

Bi-directional Transformation for DSMLs and Code

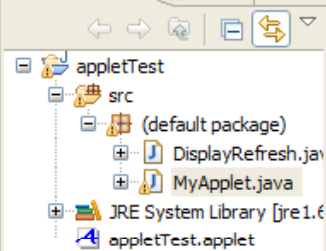
Michal Antkiewicz, Krzysztof
Czarnecki, Zinovy Diskin, Herman
Lee, Matthew Stephan

Framework-Specific Modeling Languages (FSMLs)

- FSMLs = DSMLs for framework APIs
- Used to express Framework Specific Models (FSMs)
- Use cases
 - Understand API
 - Understand and check application code (*reverse eng.*)
 - Generate application code (*forward eng.*)
 - Evolve application (*round-trip eng.*)
 - Migration application code to new API



Package Hierarchy



```

import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

public class MyApplet extends Applet implements MouseListener {

    public Runnable loadImages = new Runnable() {
        public void run() {
        }
    };

    public KeyListener keyListener = new KeyListener() {
        public void keyTyped(KeyEvent keyEvent0) {
        }

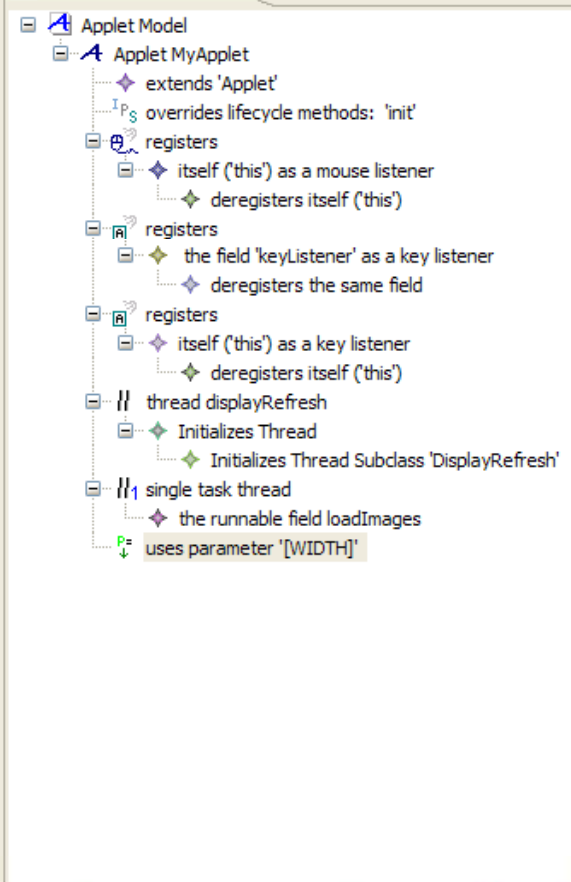
        public void keyPressed(KeyEvent keyEvent0) {
        }

        public void keyReleased(KeyEvent keyEvent0) {
        }
    };

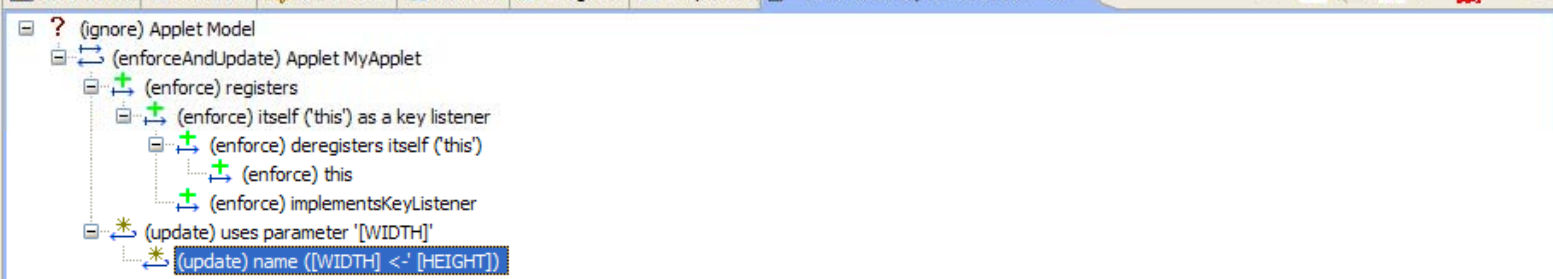
    public Thread displayRefresh = new DisplayRefresh();

    public void init() {
        addKeyListener(keyListener);
        getParameter("HEIGHT");
        new Thread(loadImages);
        addMouseListener(this);
    }
}
    
```

appletTest.applet



Problems Javadoc Declaration Console Progress Properties Model-Code Synchronization





Package Hierarchy

- appletTest
 - src
 - (default package)
 - DisplayRefresh.java
 - MyApplet.java
 - JRE System Library [jre 1.6]
 - appletTest.applet

```
public void run() {
}

public KeyListener keyListener = new KeyListener() {
    public void keyTyped(KeyEvent keyEvent0) {
    }

    public void keyPressed(KeyEvent keyEvent0) {
    }

    public void keyReleased(KeyEvent keyEvent0) {
    }
};

public Thread displayRefresh = new DisplayRefresh();

public void init() {
    addKeyListener(keyListener);
    getParameter("HEIGHT");
    new Thread(loadImages);
    addMouseListener(this);
}
```

appletTest.applet

- Applet Model
 - Applet MyApplet
 - extends 'Applet'
 - overrides lifecycle methods: 'init'
 - registers
 - itself ('this') as a mouse listener
 - deregisters itself ('this')
 - registers
 - the field 'keyListener' as a key listener
 - deregisters the same field
 - registers
 - itself ('this') as a key listener
 - deregisters itself ('this')
 - thread displayRefresh
 - Initializes Thread
 - Initializes Thread Subclass 'DisplayRefresh'
 - single task thread
 - the runnable field loadImages
 - uses parameter '[WIDTH]'

Model-Code Navig

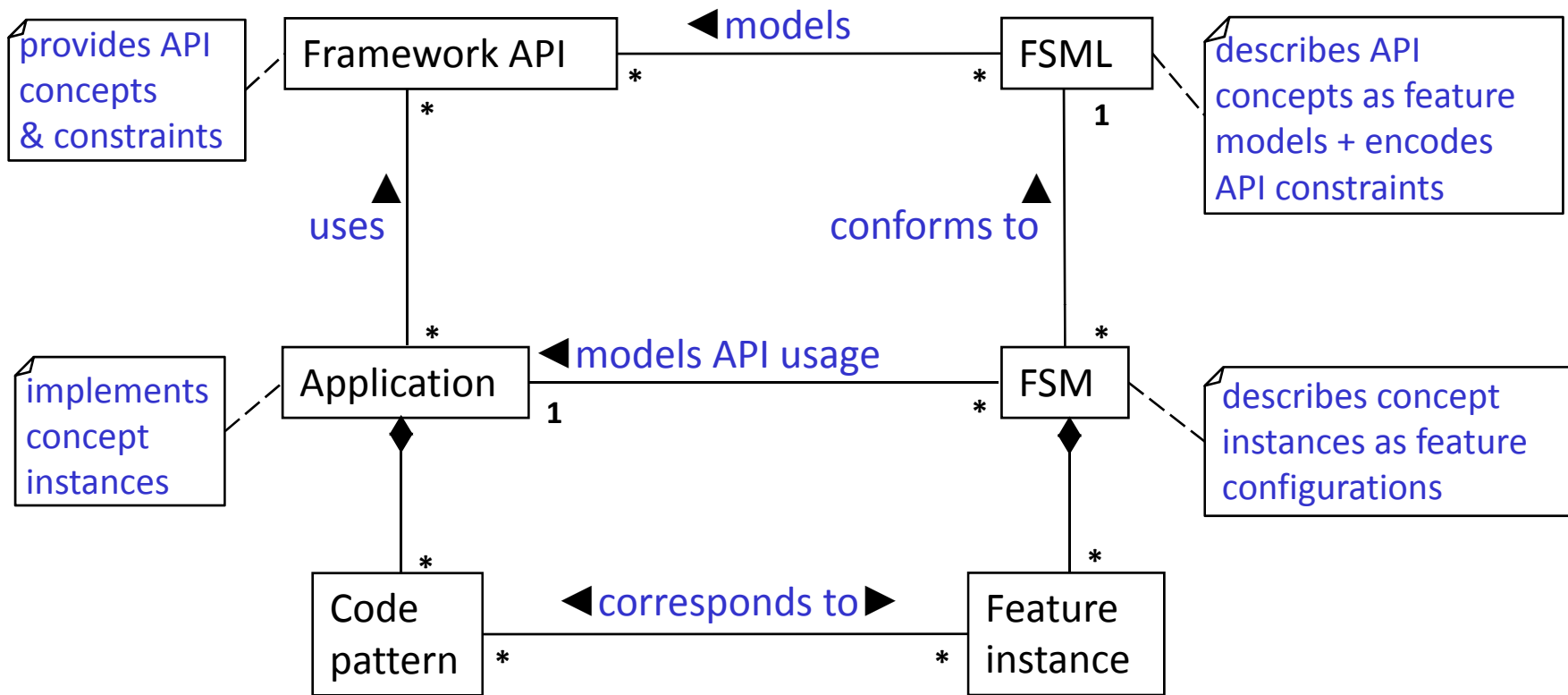
Parameter

Code completion dropdown menu:

- FSML: providesParameterInfo(Applet,EAttribute)
- FSML: Parameter(Applet,EReference)
- publ FSML: SingleTaskThread(Applet,EReference)
- FSML: Thread(Applet,EReference)
- FSML: RegistersKeyListener(Applet,EReference)
- FSML: RegistersMouseMotionListener(Applet,EReference)
- FSML: RegistersMouseListener(Applet,EReference)
- FSML: ShowsStatus(Applet,EReference)
- FSML: name(Applet,EAttribute)

Problems

- (ignore) Applet
- (enforce) registers
- (enforce) itself ('this') as a key listener
 - (enforce) deregisters itself ('this')
 - (enforce) this
- (enforce) implementsKeyListener
- (update) uses parameter '[WIDTH]'
- (update) name ([WIDTH] <' [HEIGHT])



framework-specific model

```
[0..*] Applet
[1] Applet
  [1] name ( `sun.WireFrame.ThreeD` )
  ![1] extendsApplet
  [0..*] parameter
  [1] parameter
    [1] name ( `model` )
  [1] parameter
    [1] name ( `scale` )
  [1] listensToMouse
  ![1] implementsMouseListener
  ![1] registers
  [1] deregisters
    [1] deregistersSameObject
    [1] registersBeforeDeregisters
```

application code

```
package sun.WireFrame;
...
public class ThreeD extends Applet
  implements Runnable, MouseListener, MouseMotionListener {
  ...
  mdname = getParameter("model");
  ...
  scalefudge = Float.valueOf(getParameter("scale")).floatValue();
  ...

  public class ThreeD extends Applet
    implements Runnable, MouseListener, MouseMotionListener {
  ...
  addMouseListener(this);
  ...
  removeMouseListener(this);
  ...
```

Framework-Specific Modeling Language

FSML
syntax

mapping
to code

```
[0..*] Applet
  [1] name (String)
  ![1] extendsApplet
  [0..*] parameter
    [0..1] name (String)
  [0..1] listensToMouse
  ![1] implementsMouseListener
  ![1] registers
  [1] deregisters

  [1] deregistersSameObject

  [1] registersBeforeDeregister
```

```
<class>
<fullyQualifiedName>
<assignableTo: 'Applet'>
<callsReceived: 'getParameter(String)'\>
<valueOfArg: 1>

<callsReceived: 'addMouseListener(Mous[...] '\>
<callsReceived: 'removeMouseListener(M[...] '\>

<argument:1 of call: ../../registers
  sameAsArg: 1 of call: ../../deregisters>

<methodCall: ../../../registers before:
  ../..>
```

Mapping definition and implementation

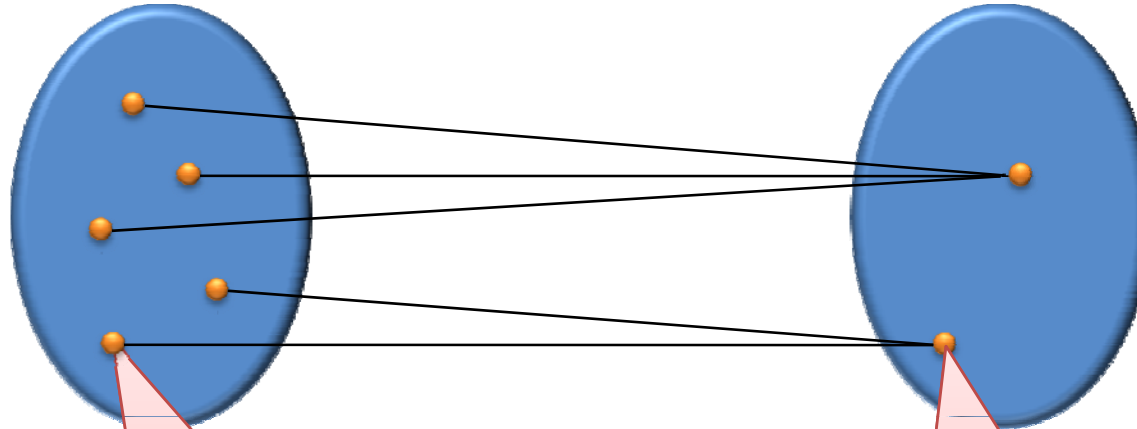
- Mapping (relation spec) defined by mapping types
- Structural and behavioural mapping types
 - Behavioural mapping types are pointcuts (e.g., cflow, dflow)
- Each mapping type has by default
 - Code query (get)
 - Code transformation (put)

Queries and trafos

- Approximations of behavioural mapping types
 - Precision and recall for queries
 - Potentially partial implementation by transformations
- Refinements through additional parameters for queries and transformations
 - Query – different precision
 - Trafo – e.g., additional control over location of additions

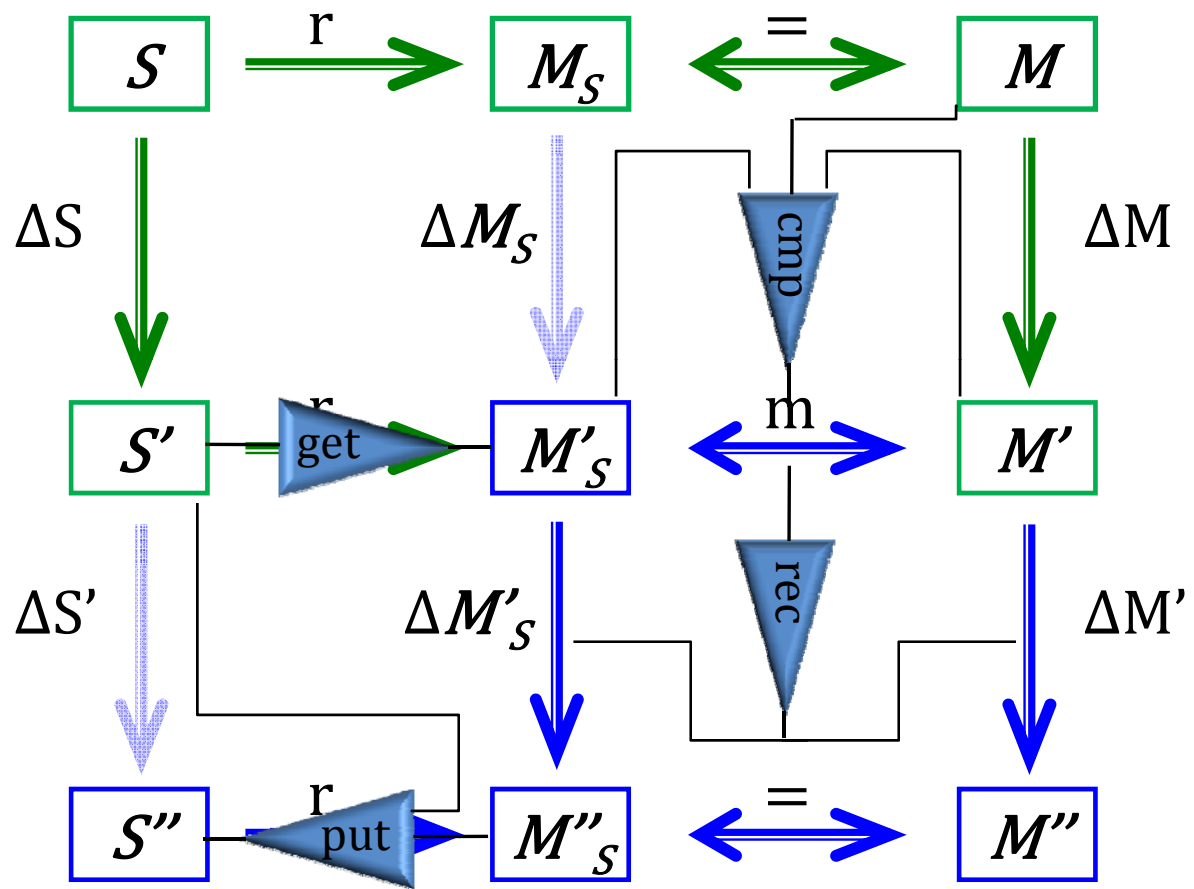
Java

Applet FSML



Java code
using Applet
framework

Applet-
specific
model



Update Reconciliation

- Three way compare
 - Recursive over containment structure
 - Use id defined based on code properties (key definitions)
- Different types of conflicts
- Review and propagate

Update Reconciliation

T'_T	S'_T	T_T	condition	detected updates to element
s	t	r	$t = s = r$	unchanged
s	t	r	$t = s \wedge t \neq r$	modified consistently in T'_T & S'_T
s	t	-	$t = s$	added consistently to T'_T & S'_T
s	t	r	$t \neq s \wedge t = r$	modified in S'_T
s	t	r	$t \neq s \wedge s = r$	modified in T'_T
s	t	r	$t \neq s \neq r \neq t$	modified inconsistently in T'_T & S'_T
s	t	-	$t \neq s$	added inconsistently to T'_T & S'_T
s	-	r	$t = r$	removed from S'_T
s	-	r	$t \neq r$	removed from S'_T , modified in T'_T
s	-	-	-	added to T'_T
-	t	r	$s = r$	removed from T'_T
-	t	r	$s \neq r$	removed from T'_T , modified in S'_T
-	t	-	-	added to S'_T
-	-	r	-	removed from T'_T & S'_T

Characterization

- Bidirectional DSLs for different artifact types to defining lenses
- Get and put given by mapping
- Mapping defined compositionally (over the metamodel)
- New mapping types can be added

Future Work

- Code removal and update
- More stable ids for code
- Formalization of the approach