Dottorato in Scienze Computazionali e Informatiche, XXV Ciclo

UNSUPERVISED MACHINE LEARNING FOR SOFTWARE MAINTENANCE

Thesis Advisors:

Dr. Sergio Di Martino

Dr. Anna Corazza

Ph.D. Candidate:

Valerio Maggio

June 5th, 2013

THESIS
G O A L

UNSUPERVISED MACHINE LEARNING FOR SOFTWARE MAINTENANCE

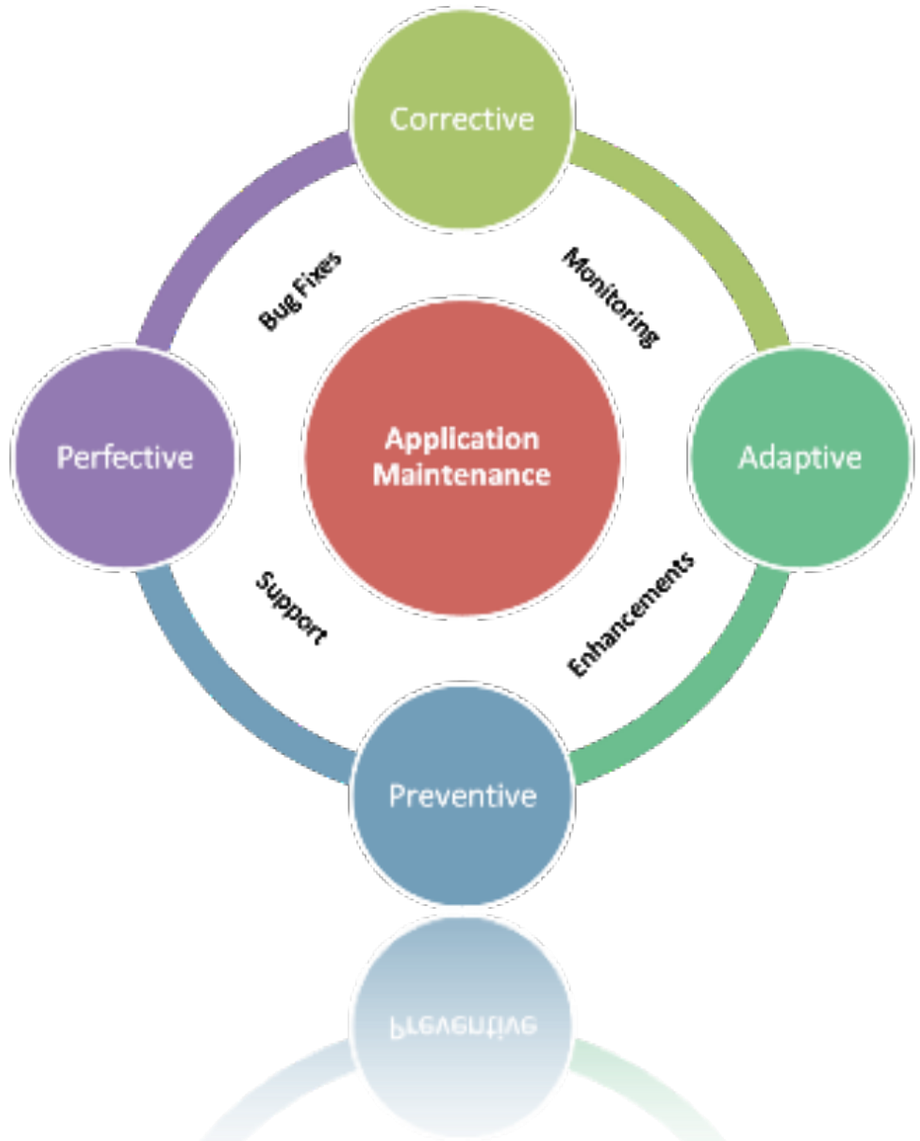
Define and experimentally evaluate solutions (techniques and prototype tools) for **automatic software analysis** to support software **maintenance** activities.

These solutions exploit (**un**supervised) **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)

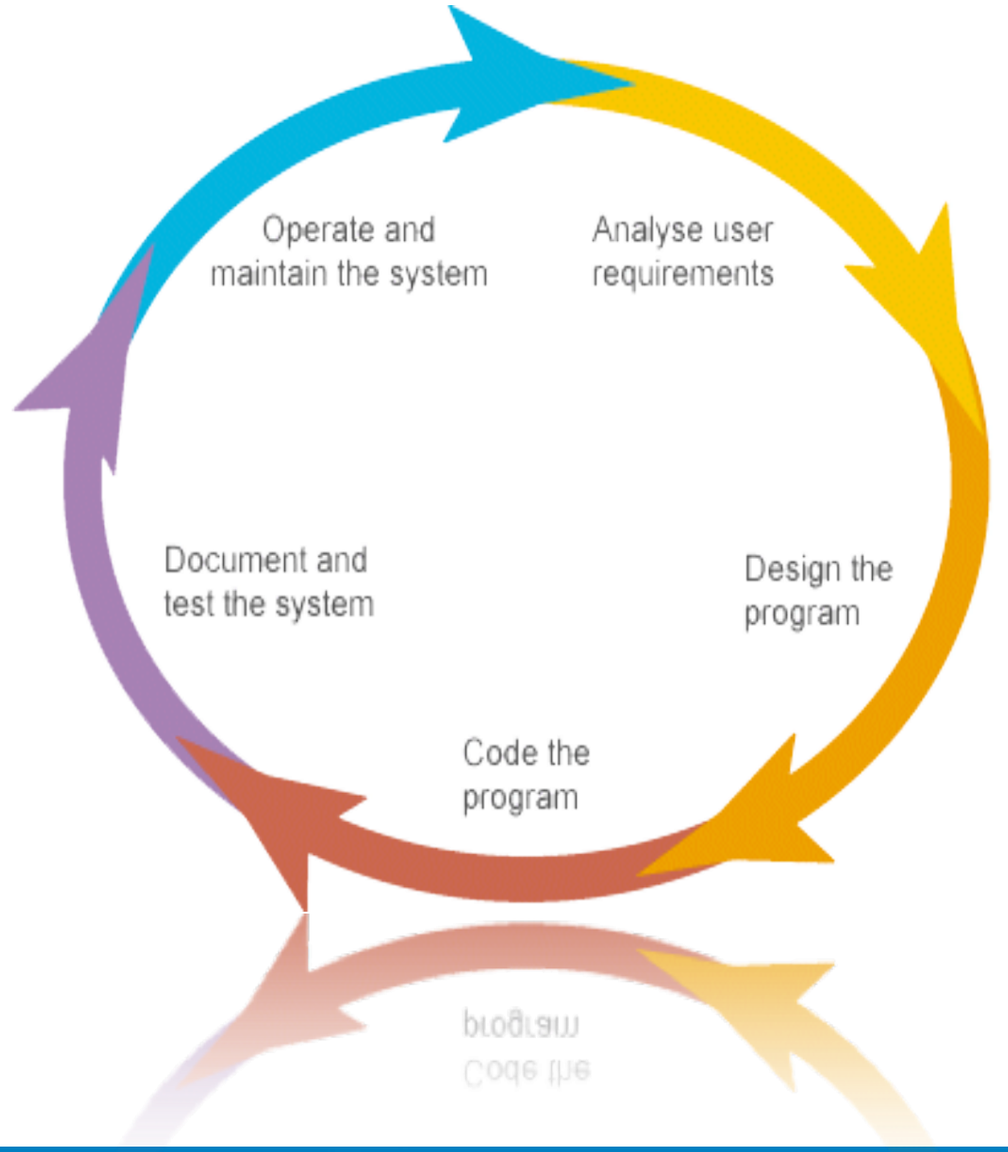
There exist different **types** of Software Maintenance.



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.

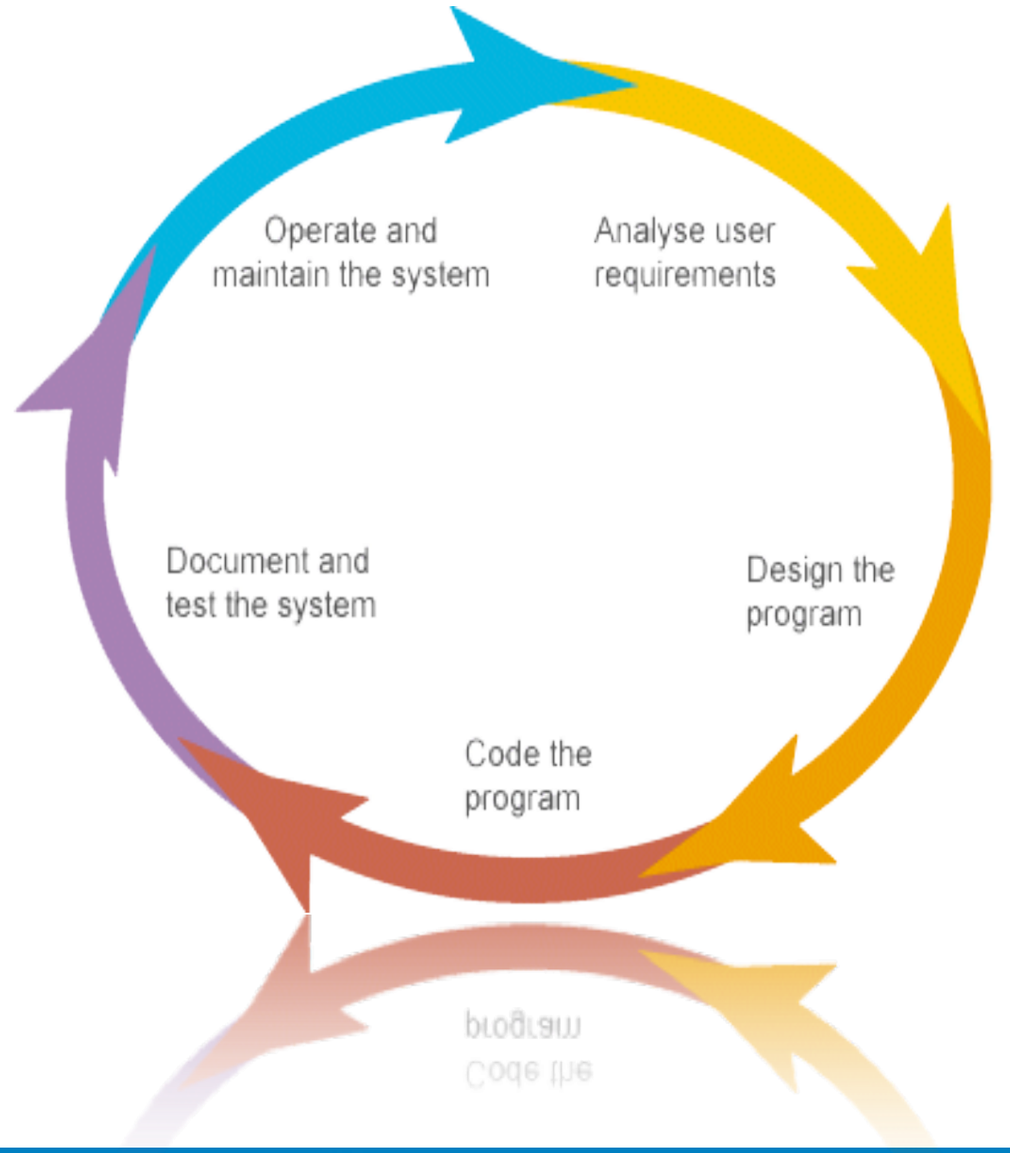


SOFTWARE MAINTENANCE

> *"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

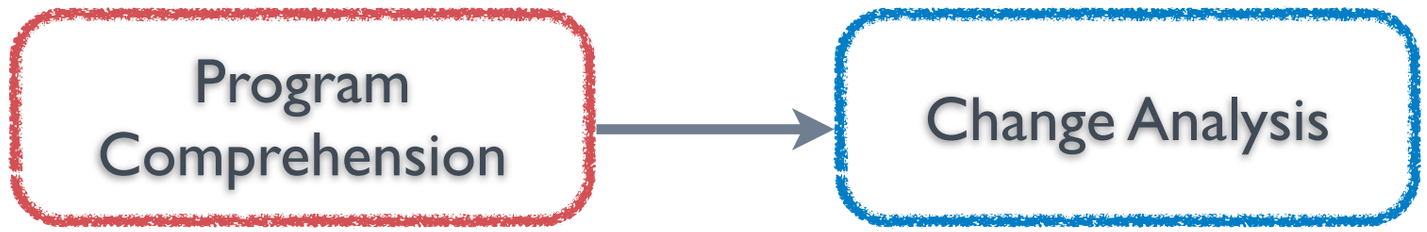
> *“Software Maintenance is about change!”*
> (cit. S. Jarzabek)

Change Analysis

ISSUES

SOFTWARE MAINTENANCE

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

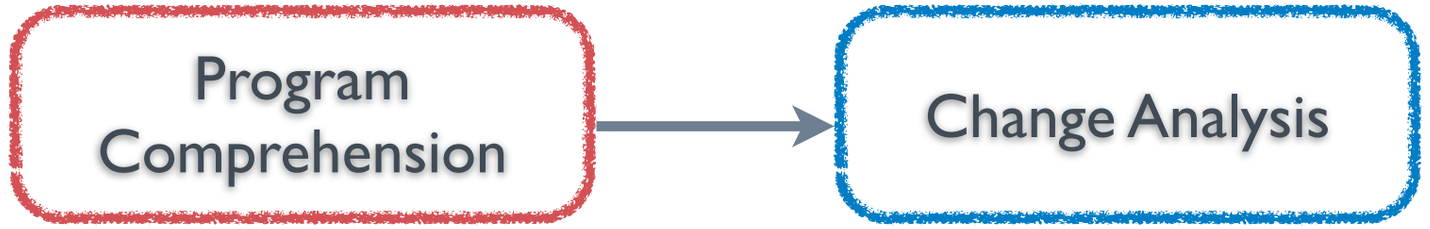


The **documentation** is usually scarce or **not up to date!**

ISSUES

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

ISSUES

SOFTWARE MAINTENANCE

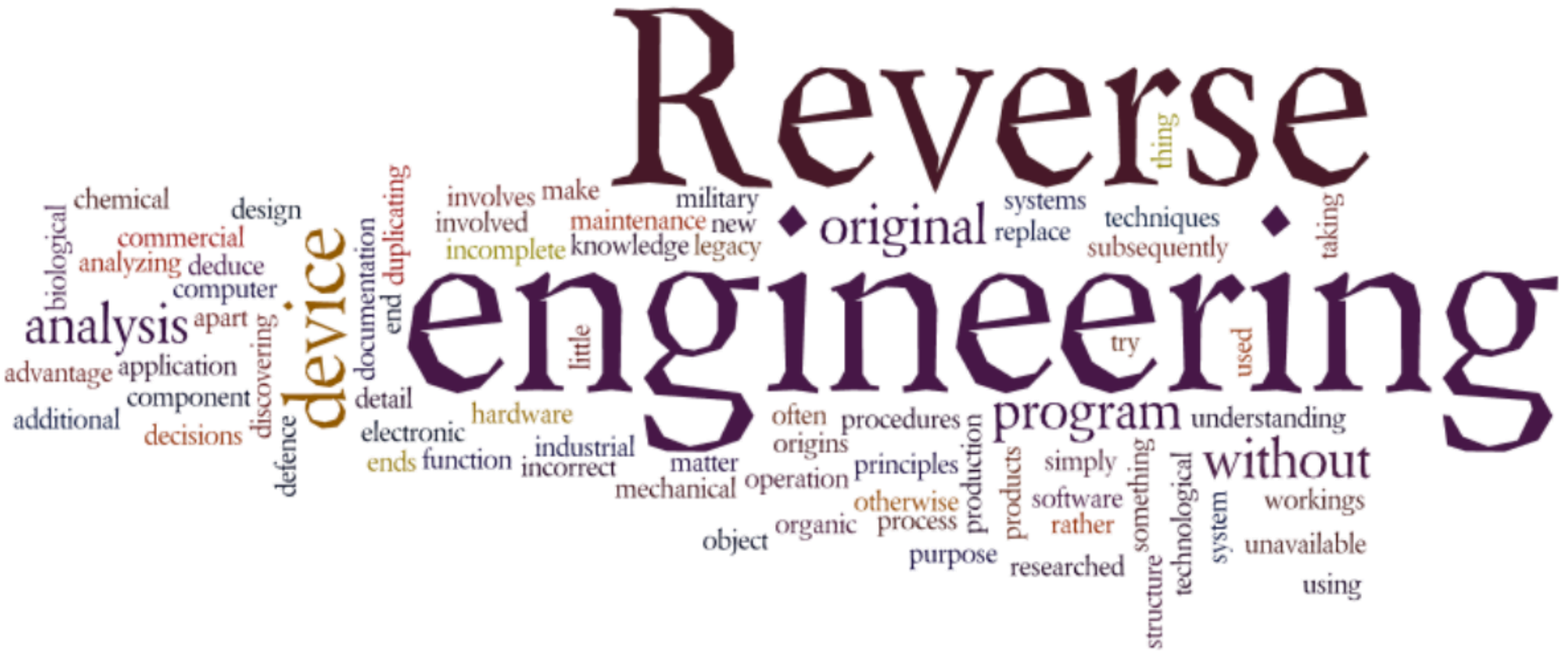
> *“Software Maintenance is about change!”*
> (cit. S. Jarzabek)

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

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

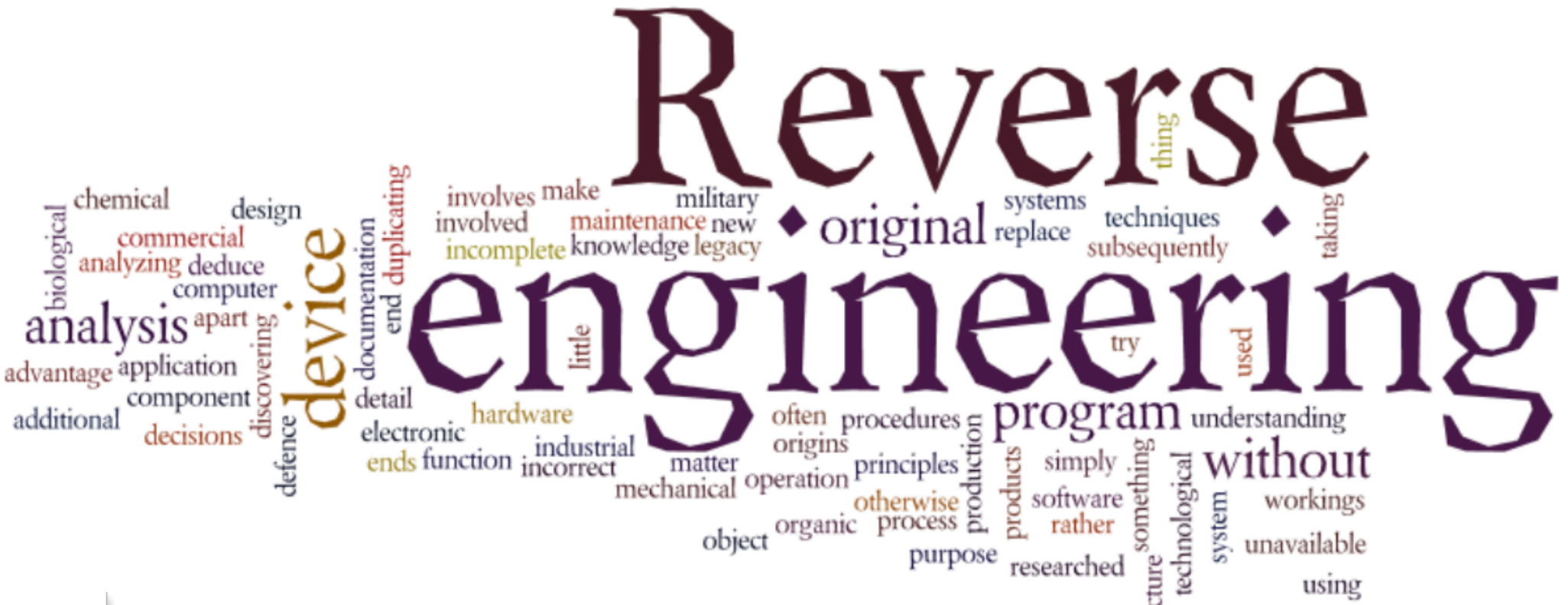
REVERSE ENGINEERING

- 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



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

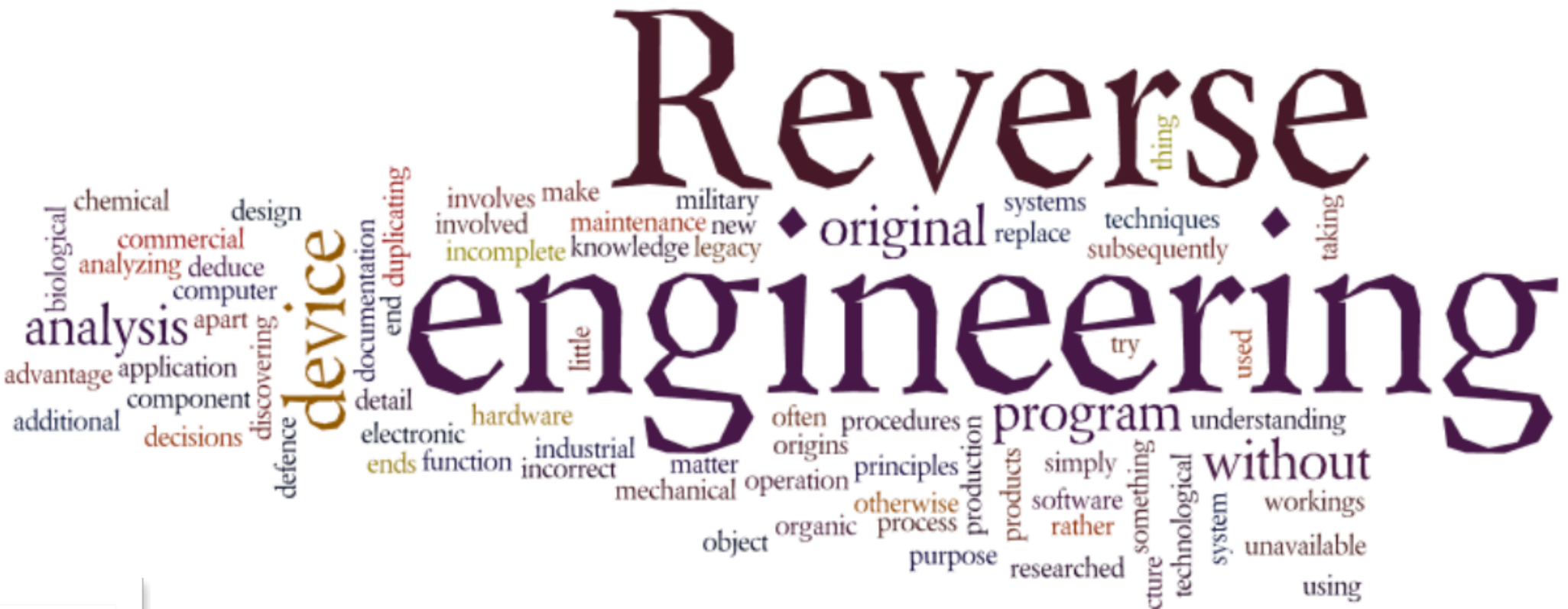


STATIC ANALYSIS

DYNAMIC ANALYSIS

REVERSE ENGINEERING

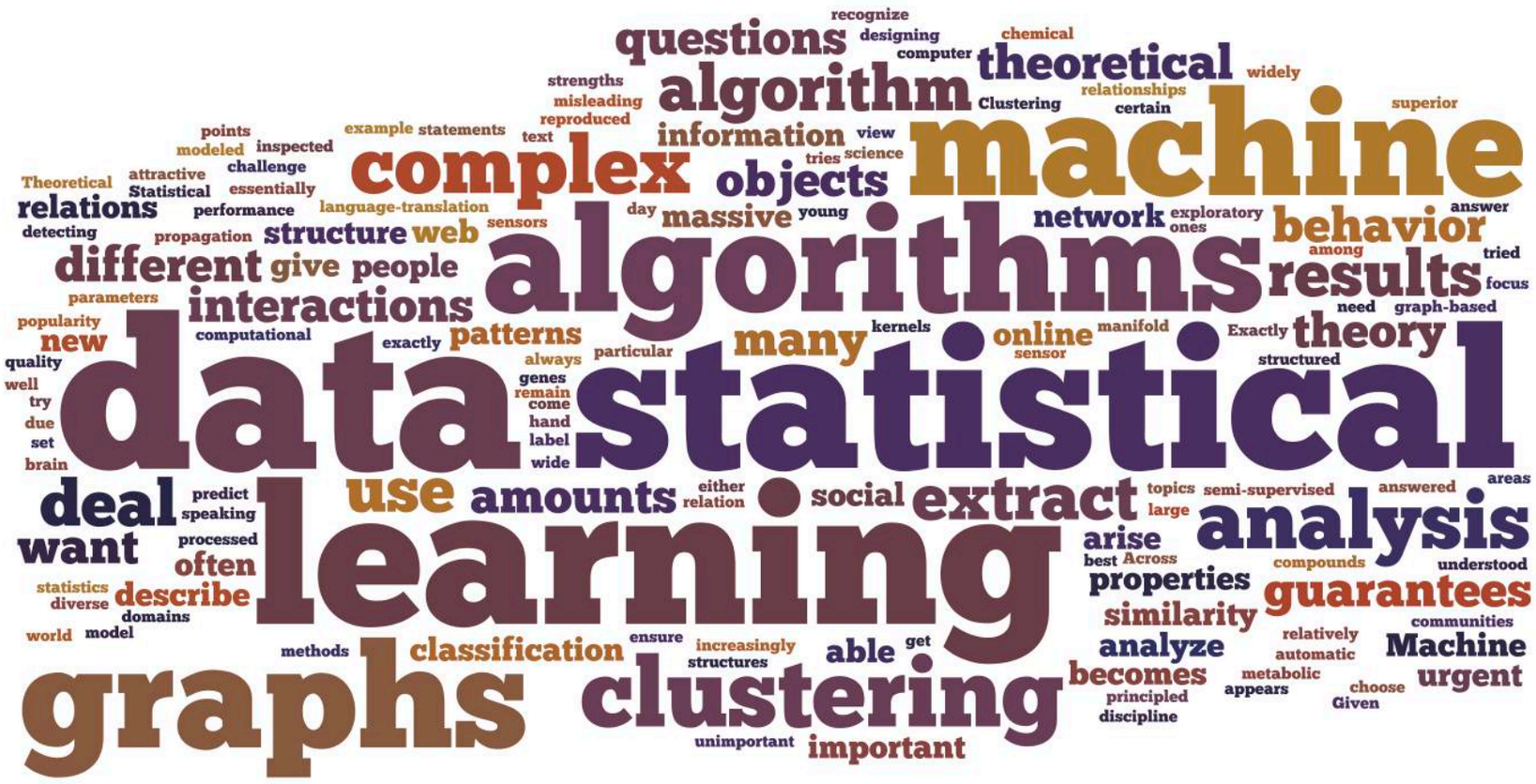
- Definition of **tools** and **techniques** to support maintenance activities
  - **Goal**: Build higher-level *software models* in an **automatic fashion** gathering information from the **source code** or any other document
  - **Goal**: To aid the comprehension of the system



STATIC ANALYSIS

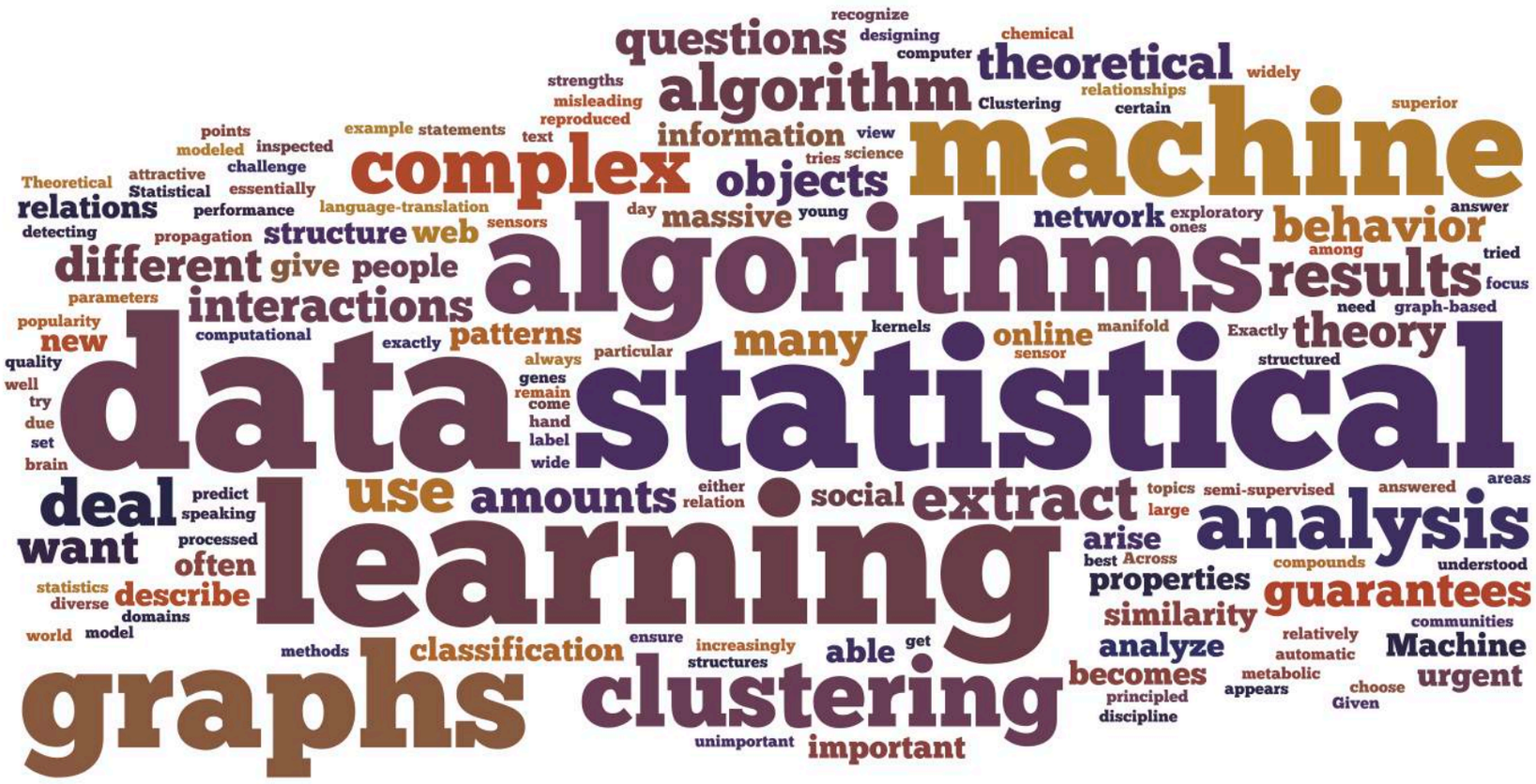
DYNAMIC ANALYSIS

MACHINE LEARNING



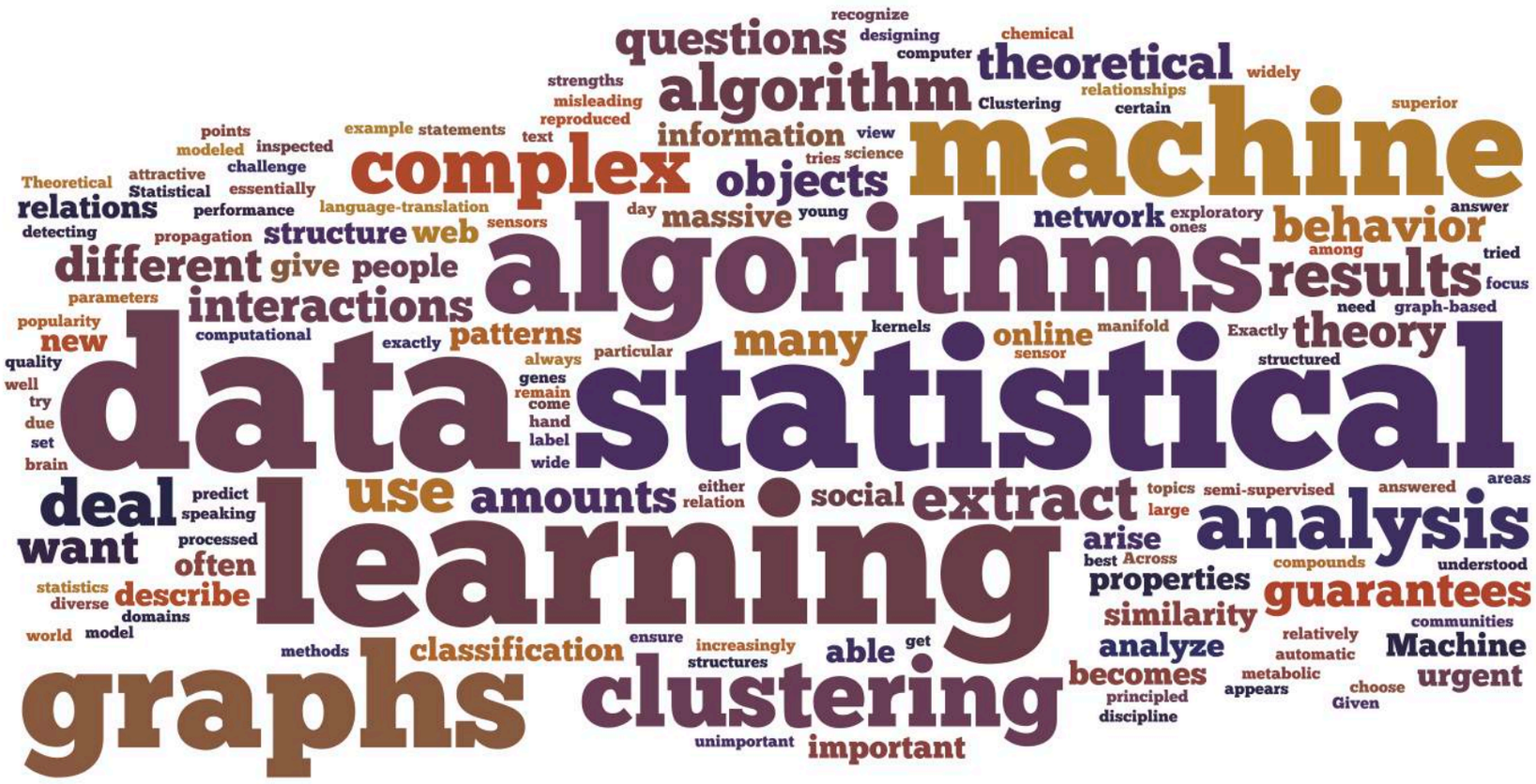
- Provides *computational effective* solutions to analyze **large** data sets

MACHINE LEARNING



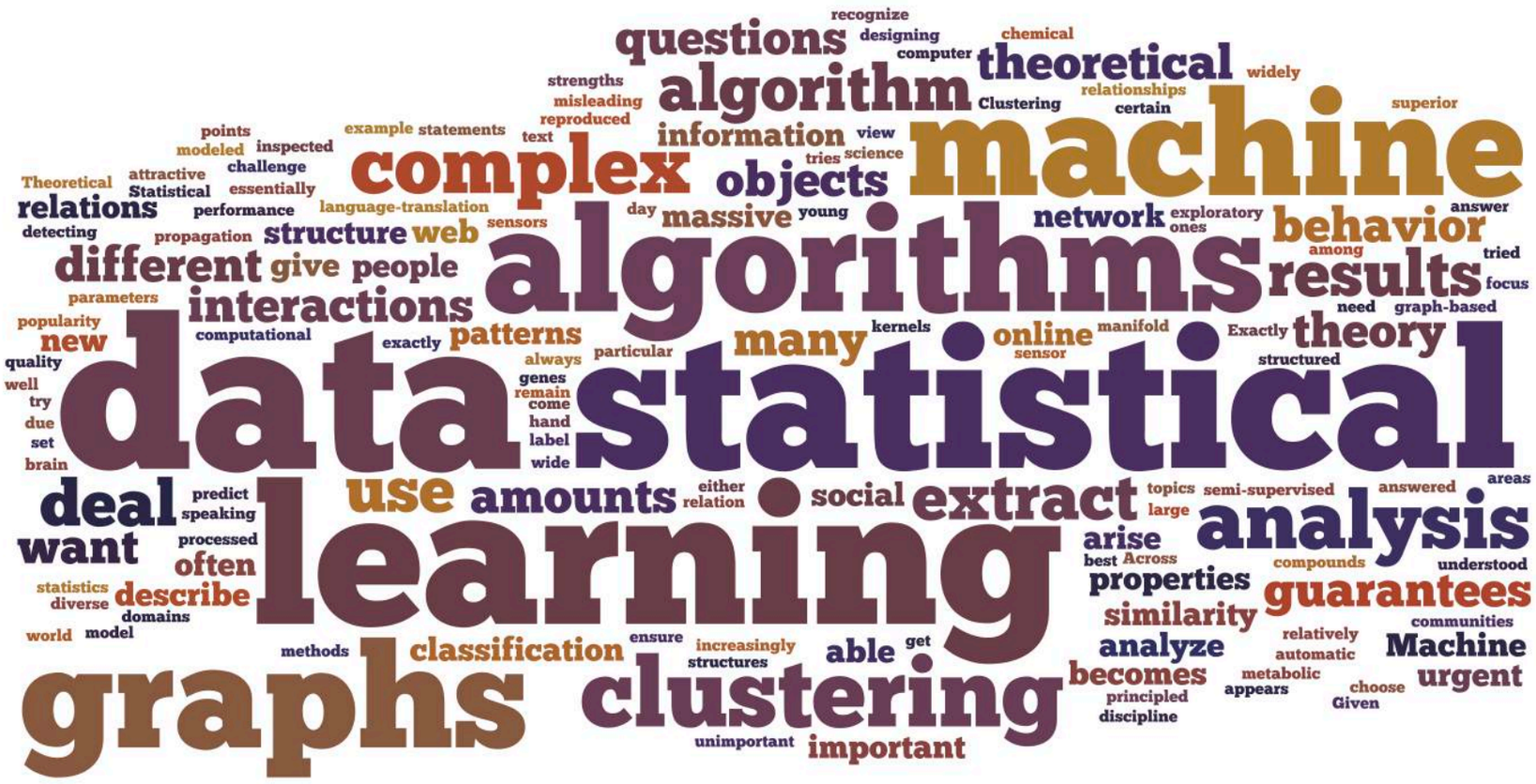
- Provides *computational effective* solutions to analyze **large** data sets
- Provides solutions that can be **tailored** to different tasks/domains

MACHINE LEARNING



- 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



- 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

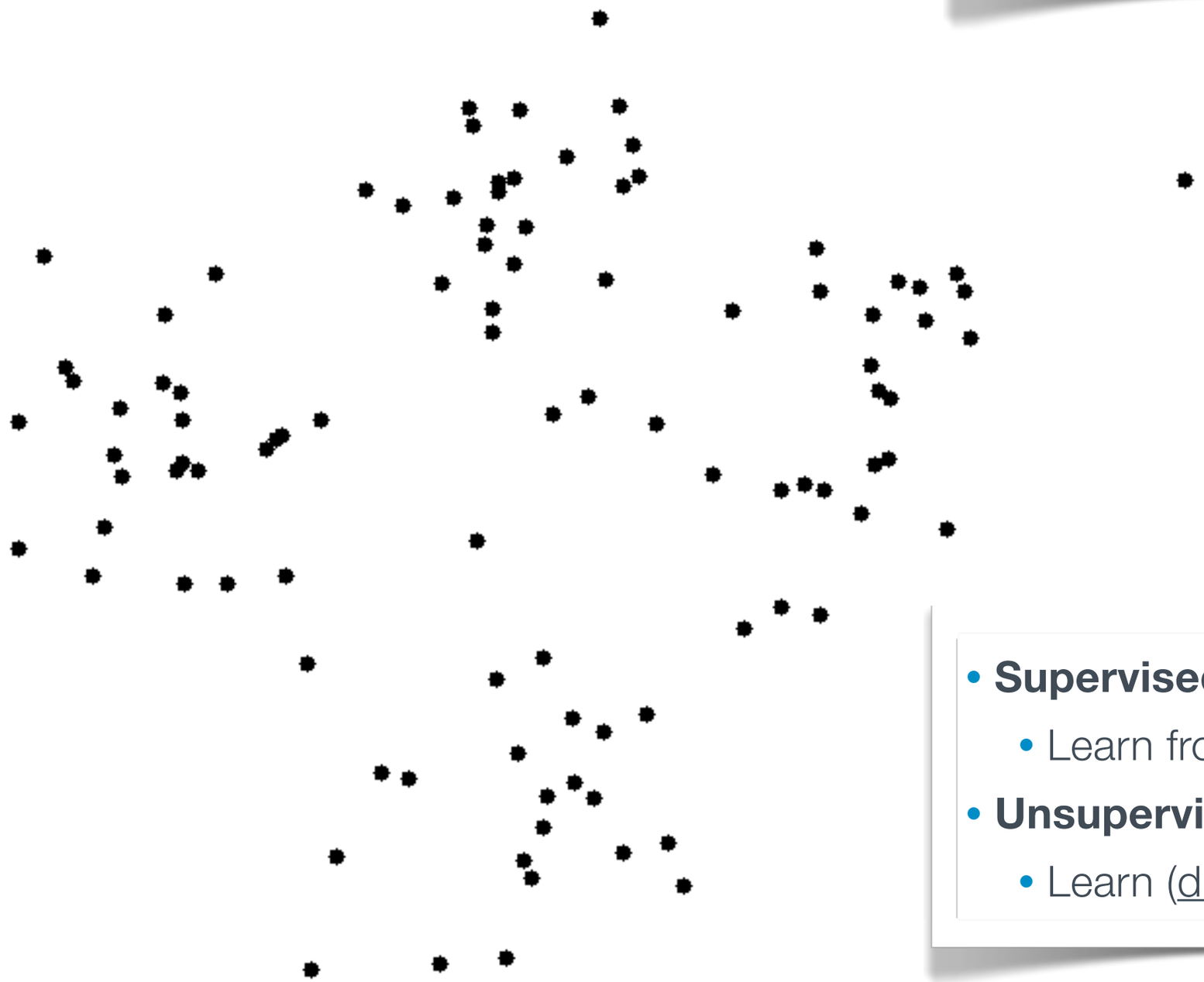
UNSUPERVISED LEARNING

Learn by examples

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

UNSUPERVISED LEARNING

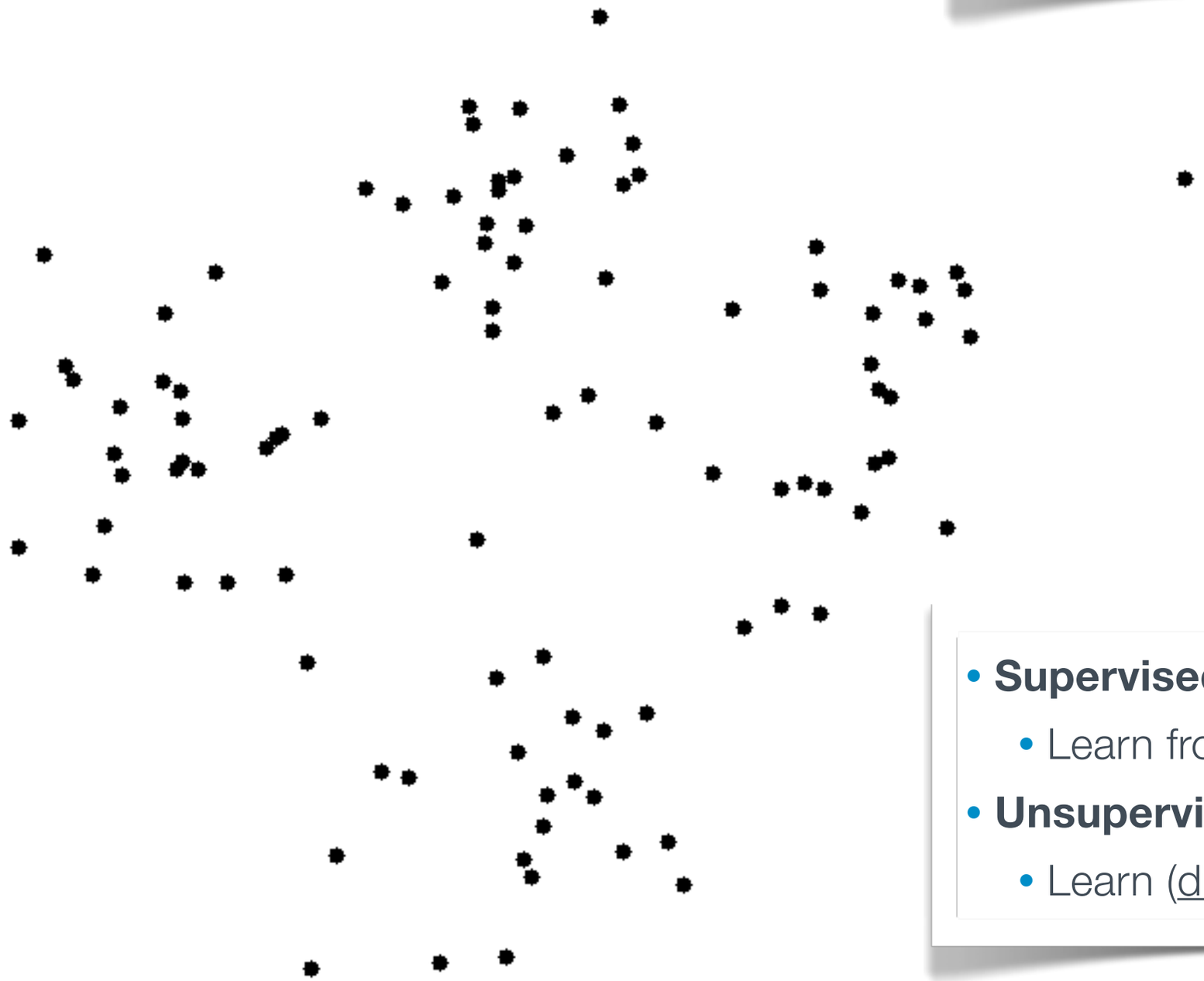
Learn by examples



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

UNSUPERVISED LEARNING

Learn by examples

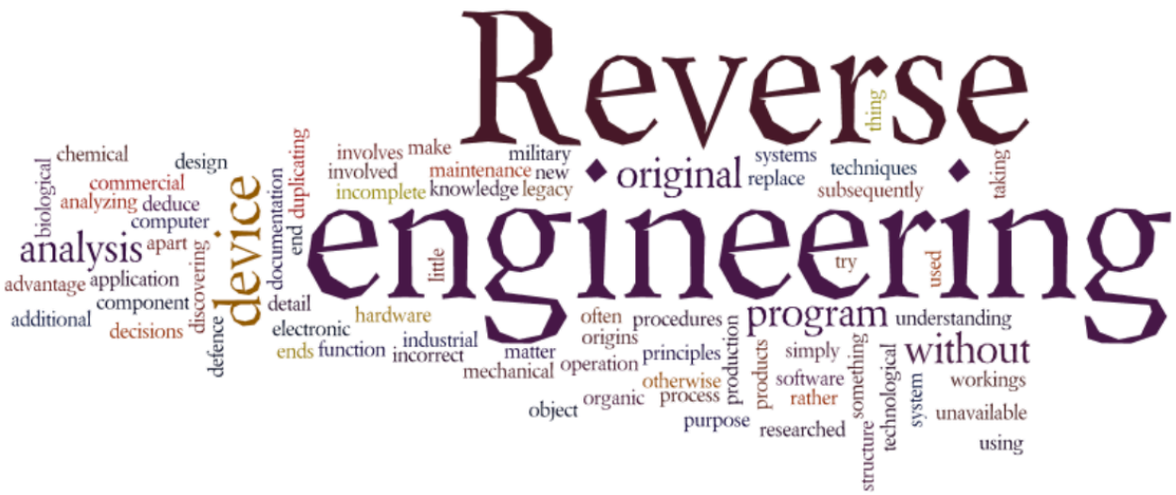


- **Supervised Learning:**
  - Learn from **labelled** samples
- **Unsupervised Learning:**
  - Learn (directly) from the data

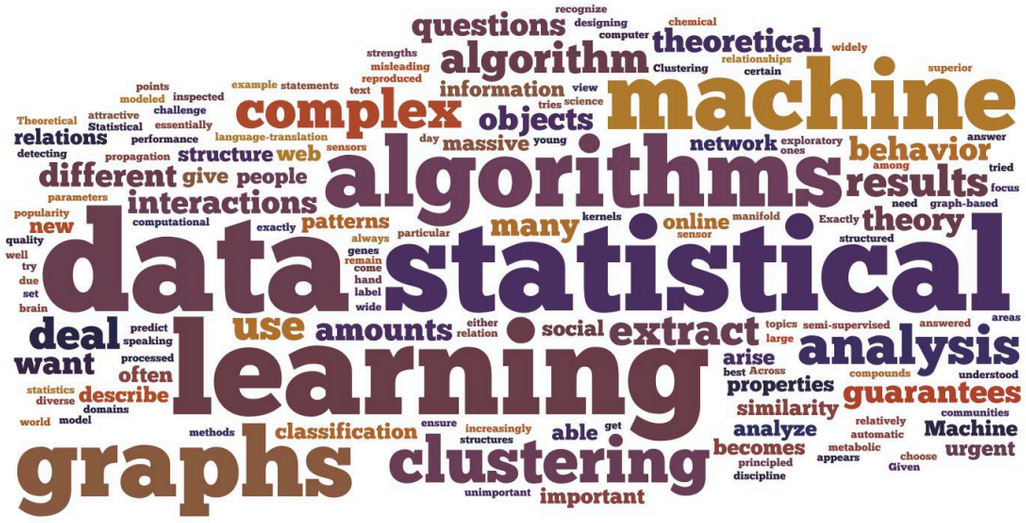
(+) No cost of labeling samples

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

THESIS CONTRIBUTIONS

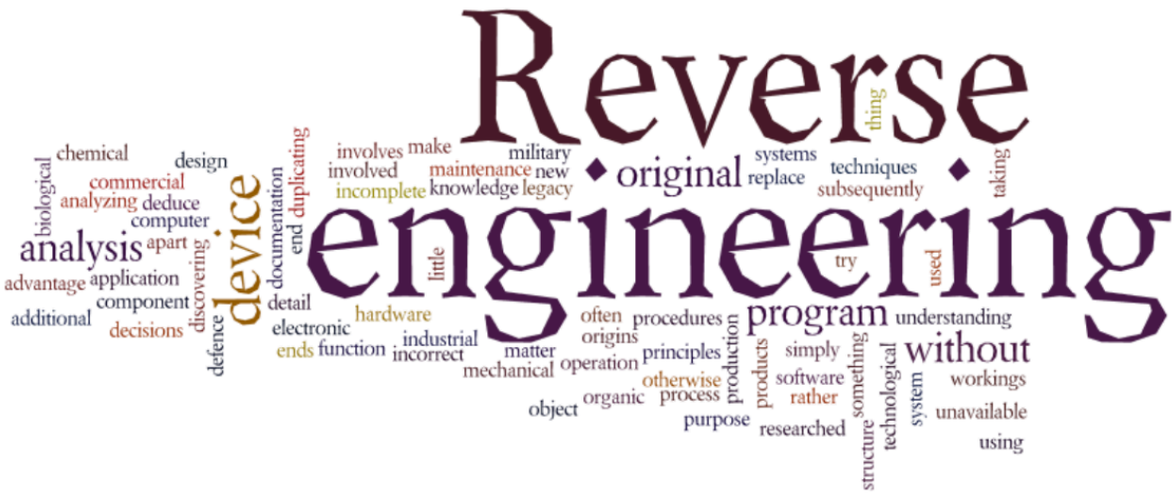


&

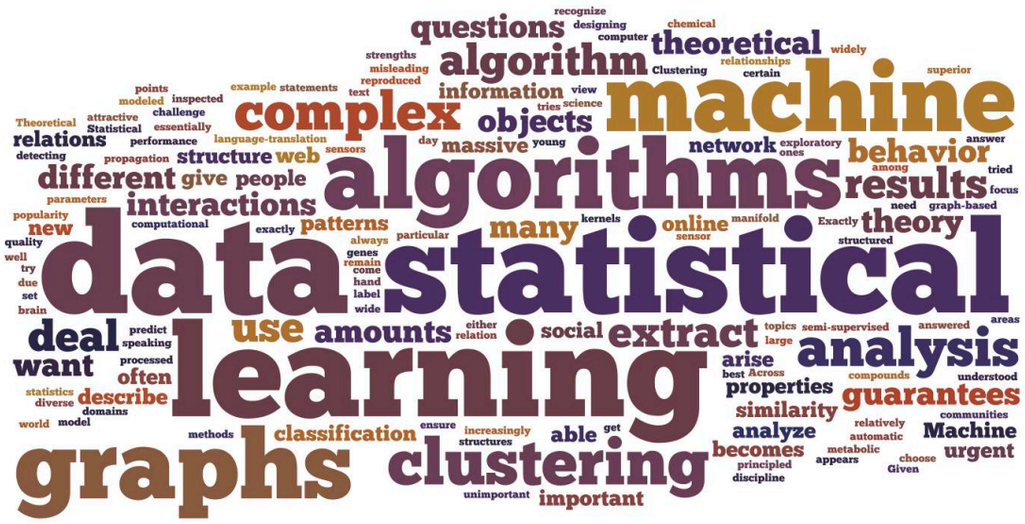


THESIS CONTRIBUTIONS

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

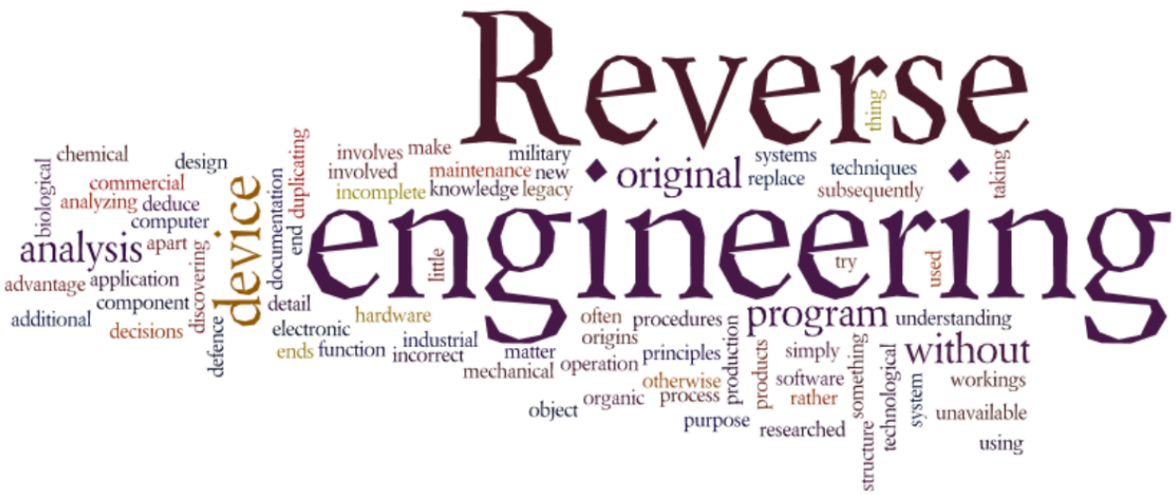


&

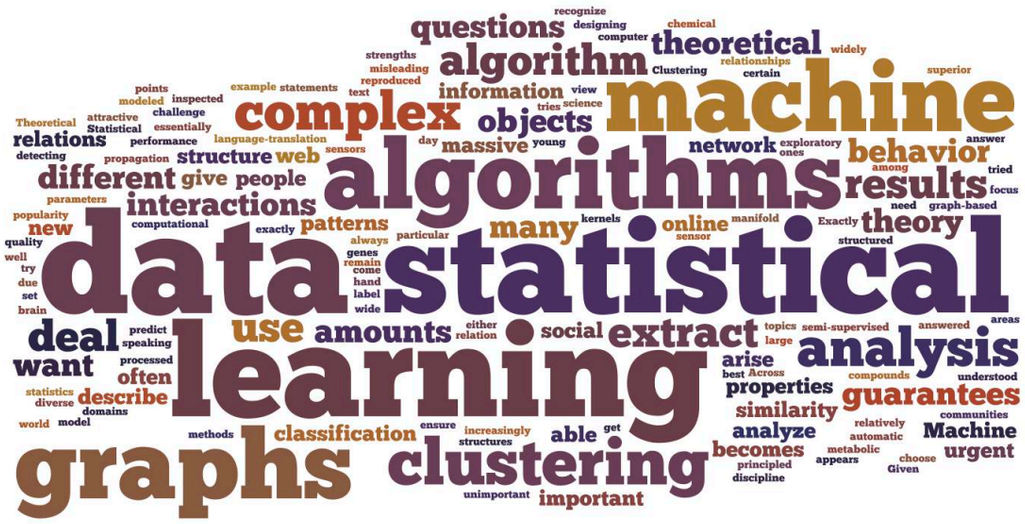


THESIS CONTRIBUTIONS

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



&



1. SOFTWARE RE-MODULARIZATION

2. SOURCE CODE NORMALIZATION

3. CLONE DETECTION

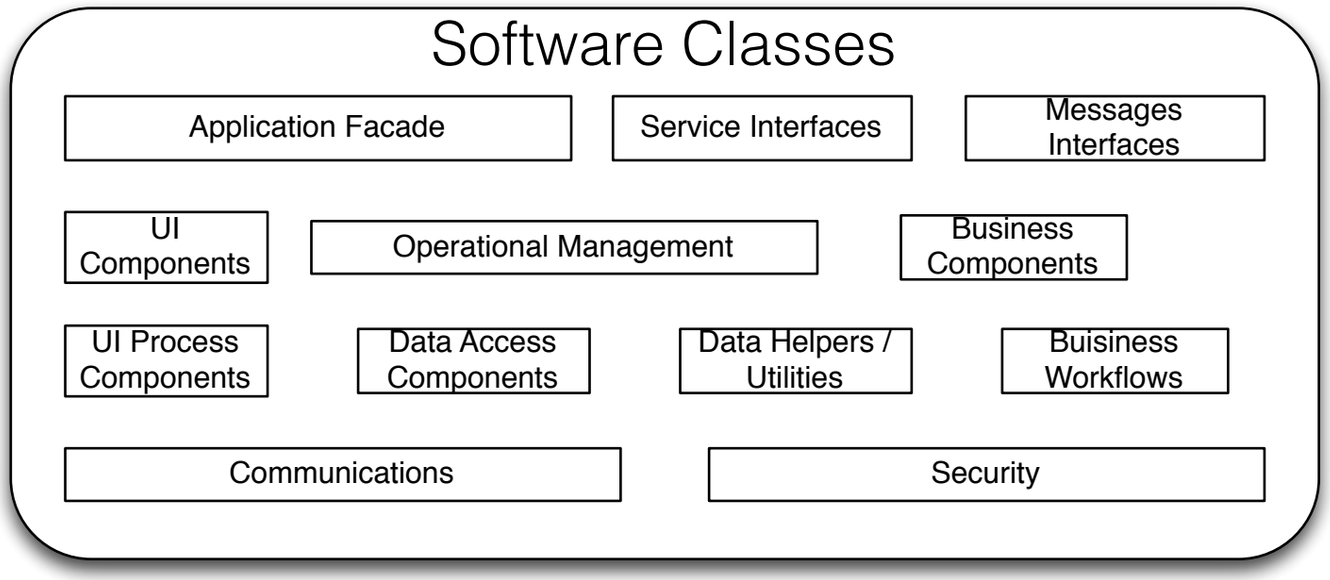
ISSUE
#1

SOFTWARE RE-MODULARIZATION

PROBLEM STATEMENT

SOFTWARE RE-MODULARIZATION

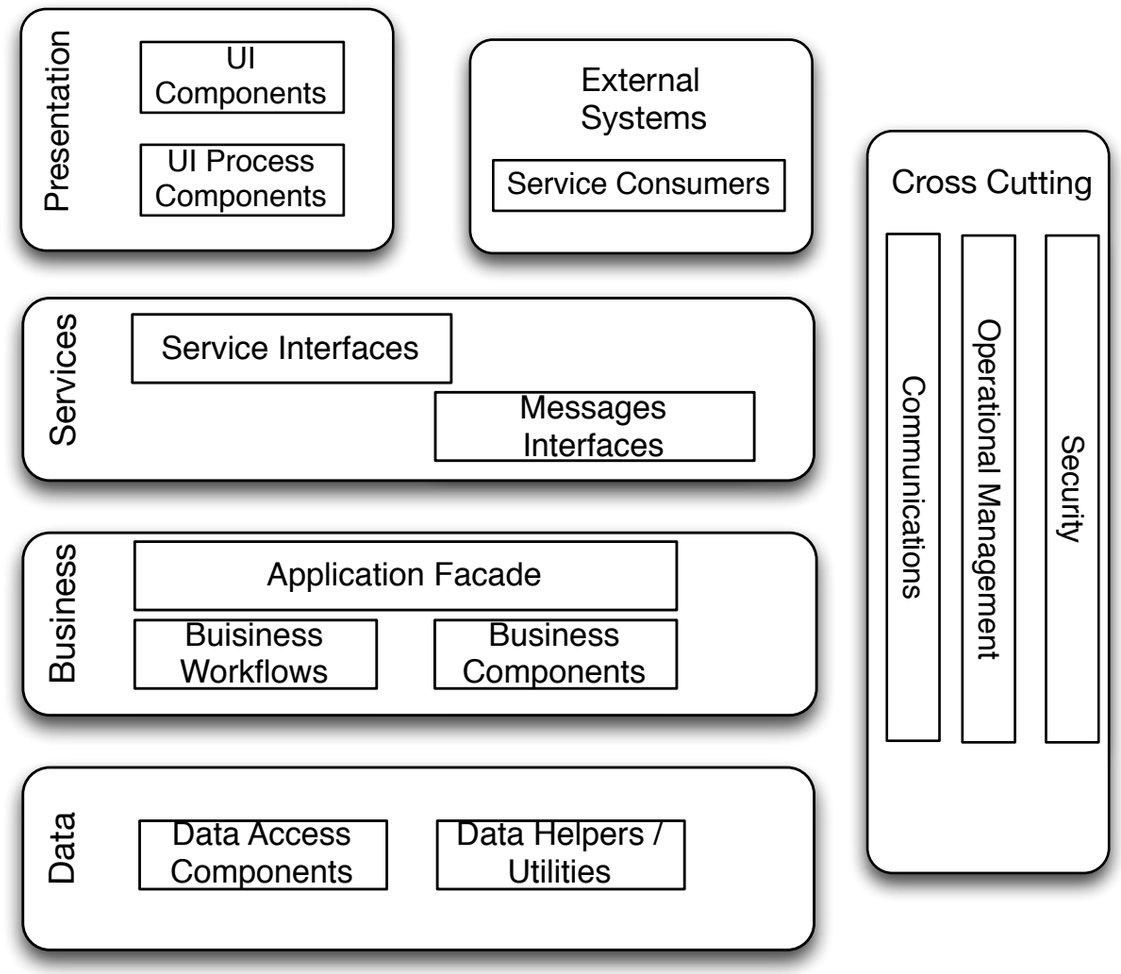
Re-modularization provides a way to support software maintainers by automatically grouping together (**clustering**) “related” software **classes**



PROBLEM STATEMENT

SOFTWARE RE-MODULARIZATION

Re-modularization provides a way to support software maintainers by automatically grouping together (**clustering**) “related” 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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);
    }
}
```

SOURCE CODE

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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
 * dimmed framed rectangle.
 */
public void draw(Graphics g) {
    Rectangle displayBox();
    g.setColor(Color.black);
    g.drawXORRect(d);
}
}
```

SOURCE CODE **LEXICAL** INFORMATION

(IDEA): Exploit the **lexical information** gathered from the source code to produce **clusters** of classes that are lexically related.

SOURCE CODE **LEXICAL** INFORMATION

(IDEA): Exploit the **lexical information** gathered from the source code to produce **clusters** of classes that are lexically related.

State of the Art: Information Retrieval (**IR**) based approaches.

```
/**
 * 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

IR INDEXING

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);
}
```

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

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

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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);
    }
}
```

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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);
    }
}
```

Class Names

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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);
    }
}
```

Class Names

Attribute Names

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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;

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

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

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

Class Names

Attribute Names

Method Names

Method Names

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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;

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

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

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

Class Names

Attribute Names

Method Names

Parameter Names

Parameter Names

Method Names

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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;

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

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

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

Comments

Class Names

Comments

Attribute Names

Method Names

Parameter Names

Comments

Parameter Names

Method Names

SOURCE CODE **ZONES**

LEXICAL 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>
 * <img src="images/red-ball-small.gif" width=6 height=6 alt=" o ">
 * <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;

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

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

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

Comments

Class Names

Comments

Attribute Names

Method Names

Parameter Names

Source Code

Comments

Parameter Names

Method Names

Source Code

```
/**
 * 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 INDEXING

1. Tokenization

```java
/**
 * 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, are,
NullHandle,
box, r,
draw, g,
Graphics,
color,
displayBox,
...

2. Normalization

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

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

SOURCE CODE LEXICON

JFreeChart

```java
/**
 * 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

```java
/**
 * 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);
```

JUnit

```java
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);
    }
```

Xerces

```java
/Returns a string representation of the specified content spec
ublic 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;
          }
```

SOURCE CODE LEXICON

JFreeChart

```java
/**
 * 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;
}
```

- Good lexicon in **every Zone**!

SOURCE CODE LEXICON

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);
```

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

SOURCE CODE LEXICON

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);
    }
```

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

SOURCE CODE 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;
}
```

```
ublic class Test
    private TestR
    private int f
    private int f
    private int f
    private int f

    public void t
        TestCase
            @Over
            publi
                f
            }
        };
        test.run(
        assertEqu
        assertEqu
    }
    public void t
        TestCase
            @Over...
            public void runTest() {
            }
```

```
* 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,

. This operand is

truction that

visited. This opcode

d. This operand is
orm, or the type
).
```

```
//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

SOURCE CODE 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;
}
```

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
ublic 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;
```

CONTRIBUTION

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

CONTRIBUTION

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

CONTRI BUTION

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

ArrayHolder

GraphicsHolder

PropertyEditPanel

NumberAxisPropertyEditPanel

DateAxisPropertyEditPanel

ChartFrame

LineAndShapeRenderer

FigureLegend

LegendItem

HorizontalNumberAxis

StandardLegend

StackedVerticalBarRenderer

Non-extreme Distribution

property

VerticalNumberAxis

VerticalXYDataPlot

DataPlot

VerticalPlot

SampleXYDataset

SampleXYDatasetThread

LinearPlotFitAlgorithm

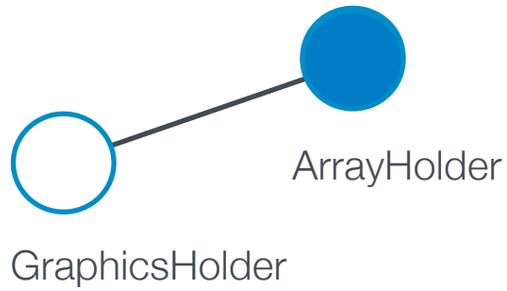
Plot

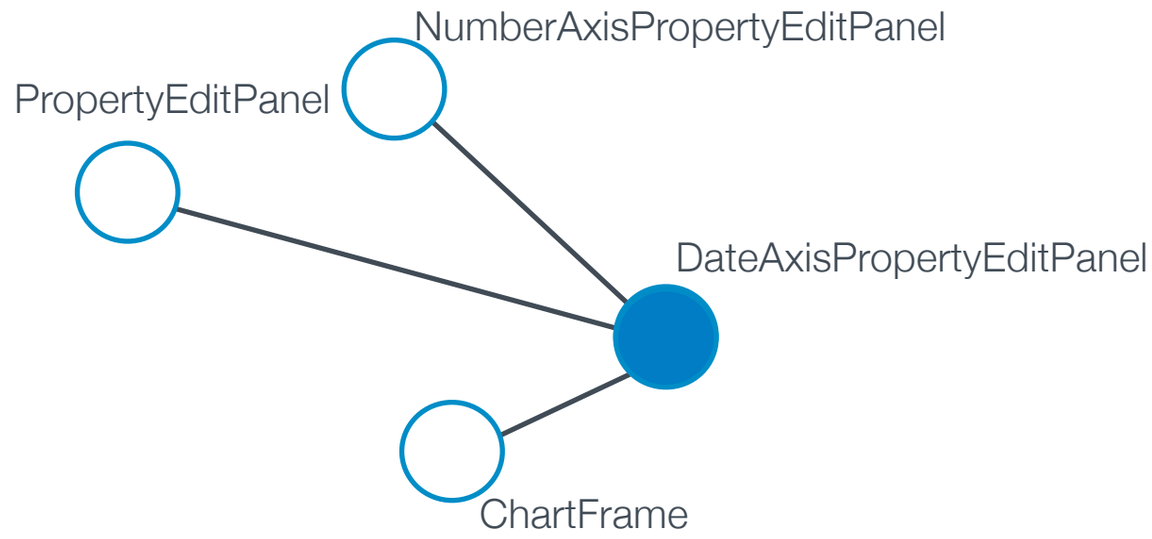
FitAlgorithm

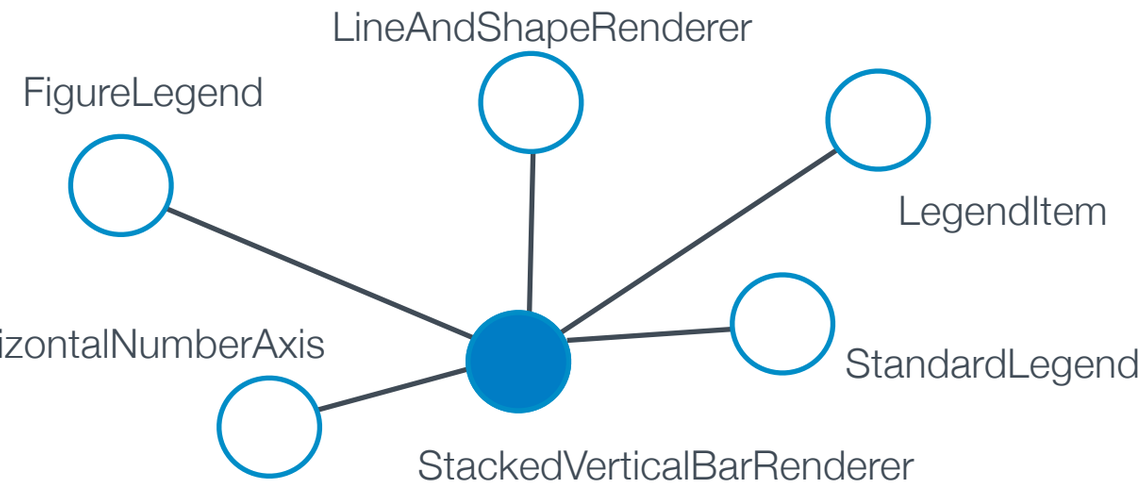
LinearPlotFit

PlotFitAlgorithm

K-MEDOID VARIANT

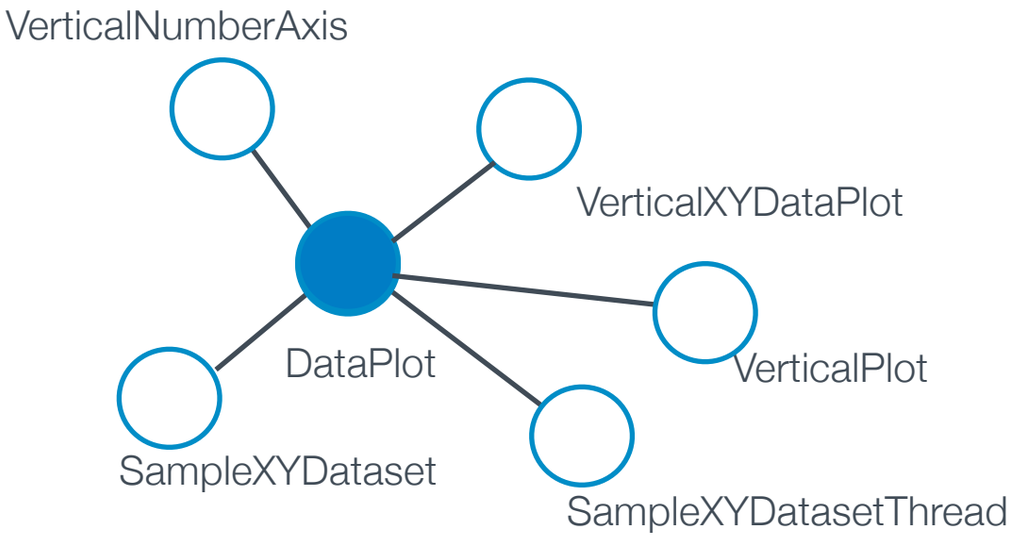


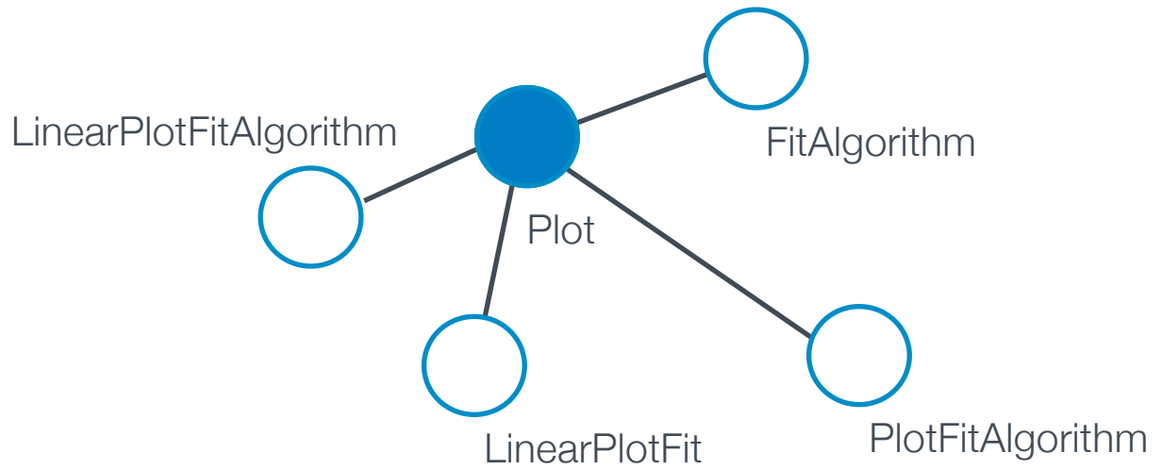




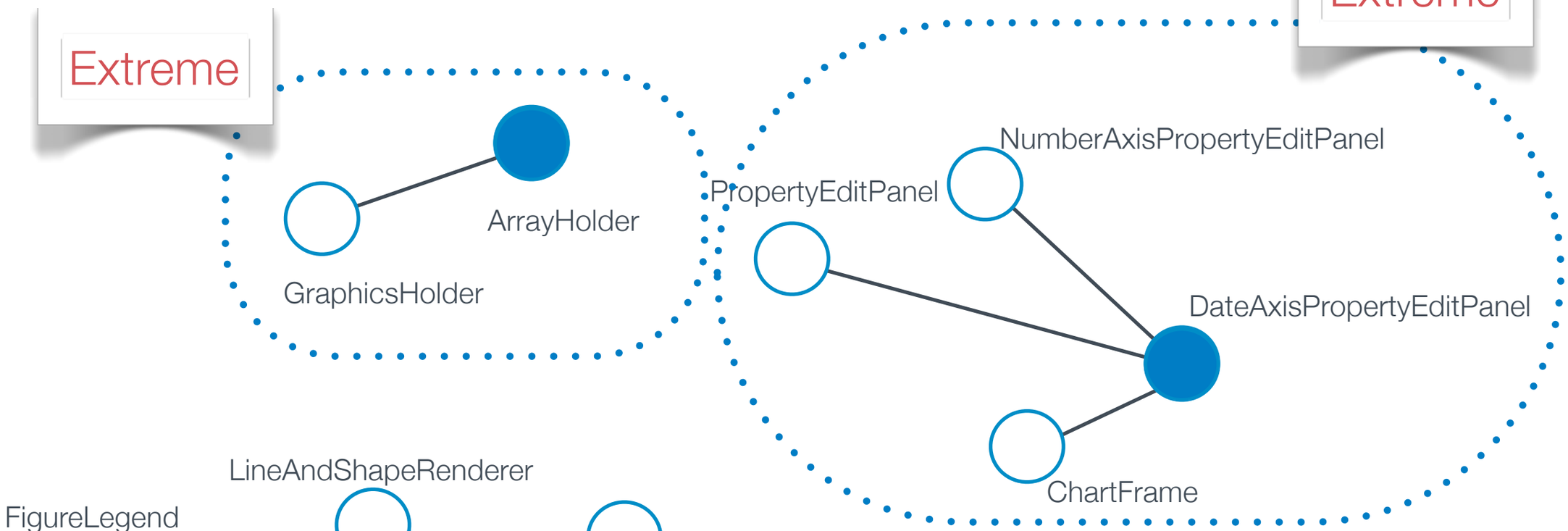
Non-extreme Distribution

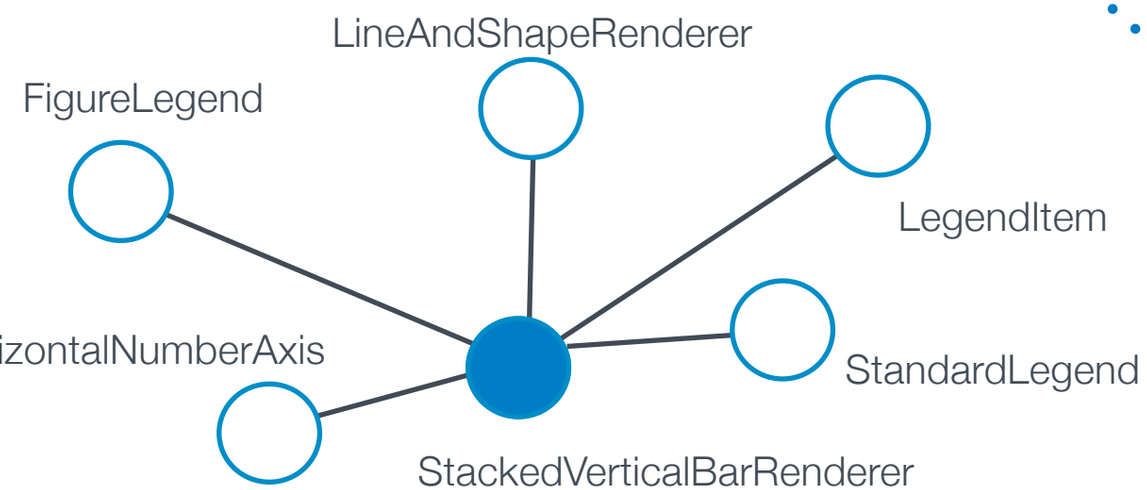
property





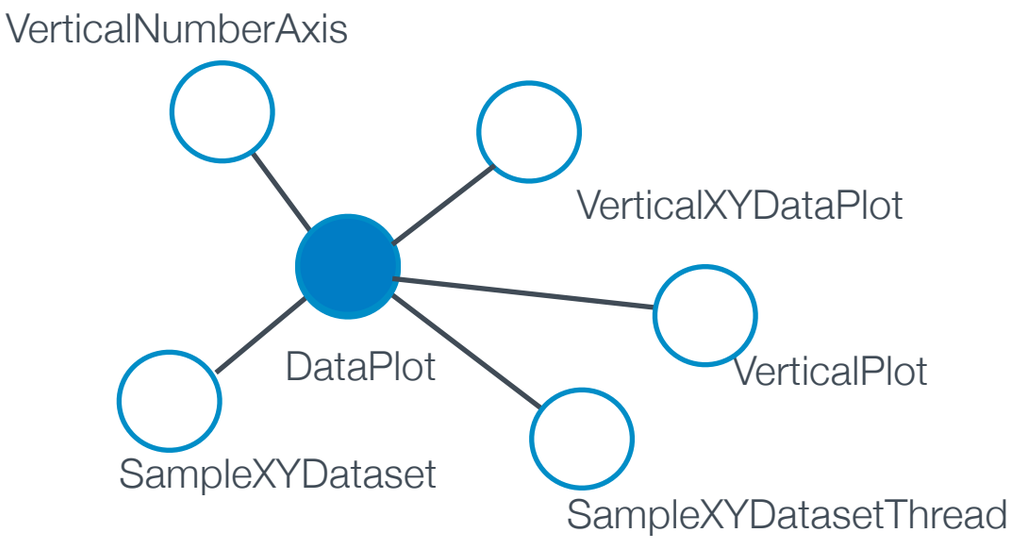
K-MEDOID VARIANT

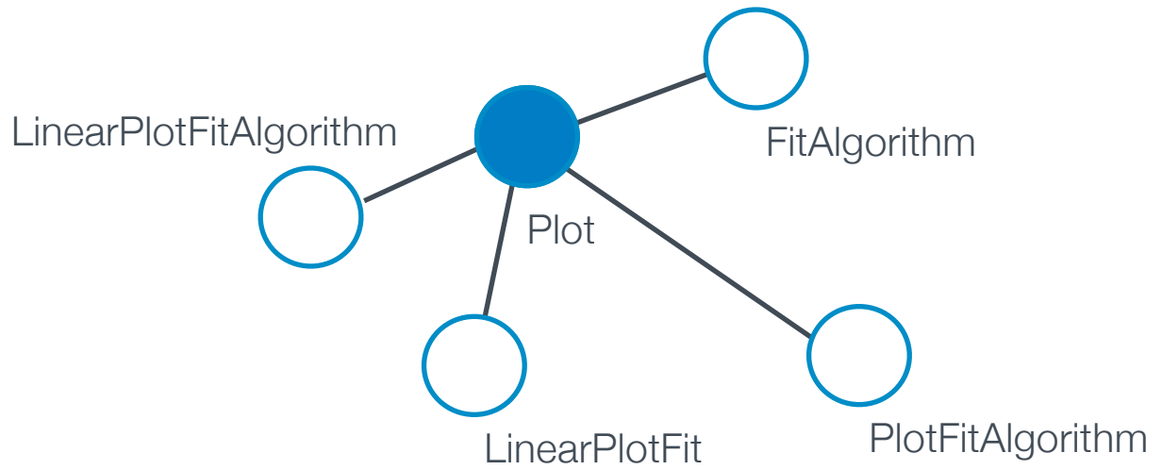




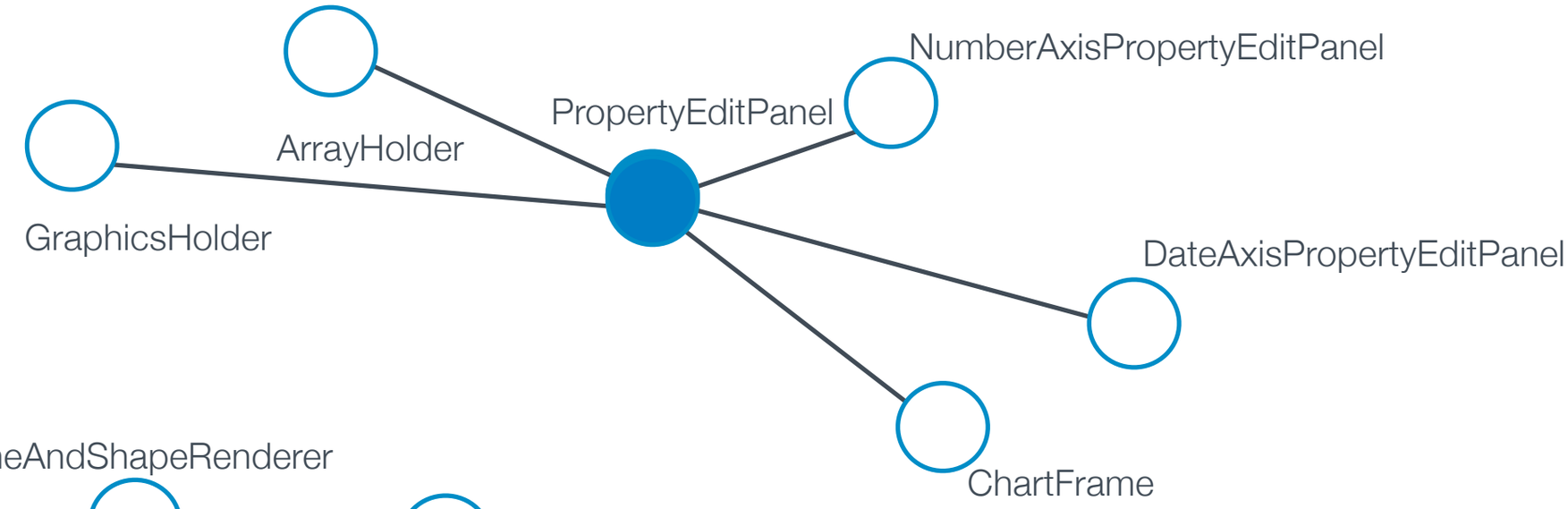
Non-extreme Distribution

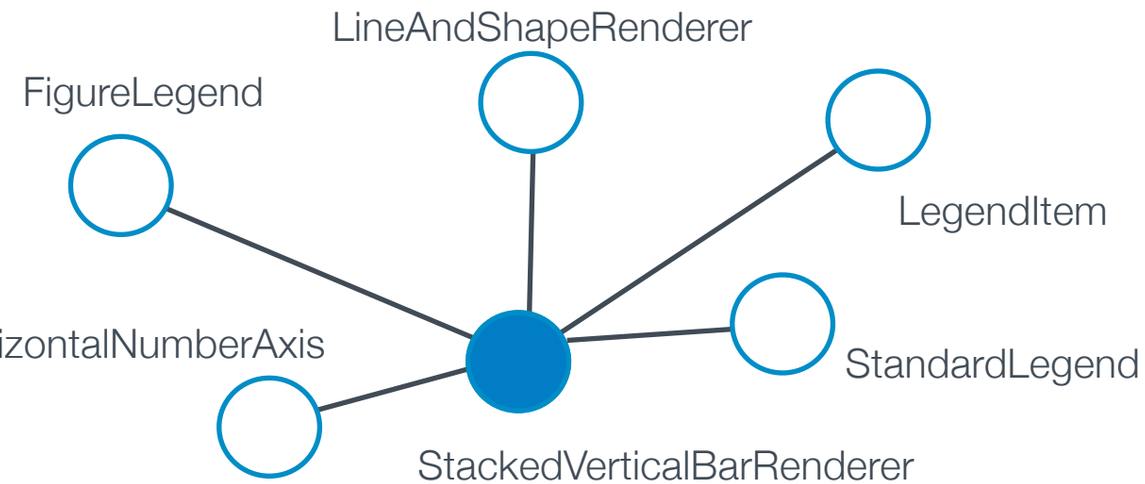
property





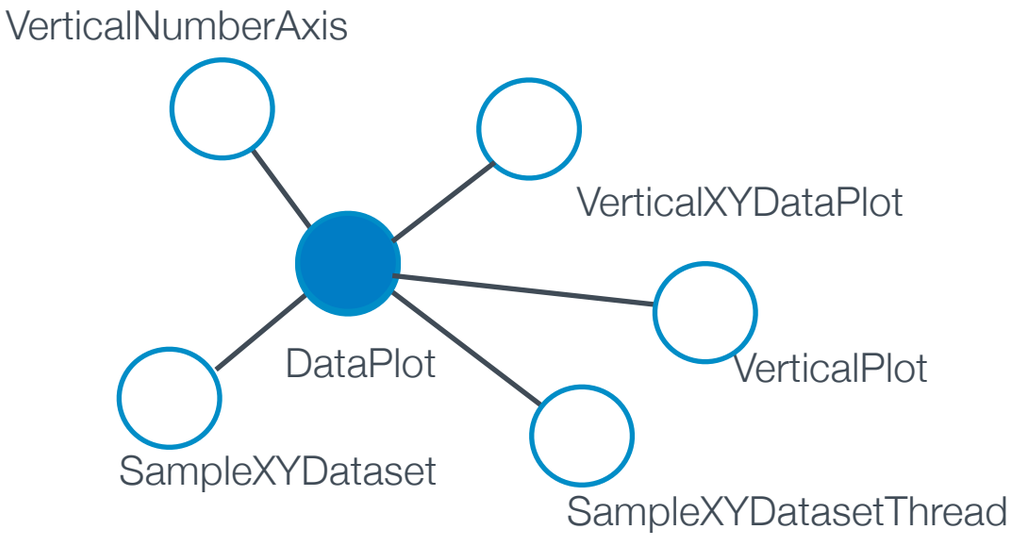
K-MEDOID VARIANT

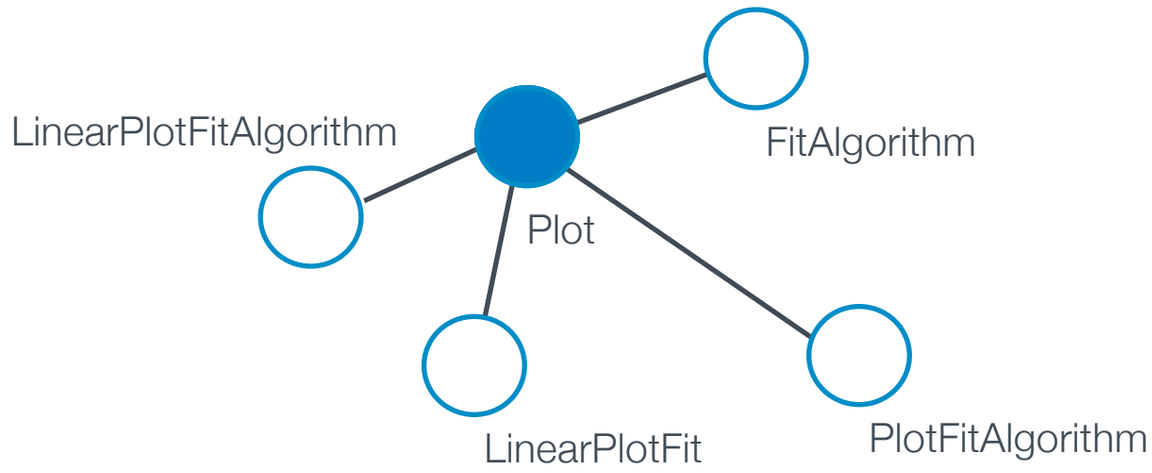




Non-extreme Distribution

property





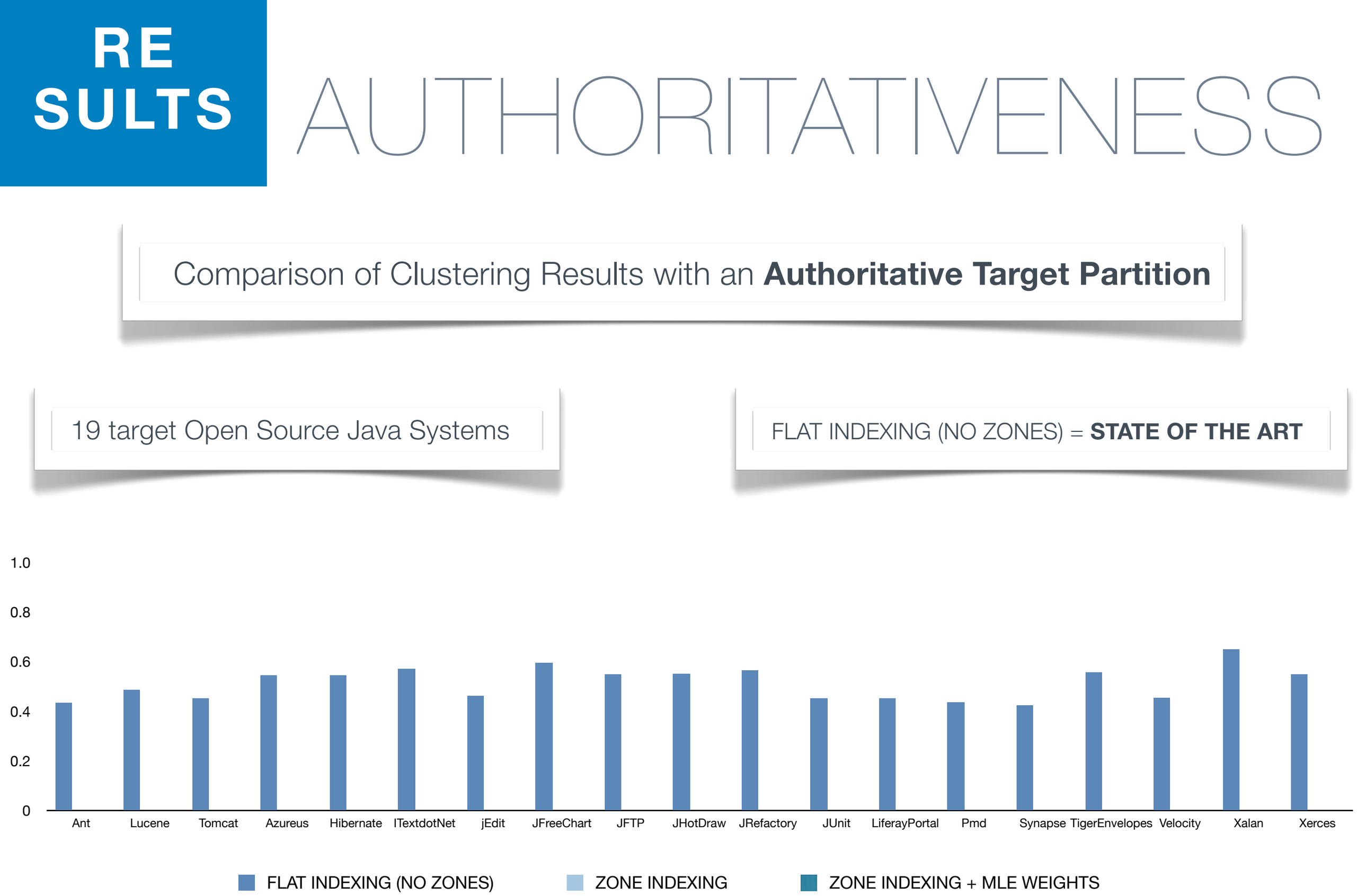
RESULTS

AUTHORITATIVENESS

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

19 target Open Source Java Systems

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



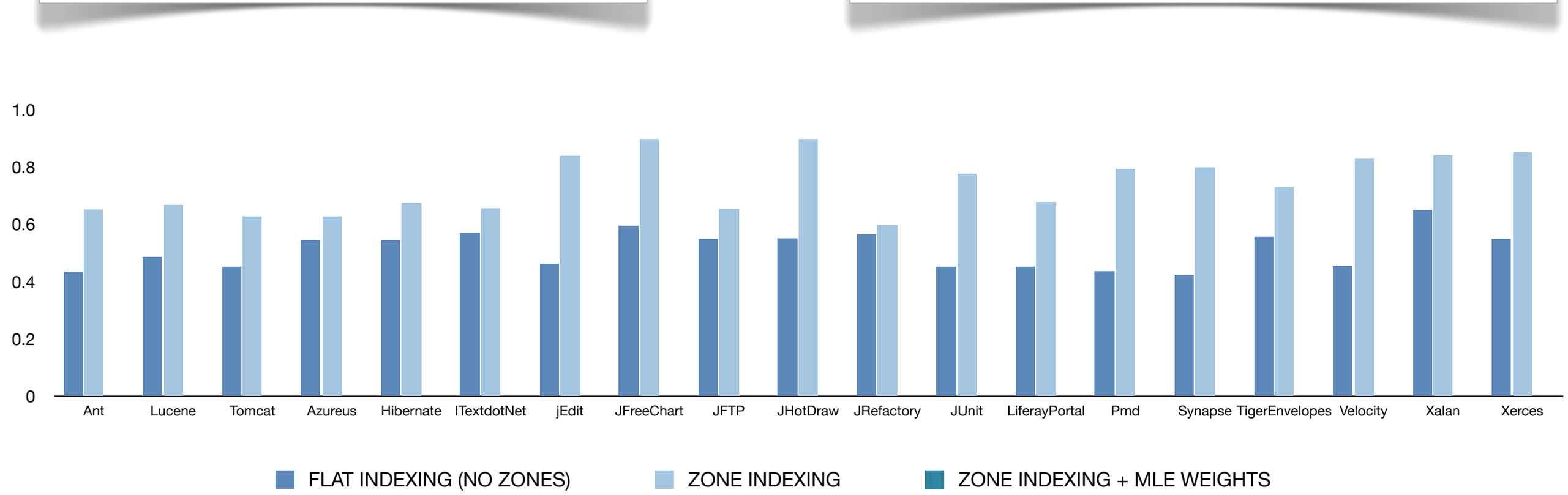
RESULTS

AUTHORITATIVENESS

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

19 target Open Source Java Systems

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



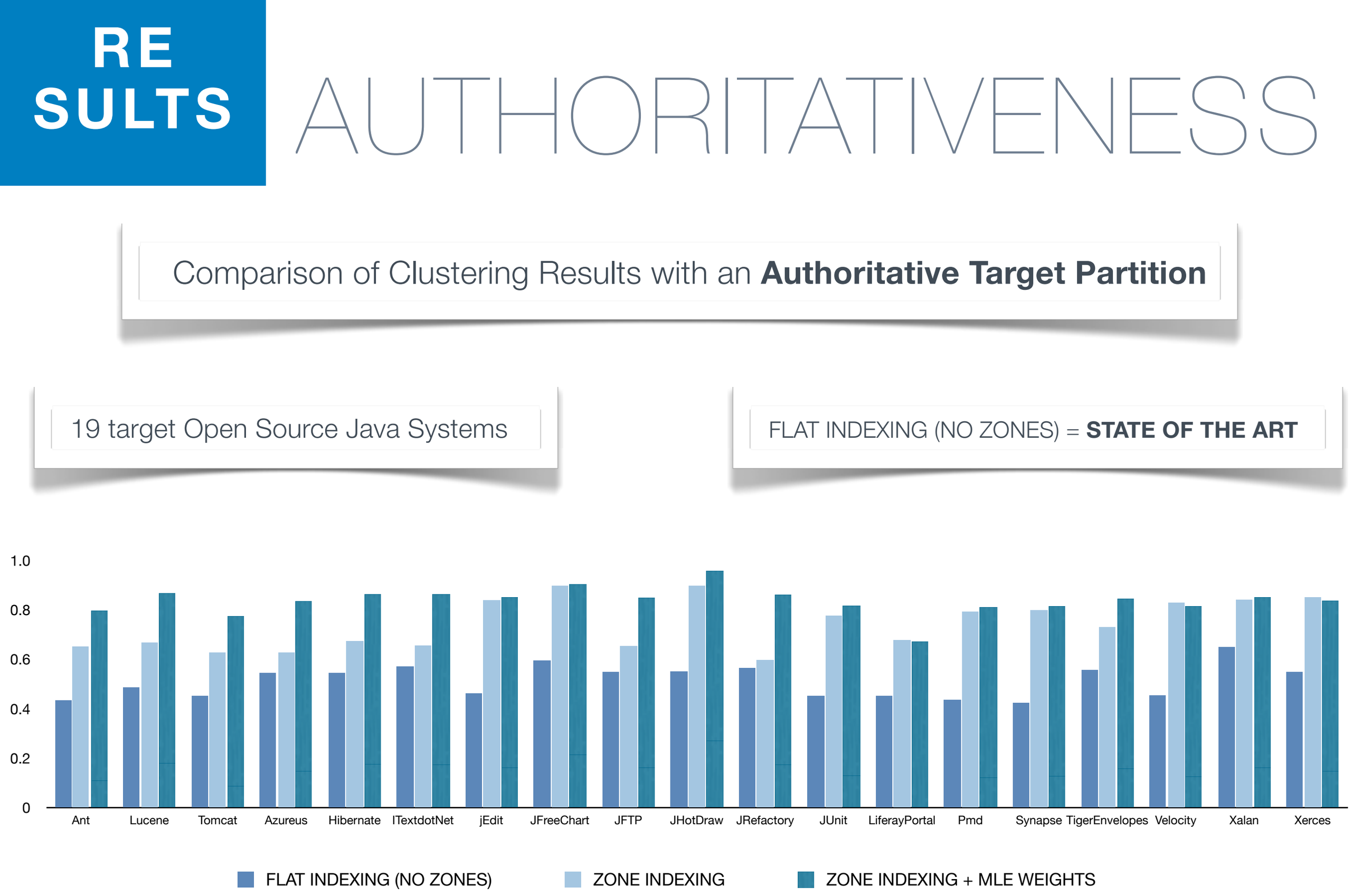
RESULTS

AUTHORITATIVENESS

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

19 target Open Source Java Systems

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



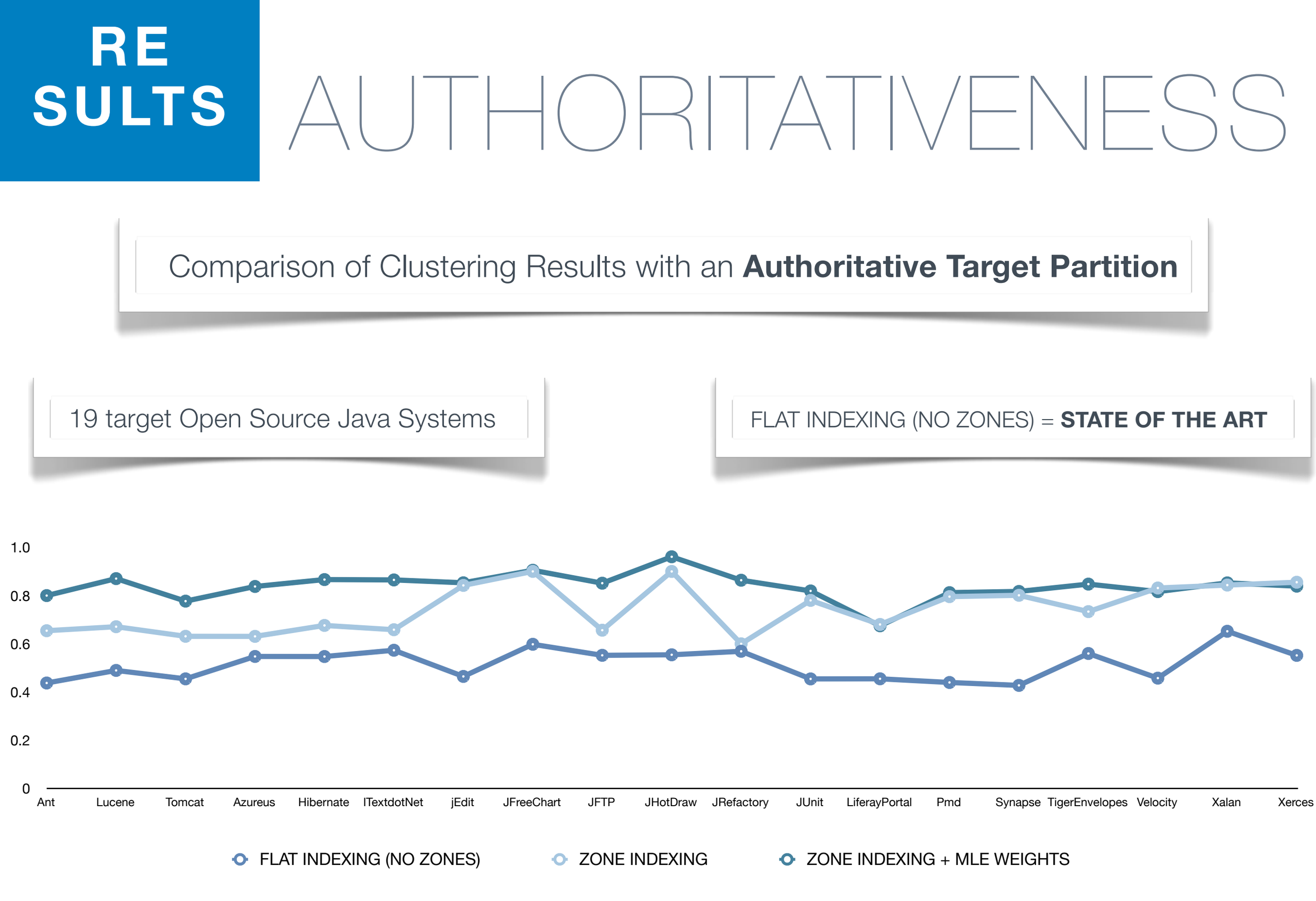
RESULTS

AUTHORITATIVENESS

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

19 target Open Source Java Systems

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



ISSUE
#2

SOURCE CODE NORMALIZATION

SOURCE CODE

LEXICAL INFORMATION

```
/**
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 */
public class NullHandle extends LocatorHandle {

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

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

    /**
     * Draws the NullHandle's framed rectangle.
     */
    public void draw(Graphics g) {
        Rectangle displayBox= displayBox();
        g.setColor(Color.black);
        g.drawDORRect(0);
    }
}
```

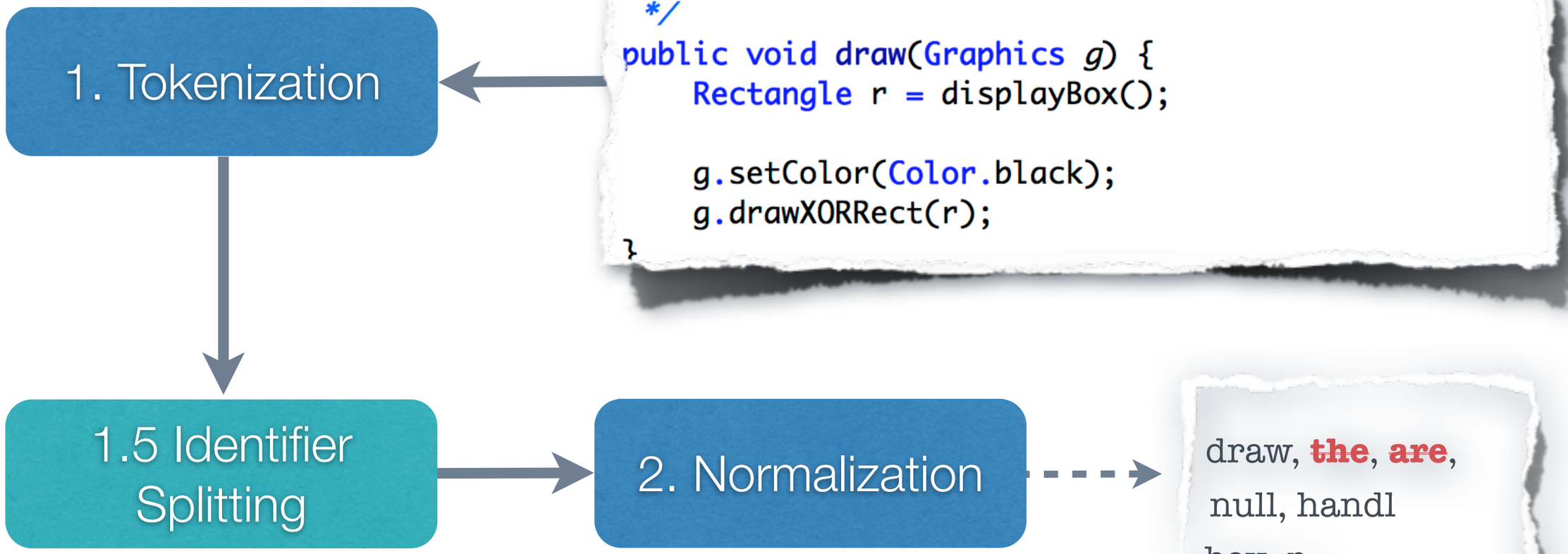
Implicit assumption: The “same” words are used whenever a particular concept occurs

SOURCE CODE

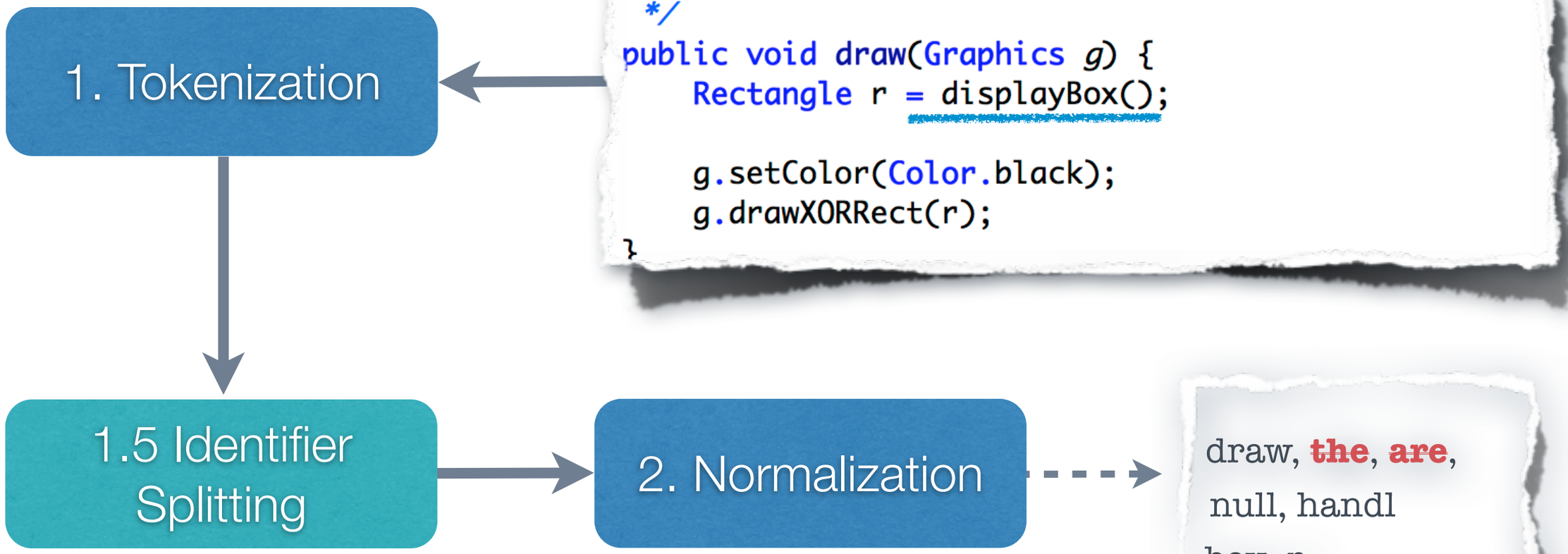
LEXICAL INFORMATION

```
/**
 * A handle that doesn't change the owned figure. Its only purpose is
 * to show feedback that a figure is selected.
 * <hr>
 */
public class NullHandle extends LocatorHandle {

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

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

    /**
     * Draws the NullHandle's framed rectangle as a
     */
    public void draw(Graphics g) {
        Rectangle displayBox();
        g.setColor(Color.black);
        g.drawXORRect(;
    }
}
```

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



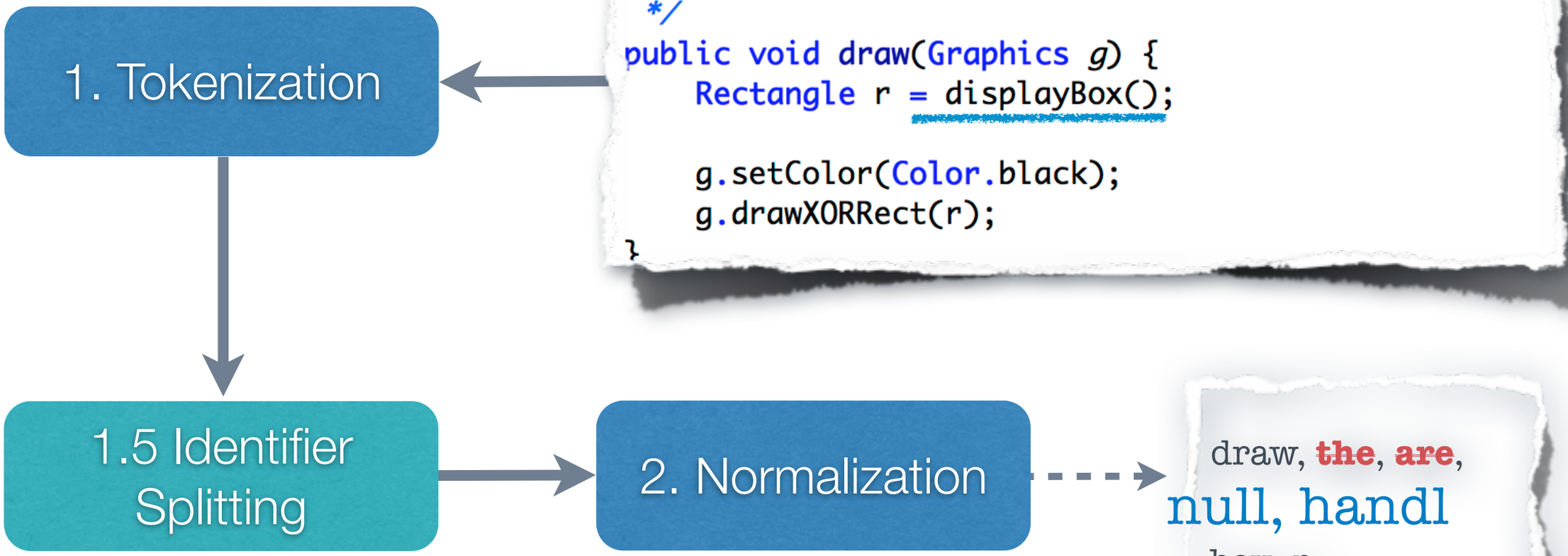
- *Splitting algorithms* based on *naming conventions*
  - **camelCase** Splitter: `r'(?<=!^)([A-Z][a-z]+)'`
    - NullHandle ==> Null | Handle
    - displayBox ==> display | Box




```
/**
 * 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*
  - **camelCase** Splitter: `r'(?<=!^)([A-Z][a-z]+)'`
    - NullHandle ==> Null | Handle
    - displayBox ==> display | Box

STATE OF THE ART TOOLS




```
/**
 * 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*
  - **camelCase** Splitter: `r'(?<=!^)([A-Z][a-z]+)'`
    - NullHandle ==> Null | Handle

STATE OF THE ART TOOLS


```
/**
 * 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, r,
rectangl, g,
graphic,
color,
display, box
....

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

IDENTIFIERS SPLITTING

```
/**
 * 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]+)'`

IDENTIFIERS SPLITTING

```
/**
 * 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`

IDENTIFIERS SPLITTING

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`

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

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 Code Normalization

2. Normalization

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

- 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)

CODE NORMALIZATION

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 Code Normalization

2. Normalization

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

- 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)

CONTRI BUTION

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

CONTRI BUTION

SOURCE CODE NORMALIZATION

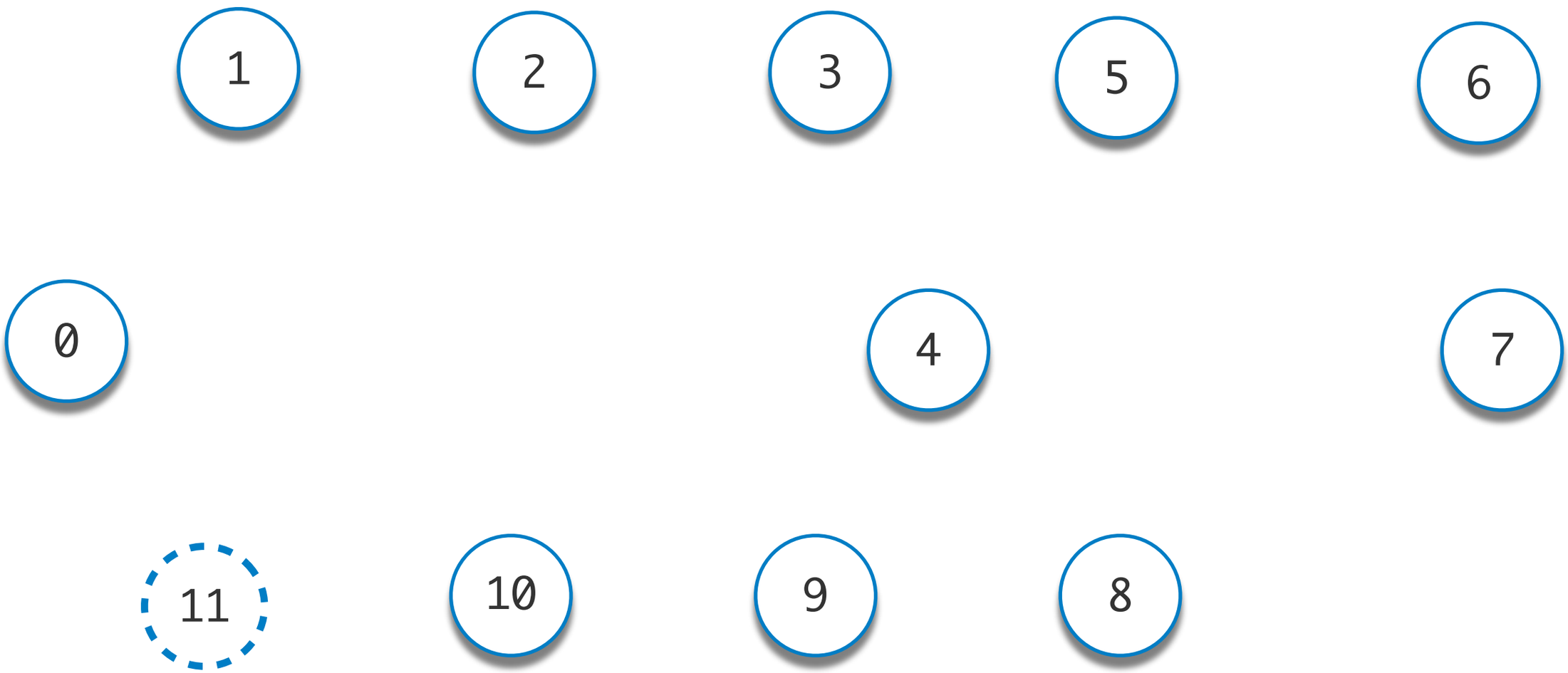
- **LINSEN:** novel technique for code normalization
  - Able to **both** *Split Identifiers* and *Expand* possible occurring abbreviations
- Based on an efficient String Matching technique: *Baeza-Yates&Perlberg Algorithm (BYP)*
  - Exploit different **Sources of Information** to find the matching words

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

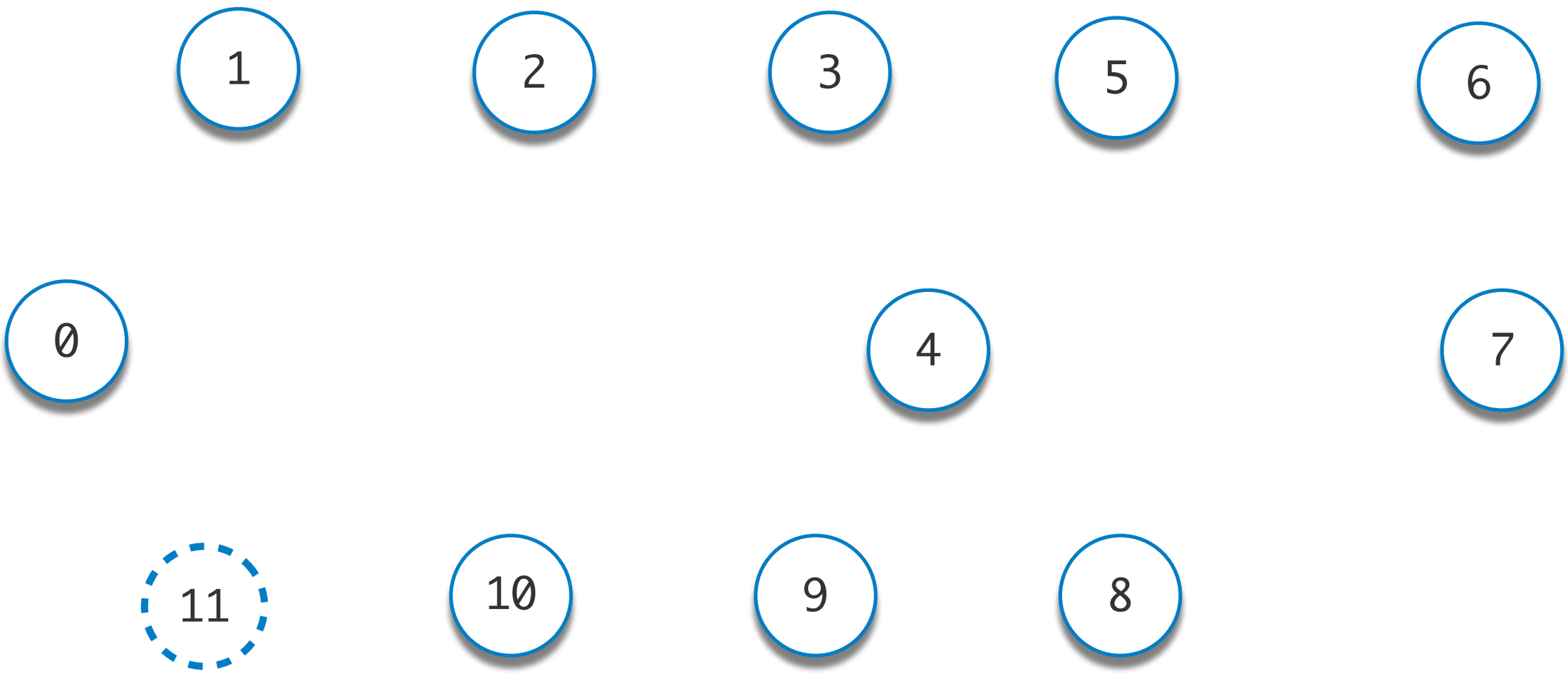


- **NODES** correspond to characters of the current identifier

SKETCH OF THE ALGORITHM

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

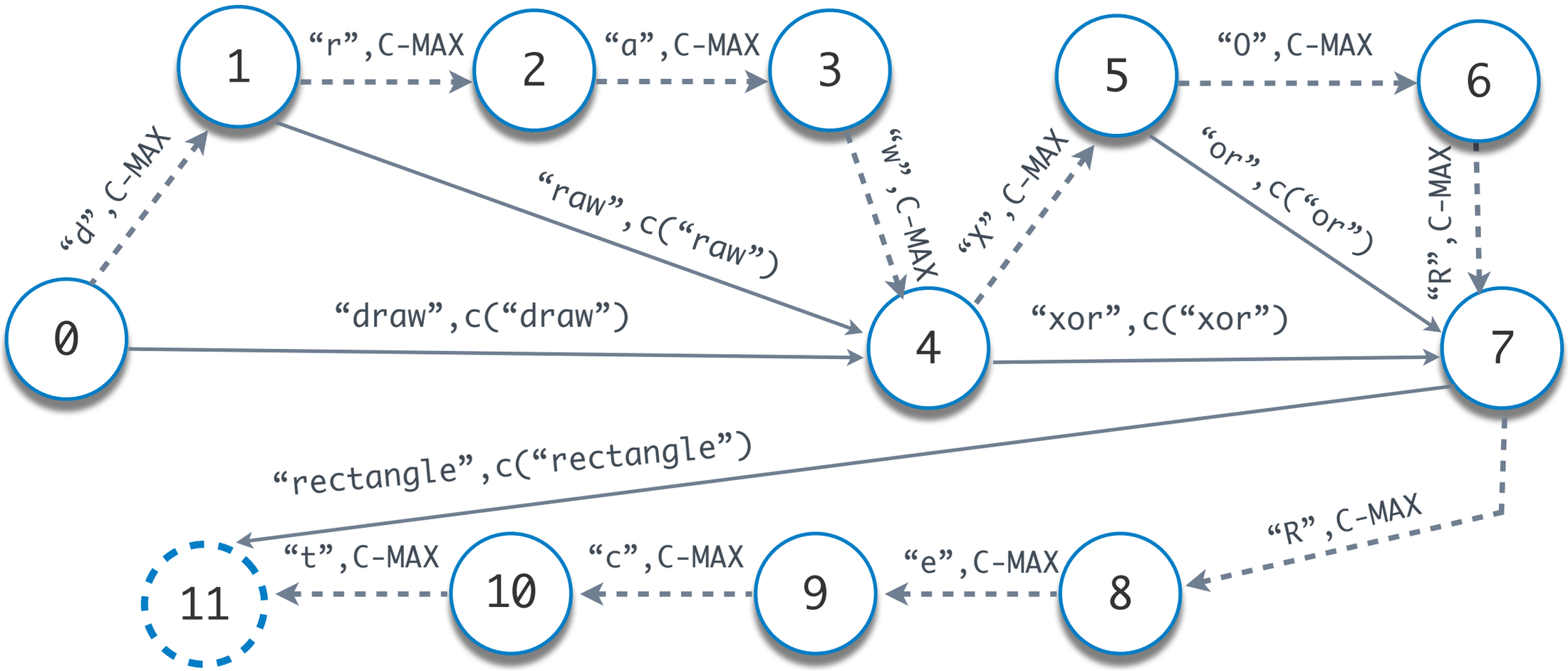


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*

SKETCH OF THE ALGORITHM

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

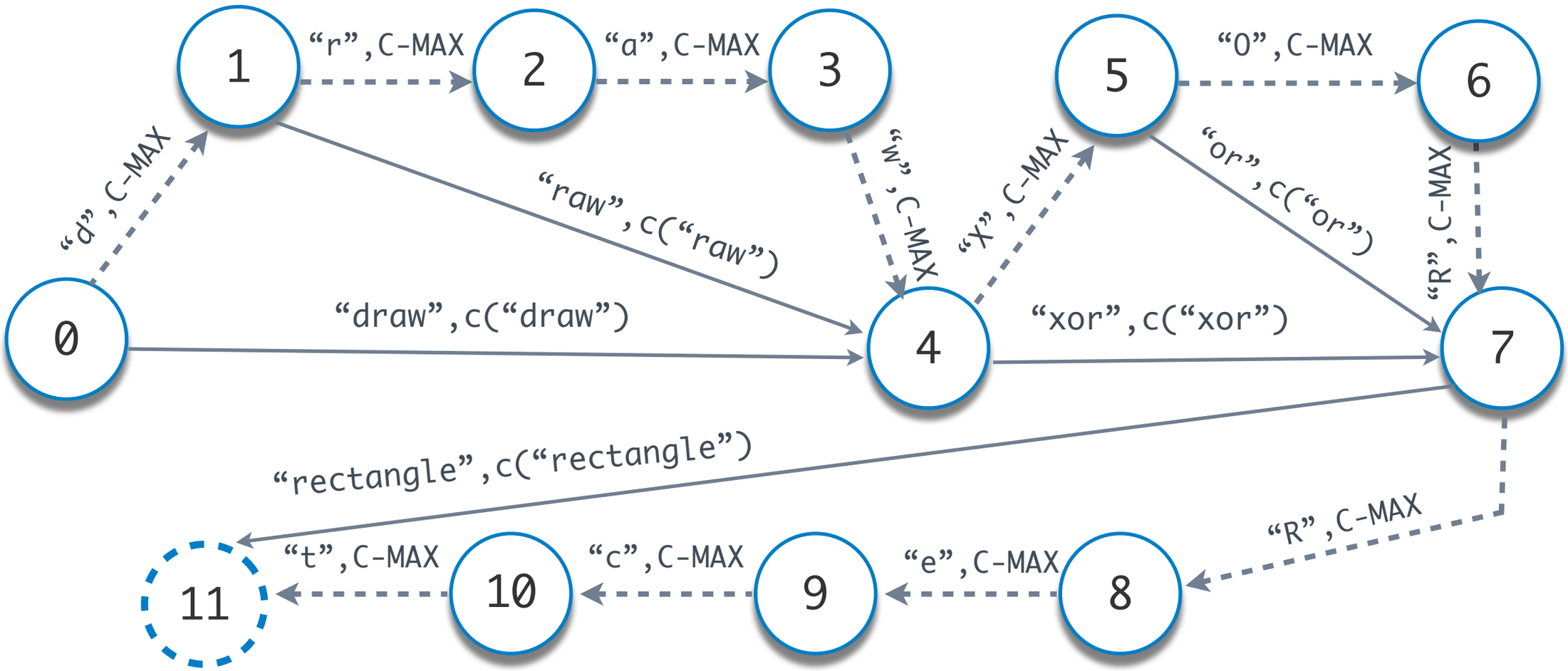


- **NODES** correspond to characters of the current identifier
- **ARCS** corresponds to matchings between *identifier substrings* and *dictionary words*
  - Padding Arcs to ensure the Graph always connected

SKETCH OF THE ALGORITHM

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier

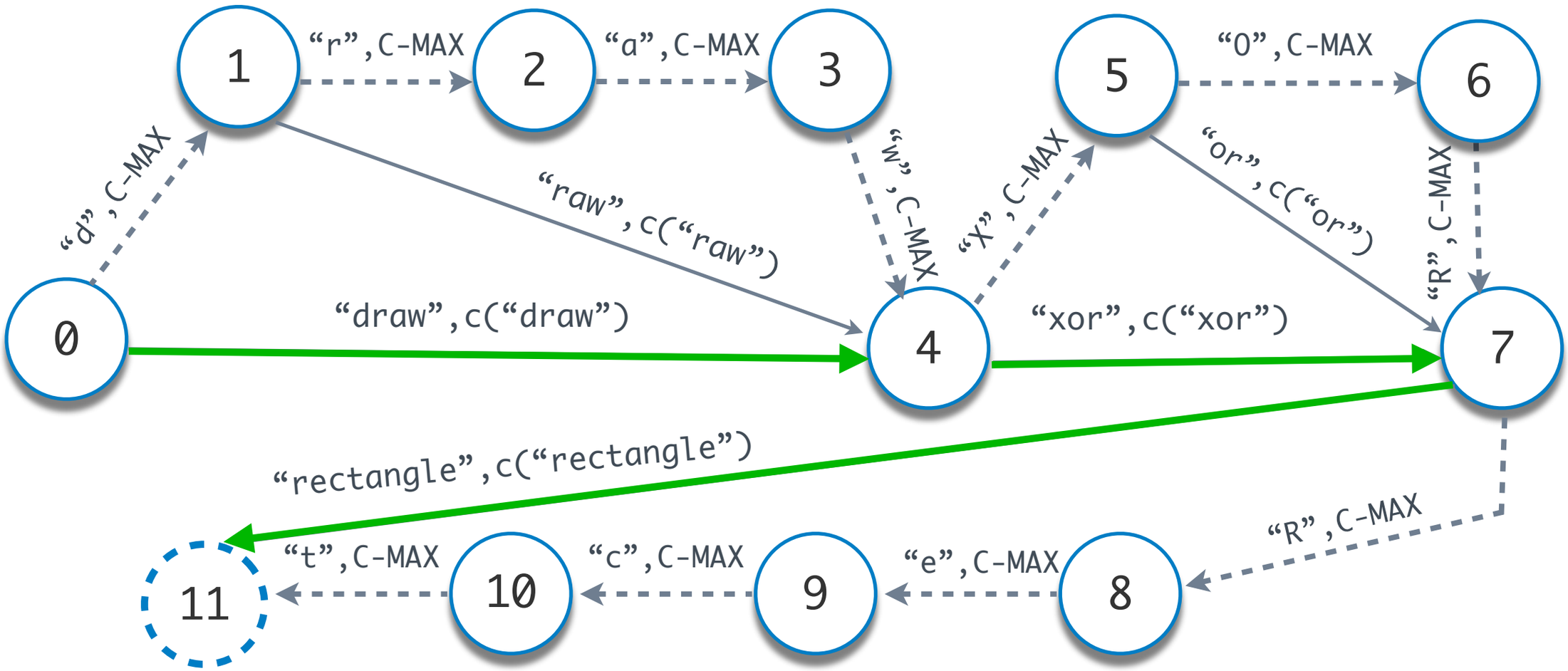


- Every Arc is **Labelled** with the corresponding dictionary word
- **Weights** represent the "cost" of each matching
- *Cost function* `[c("word")]` favors *longest words* and *domain-related* information

SKETCH OF THE ALGORITHM

Model: *Weighted* Directed Graph

Example: `drawXORRect` identifier



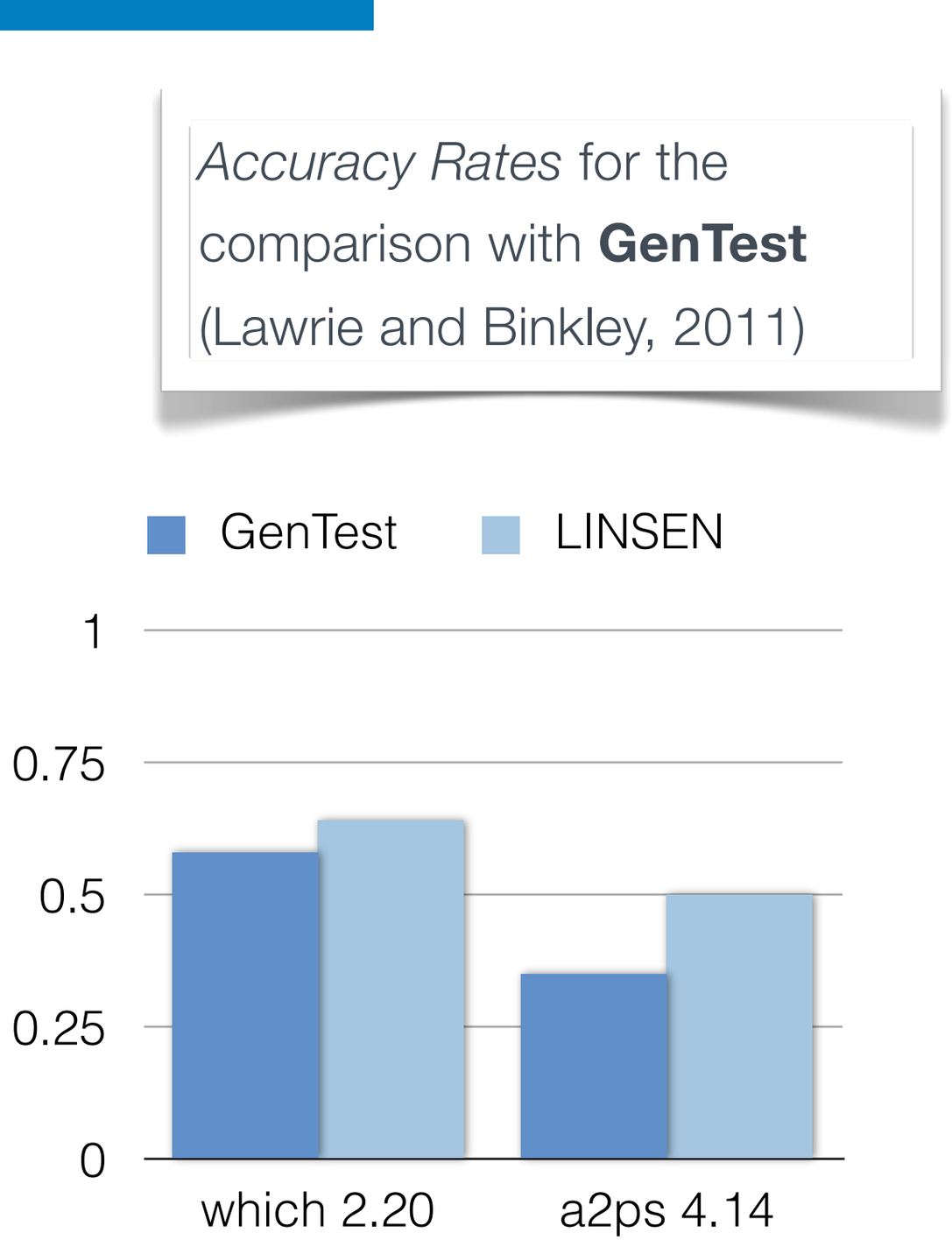
- The final **Mapping Solution** corresponds to the **sequence of labels** in the path with the **minimum cost** (*Djikstra Algorithm*)

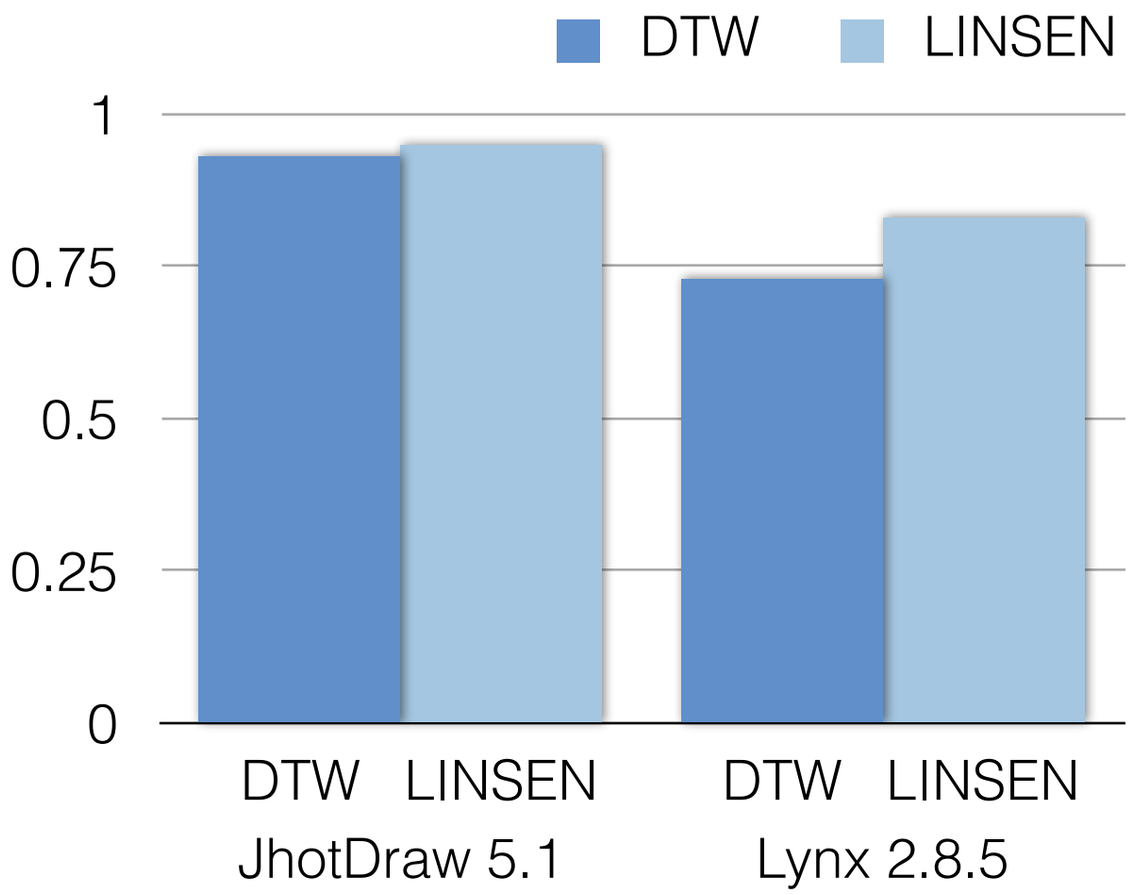
RESULTS # 1

SPLITTING

DTW $O(n^3)$

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





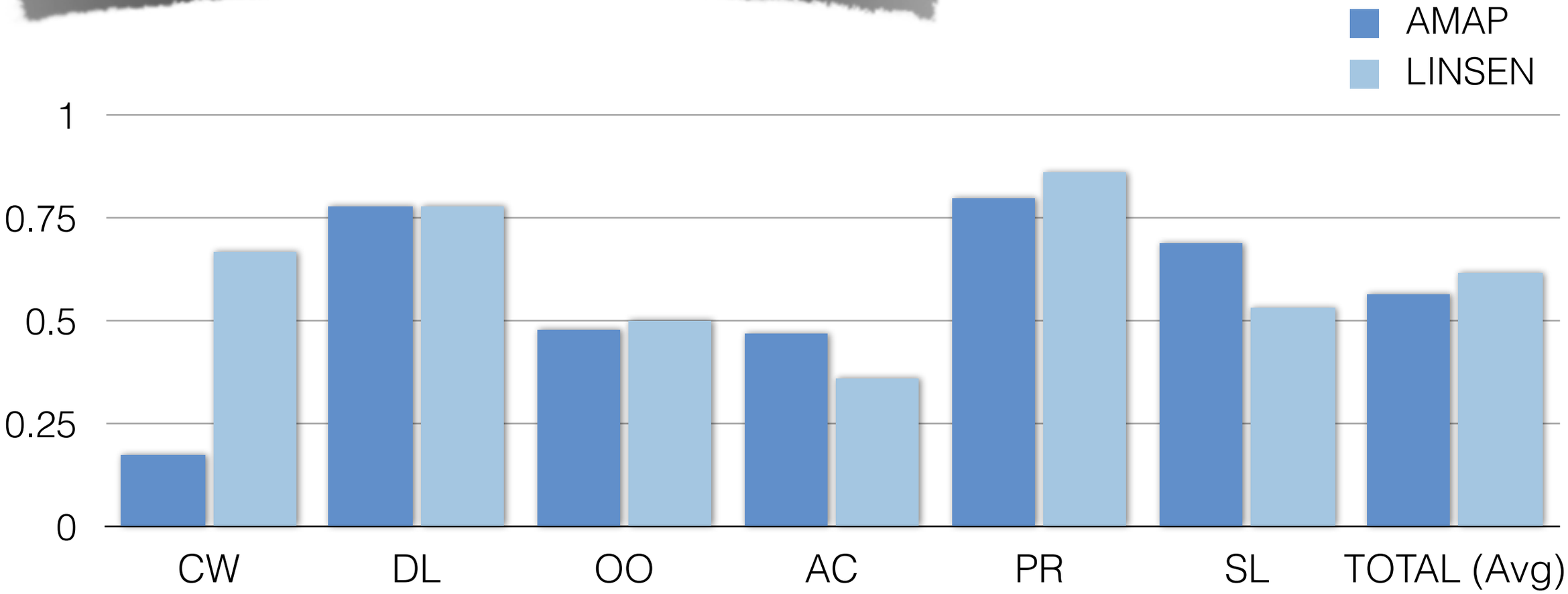
Accuracy Rates for the comparison with **DTW** (Madani et. al 2010)

RESULTS # 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)

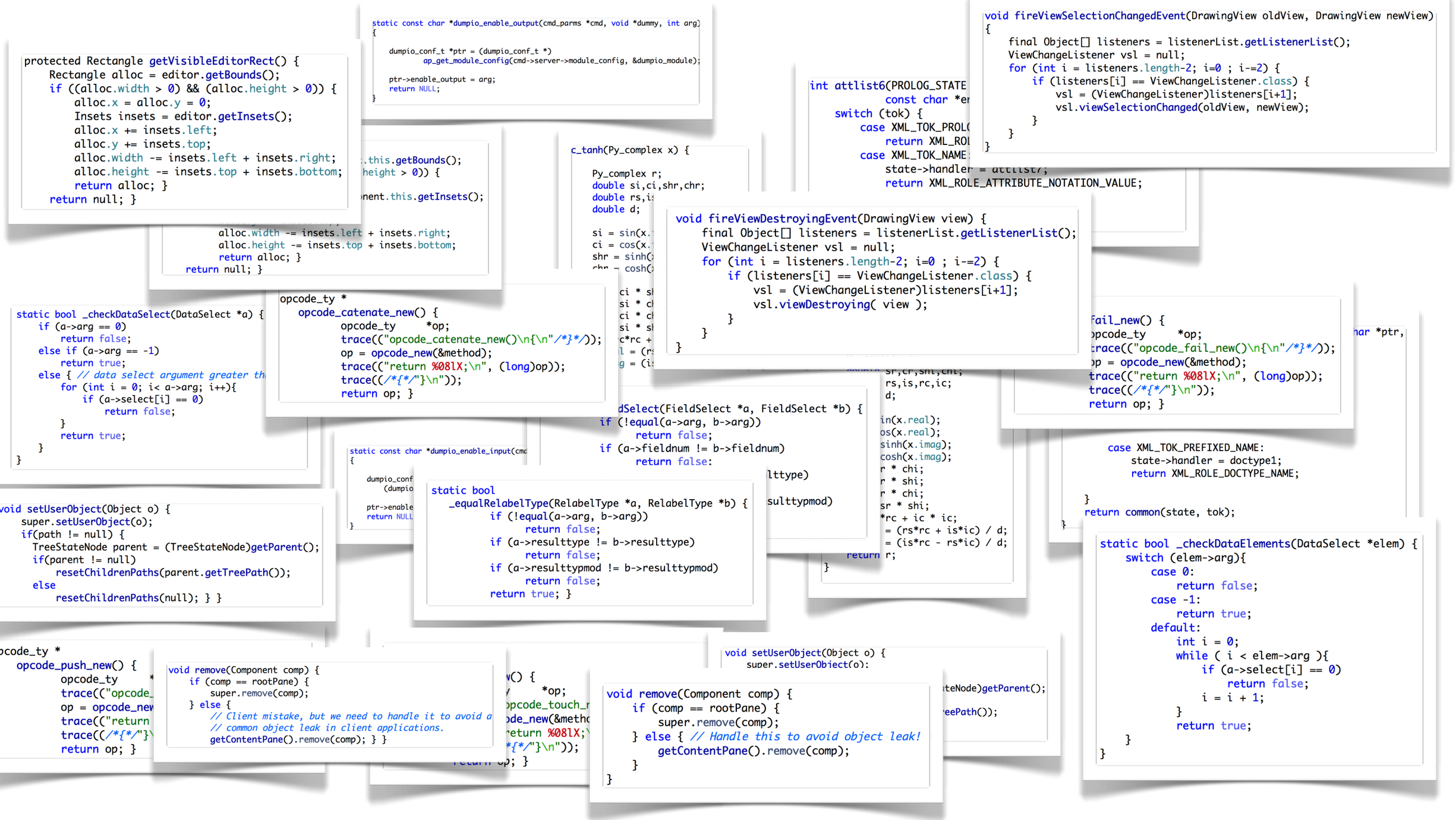


ISSUE
#3

CLONE DETECTION

PROBLEM STATEMENT

CLONE DETECTION



PROBLEM STATEMENT

CLONE DETECTION

```
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; }
```

```
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; }
```

Clones **Textual** Similarity

PROBLEM STATEMENT

CLONE DETECTION

```
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;
    }
}
```

```
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

PROBLEM STATEMENT

CLONE DETECTION

```
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

Duploc

SDD

NiCAD

Dude

Simian

CLICS

Scorpio

Duplix

PMD

Dup

CPD

Duplix

Gemini

CCFinder

Shinobi

iClones

Clone Detective

Clone Digger

ConQAT

JCCD

Deckard

CloneDr

SimScan

KClone

STATE OF THE ART TOOLS

Duploc

SDD

NiCAD

Dude

Simian

CLICS

Dup

CPD

Duplix

Gemini

Text Based Tools:

Text is compared line by line

iClones

Clone Detective

Clone Digger

ConQAT

JCCD

Deckard

Scorpio

Duplix

CloneDr

SimScan

PMD

KClone

STATE OF THE ART TOOLS

Duploc

SDD

NiCAD

Dude

Simian

CLICS

Dup

CPD

Duplix

Gemini

CCFinder

Shinobi

iClones

Clone Detective

Clone Digger

Scorpio

Duplix

PMD

KClone

Token Based Tools:

Token sequences are compared to sequences

STATE OF THE ART TOOLS

Duploc

Dup

CPD

SDD

NiCAD

Dude

Simian

CLICS

iClones

Clone Detective

Syntax Based Tools:

Syntax subtrees are compared to each other

Clone Digger

ConQAT

JCCD

Deckard

CloneDr

SimScan

KClone

Scorpio

Duplix

PMD

STATE OF THE ART TOOLS

Duploc

SDD

NiCAD

Dude

Simian

CLICS

Dup

CPD

Duplix

Gemini

CCFinder

Shinobi

iClones

Clone Detective

Clone Digger

ConQAT

ICCD

KClone

Scorpio

Duplix

PMD

Graph Based Tools:

(sub) graphs are compared to each other

CONTRI BUTION

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*

CONTRI BUTION

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*

CONTRI BUTION

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*

CONTRI BUTION

CLONE DETECTION

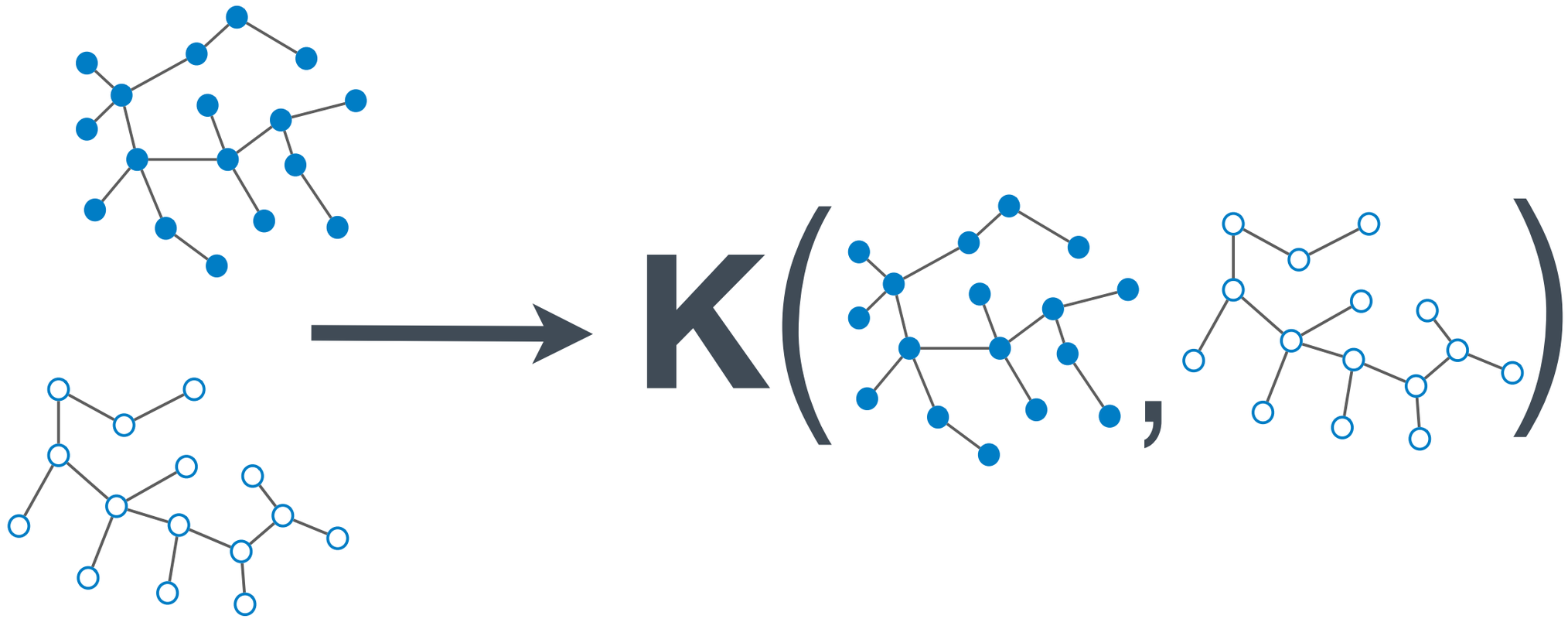
- Combining different sources of information to improve the effectiveness of the detection process
  - Structural and Lexical Information
- Application of **Kernel Methods** to Source Code **Structures**
- Typically applied in other field such as NLP or Bioinformatics

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*

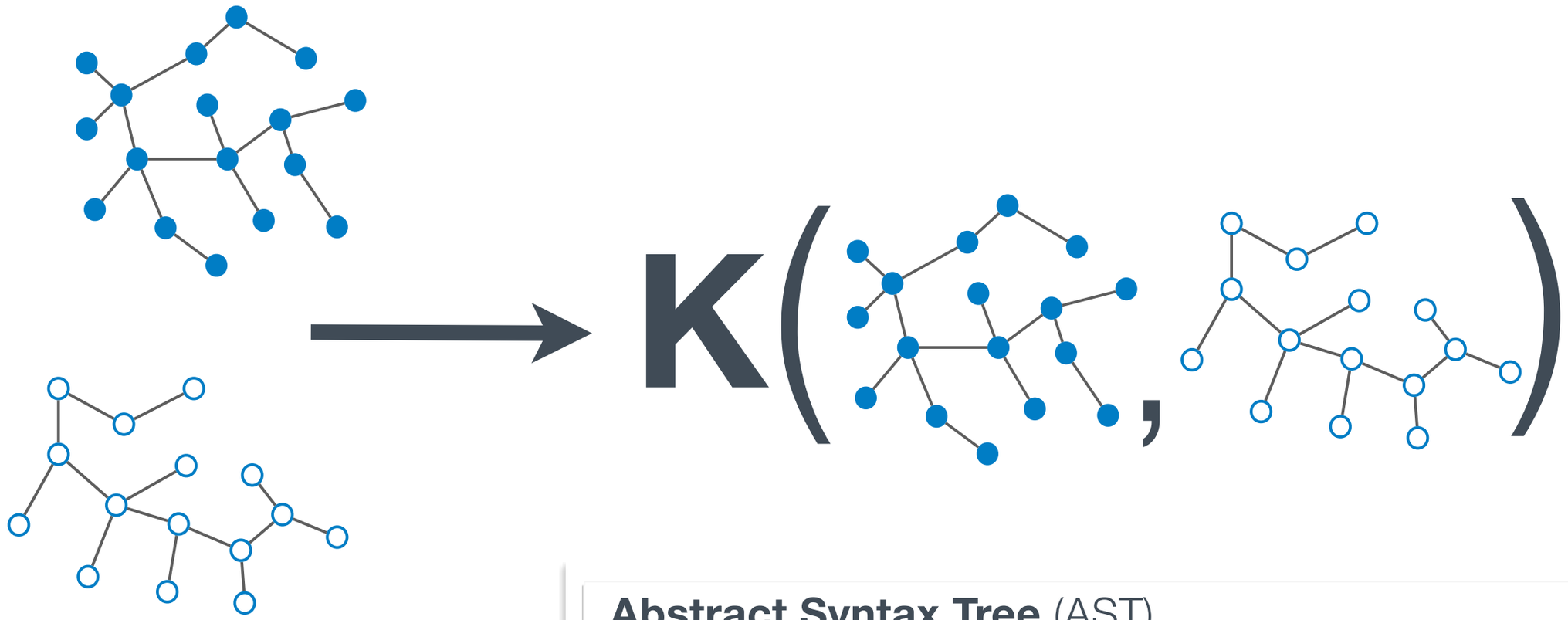
KERNELS FOR STRUCTURES

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



KERNELS FOR STRUCTURES

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



Abstract Syntax Tree (AST)

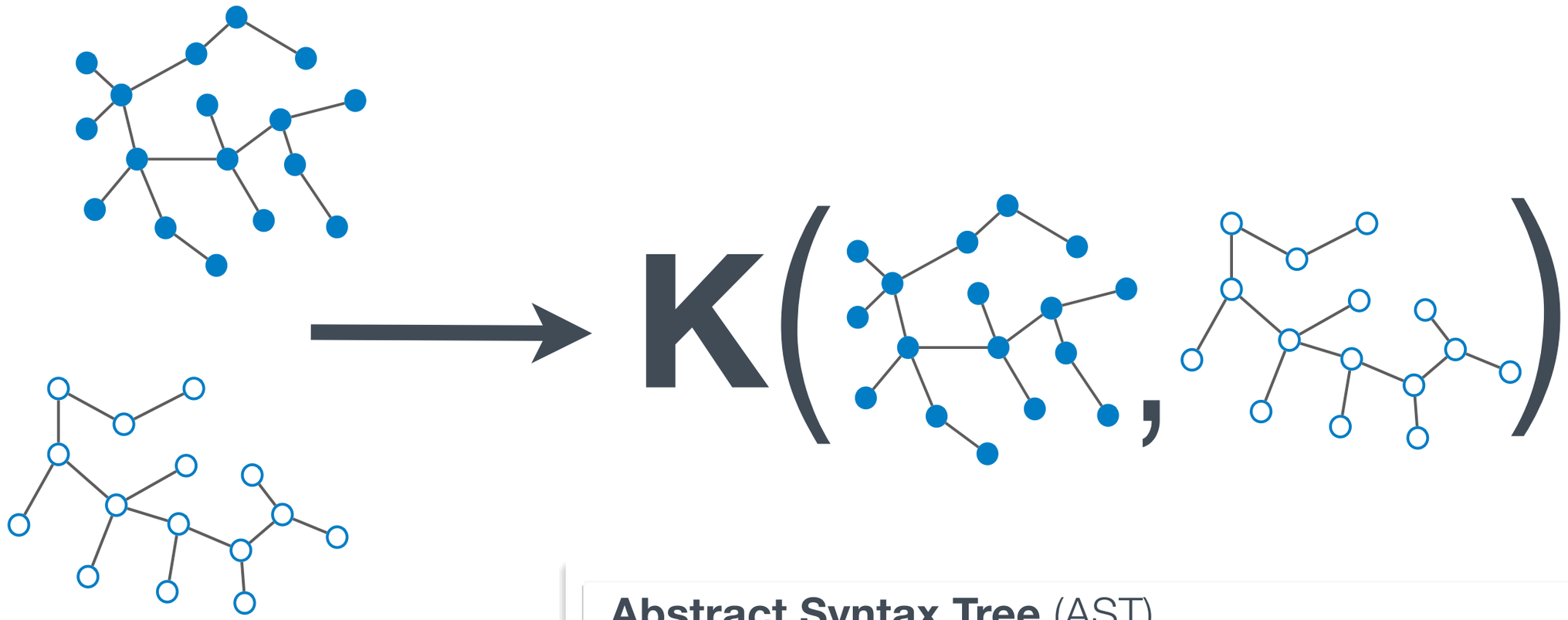
Tree structure representing the syntactic structure of the different instructions of a program (function)

Program Dependencies Graph (PDG)

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

KERNELS FOR STRUCTURES

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

KERNEL FOR CLONES

CODE

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

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

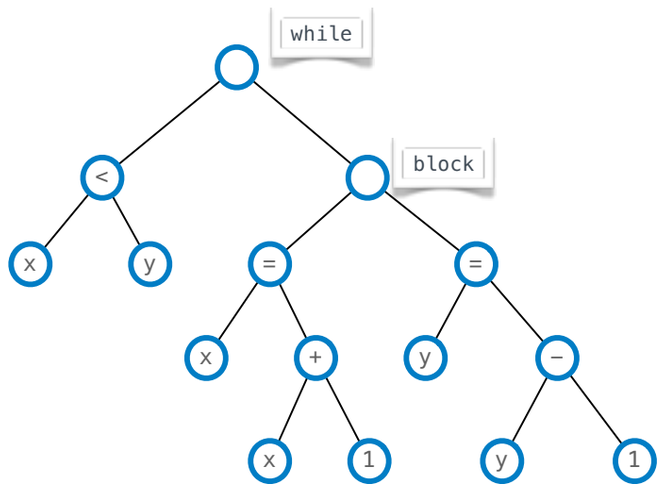
R CLONES

KERNEL FO

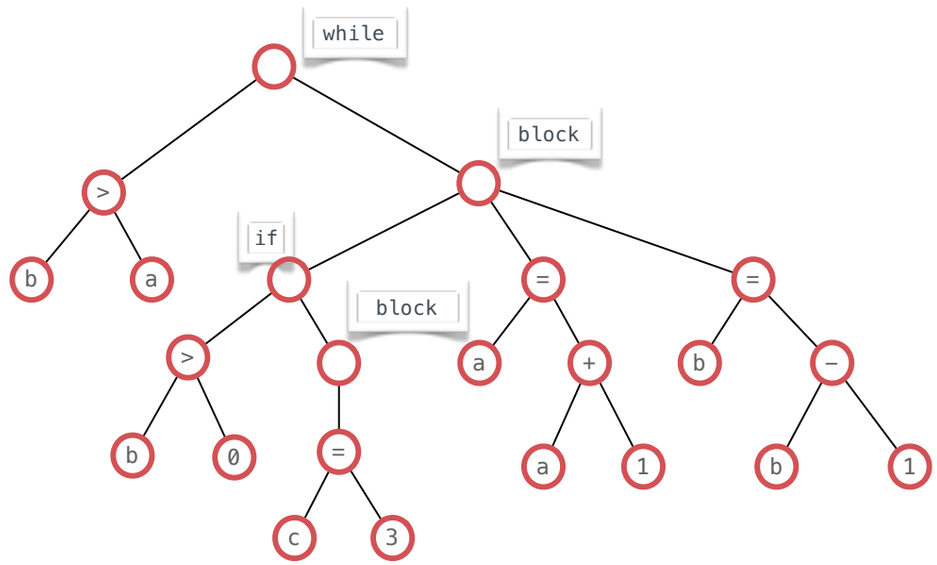
CODE

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

AST




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



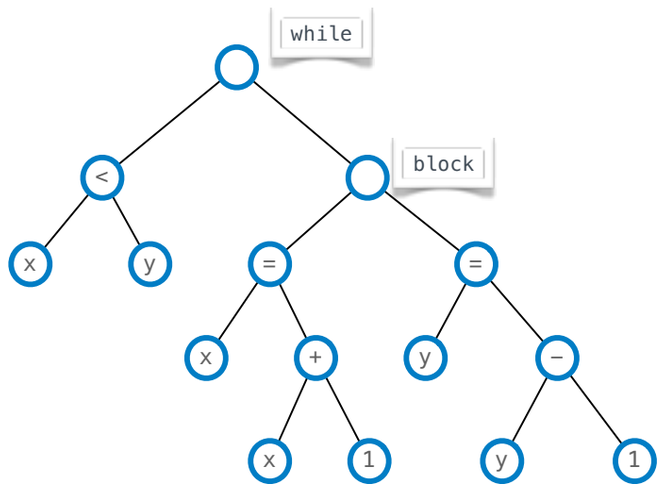
KERNEL FOR CLONES

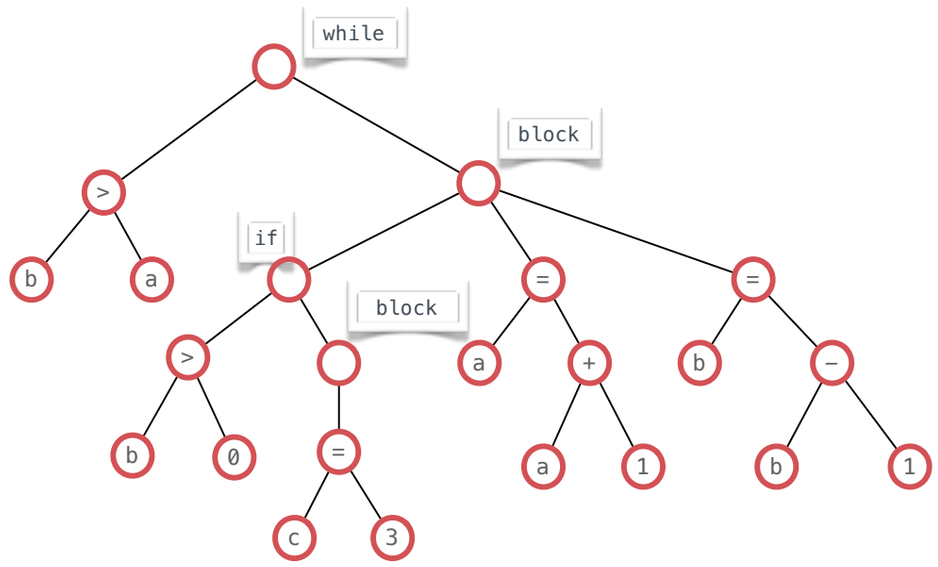
CODE

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

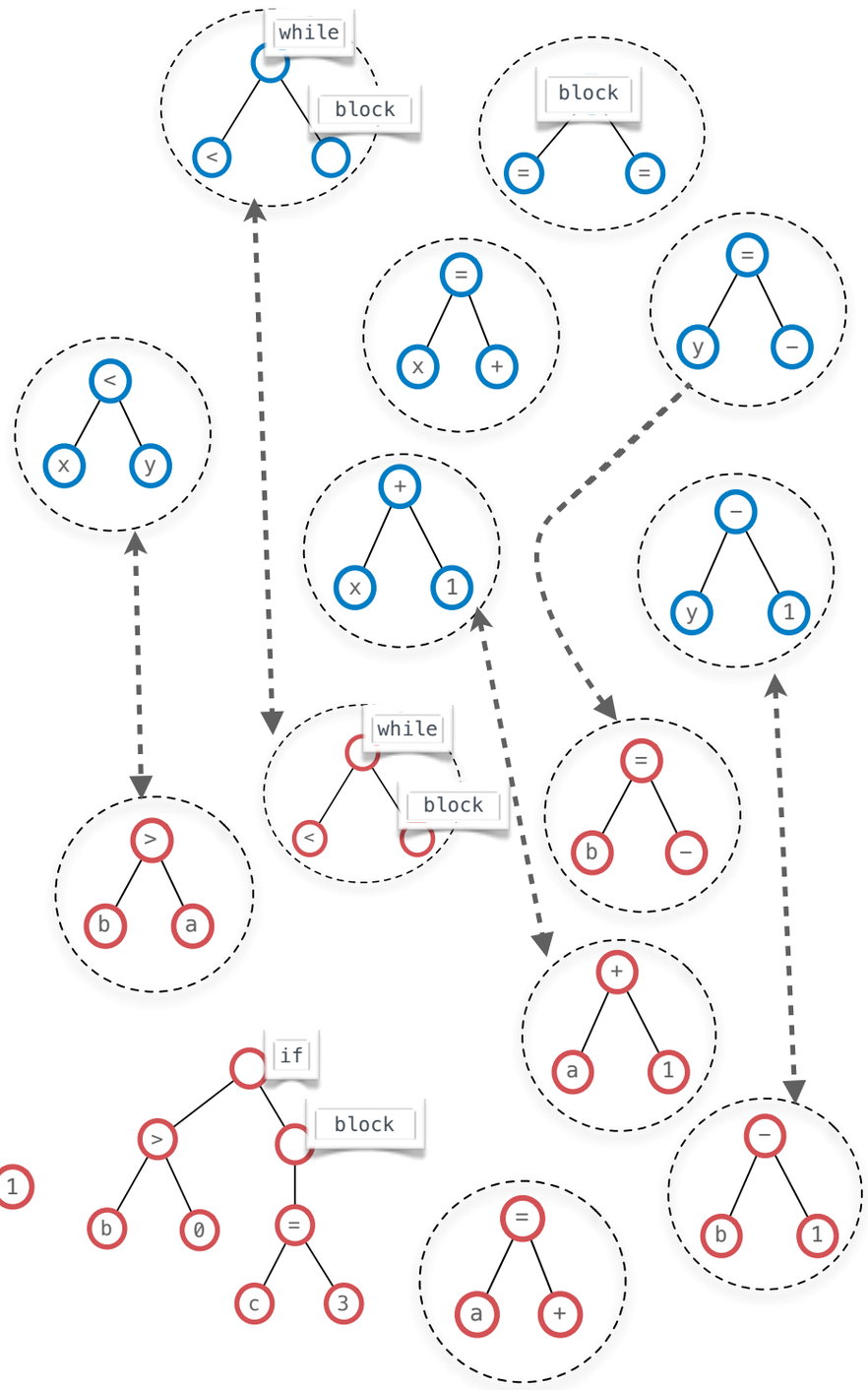
```
while (b > a){
    if (b > 0)
        c = 3;
    a = a + 1;
    b = b - 1;
}
```

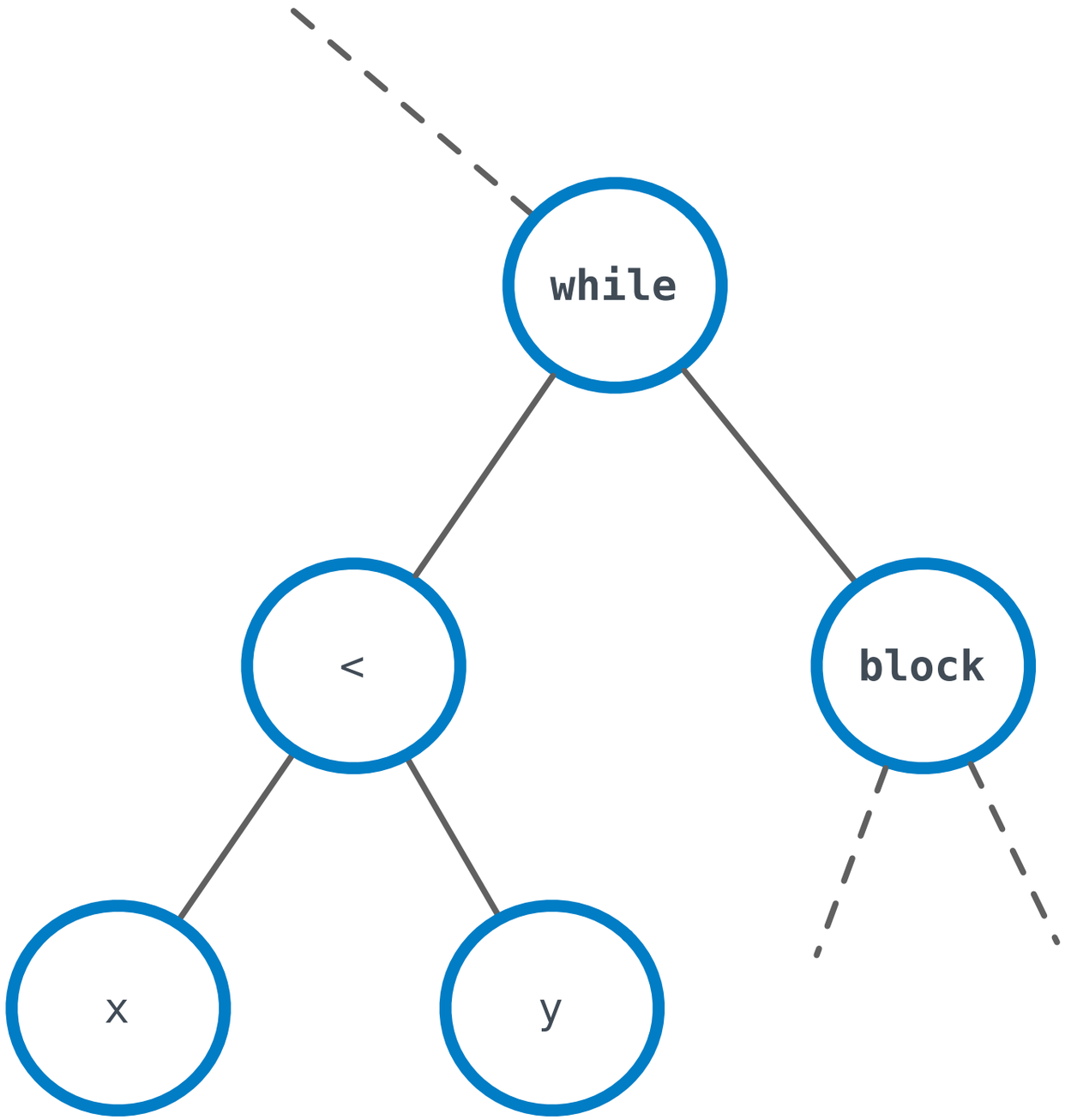
AST

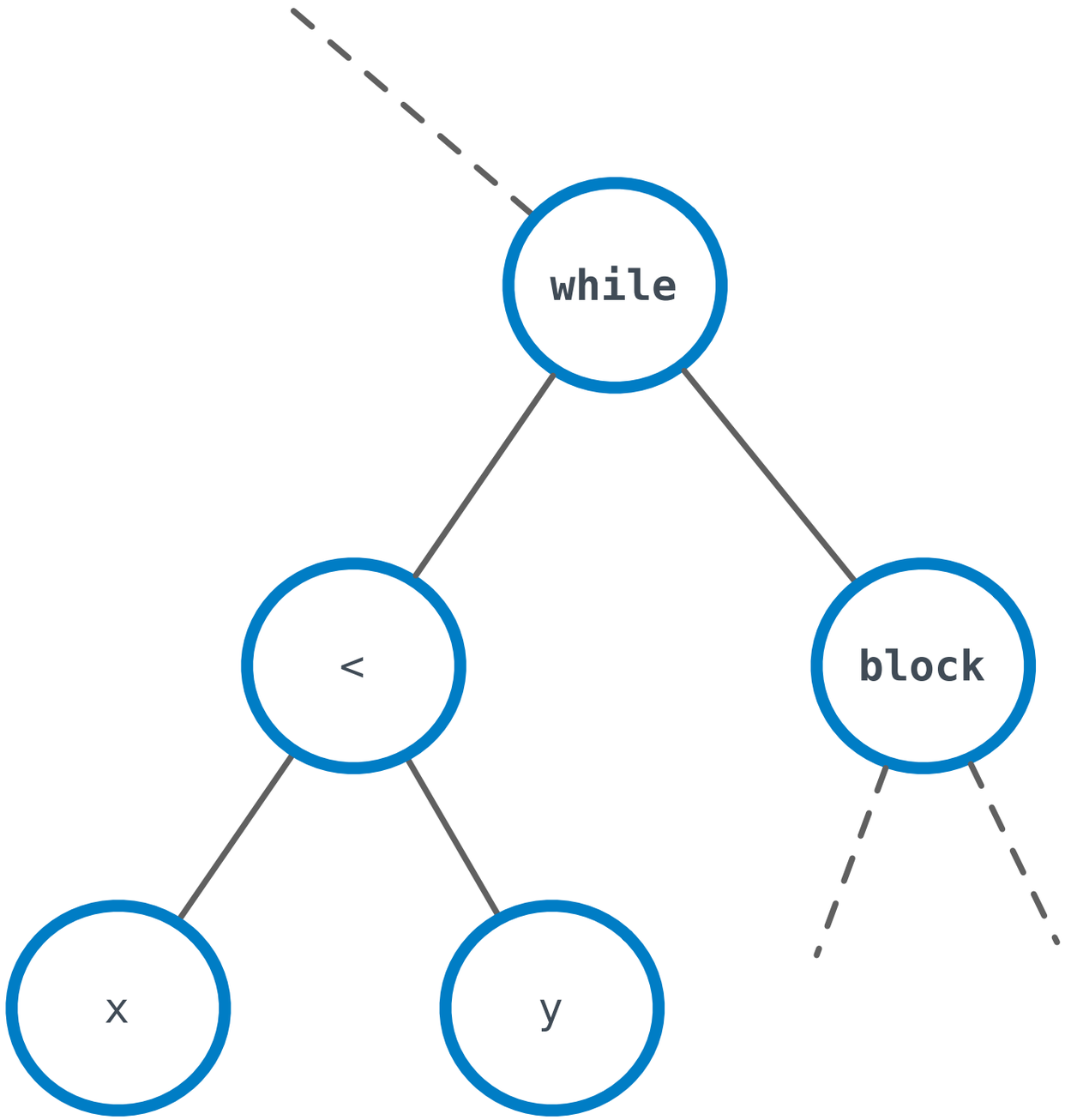




AST KERNEL



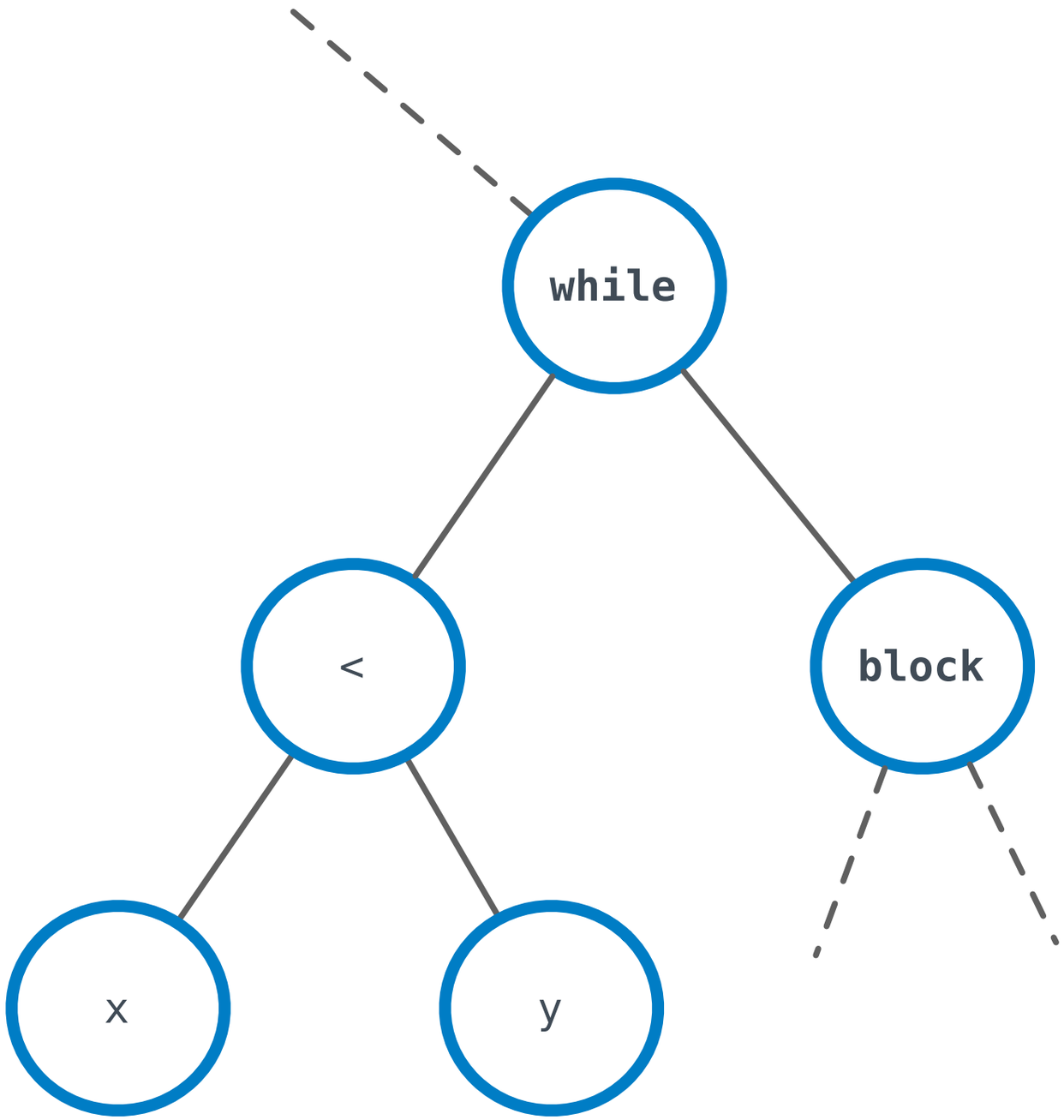

while
<
x
y
block



Instruction Class (IC)

i.e., `LOOP`, `CALL`, `CONDITIONAL_STATEMENT`

KERNEL FEATURES



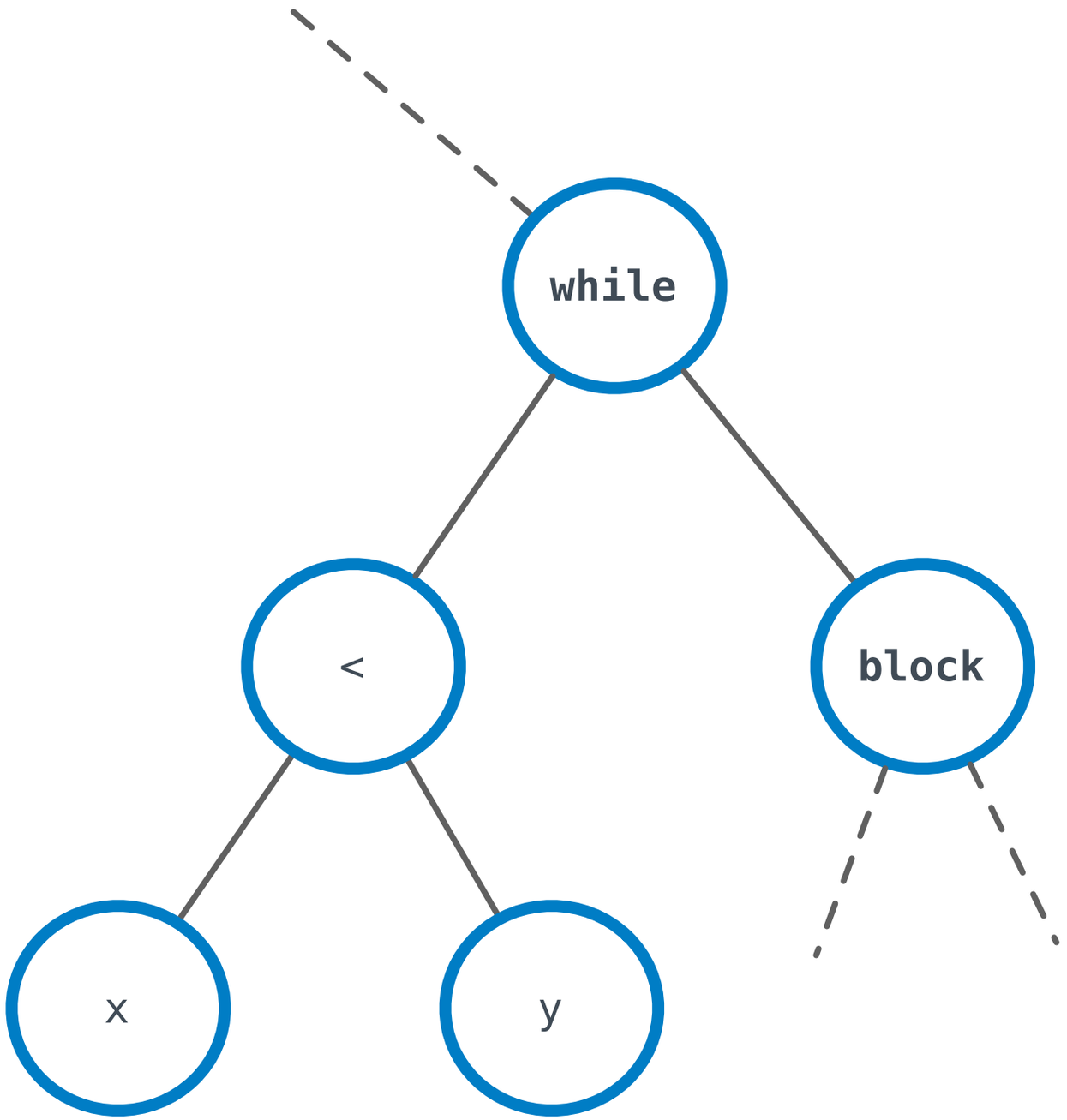
Instruction Class (IC)

i.e., `LOOP, CALL, CONDITIONAL_STATEMENT`

Instruction (I)

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

KERNEL FEATURES



Instruction Class (IC)

i.e., `LOOP, CALL, CONDITIONAL_STATEMENT`

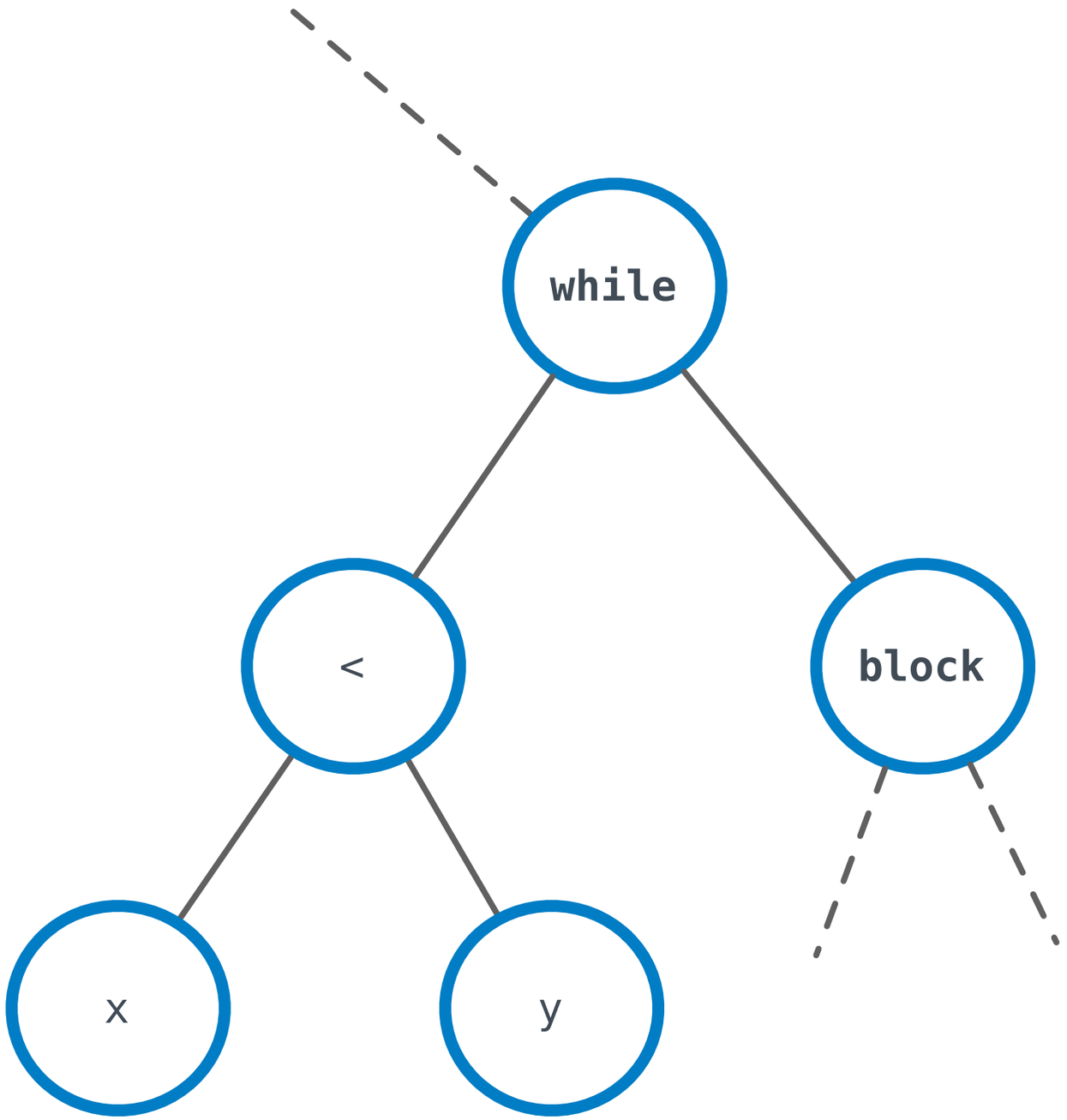
Instruction (I)

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

Context (C)

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

KERNEL FEATURES



Instruction Class (IC)

i.e., `LOOP, CALL, CONDITIONAL_STATEMENT`

Instruction (I)

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

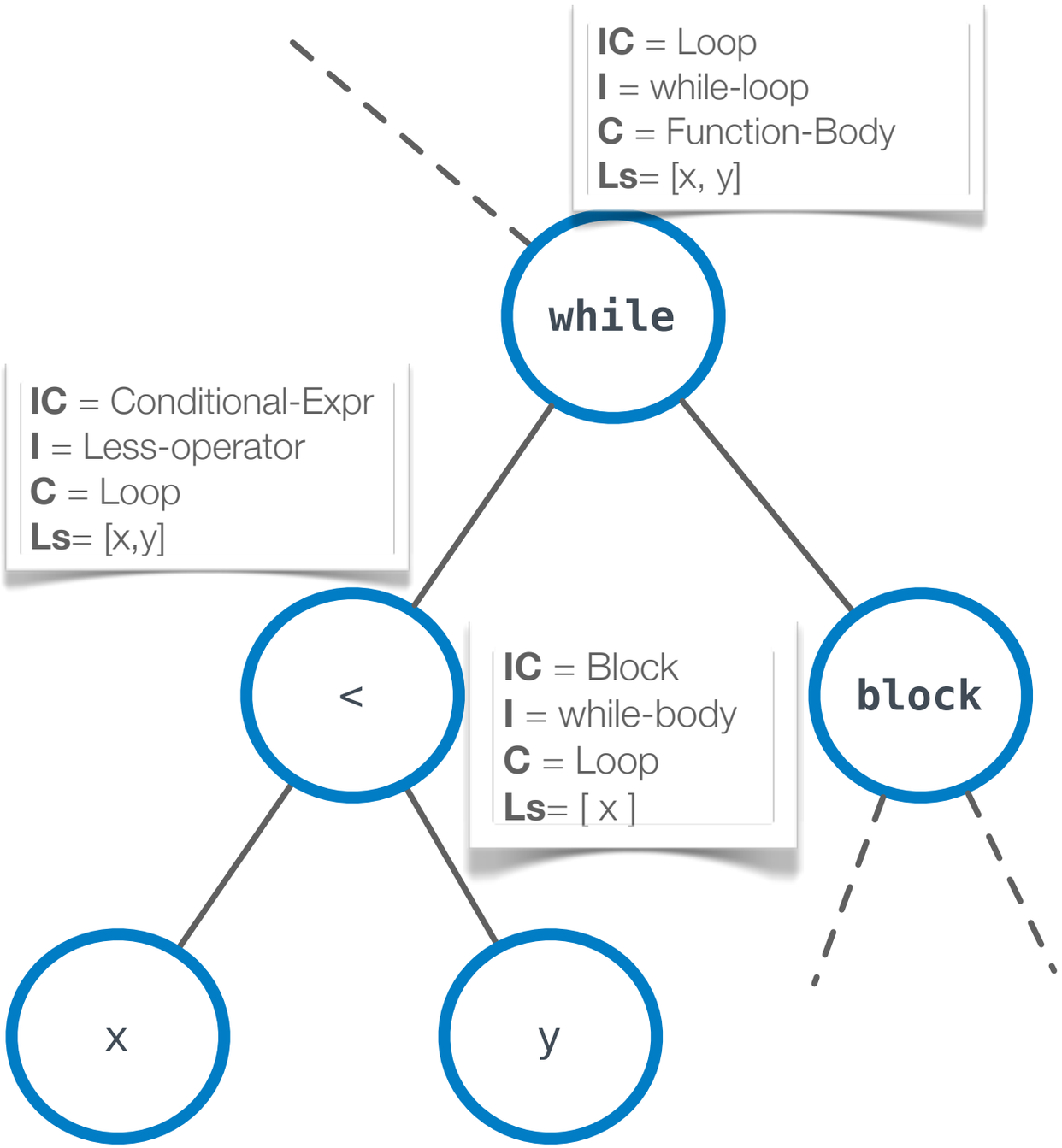
Context (C)

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

Lexemes (Ls)

Lexical information gathered (recursively) from leaves

KERNEL FEATURES



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

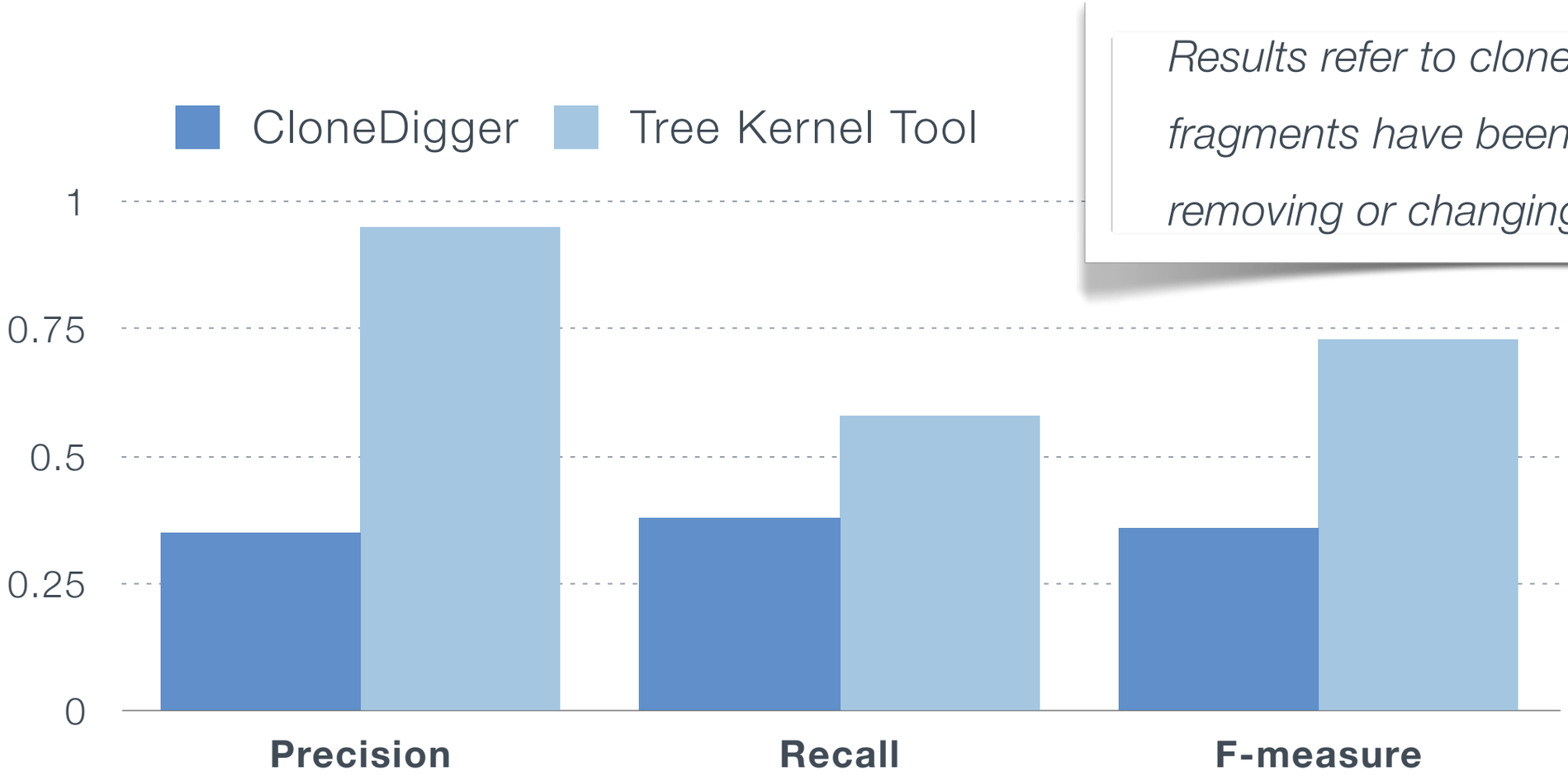
Lexemes (Ls)

Lexical information gathered (recursively) from leaves

RESULTS CLONE DETECTION

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

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



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.

THESIS CONTRIBUTIONS

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

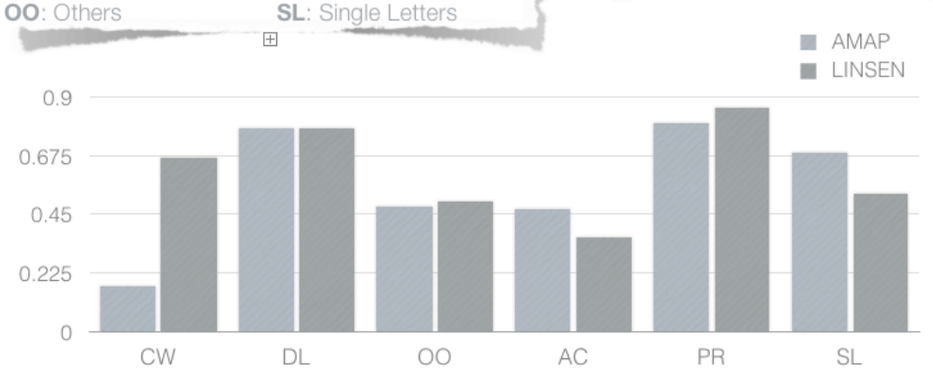
1. **SOFTWARE RE-MODULARIZATION**
2. **SOURCE CODE NORMALIZATION**
3. **CLONE DETECTION**

SOURCE CODE **LEXICAL** INFORMATION

RESULTS # 2 EXPANSION

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

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



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 CONTRIBUTIONS

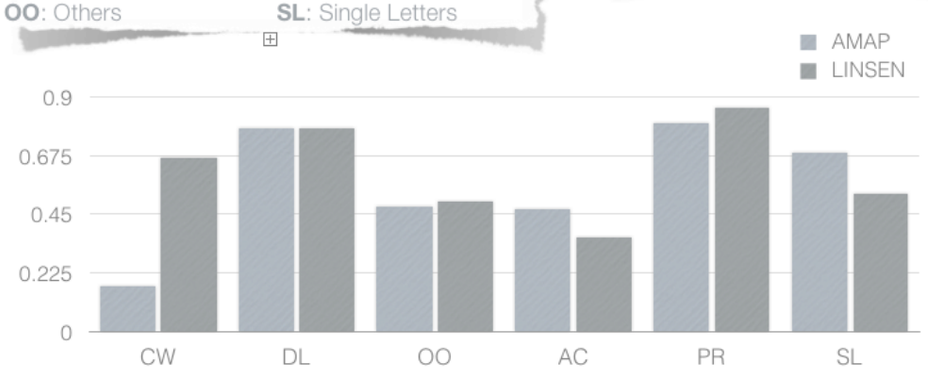
Contributions to three relevant and related **open issues** in Software Maintenance

1. **SOFTWARE RE-MODULARIZATION**
2. **SOURCE CODE NORMALIZATION**
3. **CLONE DETECTION**

RESULTS # 2 EXPANSION

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

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



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.

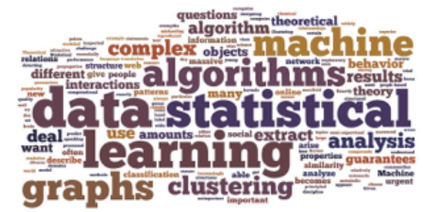
Software Maintenance could account up to the **85-90%** of the total software costs.

SOURCE CODE LEXICAL INFORMATION

THESIS CONTRIBUTIONS

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

&



1. **SOFTWARE RE-MODULARIZATION**
2. **SOURCE CODE NORMALIZATION**
3. **CLONE DETECTION**

RESULTS # 2

EXPANSION

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

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

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.

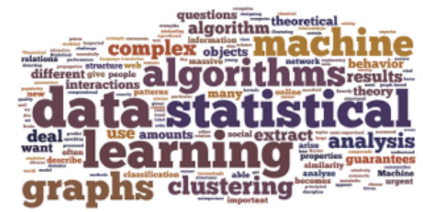
Software Maintenance could account up to the **85-90%** of the total software costs.

SOURCE CODE LEXICAL INFORMATION

THESIS CONTRIBUTIONS

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

&



1. **SOFTWARE RE-MODULARIZATION**
2. **SOURCE CODE NORMALIZATION**
3. **CLONE DETECTION**

RESULTS # 2 EXPANSION

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

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

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.

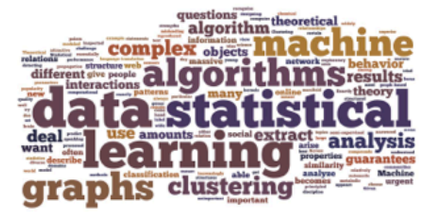
Software Maintenance could account up to the **85-90%** of the total software costs.

SOURCE CODE LEXICAL INFORMATION

THESIS CONTRIBUTIONS

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

&

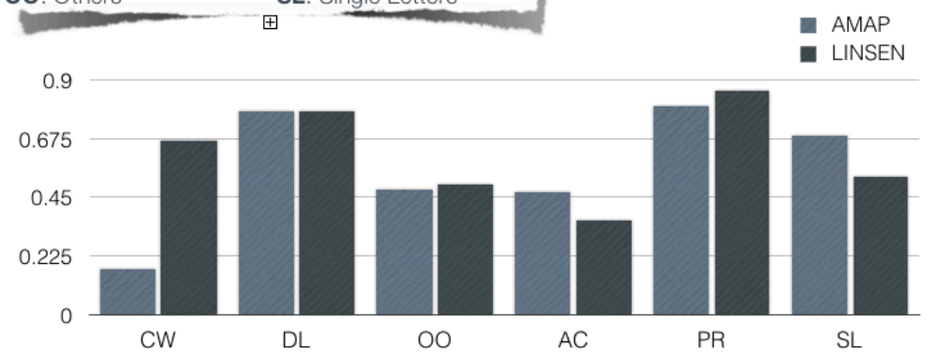


1. **SOFTWARE RE-MODULARIZATION**
2. **SOURCE CODE NORMALIZATION**
3. **CLONE DETECTION**

RESULTS # 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)



CONCLUSIONS

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*