Mining Models of Object Behavior

Andrzej Wasylkowski, Saarland University
(with Valentin Dallmeier and Nicolas Bettenburg)

Advisors: Christian Lindig and Andreas Zeller
The problem

Java.util.Vector

boolean add(Object o)
void clear()
boolean isEmpty()
Object remove(int index)
boolean removeAll(Collection c)
...

• What is the effect of calling ‘clear’?

• Can ‘remove’ be called at any time?

How to use a class correctly?
What do object behavior models look like?

Object behavior model for the java.util.Vector class
How can we mine object behavior models?

1. **Instrumenting bytecode**
   - Target class
   - Other classes

2. **Identifying inspectors**
   - Inspector methods
   - Instrumented target class

3. **Running the program**
   - Object behavior model

4. **User**
Identifying inspectors

<table>
<thead>
<tr>
<th>java.util.Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean add(Object o)</td>
</tr>
<tr>
<td>void clear()</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
</tr>
<tr>
<td>Object remove(int index)</td>
</tr>
<tr>
<td>boolean removeAll(Collection c)</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Identifying inspectors

<table>
<thead>
<tr>
<th>java.util.Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean add(Object o)</td>
</tr>
<tr>
<td>void clear()</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
</tr>
<tr>
<td>Object remove(int index)</td>
</tr>
<tr>
<td>boolean removeAll(Collection c)</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- No parameters
Identifying inspectors

<table>
<thead>
<tr>
<th>java.util.Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean add(Object o)</td>
</tr>
<tr>
<td>void clear()</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
</tr>
<tr>
<td>Object remove(int index)</td>
</tr>
<tr>
<td>boolean removeAll(Collection c)</td>
</tr>
<tr>
<td>…</td>
</tr>
</tbody>
</table>

- No parameters
- Is “pure”
Identifying inspectors

<table>
<thead>
<tr>
<th>java.util.Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean add(Object o)</td>
</tr>
<tr>
<td>void clear()</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
</tr>
<tr>
<td>Object remove(int index)</td>
</tr>
<tr>
<td>boolean removeAll(Collection c)</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- No parameters
- Is “pure”
- Returns a value
Instrumenting bytecode

public class Vector {
    ...
    public void clear() {
        ...
    }
    ...
}
public class Vector {
    ...
    public void clear() {
        State entryState = new State();
        entryState.addElement("isEmpty()", isEmpty());

        ...
    }
    ...
}


public class Vector {
    public void clear() {
        State entryState = new State();
        entryState.addElement("isEmpty()", isEmpty());
        try {
            ...
        } finally {
            State exitState = new State();
            exitState.addElement("isEmpty()", isEmpty());
        }
    }
    ...
}
public class Vector {
    ...
    public void clear() {
        State entryState = new State();
        entryState.addElement(”isEmpty()”, isEmpty());
        try {
            ...
        }
        finally {
            State exitState = new State();
            exitState.addElement(”isEmpty()”, isEmpty());
            model.addTransition(entryState, exitState, “clear()”);
        }
    }
    ...
}
How common are inspectors?

Inspector methods:
- Checkstyle: 10%
- Jalopy: 21%
- HTMLParser: 34%
- Lucene: 21%
- JSMSEngine: 17%
- PMD: 12%
- JackSum: 8%
- ProGuard: 7%
How common are classes with inspectors?

![Bar chart showing the percentage of classes with at least one inspector for different tools.]

- **Checkstyle**: 15%
- **Jalopy**: 34%
- **HTMLParser**: 64%
- **Lucene**: 53%
- **JSMSEngine**: 50%
- **PMD**: 25%
- **JackSum**: 37%

Classes with at least one inspector
How common are classes with inspectors?

On average, 45% of a program’s classes contain at least one inspector.
Application: understanding behavior of a class

org.eclipse.jdt.internal.compiler.impl.BooleanConstant

booleanValue = true
stringValue : java.lang.String
toString : java.lang.String
typeID > 0
typeName : java.lang.String

booleanValue = false
stringValue : java.lang.String
toString : java.lang.String
typeID > 0
typeName : java.lang.String
Application: understanding behavior of a class

```
org.aspectj.asm.StructureModel
```

```
<starting state>

<init>

getFileMap = null
getRoot = null
isValid = false

setRoot

getFileMap = null
getRoot : org.aspectj.asm.ProgramElementNode
isValid = false

setFileMap

getFileMap : java.util.HashMap
getRoot : org.aspectj.asm.ProgramElementNode
isValid = true
```
Application: understanding behavior of a class

org.columba.ristretto.imap.IMAPProtocol
Other applications

- Checking behavior of objects with respect to models at runtime
- Finding possible model violations in a source code
- Suggesting method calls during development
Future work: mining models statically

```java
public void foo(Data data) {
    data.startExtracting();
    while (data.hasNext())
        process(data.next());
    data.endExtracting();
}
```
Mining models statically: challenges

• How to recognize object’s state?
  ```java
class Example {
    public void foo() {
      while (a.foo() || b.bar()) {
        // what is the state of a?
        ...
      }
    }
}
```

• How to merge models based on a control flow?
  ```java
public void foo(Data data) {
  ...
  foo(data);
  ...
}
```

• How can we deal with possible aliases? Points-to analysis?
  ```java
  a.foo(); b.bar();
  ...
```

• Can we combine models mined statically with those mined dynamically?
Conclusions

The problem

- `java.util.Vector`
  - boolean `add(Object o)`
  - void `clear()`
  - boolean `isEmpty()`
  - Object `remove(int index)`
  - boolean `removeAll(Collection c)`

How to use a class correctly?

What do object behavior models look like?

Object behavior model for the `java.util.Vector` class

How can we mine object behavior models?

How common are classes with inspectors?

Application: understanding behavior of a class

Future work: mining models statically