Mining framework concepts from application code

Steven She

University of Waterloo

Generative Software Developement Lab

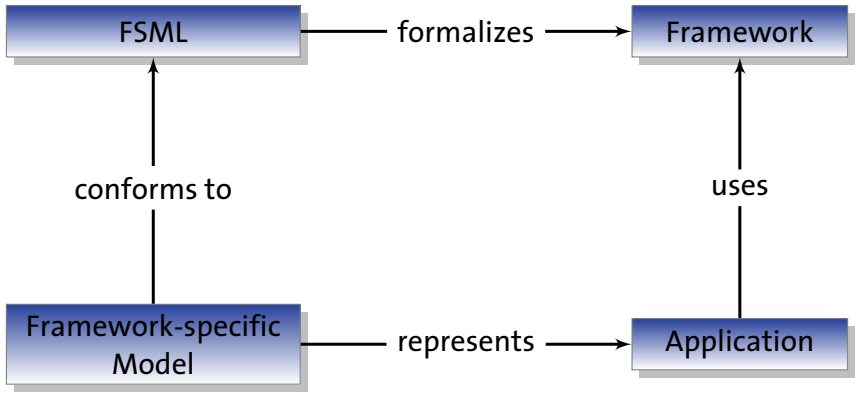
August 19, 2009

Outline

Problem - Java Applet

```
import java.applet.Applet;
import javax.swing.JApplet;
import java.awt.event.KeyListener;

public class HelloApplet extends JApplet {

    KeyListener keyList = new KeyListener() {
        public void keyTyped(KeyEvent e) { /* Do something */ }
        public void keyPressed(KeyEvent e) {}
        public void keyReleased(KeyEvent e) {}
    }

    public void init() {
        showStatus("Hello, world!");
        addKeyListener(keyList);
    }

    public void destroy() {
        removeKeyListener(keyList);
    }
}
```


Framework-Specific Modelling Languages



[o] M. Antkiewicz, T. Tonelli Bartolomei, K. Czarnecki

Applet FSML

Applet FSML
- [1..1] name (String) — **class** name
- **!**[1..1] extendsApplet — **extends** Applet
  - [0..1] extendsJApplet — **extends** JApplet
- [0..*] registersKeyListener — **calls** addKeyListener(...)
  - **!**⟨**1 - 1**⟩
  - [0..1] asThis
    - ...
  - [0..1] asAField — **parameter** is a field
    - [1..1] listenerField (String) — **field** name
    - [1..1] typedKeyListener — **typed** KeyListener(...)
    - [1..1] initialized — **assigned** an object
    - [1..1] deregisters — **calls** removeKeyListener(...)
- [0..*] showsStatus — **calls** showsStatus(...)
  - [1..1] message (String) — **parameter** value

Forward & Reverse

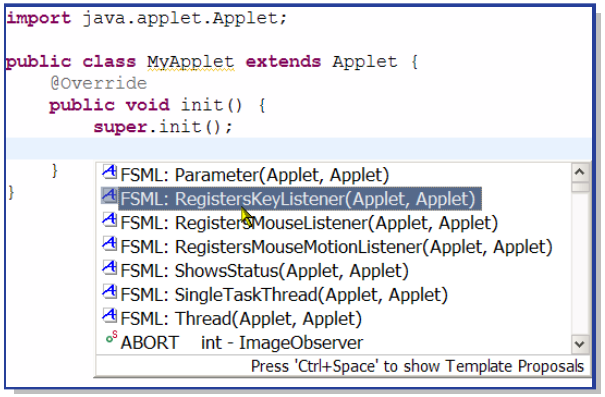
Model

Applet Instance
- name ← ``HelloApplet"
- extendsApplet ← ✓
  - extendsJApplet ← *x*
- registersKeyListener ← *x*
  - ⟨group⟩
  - asThis ← *x*
    - ...
  - asAField ← *x*
    - listenerField (String) ← *x*
    - typedKeyListener ← *x*
    - initialized ← *x*
    - deregisters ← *x*
- showsStatus ← ✓
  - message ← ``Hello, world!"

Code

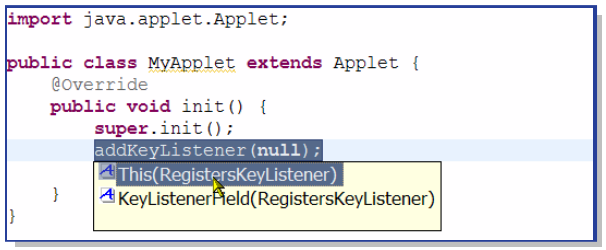
```
public class HelloApplet extends Applet {

  KeyListener keyList = new KeyListener() { … };

  public void init() {
    showStatus("Hello, world!");
    addKeyListener(keyList);
  }

  public void destroy() {
    removeKeyListener(keyList);
  }
}
```


Tool Support: Forward Eng.



[o] H. M. Lee. Model-guided Code Assistance for Framework Application Development.

Tool Support: Forward Eng. (cont.)



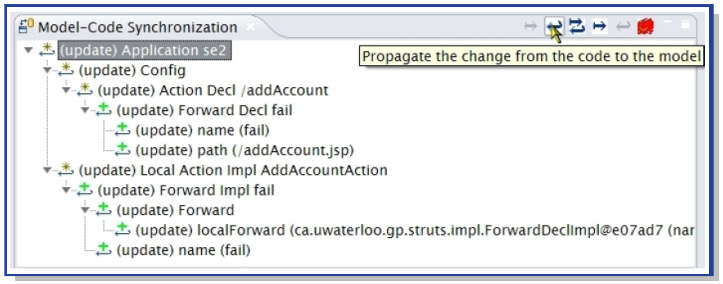
[o] H. M. Lee. Model-guided Code Assistance for Framework Application Development.

Tool Support: Forward Eng. (cont.)

```
import java.applet.Applet;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

public class MyApplet extends Applet implements KeyListener {
    @Override
    public void init() {
        super.init();
        addKeyListener(this);

    }

    public void destroy() {
        super.destroy();
        removeKeyListener(this);
    }

    public void keyTyped(KeyEvent keyEvent) {
    }

    public void keyPressed(KeyEvent keyEvent) {
    }

    public void keyReleased(KeyEvent keyEvent) {
    }
}
```

[o] H. M. Lee. Model-guided Code Assistance for Framework Application Development.

Tool Support: Reverse Engineering



[o] H. M. Lee. Model-guided Code Assistance for Framework Application Development.

Constructing FSMLs

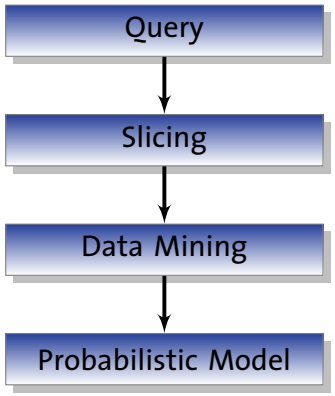
FSMLs are great!

But, how are they constructed?

- Creator must have an in-depth understanding of the framework.
- Examine tutorials and documentation.
- Examine boiler-plate code and example applications.

...currently an entirely **manual** process.

Mining for Framework Concepts



Provide hints for finding concept statements.

Find related statements.

Find usage patterns among multiple examples.

Output a probabilistic model describing usage.

Input Query

```
public class HelloWorldPresentation extends JApplet {
  public static final String HELLO_WORLD = "Hello World!";
  public HelloWorldPresentation() {
    Container contentPane = getContentPane();
    JLabel hwLabel = new JLabel(HELLO_WORLD);
    StringBuilder builder = new StringBuilder(HELLO_WORLD);
    Formatter fmt = new Formatter(builder);
    fmt.format("%s %s", HELLO_WORLD, HELLO_WORLD);
    contentPane.add(hwLabel);
  }
}
```

User selects lines of code in a framework application.

Concept Slicing

Reduce an example to a minimal set of statements that retains the specified behaviour.

Source code with Query

Program Slicer

Relevant Statements

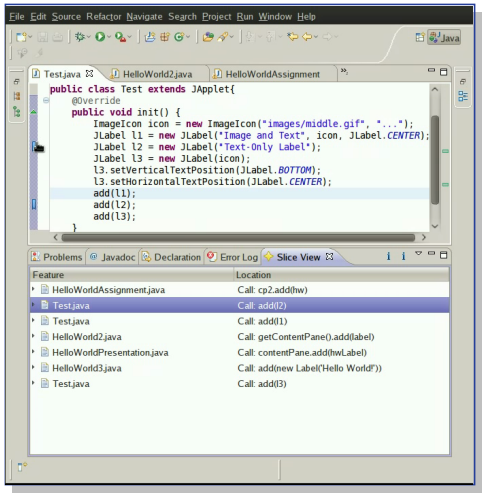
Concept Slicing

```
public class HelloApplet extends JApplet {
    KeyListener keyList = new KeyListener() { … }
    Image smiley;

    public void init() {
        smiley = getImage(getCodeBase(), "images/smile.gif");
        showStatus("Hello, world!");
        addKeyListener(keyList);
        play(getCodeBase(), "audio/woohoo.wav");
    }

    public void destroy() {
        removeKeyListener(keyList);
    }
}
```

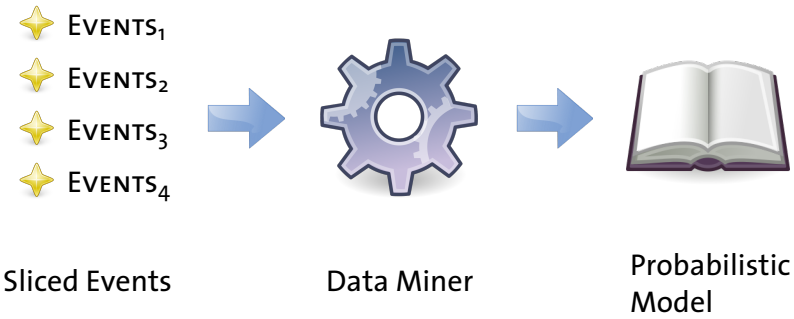

Sliced Statements

Example	Location	Statement
HelloApplet	Class	*field*:keyList (KeyListener)
HelloApplet	initializer	*assigns*:keyList = new KeyListener()
HelloApplet	init()	*calls*:addKeyListener(keyList)
HelloApplet	destroy()	*calls*:removeKeyListener(keyList)

Finding Related Usages



Reverse-Engineering the Model



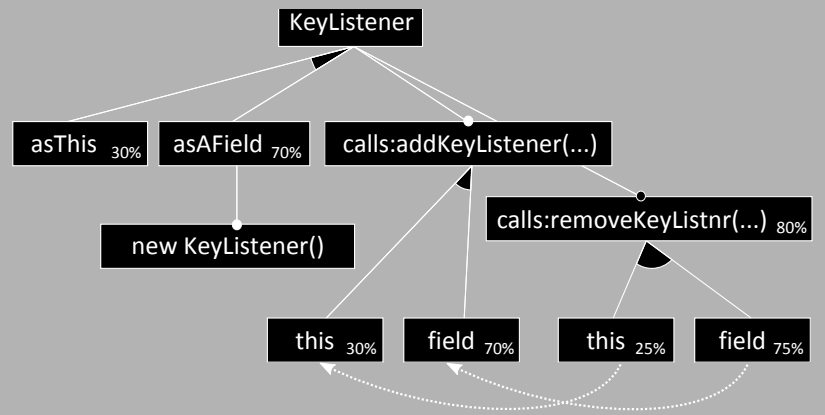
Feature Model Mining

Mine for probabilistic expressions from slices gathered from multiple applications.

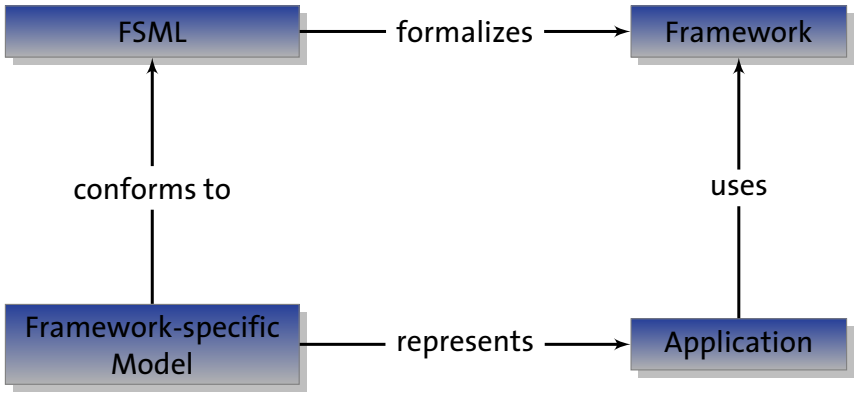
Association Rule	conf
KeyListener $\Rightarrow$ asThis $\vee$ asAField	100%
asAField $\Rightarrow$ $\neg$ asThis	100%
KeyListener $\Rightarrow$ asAField	70%
KeyListener $\Rightarrow$ asThis	30%
asAField $\Leftrightarrow$ new KeyListener()	100%

[o] K. Czarnecki, A. Wasowski, S. She.

Mined Feature Model

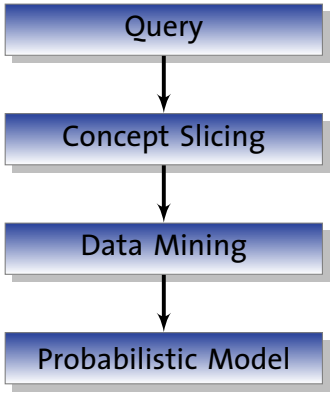


Framework-Specific Modelling Languages



[o] M. Antkiewicz, T. Tonelli Bartolomei, K. Czarnecki

Conclusions



- Constructs a **recipe** from a set of examples.
- **Slicing** finds related statements given a query.
- **Data mining** finds patterns in sliced statements to form a probabilistic model.