Developing a Model of Loop Actions by Mining Loop Characteristics from a Large Code Corpus

Xiaoran Wang, Lori Pollock, and K. Vijay-Shanker
University of Delaware
Observation 1: Current Source Code Analyses:

Unit = Method, Statement or Word
Do I need to worry about that?
"Your designs consist of soft, warm colors that are calming and soothing. No offense, but I pictured you looking a little different."

"Shouldn’t judge a book by its cover"
Individual statements are related.

Eat fruits, proteins, veggies.
Stop eating sweets and carbs.
Each less overall.
Reduce alcohol intake.
Exercise daily.
Reduce sitting time periods.
Lift weights.

“Small steps can lead to BIG CHANGES”
✓ Words have different meaning when put together.

“The whole is not always the sum of its parts.”
Text and structure analyzers in client tools care.

e.g.,

- Code Search
- Code Summary generators
- Traceability
- Code reuse analysis
Create and set up a queue menu item.

Create and set up a stop menu.

Build the menu.
Goal: Accurate method summary generator (Sridhara et al. 2010)

Approach:
1. Select individual statements to represent method’s major steps
2. Generate as natural language phrase

Observation: Needed to identify high level actions

Approach: Manually created templates (Sridhara et al. 2011)

Limitation: Not automated; Not extensible
Our Current Research Question

Can we define and automatically identify these high-level algorithmic steps in real-world codes?

*Noun. Action Unit:*

A code block that consists of a sequence of consecutive statements that logically implement a high level action as a substep within a method’s primary function.
This Paper’s Contributions

✔ Identify Java loop action units based on their structure, data flow, linguistic features learned from code corpus

✔ Demonstrate feasibility of automatically characterizing loops into stereotypes from code corpus

✔ Determine action to represent loop stereotype from clustering loops based on verb distribution on existing internal comments

✔ Evaluate effectiveness through human judgement study
Action Identification Process

1. Loop Source Code
2. Extract Feature Vector
3. Identify Action
4. Action Identification Model
5. High Level Action for Loop
**Loop-if:** Java loop (for, enhanced-for, while, do-while) with single if-statement as last lexical statement

*Of 14,317 Java projects, 1.3 M loops, 26% loop-if*
## Loop-if Feature Vectors

<table>
<thead>
<tr>
<th>Label</th>
<th>Feature</th>
<th>Possible Values and Their Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Type of ending statement</td>
<td>0: none 1: assignment 2: increment 3: decrement 4: method invocation 5: object method invocation 6: boolean assignment</td>
</tr>
<tr>
<td>F2</td>
<td>Method name of ending statement method call</td>
<td>0: none 1: add 2: addX 3: put 4: setX 5: remove</td>
</tr>
<tr>
<td>F3</td>
<td>Elements in collection get updated</td>
<td>0: false 1: true</td>
</tr>
<tr>
<td>F4</td>
<td>Usage of loop control variable in ending statement</td>
<td>0: not used 1: directly used 2: used indirectly through data flow</td>
</tr>
<tr>
<td>F5</td>
<td>Type of loop exit statement</td>
<td>0: none 1: break 2: return 3: return boolean 4: return object 5: throw</td>
</tr>
<tr>
<td>F6</td>
<td>Multiple collections in if condition</td>
<td>0: false 1: true</td>
</tr>
<tr>
<td>F7</td>
<td>Result variable used in if condition</td>
<td>0: false 1: true</td>
</tr>
<tr>
<td>F8</td>
<td>Type of if condition</td>
<td>1: &gt;/&lt;/&gt;/=/&lt;=&gt; 2: others</td>
</tr>
</tbody>
</table>

### Diagram

```
for (int i = 0; i < cands.size(); i++) {
    Candidate cand = cands.get(i);
    if (cand.label.equals(gold.toString())) {
        goldCand = cand;
        break;
    }
}
```

- **Loop Control Variable**
- **Candidate**
- **If Condition**
- **Result Variable**
- **Ending Statement of If Block**
- **Loop Exit Statement**
### Loop Action Identification Model

<table>
<thead>
<tr>
<th>Label</th>
<th>Action Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>count the number of elements in a collection that satisfy some condition</td>
</tr>
<tr>
<td>determine</td>
<td>determine if an element of a collection satisfies some condition</td>
</tr>
<tr>
<td>max/min</td>
<td>find the maximum/minimum element in a collection</td>
</tr>
<tr>
<td>find</td>
<td>find an element that satisfies some condition (other than max/min)</td>
</tr>
<tr>
<td>copy</td>
<td>copy elements that satisfy some condition from one collection to another</td>
</tr>
<tr>
<td>ensure</td>
<td>ensure that all elements in the collection satisfy some condition</td>
</tr>
<tr>
<td>compare</td>
<td>compare all pairs of corresponding elements from two collections</td>
</tr>
<tr>
<td>remove</td>
<td>remove elements when some condition is satisfied</td>
</tr>
<tr>
<td>get</td>
<td>get all elements that satisfy some condition</td>
</tr>
<tr>
<td>add</td>
<td>add a property to an object</td>
</tr>
<tr>
<td>set_one</td>
<td>set properties of an object using objects in a collection that satisfy some condition</td>
</tr>
<tr>
<td>set_all</td>
<td>set a property for all objects in a collection that satisfy some condition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>F4</th>
<th>F5</th>
<th>F6</th>
<th>F7</th>
<th>F8</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>2</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>determine</td>
<td>0</td>
<td>2,3</td>
<td>0,1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>max/min</td>
<td>1</td>
<td>1,2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>find</td>
<td>1</td>
<td>1,2</td>
<td>1,2,4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>copy</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ensure</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Developing the Loop Action Identification Model

1. Open Source Project Repository
2. Extract Comment-loop Associations
3. Extract Verbs from Comments
4. Compute Verb Distributions for Vectors
5. Cluster Vectors Based on Verb Distribution
6. Select Representative Action for Each Cluster
7. Action Identification Model
1. **Effectiveness:** 15 humans; 180 judgements on 60 loops total, 3 per loop, over all action stereotypes.
   1. How much do you agree that loop code implements this action?
   2. How confident are you in your assessment?

2. **Prevalence (Impact):**
   1. Ran prototype on test set of 7,159 projects (over 9M methods).
   2. Collected frequency of each of the 12 actions
Conclusion:
Human judges view our automatically identified descriptions as accurately expressing the high level actions of loop-ifs.

(a) Individual judgements
(b) Average judgements per loop

Fig. 4: Human judgements of identified actions (1: Strongly disagree, 2: Disagree, 3: Neither agree or disagree, 4: Agree, 5: Strongly agree)
Evaluation
Results & Conclusions

Prevalence (Impact)

Conclusion:
Our algorithm for automatically identifying loop-ifs that implement high level actions has wide applicability.

Fig. 5: Identified high level action distribution over 7,159 projects
Where do we go from here?

- Other kinds of action units
- Integrate within tools and evaluate impact
- Extensibility to other PLs