

UNSUPERVISED MACHINE LEARNING FOR SOFTWARE MAINTENANCE

Thesis Advisors:

Dr. Sergio Di Martino

Dr. Anna Corazza

Ph.D. Candidate:

Valerio Maggio

June 5th, 2013

UNSUPERVISED MACHINE LEARNING FOR SOFTWARE MAINTENANCE

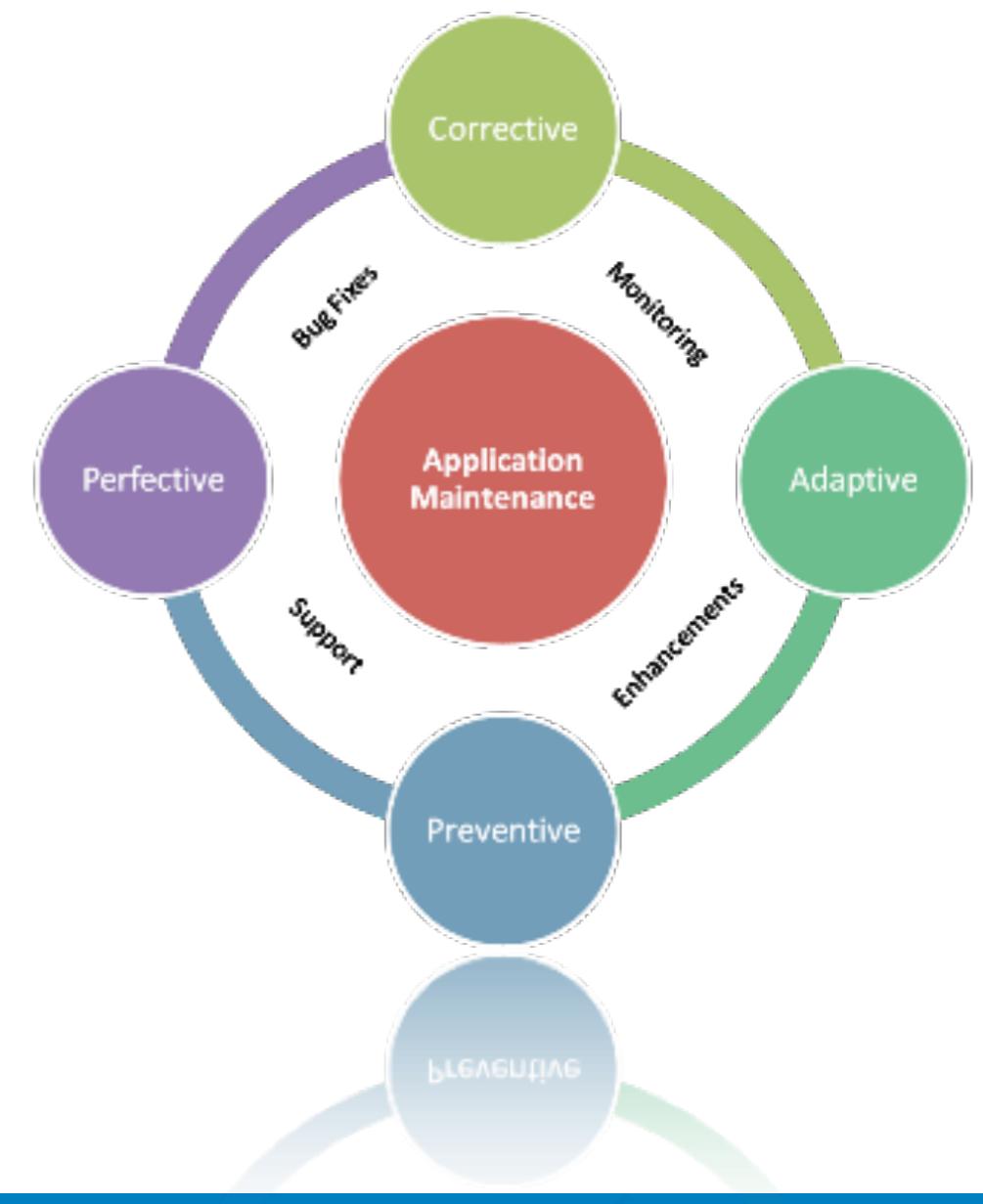
Define and experimentally evaluate solutions (techniques and prototype tools) for **automatic software analysis** to support software **maintenance** activities.

These solutions exploit (**unsupervised**) **machine learning** techniques to mine information from the **source code**

SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."
(cit. Lehman's First Law of Software Evolution)

There exist different **types** of Software Maintenance.

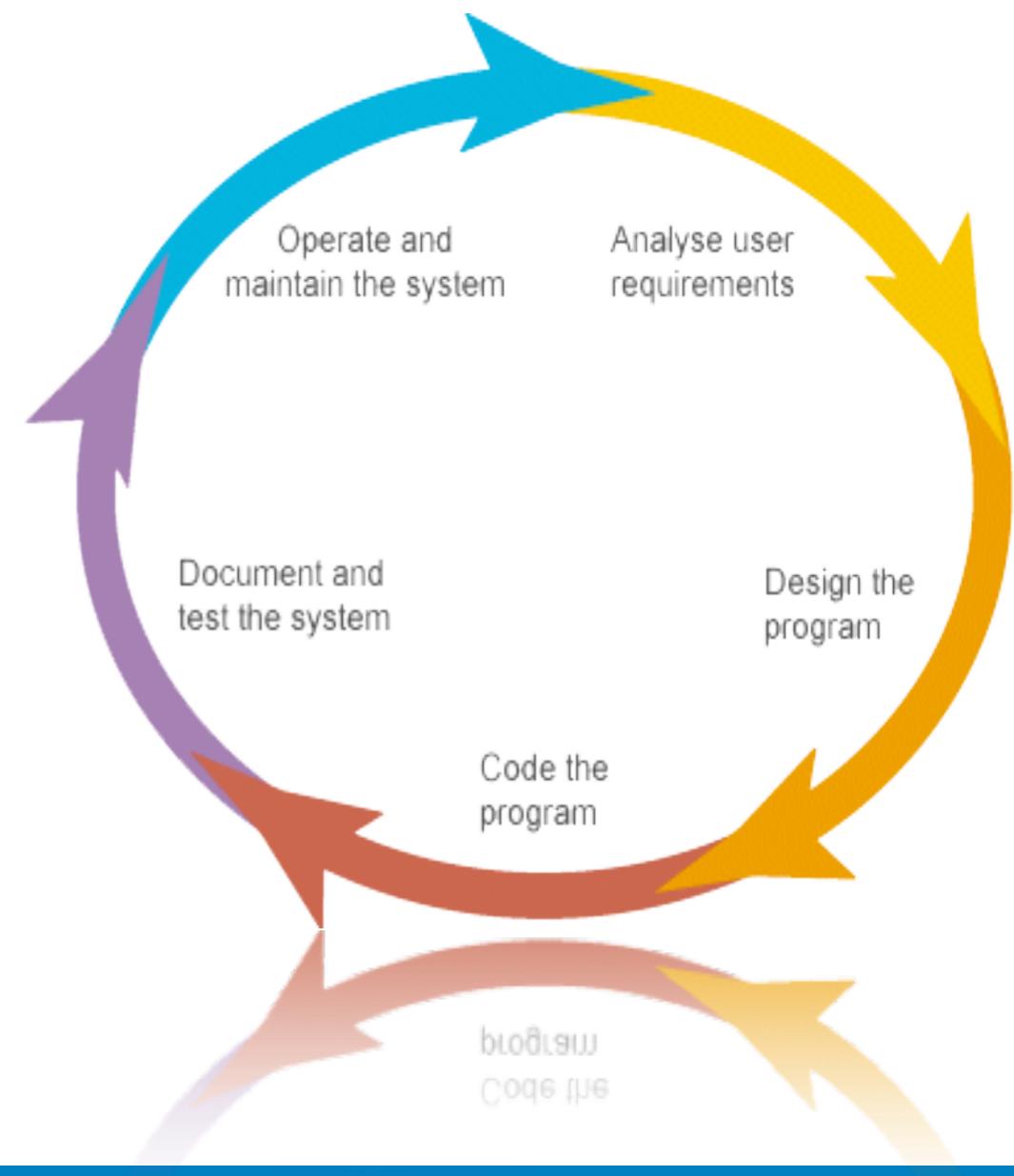


SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."

(cit. Lehman's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

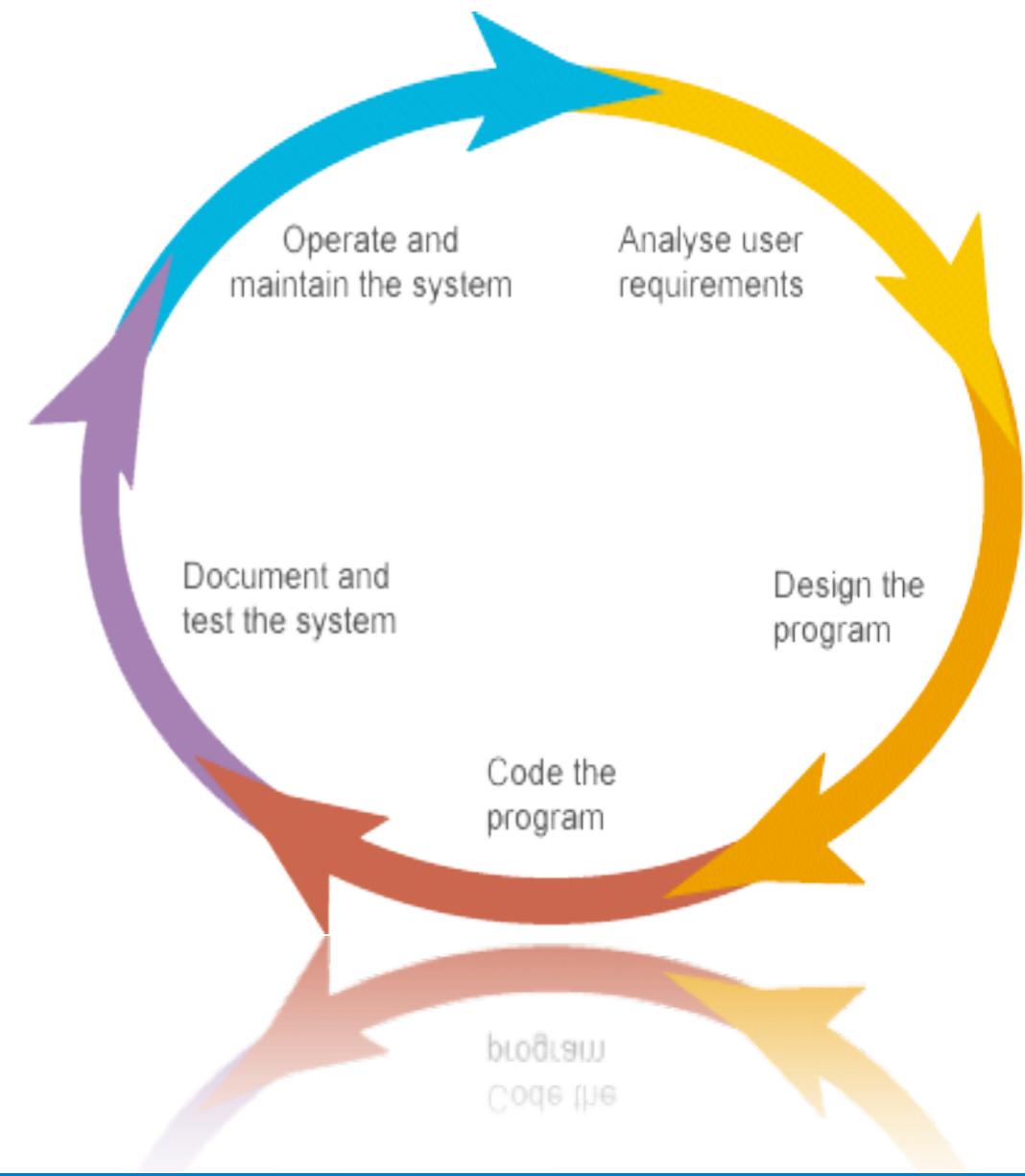


SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."
(cit. Lehman's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the **85-90%** of the total software costs.



ISSUES

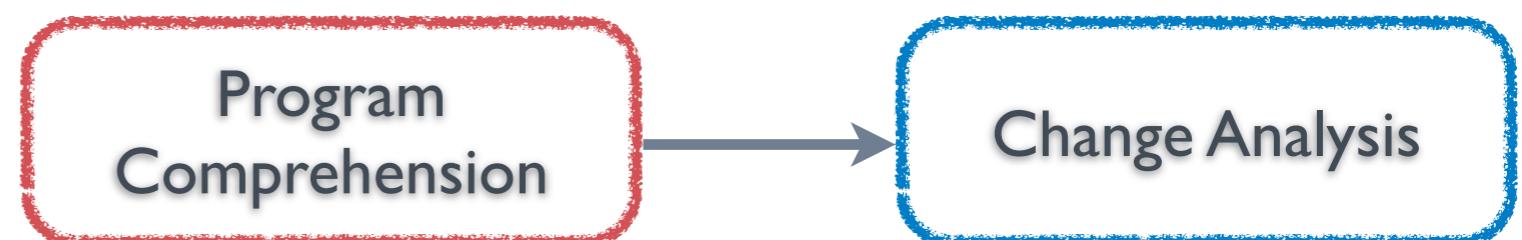
SOFTWARE MAINTENANCE

“Software Maintenance is about change!”
(cit. S. Jarzabek)

Change Analysis

SOFTWARE MAINTENANCE

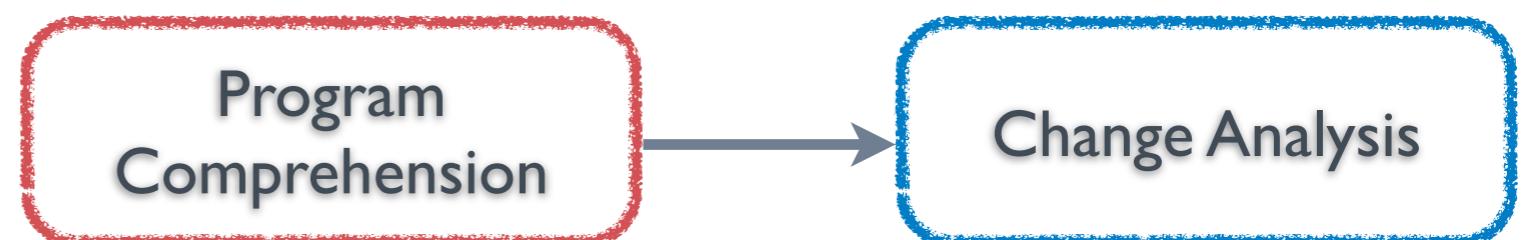
“Software Maintenance is about change!”
(cit. S. Jarzabek)



The **documentation** is usually scarce or **not up to date!**

SOFTWARE MAINTENANCE

“Software Maintenance is about change!”
(cit. S. Jarzabek)



The **documentation** is usually scarce or **not up to date!**

The **source code** (usually) represents the **most reliable** source of information about the system

SOFTWARE MAINTENANCE

“Software Maintenance is about change!”
(cit. S. Jarzabek)



The **documentation** is usually scarce or **not up to date!**

The **source code** (usually) represents the **most reliable** source of information about the system

- Definition of **tools** and **techniques** to support maintenance activities
 - **Goal:** Build higher-level software *models* in an **automatic fashion** gathering information from the **source code** or any other document
 - **Goal:** To aid the comprehension of the system

Reverse engineering

- Definition of **tools** and **techniques** to support maintenance activities
 - **Goal:** Build higher-level software models in an **automatic fashion** gathering information from the **source code** or any other document
 - **Goal:** To aid the comprehension of the system



STATIC ANALYSIS

DYNAMIC ANALYSIS

REVERSE E N G I N E E R I N G

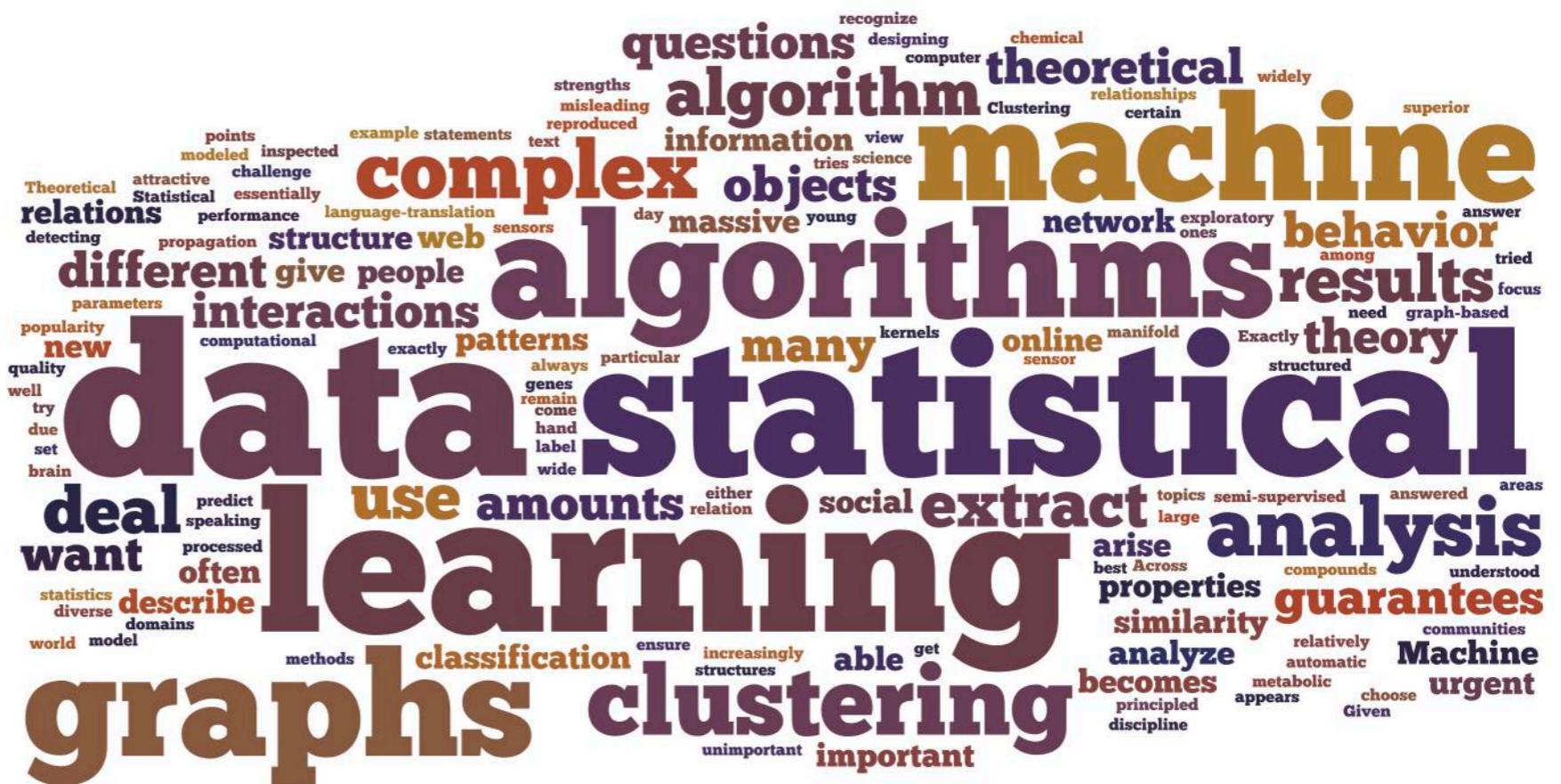
- Definition of **tools** and **techniques** to support maintenance activities
 - **Goal:** Build higher-level *software models* in an **automatic fashion** gathering information from the **source code** or any other document
 - **Goal:** To aid the comprehension of the system

engineering

STATIC ANALYSIS

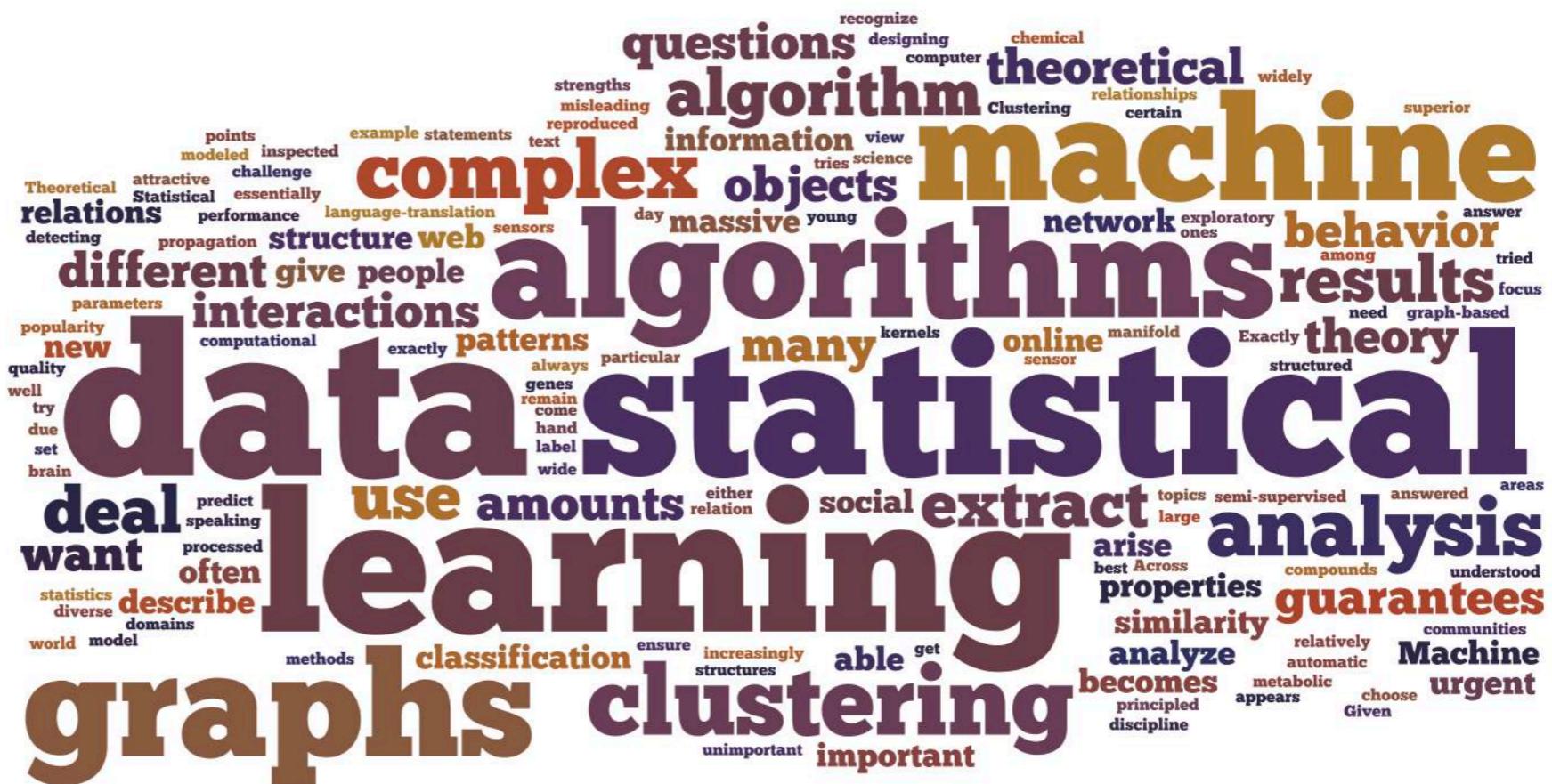
DYNAMIC ANALYSIS

MACHINE LEARNING



- Provides computational effective solutions to analyze **large** data sets

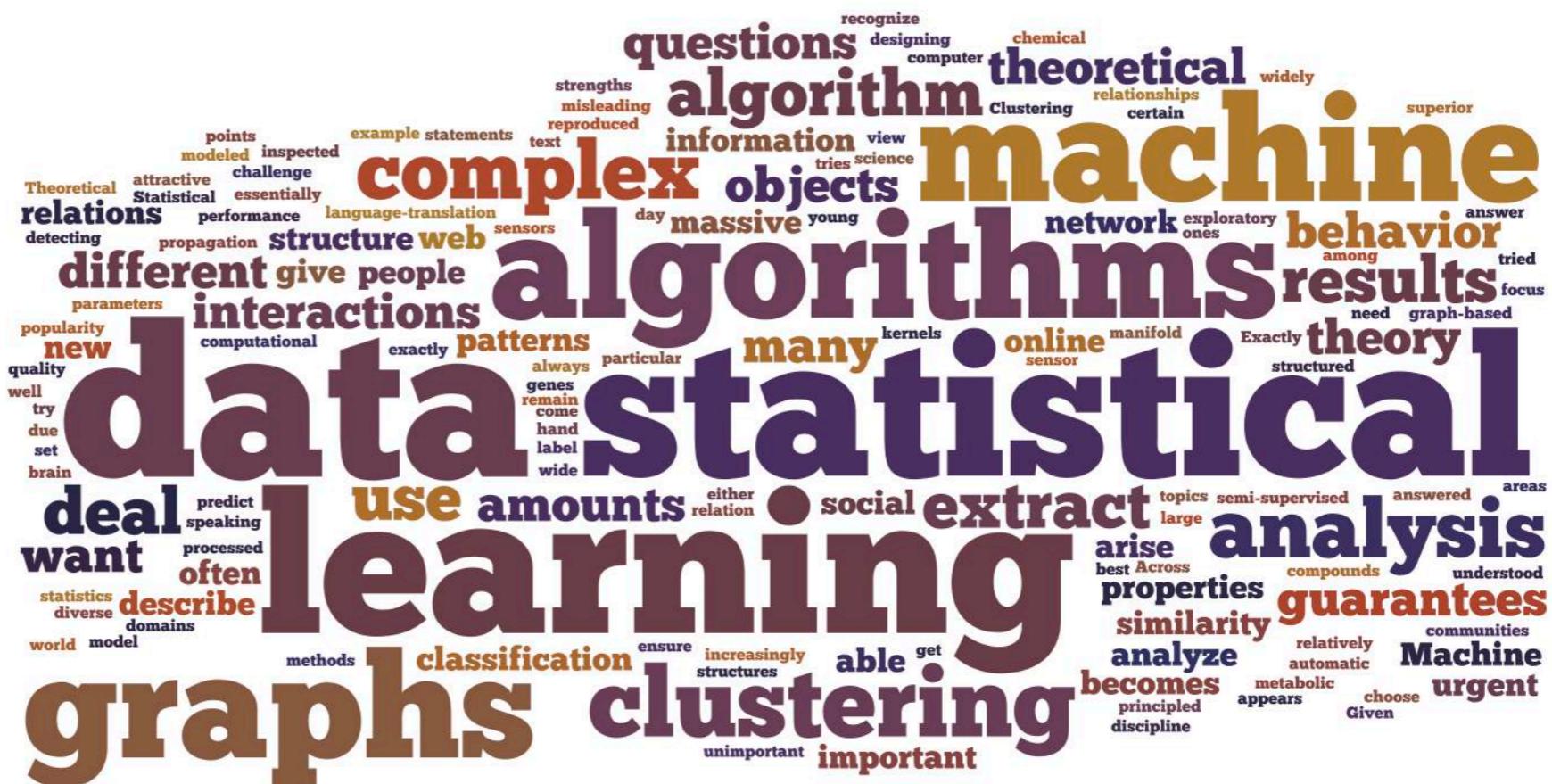
MACHINE LEARNING



არგებულის კუთხით

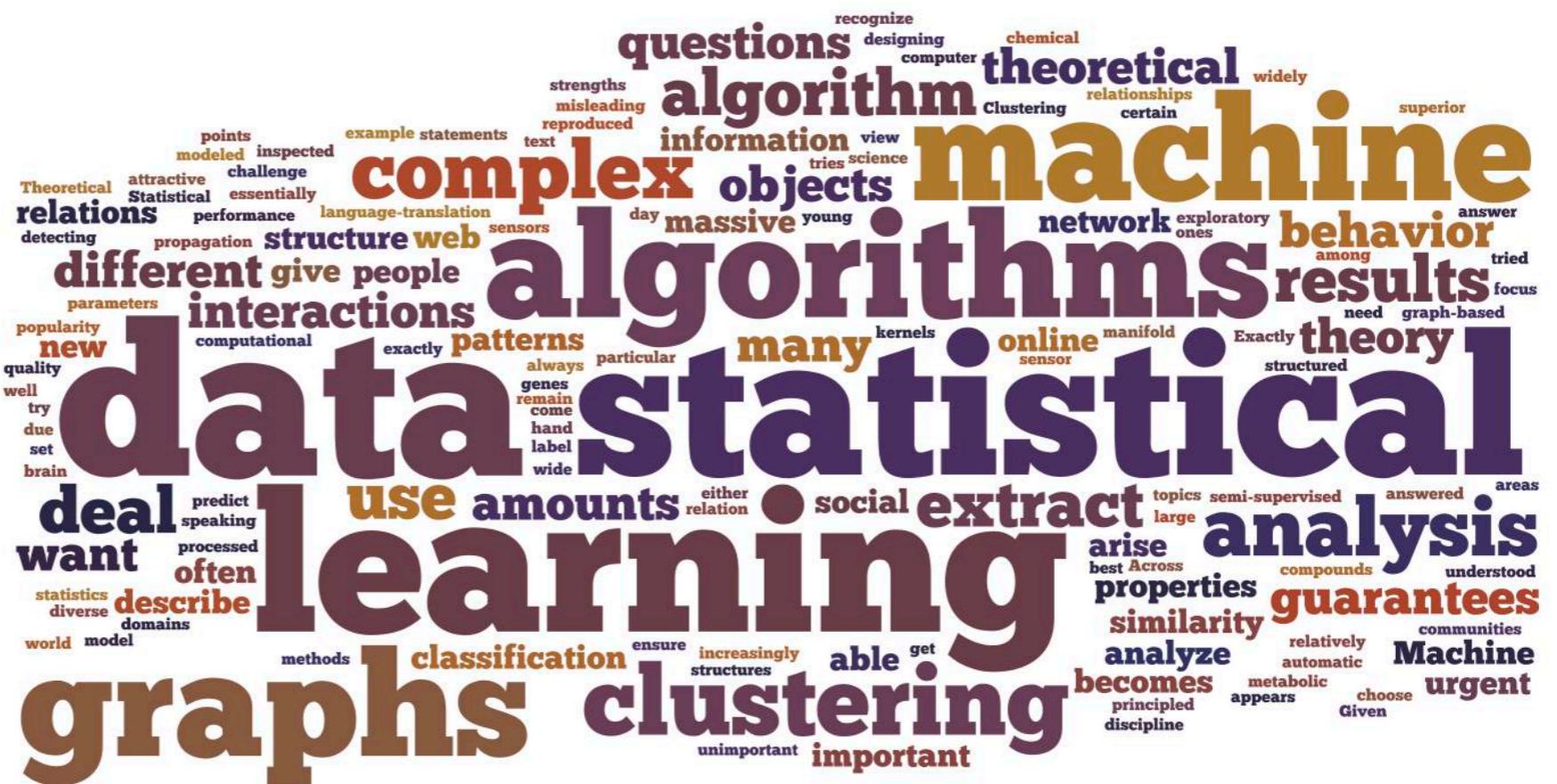
- Provides computational effective solutions to analyze **large** data sets
 - Provides solutions that can be **tailored** to different tasks/domains

MACHINE L E A R N I N G



- Provides computational effective solutions to analyze **large** data sets
 - Provides solutions that can be **tailored** to different tasks/domains
 - Requires many **efforts** in:
 - the definition of the **relevant information** best suited for the specific task/domain

MACHINE LEARNING



алгоритм
анализ
ресурса

- Provides *computational effective* solutions to analyze **large** data sets
- Provides solutions that can be **tailored** to different tasks/domains
- Requires many **efforts** in:
 - the definition of the **relevant information** best suited for the specific task/domain
 - the application of the **learning** algorithms to the considered data

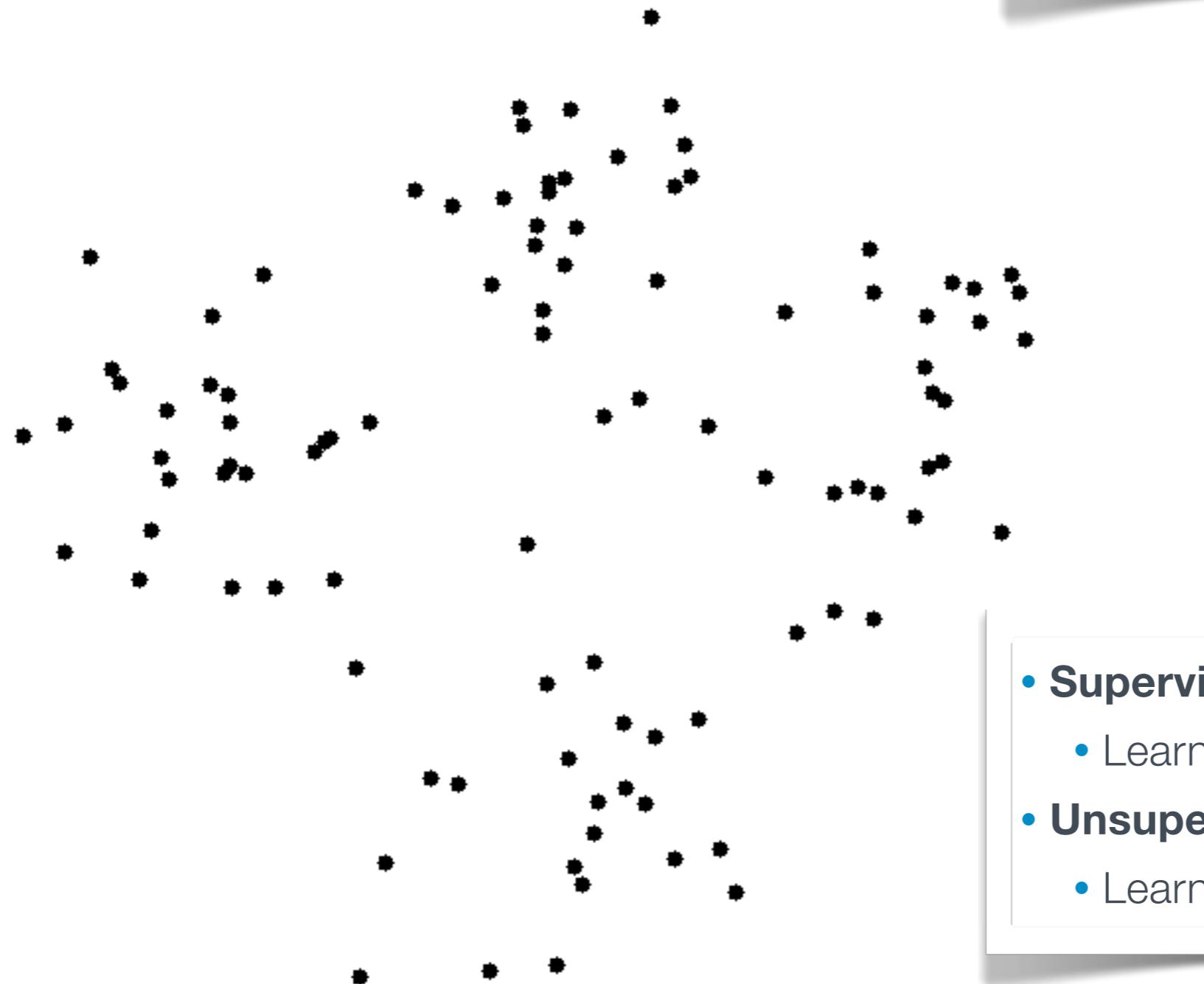
UNSUPERVISED LEARNING



Learn by examples

- **Supervised Learning:**
 - Learn from **labelled** samples
- **Unsupervised Learning:**
 - Learn (directly) from the data

UNSUPERVISED LEARNING



Learn by examples

- **Supervised Learning:**
 - Learn from **labelled** samples
- **Unsupervised Learning:**
 - Learn (directly) from the data



Learn by examples

- **Supervised Learning:**
 - Learn from **labelled** samples
- **Unsupervised Learning:**
 - Learn (directly) from the data

(+) **No cost** of labeling samples

(-) Trade-off imposed on the quality of the data

Reverse engineering &

THESES

CONTRIBUTIONS

Contributions to three relevant and related **open issues** in Software Maintenance

Reverse engineering

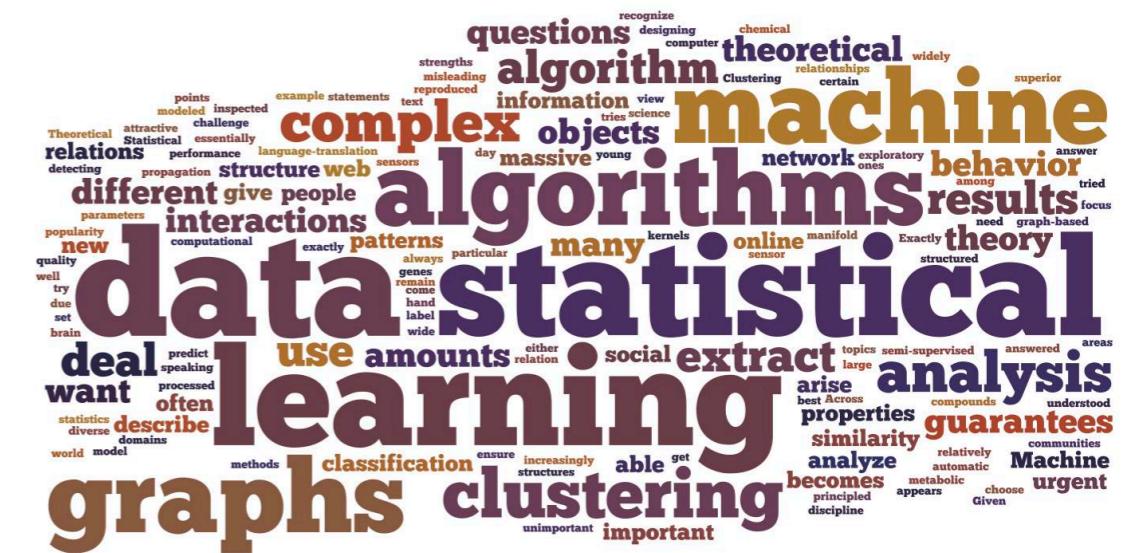
The figure is a word cloud centered around the term "algorithms". The words are arranged in a grid-like structure, with larger words representing more frequent terms. The colors of the words vary, creating a visual hierarchy. The most prominent words include "algorithms" (large, dark purple), "data" (large, dark blue), "statistical" (large, dark purple), "learning" (large, dark purple), "graphs" (large, dark purple), "clustering" (large, dark purple), "classification" (medium, orange), "use" (medium, orange), "amounts" (medium, orange), "either" (small, white), "relation" (small, white), "social" (small, white), "extract" (small, white), "topics" (small, white), "semi-supervised" (small, white), "large" (small, white), "arise" (small, white), "best" (small, white), "Across" (small, white), "properties" (small, white), "compounds" (small, white), "understood" (small, white), "analysis" (large, dark purple), "guarantees" (large, dark purple), "similarity" (small, white), "analyze" (small, white), "becomes" (small, white), "principled" (small, white), "discipline" (small, white), "metabolic" (small, white), "appears" (small, white), "choose" (small, white), "Given" (small, white), "Machine" (small, white), and "urgent" (small, white).

THESES

CONTRIBUTIONS

Contributions to three relevant and related **open issues** in Software Maintenance

Reverse engineering &



1. SOFTWARE RE-MODULARIZATION

2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

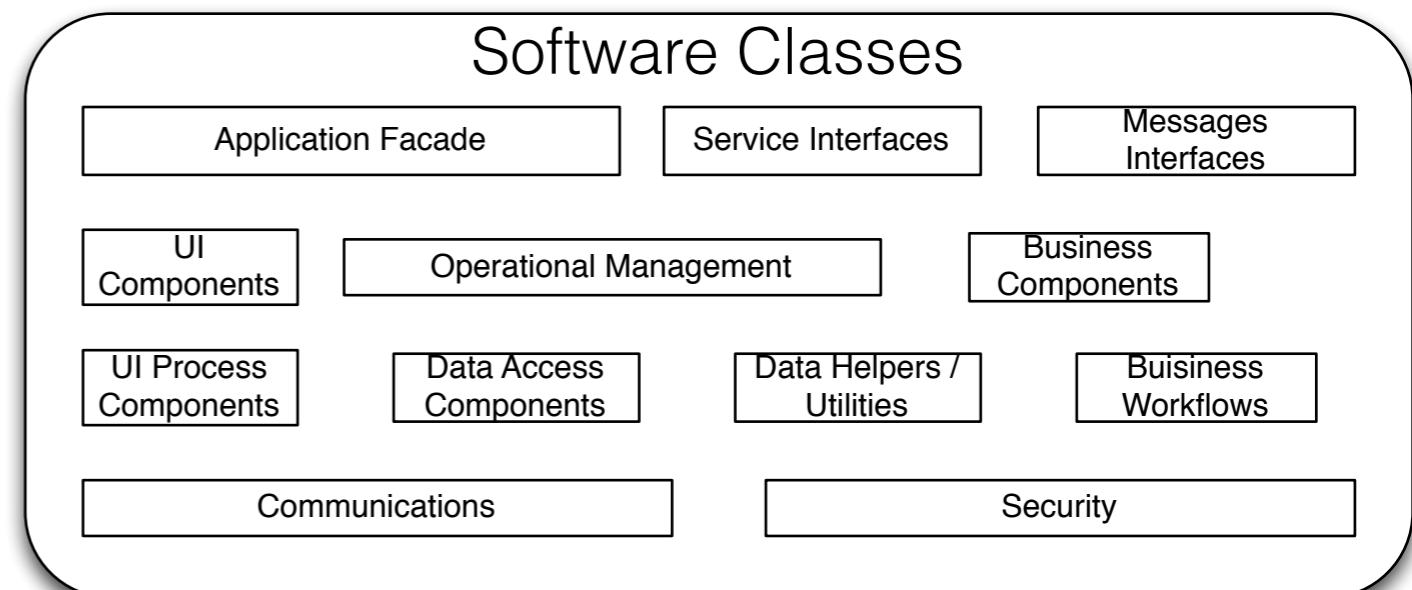
ISSUE
#1

SOFTWARE RE-MODULARIZATION

PROBLEM STATEMENT

Re-modularization provides a way to support software maintainers by automatically grouping together (**clustering**) “related” software **classes**

SOFTWARE RE-MODULARIZATION

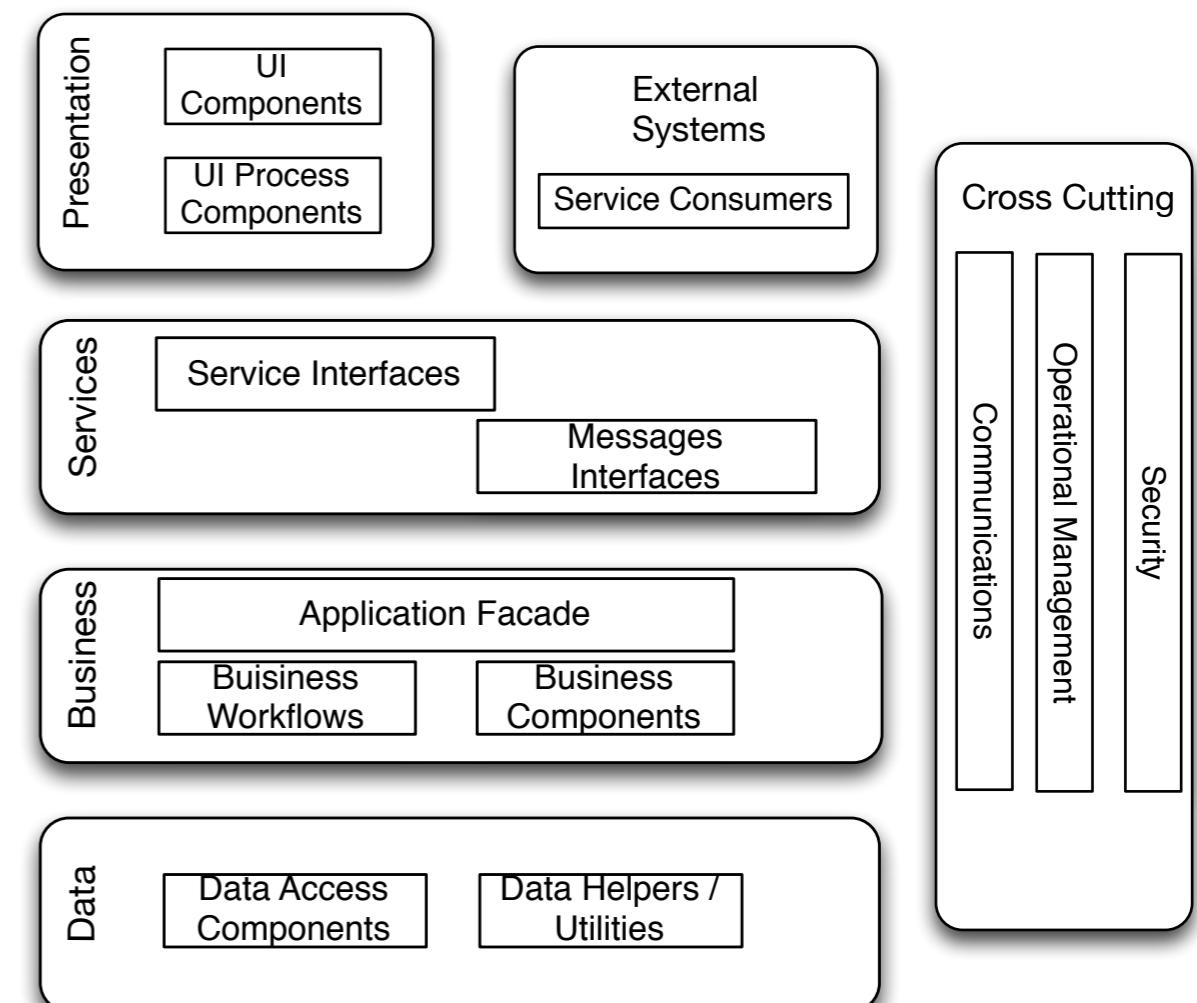


PROBLEM STATEMENT

Re-modularization provides a way to support software maintainers by automatically grouping together (**clustering**) “related” software **classes**

SOFTWARE RE-MODULARIZATION

Clusters of Software Classes



```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {

    /**
     * The handle's locator.
     */
    protected Locator fLocator;

    public NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    public void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL CONCEPTS

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {

    /**
     * The handle's locator.
     */
    protected Locator fLocator;

    public NullHandle(Figure owner, Locator locator) {
        super(owner, locator);
    }

    void draw(Owner owner, Graphics g, int x, int y, int width, int height) {
        Rectangle rect = new Rectangle(x, y, width, height);
        if (fLocator != null) {
            LocatorHandle handle = fLocator.getHandle();
            if (handle instanceof NullHandle) {
                handle.draw(owner, g, x, y, width, height);
            } else {
                handle.draw(owner, g, x, y, width, height);
            }
        } else {
            setColor(owner.getColor());
            drawRect(g, rect);
        }
    }

    void setColor(Color color) {
        g.setColor(color);
    }

    void drawRect(Graphics g, Rectangle rect) {
        g.drawRect(rect.x, rect.y, rect.width, rect.height);
    }
}
```

LEXICAL CONCEPTS

Z
O
T
A
N
R
E
L
Z

LEXICAL INFORMATION

SOURCE CODE

IR INDIAN EXPLORING

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

Implicit assumption: The “same” words are used whenever a particular concept occurs

1. Tokenization

Draws, the, are,
NullHandle,
box, r,
Rectangle, g,
Graphics,
box,
displayBox,
...

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

2. Normalization

draw, **the, are,**
null, handl,
box, **r,**
rectangl, **g,**
graphic,
box,
display, box,
...

Implicit assumption: The “same” words are used whenever a particular concept occurs

LEXICAL SOURCES CODE ZONES INFORMATION

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {

    /**
     * The handle's locator.
     */
    protected Locator fLocator;

    public NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    public void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL SOURCES CODE ZONES INFORMATION

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle { Class Names

    /**
     * The handle's locator.
     */
    protected Locator fLocator;

    public NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    public void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL SOURCES CODE ZONES INFORMATION

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle { Class Names

    /**
     * The handle's locator.
     */
    protected Locator fLocator; Attribute Names

    public NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    public void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL SOURCES CODE ZONES INFORMATION

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle { Class Names

    /**
     * The handle's locator.
     */
    protected Locator fLocator; Attribute Names

    Method Names    NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    Method Names    void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL SOURCES CODE ZONES INFORMATION

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle { Class Names

    /**
     * The handle's locator.
     */
    protected Locator fLocator; Attribute Names

    Method Names NullHandle(figure owner, locator locator) {
        super(owner, locator); Parameter Names
    }

    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    Method Names void draw(Graphics g) {
        Rectangle r = displayBox();

        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL ZONES

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {
```

Comments

```
    /**
     * The handle's locator.
     */
    protected Locator fLocator;
```

Attribute Names

Method Names

```
    NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }
```

Parameter Names

```
    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    void draw(Graphics g) {
        Rectangle r = displayBox();
```

Comments

```
        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

LEXICAL ZONES

```
/*
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {
```

[Comments](#) [Class Names](#)

```
    /**
     * The handle's locator.
     */
    protected Locator fLocator;
```

[Comments](#) [Attribute Names](#)

[Method Names](#) [Parameter Names](#) [Source Code](#)

```
    NullHandle(figure owner, Locator locator) {
        super(owner, locator);
    }
```

[Parameter Names](#)

```
    /**
     * Draws the NullHandle. NullHandles are drawn as a
     * red framed rectangle.
     */
    void draw(Graphics g) {
        Rectangle r = displayBox();
```

[Comments](#) [Parameter Names](#)

```
        g.setColor(Color.black);
        g.drawXORRect(r);
    }
}
```

[Source Code](#)

ZONE

ENDING
INDEX

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

ZONE

INDEXING

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

RQ1: Do terms in different **Zones** provide different **contributions**?

ZONE ZONE

1. Tokenization

Draws, the, are,
NullHandle,
box, r,
draw, g,
Graphics,
color,
displayBox,
...

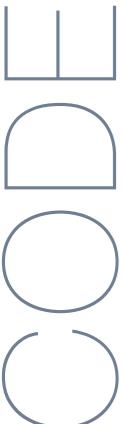
```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

2. Normalization

draw, the, are,
null, handl,
box, r,
draw, g,
graphic,
color,
display, box,
...

RQ1: Do terms in different **Zones** provide different **contributions**?

LEXICON



JFreeChart

```
/*
 * Creates a vertical bar chart with default settings.
 * @param title The chart title.
 * @param categoryAxisLabel The label for the category axis.
 * @param valueAxisLabel The label for the value axis.
 * @param data The dataset for the chart.
 * @param legend A flag specifying whether or not a legend is required.
 */
public static JFreeChart createVerticalBarChart(String title, String categoryAxisLabel,
                                                String valueAxisLabel, CategoryDataset data,
                                                boolean legend) {

    JFreeChart chart = null;

    try {
        Axis categoryAxis = new HorizontalCategoryAxis(categoryAxisLabel);
        Axis valueAxis = new VerticalNumberAxis(valueAxisLabel);
        Plot plot = new VerticalBarPlot(categoryAxis, valueAxis);
        chart = new JFreeChart(data, plot, title, JFreeChart.DEFAULT_TITLE_FONT, legend);
    } catch (AxisNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : axis not compatible.");
    } catch (PlotNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : plot not compatible.");
    }
    return chart;
}
```

JUnit

```
public class TestListenerTest extends TestCase implements TestListener {
    private TestResult fResult;
    private int fStartCount;
    private int fEndCount;
    private int fFailureCount;
    private int fErrorCount;

    public void testFailure() {
        TestCase test= new TestCase("noop") {
            @Override
            public void runTest() {
                fail();
            }
        };
        test.run(fResult);
        assertEquals(1, fFailureCount);
        assertEquals(1, fEndCount);
    }
    public void testStartStop() {
        TestCase test= new TestCase("noop") {
            @Override
            public void runTest() {
            }
        };
        test.run(fResult);
        assertEquals(1, fStartCount);
        assertEquals(1, fEndCount);
    }
}
```

JEdit

```
/*
 * Visits a local variable instruction. A local variable instruction is an
 * instruction that loads or stores the value of a local variable.
 *
 * @param opcode the opcode of the local variable instruction to be visited.
 * This opcode is either ILOAD, LLOAD, FLOAD, DLOAD, ALOAD, ISTORE,
 * LSTORE, FSTORE, DSTORE, ASTORE or RET.
 * @param var the operand of the instruction to be visited. This operand is
 * the index of a local variable.
 */
void visitVarInsn (int opcode, int var);

/*
 * Visits a type instruction. A type instruction is an instruction that
 * takes a type descriptor as parameter.
 *
 * @param opcode the opcode of the type instruction to be visited. This opcode
 * is either NEW, ANEWARRAY, CHECKCAST or INSTANCEOF.
 * @param desc the operand of the instruction to be visited. This operand is
 * must be a fully qualified class name in internal form, or the type
 * descriptor of an array type (see {@link Type Type}).
 */
void visitTypeInsn (int opcode, String desc);
```

Xerces

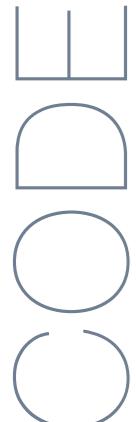
```
/Returns a string representation of the specified content spec
public static String toString(XMLContentSpec.Provider pr, StringPool strPool,
                             int contSpecInd) {

    // lookup content spec node
    XMLContentSpec contentSpec = new XMLContentSpec();

    if (pr.getContentSpec(contSpecInd, contentSpec)) {

        // build string
        StringBuffer stringBuffer = new StringBuffer();
        int parentContentSpecType = contentSpec.type & 0x0f;
        int nextContentSpec;
        switch (parentContentSpecType) {
            case XMLContentSpec.CONTENTSPECNODE_LEAF: {
                stringBuffer.append('(');
                if (contentSpec.value == -1 && contentSpec.otherValue == -1) {
                    stringBuffer.append("#PCDATA");
                }
                else {
                    stringBuffer.append(strPool.toString(contentSpec.value));
                }
                stringBuffer.append(')');
                break;
            }
            case XMLContentSpec.CONTENTSPECNODE_ZERO_OR_ONE: {
                pr.getContentSpec(contentSpec.value, contentSpec);
                nextContentSpec = contentSpec.type;
            }
        }
    }
}
```

LEXICON



OUR
ZONES

JFreeChart

This is a local variable instruction. A local variable instruction is an instruction that loads or stores the value of a local variable.

* @param opcode the opcode of the local variable instruction to be visited.
* This opcode is either ILOAD, LLOAD, FLOAD, DLOAD, ALOAD, ISTORE,
* LSTORE, FSTORE, DSTORE, ASTORE or RET.

```
/*
 * Creates a vertical bar chart with default settings.
 * @param title The chart title.
 * @param categoryAxisLabel The label for the category axis.
 * @param valueAxisLabel The label for the value axis.
 * @param data The dataset for the chart.
 * @param legend A flag specifying whether or not a legend is required.
 */
public static JFreeChart createVerticalBarChart(String title, String categoryAxisLabel,
                                                String valueAxisLabel, CategoryDataset data,
                                                boolean legend) {

    JFreeChart chart = null;

    try {
        Axis categoryAxis = new HorizontalCategoryAxis(categoryAxisLabel);
        Axis valueAxis = new VerticalNumberAxis(valueAxisLabel);
        Plot plot = new VerticalBarPlot(categoryAxis, valueAxis);
        chart = new JFreeChart(data, plot, title, JFreeChart.DEFAULT_TITLE_FONT, legend);
    } catch (AxisNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : axis not compatible.");
    } catch (PlotNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : plot not compatible.");
    }
    return chart;
}

public void test() {
    @Override
    public void runTest() {
    };
    test.run(fResult);
    assertEquals(1, fStartCount);
    assertEquals(1, fEndCount);
}
}
```

```
else if (contentSpec.type == -1) {
    stringBuffer.append(strPool.toString(contentSpec.value));
}
stringBuffer.append(')');
break;
} case XMLContentSpec.CONTENTSPECNODE_ZERO_OR_ONE: {
    ContentSpec(contentSpec.value, contentSpec);
    contentSpec = contentSpec.type;
}
}
```

- Good lexicon in **every Zone!**

LEXICON

SOURCE

- Very good in **Comments**
 - Very poor in **Method** and **Parameter** names

LEXICON

L
E
X
I
C
O
N

U
R
E
S

```
/* Creates a vertical bar chart with default settings.  
 * @param title The chart title.  
 * @param categoryAxisLabel The label for the category axis.  
 * @param valueAxisLabel The label for the value axis.  
 * @param data The dataset.  
 * @param legend A flag specifying whether to include a legend.  
 */  
public static JFreeChart createBarChart(
```

```
    JFreeChart chart = null;  
    try {  
        Axis categoryAxis = new CategoryAxis("Category");  
        Axis valueAxis = new ValueAxis("Value");  
        Plot plot = new VerticalBarPlot();  
        chart = new JFreeChart("Bar Chart", categoryAxis, valueAxis, plot);  
    } catch (AxisNotComparableException e) {  
        // this won't happen  
        System.err.println("Caught exception: " + e);  
    } catch (PlotNotComparableException e) {  
        // this won't happen  
        System.err.println("Caught exception: " + e);  
    }  
    return chart;
```

private TestResult fResult;
private int fStartCount;
private int fEndCount;
private int fFailureCount;
private int fErrorCount;

```
public void testFailure() {  
    TestCase test = new TestCase("noop") {  
        @Override  
        public void runTest() {  
            fail();  
        }  
    };  
    test.run(fResult);  
    assertEquals(1, fFailureCount);  
    assertEquals(1, fEndCount);  
}  
public void testStartStop() {  
    TestCase test = new TestCase("noop") {  
        @Override  
        public void runTest() {  
        }  
    };  
    test.run(fResult);  
    assertEquals(1, fStartCount);  
    assertEquals(1, fEndCount);  
}
```

JUnit

Visits a local variable instruction. A local variable instruction is an instruction that loads or stores the value of a local variable.

tion to be visited.
ALOAD, ISTORE,

1. This operand is

struction that

visited. This opcode

ed. This operand is
form, or the type
).

spec
StringPool strPool,

0x0f;

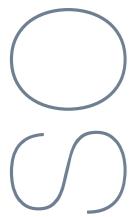
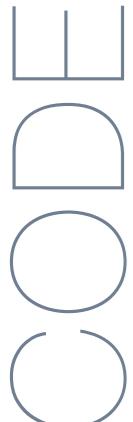
otherValue == -1) {

contentSpec.value));

CONTENTSPECNODE_ZERO_OR_ONE: {
 contentSpec.value, contentSpec);
 contentSpec.type;

- Good in **Class** and **Attribute** names
- No **Comments** at all

LEXICON



Xerces

```
/* Creates a vertical bar chart with default settings.  
 * @param title The chart title.  
 * @param categoryAxisLabel The label for the category axis.  
 * @param valueAxisLabel The label for the value axis.  
 * @param data The dataset for the chart.  
 * @param legend A flag specifying whether or not a legend is required.  
 */
```

```
public static JFreeChart
```

```
    JFreeChart chart = nu  
    try {  
        Axis categoryAxis  
        Axis valueAxis =  
        Plot plot = new V  
        chart = new JFree  
    } catch (AxisNotCompa  
        // this won't hap  
        System.err.println("C  
    } catch (PlotNotCompa  
        // this won't hap  
        System.err.println("C  
    }  
    return chart;
```

```
public class Test  
private TestR  
private int f  
private int f  
private int f  
private int f  
  
public void t  
    TestCase  
    @Over  
    publi  
        f  
    }  
};  
test.runC  
assertEqu  
assertEqu  
}  
public void t  
    TestCase  
    @Over  
    ---  
    public void runTest() {
```

```
// Returns a string representation of the specified content spec  
public static String toString(XMLContentSpec.Provider pr, StringPool strPool,  
                                int contSpecInd) {
```

```
// lookup content spec node
```

```
XMLContentSpec contentSpec = new XMLContentSpec();
```

```
if (pr.getContentSpec(contSpecInd, contentSpec)) {
```

```
// build string
```

```
StringBuffer stringBuffer = new StringBuffer();
```

```
int parentContentSpecType = contentSpec.type & 0x0f;
```

```
int nextContentSpec;
```

```
switch (parentContentSpecType) {
```

```
case XMLContentSpec.CONTENTSPECNODE_LEAF: {
```

```
    stringBuffer.append('(');
```

```
    if (contentSpec.value == -1 && contentSpec.otherValue == -1) {  
        stringBuffer.append("#PCDATA");
```

```
    }
```

```
    else {
```

```
        stringBuffer.append(strPool.toString(contentSpec.value));
```

```
    }
```

```
    stringBuffer.append(')');
```

```
    break;
```

```
} case XMLContentSpec.CONTENTSPECNODE_ZERO_OR_ONE: {
```

```
    pr.getContentSpec(contentSpec.value, contentSpec);
```

```
    nextContentSpec = contentSpec.type;
```

```
}
```

- Poor in **Method, Parameter** names and **Comments**
- Good in Method **Code** and **Variable** names

JFreeChart

```


    * Creates a vertical bar chart with default settings.
    * @param title The chart title.
    * @param categoryAxisLabel The label for the category axis.
    * @param valueAxisLabel The label for the value axis.
    * @param data The dataset for the chart.
    * @param legend A flag specifying whether or not a legend is required.
    */
public static JFreeChart createVerticalBarChart(String title, String categoryAxisLabel,
                                                String valueAxisLabel, CategoryDataset data,
                                                boolean legend) {

    JFreeChart chart = null;

    try {
        Axis categoryAxis = new HorizontalCategoryAxis(categoryAxisLabel);
        Axis valueAxis = new VerticalNumberAxis(valueAxisLabel);
        Plot plot = new VerticalBarPlot(categoryAxis, valueAxis);
        chart = new JFreeChart(data, plot, title, JFreeChart.DEFAULT_TITLE_FONT, legend);
    } catch (AxisNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : axis not compatible.");
    } catch (PlotNotCompatibleException e) {
        // this won't happen unless you mess with the axis constructors above
        System.err.println("ChartFactory.createVerticalBarChart(...) : plot not compatible.");
    }
    return chart;
}


```

JEdit

```


    * Visits a local variable instruction. A local variable instruction is an
    * instruction that loads or stores the value of a local variable.

    * @param opcode the opcode of the local variable instruction to be visited.
    * This opcode is either ILOAD, LLOAD, FLOAD, DLOAD, ALOAD, ISTORE,
    * LSTORE, FSTORE, DSTORE, ASTORE or RET.
    * @param var the operand of the instruction to be visited. This operand is
    * the index of a local variable.
    */

void visitVarInsn (int opcode, int var);

/*
    * Visits a type instruction. A type instruction is an instruction that
    * takes a type descriptor as parameter.
    *
    * @param opcode the opcode of the type instruction to be visited. This opcode
    * is either NEW, ANEWARRAY, CHECKCAST or INSTANCEOF.
    * @param desc the operand of the instruction to be visited. This operand is
    * must be a fully qualified class name in internal form, or the type
    * descriptor of an array type (see {@link Type Type}).
    */

void visitTypeInsn (int opcode, String desc);


```

RQ2: How to automatically **weight** the different contributions ?

```


private TestResult fResult;
private int fStartCount;
private int fEndCount;
private int fFailureCount;
private int fErrorCount;

public void testFailure() {
    TestCase test= new TestCase("noop") {
        @Override
        public void runTest() {
            fail();
        }
    };
    test.run(fResult);
    assertEquals(1, fFailureCount);
    assertEquals(1, fEndCount);
}
public void testStartStop() {
    TestCase test= new TestCase("noop") {
        @Override
        public void runTest() {
        }
    };
    test.run(fResult);
    assertEquals(1, fStartCount);
    assertEquals(1, fEndCount);
}


```

```


// Returns a string representation of the specified content spec
public static String toString(XMLContentSpec.Provider pr, StringPool strPool,
                               int contSpecInd) {

    // lookup content spec node
    XMLContentSpec contentSpec = new XMLContentSpec();

    if (pr.getContentSpec(contSpecInd, contentSpec)) {

        // build string
        StringBuffer stringBuffer = new StringBuffer();
        int parentContentSpecType = contentSpec.type & 0x0f;
        int nextContentSpec;
        switch (parentContentSpecType) {
            case XMLContentSpec.CONTENTSPECNODE_LEAF: {
                stringBuffer.append('(');
                if (contentSpec.value == -1 && contentSpec.otherValue == -1) {
                    stringBuffer.append("#PCDATA");
                }
                else {
                    stringBuffer.append(strPool.toString(contentSpec.value));
                }
                stringBuffer.append(')');
                break;
            }
            case XMLContentSpec.CONTENTSPECNODE_ZERO_OR_ONE: {
                pr.getContentSpec(contentSpec.value, contentSpec);
                nextContentSpec = contentSpec.type;
            }
        }
    }
}


```

SOFTWARE RE-MODULARIZATION

- We propose a source code **Zone** Indexing

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

Combining machine learning and information retrieval techniques for software clustering

(2012) Communications in Computer and Information Science, 255 CCIS, pp. 42-60. **ISSN:** 18650929 **ISBN:** 978-364228032-0

Corazza A., Di Martino, S., Maggio, V., Scanniello, G.

Investigating the use of lexical information for software system clustering

(2011) Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, art. no. 5741257, pp. 35-44.

ISSN: 15345351 **ISBN:** 978-076954343-7

SOFTWARE RE-MODULARIZATION

- We propose a source code **Zone** Indexing
- An approach to automatically **weight** the contribution of different **terms** in **Zones**
- *Maximum-Likelihood Estimation (**MLE**) Approach*

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

Combining machine learning and information retrieval techniques for software clustering

(2012) Communications in Computer and Information Science, 255 CCIS, pp. 42-60. **ISSN:** 18650929 **ISBN:** 978-364228032-0

Corazza A., Di Martino, S., Maggio, V., Scanniello, G.

Investigating the use of lexical information for software system clustering

(2011) Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, art. no. 5741257, pp. 35-44.

ISSN: 15345351 **ISBN:** 978-076954343-7

SOFTWARE RE-MODULARIZATION

- We propose a source code **Zone** Indexing
- An approach to automatically **weight** the contribution of different **terms** in **Zones**
 - *Maximum-Likelihood Estimation (MLE)* Approach
 - A variant of the **K-medoid** clustering algorithm
 - Tailored to the **software clustering** task

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

Combining machine learning and information retrieval techniques for software clustering

(2012) Communications in Computer and Information Science, 255 CCIS, pp. 42-60. **ISSN:** 18650929 **ISBN:** 978-364228032-0

Corazza A., Di Martino, S., Maggio, V., Scanniello, G.

Investigating the use of lexical information for software system clustering

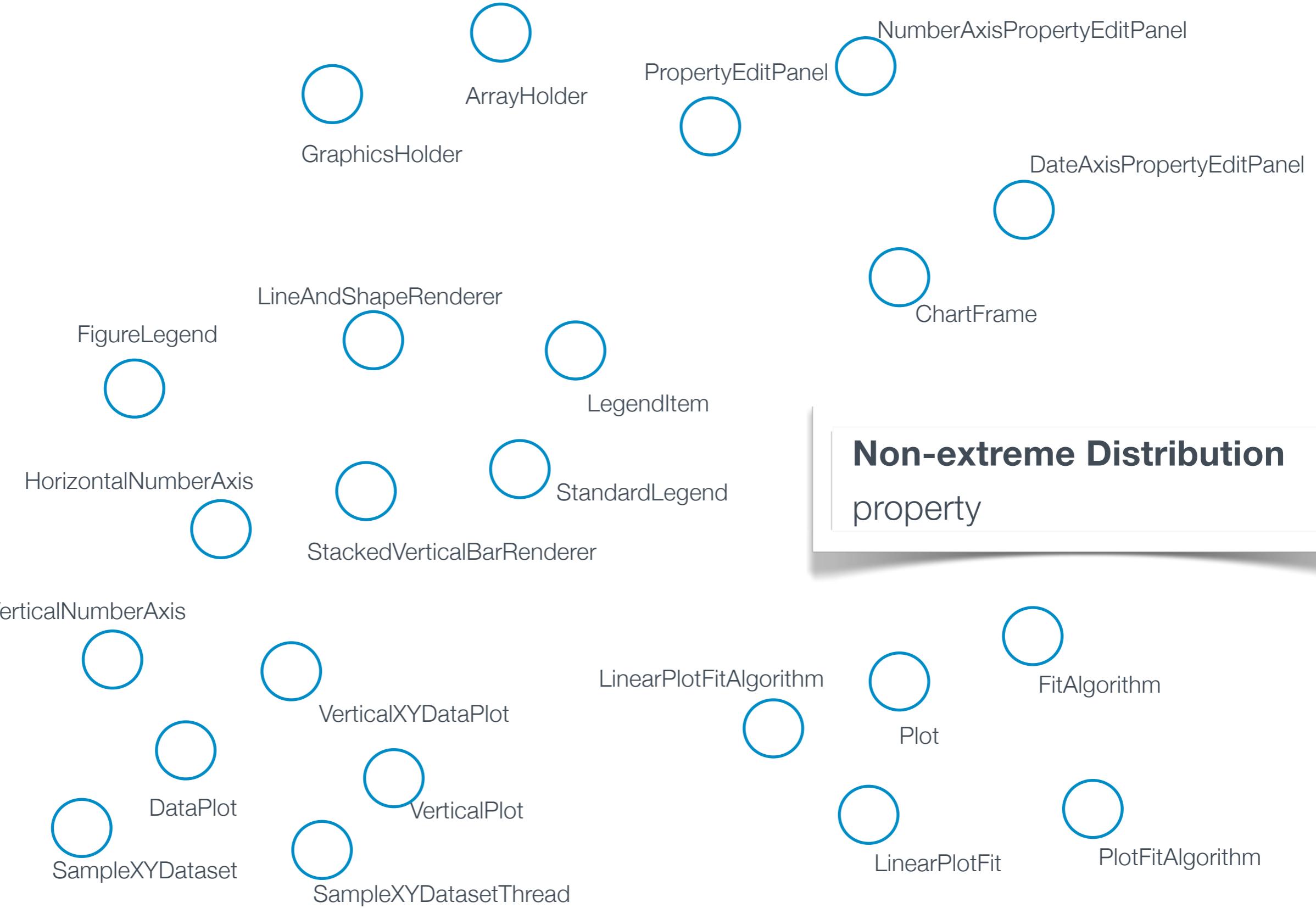
(2011) Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, art. no. 5741257, pp. 35-44.

ISSN: 15345351 **ISBN:** 978-076954343-7

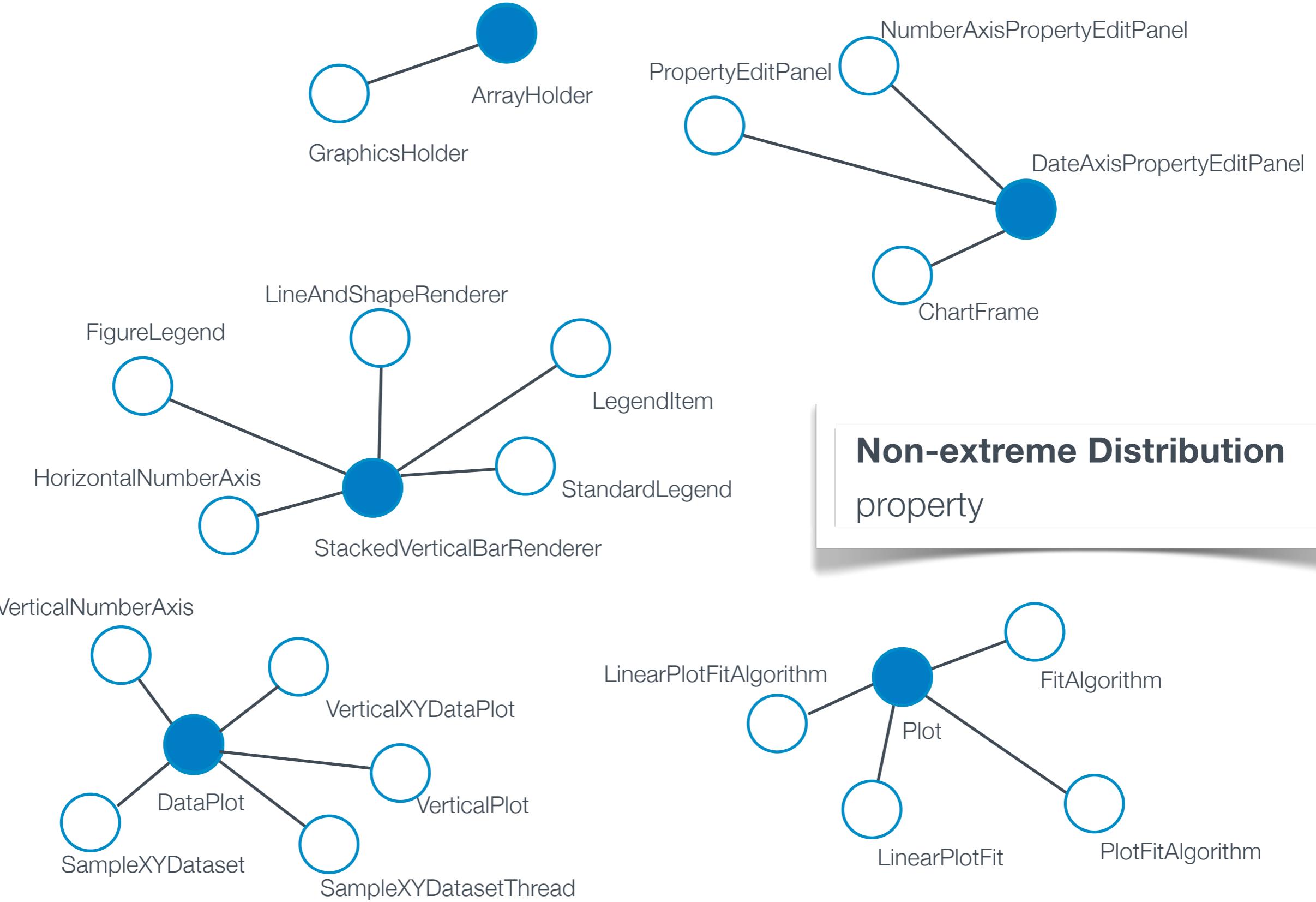
K-MEDOID VARIANT

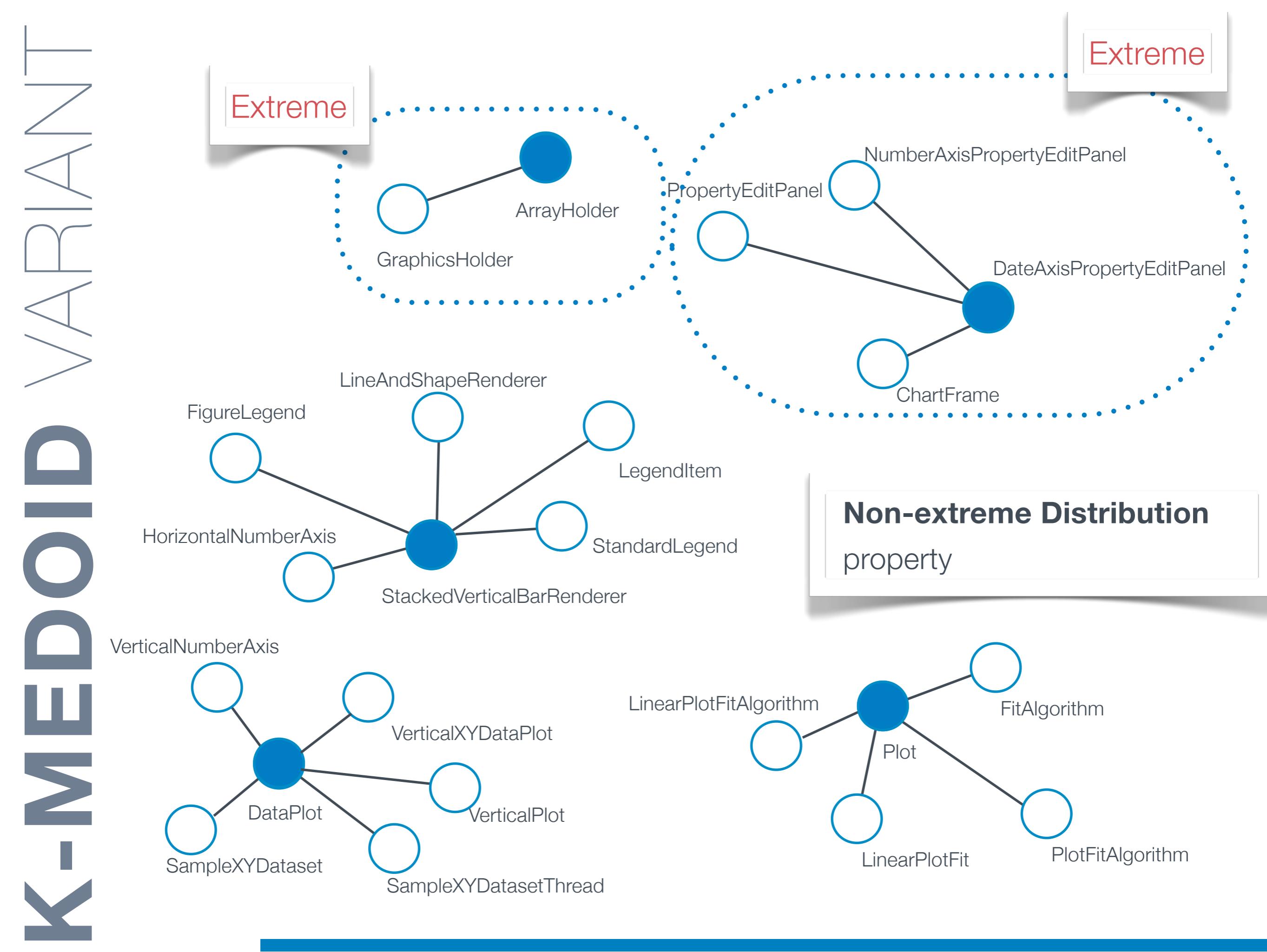


K-MEDOID VARIANT



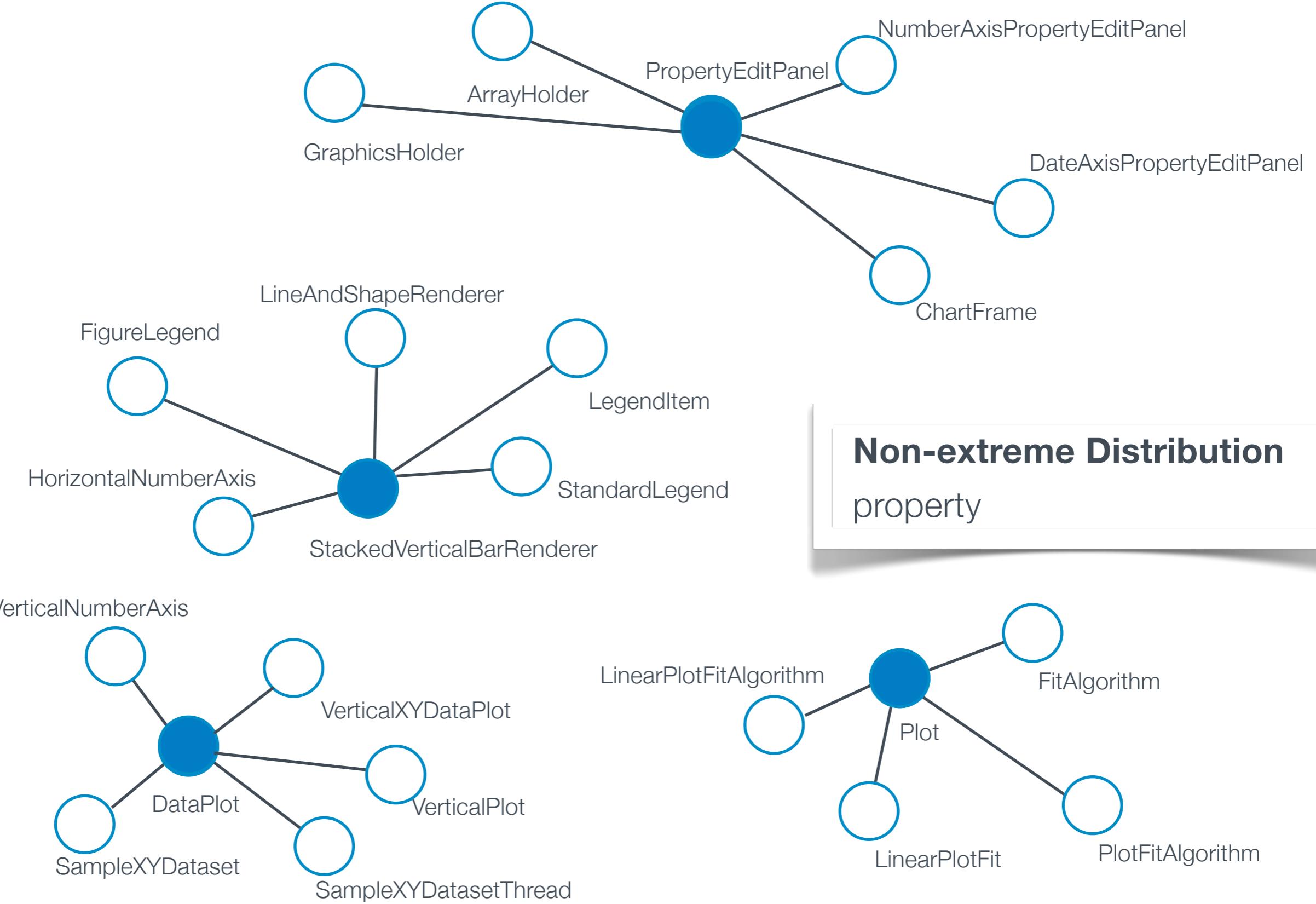
K-MEDOID VARIANT





K-MEDOID

VARIANT



RE SULTS

AUTHORITATIVENESS

Comparison of Clustering Results with an **Authoritative Target Partition**

19 target Open Source Java Systems

FLAT INDEXING (NO ZONES) = **STATE OF THE ART**



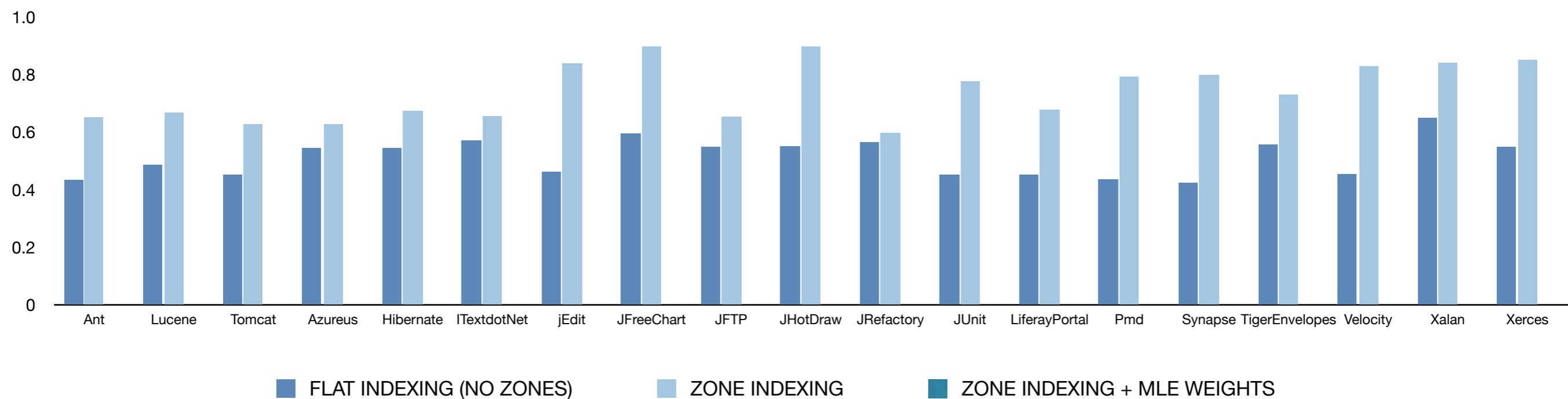
RE SULTS

AUTHORITATIVENESS

Comparison of Clustering Results with an **Authoritative Target Partition**

19 target Open Source Java Systems

FLAT INDEXING (NO ZONES) = **STATE OF THE ART**



RESULTS

AUTHORITATIVENESS

Comparison of Clustering Results with an **Authoritative Target Partition**

19 target Open Source Java Systems

FLAT INDEXING (NO ZONES) = **STATE OF THE ART**



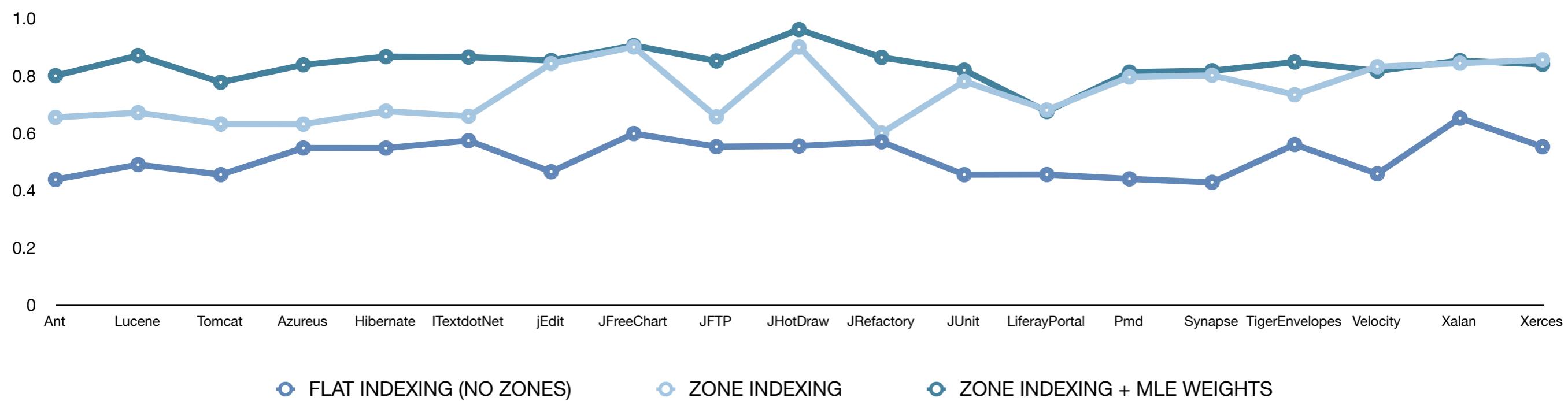
RE SULTS

AUTHORITATIVENESS

Comparison of Clustering Results with an **Authoritative Target Partition**

19 target Open Source Java Systems

FLAT INDEXING (NO ZONES) = **STATE OF THE ART**



ISSUE #2

SOURCE CODE
NORMALIZATION

LEXICAL CONCEPTS

SOURCE CODE ONE HUNDRED
LEXICAL LEVEL

STATE OF THE ART TOOLS

1. Tokenization

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

1.5 Identifier Splitting

2. Normalization

- *Splitting algorithms based on naming conventions*
- **camelCase** Splitter: `r' (?<=?^)([A-Z][a-z]+)'`
 - NullHandle ==> Null | Handle
 - displayBox ==> display | Box

draw, **the**, **are**,
null, handl
box, r,
rectangl, g,
graphic,
color,
display, box
....

STATE OF THE ART TOOLS

1. Tokenization

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

1.5 Identifier Splitting

2. Normalization

- *Splitting algorithms based on naming conventions*
- **camelCase** Splitter: `r' (?<=?^)([A-Z][a-z]+)'`
 - NullHandle ==> Null | Handle
 - displayBox ==> display | Box

draw, **the**, **are**,
null, handl
box, r,
rectangl, g,
graphic,
color,
display, box
....

STATE OF THE ART TOOLS

1. Tokenization

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

1.5 Identifier Splitting

2. Normalization

- *Splitting algorithms based on naming conventions*
- **camelCase** Splitter: `r' (?<=?^)([A-Z][a-z]+)'`
 - NullHandle ==> Null | Handle

draw, **the**, **are**,
null, **handl**
box, r,
rectangl, g,
graphic,
color,
display, box
....

STATE OF THE ART TOOLS

1. Tokenization

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

1.5 Identifier Splitting

2. Normalization

- *Splitting algorithms based on naming conventions*
- **camelCase** Splitter: `r' (?=<!^)([A-Z][a-z]+)' display, box`
 - NullHandle ==> Null | Handle
 - displayBox ==> display | Box

draw, **the**, **are**,
null, **handl**
box, r,
rectangl, g,
graphic,
color,
....

SPLITTING

IDENTIFIERS

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

- **camelCase** Splitter: `r' (?=&!=^) ([A-Z][a-z]+) '`

SPLITTING

IDENTIFIERS

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

- **camelCase** Splitter: `r' (?<=!^) ([A-Z][a-z]+) '`
- `drawXORRect ==> drawXOR | Rect`

SPLITTING SHELLS IDENTIFIERS

*Splitting algorithms based on naming conventions
are not robust enough*

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

- **camelCase** Splitter: `r' (?<=!^) ([A-Z][a-z]+) '`
 - `drawXORRect` ==> `drawXOR | Rect`
 - `drawxorrect` ==> NO SPLIT

ABBREVIATIONS

EXPANSION

*Splitting algorithms based on naming conventions
are not robust enough*

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

*Splitting algorithms based on naming conventions
are not robust enough*

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

- Heavy use of **Abbreviations** in the source code
- r as for Rectangle
- rect as for Rectangle

ABBREVIATIONS

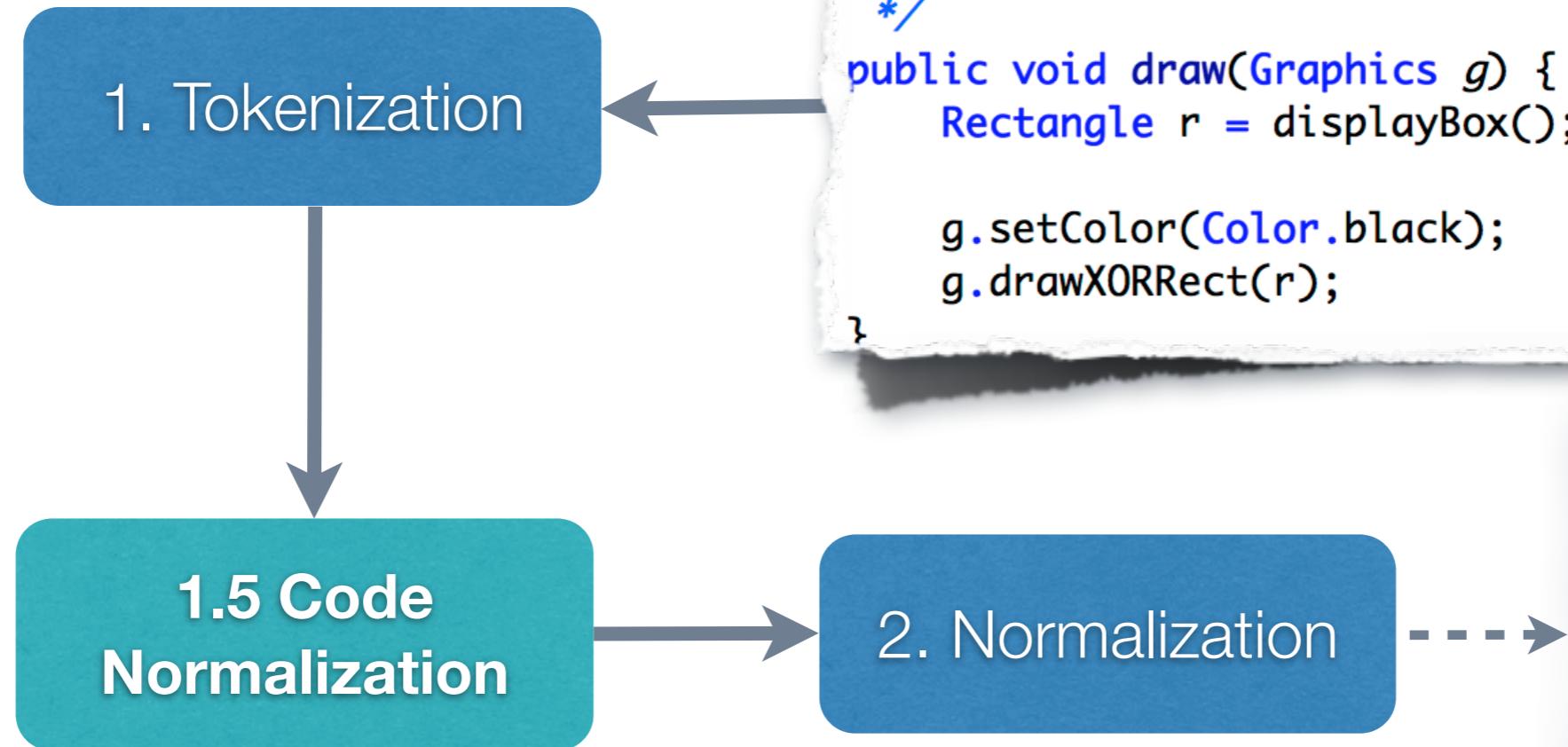
EXPANSION

*Splitting algorithms based on naming conventions
are not robust enough*

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

- Heavy use of **Abbreviations** in the source code
- r as for Rectangle
- rect as for Rectangle

CODE NORMALIZATION

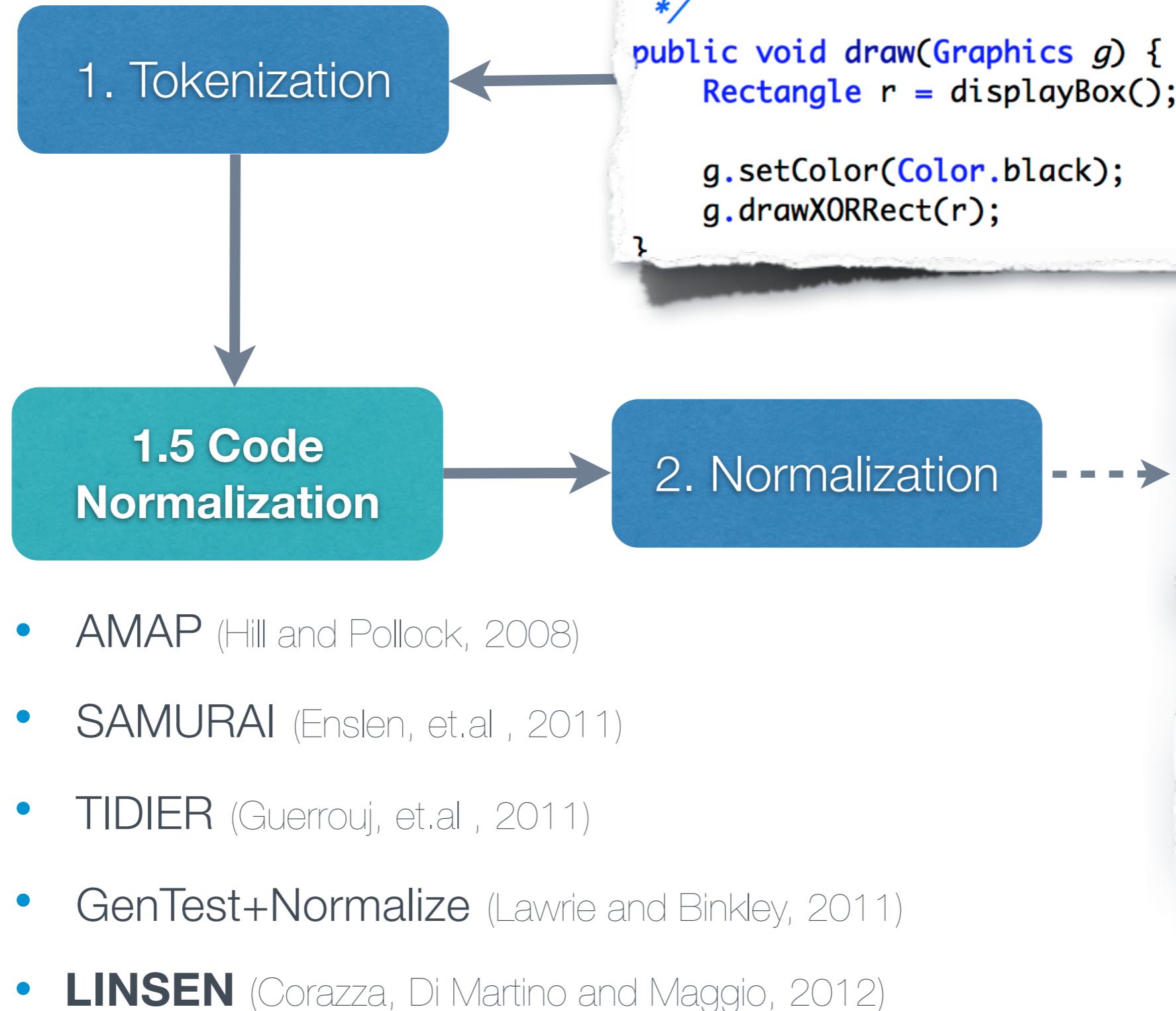


- AMAP (Hill and Pollock, 2008)
- SAMURAI (Enslen, et.al , 2011)
- TIDIER (Guerrouj, et.al , 2011)
- GenTest+Normalize (Lawrie and Binkley, 2011)
- **LINSEN** (Corazza, Di Martino and Maggio, 2012)

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

draw, ~~the~~, ~~are~~,
null, handl,
box, rectangl,
rectangl,
graphic,
color,
display, box,
draw, xor,
rectangl

CODE NORMALIZATION



- AMAP (Hill and Pollock, 2008)
- SAMURAI (Enslen, et.al , 2011)
- TIDIER (Guerrouj, et.al , 2011)
- GenTest+Normalize (Lawrie and Binkley, 2011)
- LINSEN (Corazza, Di Martino and Maggio, 2012)

```
/**  
 * Draws the NullHandle. NullHandles are drawn as a  
 * red framed rectangle box.  
 */  
public void draw(Graphics g) {  
    Rectangle r = displayBox();  
  
    g.setColor(Color.black);  
    g.drawXORRect(r);  
}
```

SOURCE CODE NORMALIZATION

- **LINSEN:** novel technique for code normalization
- Able to **both** *Split Identifiers* and *Expand* possible occurring abbreviations

Corazza, A., Di Martino, S., Maggio, V.

LINSEN: An efficient approach to split identifiers and expand abbreviations

(2012) IEEE International Conference on Software Maintenance, ICSM, art. no. 6405277, pp. 233-242. **ISBN:** 978-146732312-3

SOURCE CODE NORMALIZATION

- **LINSEN:** novel technique for code normalization
 - Able to **both** *Split Identifiers* and *Expand* possible occurring abbreviations
 - Based on an efficient String Matching technique:
Baeza-Yates&Perlberg Algorithm (BYP)
 - Exploit different **Sources of Information** to find the matching words

Corazza, A., Di Martino, S., Maggio, V.

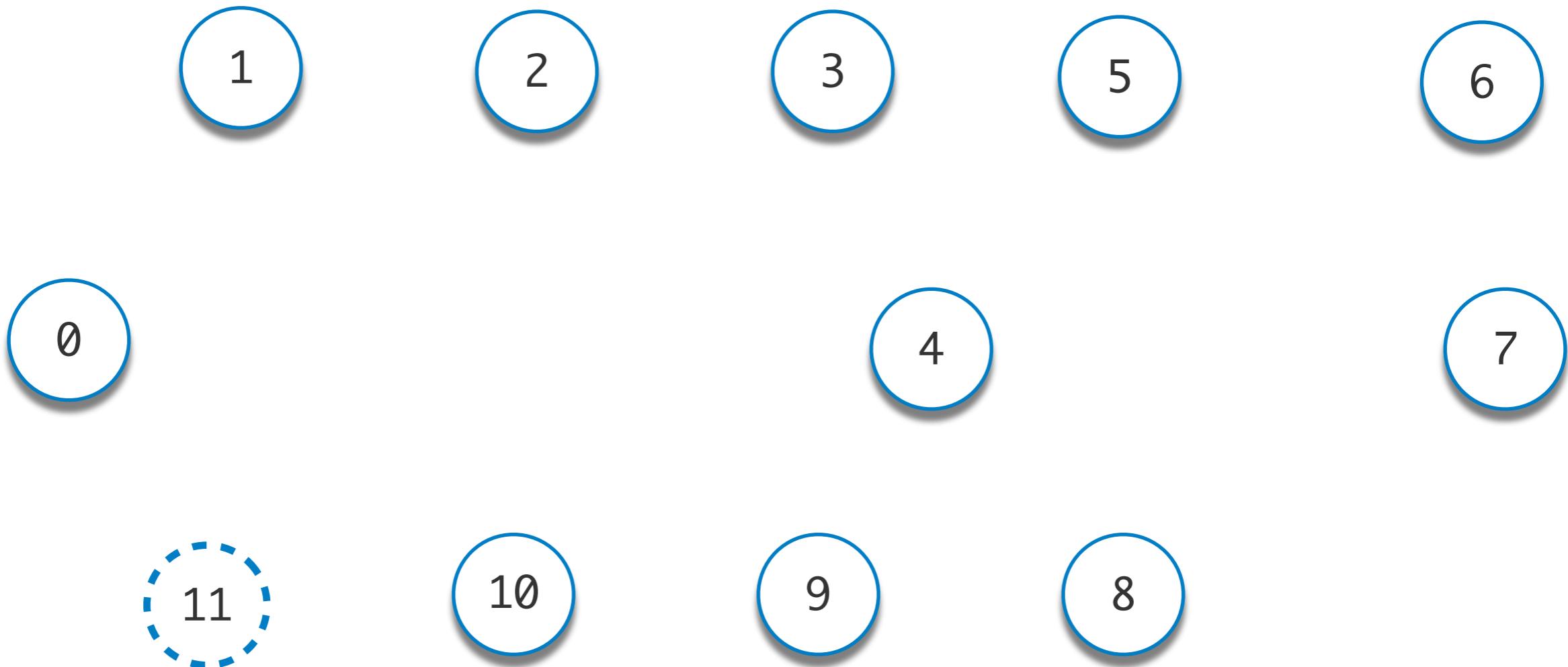
LINSEN: An efficient approach to split identifiers and expand abbreviations

(2012) IEEE International Conference on Software Maintenance, ICSM, art. no. 6405277, pp. 233-242. **ISBN:** 978-146732312-3

SKETCH OF THE ALGORITHM

Model: Weighted Directed Graph

Example: drawXORRect identifier

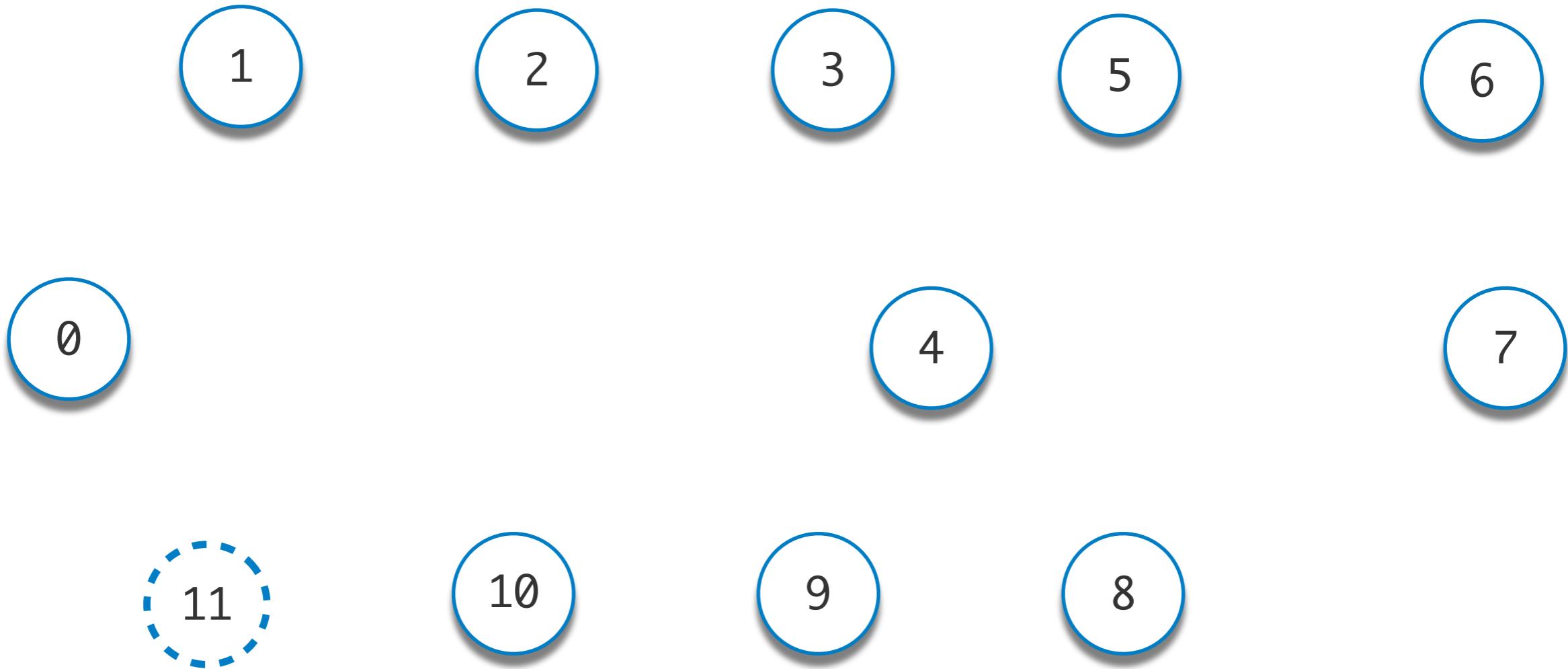


- **NODES** correspond to characters of the current identifier

SKETCH OF THE ALGORITHM

Model: Weighted Directed Graph

Example: drawXORRect identifier

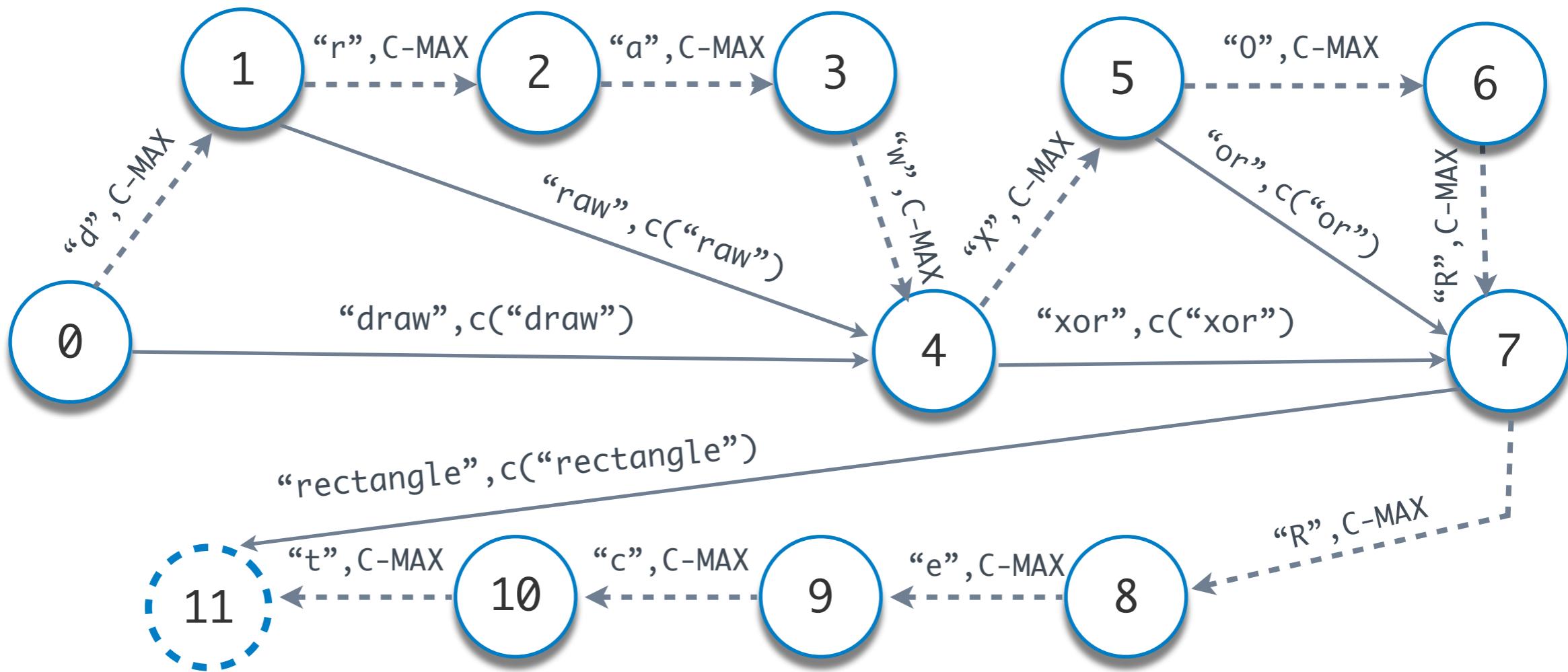


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*

SKETCH OF THE ALGORITHM

Model: Weighted Directed Graph

Example: drawXORRect identifier

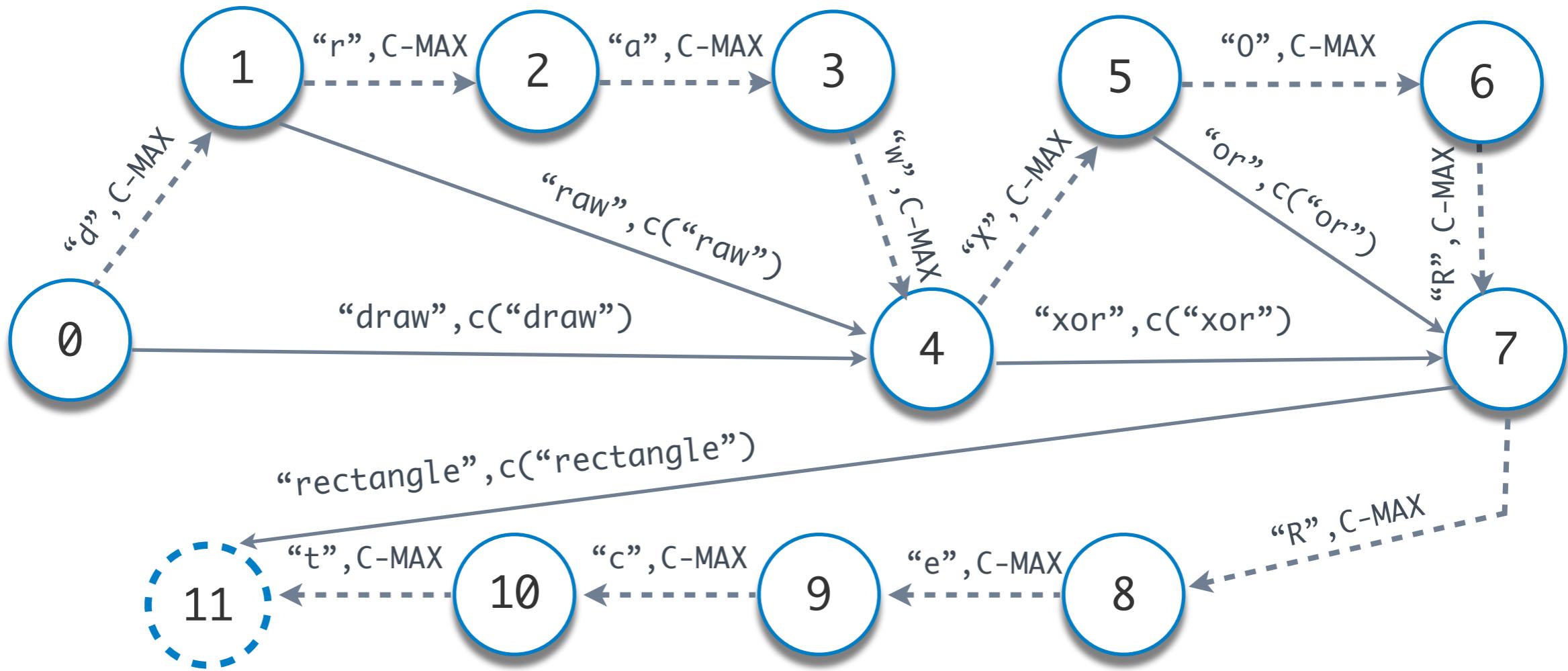


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*
- Padding Arcs to ensure the Graph always connected

SKETCH OF THE ALGORITHM

Model: Weighted Directed Graph

Example: drawXORRect identifier

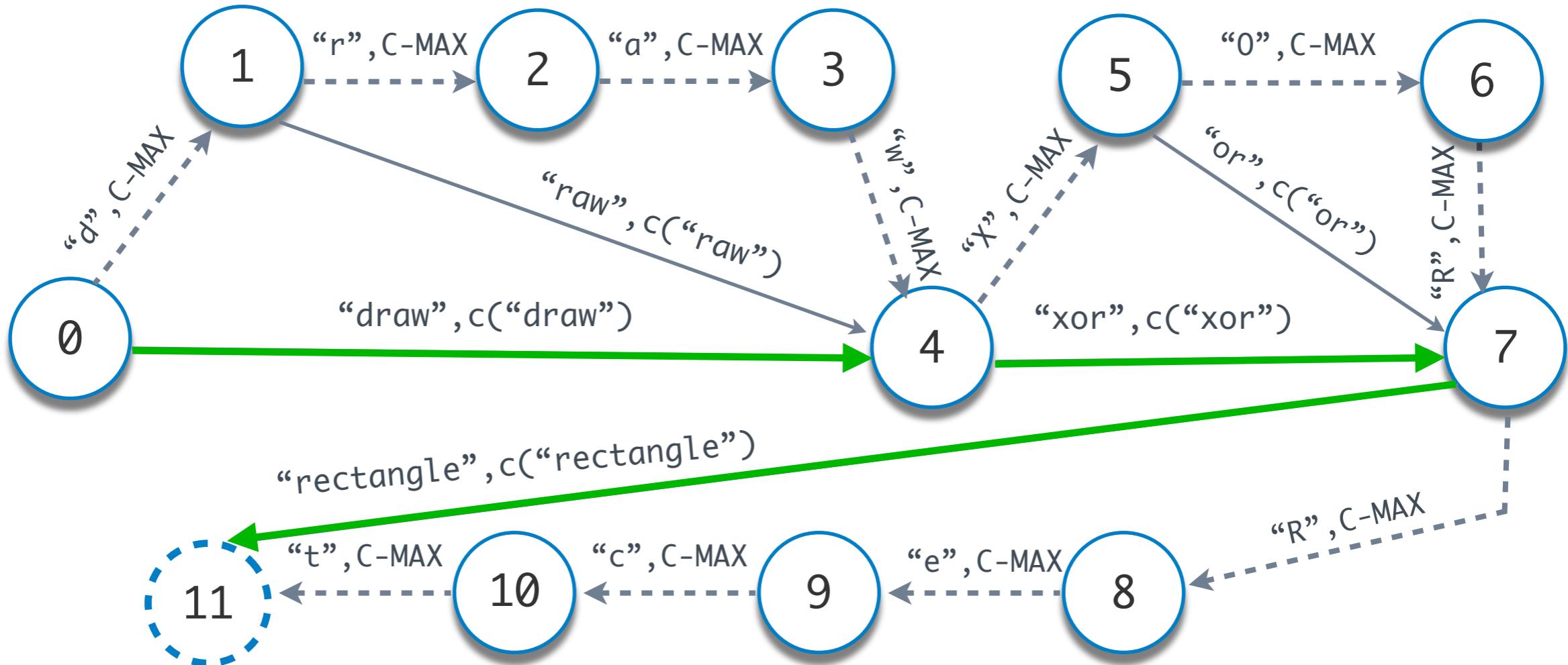


- Every Arc is **Labelled** with the corresponding dictionary word
- **Weights** represent the “cost” of each matching
- *Cost function [c(“word”)]* favors longest words and domain-related information

SKETCH OF THE ALGORITHM

Model: Weighted Directed Graph

Example: drawXORRect identifier



- The final **Mapping Solution** corresponds to the **sequence of labels** in the path with the **minimum cost** (*Dijkstra Algorithm*)

RESULTS

1

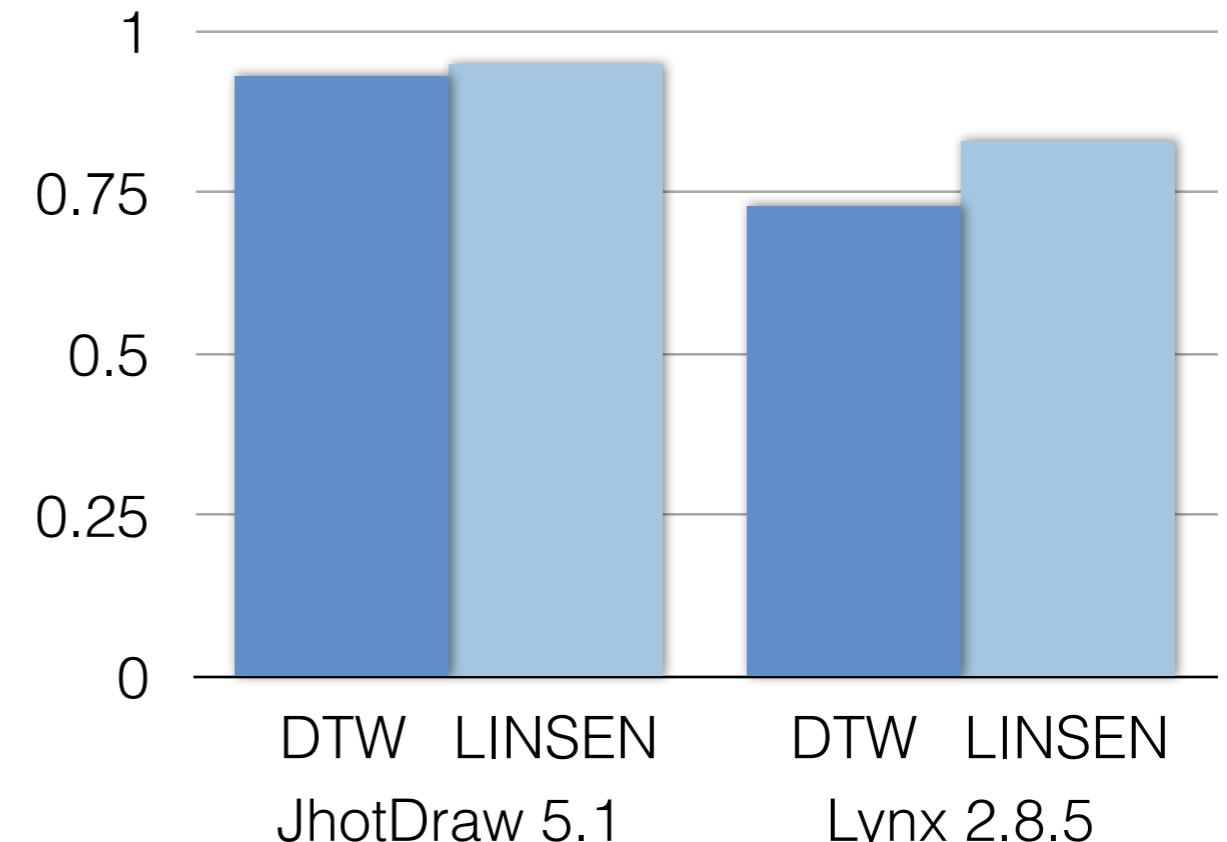
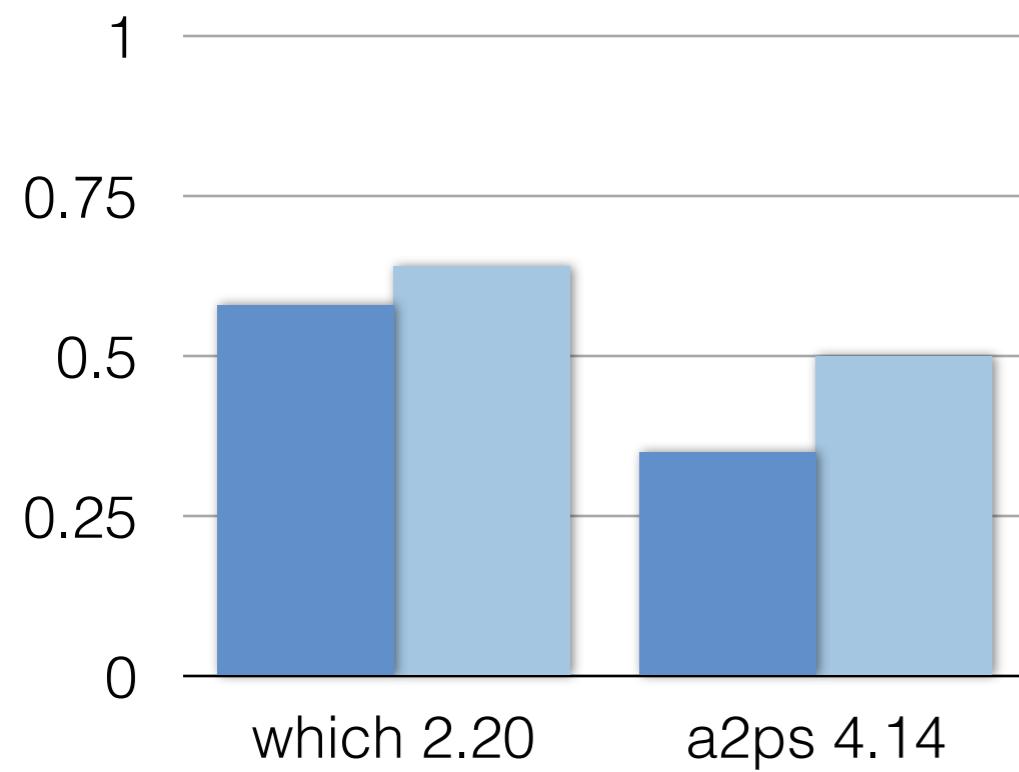
SPLITTING

DTW $O(n^3)$

DTW LINSEN

Accuracy Rates for the comparison with **GenTest** (Lawrie and Binkley, 2011)

GenTest LINSEN



Accuracy Rates for the comparison with **DTW** (Madani et. al 2010)

RE SULTS

2

EXPANSION

CW: Combination Words

DL: Dropped Letters

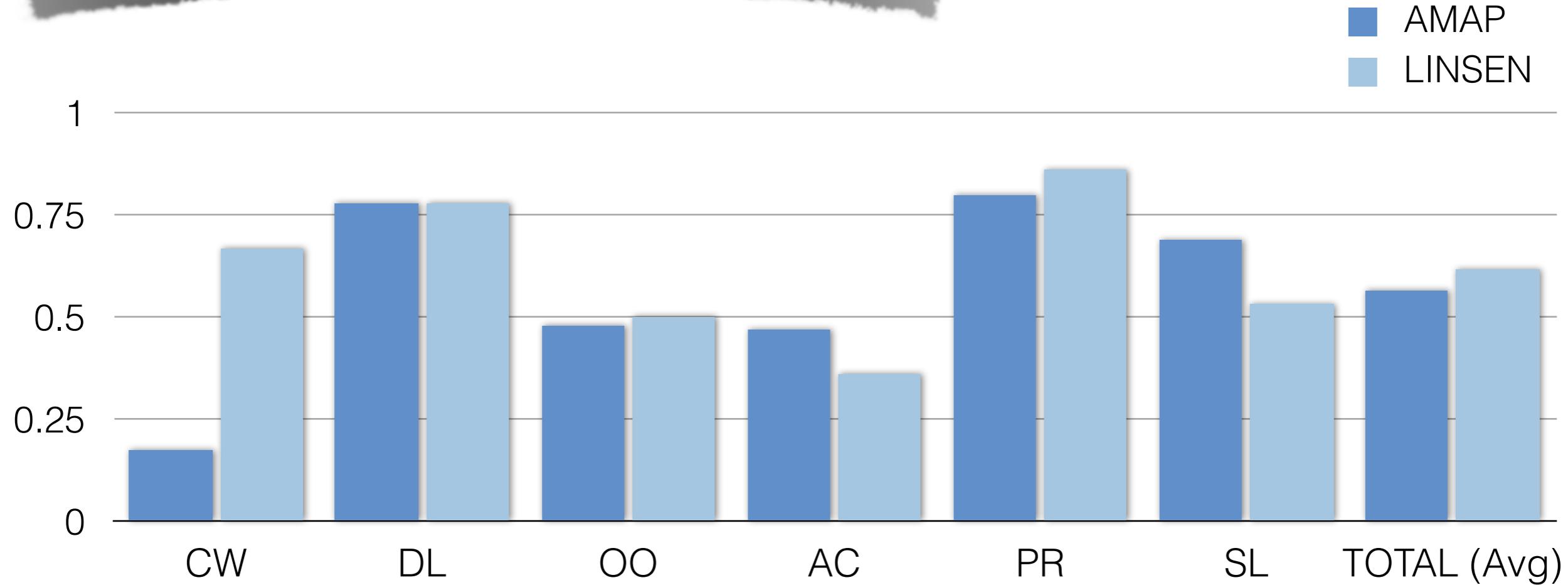
OO: Others

AC: Acronyms

PR: Prefix

SL: Single Letters

Accuracy Rates for the comparison with
AMAP (Hill and Pollock, 2008)



ISSUE #3

CLONE
DETECTION

PROBLEM STATEMENT

CLONE DETECTION

```
protected Rectangle getVisibleEditorRect() {
    Rectangle alloc = editor.getBounds();
    if ((alloc.width > 0) && (alloc.height > 0)) {
        alloc.x = alloc.y = 0;
        Insets insets = editor.getInsets();
        alloc.x += insets.left;
        alloc.y += insets.top;
        alloc.width -= insets.left + insets.right;
        alloc.height -= insets.top + insets.bottom;
        return alloc;
    }
    return null;
}
```

```
static bool _checkDataSelect(DataSelect *a) {
    if (a->arg == 0)
        return false;
    else if (a->arg == -1)
        return true;
    else { // data select argument greater than 1
        for (int i = 0; i < a->arg; i++) {
            if (a->select[i] == 0)
                return false;
        }
        return true;
    }
}
```

```
void setUserObject(Object o) {
    super.setUserObject(o);
    if(path != null) {
        TreeStateNode parent = (TreeStateNode)getParent();
        if(parent != null)
            resetChildrenPaths(parent.getTreePath());
        else
            resetChildrenPaths(null); }}
```

```
opcode_ty *
opcode_push_new() {
    opcode_ty *op =
        opcode_new(&method);
    trace(("opcode_push_new()\n"));
    return op;
}
```

```
static const char *dumpio_enable_output(cmd_parms *cmd, void *dummy, int arg)
{
    dumpio_conf_t *ptr = (dumpio_conf_t *)
        ap_get_module_config(cmd->server->module_config, &dumpio_module);

    ptr->enable_output = arg;
    return NULL;
}
```

```
alloc.width -= insets.left + insets.right;
alloc.height -= insets.top + insets.bottom;
return alloc;
}
```

```
opcode_ty *
opcode_catenate_new() {
    opcode_ty *op;
    trace(("opcode_catenate_new()\n"));
    op = opcode_new(&method);
    trace(("return %08lX;\n", (long)op));
    trace((/*{*/"\n")));
    return op;
}
```

```
static const char *dumpio_enable_input(cmd
{
    dumpio_conf_t *ptr = (dumpio_conf_t *)
        ap_get_module_config(cmd->server->module_config, &dumpio_module);

    ptr->enable_input = arg;
    return NULL;
}
```

```
static bool
_equalRelabelType(RelabelType *a, RelabelType *b) {
    if (!equal(a->arg, b->arg))
        return false;
    if (a->resulttype != b->resulttype)
        return false;
    if (a->resulttypmod != b->resulttypmod)
        return false;
    return true;
}
```

```
void remove(Component comp) {
    if (comp == rootPane) {
        super.remove(comp);
    } else {
        // Client mistake, but we need to handle it to avoid a
        // common object leak in client applications.
        getContentPane().remove(comp); }}
```

```
c_tanh(Py_complex x) {
    Py_complex r;
    double si,ci,shr,chr;
    double rs,ic;
    double d;
    si = sin(x);
    ci = cos(x);
    shr = sinh(x);
    chr = cosh(x);
    ci * si;
    si * ci;
    ci * ci;
    si * si;
    c*rc +
    l = (r+g)/2;
    g = (i-
```

```
dSelect(FieldSelect *a, FieldSelect *b) {
    if (!equal(a->arg, b->arg))
        return false;
    if (a->fieldnum != b->fieldnum)
        return false;
}
```

```
sr,cr,smt,cmr;
rs,is,rc,ic;
d;
in(x.real);
os(x.real);
sinh(x.imag);
cosh(x.imag);
r * chi;
r * shi;
r * chi;
sr * shi;
*rc + ic * ic;
= (rs*rc + is*ic) / d;
= (is*rc - rs*ic) / d;
r;
```

```
void setUserObject(Object o) {
    super.setUserObject(o);
}

void remove(Component comp) {
    if (comp == rootPane) {
        super.remove(comp);
        return %08lX;\n";
        /*{*/"\n");
    } else { // Handle this to avoid object leak!
        getContentPane().remove(comp);
    }
}
```

```
int attlist6(PROLOG_STATE
const char *er
switch (tok) {
    case XML_TOK_PROLOG:
        return XML_ROLE_ATTRIBUTE_NAME;
    case XML_TOK_NAME:
        state->handler = attlist7;
        return XML_ROLE_ATTRIBUTE_NOTATION_VALUE;
}
```

```
void fireViewDestroyingEvent(DrawingView view) {
    final Object[] listeners = listenerList.getListenerList();
    ViewChangeListener vsl = null;
    for (int i = listeners.length-2; i>0 ; i--) {
        if (listeners[i] == ViewChangeListener.class) {
            vsl = (ViewChangeListener)listeners[i+1];
            vsl.viewDestroying( view );
        }
    }
}
```

```
sr,cr,smt,cmr;
rs,is,rc,ic;
d;
in(x.real);
os(x.real);
sinh(x.imag);
cosh(x.imag);
r * chi;
r * shi;
r * chi;
sr * shi;
*rc + ic * ic;
= (rs*rc + is*ic) / d;
= (is*rc - rs*ic) / d;
r;
```

```
Fail_new() {
    opcode_ty *op;
    trace(("opcode_fail_new()\n"));
    op = opcode_new(&method);
    trace(("return %08lX;\n", (long)op));
    trace((/*{*/"\n"));
    return op;
}
```

```
case XML_TOK_PREFIXED_NAME:
    state->handler = doctype1;
    return XML_ROLE_DOCTYPE_NAME;
}
```

```
} return common(state, tok);

static bool _checkDataElements(DataSelect *elem) {
    switch (elem->arg) {
        case 0:
            return false;
        case -1:
            return true;
        default:
            int i = 0;
            while ( i < elem->arg ) {
                if (a->select[i] == 0)
                    return false;
                i = i + 1;
            }
            return true;
    }
}
```

```
void fireViewSelectionChangedEvent(DrawingView oldView, DrawingView newView)
{
    final Object[] listeners = listenerList.getListenerList();
    ViewChangeListener vsl = null;
    for (int i = listeners.length-2; i>0 ; i--) {
        if (listeners[i] == ViewChangeListener.class) {
            vsl = (ViewChangeListener)listeners[i+1];
            vsl.viewSelectionChanged(oldView, newView);
        }
    }
}
```

PROBLEM STATEMENT

CLONE DETECTION

```
protected Rectangle getVisibleEditorRect() {
    Rectangle alloc = editor.getBounds();
    if ((alloc.width > 0) && (alloc.height > 0)) {
        alloc.x = alloc.y = 0;
        Insets insets = editor.getInsets();
        alloc.x += insets.left;
        alloc.y += insets.top;
        Rectangle rootEditorRect = editor.getComponent().getBounds();
        alloc.width -= insets.left - insets.top;
        alloc.height -= insets.top - insets.bottom;
        return alloc;
    }
    return null;
}
```

```
static const char *dumpio_enable_output(cmd_parms *cmd, void *dummy, int arg)
{
    dumpio_conf_t *ptr = (dumpio_conf_t *)
        ap_get_module_config(cmd->server->module_config, &dumpio_module);
    ptr->enable_output = arg;
    return NULL;
}
```

```
opcode_ty *
opcode_catenate_new() {
    opcode_ty      *op;
    trace(("opcode_catenate_new()\n{\n/*}*/"));
    op = opcode_new(&method);
    trace(("return %08lX;\n", (long)op));
    trace((/*{/*/}\n"));
    return op; }
```

```
super.setUserObject(o);
if(path != null) {
    TreeStateNode parent = (TreeStateNode)getParent();
    if(parent != null)
        resetChildrenPaths(parent.getTreePath());
    else
        resetChildrenPaths(null); }
```

```
opcode_ty *
opcode_push_new() {
    void remove(Component comp) {
        opcode_ty      *op; if (comp == rootPane) {
            trace(("opcode_push_new(super->rootPane)\n"));
            op = opcode_new(&method);
            trace(("return %08lX;\n", (long)op));
            trace((/*{/*/}\n"));
            getContentPane().remove(comp); } }
        return op; }
```

```
c_tanh(Py_complex x) {
    Py_complex r;
    double si,ci,shr,chr;
    double rs,is,rc,ic;
    double d;
    si = sin(x.imin);
    if (rs*rc + is*ic < 0.0) {
        shr = -sqrt(-rs*rc - is*ic);
        chr = -sqrt(rs*rc - is*ic);
    } else {
        shr = sqrt(rs*rc + is*ic);
        chr = sqrt(rs*rc - is*ic);
    }
    r = PyComplex_FromDoubles(shr,chr);
    Py_XINCREF(r);
    return r;
}
```

```
void fireViewSelectionChangedEvent(DrawingView oldView, DrawingView newView)
{
    final Object[] listeners = listenerList.getListenerList();
    ViewChangeListener vsl = null;
    for (int i = listeners.length-2; i>=0 ; i--) {
        if (listeners[i] == ViewChangeListener.class) {
            vsl = (ViewChangeListener)listeners[i+1];
            vsl.viewSelectionChanged(oldView, newView);
        }
    }
}

int attlist6(PROLOG_STATE *state, int tok, const char *ptr,
            const char *end, const ENCODING *enc) {
    switch (tok) {
        case XML_TOK_PROLOG_S:
            return XML_ROLE_NONE;
        case XML_TOK_NAME:
            state->handler = attlist7;
            return XML_ROLE_ATTRIBUTE_NOTATION_VALUE;
        default:
            return common(state, tok);
    }
}

void fireViewDestroyingEvent(DrawingView view) {
    final Object[] listeners = listenerList.getListenerList();
    ViewChangeListener vsl = null;
    for (int i = listeners.length-2; i>=0 ; i--) {
        if (listeners[i] == ViewChangeListener.class) {
            vsl = (ViewChangeListener)listeners[i+1];
            vsl.viewDestroyingEvent(view);
        }
    }
}
```

```
opcode_ty *
opcode_fail_new() {
    opcode_ty      *op;
    trace(("opcode_fail_new()\n{\n/*}*/"));
    op = opcode_new(&method);
    trace(("return %08lX;\n", (long)op));
    trace((/*{/*/}\n"));
    return op; }
```

```
static bool _checkDataElements(DataSelect *elem) {
    switch (elem->arg) {
        case 0:
            return false;
        case -1:
            return true;
        default:
            int i = 0;
            while (i < elem->arg) {
                if (a->select[i] == 0)
                    return false;
                i = i + 1;
            }
            return true;
    }
}
```

Clones Textual Similarity

PROBLEM STATEMENT

CLONE DETECTION

```
protected Rectangle getVisibleEditorRect() {
    Rectangle alloc = editor.getBounds();
    if (alloc.width > 0) && (alloc.height > 0) {
        alloc.x = alloc.y = 0;
        Insets insets = editor.getInsets();
        alloc.x += insets.left;
```

```
static bool _checkDataSelect(DataSelect *a) {
    if (a->arg == 0)
        return false;
    else if (a->arg == -1)
        return true;
    else { // data select argument greater than zero
        for (int i = 0; i < a->arg; i++) {
            if (a->select[i] == 0)
                return false;
        }
        return true;
    }
}
```

```
else
    resetChildrenPaths(null); }
```

```
opcode_ty *
opcode_push_new() {
    void remove(Component comp) {
        opcode_ty *op; if (comp == rootPane) {
            trace(("opcode_push_new(%p)\n", comp));
            op = opcode_new(&methodObj);
            trace(("return %08lX;\n", (long)op));
            trace((/*{"/}\n"));
            getContentPane().remove(comp); }
        return op; }
```

```
static const char *dumpio_enable_output(cmd_parms *cmd, void *dummy, int arg)
{
    dumpio_conf_t *ptr = (dumpio_conf_t *)
        ap_get_module_config(cmd->server->module_config, &dumpio_module);
    ptr->enable_output = arg;
    return NULL;
```

```
void fireViewSelectionChangedEvent(DrawingView oldView, DrawingView newView)
{
    final Object[] listeners = listenerList.getListenerList();
    ViewChangeListener vsl = null;
    for (int i = listeners.length-2; i>=0 ; i--) {
        if (listeners[i] == ViewChangeListener.class) {
            const char *ptr;
            const ENCODING_TYPE vsl;
            ViewChangeListener(vsl) listeners[i+1];
```

```
static bool _checkDataElements(DataSelect *elem) {
    switch (elem->arg){
        case 0:
            return false;
        case -1:
            return true;
        default:
            int i = 0;
            while ( i < elem->arg ){
                if (a->select[i] == 0)
                    return false;
                i = i + 1;
            }
            return true;
    }
}
```

Clones Functional Similarity

```
return true;
case -1:
    return true;
default:
    int i = 0;
    while ( i < elem->arg ) {
        if (a->select[i] == 0)
            return false;
        i = i + 1;
    }
    return true;
```

PROBLEM STATEMENT

CLONE DETECTION

```
protected Rectangle getVisibleEditorRect() {  
    Rectangle alloc = editor.getBounds();  
    if ((alloc.width > 0) && (alloc.height > 0)) {  
        alloc.x = alloc.y = 0;  
        Insets insets = editor.getInsets();  
        alloc.x += insets.left;  
        alloc.y += insets.top;  
        Rectangle getRootEditorRect() {  
            alloc.width -= insets.left + insets.right;  
            alloc.height -= insets.top + insets.bottom;  
            return alloc; }  
        alloc.x = alloc.y = 0;  
        Insets insets = JTextField.this.getInsets();  
        alloc.x += insets.left;  
        alloc.y += insets.top;  
        alloc.width -= insets.left + insets.right;  
        alloc.height -= insets.top + insets.bottom;  
        return alloc; }  
    return null; }
```

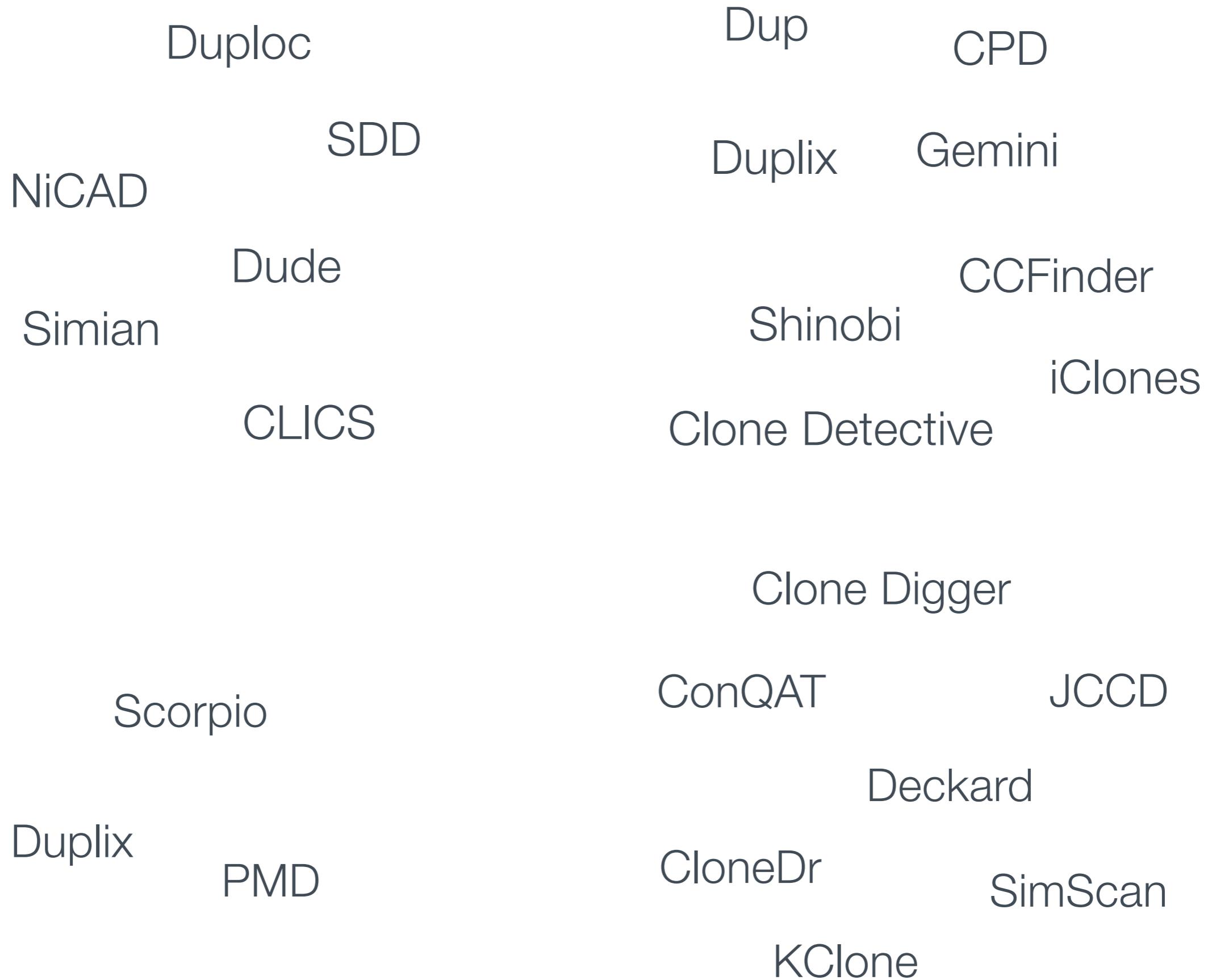
```
void fireViewSelectionChangedEvent(DrawingView oldView, DrawingView newView)  
{  
    final Object[] listeners = listenerList.getListenerList();  
    ViewChangeListener vsl = null;  
    for (int i = listeners.length-2; i=0 ; i-=2) {  
        if (listeners[i] == ViewChangeListener.class) {  
            vsl = (ViewChangeListener)listeners[i+1];  
            vsl.viewSelectionChanged(oldView, newView);  
        }  
    }  
}
```

```
void fireViewDestroyingEvent(DrawingView view) {  
    final Object[] listeners = listenerList.getListenerList();  
    ViewChangeListener vsl = null;  
    for (int i = listeners.length-2; i=0 ; i-=2) {  
        if (listeners[i] == ViewChangeListener.class) {  
            vsl = (ViewChangeListener)listeners[i+1];  
            vsl.viewDestroying( view );  
        }  
    }  
}
```

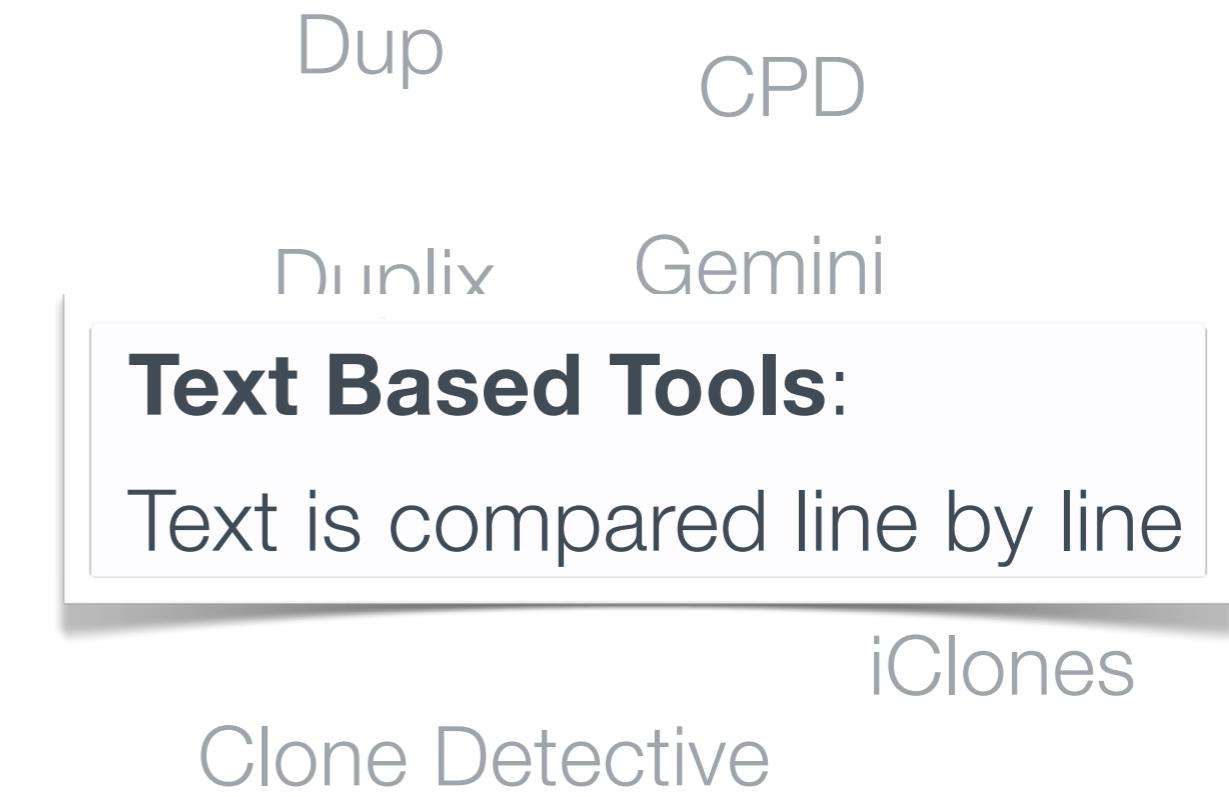
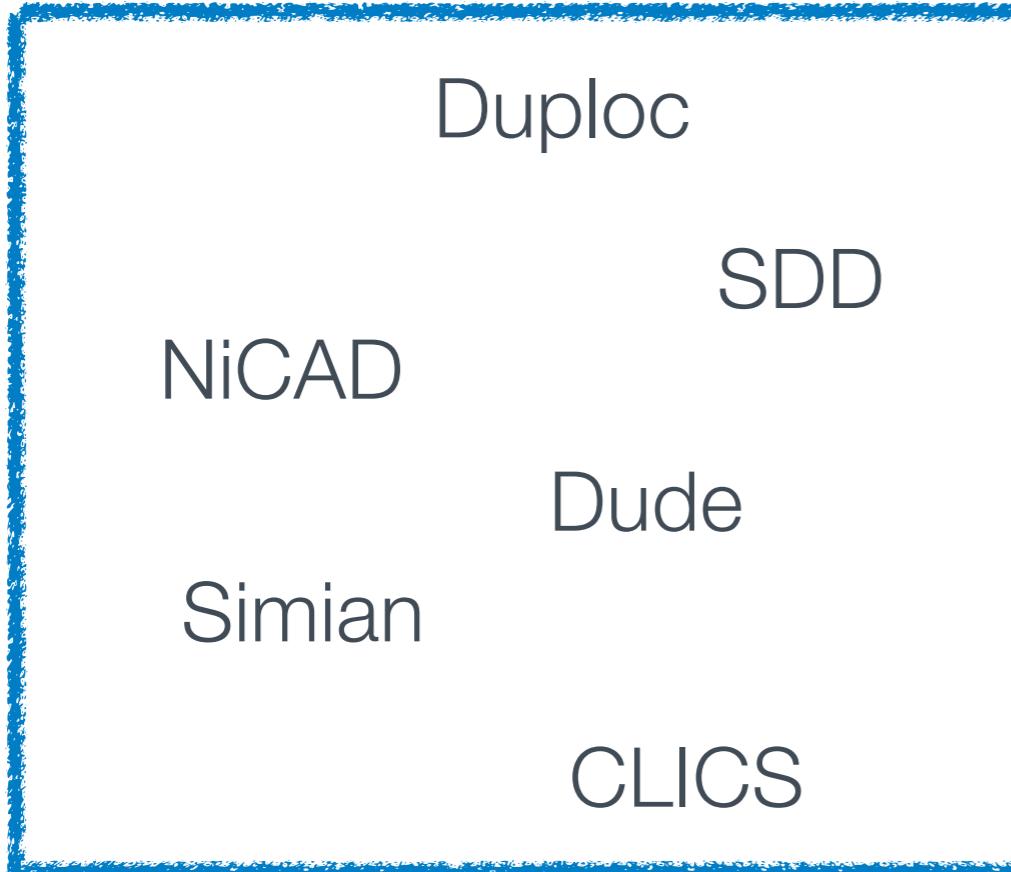
Sneaky Bug!

Clones affect the **reliability** of the system!

STATE OF THE ART TOOLS



STATE OF THE ART TOOLS



STATE OF THE ART TOOLS

Duploc
SDD
NiCAD
Dude
Simian
CLICS
Scorpio
Duplix
PMD

Dup
CPD
Duplix Gemini
CCFinder
Shinobi
iClones
Clone Detective

Clone Digger

GenCAT
Token Based Tools:
Token sequences are
compared to sequences

KClone

STATE OF THE ART

TOOLS

Duploc
NiCAD
Dude
Simian
CLICS
Scorpio
Duplix
PMD

Dup
CPD

Syntax Based Tools:
Syntax subtrees are compared
to each other

Clone Detective

Clone Digger
ConQAT
JCCD
Deckard
CloneDr
SimScan
KClone

STATE OF THE ART TOOLS

Duplix
PMD

Scorpio

NiCAD

Duploc

SDD

Simian
CLICS

Dude

CLICS

Dup
CPD

Duplix
Gemini

CCFinder

Shinobi

Clone Detective

iClones

Clone Digger

ConCAT

ICCD

Graph Based Tools:

(sub) graphs are compared to each other

KClone

CLONE DETECTION

- Combining different sources of information to improve the effectiveness of the detection process

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

A Tree Kernel based approach for clone detection

(2010) IEEE International Conference on Software Maintenance, ICSM, art. no. 5609715. **ISBN:** 978-142448629-8

Corazza, A., Di Martino, S., Maggio, V., Moschitti, A., Passerini, A., Scanniello, G., Silvestri F.

Using Machine Learning and Information Retrieval Techniques to Improve Software Maintainability

(2013) Communications in Computer and Information Science, *In Press*

CLONE DETECTION

- Combining different sources of information to improve the effectiveness of the detection process
- Structural and Lexical Information

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

A Tree Kernel based approach for clone detection

(2010) IEEE International Conference on Software Maintenance, ICSM, art. no. 5609715. **ISBN:** 978-142448629-8

Corazza, A., Di Martino, S., Maggio, V., Moschitti, A., Passerini, A., Scanniello, G., Silvestri F.

Using Machine Learning and Information Retrieval Techniques to Improve Software Maintainability

(2013) Communications in Computer and Information Science, *In Press*

CLONE DETECTION

- Combining different sources of information to improve the effectiveness of the detection process
- Structural and Lexical Information
- Application of **Kernel Methods** to Source Code **Structures**

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

A Tree Kernel based approach for clone detection

(2010) IEEE International Conference on Software Maintenance, ICSM, art. no. 5609715. **ISBN:** 978-142448629-8

Corazza, A., Di Martino, S., Maggio, V., Moschitti, A., Passerini, A., Scanniello, G., Silvestri F.

Using Machine Learning and Information Retrieval Techniques to Improve Software Maintainability

(2013) Communications in Computer and Information Science, *In Press*

CLONE DETECTION

- Combining different sources of information to improve the effectiveness of the detection process
- Structural and Lexical Information
- Application of **Kernel Methods** to Source Code **Structures**
- Typically applied in other field such as NLP or Bioinformatics

Corazza, A., Di Martino, S., Maggio, V., Scanniello, G.

A Tree Kernel based approach for clone detection

(2010) IEEE International Conference on Software Maintenance, ICSM, art. no. 5609715. **ISBN:** 978-142448629-8

Corazza, A., Di Martino, S., Maggio, V., Moschitti, A., Passerini, A., Scanniello, G., Silvestri F.

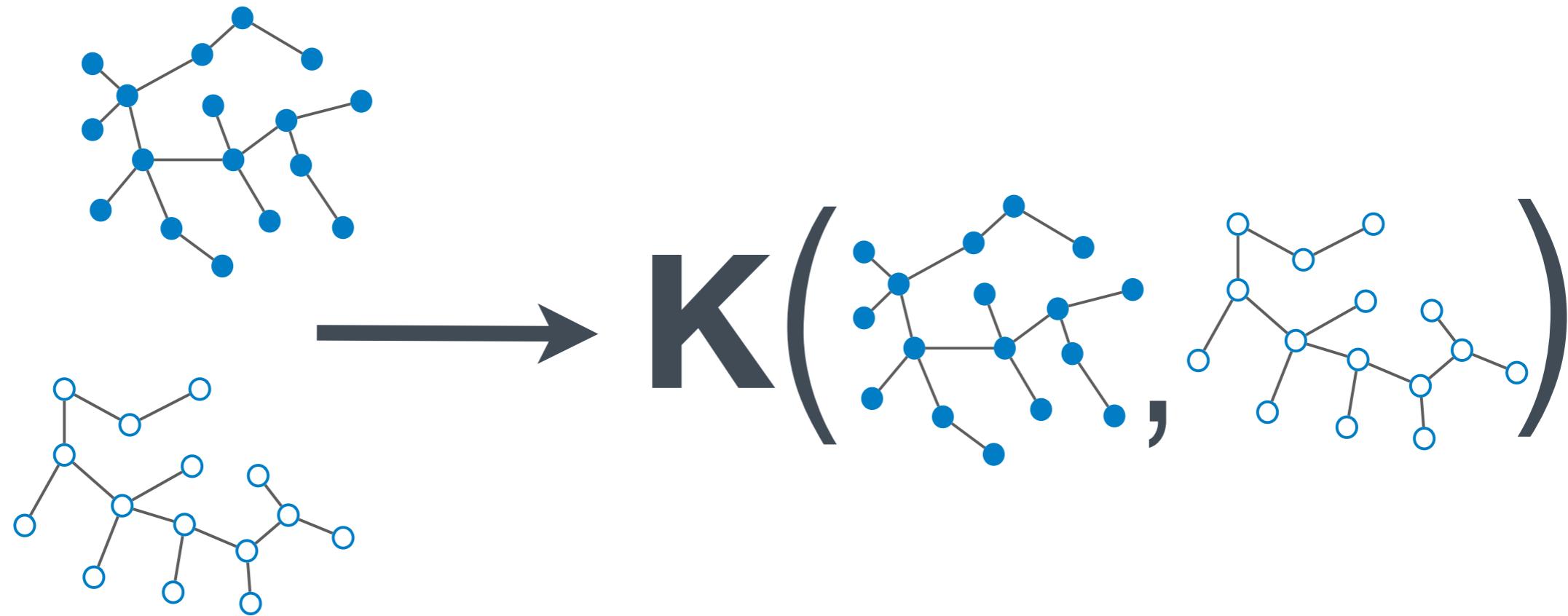
Using Machine Learning and Information Retrieval Techniques to Improve Software Maintainability

(2013) Communications in Computer and Information Science, *In Press*

KERNELS

FOR STRUCTURES

Computation of the **dot product** between (Graph) Structures

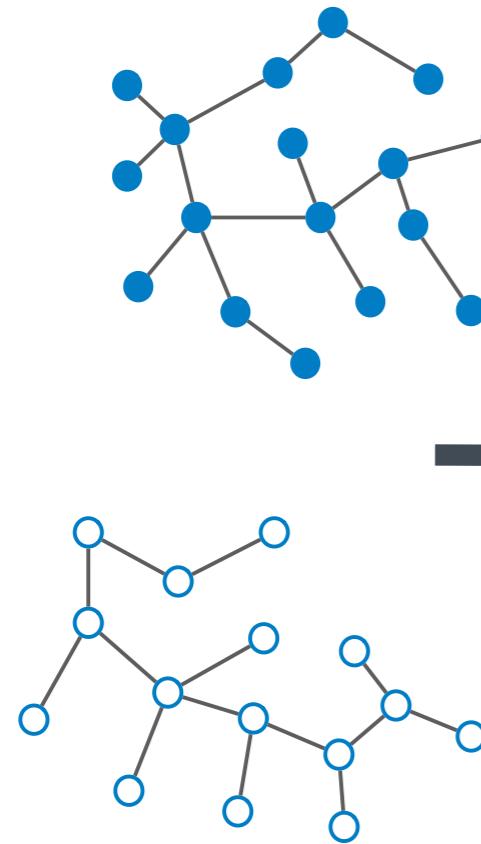


KERNELS

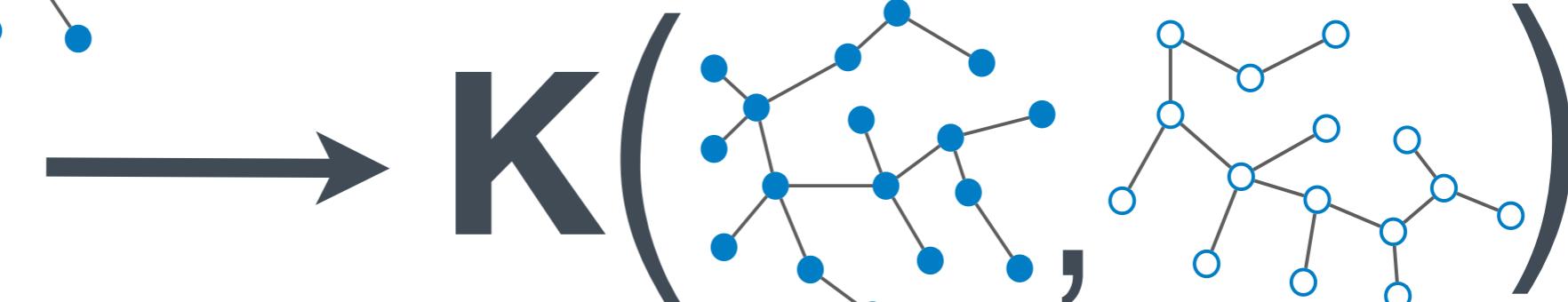
STRUCTURES FOR SSES

Program Dependencies Graph (PDG)

(Directed) Graph structure representing the relationship among the different statement of a program



Computation of the **dot product** between (Graph) Structures



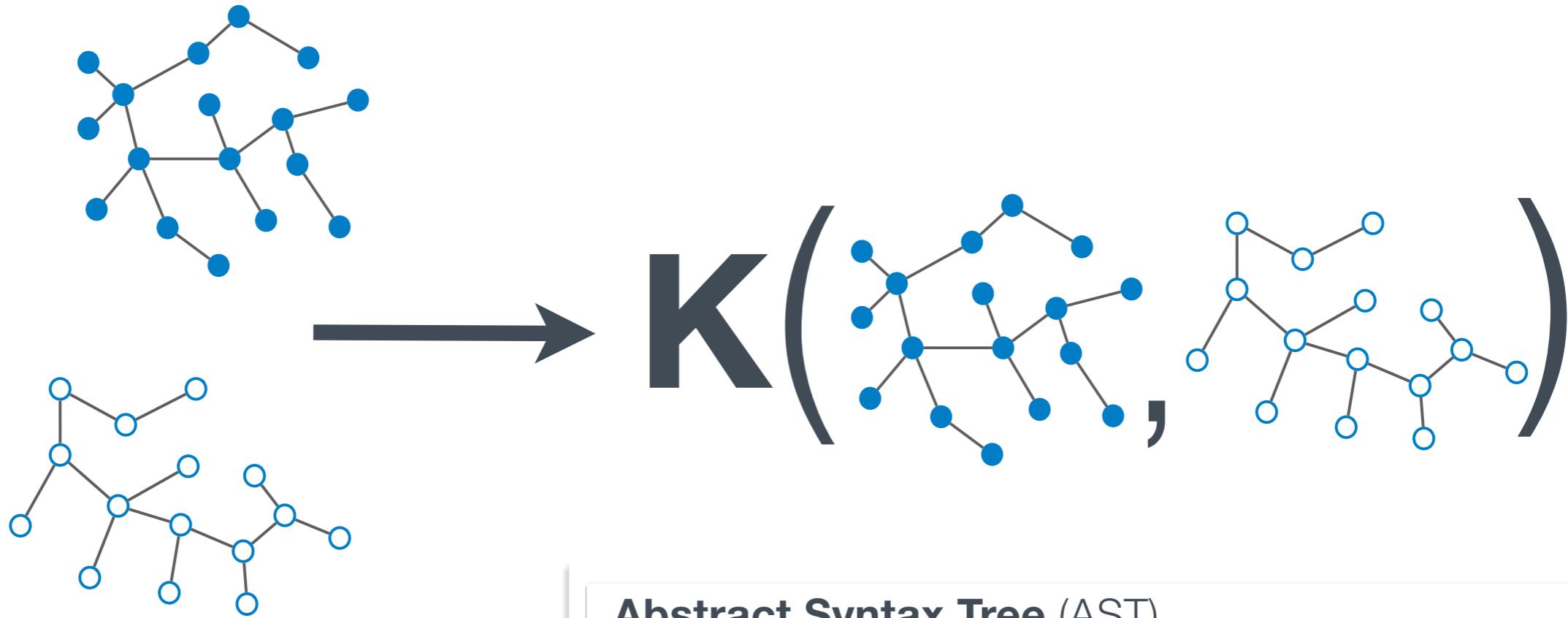
Abstract Syntax Tree (AST)

Tree structure representing the syntactic structure of the different instructions of a program (function)

KERNELS

STRUCTURES FOR KERNELESSES

Computation of the **dot product** between (Graph) Structures



Abstract Syntax Tree (AST)

Tree structure representing the syntactic structure of the different instructions of a program (function)

Program Dependencies Graph (PDG)

(Directed) Graph structure representing the relationship among the different statement of a program

CODE

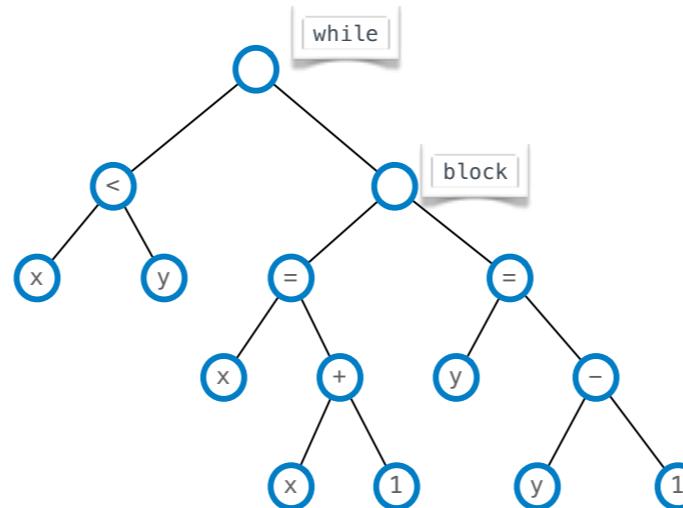
```
while (x < y){  
    x = x + 1;  
    y = y - 1;  
}
```

```
while (b > a){  
    if (b > 0)  
        c = 3;  
    a = a + 1;  
    b = b - 1;  
}
```

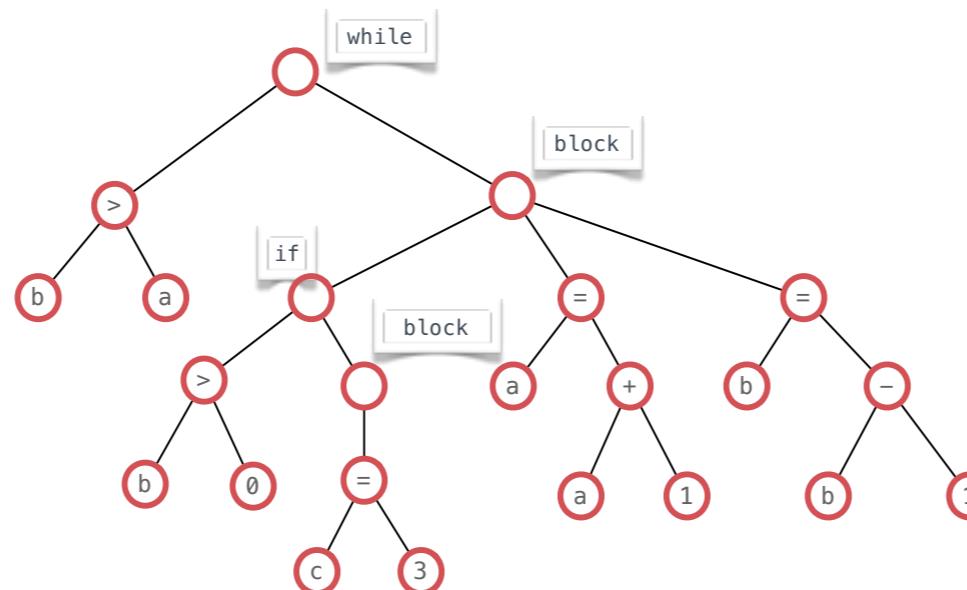
CODE

```
while (x < y){  
    x = x + 1;  
    y = y - 1;  
}
```

AST



```
while (b > a){  
    if (b > 0)  
        c = 3;  
    a = a + 1;  
    b = b - 1;  
}
```

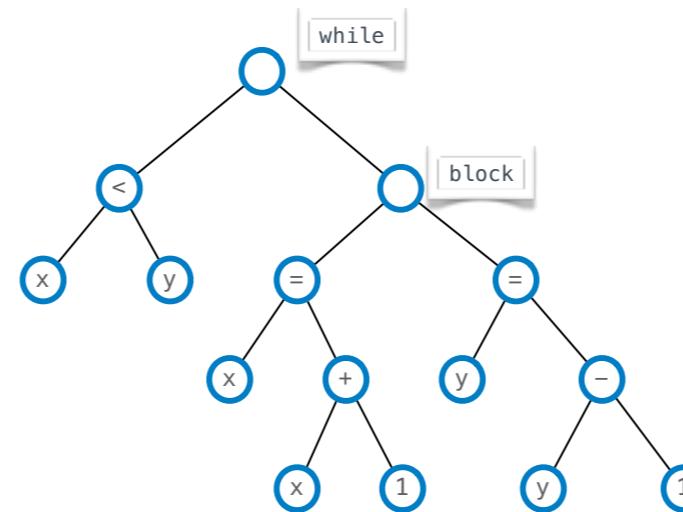


CS
11
Z
O
L
R
E
L
Z
R
U

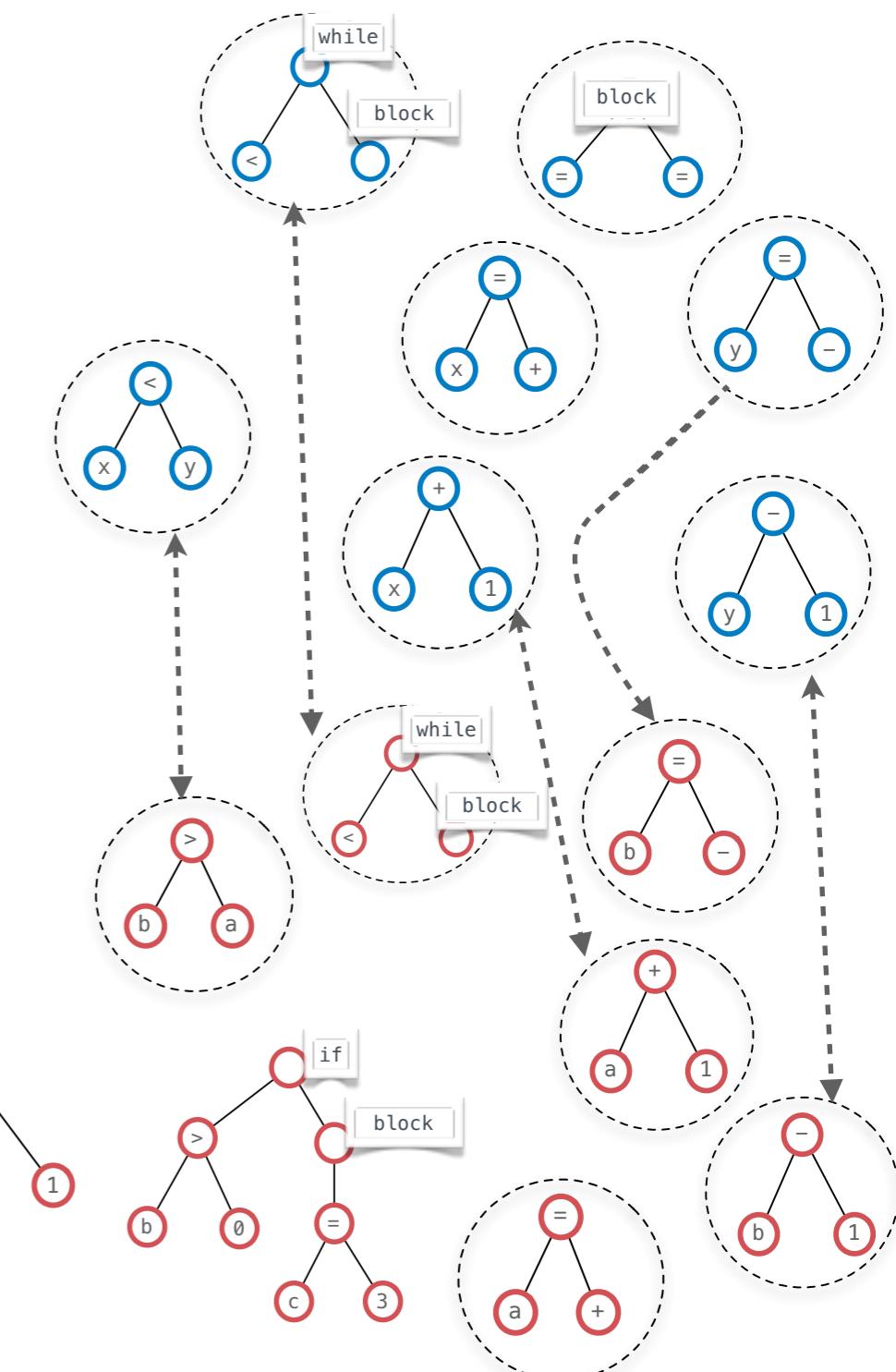
CODE

```
while (x < y){  
    x = x + 1;  
    y = y - 1;  
}
```

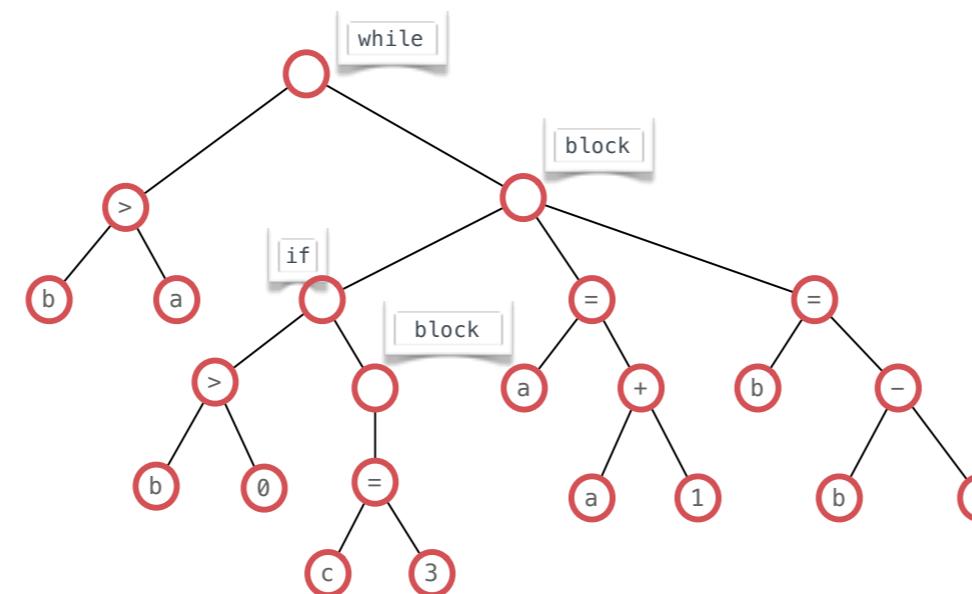
AST



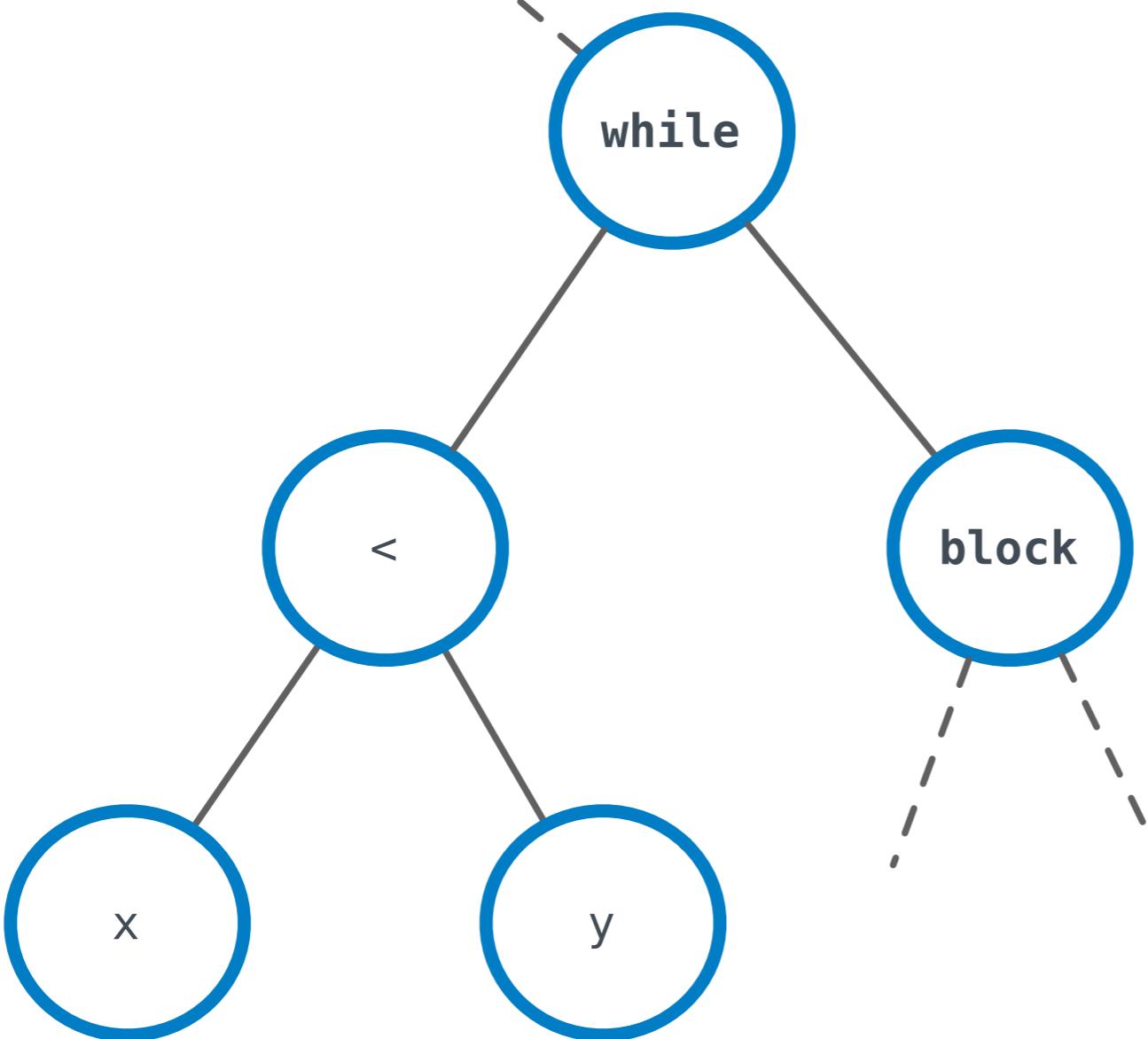
AST KERNEL



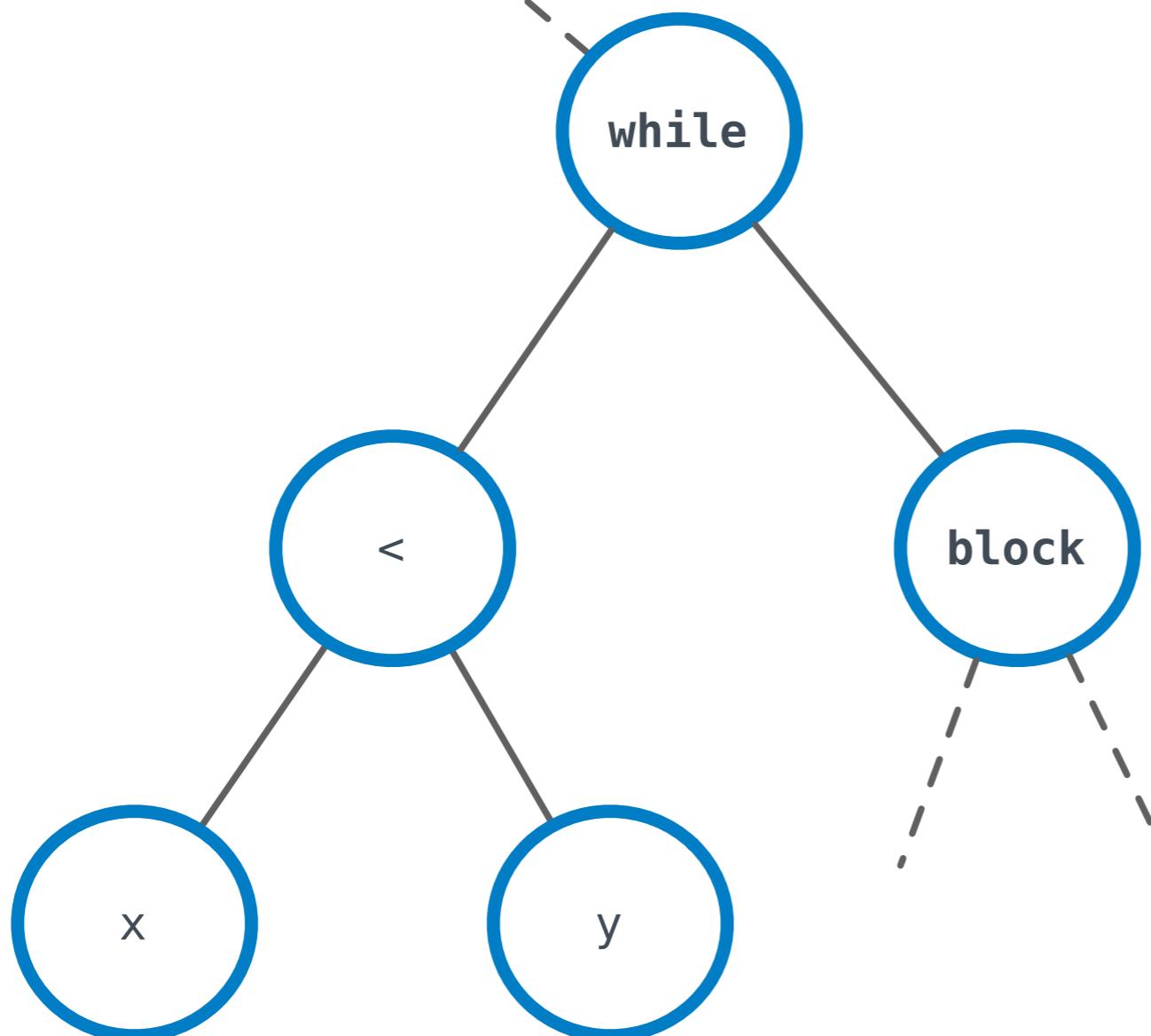
```
while (b > a){  
    if (b > 0)  
        c = 3;  
    a = a + 1;  
    b = b - 1;  
}
```



SYNTHETIC FEATURES

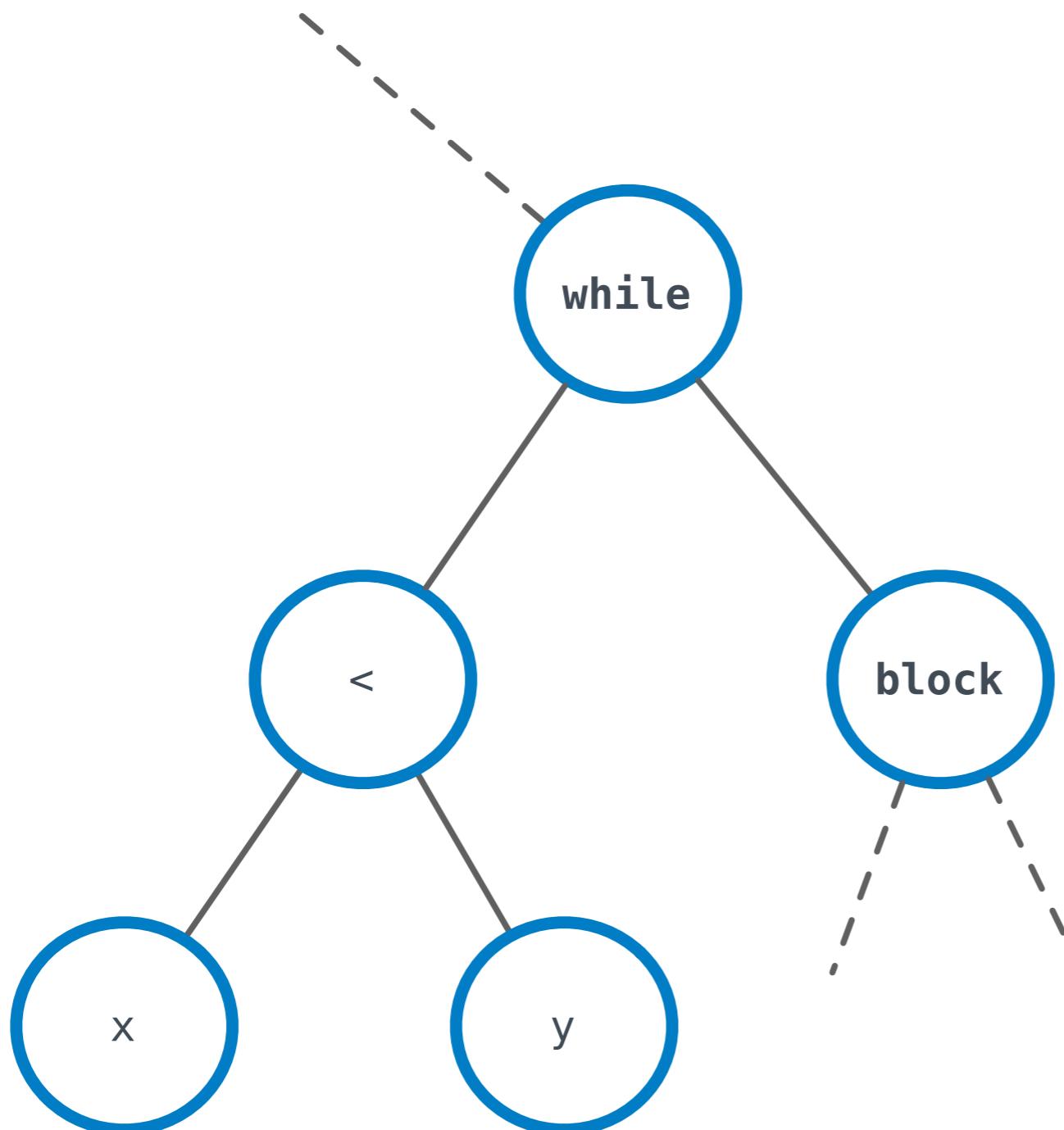


SYNTHETIC FEATURES



Instruction Class (IC)
i.e., LOOP, CALL,
CONDITIONAL_STATEMENT

SYNTHETIC FEATURES

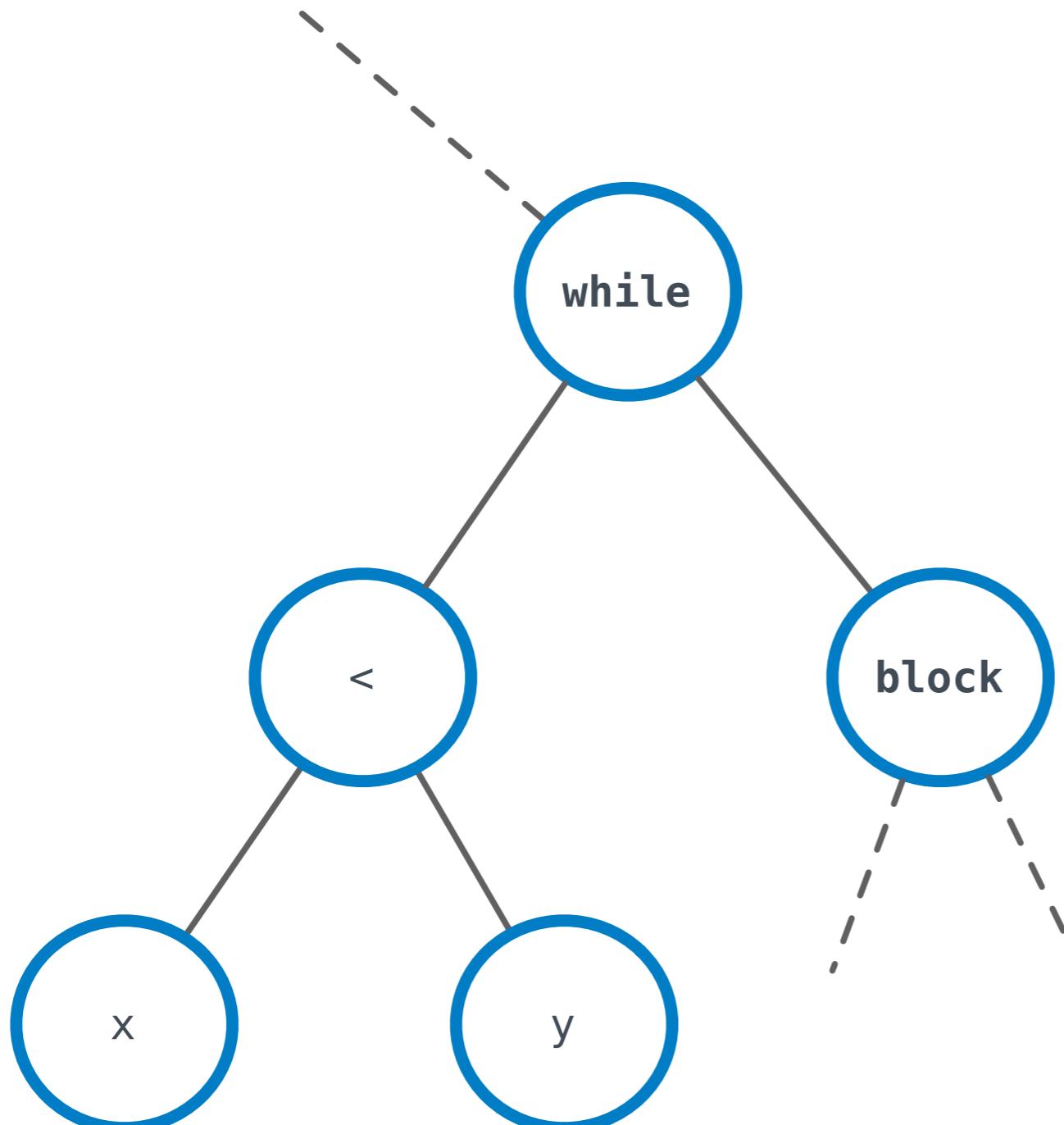


Instruction Class (IC)

i.e., LOOP, CALL,
CONDITIONAL_STATEMENT

Instruction (I)

i.e., FOR, IF, WHILE, RETURN



Instruction Class (IC)

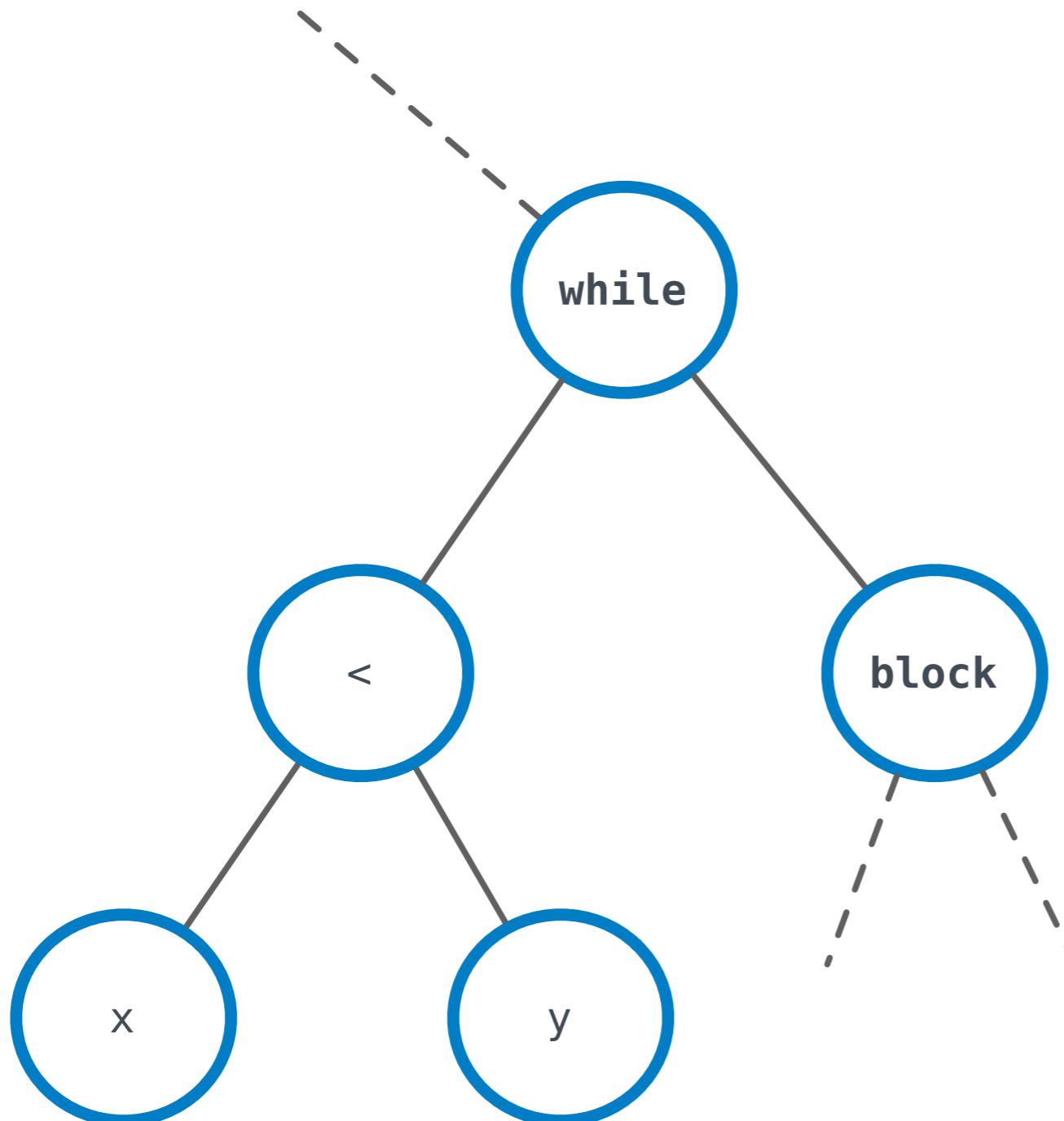
i.e., LOOP, CALL,
CONDITIONAL_STATEMENT

Instruction (I)

i.e., FOR, IF, WHILE, RETURN

Context (C)

i.e., Instruction Class of
the closer statement node



Instruction Class (IC)

i.e., LOOP, CALL,
CONDITIONAL_STATEMENT

Instruction (I)

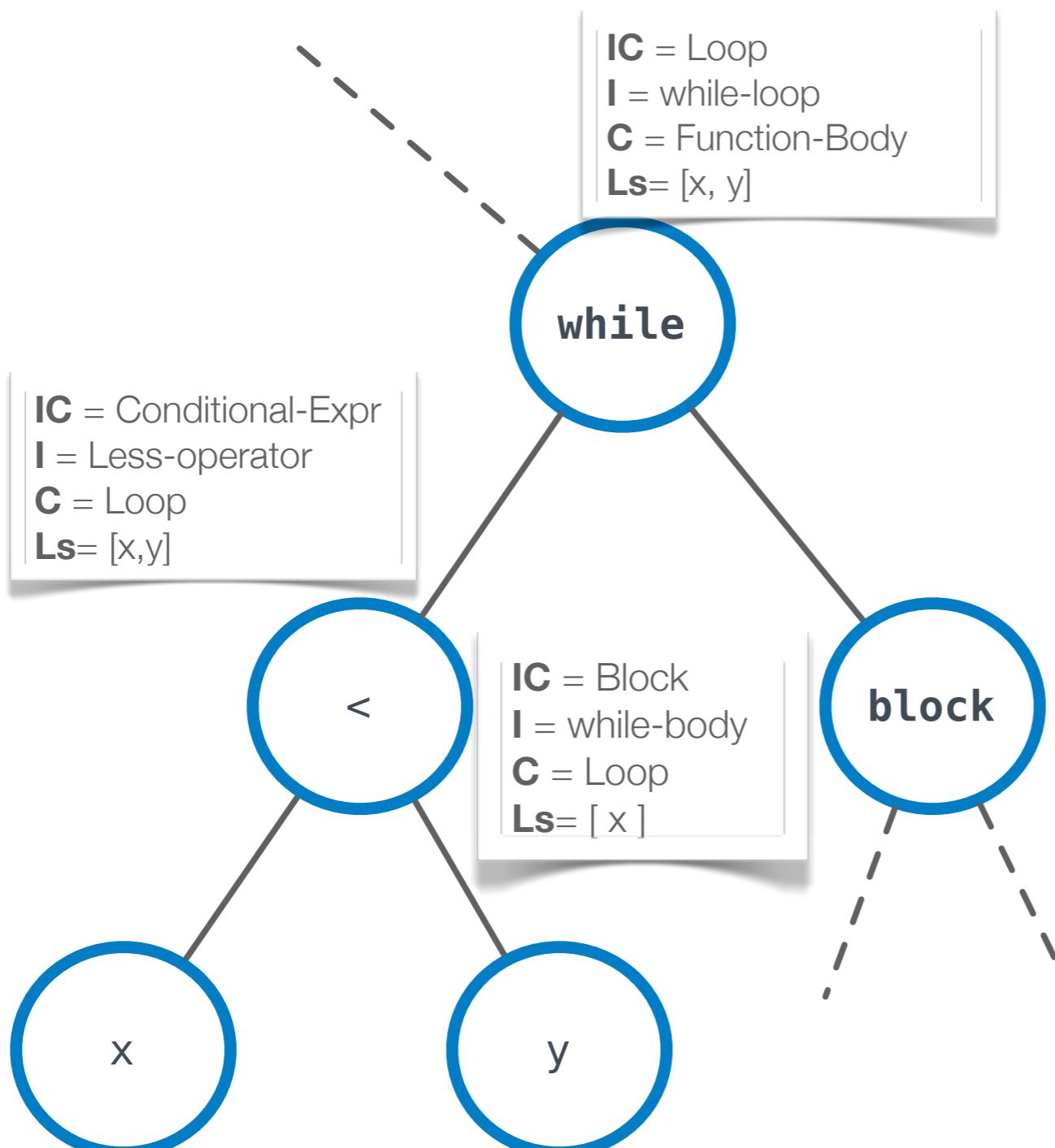
i.e., FOR, IF, WHILE, RETURN

Context (C)

i.e., Instruction Class of
the closer statement node

Lexemes (Ls)

Lexical information gathered
(recursively) from leaves



Instruction Class (IC)

i.e., LOOP, CALL,
CONDITIONAL_STATEMENT

Instruction (I)

i.e., FOR, IF, WHILE, RETURN

Context (C)

i.e., Instruction Class of
the closer statement node

Lexemes (Ls)

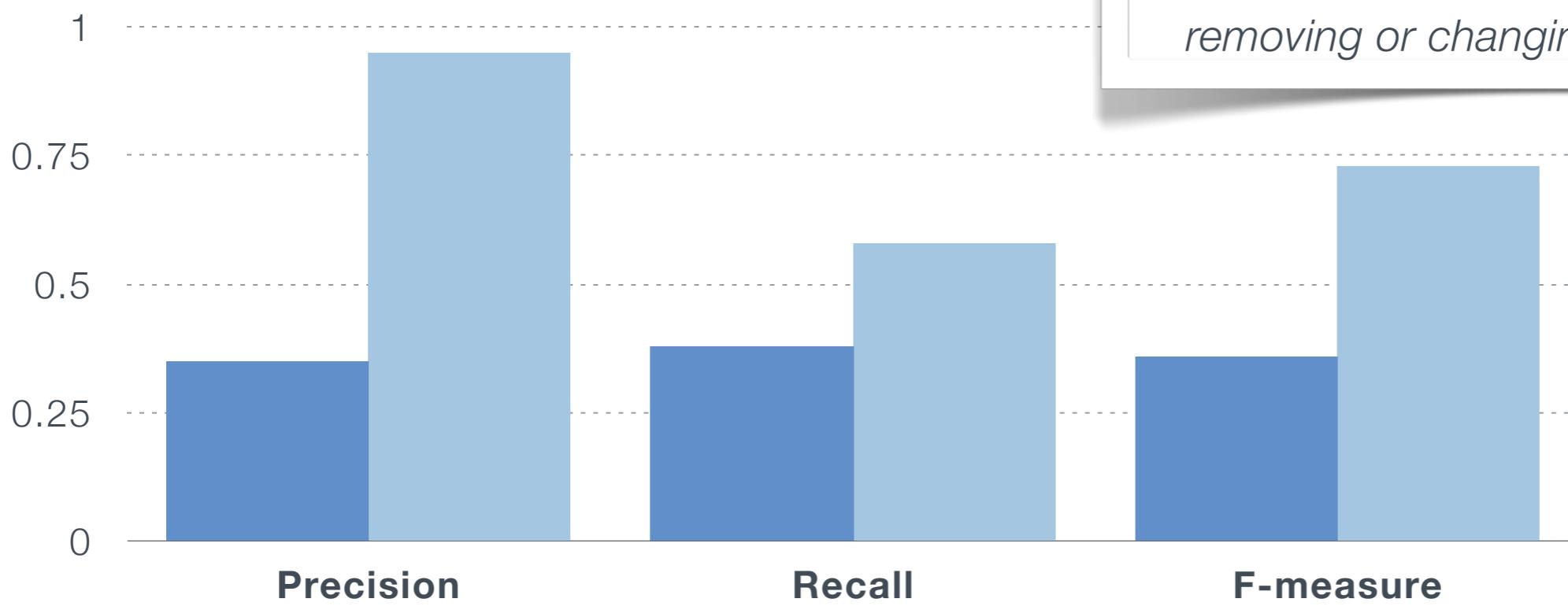
Lexical information gathered
(recursively) from leaves

RE SULTS

CLONE DETECTION

- Comparison with another (pure) AST-based clone detector
- Comparison on a system with randomly seeded clones

CloneDigger Tree Kernel Tool



Results refer to clones where code fragments have been modified by adding/removing or changing code statements

CONCLUSIONS

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."
(cit. Lehman's First Law of Software Evolution)



Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the **85-90%** of the total software costs.

SOURCE CODE
LEXICAL INFORMATION

THESIS
CONTENTS

Contributions to three relevant and related **open issues** in Software Maintenance

Reverse Engineering & Original Software Techniques

A circular diagram centered on the word "engineering". The innermost ring contains the word "engineering" in large, bold, black letters. The next ring outwards contains the words "original", "replace", "systems", "techniques", and "subsequently". The third ring contains "program", "understanding", "products", "without", "several", and "years". The fourth ring contains "procedures", "simply", "are", "the", "same", "as", "those", "used", "in", "the", "design", "of", "the", "product", "and", "therefore", "can", "be", "applied", "to", "any", "product", "without", "modification". The outermost ring contains "chemical", "commercial", "industrial", "computer", "analysis", "synthesis", "advantage", "application", "spur", "additional", "component", "decision", "device", "design", "engineering", "method", "military", "knowledge", "legacy", "original", "replace", "systems", "techniques", "subsequently", "value", "electronic", "hardware", "circuit", "function", "industrial", "design", "mechanical", "operations", "principles", "object", "region", "process", "procedure", "program", "understanding", "products", "without", "several", "years".

1. SOFTWARE RE-MODULARIZATION

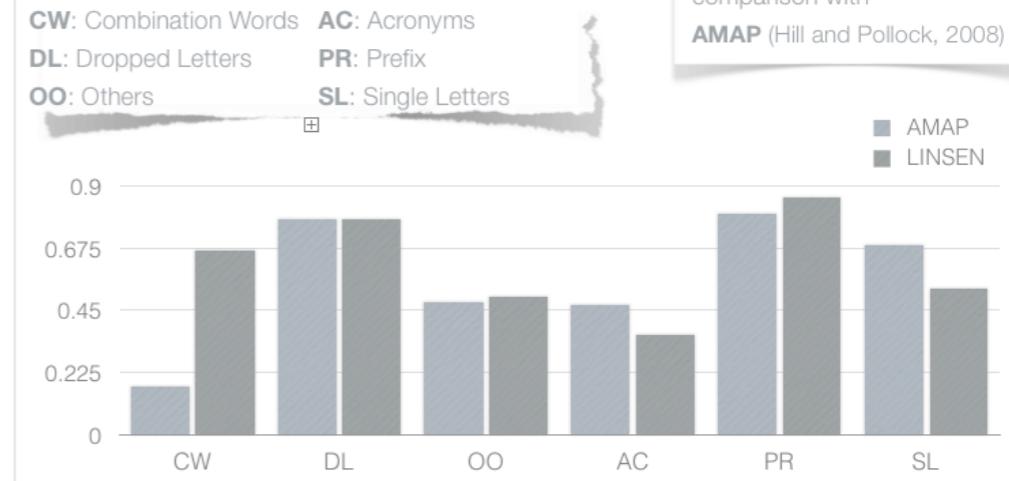
2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

RE
SULTS
2

EXPANSION

CW: Combination Words **AC:** Acronyms
DL: Dropped Letters **PR:** Prefix
OO: Others **SL:** Single Letter



CONCLUSIONS

SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."

(cit. Lehman's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the **85-90%** of the total software costs.



SOURCE CODE
LEXICAL INFORMATION

THESIS
C O N T I
I B U N T I
O N S

Contributions to three relevant and related **open issues** in Software Maintenance

Reverse engineering

1. SOFTWARE RE-MODULARIZATION

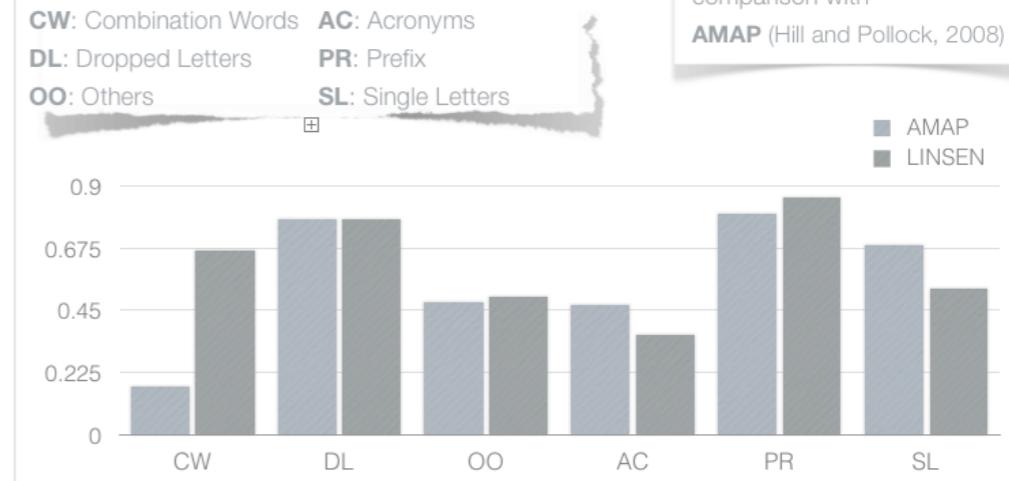
2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

RE
SULTS

EXPANSION

CW: Combination Words **AC:** Acronyms
DL: Dropped Letters **PR:** Prefix
OO: Others **SL:** Single Lett



CONCLUSIONS

SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."

(cit. Lehman's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the **85-90%** of the total software costs.



SOURCE CODE INFORMATION
LEXICAL

```
package CH.ifaf.draw.standard;

import java.awt.*;
import CH.ifaf.draw.framework.*;

/**
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 * <b>Design Patterns</b><br>
 * 
 * <b>NullObject</b><br>
 * NullObject enables to treat handles that don't do
 * anything in the same way as other handles.
 *
 */
public class NullHandle extends LocatorHandle {

    /**
     * The handle's alt purpose is to protect its locator.
     */
    public void draw(JavaFigure figure, Locator locator) {
        if (figure == null) {
            return;
        }
        Rectangle r = figure.getRectangle();
        if (r != null) {
            Graphics g = r.getGraphics();
            g.setRectColor(Color.black);
            g.setFillColor(Color.red);
            g.fillRect(r);
        }
    }
}
```

THESIS
CONTENTS

Contributions to three relevant and related
open issues in Software Maintenance

Reverse engineering

1. SOFTWARE RE-MODULARIZATION

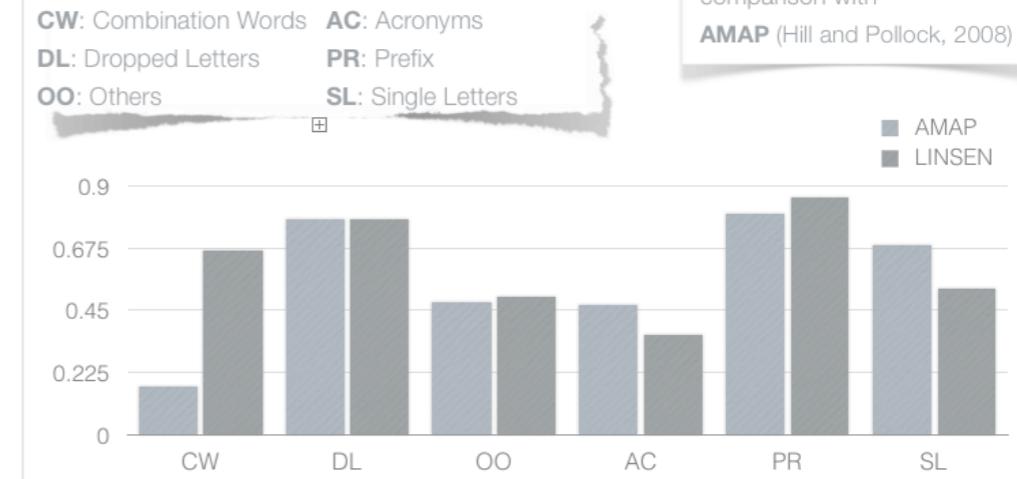
2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

RE
SULTS

EXPANSION

CW: Combination Words **AC:** Acronyms
DL: Dropped Letters **PR:** Prefix
OO: Others **SL:** Single Lett



CONCLUSIONS

SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."

(cit. Lehmann's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the
85-90% of the total software costs.



SOURCE CODE INFORMATION

THESIS
CONTENTS

Contributions to three relevant and related
open issues in Software Maintenance

Reverse engineering

A word cloud centered around the words "complex", "algorithms", "data", "statistical", "learning", and "graphs". Other visible words include "questions", "algorithm", "theoretical", "relations", "structure", "objects", "network", "behavior", "different", "people", "interactions", "many", "online", "theory", "deal", "want", "use", "amounts", "social", "extract", "analysis", "guarantees", "describe", "classification", "able", "similarity", "analyse", "homogeneous", "Machine", "urgent", and "clustering".

1. SOFTWARE RE-MODULARIZATION

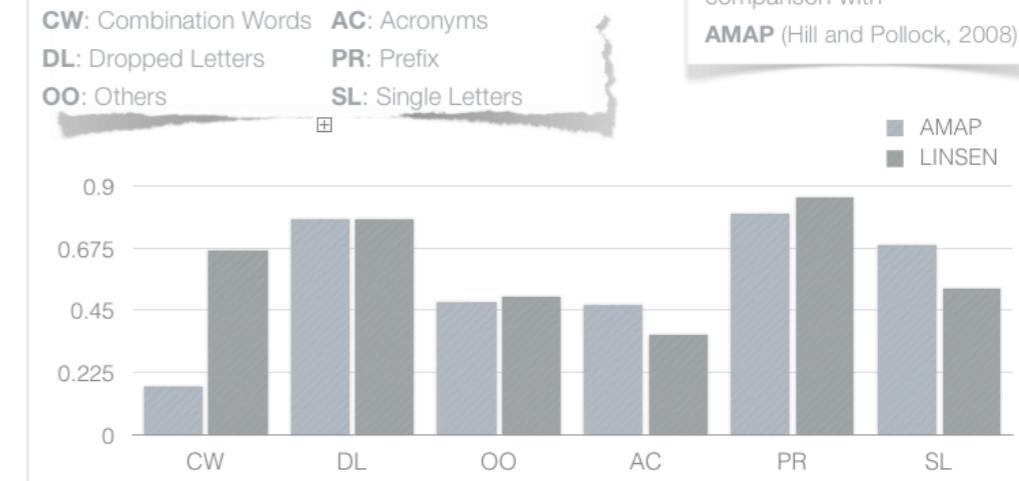
2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

RESULTS # 2

EXPANSION

CW: Combination Words **AC:** Acronyms
DL: Dropped Letters **PR:** Prefix
OO: Others **SL:** Single Lett



CONCLUSIONS

SOFTWARE MAINTENANCE

"A software system must be continually adapted during its overall life cycle or it progressively becomes less satisfactory."

(cit. Lehman's First Law of Software Evolution)

Software Maintenance represents the most **expensive, time consuming** and challenging phase of the whole development process.

Software Maintenance could account up to the **85-90%** of the total software costs.



SOURCE CODE INFORMATION
LEXICAL

THESIS
CONTENTS

Contributions to three relevant and related
open issues in Software Maintenance

Reverse engineering

engineering

1. SOFTWARE RE-MODULARIZATION

2. SOURCE CODE NORMALIZATION

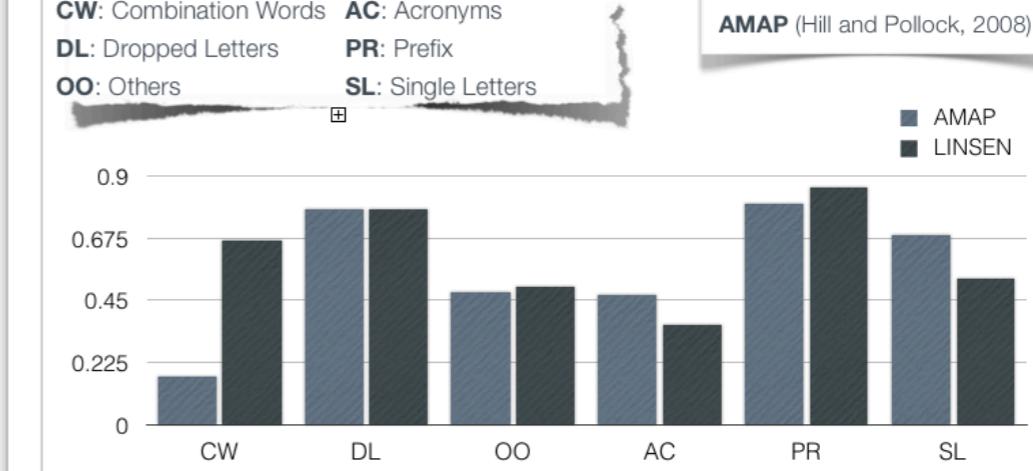
3. CLONE DETECTION

RE
SULTS

EXPANSION

CW: Combination Words **AC:** Acronyms
DL: Dropped Letters **PR:** Prefix
OO: Others **SL:** Single Lett

Accuracy Rates for the comparison with **AMAP** (Hill and Pollock, 2008)



CONCLUSIONS

SOFT



Software
expensive
phase of

Software M
85-90

SOURCE CODE
LEXICAL INFORMATION

FUTURE RESEARCH DIRECTIONS

- **Re-modularization:**

- Integrate code normalization algorithm (LINSEN) to lexical analysis

- **Code Normalization:**

- Investigate the contributions of different **sources** of information used for disambiguations

- **Clone Detection:**

- Combine Static (and Kernel methods) with Dynamic Analysis

THANK YOU

Valerio Maggio

Ph.D. Student, University of Naples “Federico II”

valerio.maggio@unina.it

June 5th, 2013 - *Ph.D. Defense*