

# Static Detection of Design Patterns in Class Diagrams

# Content

- Problem statement
- Design patterns
  - classifications
- Classifications of detection algorithms
- 3-tuples as representation of design patterns
- Feedback on elaborations of students
- Practical results
- Future work

# Problem statement

## The educational perspective

- Marking and giving feedback on elaboration of exercises is:
  - time-consuming
  - not a popular task
- Quality of marking and giving feedback may vary in time and between teachers.
- Students would like to have immediate feedback.

# Design patterns

Definition: a software design pattern is a general, reusable solution to a commonly occurring problem within a given context in software design.

It is

- based on best practices
- a combination of text and diagrams
- not finished

There are 23 standard design patterns (Eric Gamma et. al)

# Classification of design patterns

Classifications:

Based on using

- Creational
- Behavior
- Structural

Based on level of applying

- Architectural
- Design of subsystems and components
- Idiom (programming level)

# New classification

Focused on detection:

A design pattern is:

- Static, if it is completely defined by the names of their participating classes and their relationships.
- Non-static, if it needs more characteristics than names of their participating classes and relationships to be defined.

# Classifications of detection algorithms

Based on representation of a design pattern

- Matrices
- Prolog clauses
- Decision trees
- .....
- 4-tuples and 3-tuples

# Classifications of detection algorithms

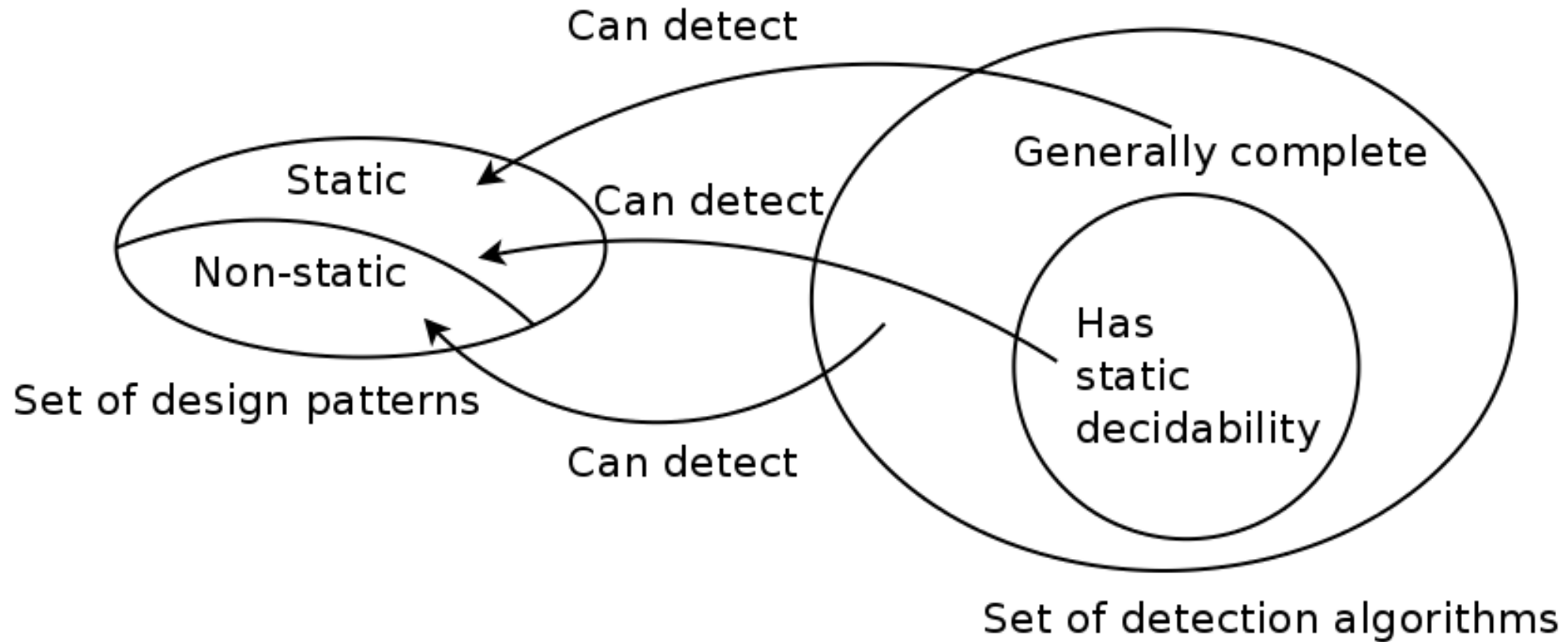
Based on their features

A detection algorithm

- offers static decidability, if it can detect all static design patterns
- is generally complete, if it can detect all design patterns



# Relations between the definitions



# 3-tuples

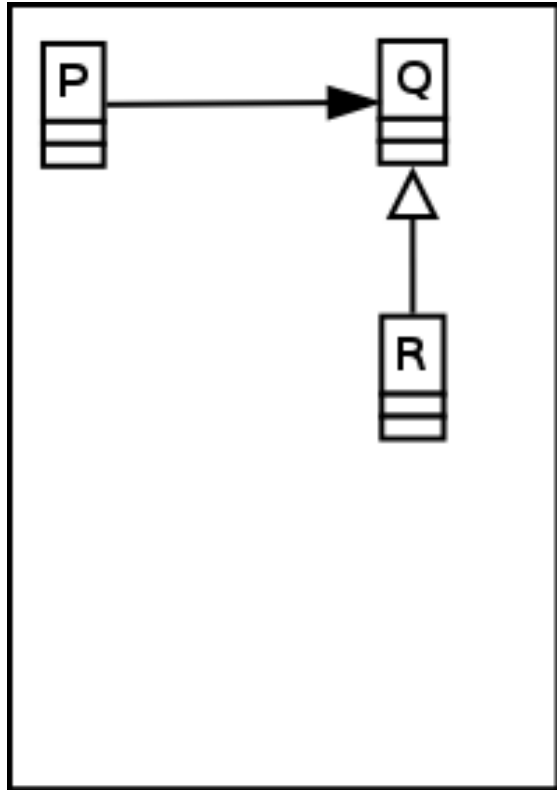
For static design patterns:

3-tuple (classname\_A, classname\_B, type of relationship)

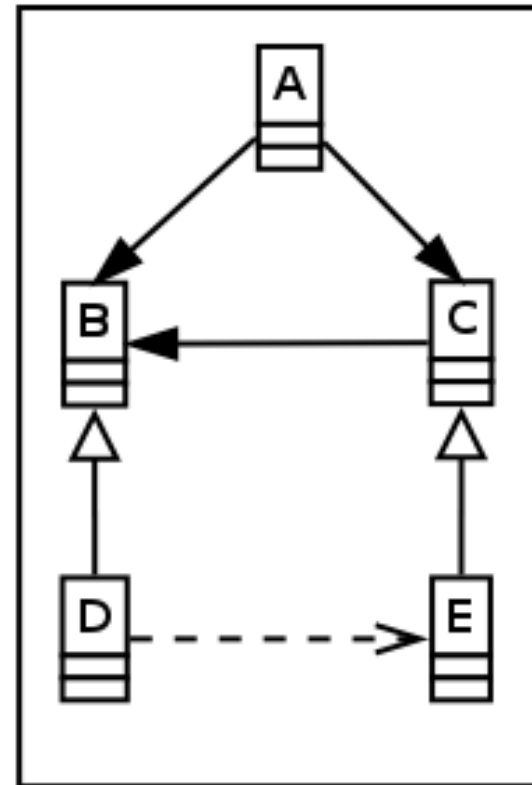
Algorithm:

- Design pattern is defined by a template of 3-tuples.
- Software design is defined by a large set of 3-tuples.
- A depth first search tries to match the template with a part of the software design.

# Example



Template of a pattern

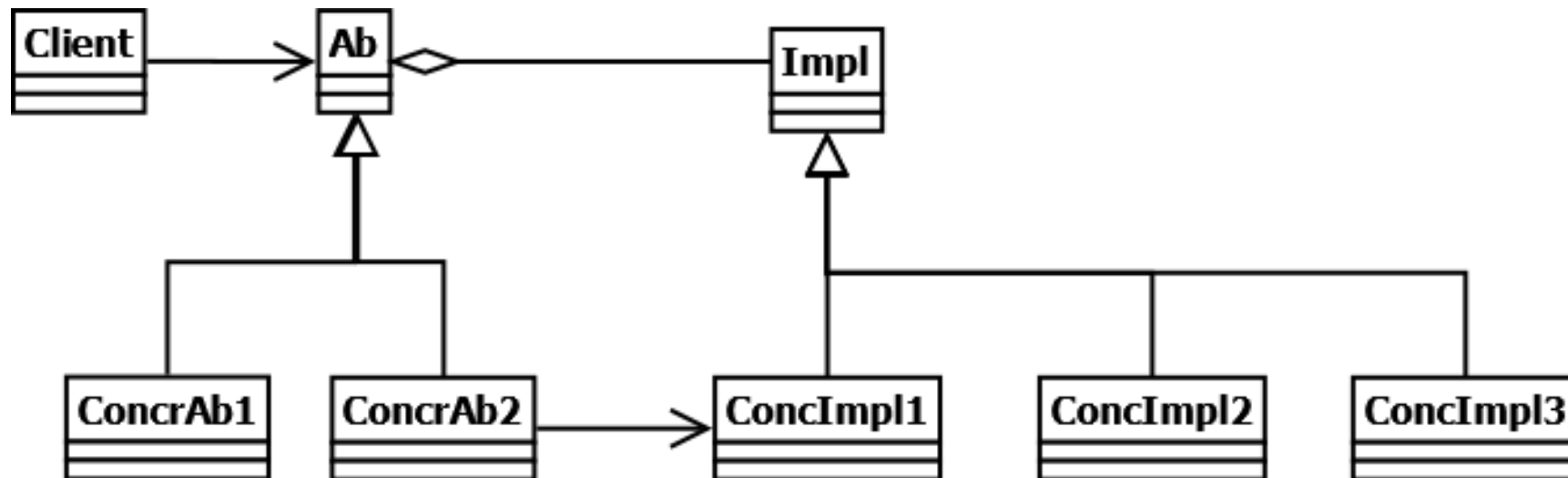
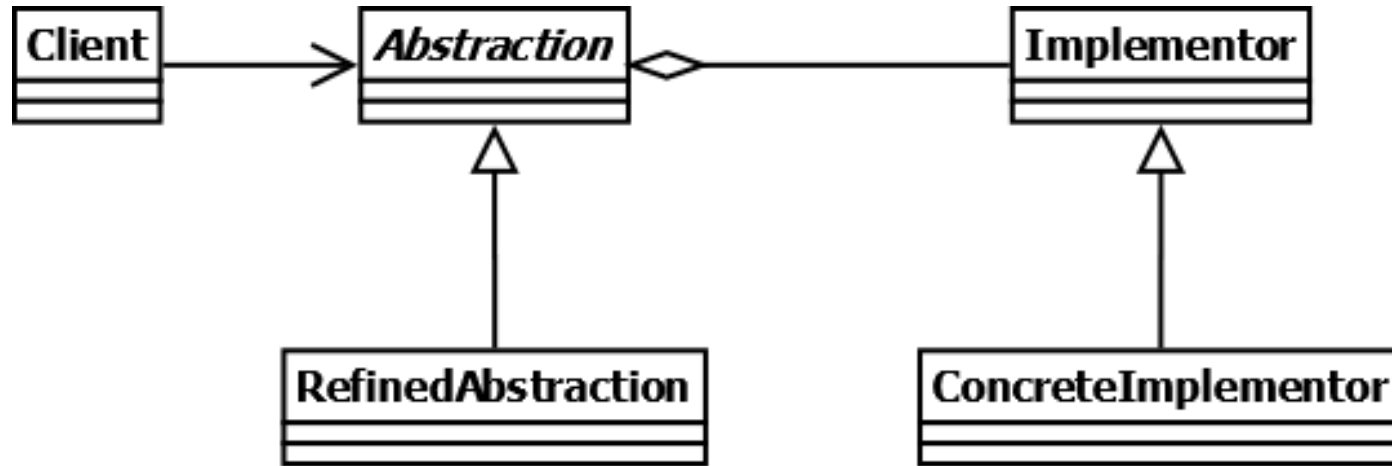


System under consideration

# Practical problems

- Multiple realizations of inheritance
- Abstract factory
- Report an instance of a design pattern only once.

# Multiple realizations of inheritance

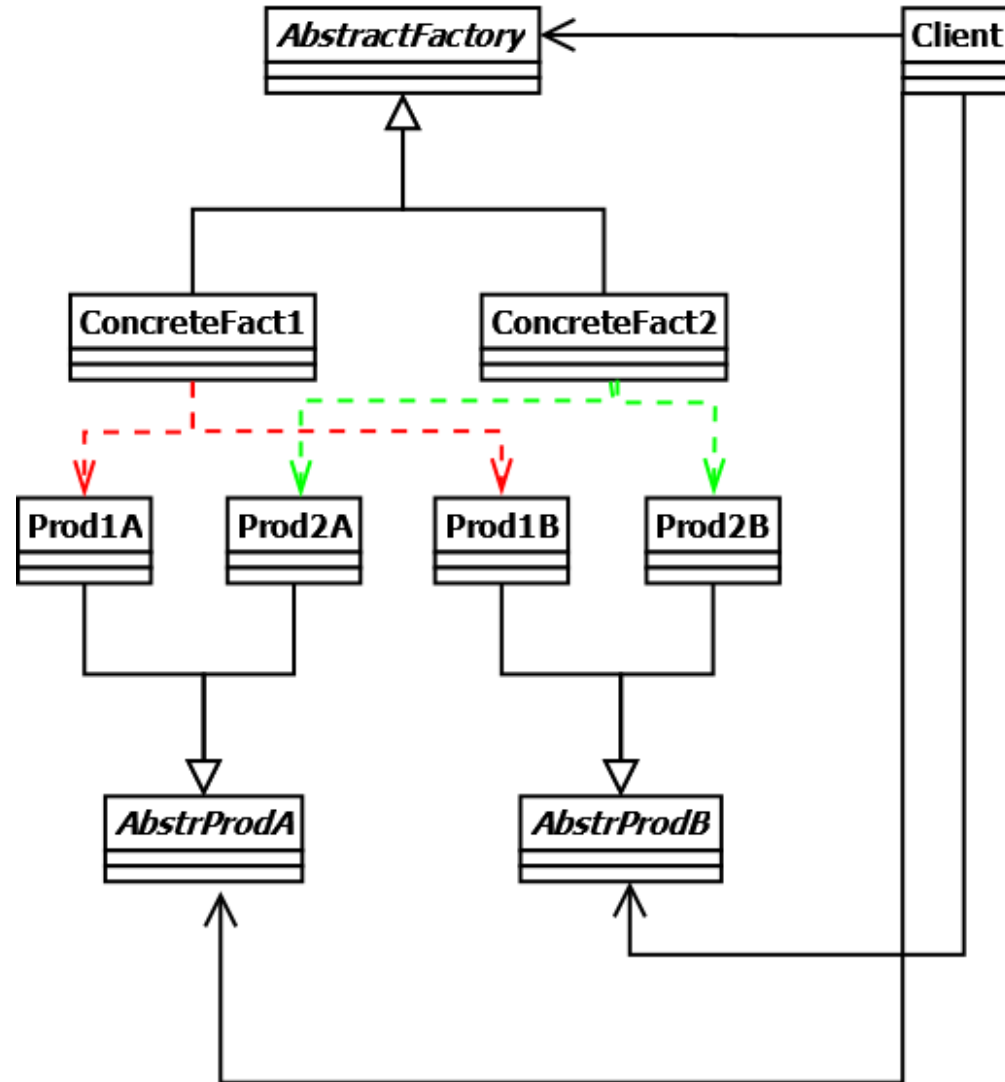


# Non-duplicating

- A detection algorithm is *non-duplicating*, if it detects every occurrence of a design pattern only once.

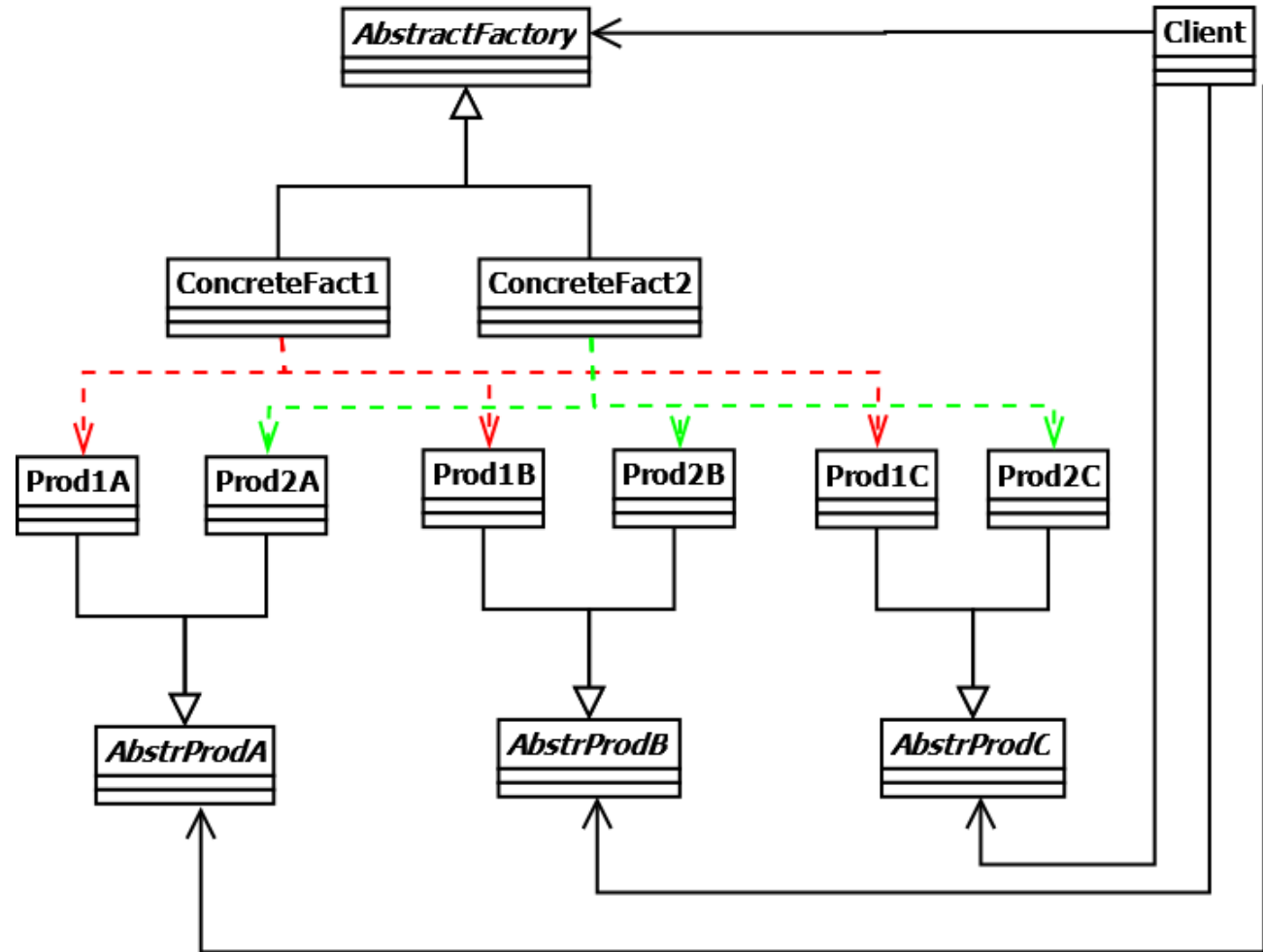
# Multiple realizations of inheritance

Abstract Factory:  
2 factories and 2  
products



# Multiple realizations of inheritance: unsolved

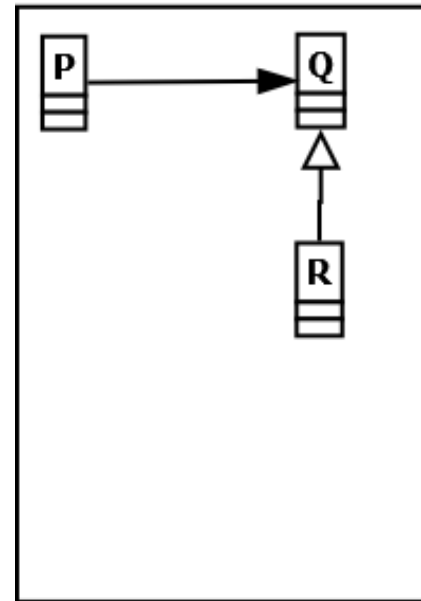
Abstract Factory  
2 factories and 3  
products



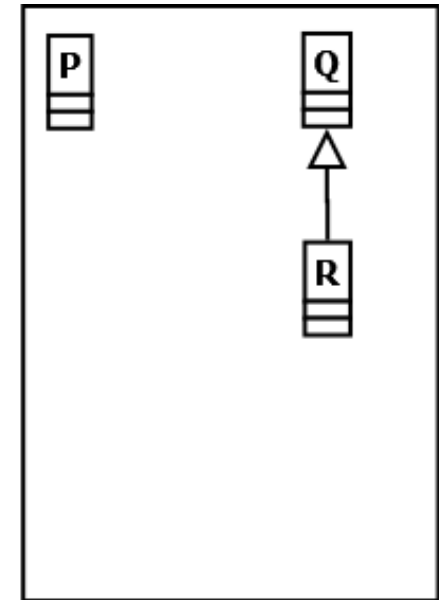


# Feedback

- Illegal relationship  
Example is shown in the bridge pattern
- Partial present  
Useful if the incomplete design pattern is a connected graph.



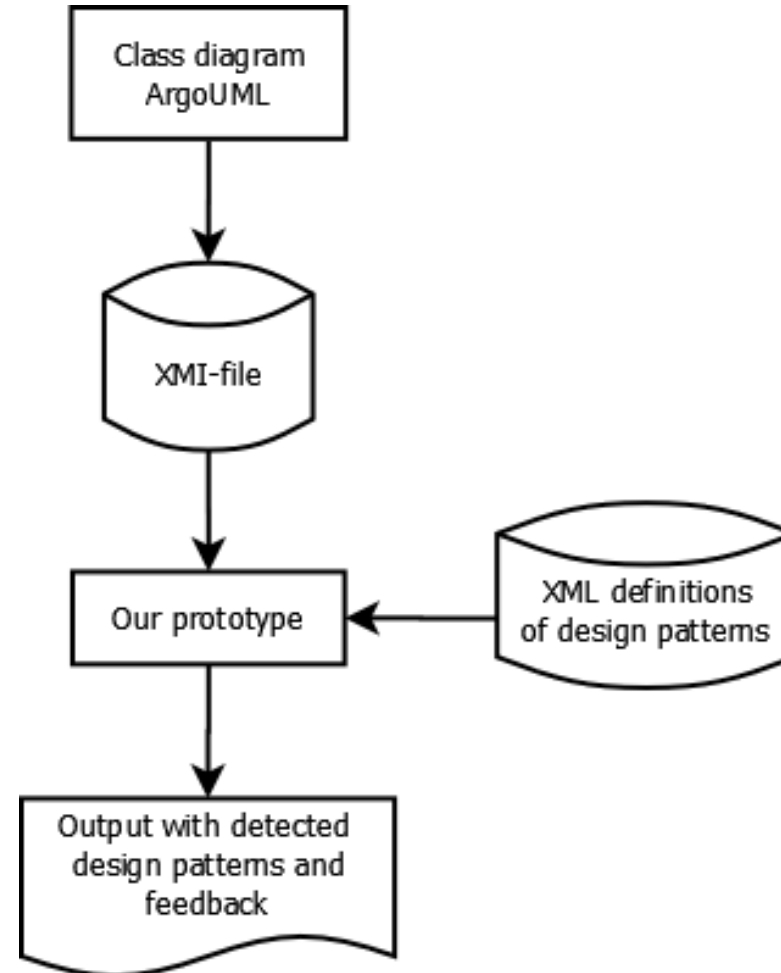
Template of a pattern



System contains a part of the DP

# In practice

- ArgoUML is a drawing tool which output can be used by our prototype
- Detecting different design patterns during a search.



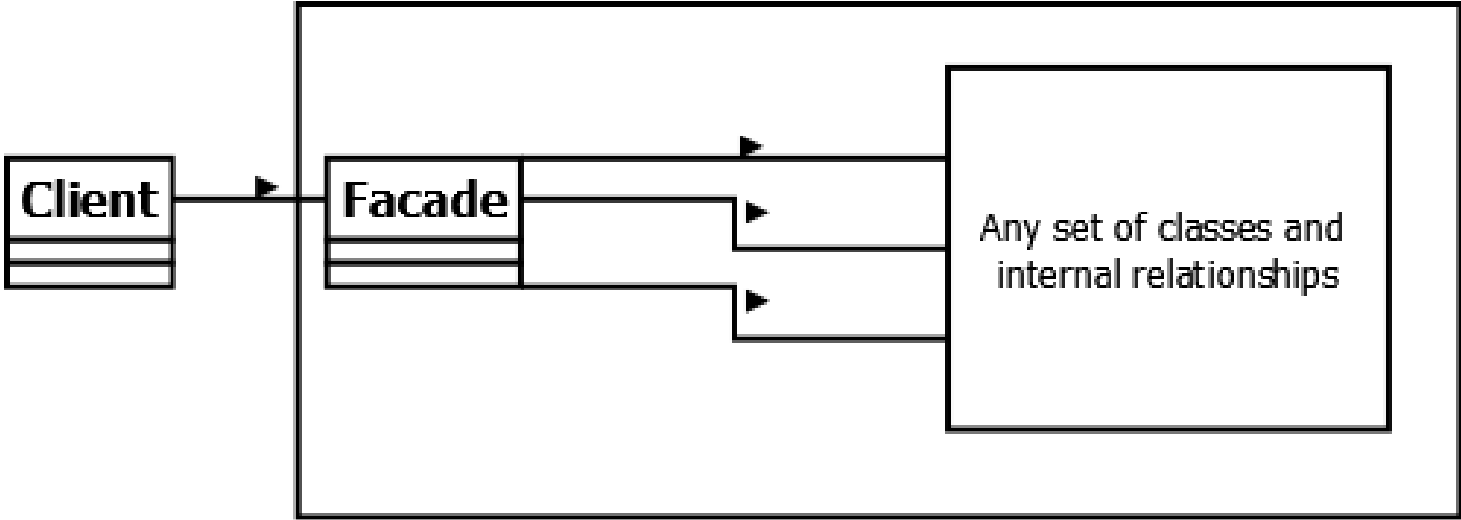
# Practical results

- 13 different design patterns are detected in a class diagram, which contains 57 classes and 61 relationships within 1 second.
- 33 classes and 49 relationships, 17 partially overlapping design patterns: 0.8 seconds
- There are 23 standard design patterns (Eric Gamma et. al)  
all 16 static patterns are detectable.

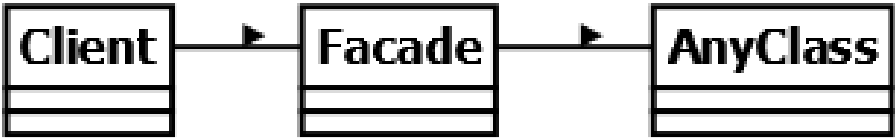
# Not detectable by our prototype

Non-static pattern	Reason
Prototype	Operation clone is necessary
Singleton	Needs static attribute, static method and private constructor
Façade	See next sheet
State / Strategy	They are structural identical
Template Method	Operations have to be taken into account.
Visitor	Number of classes depends on the number of methods in the interface Visitor.

# Façade pattern



Facade pattern in general



Specific Facade pattern

# Future work

## Specification of design patterns

- Detecting an abstract factory only once.
- Generally complete algorithm
- **Feedback on design**
  - Is the prototype of the tool useful?
  - Metrics of quality aspects
  - If design patterns are examples of high quality design, which values of the metrics do they have in common?
  - Relations between subsets of metrics and abstract features like *minimal coupling*, *maximal cohesion*.

# Contact

- E.M.vanDoorn@hhs.nl
- Source code, jar-file, ArgoUML examples and templates:  
[http://members.chello.nl/e.doorn1/DesignPatterns/static\\_decidability](http://members.chello.nl/e.doorn1/DesignPatterns/static_decidability)