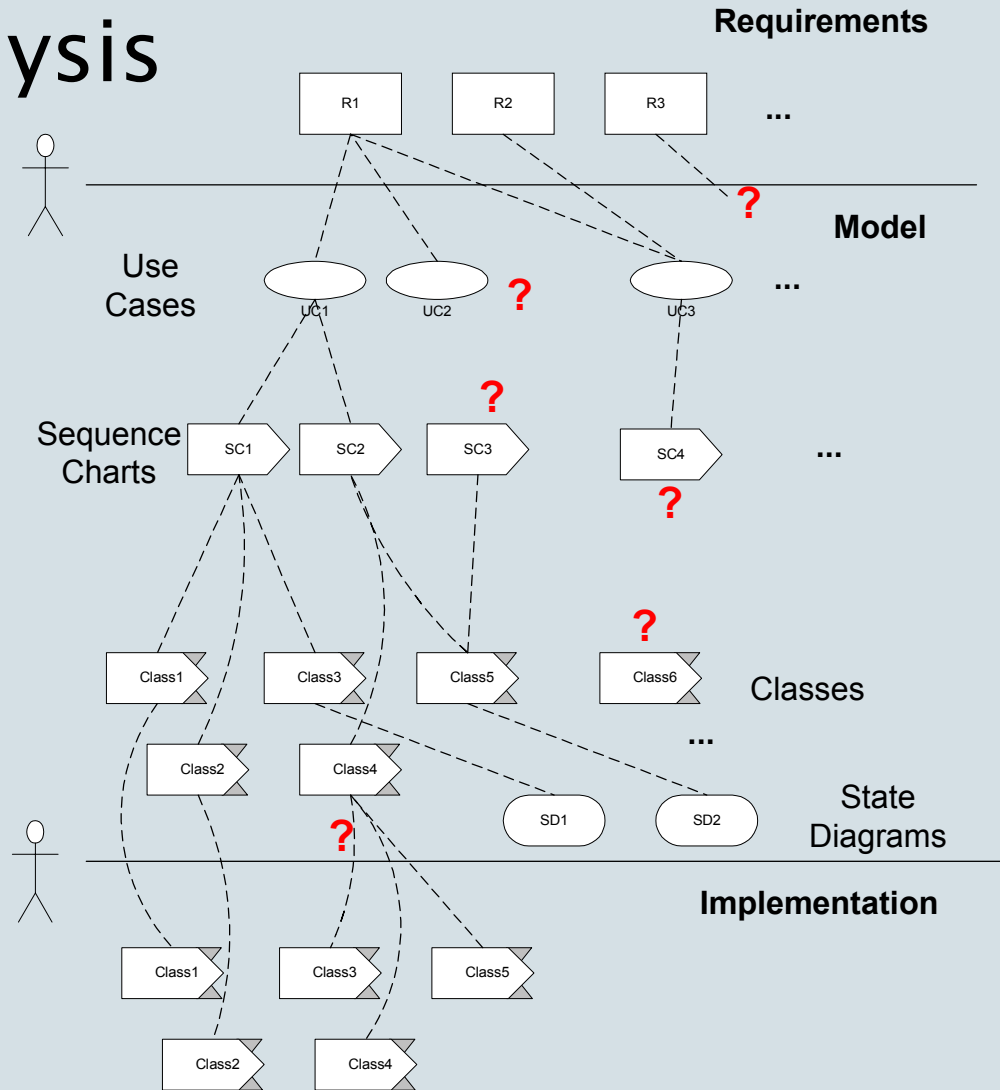
Development
of design



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

Completeness Analysis

- Requirements can be traced across multiple diagram types
- Examples of incompleteness
- Scope of analysis

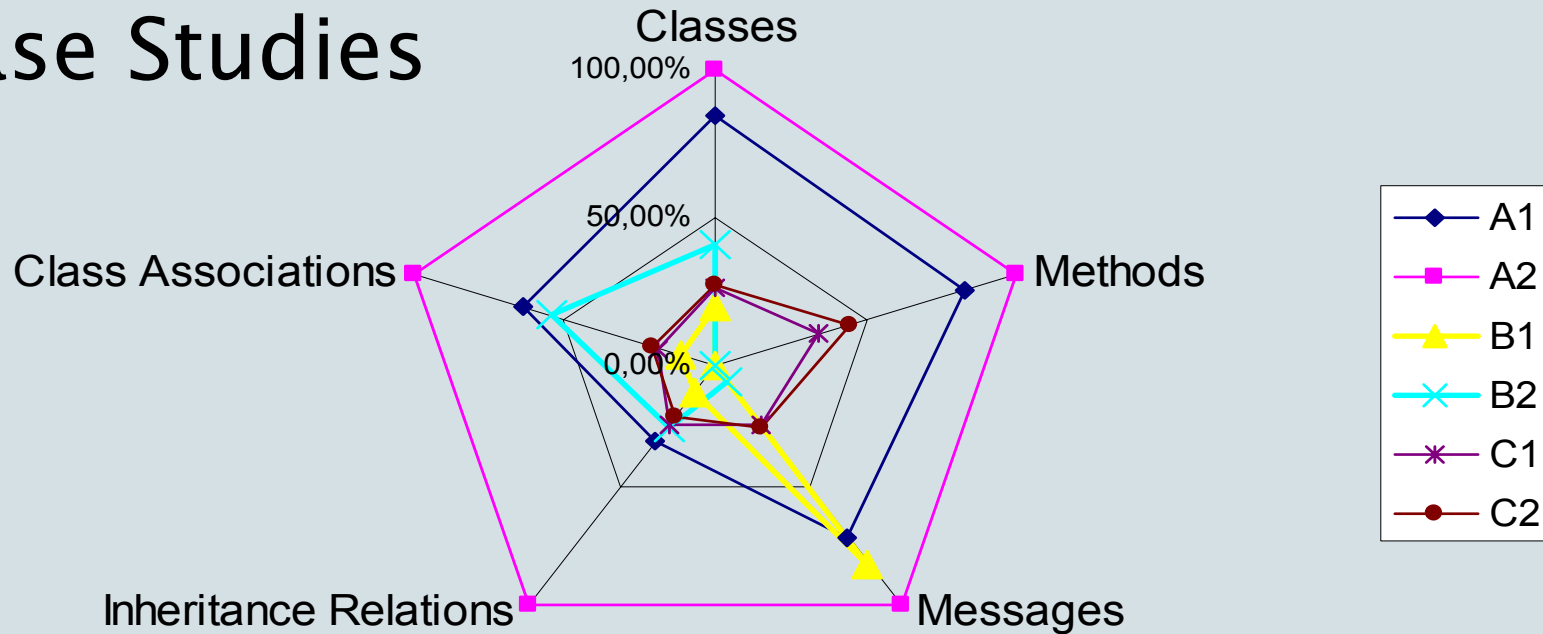


Examples of Rules

- 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

Rule-checking is implemented in a tool

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

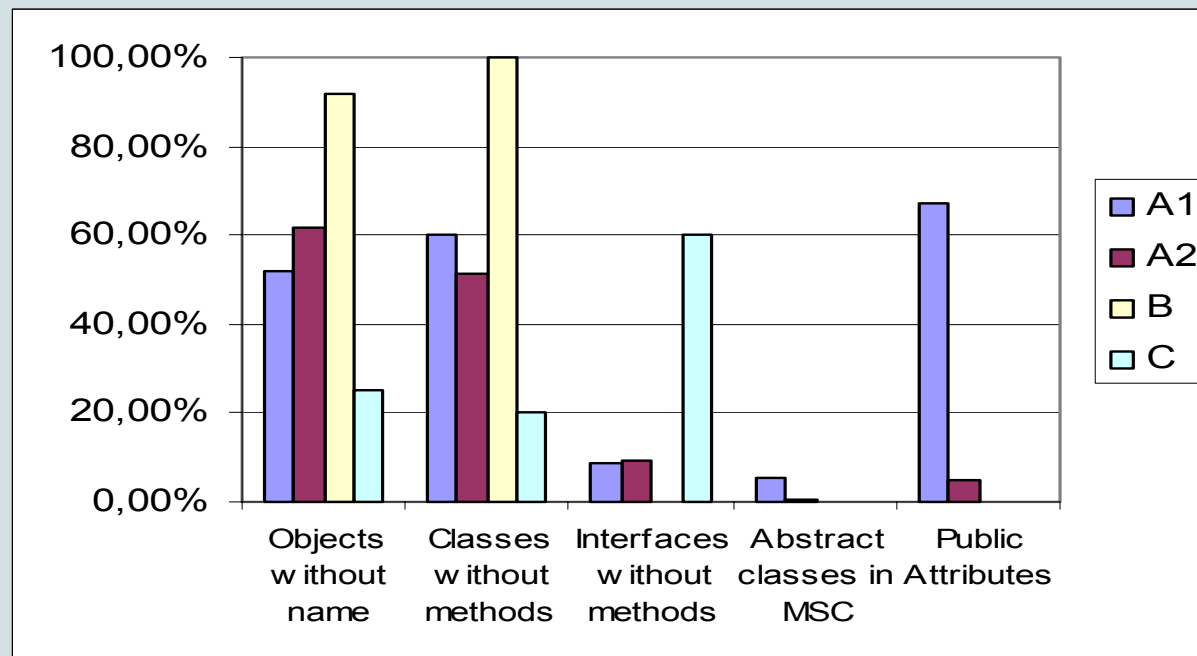
- Embedded Controller
- Subsequent version, inspection in between

• 46 classes, 219 messages

- 1,5 man years
- Post-grad students

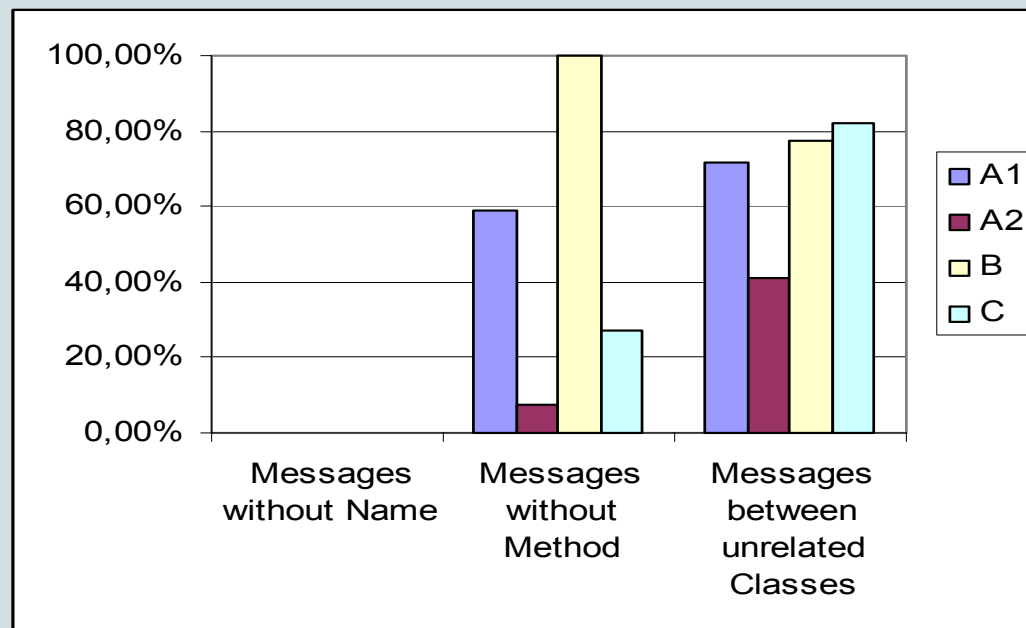
Results: Well-formedness Rules

	<i>A1</i>	<i>A2</i>	<i>B</i>	<i>C</i>
Objects without name	52.00%	61,58%	91.92%	25.24%
Classes without methods	60.19%	51.19%	100.00%	20.00%
Interfaces without methods	8.82%	9.38%	N/A	60.00%
Abstract classes in MSC	5.56%	0.60%	0.00%	0.00%
Public Attributes	67.23%	5.08%	0.00%	0.00%



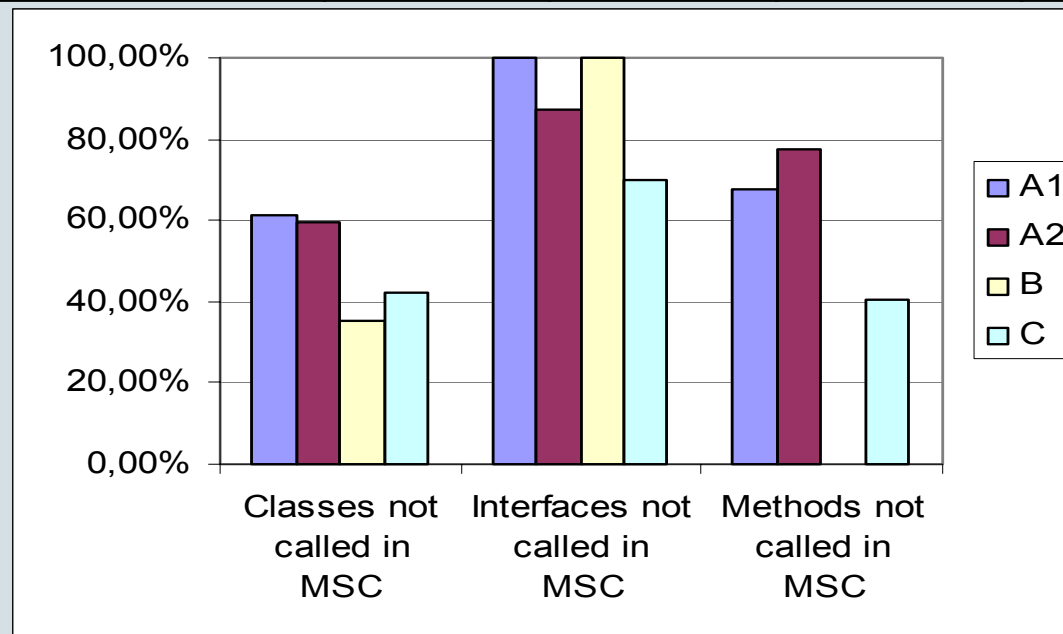
Results: Consistency Rules

	<i>A1</i>	<i>A2</i>	<i>B</i>	<i>C</i>
Messages Without Name	0.00%	0.00%	0.28%	0.00%
Messages without Method	58.73%	7.62%	100.00%	27.14%
Messages between unrelated Classes	71.94%	41.03%	77.73%	81.90%



Results: Completeness Rule

	<i>A1</i>	<i>A2</i>	<i>B</i>	<i>C</i>
Classes not called in MSC	61.11%	59.52%	35.29%	42.22%
Interfaces not called in MSC	100.00%	87.50%	100.00%	70.00%
Methods not called in MSC	67.65%	77.59%	N/A	40.14%



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 (?)

- Results of assessments can be used for
 - focusing of improvements
 - monitoring progress
 - benchmarking: comparison between different models
(versions, sub-models, processes)