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);
}

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: $\chi^2 \geq$ 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

Source Code 1

Source Code 2

Source Code Importer

AST 2 in XML

Pattern Miner

\( \chi^2 \) threshold

Change patterns

Pattern Matcher

AST 1 in XML

Matches in system 1

Matches in system 2

Pattern Cluster

Clusters

Modernization Assistant Tool

Pattern Cluster
## Case study

### Experiment 1: Mining source code changes between two versions

<table>
<thead>
<tr>
<th>System</th>
<th>Versions</th>
<th>Files</th>
<th>Time period</th>
<th>Commit summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antlr</td>
<td>4.6, 4.7</td>
<td>221-224</td>
<td>4 months</td>
<td>689 files changed</td>
</tr>
<tr>
<td>Checkstyle</td>
<td>8.20, 8.30</td>
<td>243-256</td>
<td>11 months</td>
<td>968 files changed</td>
</tr>
<tr>
<td>Jgraph</td>
<td>3.0, 4.0</td>
<td>208-192</td>
<td>59 months</td>
<td>866 files changed</td>
</tr>
<tr>
<td>Jhotdraw</td>
<td>5.1, 5.2</td>
<td>294-223</td>
<td>6 months</td>
<td>326 files changed</td>
</tr>
</tbody>
</table>
Case study

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

<table>
<thead>
<tr>
<th>Question</th>
<th>#High score group</th>
<th>#Low score group</th>
<th>Total #submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>470</td>
<td>34</td>
<td>573</td>
</tr>
<tr>
<td>2</td>
<td>360</td>
<td>129</td>
<td>575</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>258</td>
<td>573</td>
</tr>
<tr>
<td>4</td>
<td>546</td>
<td>77</td>
<td>535</td>
</tr>
<tr>
<td>5a</td>
<td>166</td>
<td>259</td>
<td>493</td>
</tr>
<tr>
<td>5b</td>
<td>107</td>
<td>86</td>
<td>341</td>
</tr>
</tbody>
</table>
Experiment 1 results

Refactoring pattern found in the Jgraph system
Experiment 1 results

Change object **Menu** to **Jmenu**
found in the Jhotdraw system
A high frequent pattern found in the question 1 (occurs in 85 good solutions, absent in bad solution)
Experiment 2 results

Patterns occur in the low score group

```python
7    pi = 0
8    for a in range(i+1):
9        pi = pi + ((-1)**i)//(2*i+1)
10   pi = 4*pi
11   return pi
```

Using wrong variable
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