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]

  x Usability

  x Data model

  ✓ Open web application

  ✓ Interesting choice of functionalities

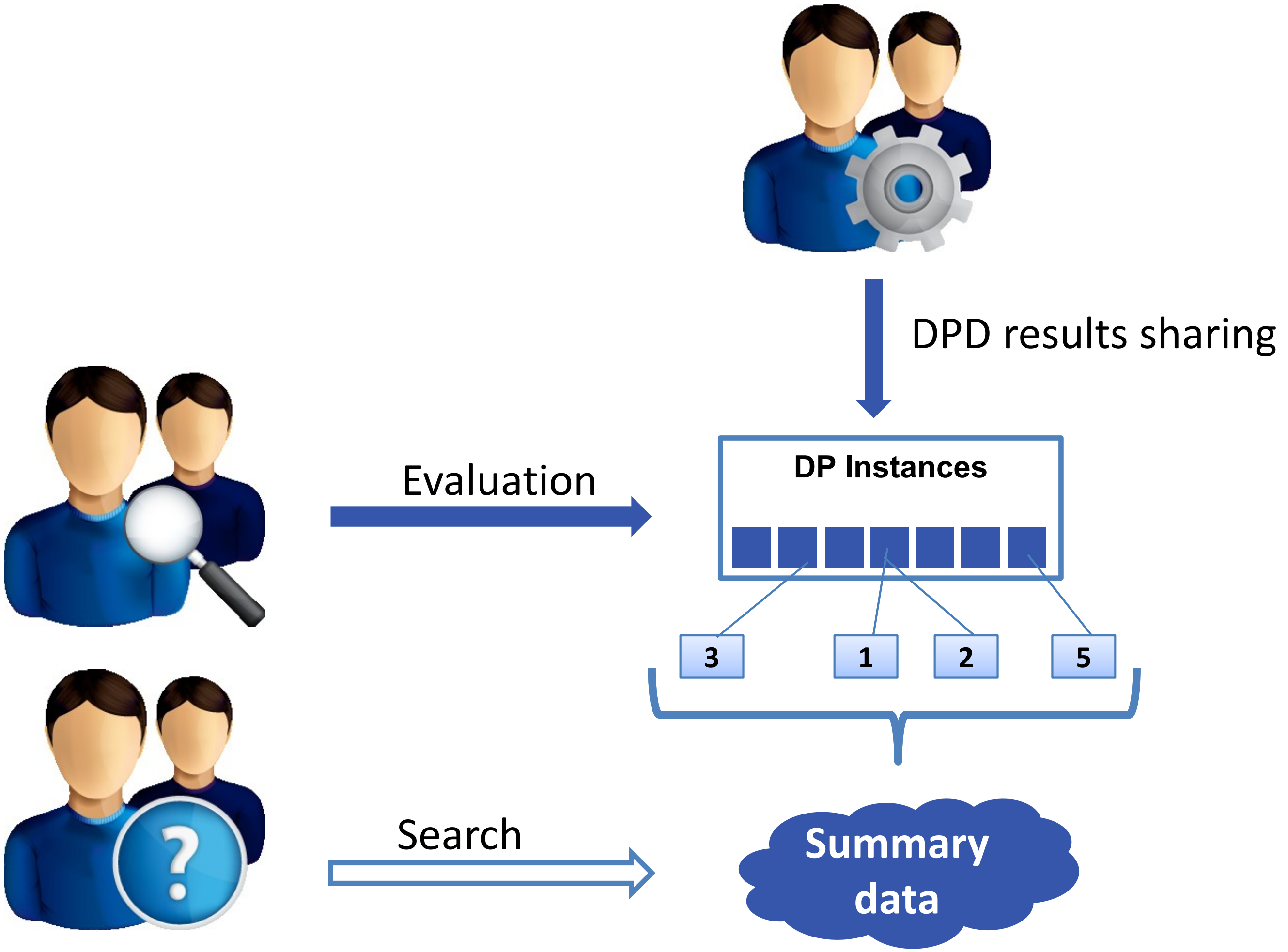
- P-MARt [Guéhéneuc, 2007]

  x No support for discussion

  x No way to measure reliability

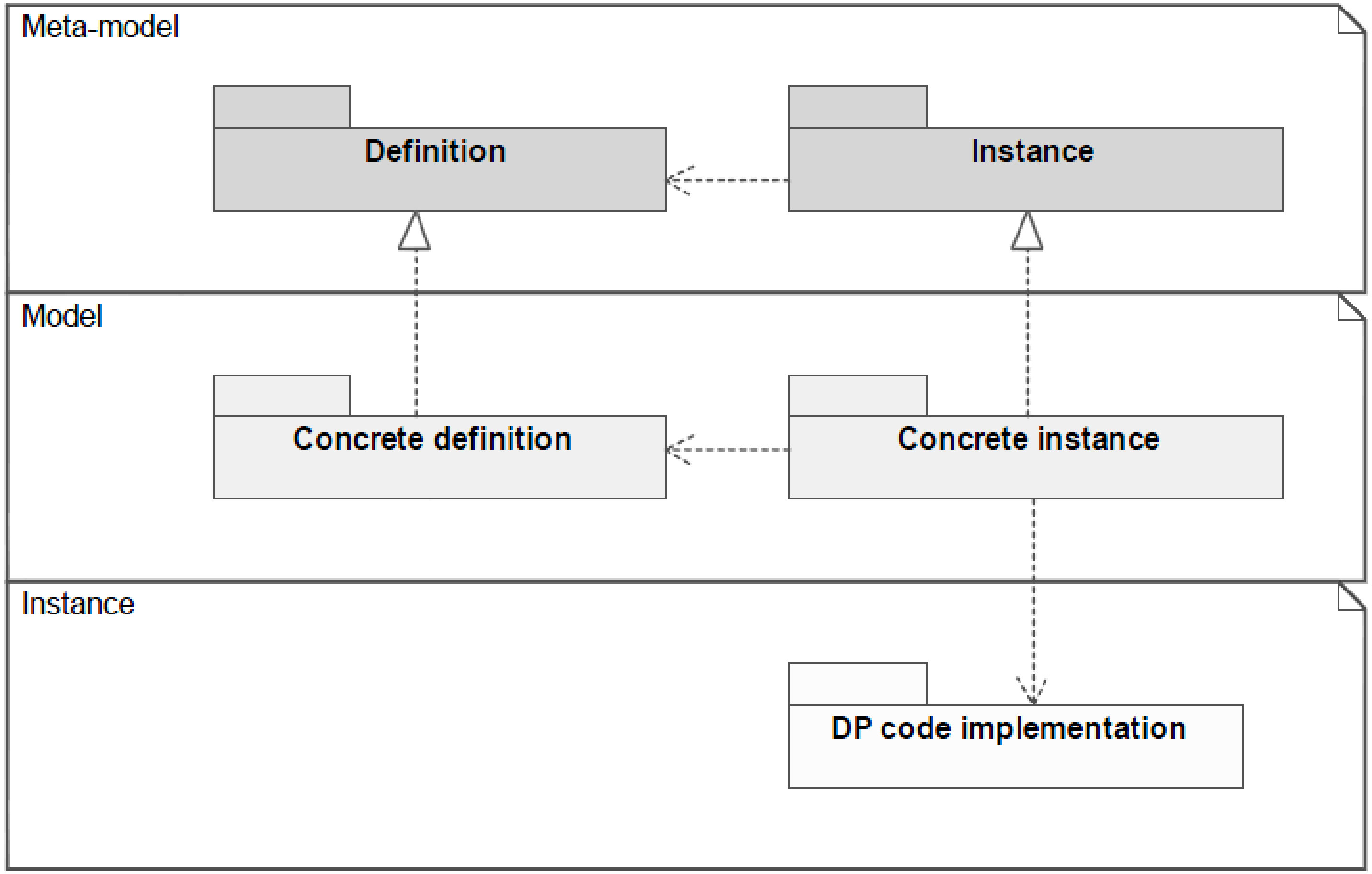
  ✓ Pattern instances identified by experts

Proposed Solution

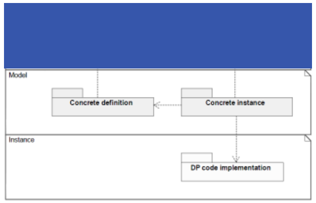


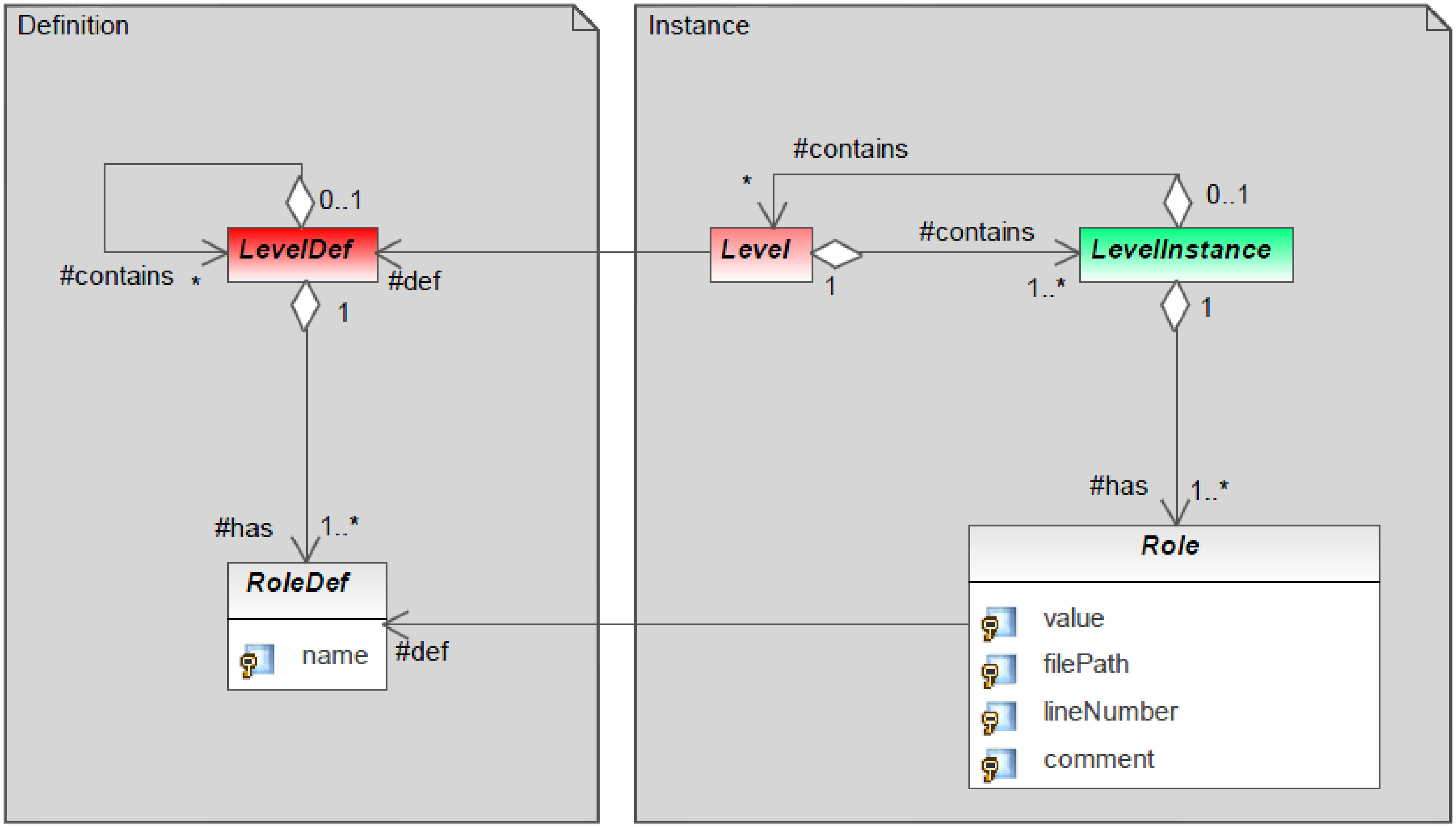
1. Representation
2. Analysis and Evaluation
3. Comparison
4. Search

Representation (panoramic)

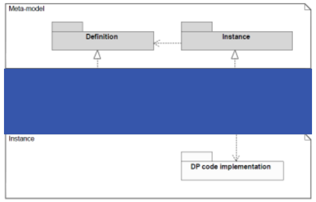


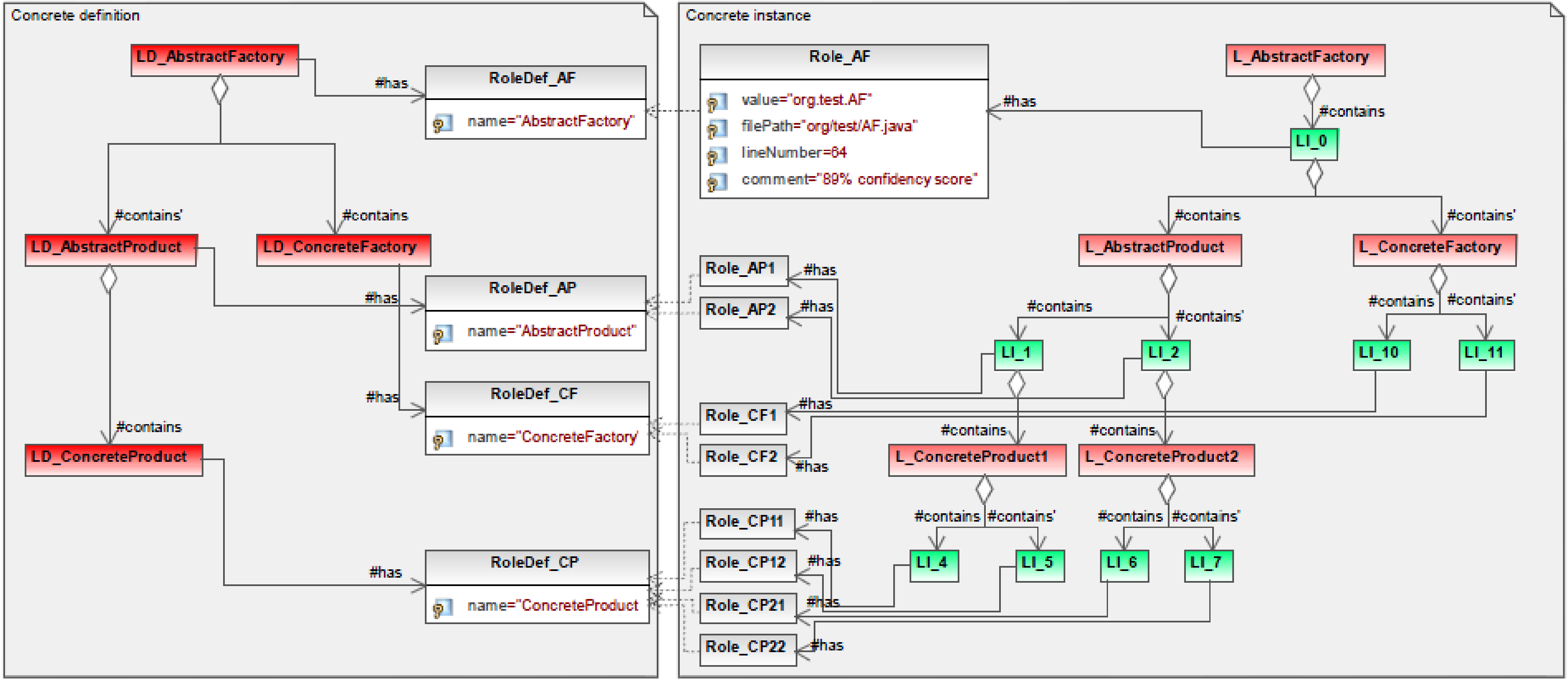
Representation (meta-model)



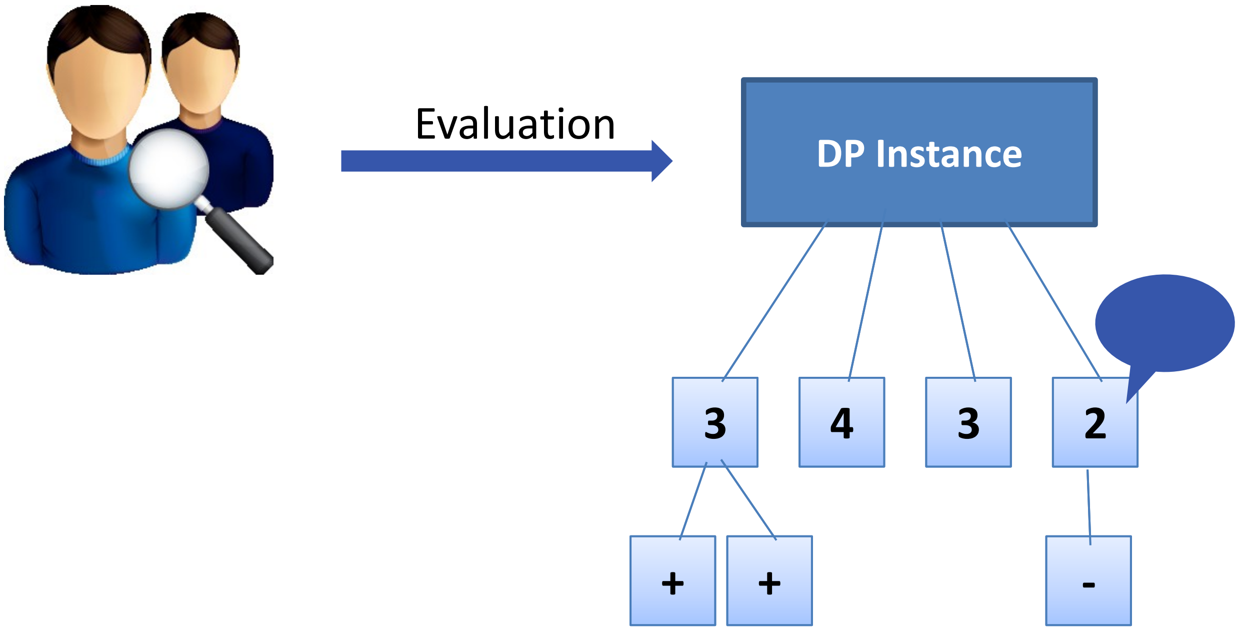


Representation (model)





Analysis and Evaluation



Analysis

Discussion

Evaluation 1 – 5
With comment

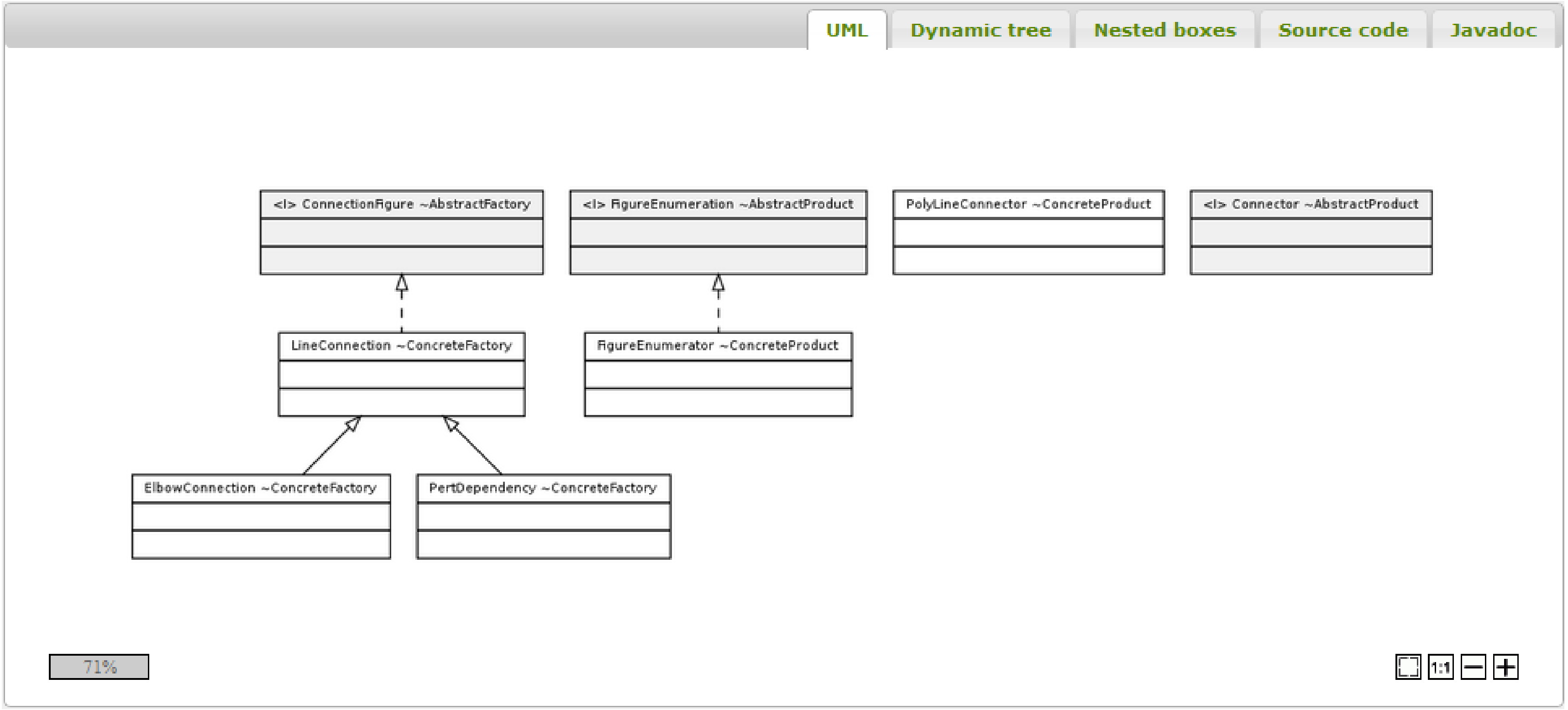
Vote

Instance score

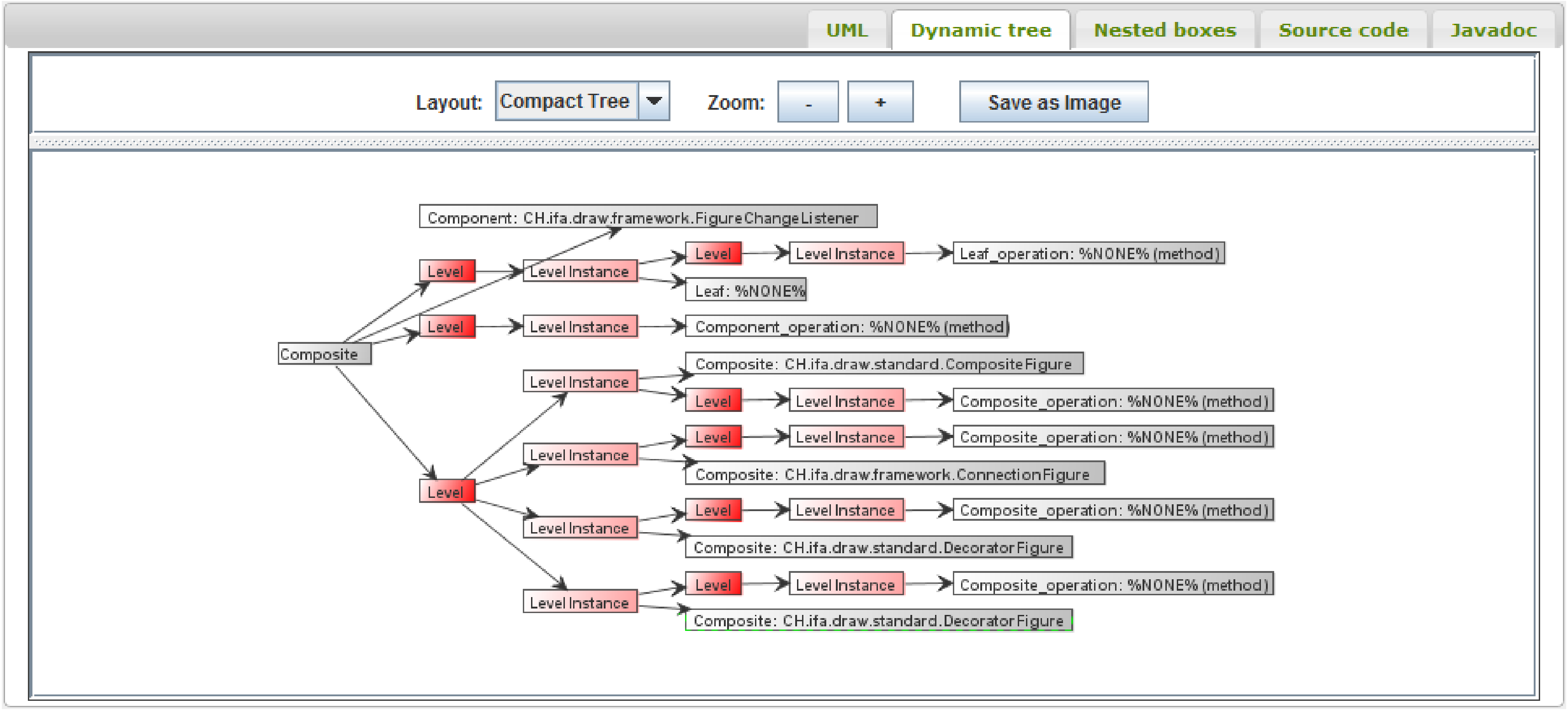
$$rating(instance) = \frac{\sum_{i=1}^{|evals|} eval_i \cdot votesBalance_i}{\sum_{i=1}^{|evals|} votesBalance_i}$$

$$votesBalance_i = max(def + votes_i^{+} - votes_i^{-}, 0)$$

Analysis and Evaluation (UML diagram)


UML
Dynamic tree
Nested boxes
Source code
Javadoc
<I> ConnectionFigure ~AbstractFactory
<I> FigureEnumeration ~AbstractProduct
PolyLineConnector ~ConcreteProduct
<I> Connector ~AbstractProduct
LineConnection ~ConcreteFactory
FigureEnumerator ~ConcreteProduct
ElbowConnection ~ConcreteFactory
PertDependency ~ConcreteFactory
71%
1:1

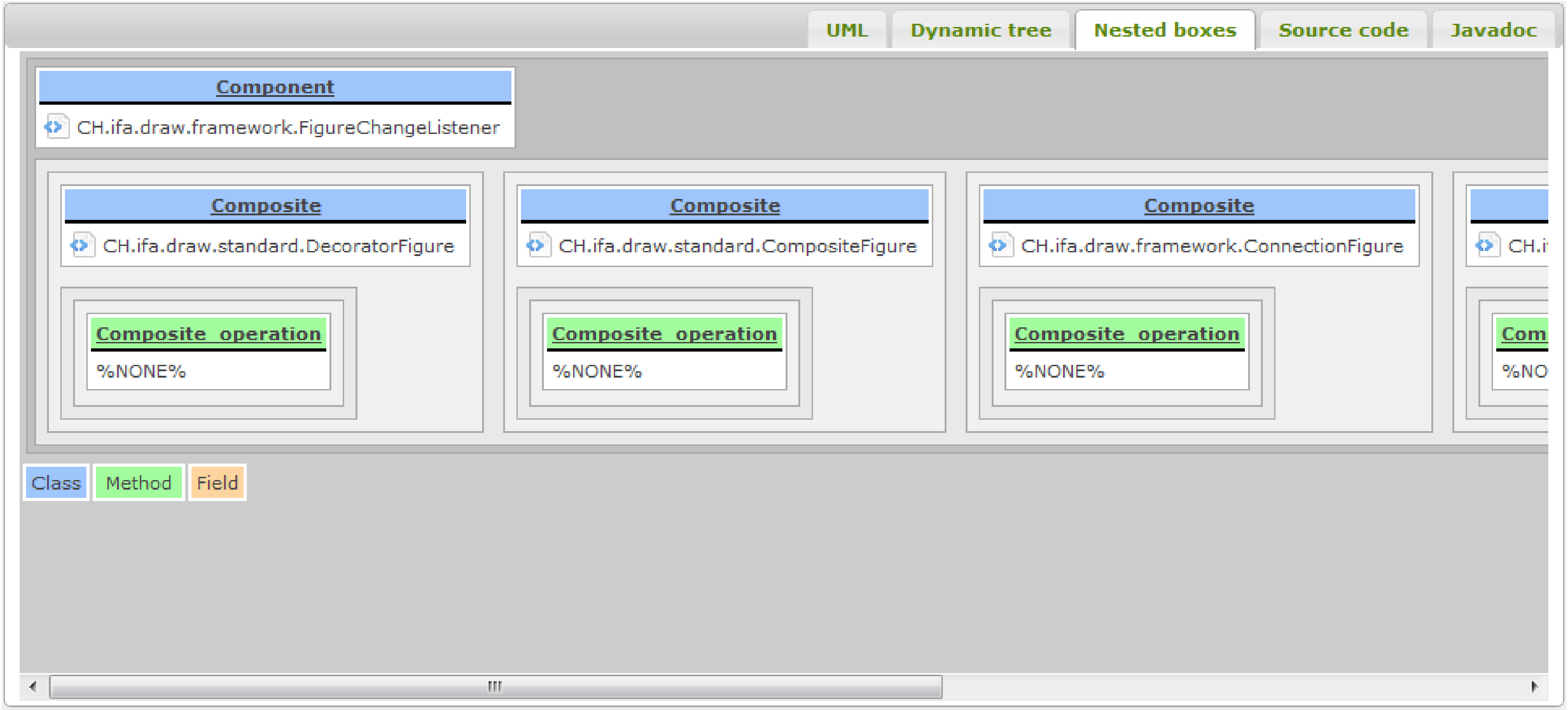
Analysis and Evaluation (structural view)



Layout: Compact Tree

UML
Dynamic tree
Nested boxes
Source code
Javadoc
Zoom:
-
+
Save as Image
Component: CH.ifa.draw.framework.FigureChangeListener
Level
Level Instance
Level
Level Instance
Leaf_operation: %NONE% (method)
Leaf: %NONE%
Level
Level Instance
Component_operation: %NONE% (method)
Composite
Composite: CH.ifa.draw.standard.CompositeFigure
Level Instance
Level
Level Instance
Composite_operation: %NONE% (method)
Level
Level Instance
Composite_operation: %NONE% (method)
Level Instance
Composite: CH.ifa.draw.framework.ConnectionFigure
Level
Level
Level Instance
Composite_operation: %NONE% (method)
Level Instance
Composite: CH.ifa.draw.standard.DecoratorFigure
Level
Level Instance
Composite_operation: %NONE% (method)
Level Instance
Composite: CH.ifa.draw.standard.DecoratorFigure

Analysis and Evaluation (structural view 2)



Component
CH.ifa.draw.framework.FigureChangeListener

Composite
CH.ifa.draw.standard.DecoratorFigure

Composite_operation
%NONE%

Composite
CH.ifa.draw.standard.CompositeFigure

Composite_operation
%NONE%

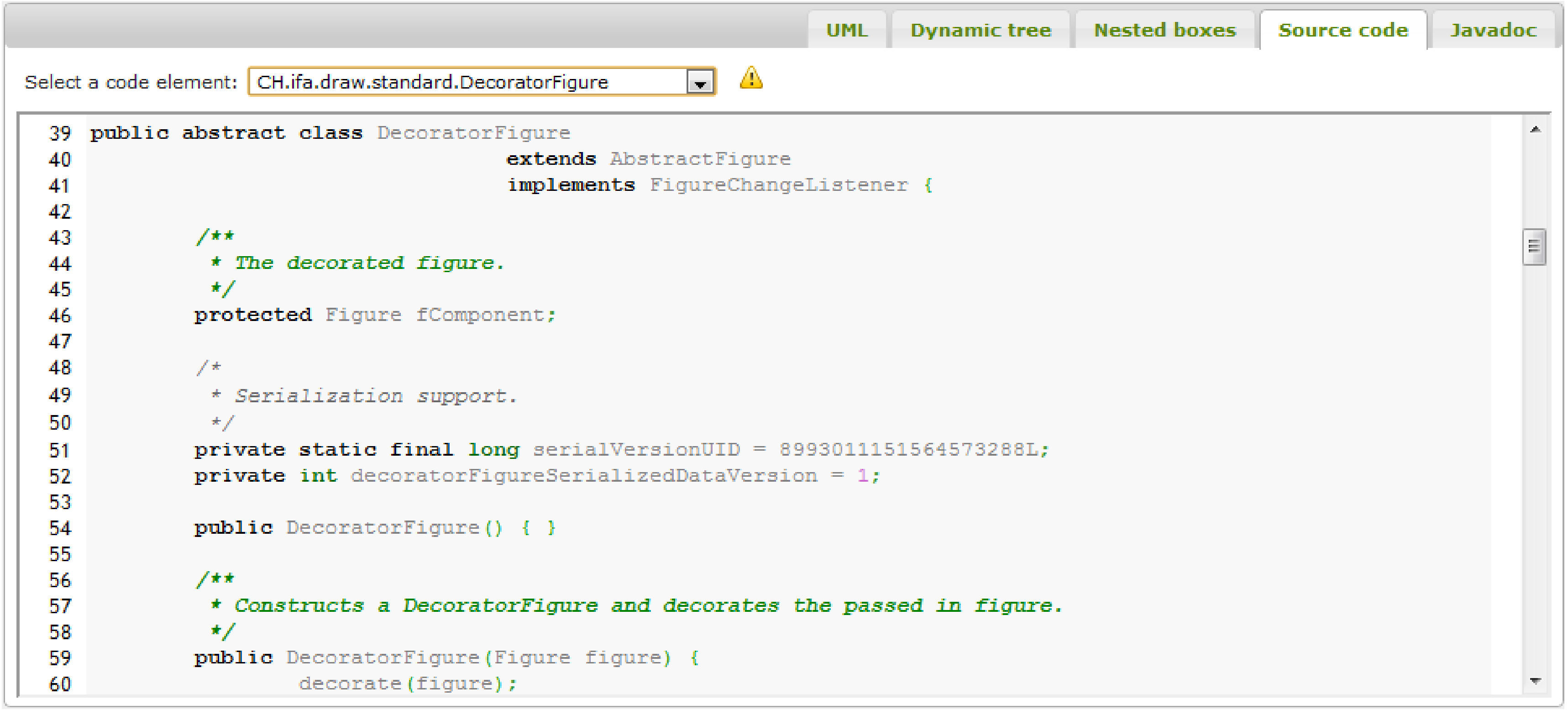
Composite
CH.ifa.draw.framework.ConnectionFigure

Composite_operation
%NONE%

CH.i

Com
%NO

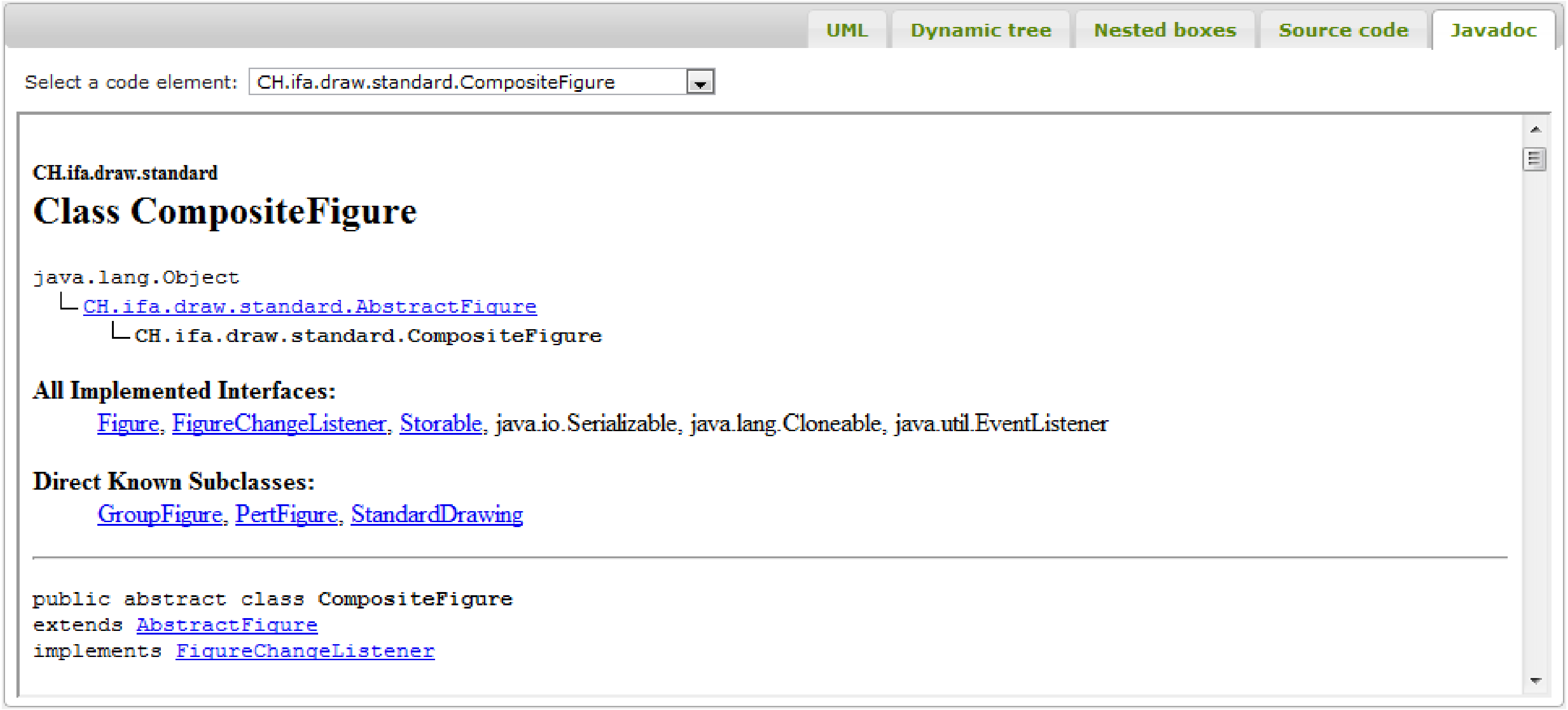
Analysis and Evaluation (source code)



Select a code element: CH.ifa.draw.standard.DecoratorFigure

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

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

        /*
         * Serialization support.
         */
        private static final long serialVersionUID = 8993011151564573288L;
        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)



Select a code element: CH.ifa.draw.standard.CompositeFigure

CH.ifa.draw.standard

Class CompositeFigure

```
java.lang.Object
  └CH.ifa.draw.standard.AbstractFigure
      └CH.ifa.draw.standard.CompositeFigure
```

All Implemented Interfaces:

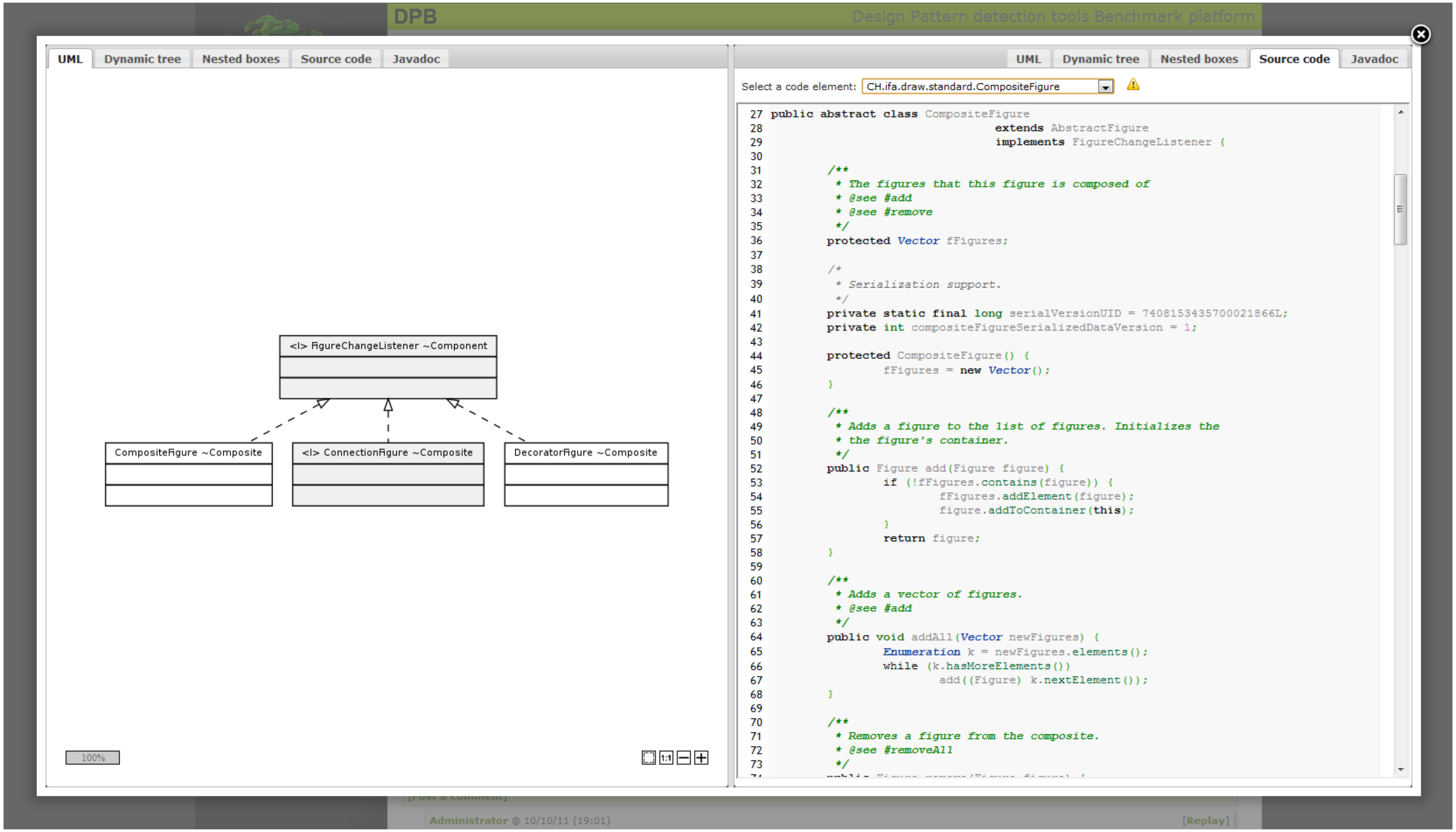
Figure, FigureChangeListener, Storable, java.io.Serializable, java.lang.Cloneable, java.util.EventListener

Direct Known Subclasses:

GroupFigure, PertFigure, StandardDrawing

```
public abstract class CompositeFigure
extends AbstractFigure
implements FigureChangeListener
```

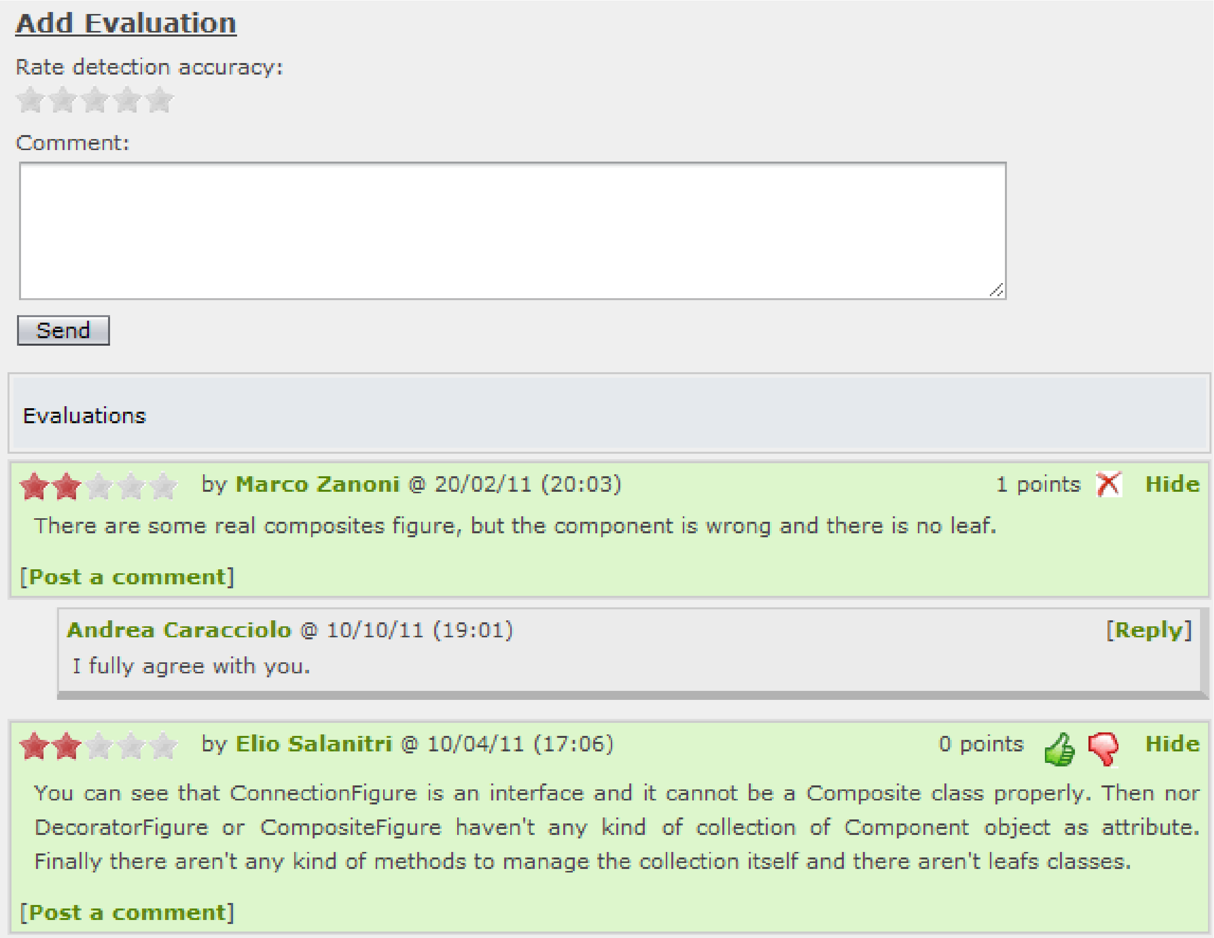
Analysis and Evalutation (side by side view)


DPB
Design Pattern detection tools Benchmark platform
UML
Dynamic tree
Nested boxes
Source code
Javadoc
<I> FigureChangeListener ~Component
CompositeFigure ~Composite
<I> ConnectionFigure ~Composite
DecoratorFigure ~Composite
100%
1:1
UML
Dynamic tree
Nested boxes
Source code
Javadoc
[Post a comment]
Administrator @ 10/10/11 (19:01)
[Replay]

Select a code element: CH.ifa.draw.standard.CompositeFigure

```
public abstract class CompositeFigure
                                extends AbstractFigure
                                implements FigureChangeListener {

        /**
         * The figures that this figure is composed of
         * @see #add
         * @see #remove
         */
        protected Vector fFigures;

        /*
         * Serialization support.
         */
        private static final long serialVersionUID = 7408153435700021866L;
        private int compositeFigureSerializedDataVersion = 1;

        protected CompositeFigure() {
                fFigures = new Vector();
        }

        /**
         * Adds a figure to the list of figures. Initializes the
         * the figure's container.
         */
        public Figure add(Figure figure) {
                if (!fFigures.contains(figure)) {
                        fFigures.addElement(figure);
                        figure.addToContainer(this);
                }
                return figure;
        }

        /**
         * Adds a vector of figures.
         * @see #add
         */
        public void addAll(Vector newFigures) {
                Enumeration k = newFigures.elements();
                while (k.hasMoreElements())
                        add((Figure) k.nextElement());
        }

        /**
         * Removes a figure from the composite.
         * @see #removeAll
         */
        public Figure remove(Figure figure) {
```

Analysis and Evaluation (evaluations)



Rate detection accuracy:

☆☆☆☆☆

Comment:

Evaluations

★★☆☆☆ by **Marco Zanoni** @ 20/02/11 (20:03) 1 points **Hide**

There are some real composites figure, but the component is wrong and there is no leaf.

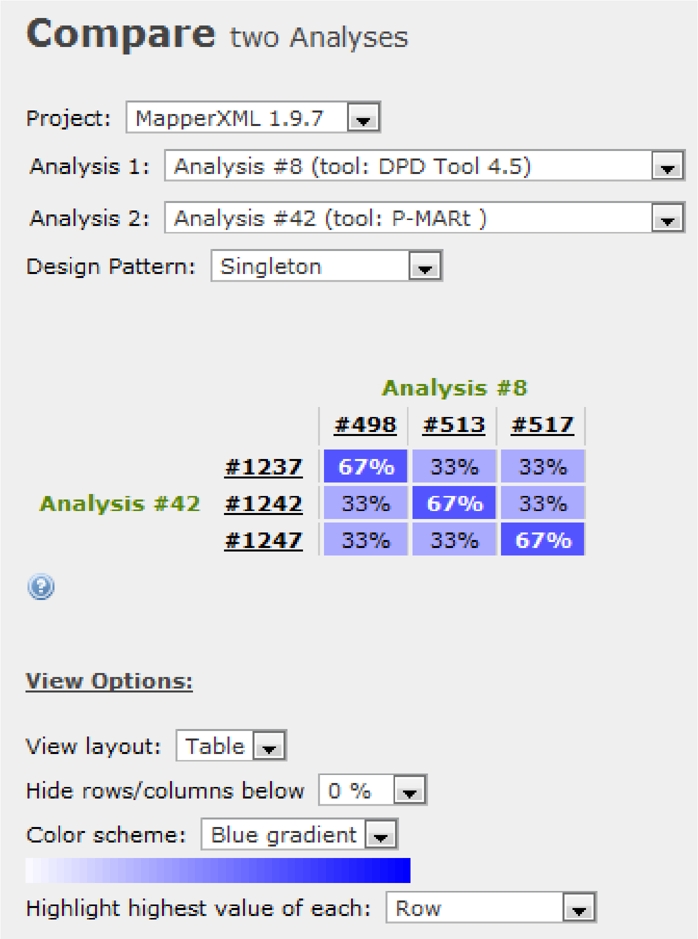
Andrea Caracciolo @ 10/10/11 (19:01) **[Reply]**

I fully agree with you.

★★☆☆☆ by **Elio Salanitri** @ 10/04/11 (17:06) 0 points **Hide**

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.

Comparison (system analysis comparison)



Compare two Analyses

Project: MapperXML 1.9.7

Analysis 1: Analysis #8 (tool: DPD Tool 4.5)

Analysis 2: Analysis #42 (tool: P-MARt)

Design Pattern: Singleton

		Analysis #8		
		#498	**#513**	**#517**
	#1237	67%	33%	33%
Analysis #42	**#1242**	33%	67%	33%
	#1247	33%	33%	67%

View Options:

View layout: Table

Hide rows/columns below 0 %

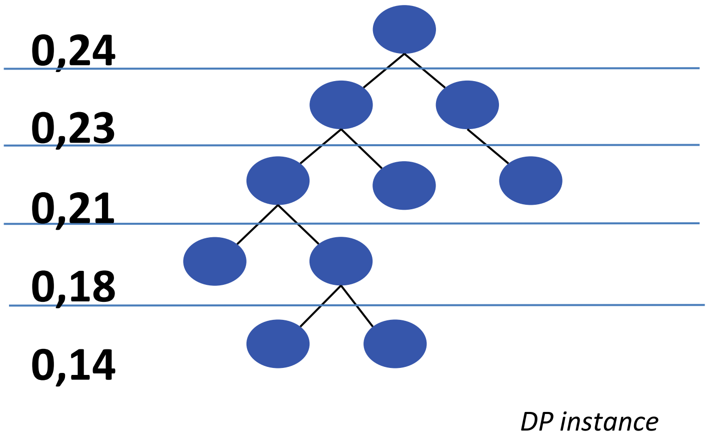
Color scheme: Blue gradient

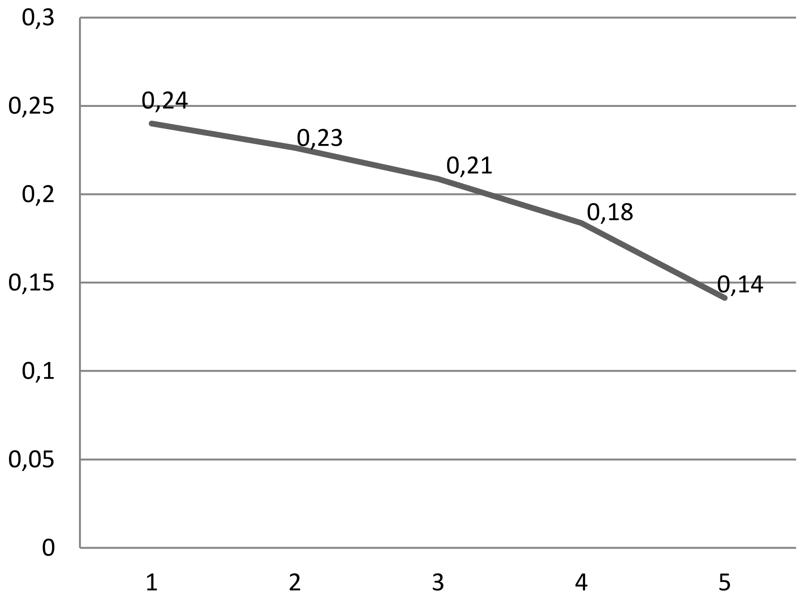
Highlight highest value of each: Row

Comparison (algorithm: weights setting)

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

$$weight_i = \frac{depthScore_i}{\sum_{j=0}^{treeH} depthScore_i \cdot |levels_i|}$$

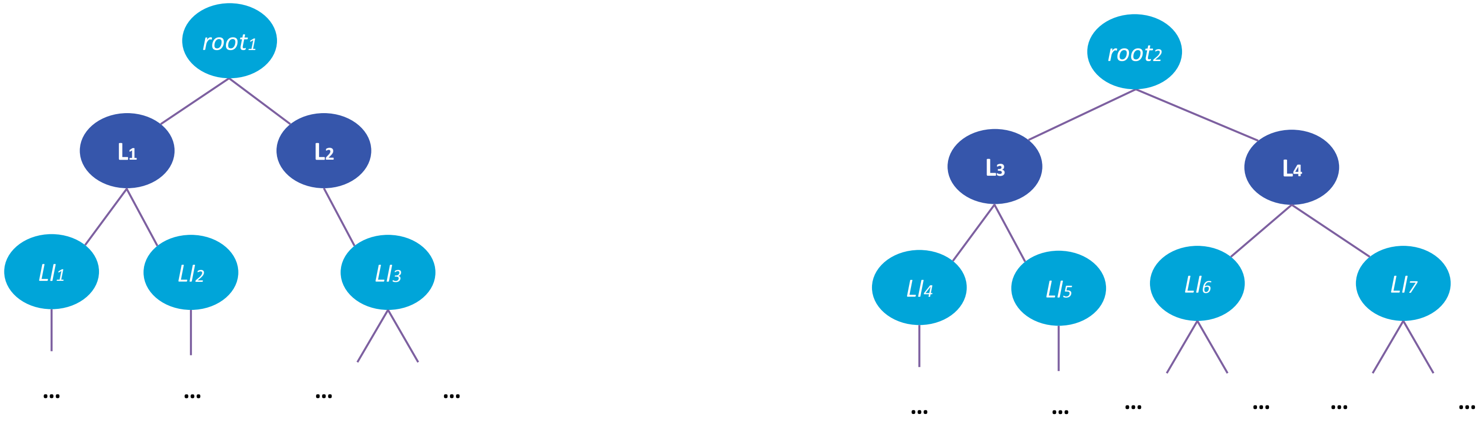




Comparison (algorithm: similarity computation)

$$sim(inst_1, inst_2) = \begin{cases} simLI(root_1, root_2) \cdot weight_0 & \\ +\sum_{i=1}^{n} simL(subL_{1,i}, subL_{2,i}, 1) & \text{se } simLI(root_1, root_2) > 0 \\ & \\ 0 & \text{altrimenti} \end{cases}$$

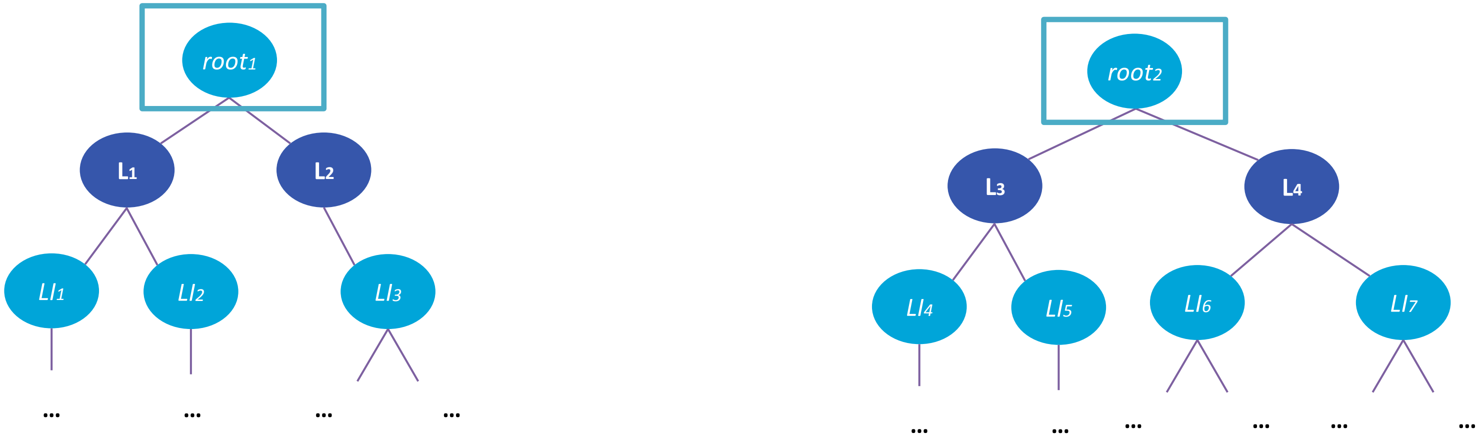
Similarity = simLI($root_1$,$root_2$) * $weight_0$ + simL(L_1, L_3) + simL(L_2, L_4) + ...



Comparison (algorithm: similarity computation)

$$sim(inst_1, inst_2) = \begin{cases} simLI(root_1, root_2) \cdot weight_0 & \\ + \sum_{i=1}^{n} simL(subL_{1,i}, subL_{2,i}, 1) & \text{se } simLI(root_1, root_2) > 0 \\ & \\ 0 & \text{altrimenti} \end{cases}$$

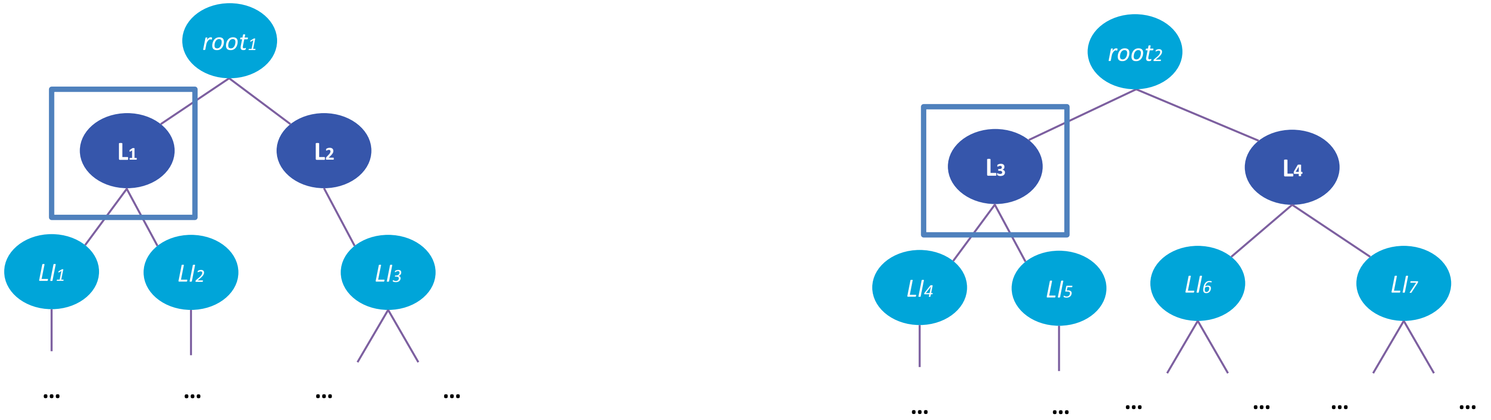
Similarity = **simLI(root$_1$,root$_2$)** * weight$_0$+ simL(L$_1$, L$_3$) + simL(L$_2$, L$_4$) + …



Comparison (algorithm: similarity computation)

$$sim(inst_1, inst_2) = \begin{cases} simLI(root_1, root_2) \cdot weight_0 & \\ + \sum_{i=1}^{n} simL(subL_{1,i}, subL_{2,i}, 1) & \text{se } simLI(root_1, root_2) > 0 \\ 0 & \text{altrimenti} \end{cases}$$

Similarity = simLI($root_1$,$root_2$) * $weight_0$+ **simL(L_1, L_3)** + simL(L_2, L_4) + …

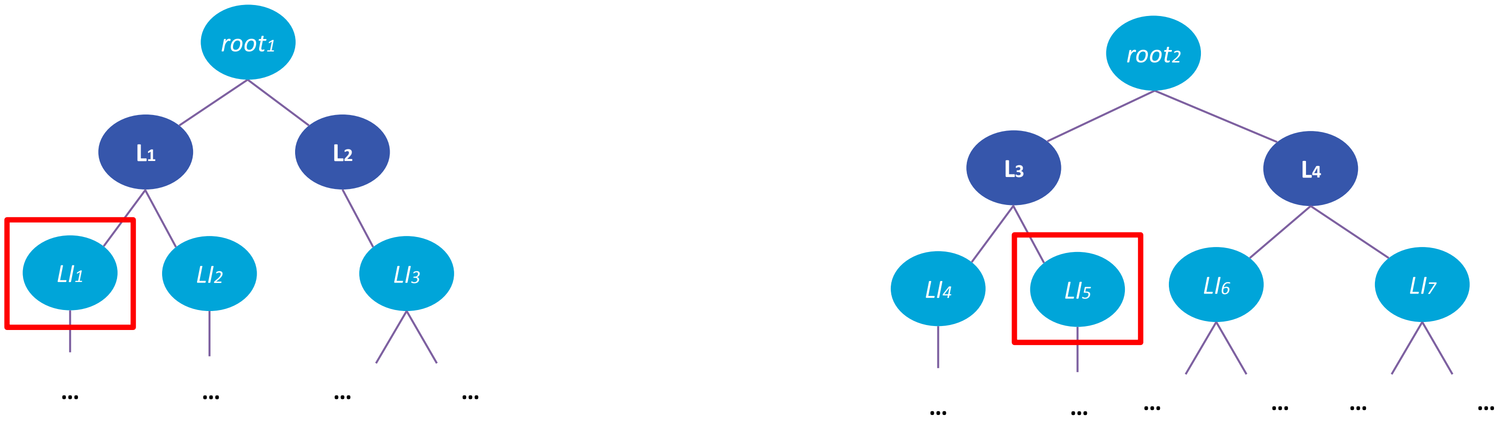


Comparison (algorithm: similarity computation)

$$simL(l_1, l_2, depth) = \sum_{i=1}^{n} simLI(subLi_{1,i}, mostSim(subLi_{1,i}, subLis_2)) \cdot \frac{weight_{depth}}{n} + \sum_{i=1}^{m} simL(subL_{1,i}, subL_{2,i}, depth + 1)$$

Similarity = simLI($root_1$,$root_2$) * $weight_0$+ **simL(L_1, L_3)** + simL(L_2, L_4) + …

simL(L_1, L_3) = [**simLI(LI_1, LI_5)** + simLI(LI_2, LI_4)] * $weight_1$ / 2



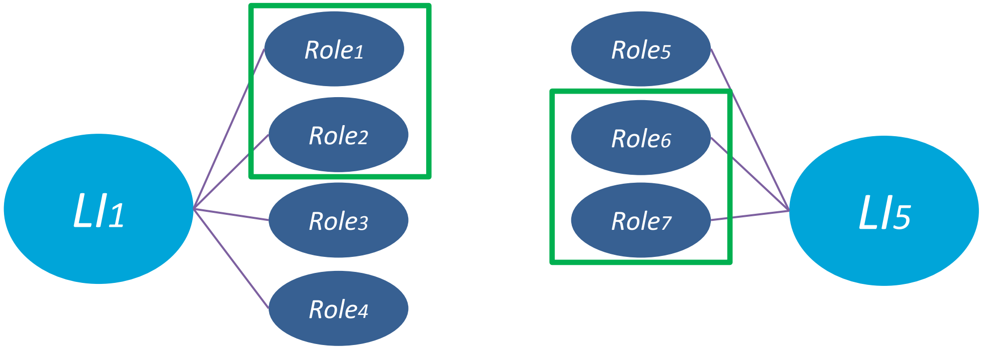
Comparison (algorithm: similarity computation)

$$simLI(li_1, li_2) = \frac{|sharedRoles|}{\max(|subRoles_1|, |subRoles_2|)}$$

Similarity = simLI(root$_1$,root$_2$) * weight$_0$ + **simL(L$_1$, L$_3$)** + simL(L$_2$, L$_4$) + …

simL(L$_1$, L$_3$) = [**simLI(LI$_1$, LI$_5$)** + simLI(LI$_2$, LI$_4$)] * weight$_1$ / 2

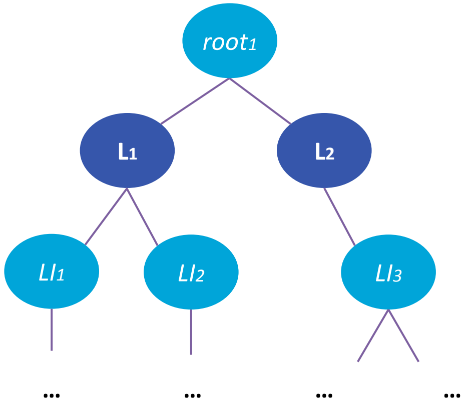
simLI(LI$_1$, LI$_5$) = 2 / 4 = **0.5**

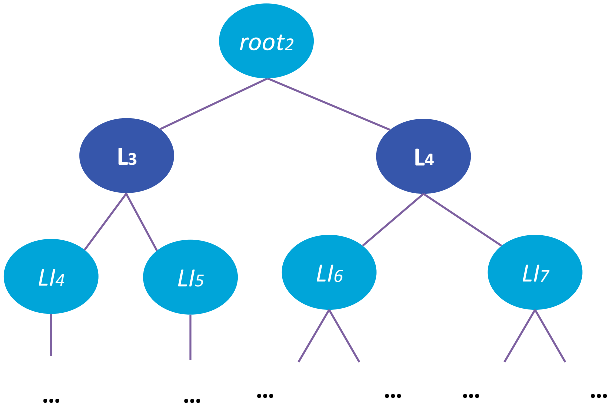


Comparison (algorithm: similarity computation)

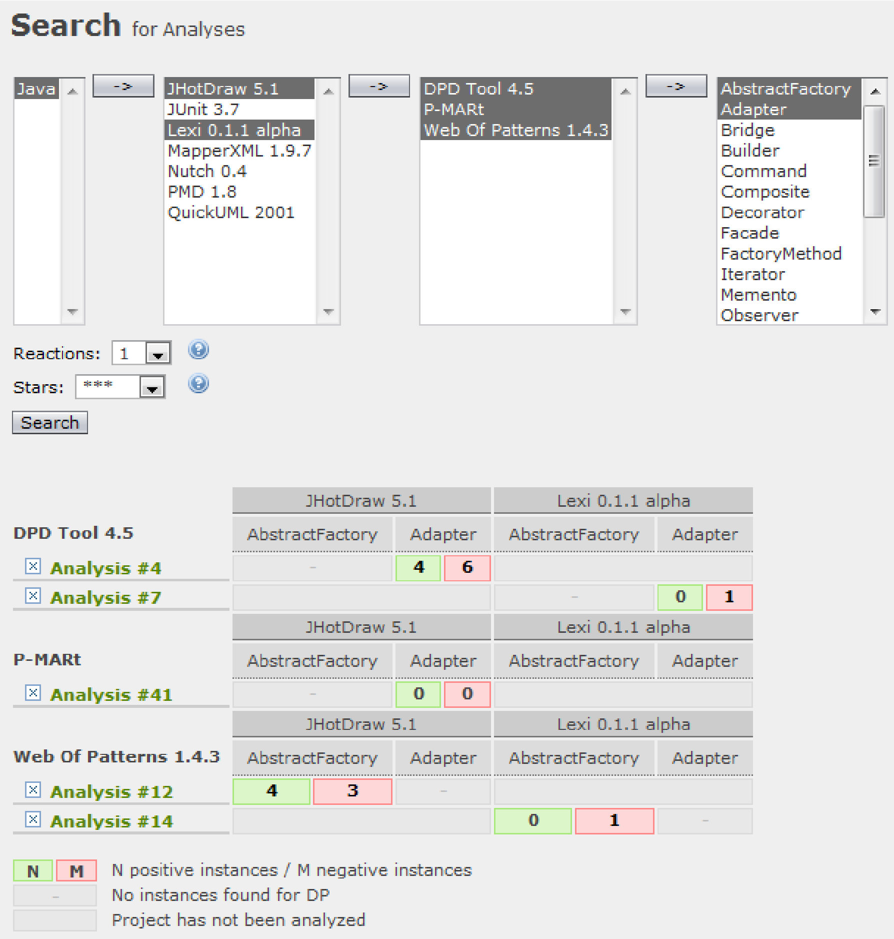
$$sim(inst_1, inst_2) = \begin{cases} simLI(root_1, root_2) \cdot weight_0 \\ + \sum_{i=1}^{n} simL(subL_{1,i}, subL_{2,i}, 1) & \text{se } simLI(root_1, root_2) > 0 \\ \\ 0 & \text{altrimenti} \end{cases}$$

Similarity = $simLI(root_1,root_2) * weight_0 + simL(L_1, L_3) + simL(L_2, L_4) + \ldots$





Search (example)



Search for Analyses

Java

->

JHotDraw 5.1
JUnit 3.7
Lexi 0.1.1 alpha
MapperXML 1.9.7
Nutch 0.4
PMD 1.8
QuickUML 2001

->

DPD Tool 4.5
P-MARt
Web Of Patterns 1.4.3

->

AbstractFactory
Adapter
Bridge
Builder
Command
Composite
Decorator
Facade
FactoryMethod
Iterator
Memento
Observer

Reactions: 1

Stars: ***

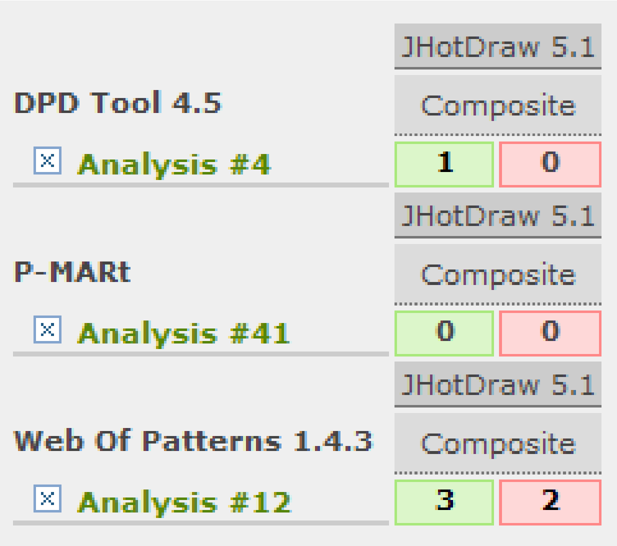
Search

	JHotDraw 5.1		Lexi 0.1.1 alpha	
DPD Tool 4.5	AbstractFactory	Adapter	AbstractFactory	Adapter
☒ **Analysis #4**	-	4 6		
☒ **Analysis #7**			-	0 1
	JHotDraw 5.1		**Lexi 0.1.1 alpha**	
P-MARt	AbstractFactory	Adapter	AbstractFactory	Adapter
☒ **Analysis #41**	-	0 0		
	JHotDraw 5.1		**Lexi 0.1.1 alpha**	
Web Of Patterns 1.4.3	AbstractFactory	Adapter	AbstractFactory	Adapter
☒ **Analysis #12**	4 3	-		
☒ **Analysis #14**			0 1	-

N M	N positive instances / M negative instances
-	No instances found for DP
	Project has not been analyzed

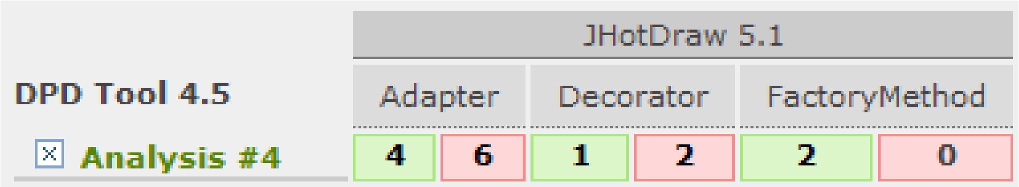
Search (results analysis)

Comparison for the same context



	JHotDraw 5.1	
DPD Tool 4.5	Composite	
☒ Analysis #4	1	0
	JHotDraw 5.1	
P-MARt	Composite	
☒ Analysis #41	0	0
	JHotDraw 5.1	
Web Of Patterns 1.4.3	Composite	
☒ Analysis #12	3	2

Comparison respect to patterns



	JHotDraw 5.1					
DPD Tool 4.5	Adapter		Decorator		FactoryMethod	
☒ Analysis #4	4	6	1	2	2	0

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

Q&A

Statistics

- The platform is currently poplated 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).
3. Support for DP instances having multi-value roles.
4. Flexible enough to support any DP definition

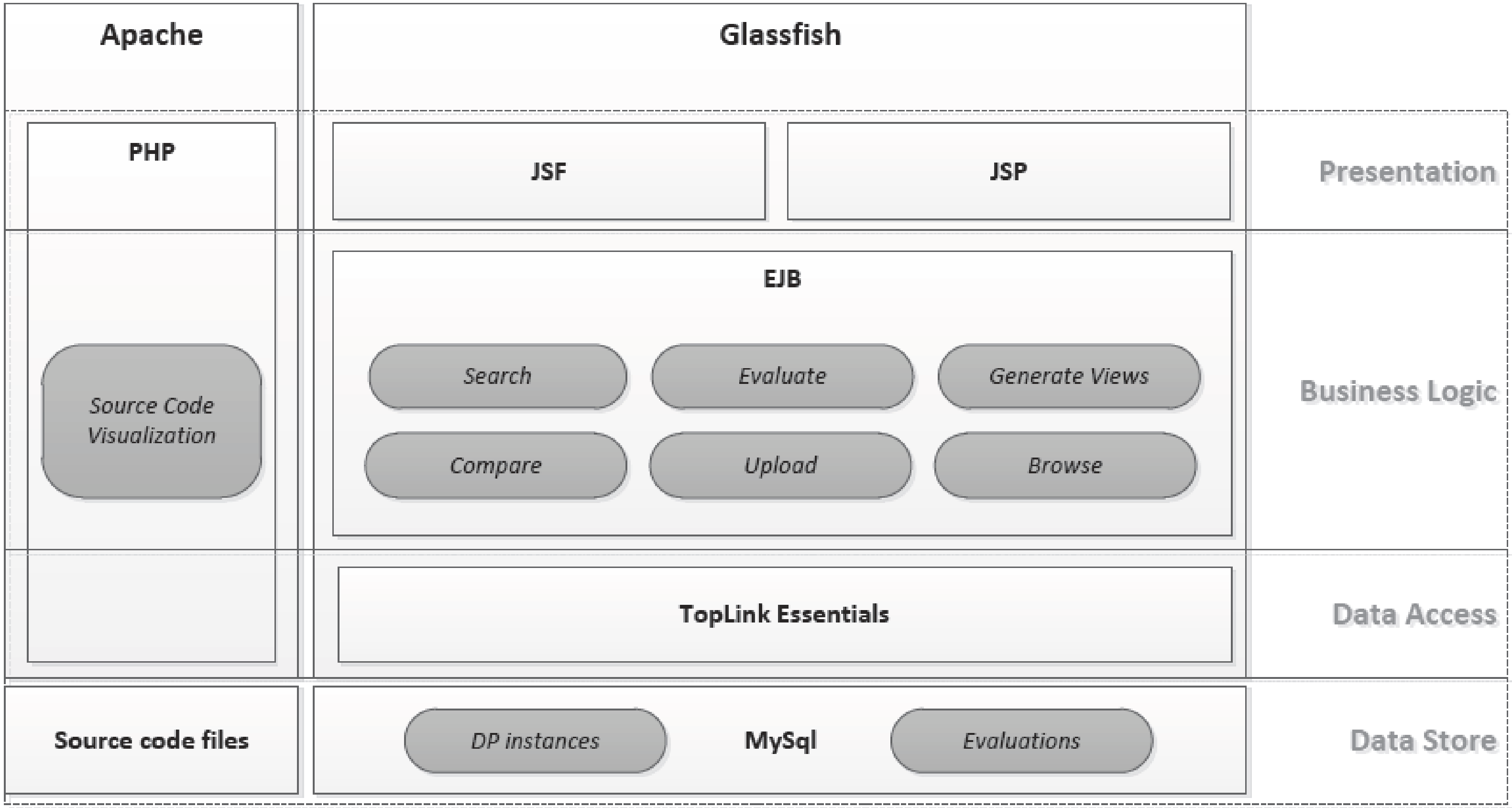
- Requisiti soddisfatti
  - DPB: all ☺
  - DPDX: only 3 and 4
    - Quite big and too generic in many cases
    - Models code is not very readable
    - The lack of a shared set of *Schema meta-models* does not allow to make the models reallly interoperable
  - KDM: needs extension
  - FAMIX, Dagstuhl,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

Principles for the definition of the specification

- Multiplicity principle: Given the level *A* and *B*, having respectively the associed 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 *A*1 and *A*2 are associated to the same levele, if every time an element playing a role *A*1 is present it is possible to observe one and only one element playing role *A*2.

Technologies



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:
  - $votesBalance1 = 3 + 3 - 1 = \mathbf{+5}$
  - $votesBalance2 = 3 + 8 - 0 = \mathbf{+11}$
  - $votesBalance3 = 3 + 0 - 8 = \mathbf{-5}$ (< 0, => $votesBalance3 = \mathbf{0}$)
  - $votesBalance4 = 3 + 1 - 4 = \mathbf{0}$ (< 0, => $votesBalance4 = \mathbf{0}$)
- Result:
  - $rating(instance) = (4 * 5 + 3 * 11 + 1 * 0 + 4 * 0) / (5 + 11 + 0 + 0)$

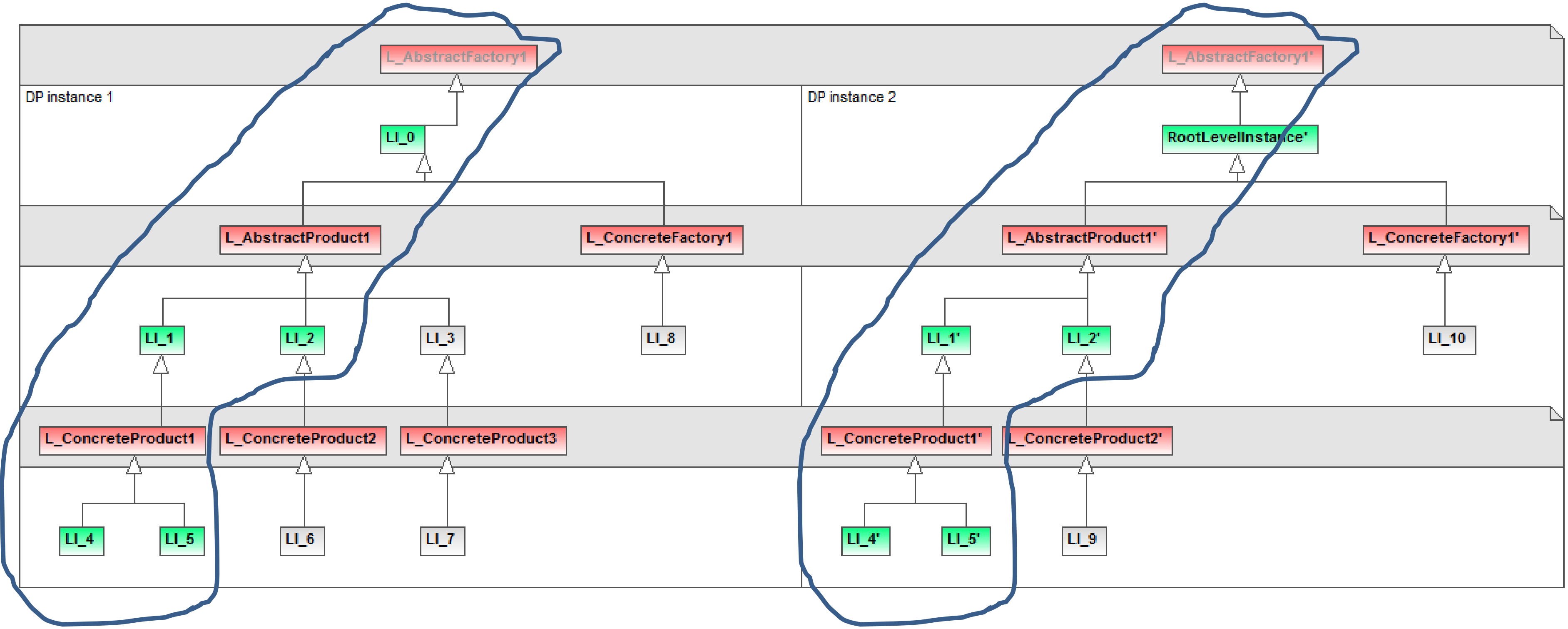
    $= (20 + 33) / 16$

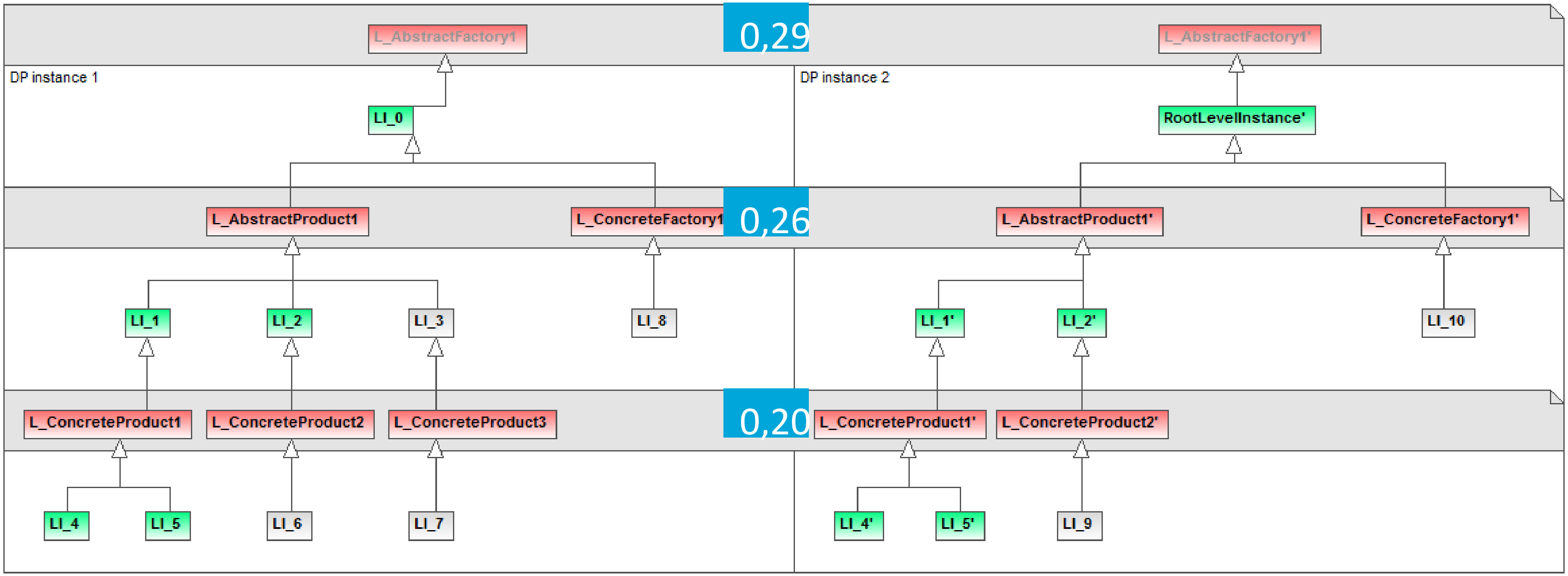
    $= \mathbf{3.31}$

Online examples

- System analysis:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/Analysis.jsp?id=12
- Instance:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/ViewDP.jsp?id=692&dpa=83
- Search:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/Search.jsp?new=1
  - Java – JHotDraw+QuickUML – DPD+WOP – AbstractFactory+Adapter+Bridge
- Comparison:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/Compare.jsp?new=1
  - JHotDraw - #4 - #41 – Strategy
  - 64%
- Definition:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/Doc_DpDef.jsp?id=28&name=AbstractFactory
- Browse:
  - http://essere.disco.unimib.it:8080/DPBWeb/faces/Browse.jsp

Similarity algorithm – Example





Calculate weights (based on definition’s structure; see slide 4):

# depthScore_0: log10(3-0)+1 = **1,48**

depthScore_1: log10(3-1)+1 = **1,3**

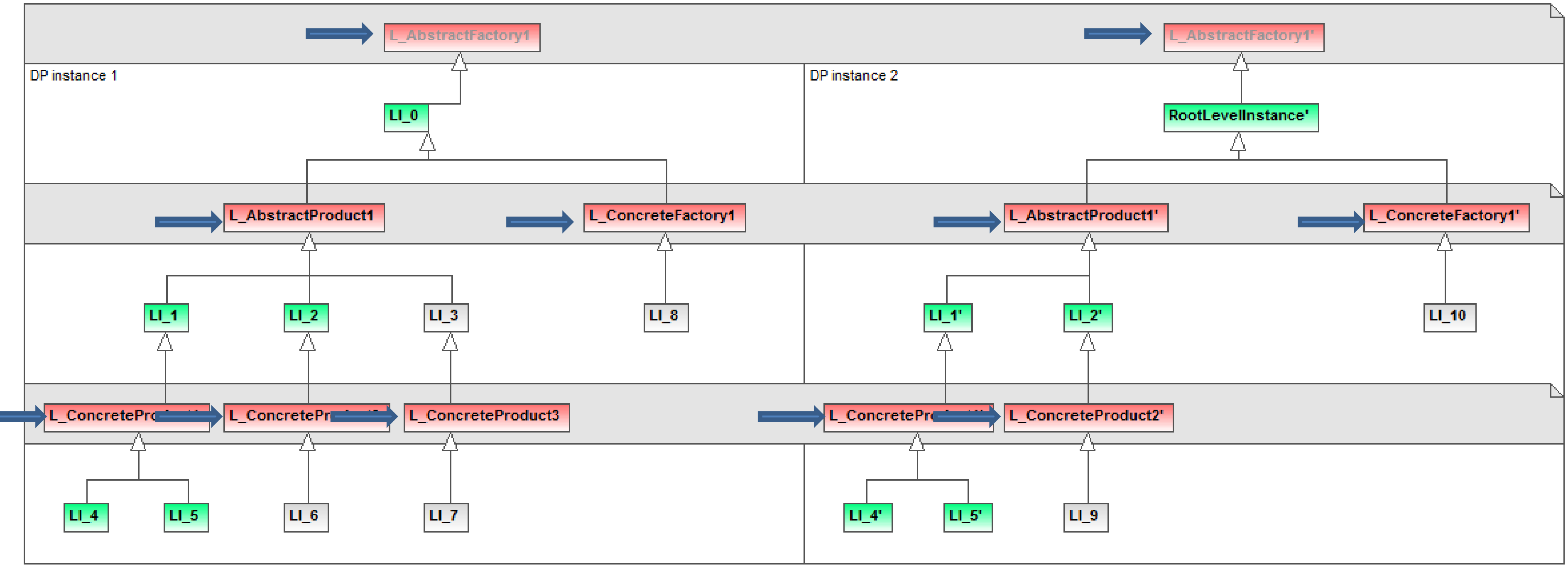
# depthScore_2: log10(3-2)+1 = **1**

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)

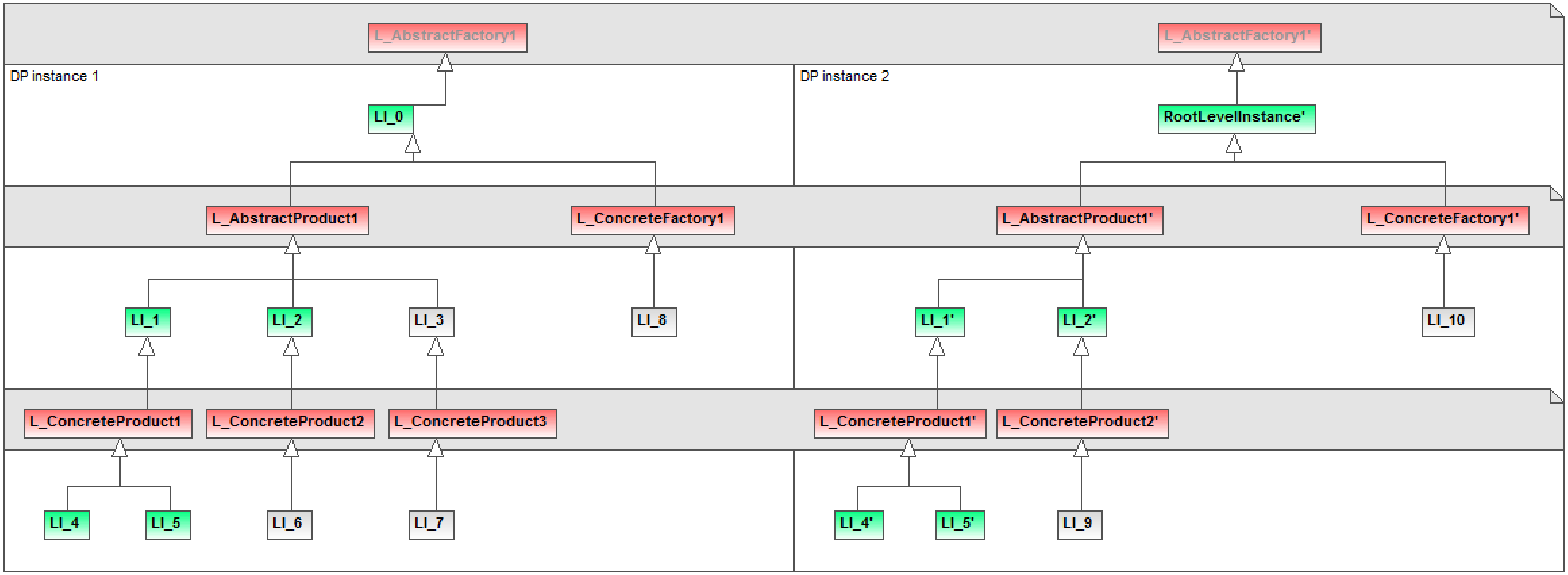
simL(L_AP1,L_AP10, 1) = (simLI(LI_1, LI_1') + simLI(LI_2, LI_2') + simLI(LI_3, null)) * weight_1 / 3 +(simL(L_CP1,L_CP1', 2) + simL(L_CP2,L_CP2', 2) + simL(L_CP3, null, 2))

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

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

simL(L_CP3, null, 2) = 0

simL(L_CF1,L_CF1',1) = simLI(LI_8,LI_10)*weight_1 / 1



similarity = 1 * weight_0 + simL(L_AP1, L_AP1',1) + simL(L_CF1, L_CF1',1)

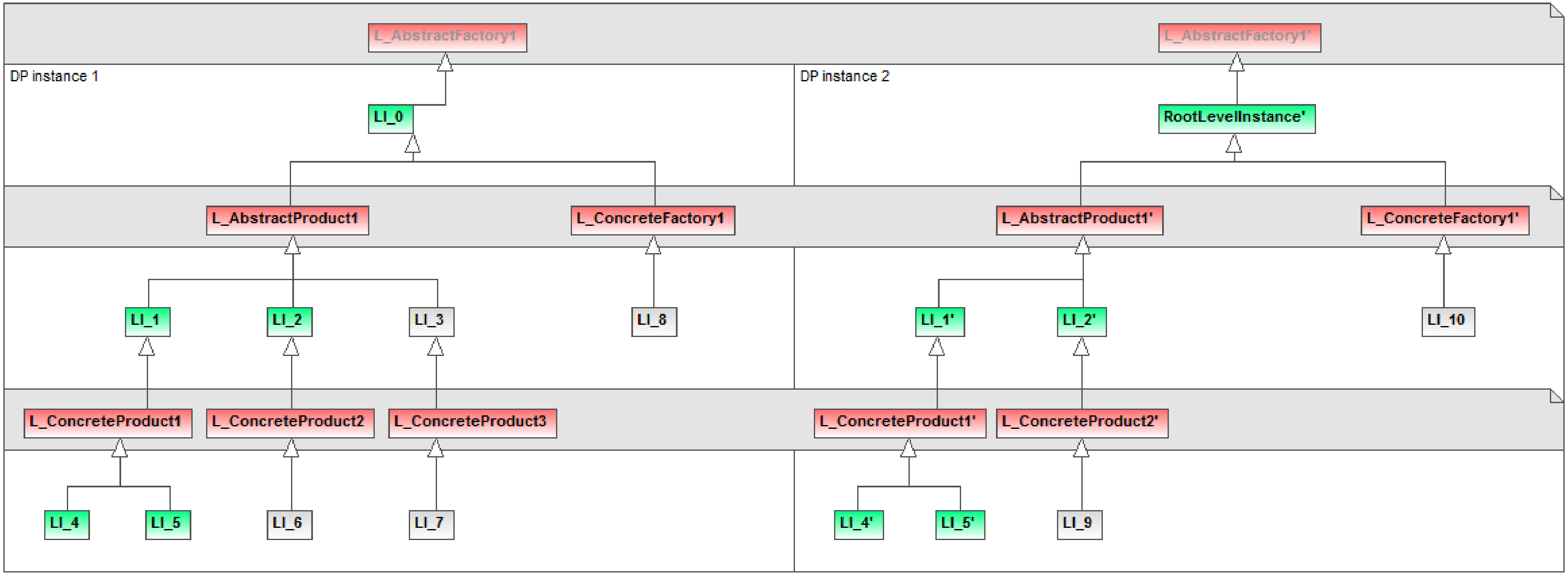
simL(L_AP1,L_AP10, 1) = (1+ 1 + 0) * 0.26 / 3 +(0.2 + 0 + 0) = 0.37

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

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

simL(L_CP3, null, 2) = **0**

simL(L_CF1,L_CF1',1) = 0 * 0,26 / 1 = 0



similarity = 1 * 0,29 + 0,37 + 0 = 0.66 => **66%**

simL(L_AP1,L_AP10, 1) = (1+ 1 + 0) * 0.26 / 3 +(0.2 + 0 + 0) = **0.37**

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

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

simL(L_CP3, null, 2) = **0**

simL(L_CF1,L_CF1',1) = 0 * 0,26 / 1 = **0**