Automatic Model Generation from Documentation for Java API Functions

Juan Zhai, Jianjun Huang, Shiqing Ma, Xiangyu Zhang,
Lin Tan, Jianhua Zhao, Feng Qin
Motivation

• Libraries
  • Part of software behaviors
  • As important as the software itself

• Challenges
  • Binary only, no source code
  • Implemented in different languages
  • Complex optimizations
  • Many more......

• Previous works: generate models manually

Automatically Generate Models for Libraries
How to model library behaviors?

- Q: How do we know how to use libraries?
  - A: By reading API documents.

- Q: Can we model libraries from API documents?
  - A: Yes. This is what we do.

```java
public void add(int index, E element) {
    if (index < 0 || index > size())
        throw new IndexOutOfBoundsException();
    size = size + 1;
    for (int i = size - 1; i > index; i --)
        elements[i] = elements[i - 1];
    elements[index] = element;
}
```
Design: Text Analyzer

- State-of-the-art work: Stanford Parser

Not perfect because of the nature (e.g. ambiguity) of natural languages
Returns the head of this deque, or null if this deque is empty.
Design: Tree Node Transformer

• **Ambiguities** of natural languages
  • $K$ tree candidates
  • Highest-scoring tree is not always the correct one
  • Too much time to try all candidates

• **Observation:**
  • Caused by phrases starting with “; or” and “; and”

• **Solution:**
  • Lift up & Push down “; or”, “; and” and all their right siblings
Design: Generators

- Javadoc
  - Formal Definition
  - Natural Language

- Java code (Model)
  - Package, class, function metadata
  - Function body

Variables, Structures, Operations
Design: IR Generator -- Variables

• Internal variables: any names
• *Parameters*: identify from documents

```java
add(int index, E element)
```

Inserts the specified $element$ at the specified $position$ in this list.

Inserts $element$ at $index$ in this list.
Design: IR Generator -- Program structures

• Sequential
  • Default

• Loop structure
  • plurals
  • singular nouns modified by “each”
  • the first/last occurrence → indicate the loop iteration order

• Conditional structure
  • if/when
  • otherwise
Design: Model Generator

- Tile the IR (tree)

Tree Pattern:

- Returns
- ,or
- -1 tag:if
- the index tag:ltr
- of
- o tag:this
- contain tag:-
- o

Tree Pattern:

- return o1 ,or o2 o3
- (-)? is empty
- the index (tag:t)
- of o1 (tag:o2)
- (-)? contain o2
- copy o1 into o2
- throw o1
Design: Model Generator

• Unified data structure model: one-dimensional array

• Primitive: tree pattern corresponds to a piece of code template

```
insert

o1

at

o2(tag:this)

insert

elements[o2] = o1

elements[index] = element
```
Design: Model Generator

- Tile the IR (tree)

```java
int index1 = -1;
if(o == null){
    for(int i=0; i<size; i++){
        if(element[i] == null){
            index1 = i;
            break;
        }
    }
} else{
    for(int i=0; i<size; i++){
        if(o.equals(element[i])){
            index1 = i;
            break;
        }
    }
}
if(index2 == -1) return -1;
else return index1;
```
Design: Model Validator
Evaluation Setup

• Hardware
  • CPU: Intel® i7-3770
  • RAM: 8GB

• Operating system
  • Ubuntu 12.04
## Evaluation: Overall Result

<table>
<thead>
<tr>
<th>Class</th>
<th># Total Methods</th>
<th># Modeled Methods</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayList</td>
<td>34</td>
<td>29</td>
<td>85.29%</td>
</tr>
<tr>
<td>Vector</td>
<td>54</td>
<td>46</td>
<td>85.19%</td>
</tr>
<tr>
<td>Stack</td>
<td>6</td>
<td>5</td>
<td>83.33%</td>
</tr>
<tr>
<td>ArrayDeque</td>
<td>36</td>
<td>35</td>
<td>97.22%</td>
</tr>
<tr>
<td>LinkedList</td>
<td>42</td>
<td>41</td>
<td>97.62%</td>
</tr>
<tr>
<td>HashMap</td>
<td>28</td>
<td>23</td>
<td>82.14%</td>
</tr>
<tr>
<td>LinkedHashSet</td>
<td>15</td>
<td>14</td>
<td>93.33%</td>
</tr>
<tr>
<td>HashSet</td>
<td>13</td>
<td>12</td>
<td>92.31%</td>
</tr>
<tr>
<td>LinkedHashMap</td>
<td>5</td>
<td>4</td>
<td>80.00%</td>
</tr>
<tr>
<td>AttributeList</td>
<td>15</td>
<td>11</td>
<td>73.33%</td>
</tr>
<tr>
<td>RoleList</td>
<td>14</td>
<td>9</td>
<td>64.29%</td>
</tr>
<tr>
<td>RoleUnresolvedList</td>
<td>14</td>
<td>9</td>
<td>64.29%</td>
</tr>
<tr>
<td>StringBuffer</td>
<td>54</td>
<td>40</td>
<td>74.07%</td>
</tr>
<tr>
<td>StringBuilder</td>
<td>54</td>
<td>40</td>
<td>74.07%</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td><strong>397</strong></td>
<td><strong>326</strong></td>
<td><strong>82.12%</strong></td>
</tr>
</tbody>
</table>
Evaluation: Cases that Cannot be Handled

• Incompleteness of API documents
  • `add(int index, Object element)` in `AttributeList`
    lack of descriptions about the `IndexOutOfBoundsException`

• Describe one primitive behavior with several sentences
  • `insert(int index, Char[] str, int offset, int len)` in `StringBuffer`
    Inserts the string representation of a subarray of the str array argument into this sequence.
    The subarray begins at the specified offset and extends len chars.
    The characters of the subarray are inserted into this sequence at the position indicated by index
Evaluation: Static Taint Analysis

• Undesirable information flow

• Compare paths found by using *our model V.S. official JDK*

• Set up
  • Android: 96 apps
  • Sources: User input
  • Sinks: Internet, Log
Evaluation: Static Taint Analysis

• Results
  • the same set of information leak warnings for both versions for almost all apps
  • except app com.yes123.mobile

• Case: com.123yes.mobil
  • Our model - 16 paths V.S. JDK – 14 paths
  • Java Native Interface (JNI) function call: toArray(object[]) -> System.arraycopy()
Evaluation: Static Taint Analysis

• Efficiency improvement distribution

Maximum: ~50%
Average: ~16%
Related Work

• Documentation Analysis
  • Sarah [ICSE ’16], Zhong [OOPSLA ’13, ASE ’09], Tan [ICST ’12, ICSE ’11, SOSP ’07], Pandita [ICSE ’12], Sun [ICSE ’10], Sinha [ICST ’10], Runeson [ICSE ’07], Henkel [TSE ’07]

• Environment Modeling
  • Jeon [ICSE ’16], Merwe [SEN ’15], Ceccarello [SEN ’14], Palepu [ASE ’13], Qi [WCRE ’12], Cadar [OSDI ’08], Tkachuk [ASE ’03]
Conclusion

• **Idea**: modeling Java library from Java API documents
  • A combination of NLP and auto-testing

• **Advantages**
  • Expected behaviors with simpler code (no JNI code, no other languages etc.)
  • Helps many program analysis techniques
Thank You

Q&A