Folding Repeated Instructions for Improving Token-based Code Clone Detection

Hiroaki Murakami, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki, Shinji Kusumoto

Graduate School of Information Science and Technology
Osaka University
Outline

• Background
• Problem of existing methods
• Proposed method
• Experiments
• Conclusion
What is code clone?

- A code clone is a code fragment in source files that is identical or similar to another.
- To detect code clones automatically, a variety of detection methods have been proposed.
Overlapped code clones

- Line-based/Token-based detections report many overlapped code clones.

```java
1: Public class Sample{
2:   String method1(){
3:       StringBuilder txt = new StringBuilder();
4:       txt.append("A");
5:       txt.append("B");
6:       return txt.toString();
7:   }
8: }
9: String method2(){
10:    StringBuilder txt = new StringBuilder();
11:    txt.append("C");
12:    txt.append("D");
13:    txt.append("E");
14:    return txt.toString();
15: }
16: }
```

Existing methods find 5 clone pairs consisting of overlapped code clones.
Characteristics of overlapped code clones

(1) almost the same location

```java
1: Public class Sample{
2:      String method1(){
3:          StringBuilder txt = new StringBuilder();
4:          txt.append("A");
5:          txt.append("B");
6:          return txt.toString();
7:      }
8:}
9:  
10:     String method2(){
11:          StringBuilder txt = new StringBuilder();
12:          txt.append("C");
13:          txt.append("D");
14:          txt.append("E");
15:          return txt.toString();
16:  }
```

(2) self-overlapping

```java
1: Public class Sample{
2:      String method1(){
3:          StringBuilder txt = new StringBuilder();
4:          txt.append("A");
5:          txt.append("B");
6:          return txt.toString();
7:      }
8:}
9:  
10:     String method2(){
11:          StringBuilder txt = new StringBuilder();
12:          txt.append("C");
13:          txt.append("D");
14:          txt.append("E");
15:          return txt.toString();
16:  }
```
Desirable detection result

1: Public class Sample{
2:     String method1(){
3:         StringBuilder txt = new StringBuilder();
4:         txt.append("A");
5:         txt.append("B");
6:             return txt.toString();
7:         }
8:     }
9:     String method2(){
10:        StringBuilder txt = new StringBuilder();
11:        txt.append("C");
12:        txt.append("D");
13:        txt.append("E");
14:            return txt.toString();
15:         }
16:     }

5 clone pairs

1: Public class Sample{
2:     String method1(){
3:         StringBuilder txt = new StringBuilder();
4:         txt.append("A");
5:         txt.append("B");
6:             return txt.toString();
7:         }
8:     }
9:     String method2(){
10:        StringBuilder txt = new StringBuilder();
11:        txt.append("C");
12:        txt.append("D");
13:        txt.append("E");
14:            return txt.toString();
15:         }
16:     }

single clone pair

2012/9/23

SCAM 2012
If repeated instructions are folded, the problem of overlapped code clones can be solved.
Example detection of proposed method

Result of existing method

1: Public class Sample{
2:     String mathod1(){
3:         StringBuilder txt = new StringBuilder();
4:            txt.append("A");
5:            txt.append("B");
6:            return txt.toString();
7:     }
8: }
9: String mathod2(){
10:    StringBuilder txt = new StringBuilder();
11:        txt.append("C");
12:        txt.append("D");
13:        txt.append("E");
14:        return txt.toString();
15:    }
16: }

Result of the proposed method

1: Public class Sample{
2:     String mathod1(){
3:         StringBuilder txt = new StringBuilder();
4:            txt.append("A");
5:            txt.append("B");
6:            return txt.toString();
7:     }
8: }
9: String mathod2(){
10:    StringBuilder txt = new StringBuilder();
11:        txt.append("C");
12:        txt.append("D");
13:        txt.append("E");
14:        return txt.toString();
15:    }
16: }

Our proposed method prevents overlapped code clones from being detected
How to evaluate proposed method?

- We implemented the proposed method as a tool, FRISC.
- Targets are 8 open source software systems.
- We compare precision and recall of FRISC with those of code clone detection tools.
  - Clone references are given by Bellon’s experiments[1].

<table>
<thead>
<tr>
<th>software</th>
<th>lang.</th>
<th>LOC</th>
<th>software</th>
<th>lang.</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>netbeans</td>
<td>Java</td>
<td>14,360</td>
<td>weltab</td>
<td>C</td>
<td>11,460</td>
</tr>
<tr>
<td>ant</td>
<td>Java</td>
<td>34,744</td>
<td>cook</td>
<td>C</td>
<td>70,008</td>
</tr>
<tr>
<td>jdtcore</td>
<td>Java</td>
<td>147,634</td>
<td>snns</td>
<td>C</td>
<td>93,867</td>
</tr>
<tr>
<td>swing</td>
<td>Java</td>
<td>204,037</td>
<td>postgresql</td>
<td>C</td>
<td>201,686</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>detectors</th>
<th>detection method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloneDR</td>
<td>AST-based</td>
</tr>
<tr>
<td>CLAN</td>
<td>metrics-based</td>
</tr>
<tr>
<td>CCFinder</td>
<td>token-based</td>
</tr>
<tr>
<td>Dup</td>
<td>token-based</td>
</tr>
<tr>
<td>Duploc</td>
<td>line-based</td>
</tr>
<tr>
<td>Duplix</td>
<td>PDG-based</td>
</tr>
<tr>
<td>Nicad</td>
<td>token-based</td>
</tr>
</tbody>
</table>

Precision and Recall without and with folding

Precision

Recall

without folding

with folding

without folding

with folding

70% up

0.022

0.037

4% down

0.88

0.84

2012/9/23

SCAM 2012

KUSUMOTO LABORATORY - Software Design Laboratory
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University. http://sdl.ist.osaka-u.ac.jp/
Precision and Recall of clone detectors

<table>
<thead>
<tr>
<th>Detectors</th>
<th>CLAN</th>
<th>FRISC</th>
<th>CCFinder</th>
<th>Dup</th>
<th>Duploc</th>
<th>Duplix</th>
<th>Nicad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection method</td>
<td>Metrics</td>
<td>Token</td>
<td>Token</td>
<td>Token</td>
<td>Line</td>
<td>PDG</td>
<td>Token</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>6</th>
<th>7</th>
<th>5</th>
<th>3</th>
<th>8</th>
<th>4</th>
</tr>
</thead>
</table>

Precision and Recall box plots for different clone detectors.

**Detectors**
- CloneDR
- CLAN
- FRISC
- CCFinder
- Dup
- Duploc
- Duplix
- Nicad

**Detection methods**
- AST
- Metrics
- Token
- Token
- Token
- Line
- PDG
- Token

**2012/9/23**

SCAM 2012
Conclusion

Overlapped Code Clones

- Line-based/Token-based detections report many overlapped code clones.

```
1: Public class Sample{
2:    String method1(){
3:        StringBuilder txt = new StringBuilder();
4:        txt.append("A");
5:        return txt.toString();
6:    }
7:
8:    String method2(){
9:        StringBuilder txt = new StringBuilder();
10:       txt.append("C");
11:       txt.append("D");
12:       return txt.toString();
13:    }
14: }
15: }
16: }
```

Proposed Method

If repeated instructions exist in both A1 and A2, the problem of overlapped code clones is solved.

Precision and Recall of clone detectors

- Precision and Recall for various clone detectors are evaluated using box plots.
- The detectors include CloneDR, CLAN, FRISC, CCFinder, Dup, Duploc, DupliX, and Nicad.
- Detection methods include AST, Metrics, Token, and Line.

2012/9/23

SCAM 2012