On executable models that are integrated with program code

Marco Konersmann
package org.example.cashdesk;

@Stateful
public class CashDesk {
    @EJB
    IBarcodeScanner barcodeScanner;
    [...]
}
<table>
<thead>
<tr>
<th>Modeling Meta Model and Model</th>
<th>Program Code in Java</th>
</tr>
</thead>
</table>
| ![ComponentType](image1)    | public @interface ComponentType {
| name : String               |   String version();
| version : String            | } |
| ![ComponentType](image2)    | @ComponentType(version="1.0")
| name=BarcodeScanner         | public class BarcodeScanner {
| version=1.0                 |   Entry Point
|                              | } |
public @interface ComponentType {

}

@ComponentType
public class BarcodeScanner {

}

Entry Point
Mapping

https://codeling.de
https://codeling.de/language-integrator

[Kon18]
Meta Model and Model

Program in the Programming Language Java

```
public interface State {
    String name;
}

public class Ready implements State {
    private String name = "Ready";
}
```

Entry Point

Meta Model and Model

Program in the Programming Language Java

```java
@Retention(RetentionPolicy.RUNTIME)
public @interface Transition {
    
}
```

```java
@Transition(name=Ready)
public void scanCode() {
    
}
```

Entry Point
public class Ready implements State {

    @Transition(target = WithinSale.class)
    public void scanCode() {
        // ...
    }

}
@ComponentType

public class CashDesk {

    @StateMachine
    CashDeskStateMachine cashDeskStateMachine;

    @Reference
    BarcodeScanner barcodeScanner;

    @Operations
    public void addItemToCart() { /* ... */ }

    @Operations
    public void checkout() { /* ... */ }

}
Specific Runtimes (Generated)

Integration Mechanisms

Ecore

Root
public class CashDesk {

    @StateMachine
    CashDeskStateMachine cashDeskStateMachine;

    @Reference
    BarcodeScanner barcodeScanner;

    @Operations
    public void addItemToCart() {/* */}

    @Operations
    public void checkout() {/* */}
}

public class CashDesk {

    @StateMachine
    CashDeskStateMachine cashDeskStateMachine;

    @Reference
    BarcodeScanner barcodeScanner;

    public void postConstruct() {
        System.out.println("Cash Desk is now ready.");
    }

    @Operations
    public void addItemToCart() {
        items.add(barcodeScanner.scanItem());
        StateMachineRuntime<CashDeskStateMachine> smr =
            Runtimes.getInstance().get(cashDeskStateMachine);
        smr.executeTransition("scanCode");
    }

    ...
}

Discussion

- Codeling expresses models as program code - bidirectionally
- Assumed an execution environment (e.g. JavaEE) – until now
- Integration Mechanisms as common ground for translations
- Extendable runtime classes generated for each model element
  - Good for maintenance and evolution
  - Bad for performance
- Currently only implemented for Java
Further Reading

• Paper in Post-Proceedings

• [Kon18]
  Marco Konersmann
  Explicitly Integrated Architecture - An Approach for Integrating Software Architecture Model Information with Program Code
  https://mkonersmann.de/perm/publications/Konersmann2018Codeling_PhD.pdf

• https://www.codeling.de

• https://www.codeling.de/language-integrator