Discovery of design patterns based on good practices

Rafael Barbudo Lunar

Dept. of Computer Science and Numerical Analysis

University of Córdoba, Spain

Content

- Introduction
- Conceptual framework
  - ✓ *Our approach in a nutshell*
  - ✓ *What do we do?*
- Proposed approach
  - ✓ *Our approach in detail*
- Open issues

Introduction

Building software systems is not easy

Software systems are **COMPLEX**

Their developing process **TOO**…

How can we make it more accurate?

***Best practices*.** *Working method or set of working methods that are accepted as being the best to use in a particular area, usually described formally.*

- *Compliance with standards*
- *Documented code*
- *UML models*
- ***Design patterns (DP)***

Using them is not trivial →

We **must learn** how they have been used

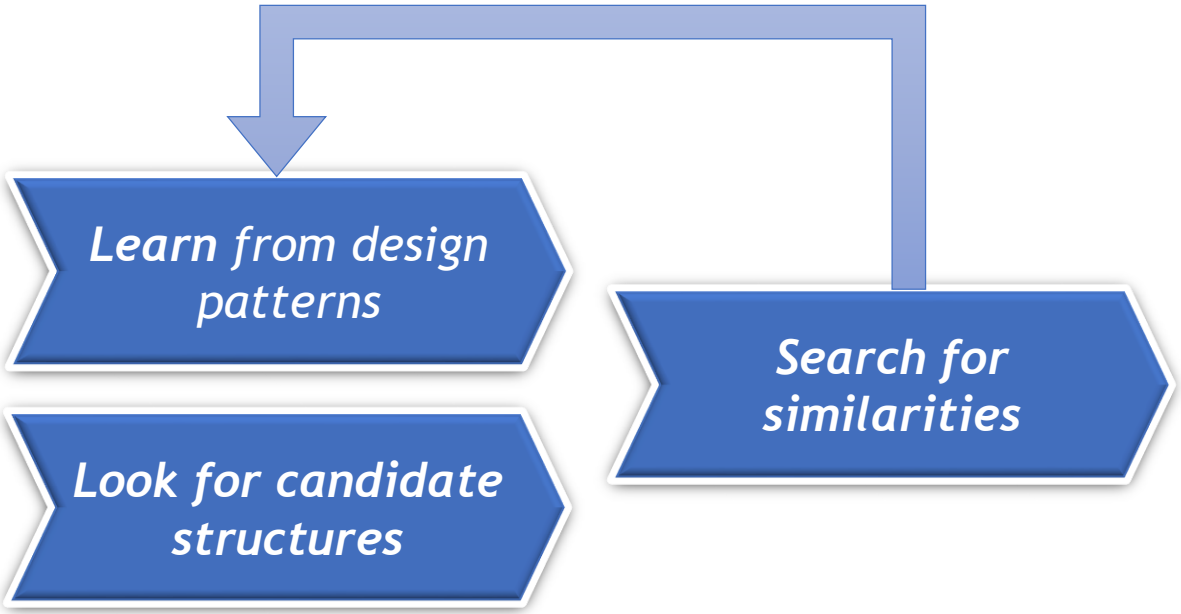
Mining Software Repositories

← Still difficult

*Can we learn how other developers are using **Design Patterns** in their projects?*
*Can we use this **knowledge to assist** the engineer?*

Conceptual framework

Our approach in a nutshell



Characterisation:

- ✓ Software metrics
- ✓ Relationships
- ✓ Internal structure

Known design pattern structure:

- ✓ Roles involved
- ✓ Roles relationships
- ✓ Roles cardinality

Useful knowledge allow us to know if:

- ✓ Is it similar?
- ✓ Could it be a pattern?

How are we going to learn?

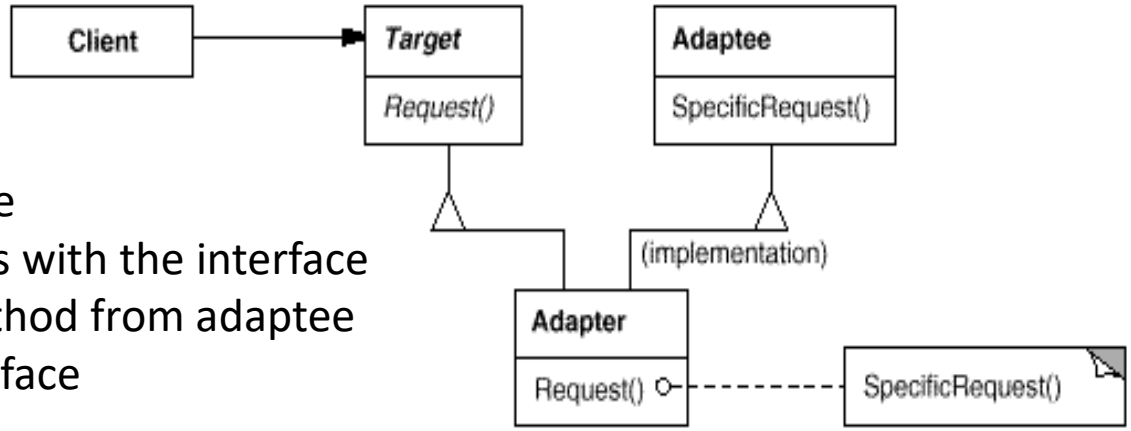
Mining Software Repositories → Data mining

Mining frequent itemsets

Conceptual framework

What do we do?

The case of the Adapter design pattern



Gamma et al. (1995)

- ✓ Target sets methods' interface
- ✓ Adapter implements methods with the interface
- ✓ Adapter overrides/wraps method from adaptee
- ✓ Adapter only adapts the interface

… many others

How can we represent this knowledge?

***type**(target) = interface* **AND** ***connect**(adapter, target) = implements* **AND** ***callMethodOf**(adapter, adaptee)* **AND** ***WMC**(adapter) < 0.5*

***connect**(adapter, adaptee) = extends* **AND** ***type**(target) = interface* **AND** ***connect**(adapter, target) = implements*

Proposed approach

Our approach in detail (I)

Grammar

<pattern> ::= <cmp> | *AND* <pattern> <cmp>
<cmp> ::= <catCmptr> <catUnProperty> <role>
| <catCmptr> <catBinProperty> <role> <role>
| <numCmptr> <metric> <role>
| <boolCmptr> <role>
<catCmptr> ::= *equal* | *notEqual*
<catUnProperty> ::= un*Property value*
<catBinProperty> ::= bin*Property value*
<numCmptr> ::= > | < | ≥ | ≤
<boolCmptr> ::= bool*Property value*
<metric> *::= LOC valueLOC* | *WMC valueWMC*
<role> ::= *adapter* | *adaptee* | *client* | *target*

Labelled design patterns

Adapter 1:
client
net.nutch.db.WebDBWriter.PageInstructionWriter
target
net.nutch.io.Writable
adapter
net.nutch.db.Page
adaptee
java.lang.CloneableAdapter
…

P-mart repository

Itemsets

type(target) = interface ***AND*** *connect(adapter, target) = implements* ***AND*** *WMC(adapter) < 0.5* ***AND*** *connect(adapter, adaptee) = extends*
…

Learning phase

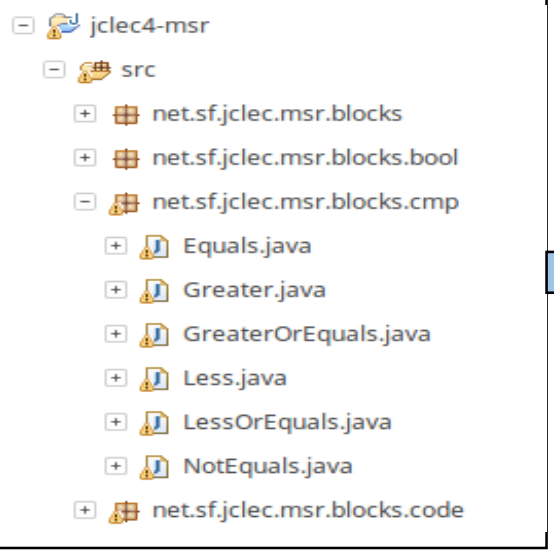
- ✓ *G3P algorithm*
- ✓ *Stores the best itemsets*
- ✓ *Custom fitness measures*
- ✓ *Parse source code*

Property	Values
connect	extends, implements, notLinked
type	concreteClass, abstractClass, interface
isSubclass	true, false
privateConst	true, false

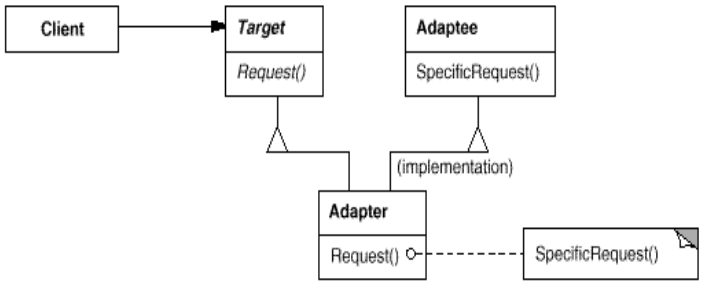
Proposed approach

Our approach in detail (II)

Source code



Structural description



Extraction phase

- ✓ *Look for candidates*
- ✓ *Pattern structure is known*
- ✓ *Subgraph algorithm*
- ✓ *Isomorphic problem*

Candidates

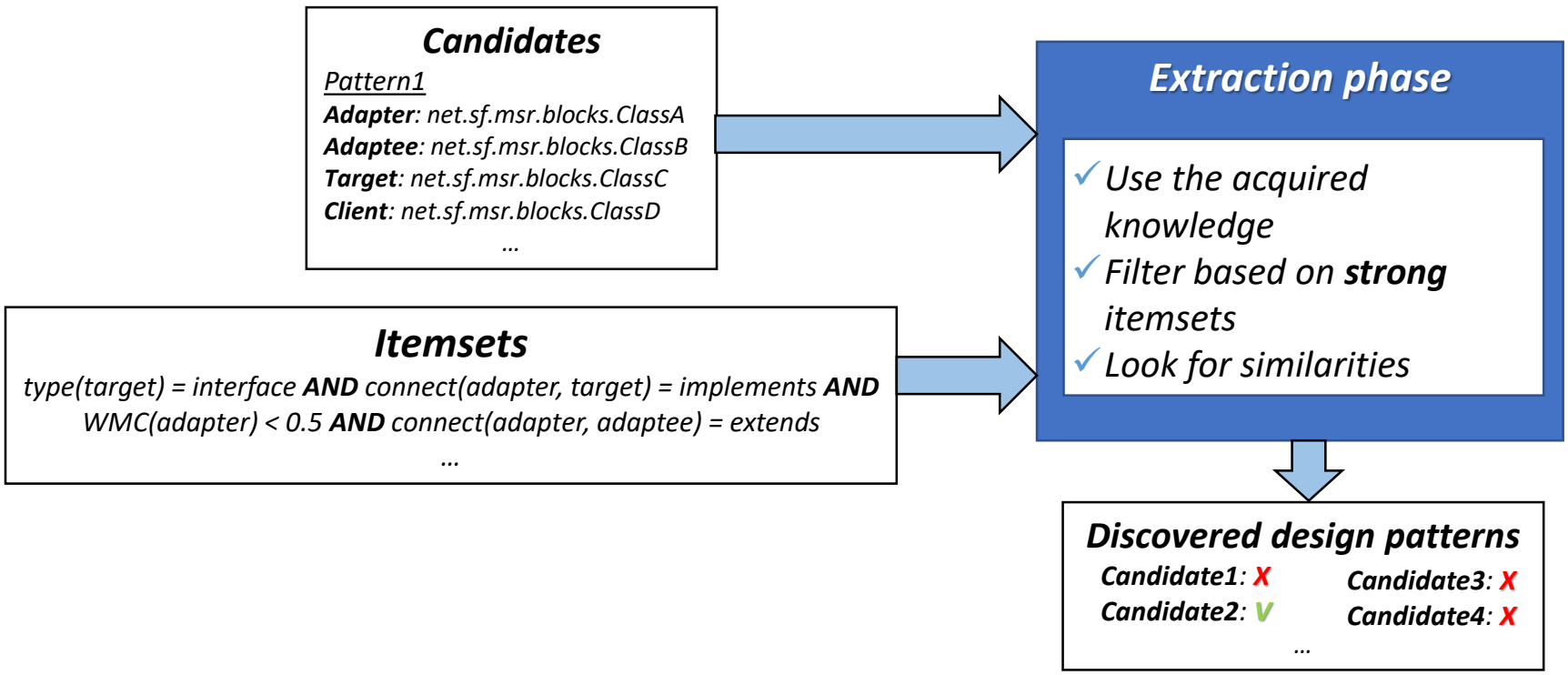
Pattern1

***Adapter**: net.sf.msr.blocks.ClassA*
***Adaptee**: net.sf.msr.blocks.ClassB*
***Target**: net.sf.msr.blocks.ClassC*
***Client**: net.sf.msr.blocks.ClassD*
...

Proposed approach

Our approach in detail (III)

- Use the knowledge from previous phases



Open issues

The road ahead

- *Which are the most interesting software metrics to be studied?*

  There are a lot … DIT, LCOM, CBO, NOC…

- *Can roles have different cardinalities?*

  - ✓ Some roles could not be present
  - ✓ Some roles could be played by more than one element

  Extraction phase becomes more difficult

- *Are data **good** and **enough**?*

  - ✓ P-mart repository has few labelled instances
  - ✓ Are different these patterns implemented by different developers?
    - Cultural aspects
    - Sort of project
    - Different communities

UNIVERSIDAD DE CÓRDOBA

KDIS Lab
KNOWLEDGE DISCOVERY AND INTELLIGENT SYSTEMS

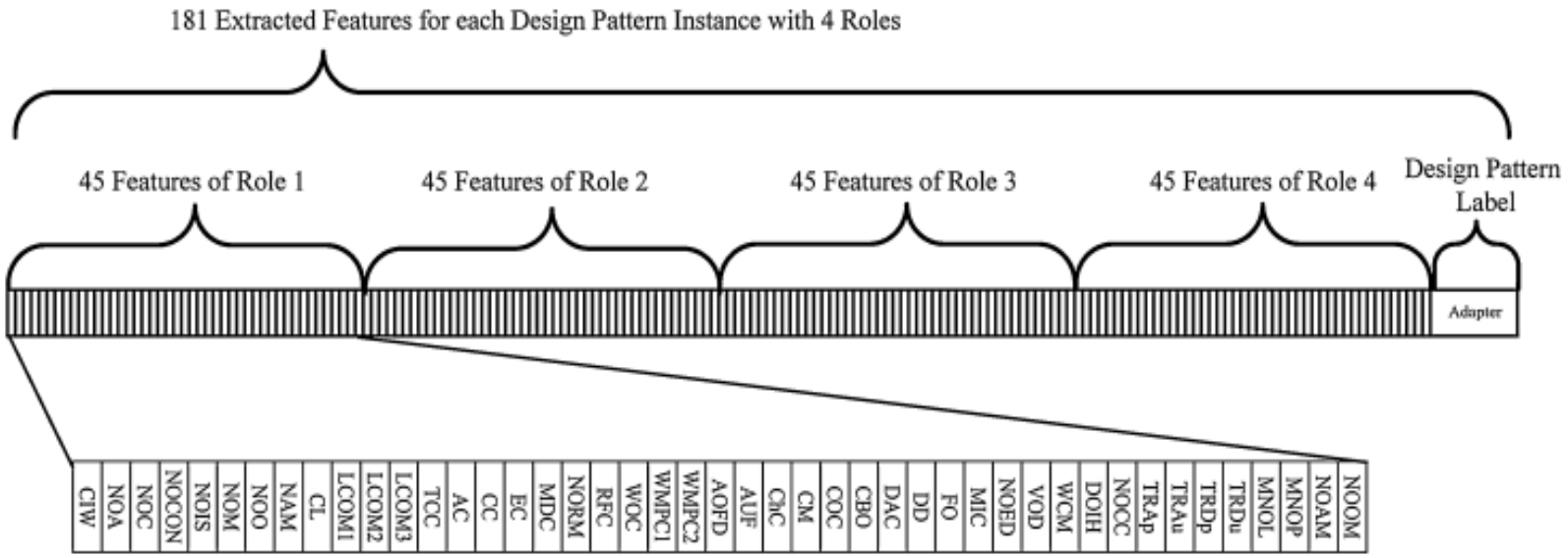
Discovery of design patterns based on good practices

Any question?

Conceptual framework

Previous work

➢ *Design pattern detection based on software metrics*



Chihada et al (2015)

- ✓ Extract metrics from labelled design patterns
- ✓ Build a classification model
- ✓ Search candidates based on the structure

Conceptual framework

Previous work

➢ *Design pattern detection based on categorical properties*

- ✓ Build enriched graph from code
- ✓ Search subgraphs based on the edge labels
- ✓ Manually defines the pattern structure

Edge Label	Relation Type[a]
X	Class A extends Class B
I	Class A implements Class B
C	Class A creates object of Class B
O	Class A overrides a method of Class B
MC	Class A calls a method of Class B
F	Class A has the field type of Class B
MR	Class A has a method with the return type of Class B
ML	Class A has a method that defines a local variable with the type of Class B
MI	Class A has a method that has an input parameter with the type of Class B
M	Class A has related with its method of Class B
R	Class A has the return type of Class B
G	Class A uses Class B in a generic type declaration

Oruc et al (2016)

Why do not combine?

Remember → ✓ Software metrics ✓ Relationships ✓ Internal structure … many others → Grammar → ✓ ***Flexible*** ✓ ***Extensible***