Assessing Correspondence between Design and Implementation

Dennis van Opzeeland

d.j.a.v.opzeeland@student.tue.nl

June 21, 2005

Eindhoven University of Technology,
The Netherlands

Outline

- Introduction
- What is correspondence?
- Matching of implementation pieces to design elements
- Highlighting differences
- Case study
- Conclusion

Introduction

- Correspondence:
  - Similarity between design and implementation
- Correspondence vs. evolution
  - Correspondence degrades if implementation evolves but design doesn’t
  - Correspondence ↓

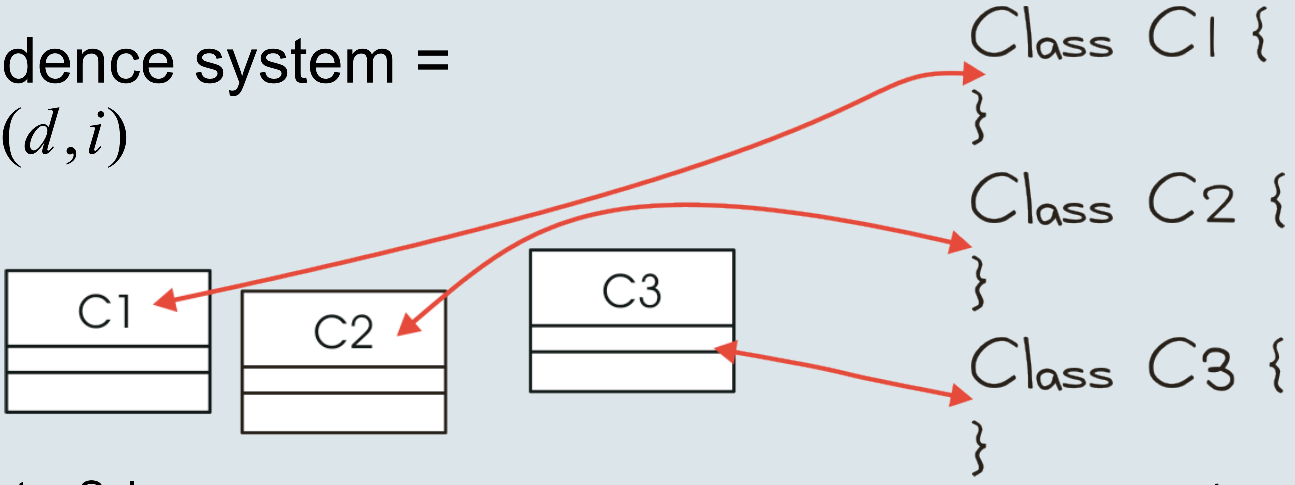
    ⇒ Maintainability ↓

    ⇒ Evolution effort ↑

What is correspondence?

- Expressed in terms of the model elements
  - Design: classes, interfaces, ...
  - Implementation: class declaration, interface specification,...
- Mapping between design elements and implementation elements
- Correspondence system =

$$\sum_{d \in D, i \in I | \mathrm{eq}(d,i)} \mathrm{sim}(d,i)$$



Typical deviations from design

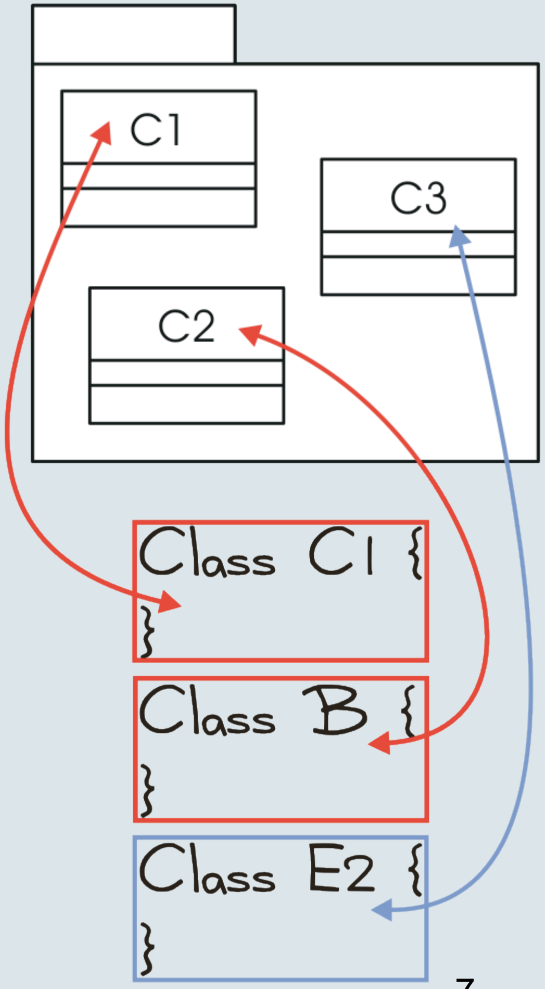
- Structural
  - Easy to check
  - Examples
    - Introduction of new classifiers
    - Differences in names
    - Introduction of new operations and attributes
    - Introduction of dependencies and associations
- Behavioral
  - Hard to check
  - Examples
    - Incompatible message sequences
- Not all deviations are equally problematic

Finding the matching

- Given:
  - Set of design classifiers
  - Set of implementation classifiers
- Problem:
  - Find the design pieces and implementation pieces that were meant to be “the same”
- Different approaches
  - Classifier names
  - Structural properties
  - Package information
  - Metric profile

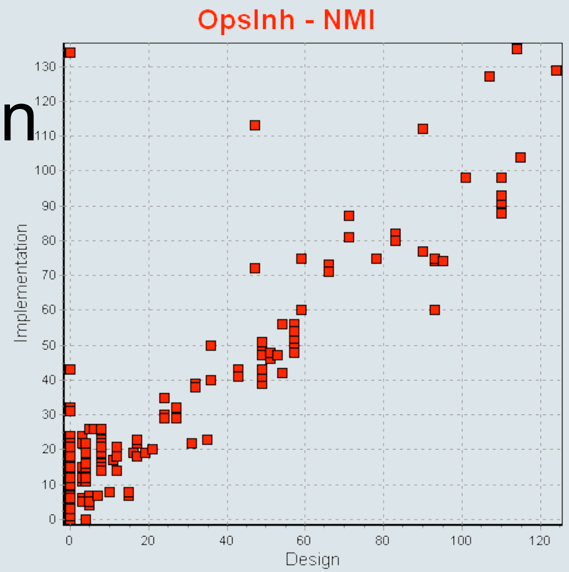
Using package information

- Heuristic:
  - Existing relations between two packages predict other relations
- Requirements
  - Development view in design
  - Directory layout for source code
  - Partial matching exists
- Purpose
  - Limit search space of other methods



Matching with Metric profiles (1)

- There exist correlations between design metrics and implementation metrics of a system



- Correlating metrics define *metric profile* of a class
  - Let *c* be a class, then $m(c)=(m_{1,c}, \dots, m_{n,c})$
  - Pairwise correlations between metrics in design profile and implementation profile

Matching with Metric Profiles (2)

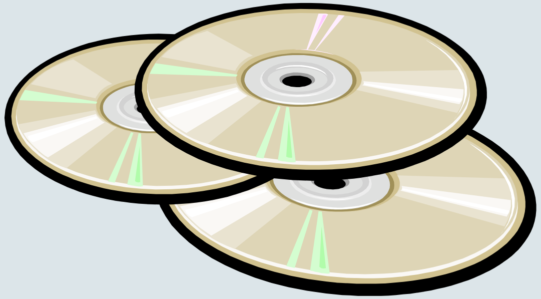
- Let *d* be a design class and *i* an implementation class
- Given metric value for design predict value for implementation metric and compare with real value

$$\mathrm{sim}(d,i) = \sum_n \rho_n \,|\, \beta_{0,n} + \mathrm{m}(d)_n \beta_{1,n} - \mathrm{m}(i)_n \,|$$

- The implementation class that fits best matches to the design class

Case study

- Characteristics
  - Industrial case
    - Firmware for DVD recorder
  - Design
    - UML 1.4
    - 346 classes
  - Implementation
    - C++
    - 777 classes
    - Lines of Code: 2,558,216
- Approach:
  - Initial matching based on names
  - Empirical analysis for metric profile approach



Correlating metrics

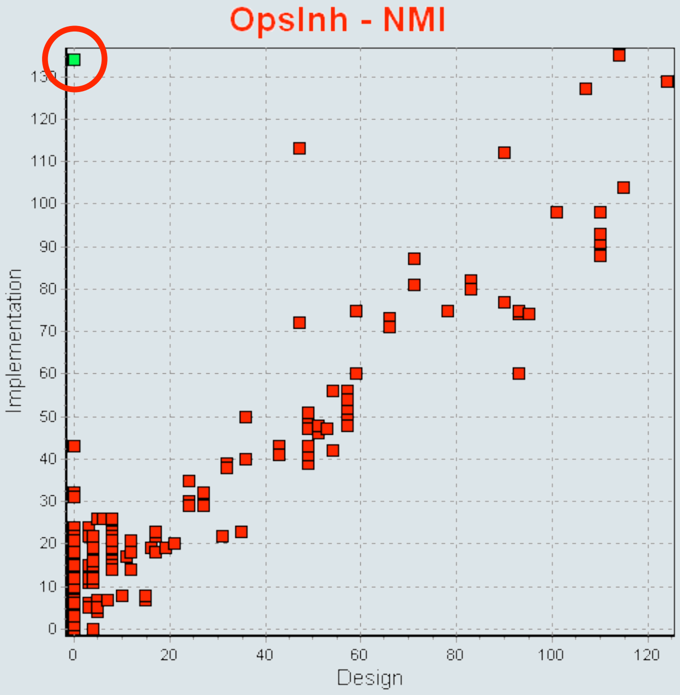
Design	Implementation	Corr. Coefficient
# Ops. inherited	# Ops. inherited	0.924
Depth of inh. tree	Depth of Inh. tree	0.883
Coupl. objects	Data abstr. coupl.	0.816
# Ops. inherited	# Protected ops.	0.889
# Ops. inherited	Depth of inh. tree	0.829
# Priv. operations	# Priv. operations	0.223
# Attributes	# Attributes	0.184

For all correlation coefficient measures, the significance level $p < 0.01$

Case study results

- Classification of deviations from design found
  - Introduction of (private/protected) attributes and operations
  - Introduction of new classes (decomposition of design classes)
  - Unused dependencies
  - Changes in inheritance tree

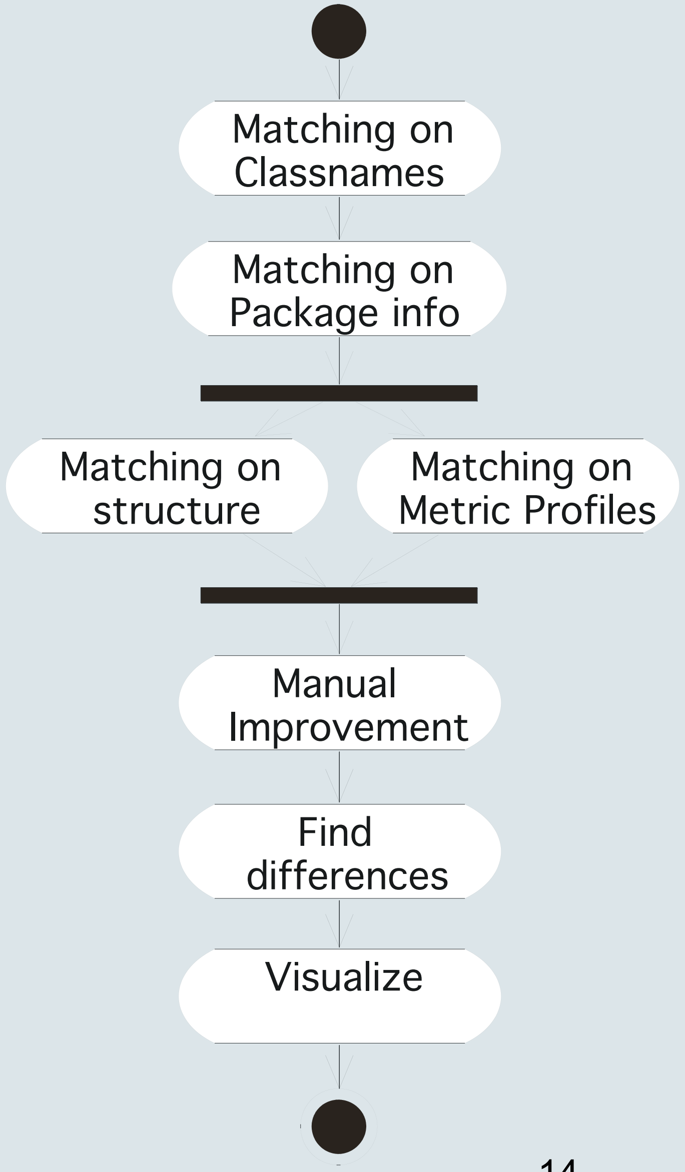
Conclusions



- Matching approaches
  - Matching based on names:
    - 77 % of design matched
    - ? % of implementation matched
  - Matching based on Metric Profiles
    - 0 % of design matched
    - 0 % of implementation matched
  - Metric Profile useful for highlighting deviations

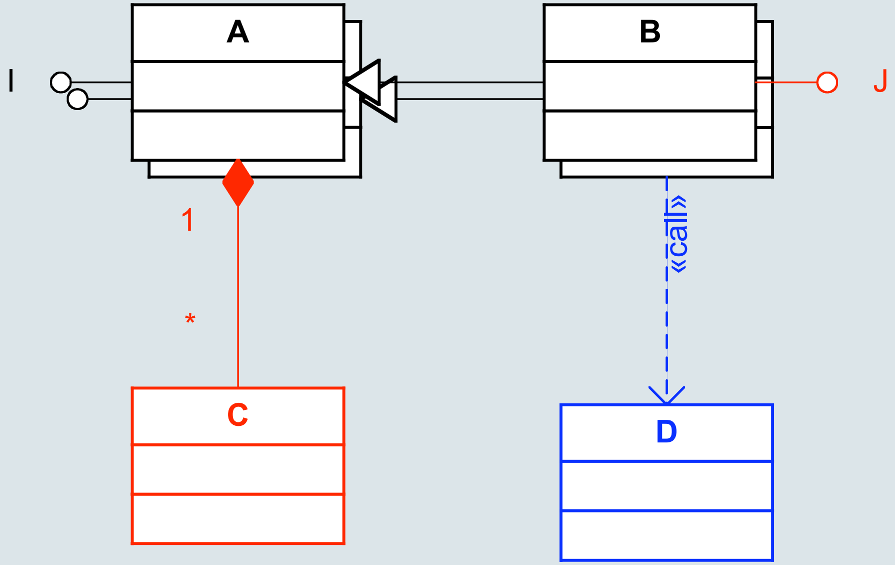
Combine strategies

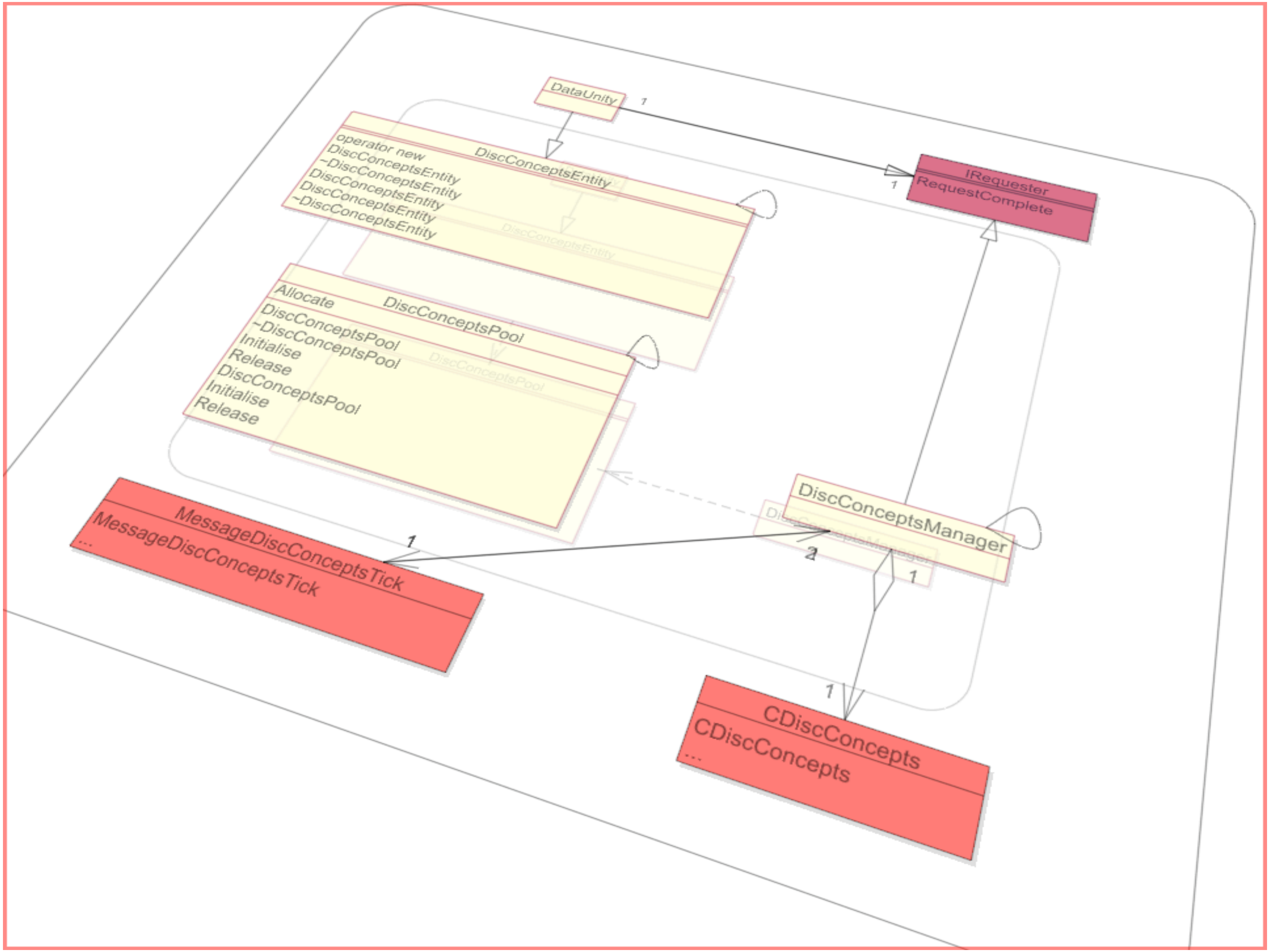
- None of the approaches defines a complete matching
- Find initial matching using a good approach
- Cluster classifiers using package information
- Apply other matching approaches on clusters
- If everything else fails: human intelligence



Visualization of differences

- Given a mapping, finding differences is quite straightforward
- Visualization using MetricView
- Overlay diagrams




DataUnity
1
DiscConceptsEntity
operator new
DiscConceptsEntity
~DiscConceptsEntity
DiscConceptsEntity
DiscConceptsEntity
~DiscConceptsEntity
IRequester
RequestComplete
1
DiscConceptsPool
Allocate
DiscConceptsPool
~DiscConceptsPool
Initialise
Release
DiscConceptsPool
Initialise
Release
DiscConceptsManager
1
2
1
MessageDiscConceptsTick
MessageDiscConceptsTick
...
1
CDiscConcepts
CDiscConcepts
...

Further work

- What can be done to prevent correspondence issues?
- How can correspondence be established?
- What is the impact of correspondence issues?
- How much correspondence is needed?
- What about clustering