DPB: A Benchmark for Design Pattern Detection tools

Francesca Arcelli Fontana, Andrea E. F. Caracciolo, Marco Zanoni
University of Milano-Bicocca, Italy

30/03/2012 - CSMR 2012 - Szeged, Hungary
Main Goal

Define a system allowing users to compare the quality of Design Pattern Detection (DPD) tools results

Who cares?

- **End users**: to be able to choose a tool
- **Researchers**: compare existing techniques/ reuse valid techniques
Related works

• DEEBEE [Fülöp et al., 2008]
  ✓ Usability
  x Data model

✓ Open web application
✓ Interesting choice of functionalities

• P-MARt [Guéhéneuc, 2007]
  x No support for discussion
  x No way to measure reliability

✓ Pattern instances identified by experts
Proposed Solution

1. Representation
2. Analysis and Evaluation
3. Comparison
4. Search

DPB: A Benchmark for Design Pattern Detection tools
Representation (panoramic)
Representation (meta-model)
DPB: A Benchmark for Design Pattern Detection tools

Representation (model)
Analysis and Evaluation

Analysis

Discussion

Evaluation 1 – 5
With comment

Vote

Instance score

DP Instance

3
4
3
2

+ + -

rating(instance) = \frac{\sum_{i=1}^{|evals|} \text{eval}_i \cdot \text{votesBalance}_i}{\sum_{i=1}^{|evals|} \text{votesBalance}_i}

\text{votesBalance}_i = \max(\text{def} + \text{votes}_i^+ - \text{votes}_i^-, 0)

DPB: A Benchmark for Design Pattern Detection tools
Analysis and Evaluation (UML diagram)

DPB: A Benchmark for Design Pattern Detection tools
Analysis and Evaluation (structural view)
Analysis and Evaluation (structural view 2)
Analysis and Evaluation (source code)

```java
public abstract class DecoratorFigure
    extends AbstractFigure
    implements FigureChangeListener {

    /**
     * The decorated figure.
     */
    protected Figure fComponent;

    /**
     * Serialization support.
     */
    private static final long serialVersionUID = 899301151564573288L;
    private int decoratorFigureSerializedDataVersion = 1;

    public DecoratorFigure() { }

    /**
     * Constructs a DecoratorFigure and decorates the passed in figure.
     */
    public DecoratorFigure(Figure figure) {
        decorate(figure);
    }
```
Analysis and Evaluation (javadoc)

CH.ifa.draw.standard

**Class CompositeFigure**

java.lang.Object

- CH.ifa.draw.standard.AbstractFigure
  - CH.ifa.draw.standard.CompositeFigure

**All Implemented Interfaces:**

- Figure, FigureChangeListener, Storable, java.io.Serializable, java.lang.Cloneable, java.util.EventListener

**Direct Known Subclasses:**

- GroupFigure, PertFigure, StandardDrawing

```java
public abstract class CompositeFigure
extends AbstractFigure
implements FigureChangeListener
```
Analysis and Evaluation (side by side view)
Analysis and Evaluation (evaluations)

Add Evaluation
Rate detection accuracy:

Comment:

Send

Evaluations

⭐⭐⭐⭐⭐ by Marco Zanoni @ 20/02/11 (20:03)
There are some real composites figure, but the component is wrong and there is no leaf.

[Post a comment]

Andrea Caracciolo @ 10/10/11 (19:01)
I fully agree with you.

[Reply]

⭐⭐⭐⭐⭐ by Elio Salanitri @ 10/04/11 (17:06)
You can see that ConnectionFigure is an interface and it cannot be a Composite class properly. Then nor DecoratorFigure or CompositeFigure haven't any kind of collection of Component object as attribute. Finally there aren't any kind of methods to manage the collection itself and there aren't leafs classes.

[Post a comment]
Comparison (system analysis comparison)
Comparison (algorithm: weights setting)

\[ depthScore_{depth} = \log_{10}(treeHeight - depth) + 1 \]

\[ weight_i = \frac{depthScore_i}{\sum_{j=0}^{treeH} depthScore_i \cdot |levels_i|} \]
Comparison (algorithm: similarity computation)

\[
sim(inst_1, inst_2) = \begin{cases} 
  simLI(root_1, root_2) \cdot \text{weight}_0 \\ 
  + \sum_{i=1}^{n} simL(subL_1,i, subL_2,i, 1) \\ 
  0 
\end{cases} 
\]

\[
\text{Similarity} = \text{simLI}(root_1, root_2) \cdot \text{weight}_0 + \text{simL}(L_1, L_3) + \text{simL}(L_2, L_4) + \ldots
\]
Comparison (algorithm: similarity computation)

\[
sim(inst_1, inst_2) = \begin{cases} 
    simLI(root_1, root_2) \cdot \text{weight}_0 \\
    + \sum_{i=1}^{n} simL(subL_{1,i}, subL_{2,i}, 1) & \text{se } simLI(root_1, root_2) > 0 \\
    0 & \text{altrimenti}
\end{cases}
\]

Similarity = \( simLI(root_1, root_2) \cdot \text{weight}_0 + simL(L_1, L_3) + simL(L_2, L_4) + \ldots \)
Comparison (algorithm: similarity computation)

\[
sim(inst_1, inst_2) = \begin{cases} 
\text{simLI(root}_1, \text{root}_2) \cdot \text{weight}_0 \\
+ \sum_{i=1}^{n} \text{simL(subL}_1,i, \text{subL}_2,i, 1) & \text{se simLI(root}_1, \text{root}_2) > 0 \\
0 & \text{altrimenti}
\end{cases}
\]

Similarity = simLI(root_1, root_2) * weight_0 + simL(L_1, L_3) + simL(L_2, L_4) + ...
Comparison (algorithm: similarity computation)

\[
simL(l_1, l_2, \text{depth}) = \sum_{i=1}^{n} \text{simLI}(\text{subLI}_{1,i}, \text{mostSim}(\text{subLI}_{1,i}, \text{subLI}_{2,i})) \cdot \frac{\text{weight}_{\text{depth}}}{n} + \sum_{i=1}^{m} \text{simL}(\text{subL}_{1,i}, \text{subL}_{2,i}, \text{depth} + 1)
\]

Similarity = \text{simLI}(\text{root}_1, \text{root}_2) \cdot \text{weight}_0 + \text{simL}(L_1, L_3) + \text{simL}(L_2, L_4) + \ldots
\text{simL}(L_1, L_3) = \left[ \text{simLI}(\text{LI}_1, \text{LI}_5) + \text{simLI}(\text{LI}_2, \text{LI}_4) \right] \cdot \text{weight}_1 / 2
Comparison (algorithm: similarity computation)

\[
simLI(li_1, li_2) = \frac{|\text{sharedRoles}|}{\max(|\text{subRoles}_1|, |\text{subRoles}_2|)}
\]

Similarity = \(\text{simLI}(\text{root}_1, \text{root}_2) \times \text{weight}_0 + \text{simL}(L_1, L_3) + \text{simL}(L_2, L_4) + \ldots\)

\(\text{simL}(L_1, L_3) = [\text{simLI}(LI_1, LI_5) + \text{simLI}(LI_2, LI_4)] \times \text{weight}_1 / 2\)

\(\text{simLI}(LI_1, LI_5) = 2 / 4 = 0.5\)
Comparison (algorithm: similarity computation)

\[
sim(inst_1, inst_2) = \begin{cases} 
\text{simLI}(\text{root}_1, \text{root}_2) \cdot \text{weight}_0 \\
+ \sum_{i=1}^{n} \text{simL}(\text{subL}_{1,i}, \text{subL}_{2,i}, 1) & \text{se } \text{simLI}(\text{root}_1, \text{root}_2) > 0 \\
0 & \text{altrimenti}
\end{cases}
\]

Similarity = \text{simLI}(\text{root}_1, \text{root}_2) \cdot \text{weight}_0 + \text{simL}(L_1, L_3) + \text{simL}(L_2, L_4) + ...
Search (example)
Search (results analysis)

Comparison for the same context

Comparison respect to patterns
Collaboration, beta-testing and feedback

Günter Kniesel and Alex Binun (Universität Bonn, Germany)

Nikos Tsantalis (University of Alberta, Canada)

Yann-Gaël Guéhéneuc (École Polytechnique de Montréal, Canada)
Conclusions

• A benchmark for DPD tools
  – Specific meta-model for DP representation
  – A new algorithm for DP instances comparison
  – Largely Experimented

www.essere.disco.unimib.it/DPB
Future work

• Simplify the results importing process
  – Compatibility extension for other meta-models
  – Web service for results upload

• Add statistical analyses

• Think at new interaction types
  – Eclipse plug-in
Statistics

• The platform is currently populated with:
  – 2 DPD tools (WOP and DPD-tool(Tsantalis))
  – 1 verified instances dataset (P-Mart)
  – 20+ system analysis
  – 700+ DP instances.
  – 160+ evaluations.

• There are 36 registered users.

• Access statistics:
  – 900+ visits e 360 unique users.
  – 13.000+ page visualization.
  – 15 minutes of average spent time on the web site
Meta-model requirements

1. Minimum effort to understand how to define a new DP instance
2. Compact representation (to make data store and elaboration faster).
4. Flexible enough to support any DP definition

• Requisiti soddisfatti
  – DPB: all 😊
  – DPDX: only 3 and 4
    - Quite big and too generic in many cases
    - Models code is not very readable
    - The lack of a shared set of Schema meta-models does not allow to make the models really interoperable
  – KDM: needs extension
  – FAMIX, Dagstuhl, Marple, other: only code representation
Principles for the definition of the specification

• Multiplicity principle: Given the level A and B, having respectively the associated roles (A1,A2, ..,An) and (B1,B2, ...,Bn), it is possible to state that B is sublevel of A if (and only if) for each instance of any role associated to level A, at least one instance exists of each role associated to level B. In other words, the multiplicity rate between the number of instances of any role Ai (belonging to A) and any role Bj (belonging to B) is always 1:1 or 1:many.

• Coupling principle: Two roles A1 and A2 are associated to the same level, if every time an element playing a role A1 is present it is possible to observe one and only one element playing role A2.
## Technologies

<table>
<thead>
<tr>
<th>Apache</th>
<th>Glassfish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PHP</strong></td>
<td><strong>JSF</strong></td>
</tr>
<tr>
<td><strong>Source Code Visualization</strong></td>
<td><strong>EJB</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Search</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Compare</strong></td>
</tr>
<tr>
<td><strong>Source code files</strong></td>
<td><strong>DP instances</strong></td>
</tr>
</tbody>
</table>

**Presentation**

**Business Logic**

**Data Access**

**Data Store**

DPB: A Benchmark for Design Pattern Detection tools
Example: DP instance scoring

- evaluations:
  - evaluation 1: **4 stars** (3 agreements / 1 disagreement)
  - evaluation 2: **3 stars** (8 agreements / 0 agreements)
  - evaluation 3: **1 stars** (0 agreements / 8 agreements)
  - evaluation 4: **4 stars** (1 agreement / 4 disagreements)

- formula applications brings these results:
  - \( \text{votesBalance}_1 = 3 + 3 - 1 = +5 \)
  - \( \text{votesBalance}_2 = 3 + 8 - 0 = +11 \)
  - \( \text{votesBalance}_3 = 3 + 0 - 8 = -5 \) \(< 0, \Rightarrow \text{votesBalance}_3 = 0 \)
  - \( \text{votesBalance}_4 = 3 + 1 - 4 = 0 \) \(< 0, \Rightarrow \text{votesBalance}_4 = 0 \)

- Result:
  - \( \text{rating}(\text{instance}) = (4 \times 5 + 3 \times 11 + 1 \times 0 + 4 \times 0) / (5 + 11 + 0 + 0) \)
    = \( (20 + 33) / 16 \)
    = \( 3.31 \)
Online examples

- **System analysis:**

- **Instance:**

- **Search:**
  - Java – JHotDraw+QuickUML – DPD+WOP – AbstractFactory+Adapter+Bridge

- **Comparison:**
  - JHotDraw - #4 - #41 – Strategy
  - 64%

- **Definition:**

- **Browse:**
  - [http://essere.disco.unimib.it:8080/DPBWeb/faces/Browse.jsp](http://essere.disco.unimib.it:8080/DPBWeb/faces/Browse.jsp)
Similarity algorithm – Example

DPB: A Benchmark for Design Pattern Detection tools
Calculate weights (based on definition’s structure; see slide 4):

\[
\text{depthScore}_0: \log_{10}(3-0)+1 = 1.48
\]
\[
\text{depthScore}_1: \log_{10}(3-1)+1 = 1.3
\]
\[
\text{depthScore}_2: \log_{10}(3-2)+1 = 1
\]

Sum(depthScore_i * numLevels_i) = 1.48 * 1 + 1.3 * 2 + 1 * 1 = 5.08

weight_0 = 1.48/5.08 = 0.29
weight_1 = 1.3/5.08 = 0.26
weight_2 = 1/5.08 = 0.20
similarity = 1 * weight_0 + simL(L_AP1, L_AP1', 1) + simL(L_CF1, L_CF1', 1)

simL(L_AP1, L_AP10, 1) = (simL(LI_1, LI_1') + simL(LI_2, LI_2') + simL(LI_3, null)) * weight_1 / 3 + (simL(L_CP1, L_CP1', 2) + simL(L_CP2, L_CP2', 2) + simL(L_CP3, null, 2))

simL(L_CP1, L_CP1', 2) = (simL(LI_4, LI_4') + simL(LI_5, LI_5')) * weight_2 / 2 = (1+1) * 0.2 / 2 = 0.2

simL(L_CP2, L_CP2', 2) = (simL(LI_6, LI_9)) * weight_2 / 1 = 0 * 0.2 = 0

simL(L_CP3, null, 2) = 0

simL(L_CF1, L_CF1', 1) = simL(LI_8, LI_10) * weight_1 / 1
similarity = 1 * weight_0 + \( \text{simL}(L_{AP1}, L_{AP1}', 1) \) + \( \text{simL}(L_{CF1}, L_{CF1}', 1) \)

\[
\text{simL}(L_{AP1}, L_{AP10}, 1) = (1+ 1 + 0) * 0.26 / 3 + (0.2 + 0 + 0) = \boxed{0.37}
\]

\[
\text{simL}(L_{CP1}, L_{CP1}', 2) = (\text{simLI}(L_{4}, L_{4}') + \text{simLI}(L_{5}, L_{5}')) * \text{weight}_2 / 2 = (1+1) * 0.2 / 2 = \boxed{0.2}
\]

\[
\text{simL}(L_{CP2}, L_{CP2}', 2) = (\text{simLI}(L_{6}, L_{9})) * \text{weight}_2 / 1 = 0 * 0.2 = \boxed{0}
\]

\[
\text{simL}(L_{CP3}, \text{null}, 2) = 0
\]

\[
\text{simL}(L_{CF1}, L_{CF1}', 1) = 0 * 0.26 / 1 = \boxed{0}
\]
similarity = 1 * 0.29 + 0.37 + 0 = 0.66  =>  66%

\[ \text{simL}(L_{\text{AP1}}, L_{\text{AP10}}, 1) = (1+ 1 + 0) * 0.26 / 3 + (0.2 + 0 + 0) = 0.37 \]
\[ \text{simL}(L_{\text{CP1}}, L_{\text{CP1}'}, 2) = (\text{simL}(L_{\text{I4}}, L_{\text{I4}'}) + \text{simL}(L_{\text{I5}}, L_{\text{I5}'})) * \text{weight}_2 / 2 = (1+1) * 0.2 / 2 = 0.2 \]
\[ \text{simL}(L_{\text{CP2}}, L_{\text{CP2}'}, 2) = (\text{simL}(L_{\text{I6}}, L_{\text{I9}})) * \text{weight}_2 / 1 = 0 * 0.2 = 0 \]
\[ \text{simL}(L_{\text{CP3}}, \text{null}, 2) = 0 \]
\[ \text{simL}(L_{\text{CF1}}, L_{\text{CF1}'}, 1) = 0 * 0.26 / 1 = 0 \]