# Pattern Mining for Systematic Code Changes

Kim Mens, Siegfried Nijssen, **Hoang Son Pham** – UCLouvain, Belgium

Johan Fabry – Raincode Labs, Belgium

Vadim Zaytsev – University Twente, Netherlands

THE 19TH BELGIUM-NETHERLANDS SOFTWARE EVOLUTION WORKSHOP

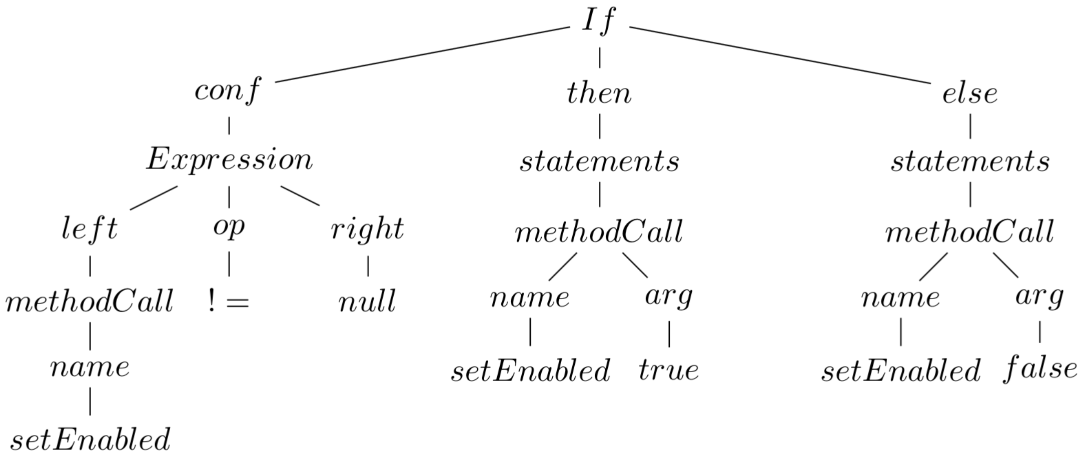
**LUXEMBOURG, 3/4 DECEMBER 2020**

# Introduction

- Systematic code changes
- How to discover these changes
  - Step 1: create database of changes
  - Step 2: mining patterns from database
- Our new approach
  - Applying Pattern mining algorithm
  - Mining patterns from abstract syntax trees

# Problem

```
if (getView() != null) {
      setEnabled(true);
   } else {
      setEnabled(false);
    }
```

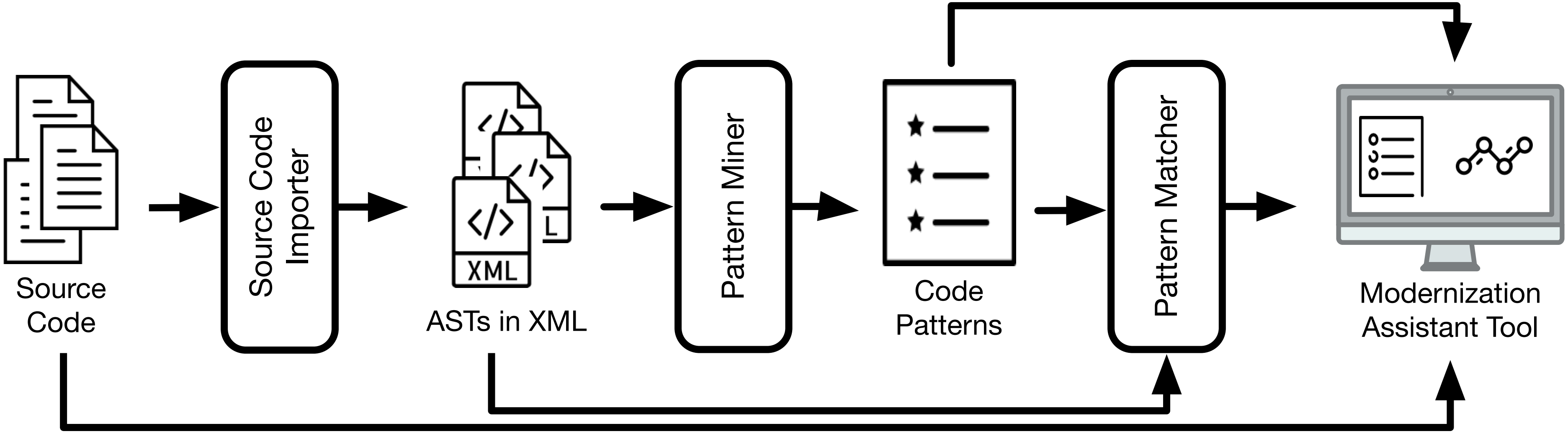


Source code ↔ Abstract syntax tree

Code fragments ↔ Subtrees

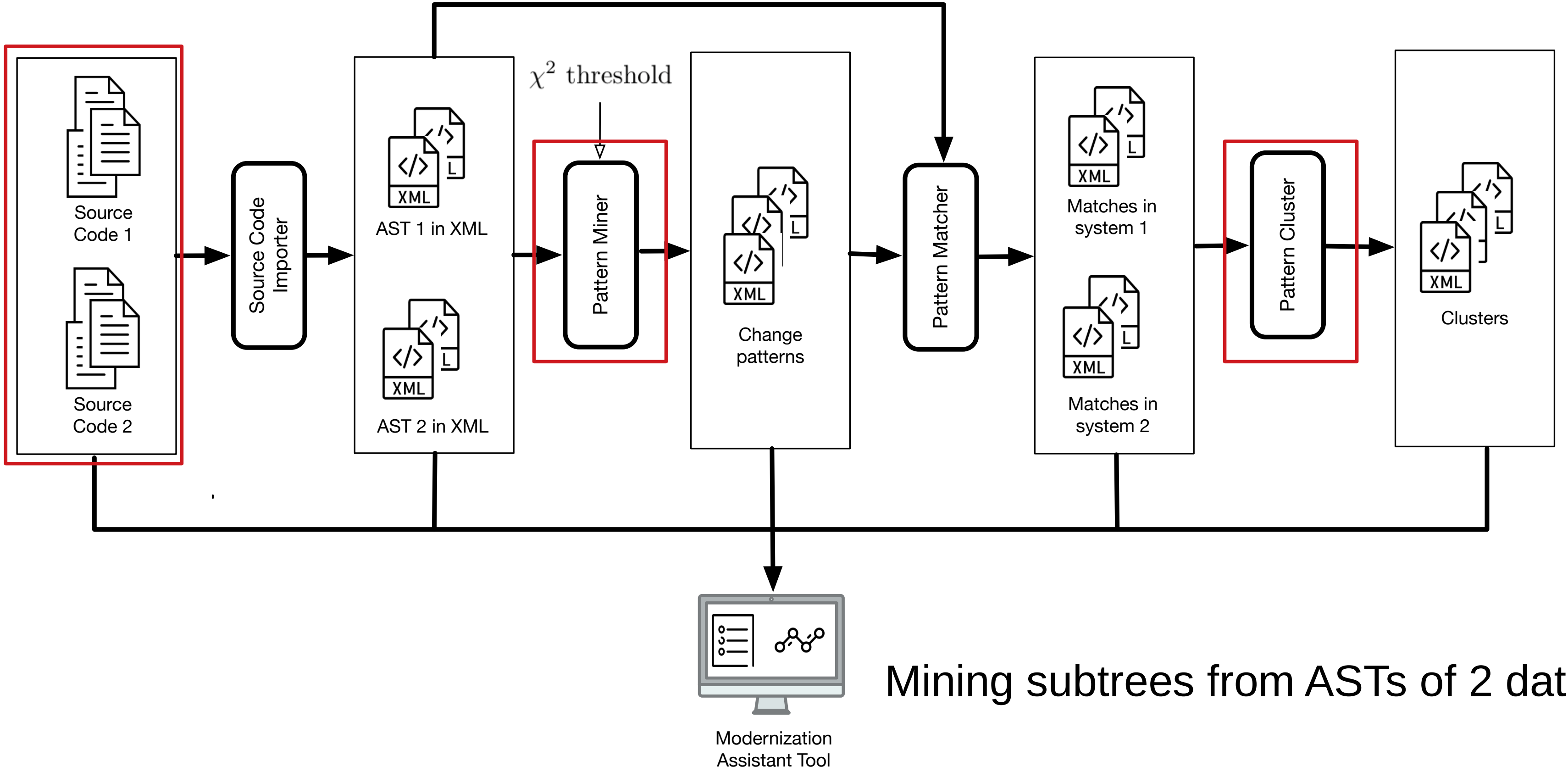
**Given ASTs of two versions of a system how to mine subtrees from these ASTs**

# Original Framework



Mining subtrees from ASTs of single dataset

# Extended Framework



Mining subtrees from ASTs of 2 datasets

# FREQTALS algorithm

mining frequent subtrees in ASTs

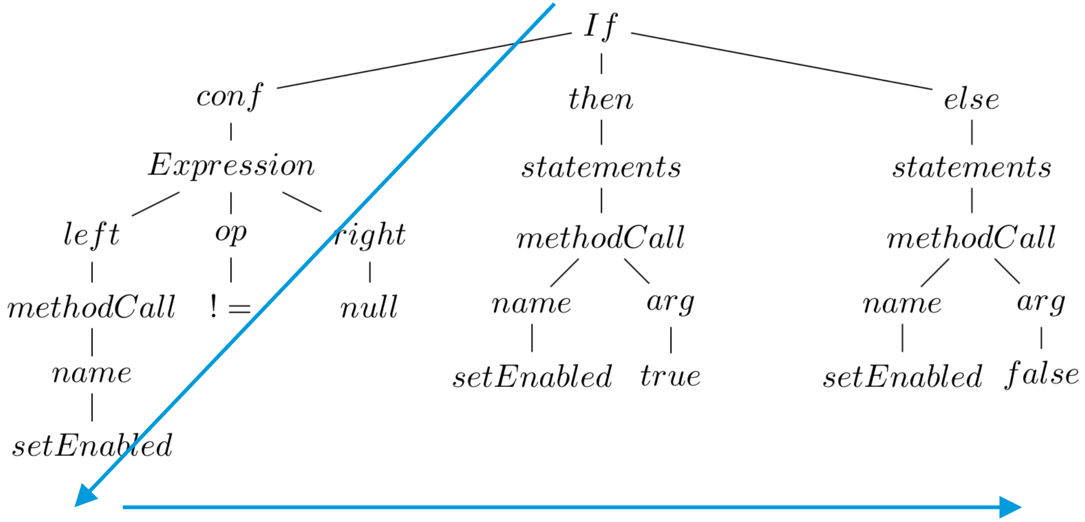
searching strategy

- depth-first, left-to-right
- right-most extension

constraints

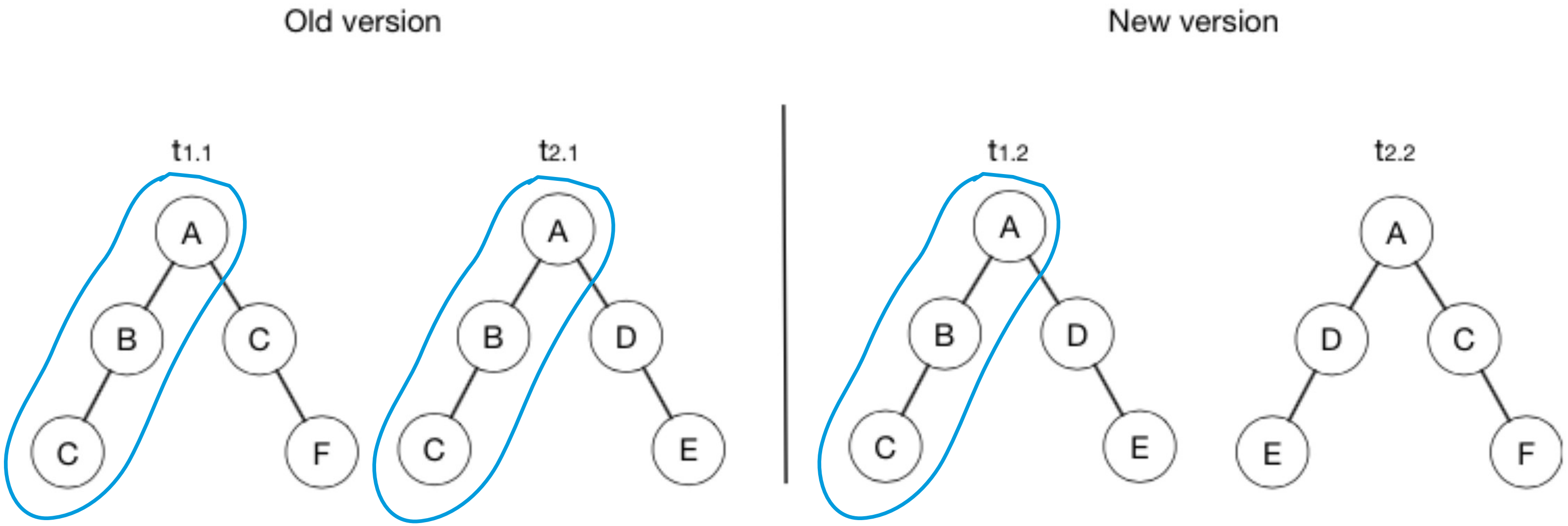
- support
- size
- labels

maximality: output maximal patterns



# Adapted FREQTALS

mine subtrees from ASTs of 2 datasets

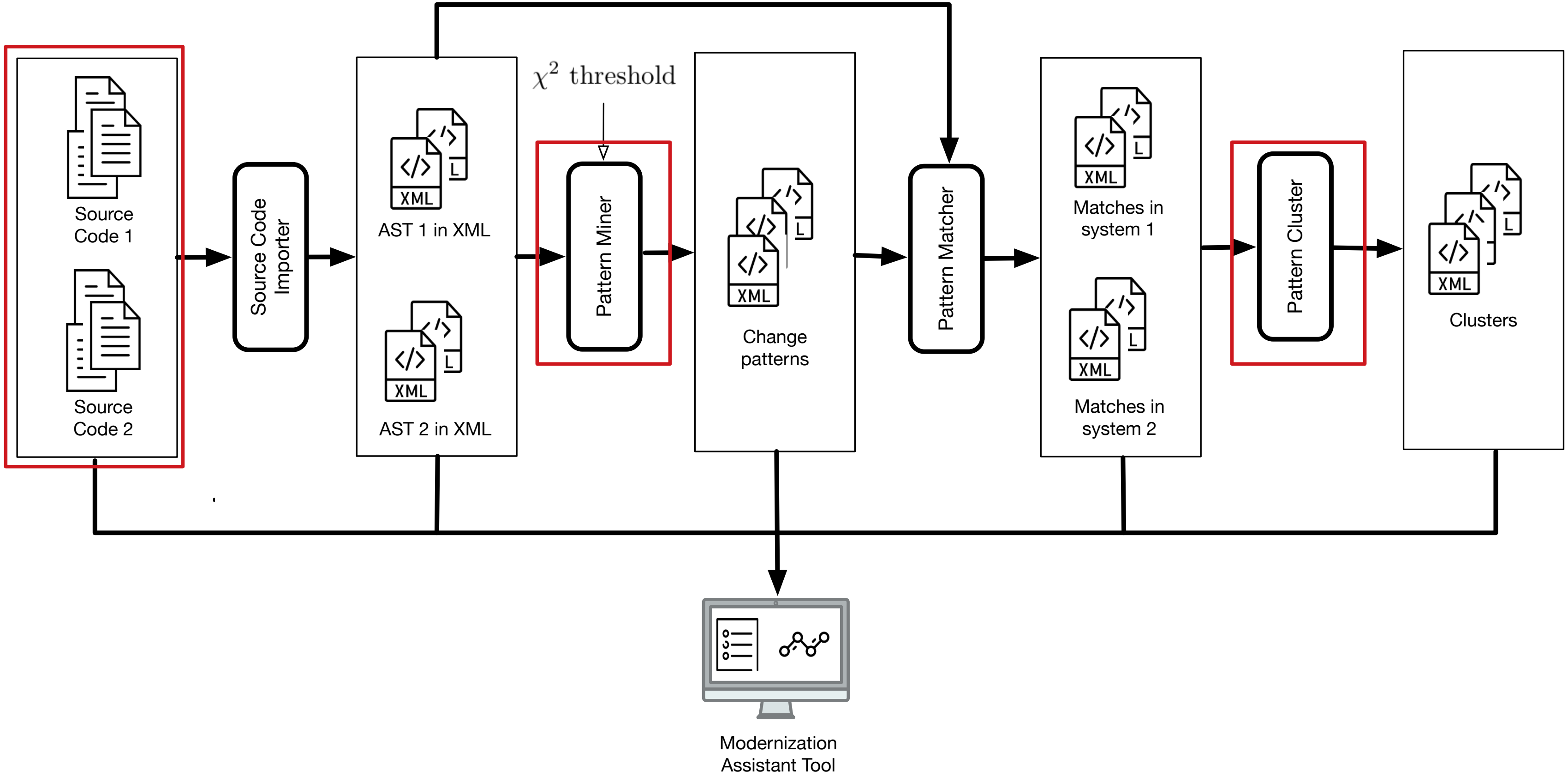


Interesting pattern: $\mathcal{X}^2$ >= minimum threshold

# Pattern clustering

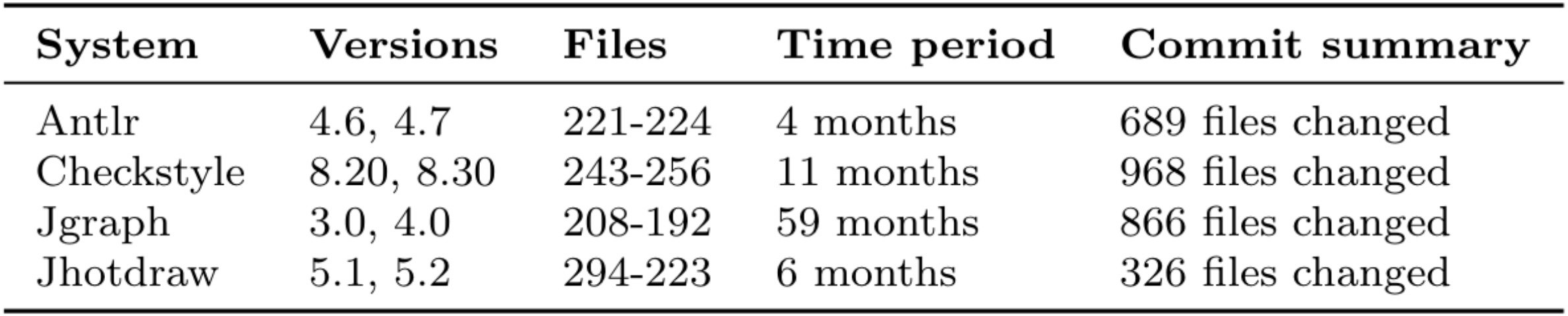
- Regroup similar patterns
  - set of labels
  - similar matches
  - tree edit distance
- Clustering algorithms
  - K-means
  - K-medoids
  - Affinity Propagation

# Extended Framework



# Case study

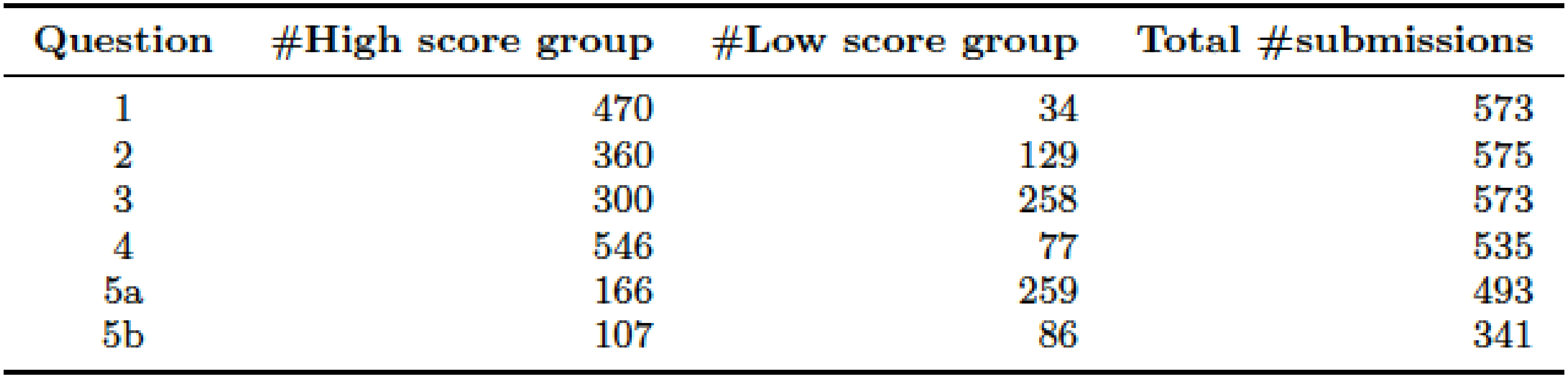
Experiment 1 : Mining source code changes between two versions



| System | Versions | Files | Time period | Commit summary |
|---|---|---|---|---|
| Antlr | 4.6, 4.7 | 221-224 | 4 months | 689 files changed |
| Checkstyle | 8.20, 8.30 | 243-256 | 11 months | 968 files changed |
| Jgraph | 3.0, 4.0 | 208-192 | 59 months | 866 files changed |
| Jhotdraw | 5.1, 5.2 | 294-223 | 6 months | 326 files changed |

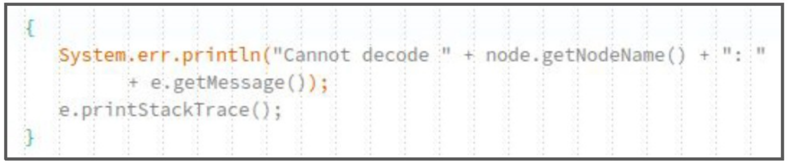
# Case study

Experiment 2 : Mining source code differences between high and low scoring students

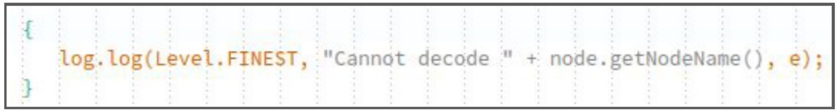


| Question | #High score group | #Low score group | Total #submissions |
|---|---|---|---|
| 1 | 470 | 34 | 573 |
| 2 | 360 | 129 | 575 |
| 3 | 300 | 258 | 573 |
| 4 | 546 | 77 | 535 |
| 5a | 166 | 259 | 493 |
| 5b | 107 | 86 | 341 |

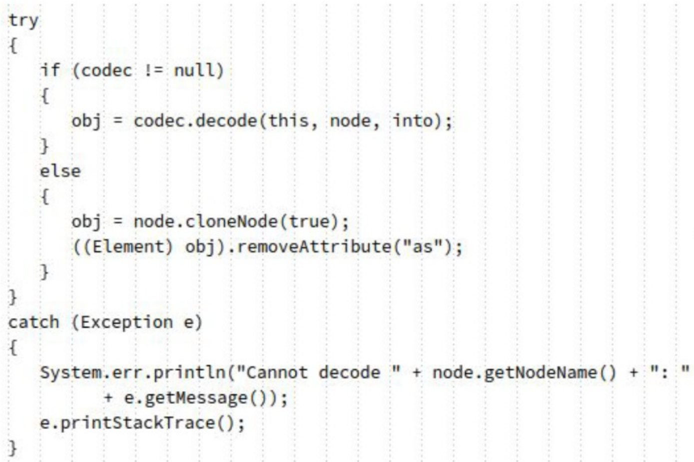
# Experiment 1 results



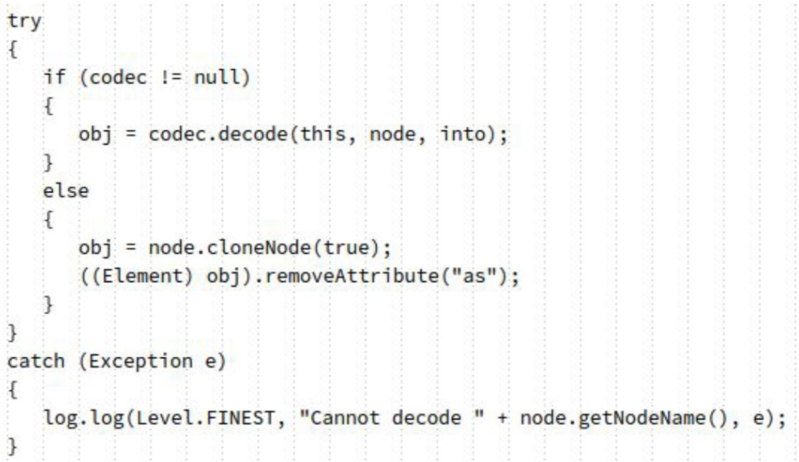
```
{
    System.err.println("Cannot decode " + node.getNodeName() + ": "
            + e.getMessage());
    e.printStackTrace();
}
```



```
{
    log.log(Level.FINEST, "Cannot decode " + node.getNodeName(), e);
}
```



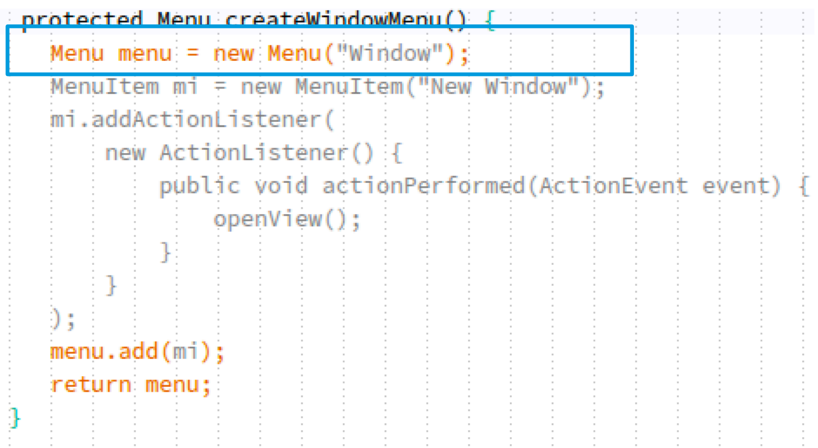
```
try
{
    if (codec != null)
    {
        obj = codec.decode(this, node, into);
    }
    else
    {
        obj = node.cloneNode(true);
        ((Element) obj).removeAttribute("as");
    }
}
catch (Exception e)
{
    System.err.println("Cannot decode " + node.getNodeName() + ": "
            + e.getMessage());
    e.printStackTrace();
}
```



```
try
{
    if (codec != null)
    {
        obj = codec.decode(this, node, into);
    }
    else
    {
        obj = node.cloneNode(true);
        ((Element) obj).removeAttribute("as");
    }
}
catch (Exception e)
{
    log.log(Level.FINEST, "Cannot decode " + node.getNodeName(), e);
}
```

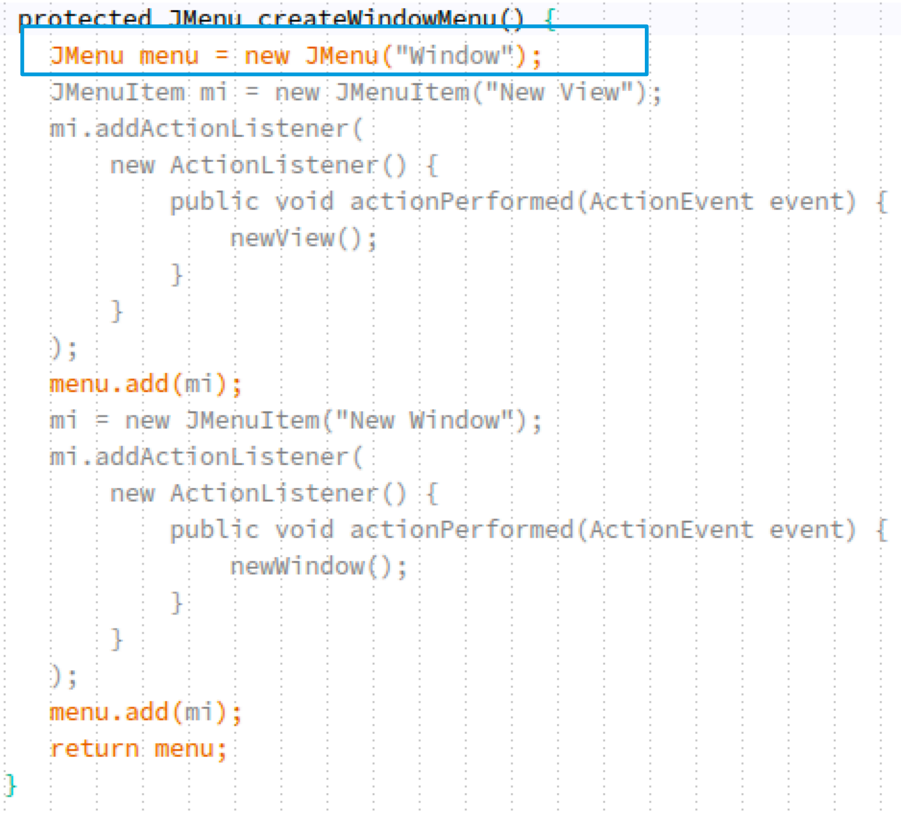
Refactoring pattern found in the Jgraph system

# Experiment 1 results



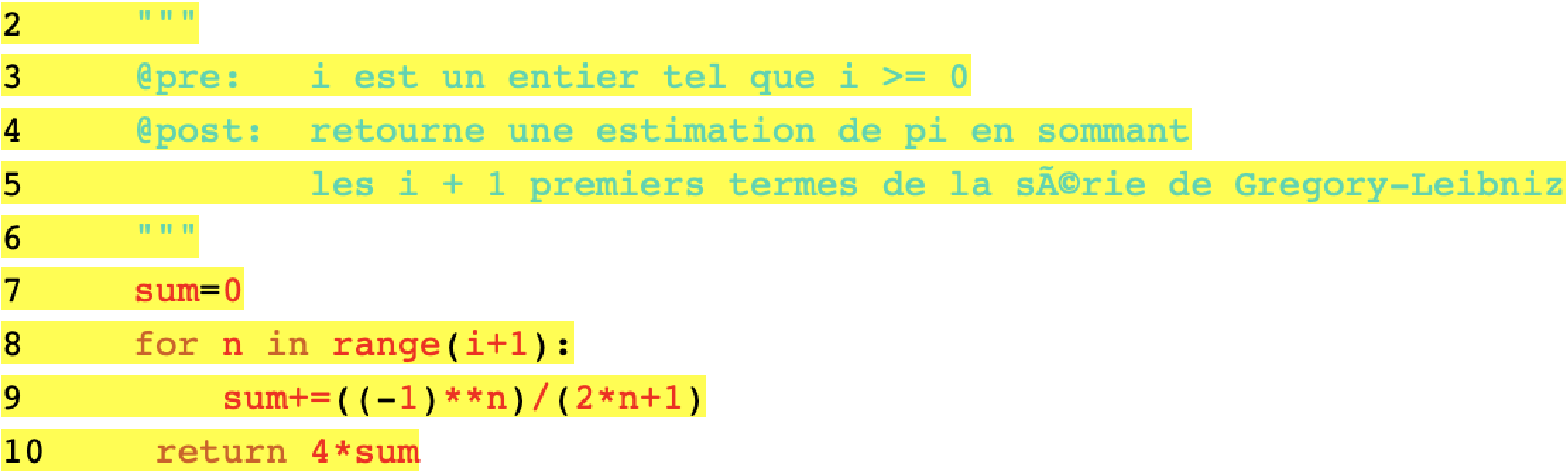
```
protected Menu createWindowMenu() {
    Menu menu = new Menu("Window");
    MenuItem mi = new MenuItem("New Window");
    mi.addActionListener(
        new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                openView();
            }
        }
    );
    menu.add(mi);
    return menu;
}
```

Change object **Menu** to **Jmenu found in the Jhotdraw system**



```
protected JMenu createWindowMenu() {
    JMenu menu = new JMenu("Window");
    JMenuItem mi = new JMenuItem("New View");
    mi.addActionListener(
        new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                newView();
            }
        }
    );
    menu.add(mi);
    mi = new JMenuItem("New Window");
    mi.addActionListener(
        new ActionListener() {
            public void actionPerformed(ActionEvent event) {
                newWindow();
            }
        }
    );
    menu.add(mi);
    return menu;
}
```

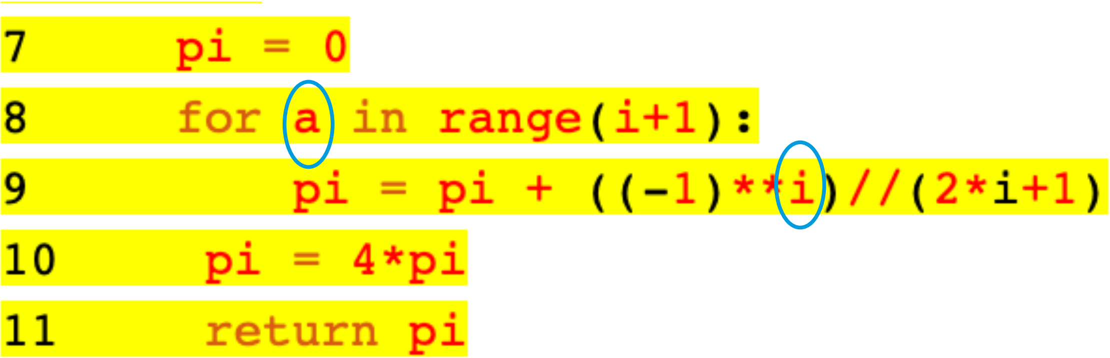
# Experiment 2 results



```
    """
    @pre:   i est un entier tel que i >= 0
    @post:  retourne une estimation de pi en sommant
            les i + 1 premiers termes de la sÃ©rie de Gregory-Leibniz
    """
    sum=0
    for n in range(i+1):
        sum+=((-1)**n)/(2*n+1)
     return 4*sum
```

A high frequent pattern found in the question 1 (occurs in 85 good solutions, absent in bad solution)

# Experiment 2 results



```
    pi = 0
    for a in range(i+1):
        pi = pi + ((-1)**i)//(2*i+1)
     pi = 4*pi
     return pi
```

Using wrong variable

Patterns occur in the low score group

# Conclusion

Contribution:

The adapted algorithm is able to discover interesting source code changes between two versions of a system or code differences between two groups

Limitation:

It cannot turn out interesting patterns if the changes are not frequent

Future works:

Evaluate the algorithm on larger datasets

Compare to other methods