A Model-Driven Graph-Matching Approach for Design Pattern Detection

Mario Luca Bernardi\textsuperscript{1} Marta Cimitile\textsuperscript{2}
Giuseppe A. Di Lucca\textsuperscript{1}

\textsuperscript{1}University of Sannio, Benevento, Italy
\textsuperscript{2}Te.L.Ma. University, Rome, Italy

WC\textsuperscript{RE} 2013, 14-17 October
Koblenz, Germany
Outline

1 Motivations

2 Design Pattern Detection: a Model-Driven Graph-Matching Approach

3 Tool Support

4 Case Study and Results

5 Conclusions and Future Work
While Design Patterns (DPs) usage in Object Oriented (OO) software systems can help to better structure, understand, maintain and reuse them . . .
Motivations

- While Design Patterns (DPs) usage in Object Oriented (OO) software systems can help to better **structure**, **understand**, **maintain** and **reuse** them . . .
- . . . but the **lack of adequate documentation** in a software system may make it hard to understand **which**, and **where** in the system, **design patterns are implemented**
Motivations

While Design Patterns (DPs) usage in Object Oriented (OO) software systems can help to better **structure**, **understand**, **maintain** and **reuse** them . . .

. . . but the **lack of adequate documentation** in a software system may make it hard to understand **which**, and **where** in the system, **design patterns are implemented**

Existing pattern mining approaches and tools are **too much sensitive to structural differences** of searched patterns with respect their specifications.
The proposed approach in brief

- centered on a single metamodel that represents both the software system and the patterns to be detected as annotated graphs.

System Graph (S):
- Circle
- Rect
- Text
- Line
- text
- src
- dst
- add
- remove

Observer Graph (DP):
- N
- Container-of
- AO(1)
- R
- A
- CS(*)
- AS(1)
- calls
- delegates
- Container-of
- calls
- delegates
- Container-of
- calls
- delegates
- Container-of
- calls
- delegates
The proposed approach in brief

- exploits a **wider set of high level properties** related to the source code elements, the static relationships among them, and their behavior

**DELEGATION**
**ALLOCATION**
**BEHAVIOUR**
**OVERRIDING**
**INVOCATION**

**COMPOSITION**
**INHERITANCE**
**STRUCTURE**
**IMPLEMENTATION**
**AGGREGATION**
**NESTING**
The proposed approach in brief

- organizes the design pattern models as a hierarchy of declarative specifications using a DSL in order to take into account structural implementation differences
Model-Driven Graph-Matching Pattern Mining

The Metamodel and DSL
A Design Pattern example specification
The Detection Approach

Outline
Motivations
Model-Driven Graph-Matching Pattern Mining
Tool Support
Case Study and Results
Conclusions and Future Work

Bernardi, Cimitile, Di Lucca

Model-Driven Graph-Matching Pattern Detection
The metamodel is also the base for a DSL that can be used to represent design patterns declaratively.

```
pattern observer {
  type AS(1) {
    has method A,R;
    has method N;
    has container o of type AO;
  }
  type AO(1) {
    has method U;
  }
  type CO(*) {
    inherits from AO;
  }
  type CS(*) {
    inherits from AS;
    has constructor c {
      object-creation o;
    }
    overrides methods [A,R] each {
      delegates to o;
    }
    overrides method N each {
      delegates to o;
      calls U in AO.U;
    }
  }
}
```
The Metamodel and DSL
A Design Pattern example specification

The Detection Approach

1. Bytecode or Java Files
2. Bytecode or Source Code Analysis
3. System ASTs
4. AST to GRAPH
5. System Graph
6. Patterns Templates (DSL)
7. DSLs to GRAPHS
8. Patterns Sub-Graphs
9. Matching Engine
   - CANDIDATE NEIGHBORHOODS IDENTIFICATIONS
   - GRAPH MATCHER
10. Matched Patterns Instances
The Metamodel and DSL
A Design Pattern example specification

The Detection Approach

Figure
Text
Line
Drawing

System Graph (S)

Observer Graph (DP)

outcome: False

Bernardi, Cimitile, Di Lucca
Model-Driven Graph-Matching Pattern Detection
The Metamodel and DSL
A Design Pattern example specification

The Detection Approach

outcome: True
added candidate bindings:
Figure <-> AS
Line <-> CS
Text <-> CS
obs <-> o
add <-> A
remove <-> R
notify <-> N

Bernardi, Cimitile, Di Lucca
Model-Driven Graph-Matching Pattern Detection
Figure FigureListener

System Graph (S)

Observer Graph (DP)

outcome: False
The Metamodel and DSL
A Design Pattern example specification

The Detection Approach

Bernardi, Cimitile, Di Lucca
Model-Driven Graph-Matching Pattern Detection
Model-Driven Graph-Matching Pattern Mining

Outline
Motivations
Model-Driven Graph-Matching Pattern Mining
Tool Support
Case Study and Results
Conclusions and Future Work

The Design Pattern Finder Architecture
The Design Pattern Finder Screenshot
Outline
Motivations
Model-Driven Graph-Matching Pattern Mining
Tool Support
Case Study and Results
Conclusions and Future Work

The Design Pattern Finder Architecture
The Design Pattern Finder Screenshot
### The Case Study Setup

Results from group 1 are from systems of the online benchmark cited in the paper. Results from group 2 are from bigger systems directly compared with DPD tool.

<table>
<thead>
<tr>
<th>System Name</th>
<th>Version</th>
<th>Size (KLOC)</th>
<th>#Types</th>
<th>#Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junit</td>
<td>3.7</td>
<td>4,9K</td>
<td>104</td>
<td>648</td>
</tr>
<tr>
<td>Lexi</td>
<td>0.1</td>
<td>7,1K</td>
<td>100</td>
<td>677</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>5.1</td>
<td>8,9K</td>
<td>174</td>
<td>1316</td>
</tr>
<tr>
<td>QuickUML</td>
<td>2.1</td>
<td>9,2K</td>
<td>230</td>
<td>1082</td>
</tr>
<tr>
<td>Nutch</td>
<td>0.4</td>
<td>23,6K</td>
<td>335</td>
<td>1854</td>
</tr>
<tr>
<td>PMD</td>
<td>1.8</td>
<td>41,5K</td>
<td>519</td>
<td>3665</td>
</tr>
<tr>
<td>Log4J</td>
<td>1.2.15</td>
<td>43,7K</td>
<td>176</td>
<td>863</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>7</td>
<td>78,5K</td>
<td>567</td>
<td>5728</td>
</tr>
<tr>
<td>Voldemort</td>
<td>1.3.x</td>
<td>85,9K</td>
<td>382</td>
<td>5312</td>
</tr>
<tr>
<td>Apache Avro</td>
<td>1.6</td>
<td>125,2K</td>
<td>1085</td>
<td>1854</td>
</tr>
<tr>
<td>JDT</td>
<td>3.6.1</td>
<td>511,5K</td>
<td>1655</td>
<td>24153</td>
</tr>
</tbody>
</table>
### The Case Study Setup

#### Results on Design Pattern Mining Benchmark

A Comparison with DPD

<table>
<thead>
<tr>
<th>System →</th>
<th>QuickUML 2001</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GS</td>
</tr>
<tr>
<td>Down Design Pattern</td>
<td></td>
</tr>
<tr>
<td>Abstract Factory</td>
<td>16</td>
</tr>
<tr>
<td>Builder</td>
<td>12</td>
</tr>
<tr>
<td>Command</td>
<td>6</td>
</tr>
<tr>
<td>Composite</td>
<td>29</td>
</tr>
<tr>
<td>Observer</td>
<td>17</td>
</tr>
<tr>
<td>Singleton/spec {gof-relaxed}</td>
<td>1</td>
</tr>
<tr>
<td>Singleton</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System →</th>
<th>Lexi v0.1.1 alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GS</td>
</tr>
<tr>
<td>Down Design Pattern</td>
<td></td>
</tr>
<tr>
<td>Builder</td>
<td>5</td>
</tr>
<tr>
<td>Observer</td>
<td>11</td>
</tr>
<tr>
<td>Singleton/spec {gof-relaxed}</td>
<td>2</td>
</tr>
<tr>
<td>Singleton</td>
<td>2</td>
</tr>
<tr>
<td>Design Pattern</td>
<td>GS</td>
</tr>
<tr>
<td>------------------</td>
<td>----</td>
</tr>
<tr>
<td>Adapter</td>
<td>7</td>
</tr>
<tr>
<td>Bridge</td>
<td>25</td>
</tr>
<tr>
<td>Command</td>
<td>14</td>
</tr>
<tr>
<td>Iterator</td>
<td>7</td>
</tr>
<tr>
<td>Memento</td>
<td>15</td>
</tr>
<tr>
<td>Singleton</td>
<td>1</td>
</tr>
<tr>
<td>Strategy</td>
<td>8</td>
</tr>
<tr>
<td>Template Method</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design Pattern</th>
<th>GS</th>
<th>D</th>
<th>TP</th>
<th>FP</th>
<th>P</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapter</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>Builder</td>
<td>12</td>
<td>10</td>
<td>9</td>
<td>1</td>
<td>0.9</td>
<td>0.75</td>
</tr>
<tr>
<td>Composite</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>Factory Method</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>1</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>Iterator</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>Observer</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>1</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>Proxy</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>Template Method</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.67</td>
<td>1</td>
</tr>
<tr>
<td>Visitor</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>System →</td>
<td>JUnit v3.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>↓ Design Pattern</td>
<td>GS</td>
<td>D</td>
<td>TP</td>
<td>FP</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Composite</td>
<td>39</td>
<td>34</td>
<td>33</td>
<td>1</td>
<td>0.97</td>
<td>0.85</td>
</tr>
<tr>
<td>Decorator</td>
<td>37</td>
<td>31</td>
<td>30</td>
<td>1</td>
<td>0.97</td>
<td>0.81</td>
</tr>
<tr>
<td>Iterator</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>0.83</td>
<td>0.83</td>
</tr>
<tr>
<td>Observer</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>1</td>
<td>0.92</td>
<td>0.79</td>
</tr>
<tr>
<td>Singleton/spec {gof-relaxed}</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.67</td>
<td>1</td>
</tr>
<tr>
<td>Singleton</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.67</td>
<td>1</td>
</tr>
</tbody>
</table>

| System → | JHotDraw v5.1 | | | | | |
| --- | --- | --- | --- | --- | --- | |
| ↓ Design Pattern | GS | D | TP | FP | P | R |
| Adapter | 32 | 29 | 27 | 2 | 0.93 | 0.84 |
| Command | 25 | 25 | 22 | 3 | 0.88 | 0.88 |
| Composite | 30 | 32 | 29 | 3 | 0.91 | 0.97 |
| Decorator | 23 | 23 | 20 | 3 | 0.87 | 0.87 |
| Factory Method | 55 | 52 | 49 | 3 | 0.94 | 0.89 |
| Observer | 29 | 32 | 29 | 3 | 0.91 | 1 |
| Prototype | 26 | 25 | 22 | 3 | 0.88 | 0.85 |
| Singleton/spec {gof-relaxed} | 2 | 7 | 2 | 5 | 0.29 | 1 |
| Singleton | 2 | 2 | 2 | 0 | 1 | 1 |
| State | 18 | 25 | 18 | 7 | 0.72 | 1 |
| Strategy | 34 | 41 | 34 | 7 | 0.83 | 1 |
| Template Method | 24 | 29 | 22 | 7 | 0.76 | 0.92 |
The Case Study Setup
Results on Design Pattern Mining Benchmark
A Comparison with DPD

The Case Study Setup
Results on Design Pattern Mining Benchmark
A Comparison with DPD
Conclusions

- A DSL-based Graph-matching approach to detect design pattern has been presented.
Conclusions

- A DSL-based Graph-matching approach to detect design pattern has been presented.
- It allows to specify pattern variants by overriding already defined pattern specifications.

Bernardi, Cimitile, Di Lucca
Conclusions

- A DSL-based Graph-matching approach to detect design pattern has been presented.
- It allows to specify pattern variants by overriding already defined pattern specifications.
- A validation on eleven open-source OO systems from an open benchmark shows an improvement on average values of precision and recall with respect to the state of art (respectively 0.89 and 0.93).
Future Work

- More systems and tools will be considered
Future Work

- **More systems and tools** will be considered.
- Work also involves the **improvement of the metamodel** (and the related DSL) in order to infer **a wider sets of source code properties**.
Future Work

- **More systems and tools** will be considered.
- Work also involves the **improvement of the metamodel** (and the related DSL) in order to infer **a wider sets of source code properties**.
- Use the approach and tool to mine **other information** (like idioms and anti-patterns) and to study **design pattern evolution**.
Thank you for listening!

Any questions?