



# **A Novel Client-Driven Perspective on Class Hierarchy Understanding and Quality Assessment**

**Petru Florin Mihancea**  
**LOOSE Research Group**  
**Politehnica University**  
**Timișoara, Romania**

# Context

**> 50 %** of software costs generated by maintenance

## Class hierarchies

Keys for easy to maintain software (via **polymorphism**)

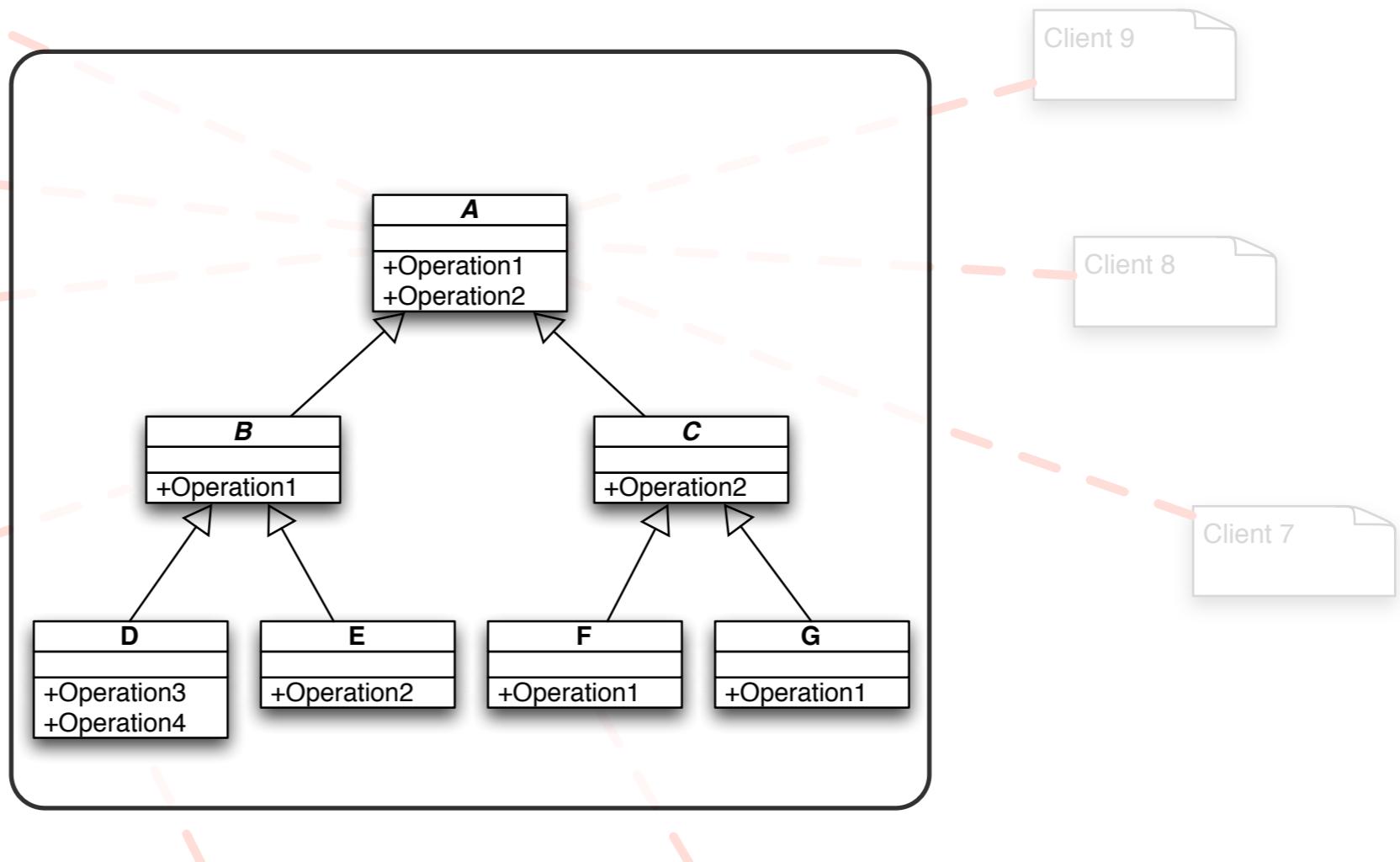
Also affected by the software **aging** phenomenon

## Class hierarchy **geriatrics**

Support to understand them and assess their quality

Do not capture how hierarchies are  
actually used via polymorphism in a  
system!

## Current Limitation



A model [class hierarchy] cannot be meaningfully validated  
in **isolation** [...] it should be validated in terms of its **clients**

Robert Martin

# Current Limitation



## **Issue I**

**What could be revealed about a hierarchy in this way ?**

1. **Observe** the usage of polymorphism in clients
2. **Discover** patterns of polymorphism usage
3. **Interpret** these patterns

# Issue I

9 out of 12 external clients are visible!

```
s.jung.graph.Graph,edu.uci.ics.jung.graph.Vertex,edu.uci.ics.jung.gra
if (undirected && firstEnd == secondEnd) return;
if (parallel) {
    Edge v_t;
    if (undirected)
        v_t = new UndirectedSparseEdge(firstEnd, secondEnd);
    else
        v_t = new DirectedSparseEdge(firstEnd, secondEnd);
    if (nev != null)
        nev.setNumber(v_t, new Integer(1));
    else
        v_t.addUserDatum(FOLDED_DATA, intermediate.copy_sc
newGraph.addEdge(v_t);
} else {
    Edge v_t= firstEnd.findEdge(secondEnd);
    if (v_t == null) {
        if (undirected)
            v_t = new UndirectedSparseEdge(firstEnd, second
        else
            v_t = new DirectedSparseEdge(firstEnd, secondEnd);
        if (nev != null)
            nev.setNumber(v_t, new Integer(0));
        else
            v_t.addUserDatum(FOLDED_DATA, new HashSet(), copy_
newGraph.addEdge(v_t);
    }
    if (nev != null)
        nev.setNumber(v_t, new Integer(nev.getNumber(v_t)));
    else
        Set folded_vertices= (Set) v_t.getUserDatum(FOLDED_D
folded_vertices.add(intermediate);
}
}

edu.uci.ics.jung.algorithms.flows.EdmondsKarpMaxFlow.updateResidualCapacities()
{
    MutableInteger augmentingPathCapacity= (MutableInteger) mTarget.getUserDatum(PARENT_CAPACITY_KEY);
    mMaxFlow += augmentingPathCapacity.intValue();
    Vertex currentVertex= mTarget;
    Vertex parentVertex= null;
    while ((parentVertex = (Vertex) currentVertex.getUserDatum(PARENT_KEY)) != currentVertex) {
        DirectedEdge currentEdge= (DirectedEdge) parentVertex.findEdge(currentVertex);
        MutableInteger residualCapacity= (MutableInteger) currentEdge.getUserDatum(RESIDUAL_CAPACITY_KEY);
        residualCapacity.subtract(augmentingPathCapacity.intValue());
        currentVertex = parentVertex;
    }
    DirectedEdge backEdge= (DirectedEdge) currentVertex.findEdge(parentVertex);
    residualCapacity = (MutableInteger) backEdge.getUserDatum(RESIDUAL_CAPACITY_KEY);
    residualCapacity.add(augmentingPathCapacity.intValue());
    currentVertex = parentVertex;
}

edu.uci.ics.jung.visualization.SpringLayout.update()
{
    try {
        for( Iterator iter= getGraph().getVertices().iterator(); iter.
            hasNext(); ) {
            Vertex v= (Vertex) iter.next();
            Coordinates coord= (Coordinates) v.getUserDatum(getBaseKey());
            if (coord == null) {
                coord = new Coordinate();
                v.getUserDatum(g_baseKey), coord, UserDat
                initializeLocation(v, coord, getCurrentSize(
                initialize_local_vertex(v);
            }
        }
    } catch( ConcurrentModificationException cme){
        update();
    }
}

fwd.level = 0
// the node should also be moved to near its neighbors
moveVertexPrettily((Vertex) vert.getUserDatum(g_int));
if (!nowShowingNodes.contains(vert))
    hiddenNodes.contains(vert));
// System.out.println(v + " " + v.getClass());
fwdlyHidden++;
// move vertex outward
if (fwdlyHidden > 1000) {
    fndv = (FadingVertexLayoutData) vert.getUserDatum(getFadingKey());
    fndv.isHidden = true;
    fwd.level = 0
    // the node should gradually surf away from its neighbors
}

edu.uci.ics.jung.visualization.FadingVertexLayout.tick()
{
    try {
        for( Iterator iter= getGraph().getVertices().itera
            hasNext(); ) {
            Vertex v= (Vertex) iter.next();
            Coordinates coord= getCoordinates(v);
            Coordinates coord = (Coordinates) v.getUserDa
            if (coord == null) {
                coord = new Coordinates();
                v.addUserDatum(getBaseKey(), coord, UserDat
                initializeLocation(v, coord, getCurrentSize(
                initialize_local_vertex(v);
            }
        }
    } catch( ConcurrentModificationException cme){
        update();
    }
    initialize_local();
}

edu.uci.ics.jung.visualization.FadingVertexLayout
{
    super.initialize(graph, true, false);
    nAlpha = alpha;
    nMaxDepth = maxDepth;
    nPriors = priors;
    fwd = new FadingVertexLayoutData();
    Iterator vIt= graph.getVertices().iterator(); vIt.hasNext(); )
        vIt.next();
        vIt.setUserData(WEIGHTED_NIPATHS_KEY, new MutableDouble(0)
    }

    Graph graph= layout.getGraph();
    for( Iterator iter= graph.getVertices().iterator(); iter.hasNext(); ) {
        Vertex av= (Vertex) iter.next();
        FadingVertexLayoutData fwd=
            (FadingVertexLayoutData) av.getUserDatum(getFadingKey());
        if (fwd.level < fadelevels) {
            fwd.level++;
            if (fwd.isHidden) {
                moveOutward(av, getX(av), getY(av), 1.01);
            }
        }
    }
}
```

# Issue I

## Type Highlighting (TH) Technique

8th IEEE International Working Conference  
on Source Code Analysis and Manipulation

The screenshot shows a Mac OS X desktop environment with several windows open, likely representing different parts of a Java application or library. The central focus is a window titled "edu.uci.ics.jung.visualization.FadingVertexLayout.tick0" which displays Java code. The word "FADING" is highlighted in red, indicating the target for type highlighting. Other windows visible include:

- A window titled "edu.uci.ics.jung.algorithms.flows.EdmondsKarpMaxFlow.updateResidualCapacities0" showing code related to graph algorithms.
- A window titled "edu.uci.ics.jung.visualization.SpringLayout.update0" showing code related to spring layout.
- A window titled "edu.impl.NumericDecorationFilter.acceptVerte" showing code related to vertex filtering.
- A window titled "s.jung.graph.Graph,edu.uci.ics.jung.graph.Vertex,edu.uci.ics.jung.gra" showing code related to graph vertex operations.

The desktop interface includes a Dock at the bottom with icons for various applications like Finder, Mail, Safari, and others.

```
if (undirected && firstEnd == secondEnd) return;
if (parallel) {
    Edge v_t;
    if (undirected)
        v_t = new UndirectedSparseEdge(firstEnd, secondEnd);
    else
        v_t = new DirectedSparseEdge(firstEnd, secondEnd);
    if (nev != null)
        nev.setNumber(v_t, new Integer(1));
    else
        v_t.addUserDatum(FOLDED_DATA, intermediate.copy_sc);
    newGraph.addEdge(v_t);
} else {
    Edge v_t= firstEnd.findEdge(secondEnd);
    if (v_t == null) {
        if (undirected)
            v_t = new UndirectedSparseEdge(firstEnd, secondEnd);
        else
            v_t = new DirectedSparseEdge(firstEnd, secondEnd);
        if (nev != null)
            nev.setNumber(v_t, new Integer(1));
        else
            v_t.addUserDatum(FOLDED_DATA, intermediate.copy_sc);
        newGraph.addEdge(v_t);
    }
    if (nev != null)
        nev.setNumber(v_t, new Integer(1));
    else
        Set folded_vertices= (Set) v_t.getVertices();
        folded_vertices.add(intermediate);
}
```

```
Number n= (Number) vertex.getUserDatum(FADING);
if (n.doubleValue() > mThreshold) {
    return true;
}
edu.uci.ics.jung.graph.filters.UnassembledGraphs AG from the various edges
Graph subgraph= (Graph) originalGraph;
if (shouldPreserveRecord) {
    subgraph.addUserDatum(GraphAssemblyRecord.FILTER_GRAPH,
        new GraphAssemblyRecord(this), UserData.PEDOMETER);
}
Set localVertices= getUntouchedVertices();
// now, we scoop up vertices
for( Iterator iter= localVertices.iterator(); iter.hasNext();
    ArchetypeVertex newV= (ArchetypeVertex) iter.next();
    newV.copy(subgraph);
)
// and only the edges that actually match
for( Iterator iter= getUntouchedEdges().iterator(); iter.hasNext();
    ArchetypeEdge newE= (ArchetypeEdge) iter.next();
    if (localVertices.containsAll(newE.getIncidentVertices()))
        newE.copy(subgraph);
)
```

```
MutableInteger augmentingPathCapacity= (MutableInteger) mTarget.getUserDatum(PARENT_CAPACITY_KEY);
mMaxFlow += augmentingPathCapacity.intValue();
Vertex currentVertex= mTarget;
Vertex parentVertex= null;
while ((parentVertex = (Vertex) currentVertex.getUserDatum(PARENT_KEY)) != currentVertex) {
    DirectedEdge currentEdge= (DirectedEdge) parentVertex.findEdge(currentVertex);
    MutableInteger residualCapacity= (MutableInteger) currentEdge.getUserDatum(RESIDUAL_CAPACITY_KEY);
    residualCapacity.subtract(augmentingPathCapacity.intValue());
    currentVertex = parentVertex;
}
DirectedEdge backEdge= (DirectedEdge) currentVertex.findEdge(parentVertex);
residualCapacity = (MutableInteger) backEdge.getUserDatum(RESIDUAL_CAPACITY_KEY);
residualCapacity.add(augmentingPathCapacity.intValue());
currentVertex = parentVertex;
```

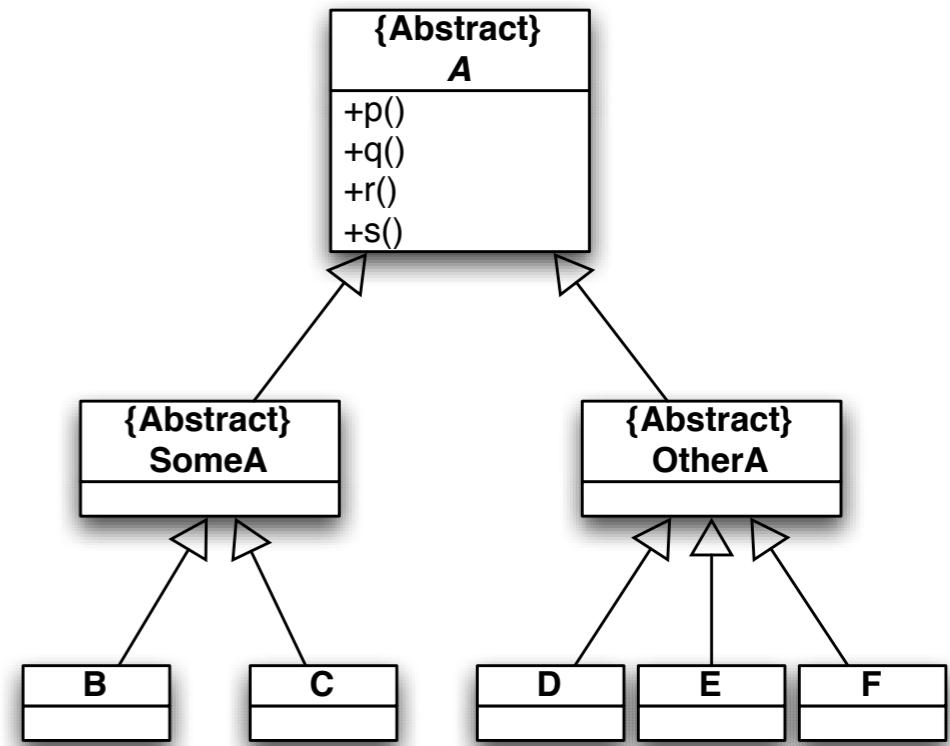
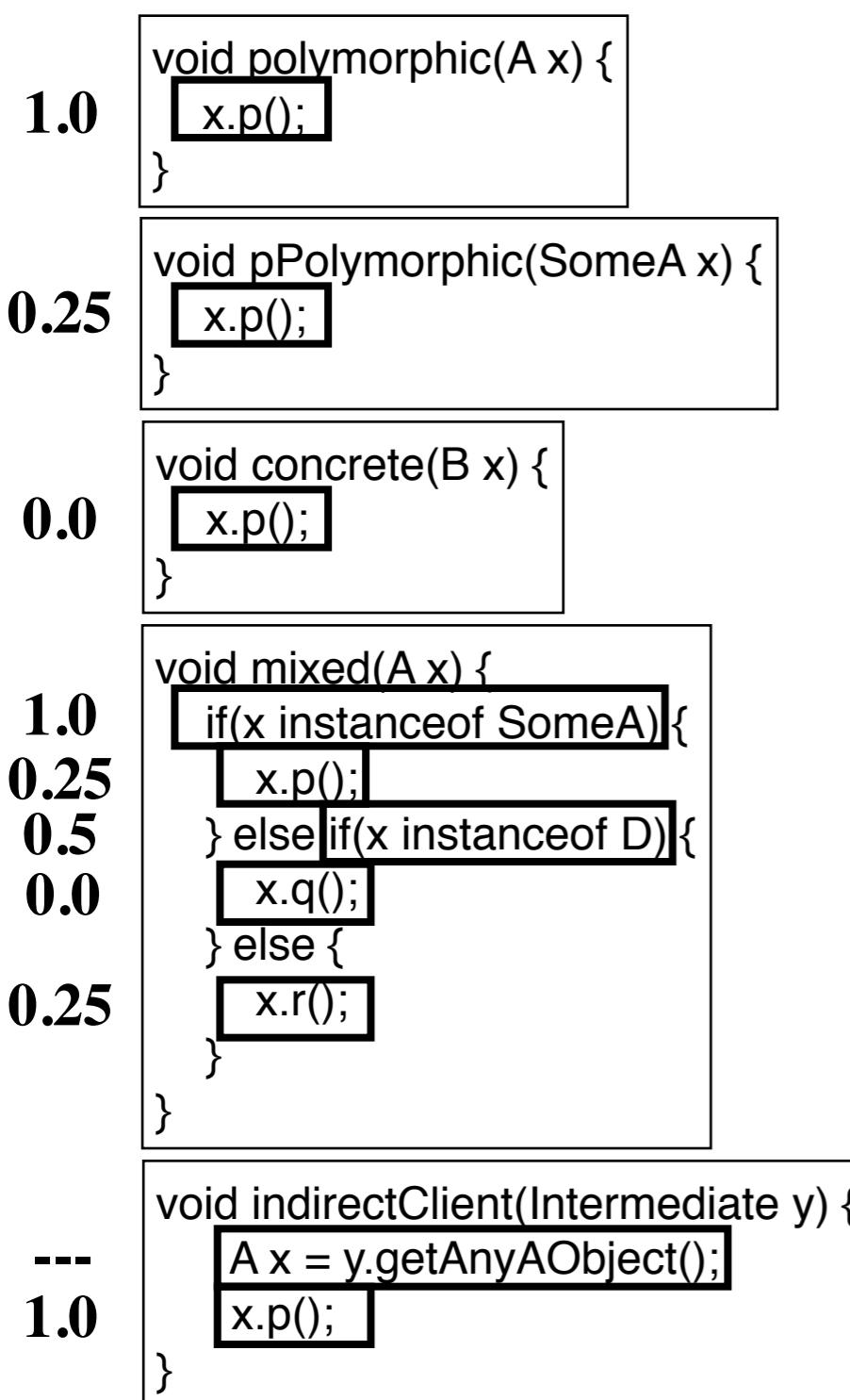
```
try {
    FADING fndv= (FADING) vertex.getUserDatum(FADING);
    if (fndv != null) {
        fndv.level = 0;
        // the node should also be moved to near its neighbors
        for( Iterator iter= vertex.getNeighbors(); iter.hasNext();
            iter.next();
            vert= iter.next();
            if (vert.equals(vertex))
                continue;
            if (fndv.level < fndv.fadelevels) {
                fndv.level++;
                if (fndv.isHidden)
                    moveOutward(av, getX(av), getY(av), 1.01);
            }
        )
    }
}
```

```
fndv.level = 0
// the node should also be moved to near its neighbors
for( Iterator iter= vertex.getNeighbors(); iter.hasNext();
    iter.next();
    vert= iter.next();
    if (vert.equals(vertex))
        continue;
    if (fndv.level < fndv.fadelevels) {
        fndv.level++;
        if (fndv.isHidden)
            moveOutward(av, getX(av), getY(av), 1.01);
    }
)
```

```
try {
    FADING fndv= (FADING) vertex.getUserDatum(FADING);
    if (fndv != null) {
        fndv.level = 0;
        // the node should also be moved to near its neighbors
        for( Iterator iter= vertex.getNeighbors(); iter.hasNext();
            iter.next();
            vert= iter.next();
            if (vert.equals(vertex))
                continue;
            if (fndv.level < fndv.fadelevels) {
                fndv.level++;
                if (fndv.isHidden)
                    moveOutward(av, getX(av), getY(av), 1.01);
            }
        )
    }
}
```

```
iter= getGraph().getVertices().iterator();
(Verte iter.next();
    if (coord= getCoordinates(v) != null) {
        UserDat userDat= v.getUserDatum(FADING);
        if (userDat == null) {
            coord= new Coordinates();
            v.addUserDatum(getBaseKey(), coord);
            initializeLocation(v, coord, getCurrentSize);
            initialize_local_vertex(v);
        }
    }
)
} catch( ConcurrentModificationException cme){
    update();
}
initialize_local();
```

# TH : Quantifying Polymorphism Usage



## Level of Abstraction Metric

Source code token level

~ Relative number of classes whose instances may be referred at the token execution point

# TH : Seeing Polymorphism Usage

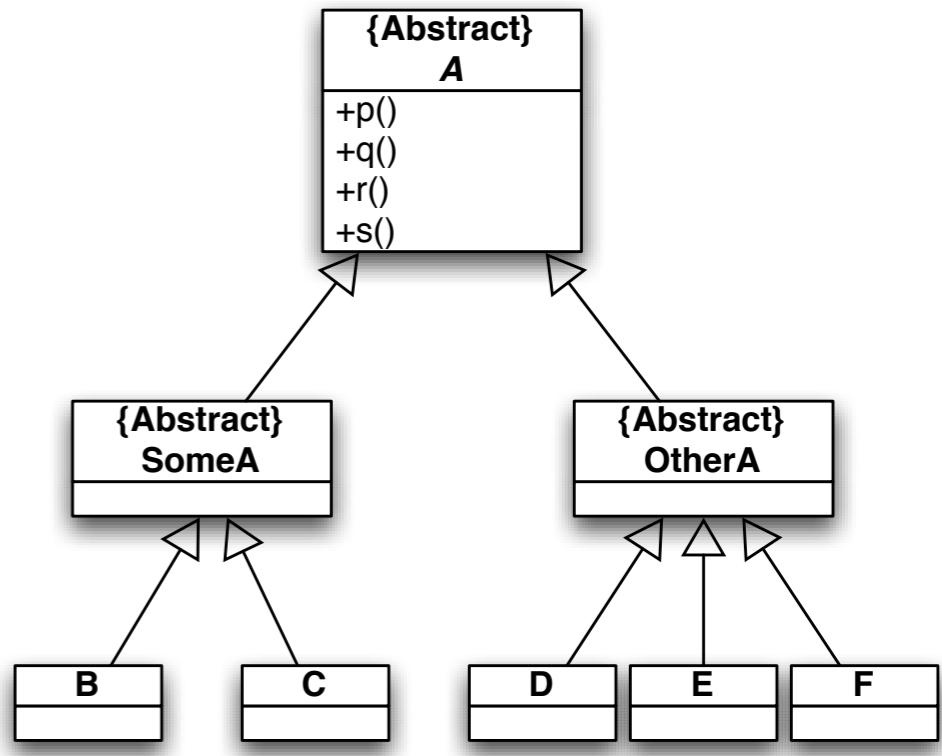
```
void polymorphic(A x) {
    x.p();
}
```

```
void pPolymorphic(SomeA x) {
    x.p();
}
```

```
void concrete(B x) {
    x.p();
}
```

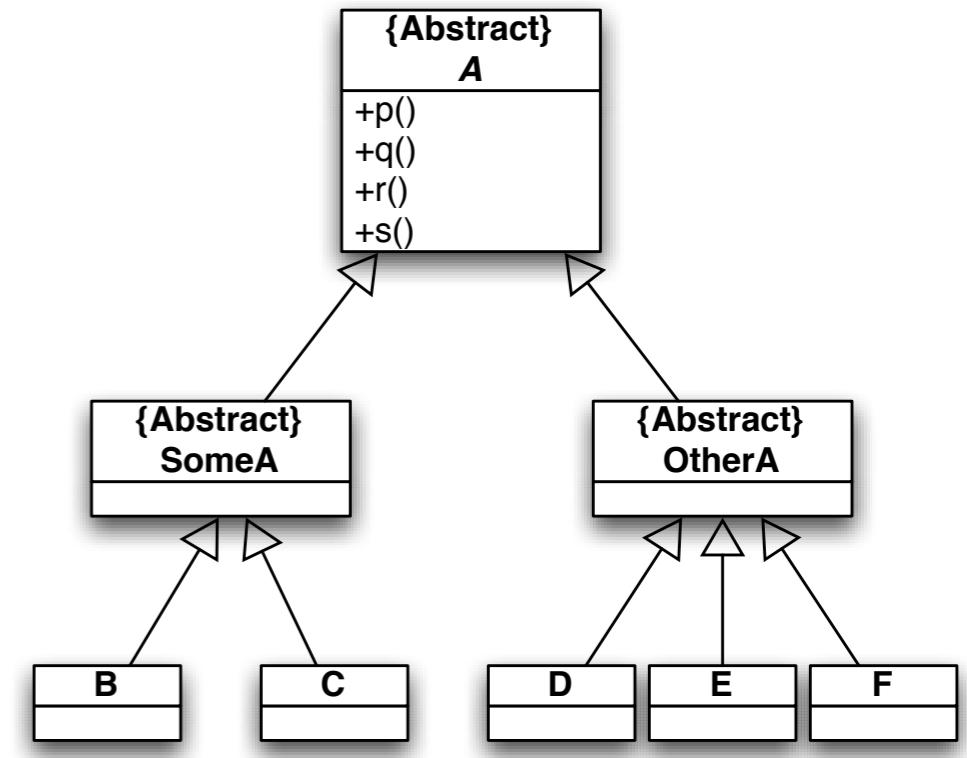
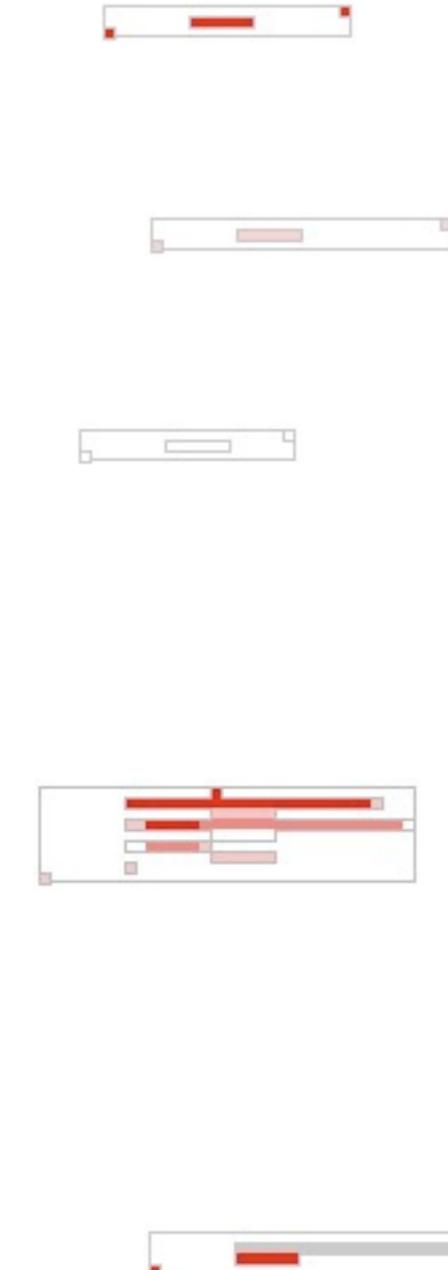
```
void mixed(A x) {
    if(x instanceof SomeA) {
        x.p();
    } else if(x instanceof D) {
        x.q();
    } else {
        x.r();
    }
}
```

```
void indirectClient(Intermediate y) {
    A x = y.getAnyAObject();
    x.p();
}
```



LA Values	Color
$LA(tk) = 1$	- Red
$LA(tk) \geq 0.5 \wedge LA(tk) < 1$	- Diluted Red
$LA(tk) > 0 \wedge LA(tk) < 0.5$	- More Diluted Red
$LA(tk) = 0$	- White
<i>undefined</i>	- Light-gray

# TH : Seeing Polymorphism Usage

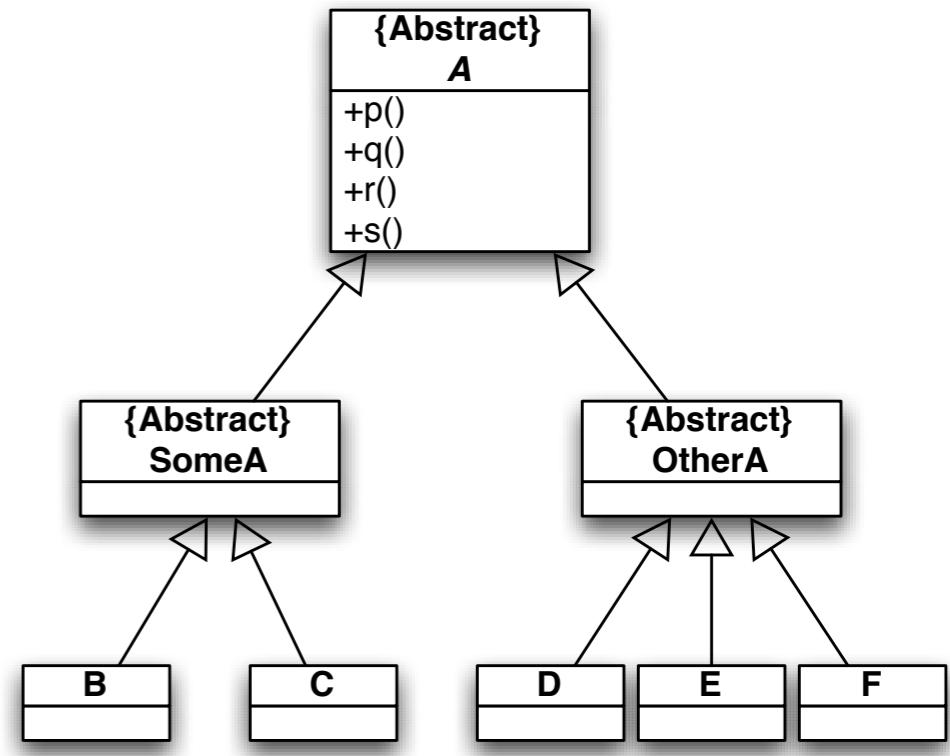


LA Values	Color
$LA(tk) = 1$	- Red
$LA(tk) \geq 0.5 \wedge LA(tk) < 1$	- Diluted Red
$LA(tk) > 0 \wedge LA(tk) < 0.5$	- More Diluted Red
$LA(tk) = 0$	- White
<i>undefined</i>	- Light-gray

# TH : Seeing Polymorphism Usage



Level of Abstraction  
View

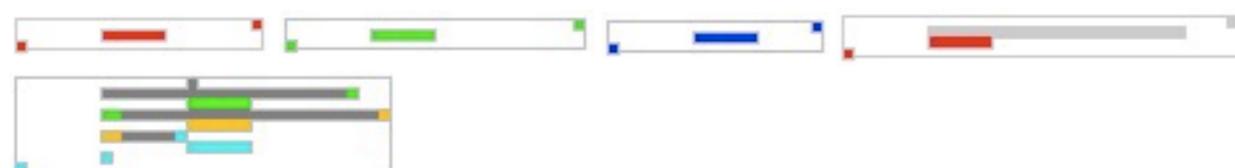


LA Values	Color
$LA(tk) = 1$	- Red
$LA(tk) \geq 0.5 \wedge LA(tk) < 1$	- Diluted Red
$LA(tk) > 0 \wedge LA(tk) < 0.5$	- More Diluted Red
$LA(tk) = 0$	- White
<i>undefined</i>	- Light-gray

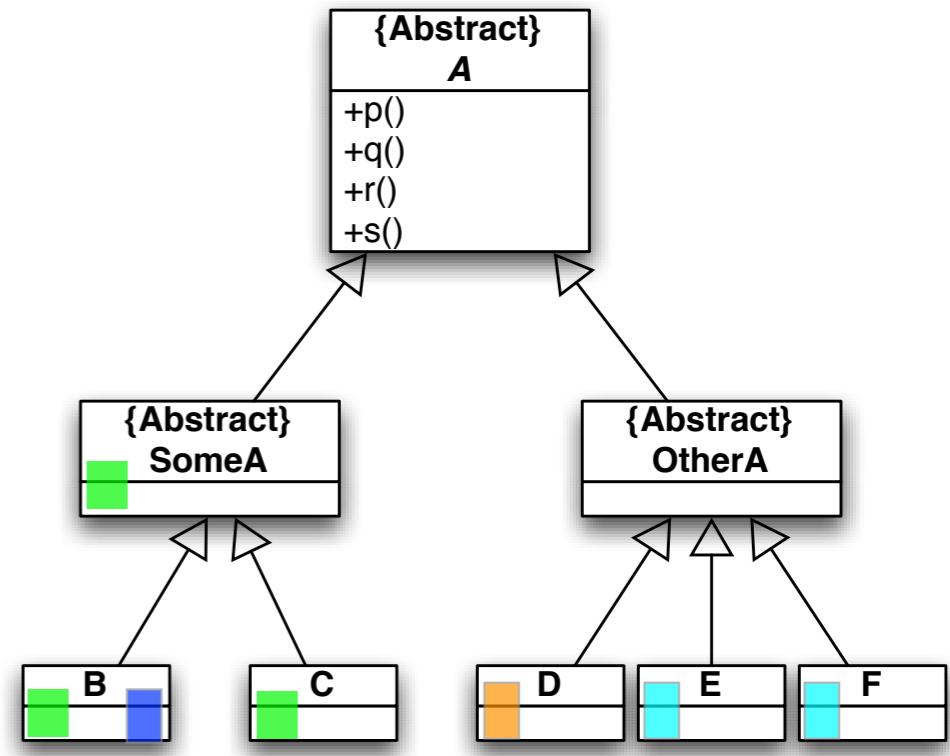
# TH : Seeing Polymorphism Usage



Level of Abstraction  
View



Group Discrimination  
View



LA Values	Color
$LA(tk) = 1$	- Red
$LA(tk) \geq 0.5 \wedge LA(tk) < 1$	- Diluted Red
$LA(tk) > 0 \wedge LA(tk) < 0.5$	- More Diluted Red
$LA(tk) = 0$	- White
<i>undefined</i>	- Light-gray

# **Some Experimental Results. Examples**

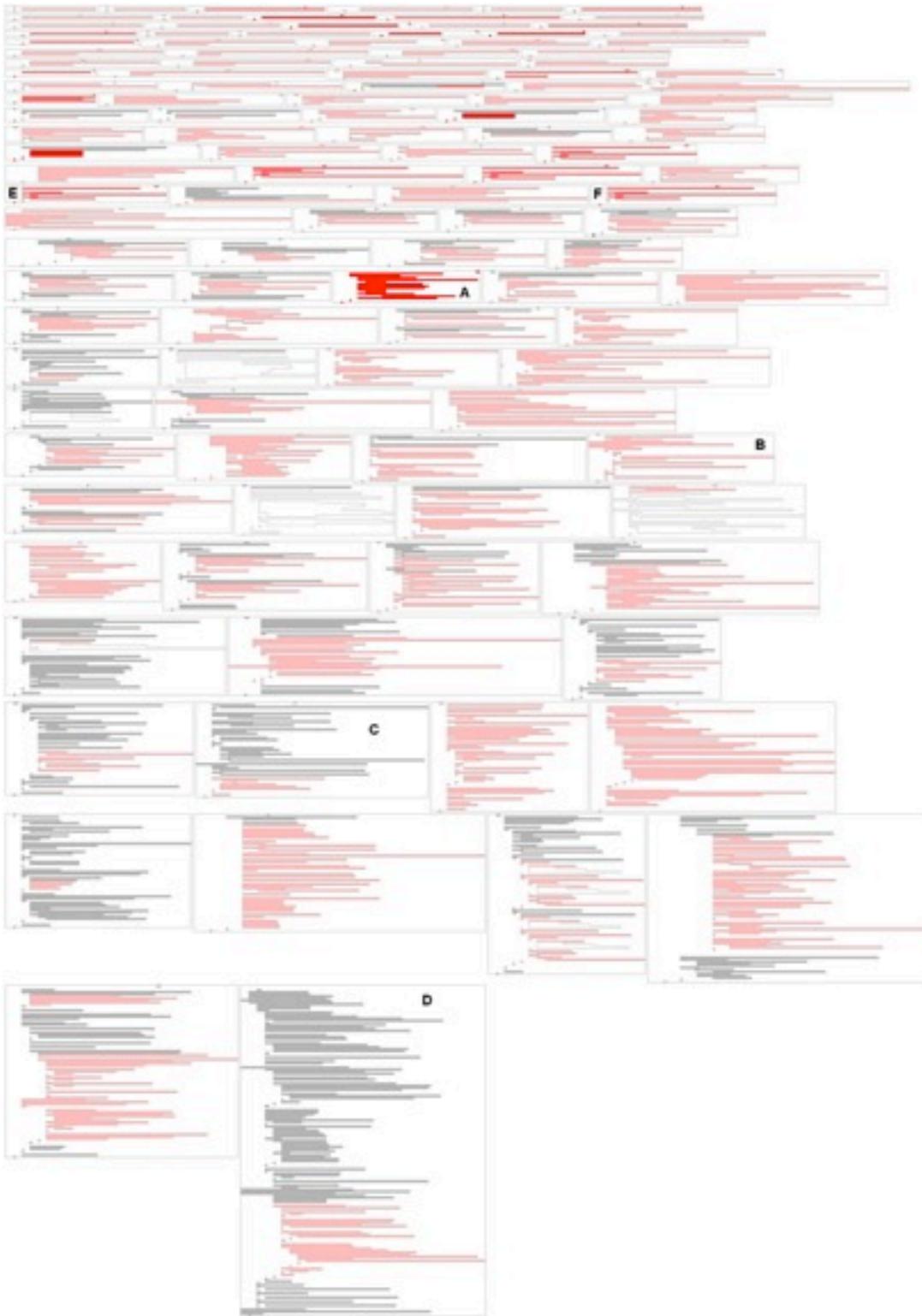
## **5 Fine-Grained Visual Patterns**

**Client level**

## **6 Coarse-Grained Visual Patterns**

**Client Population Level**

# Some Experimental Results. Examples



Partially Polymorphic  
External Population

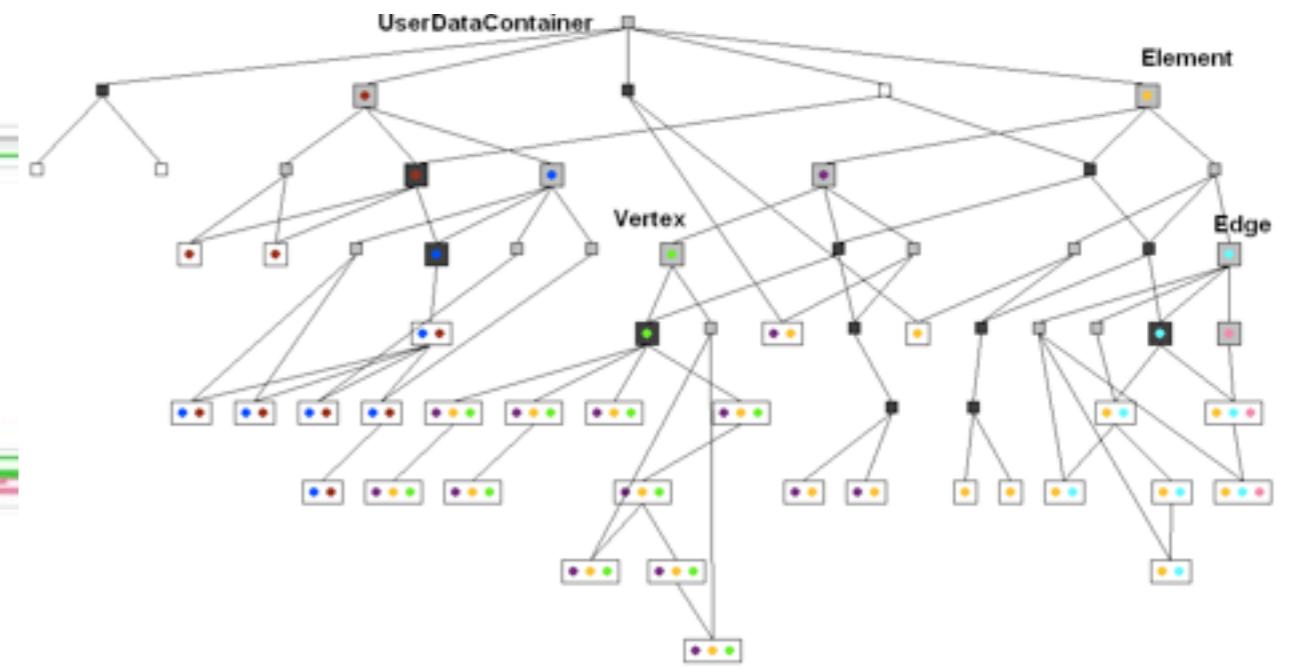
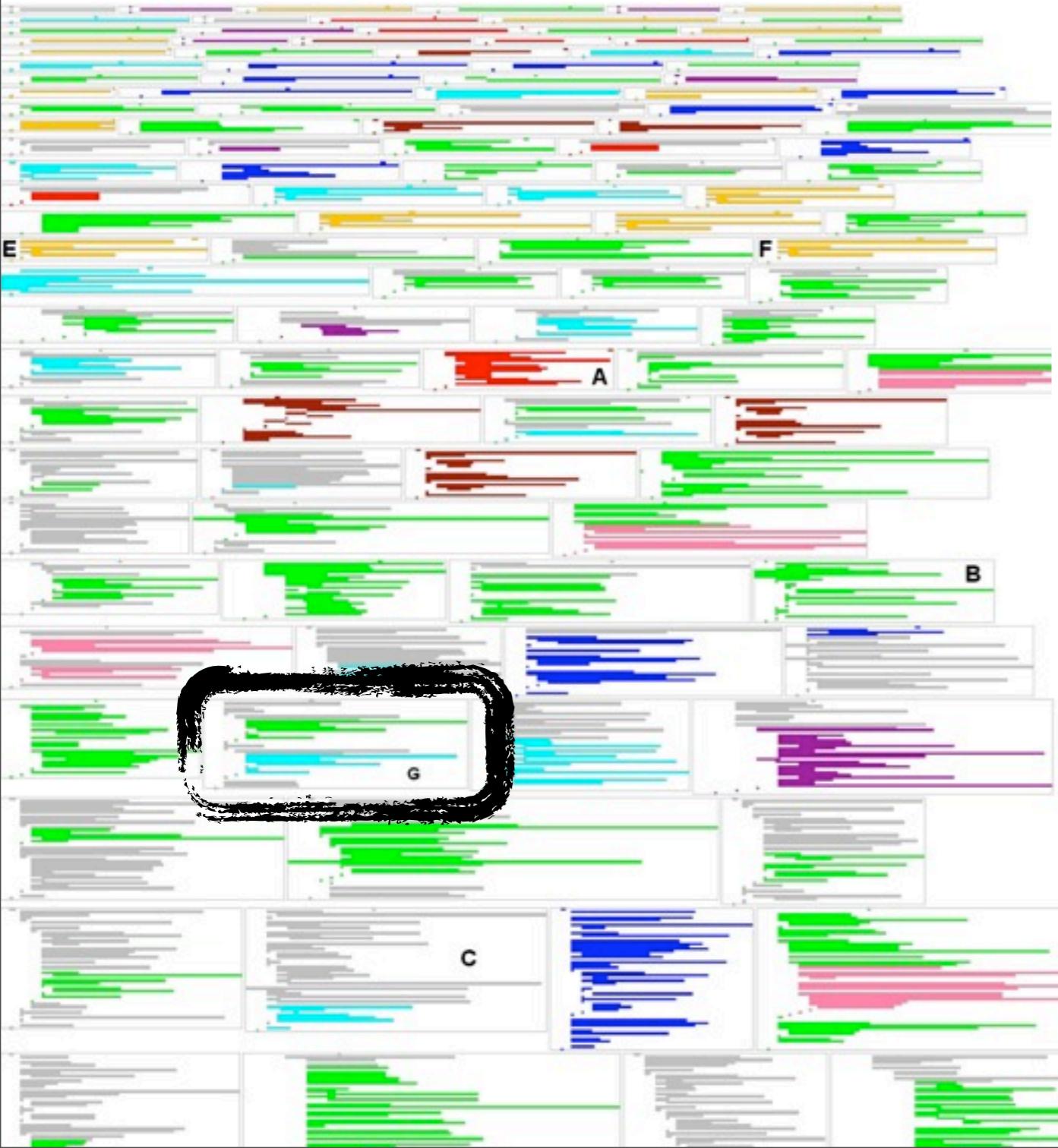
# Some Experimental Results. Examples



Partially Polymorphic  
External Population

Polymorphic Island

# Some Experimental Results. Examples



Mixed Twins

## Issue 2 : The Dual Nature of Inheritance

Class hierarchies can model

Type hierarchies

Implementation hierarchies

How are they  
used in a legacy system?

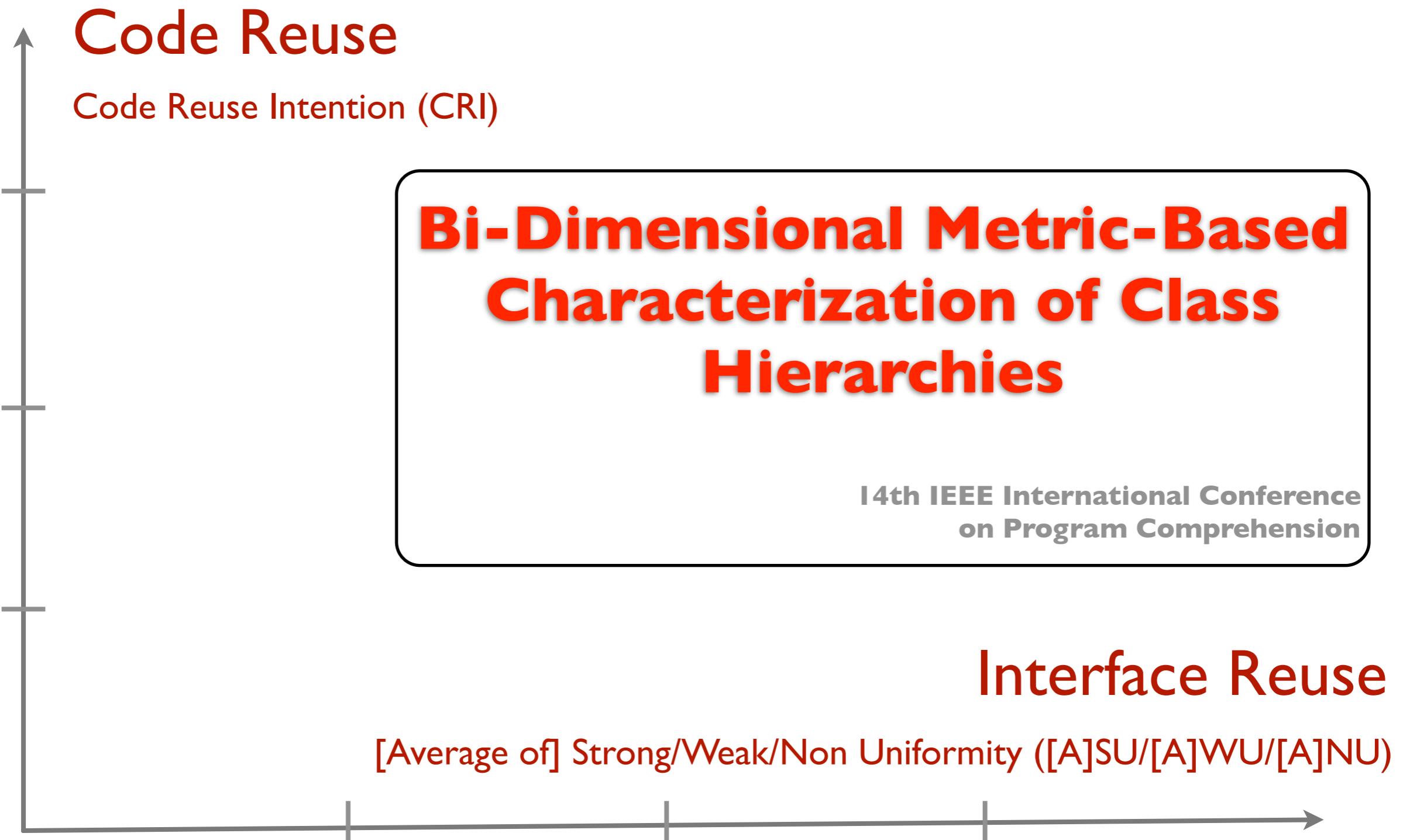


Level of Abstraction View Limitations

Visual detection of interface reuse nature

Almost nothing about code reuse nature

# Approach



# Experimental Results

## Case studies

System \ Metrics	NOC	NOM	LOC
Recoder	490	6 795	42 259
Jung	391	3 038	22 447
FreeMind	455	5 228	52 904
InternalProduct	124	1 002	11 210

## Base classes

System	All Base Classes	Base Classes having $CRI > 0.75$
Recoder	219	40
Jung	168	54
Freemind	239	29
InternalProduct	20	5

## Detailed investigation

Class	$CRI$	$ASU$	$AWU$	$ANU$
AbstractArrayList	1.0	0.0	0.12	0.88
LineAdapter	0.77	0.0	0.88	0.12
AbstractLayout	0.94	1.0	0.0	0.0

# Experimental Results

## Case studies

System \ Metrics	NOC	NOM	LOC
Recoder	490	6 795	42 259
Jung	391	3 038	22 447
FreeMind	455	5 228	52 904
InternalProduct	124	1 002	11 210

## Base classes

System	All Base Classes	Base Classes (%)
Recoder	219	
Jung	168	
Freemind	239	
InternalProduct	20	

## Detailed investigation

Class	NU	ANU
AbstractArr	0.12	0.88
LineAd	0.88	0.12
Abstr	0.0	0.0

Metrics interpretation is consistent with our manual observations from source code

## **Issue 3 : Understanding Cost**

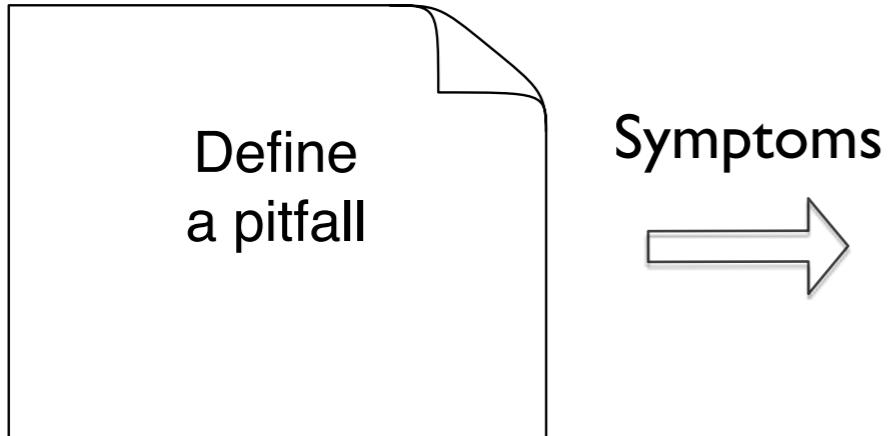
**~ 50 %** of maintenance costs and includes the time wasted due to **misunderstanding**

Class hierarchies may not be fully intended for polymorphism !

**Discover Comprehension  
Pitfalls in Class Hierarchies**

13th European Conference on Software  
Maintenance and Reengineering

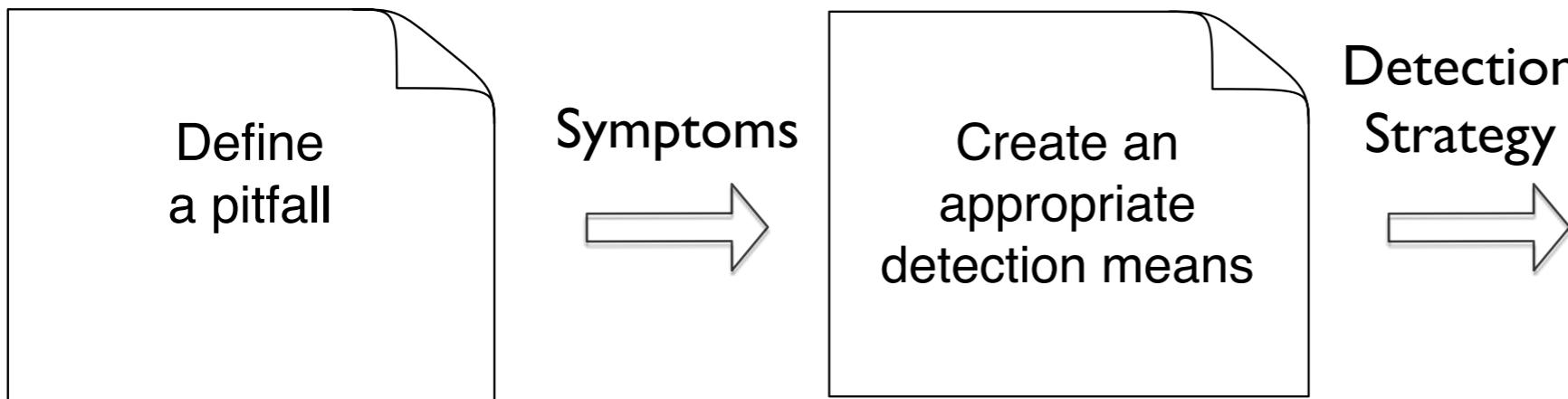
# Approach



## Fine-grained deviations from design heuristics

Related to polymorphism

# Approach

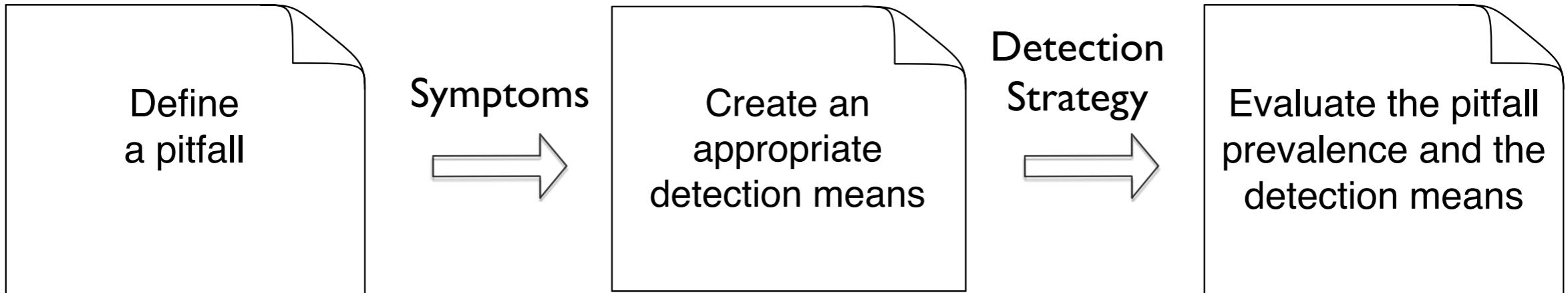


Create a corresponding **metric-based rule**

Uniformity Metrics (SU, ASU, WU, AWU, NU, ANU)

Type Affinity (TA)

# Approach



## Apply the rule on case studies

Manual investigation of the suspected entities

	LOC	CLASSES	METHODS
Recoder	42 259	490	6 795
FreeMind	52 904	455	5 228
Jung	22 447	391	3 038

# **Identified Comprehension Pitfalls**

**3 Pitfalls**

**Partial Typing**

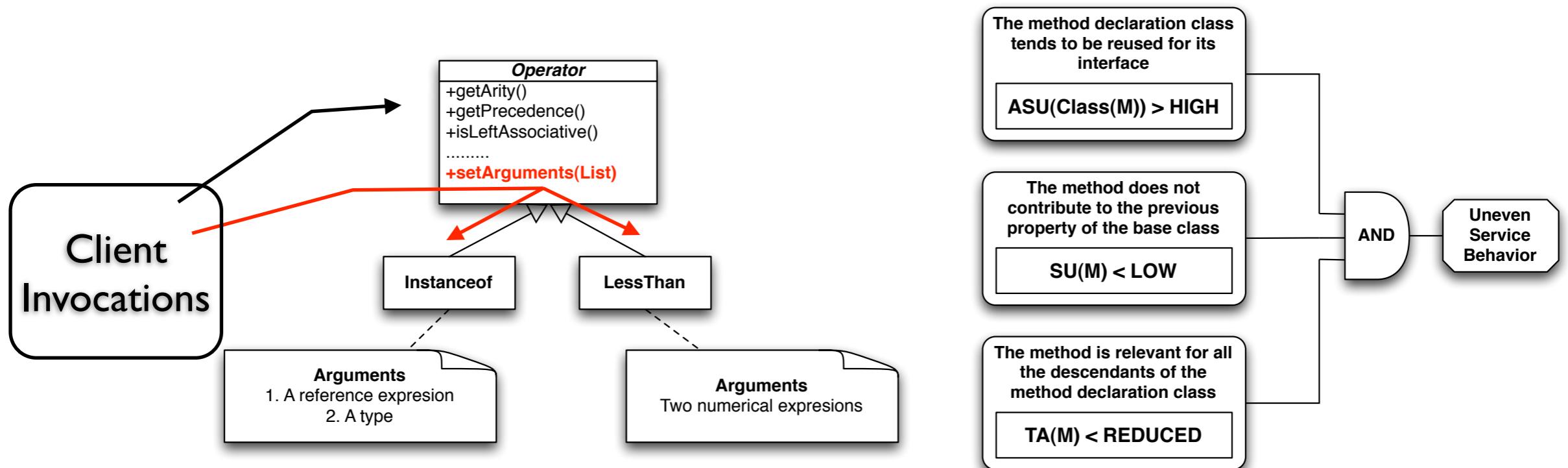
**Uneven Service Behavior**

**Premature Service**

# Identified Comprehension Pitfalls

## Uneven Service Behavior

- Affected entity - base class method
- The method only enforces a **non-uniform** service
- The other methods in base class have an **uniform** semantics



# Experimental Results. Detection Precision

Uneven Service Behavior

	Suspected	True Positives
Recoder	21	13
FreeMind	8	7
Jung	3	3
TOTAL	32	23

~ 72 %

Partial Typing

	Suspected	True Positives
Recoder	9	9
FreeMind	2	2
Jung	10	8
TOTAL	21	19

~90 %

Premature Service

	Suspected	True Positives
Recoder	7	4
FreeMind	6	2
Jung	0	0
TOTAL	13	6

~ 46 %

# **Conclusions & Future Work**

## **3 promising contributions**

Type highlighting

Metric-based bi-dimensional characterization

Comprehension pitfalls and their detection strategies

## **Some limitations**

Metrics estimation using intra-procedural SCA

## **Some future work**

Stronger case studies with disinterested human subjects