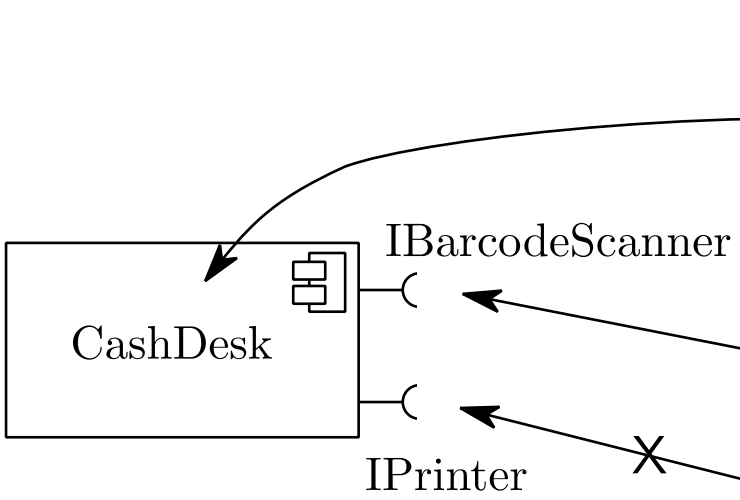
UNIVERSITÄT
DUISBURG
ESSEN

On executable models that are integrated with program code

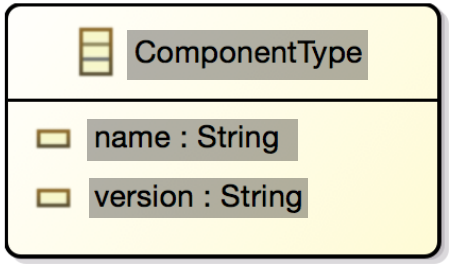
Marco Konersmann

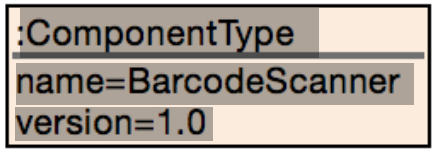
PALUNO
The Ruhr Institute for Software Technology

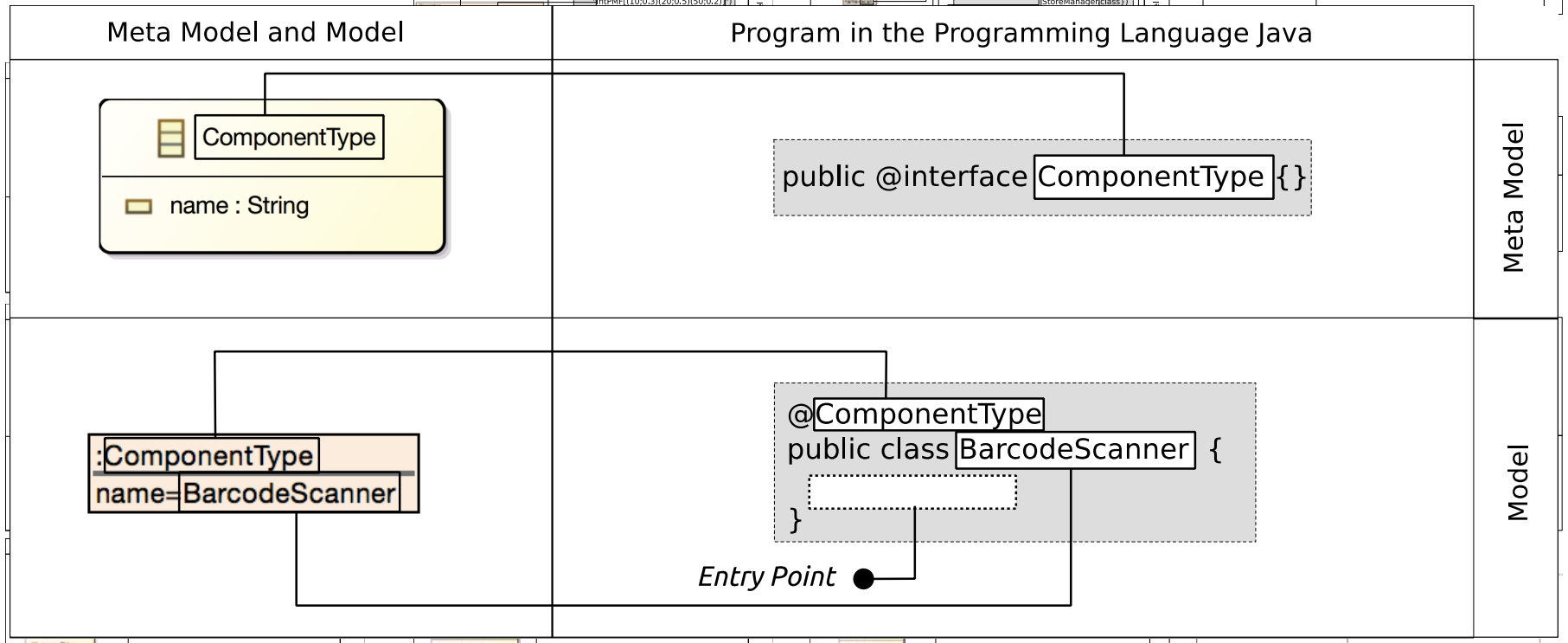


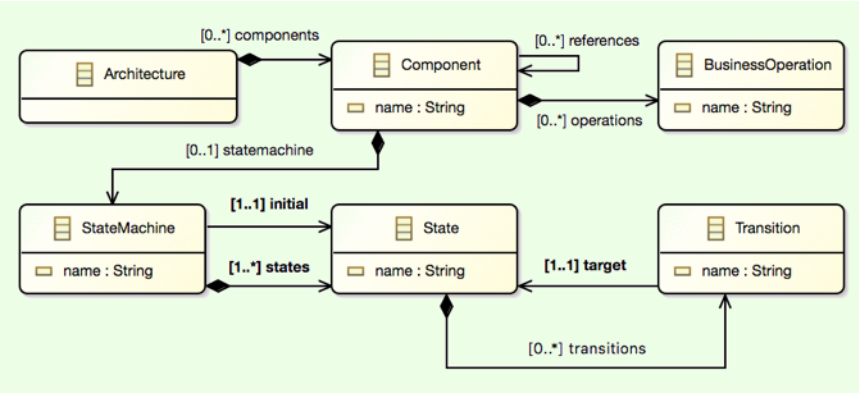

```
package org.example.cashdesk;

@Stateful
public class CashDesk {
  @EJB
  IBarcodeScanner barcodeScanner;

  [...]
}
```

Modeling Meta Model and Model	Program Code in Java	

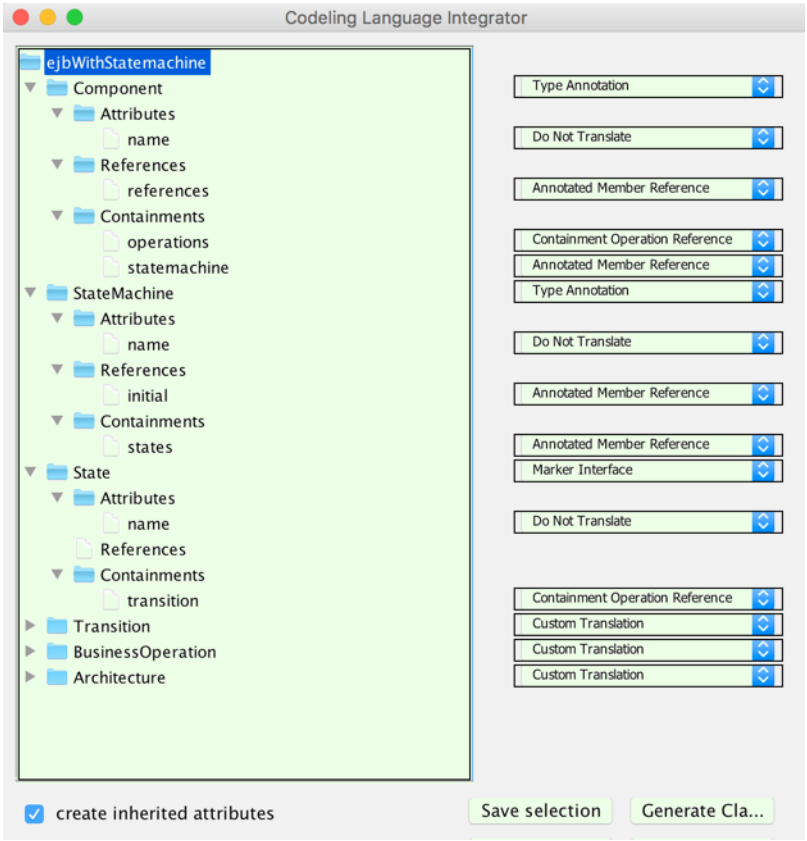
	public @interface ComponentType {
	String version();
	}
		Meta Model Level

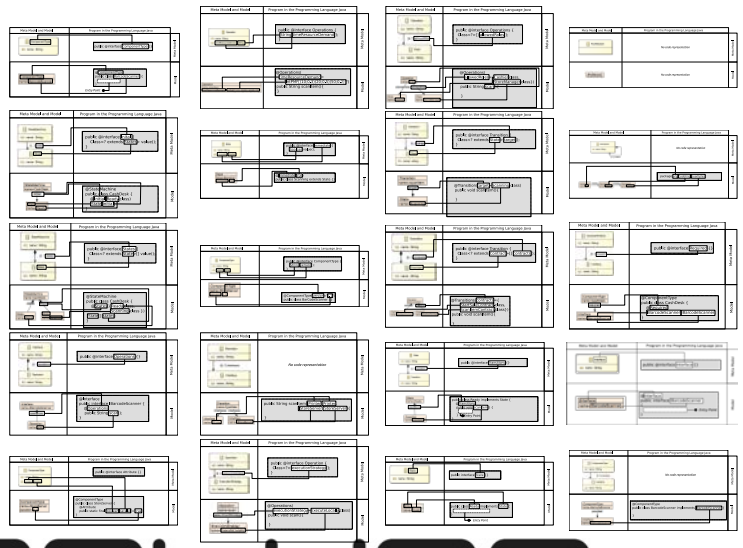
	@ComponentType(version="1.0")
	public class BarcodeScanner {
	}
	Entry Point
		Model Level


Meta Model and Model
Program in the Programming Language Java
ComponentType
name : String
public @interface ComponentType {}
Meta Model
:ComponentType
name=BarcodeScanner
@ComponentType
public class BarcodeScanner {
}
Entry Point
Model



Mapping

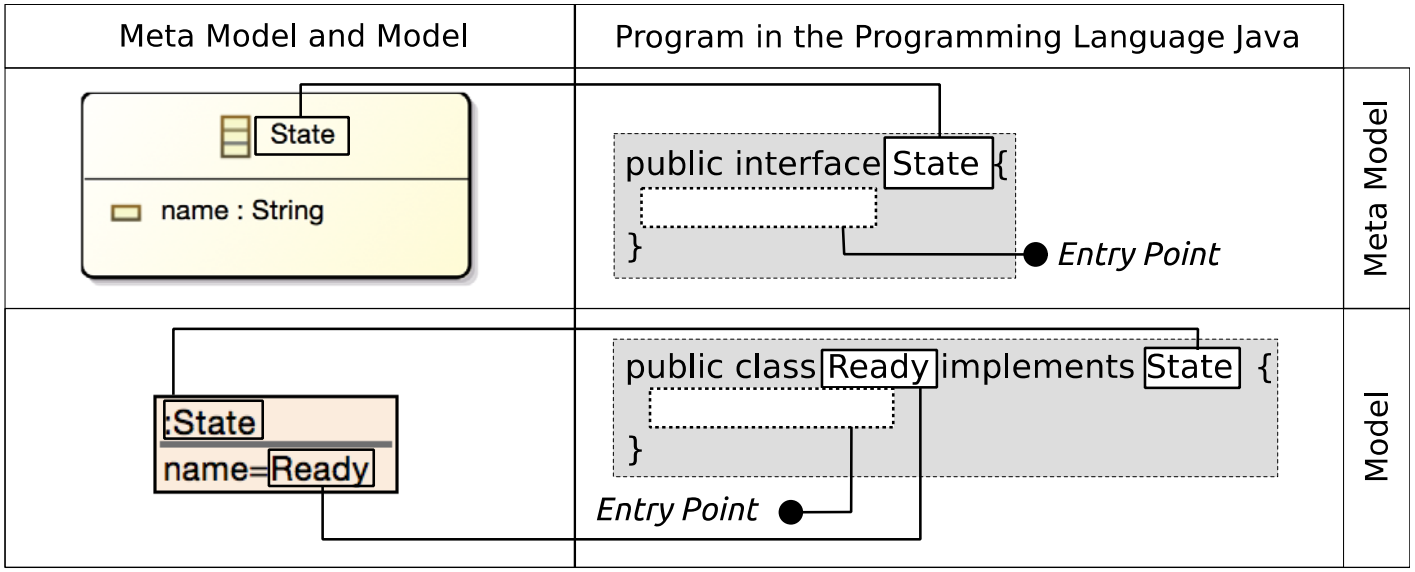


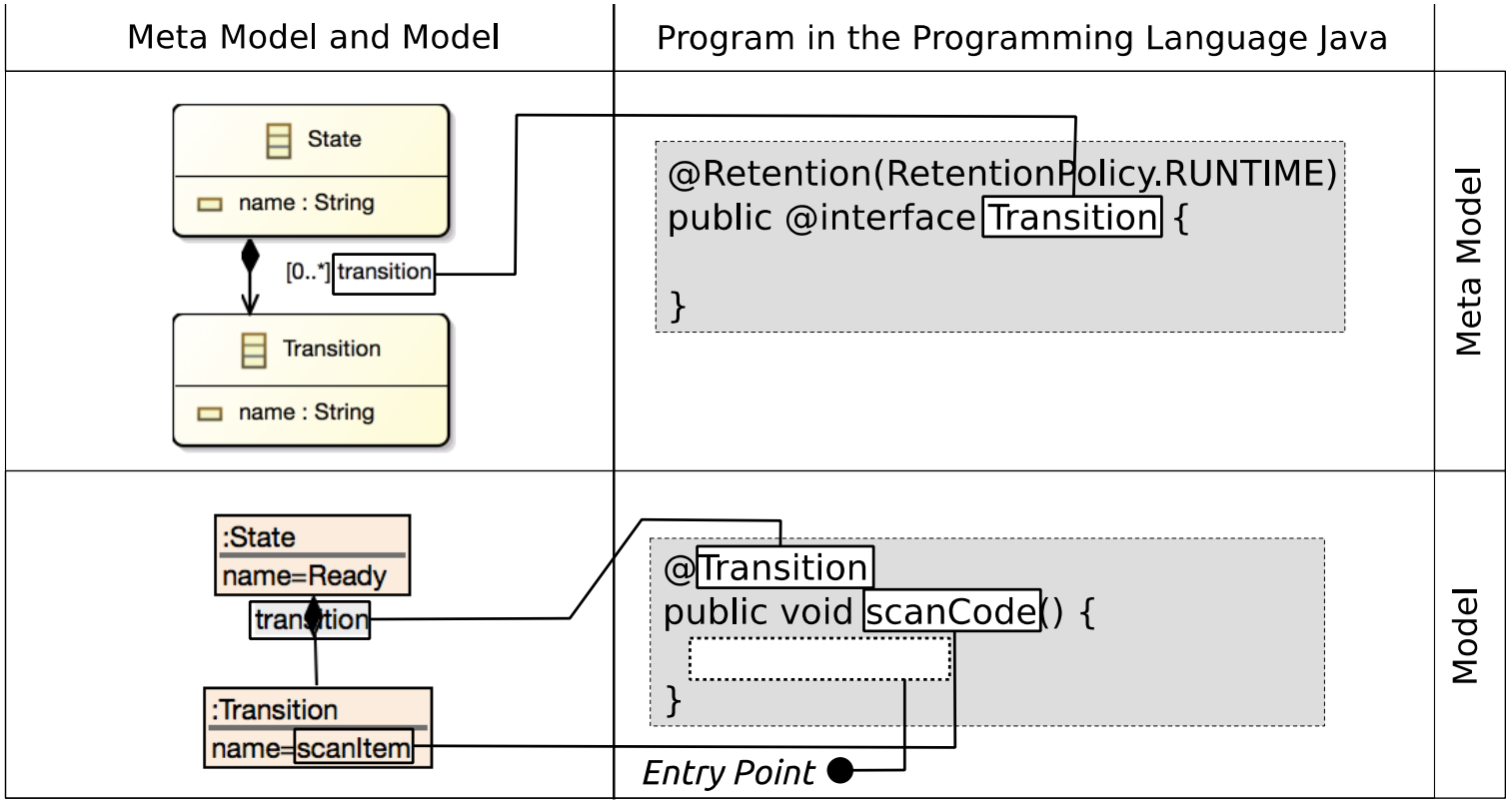


https://codeling.de

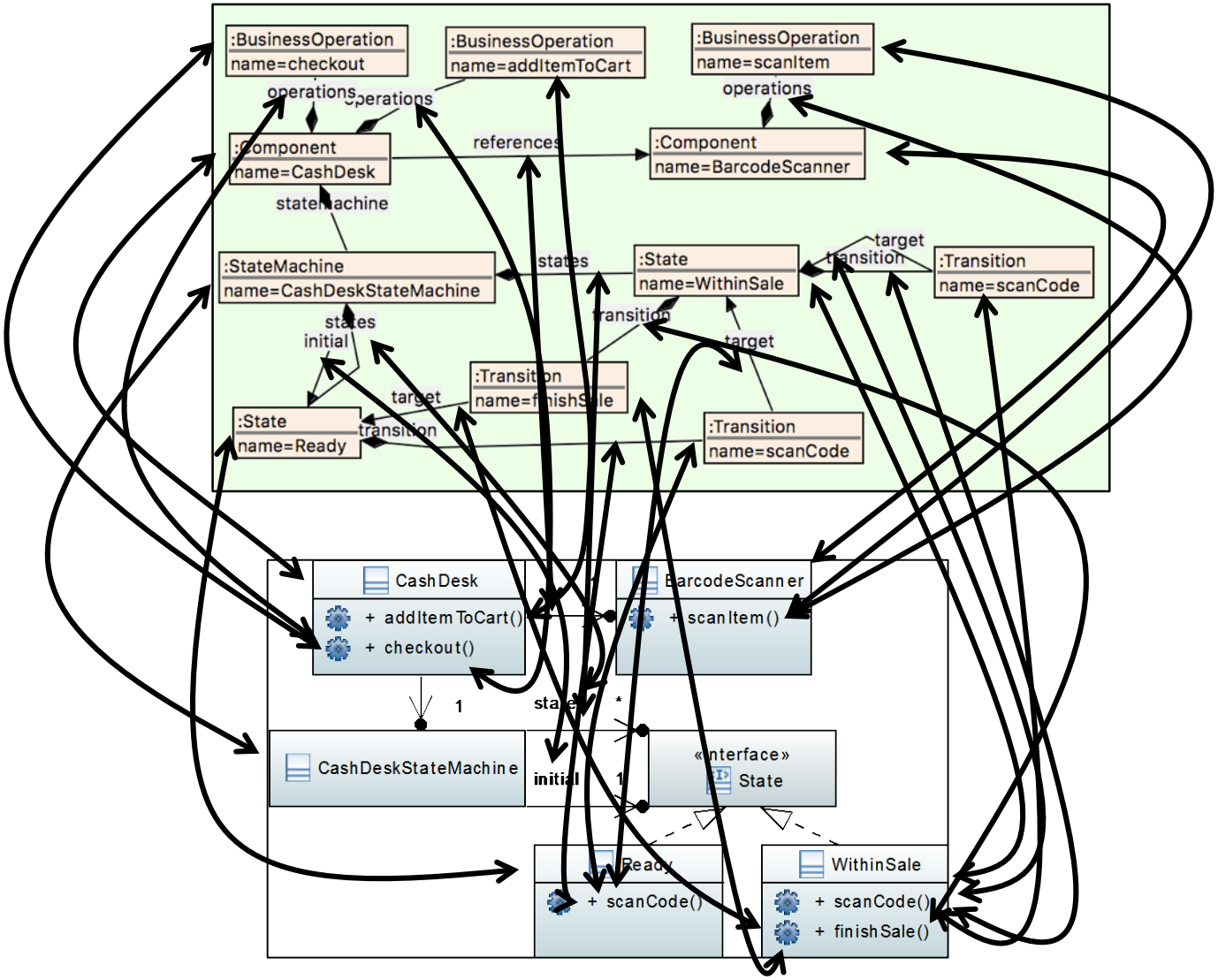
https://codeling.de/language-integrator

[Kon18]

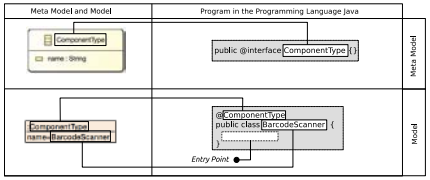

Meta Model and Model
Program in the Programming Language Java
State
name : String
public interface State {
}
Entry Point
Meta Model
:State
name=Ready
public class Ready implements State {
}
Entry Point
Model

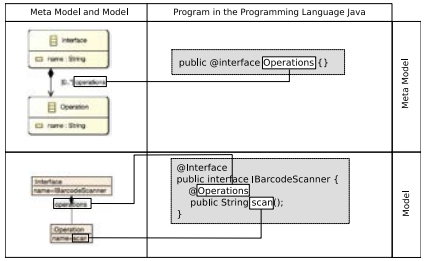

Meta Model and Model
Program in the Programming Language Java
State
name : String
[0..*] transition
Transition
name : String
@Retention(RetentionPolicy.RUNTIME)
public @interface Transition {
}
Meta Model
:State
name=Ready
transition
:Transition
name=scanItem
@Transition
public void scanCode() {
}
Entry Point
Model

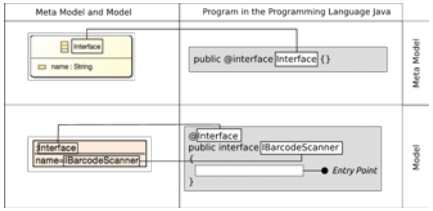
```
public class Ready implements State {

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

}
```


:BusinessOperation
name=checkout
:BusinessOperation
name=addItemToCart
:BusinessOperation
name=scanItem
operations
operations
operations
:Component
name=CashDesk
references
:Component
name=BarcodeScanner
stateMachine
:StateMachine
name=CashDeskStateMachine
states
:State
name=WithinSale
target
transition
:Transition
name=scanCode
transition
target
states
initial
:Transition
name=finishSale
target
:State
name=Ready
transition
:Transition
name=scanCode
CashDesk
+ addItemToCart()
+ checkout()
BarcodeScanner
+ scanItem()
1
state
*
CashDeskStateMachine
initial
1
«interface»
State
Ready
+ scanCode()
WithinSale
+ scanCode()
+ finishSale()

```
@ComponentType
public class CashDesk {

    @Statemachine
    CashDeskStateMachine cashDeskStateMachine;


    @Reference
    BarcodeScanner barcodeScanner;


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


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

}
```







ReferencesRuntime

ComponentTypeRuntime

OperationsRuntime

BusinessOperationRuntime

StateMachineRuntime

StateMachineRuntime

StatesRuntime

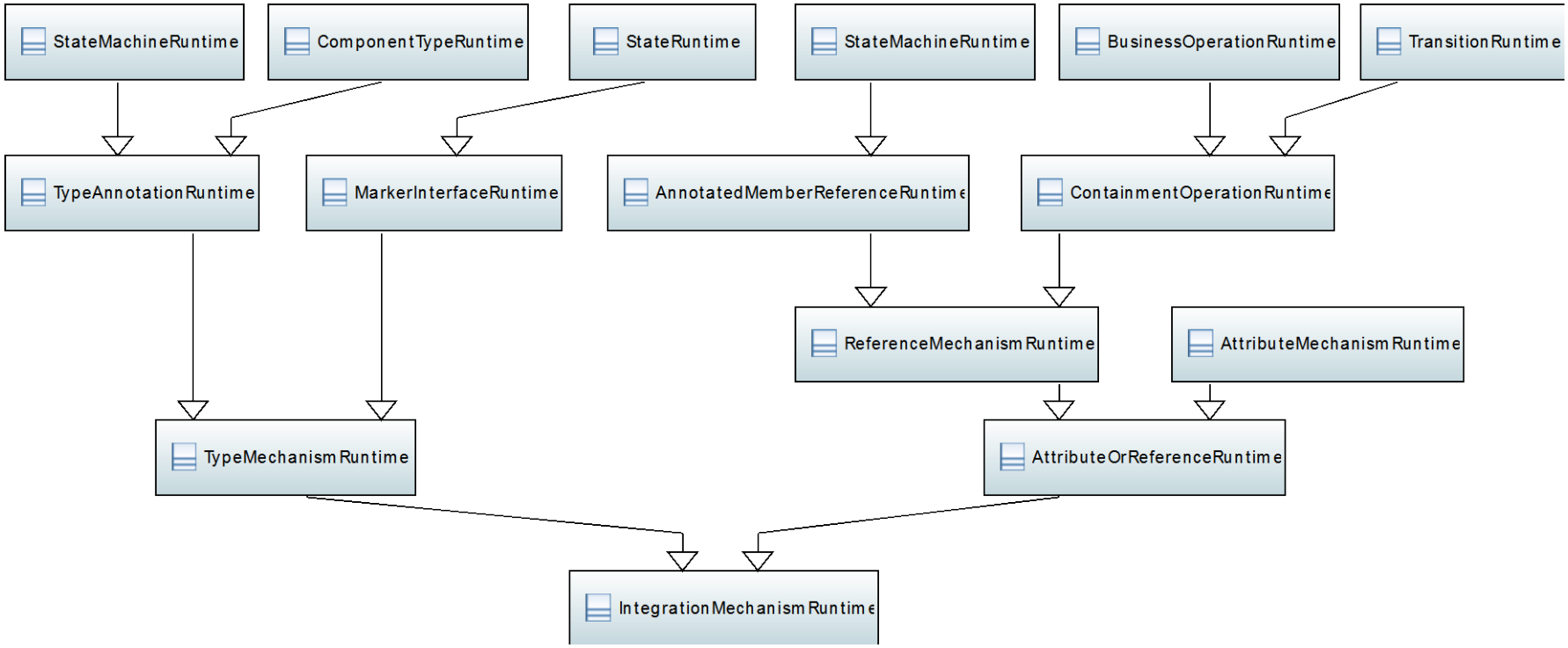
InitialRuntime

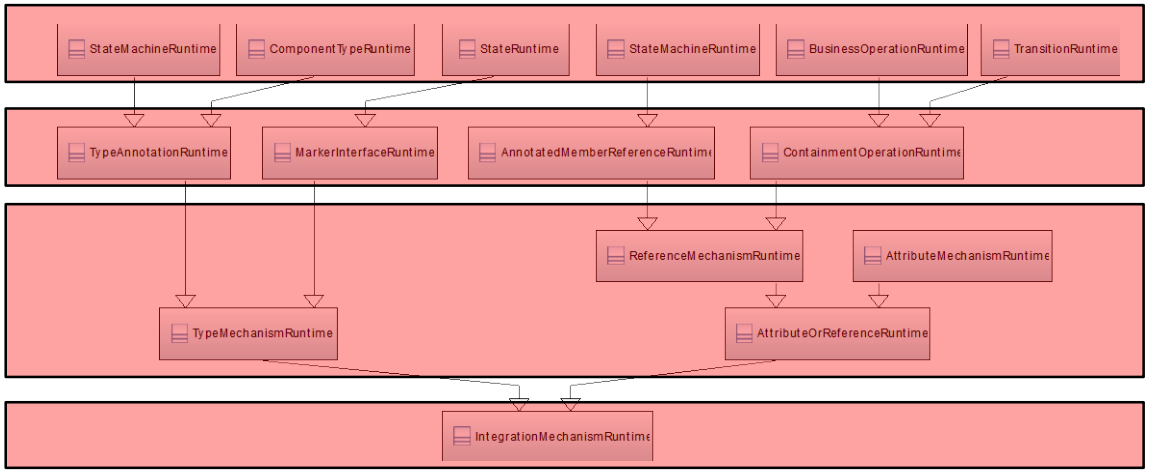
StateRuntime

TransitionsRuntime

TargetRuntime

TransitionRuntime


StateMachineRuntime
ComponentTypeRuntime
StateRuntime
StateMachineRuntime
BusinessOperationRuntime
TransitionRuntime
TypeAnnotationRuntime
MarkerInterfaceRuntime
AnnotatedMemberReferenceRuntime
ContainmentOperationRuntime
ReferenceMechanismRuntime
AttributeMechanismRuntime
TypeMechanismRuntime
AttributeOrReferenceRuntime
IntegrationMechanismRuntime



Specific Runtimes (Generated)

Integration Mechanisms

Ecore

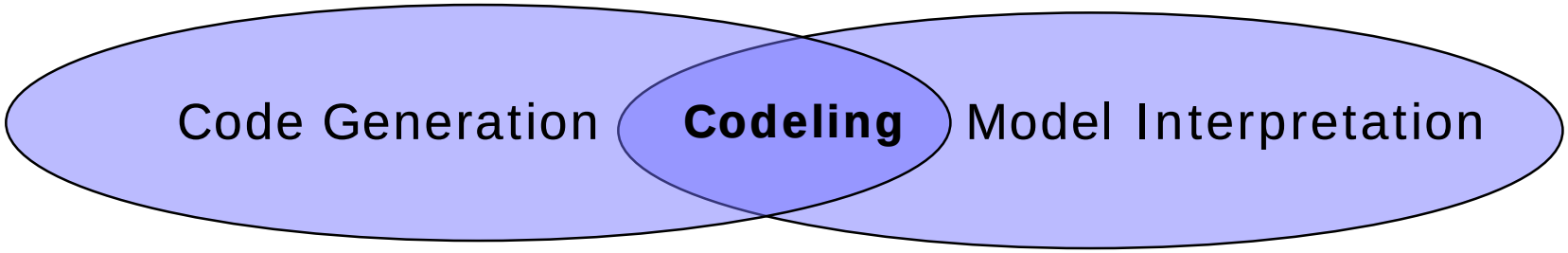
Root

```
@ComponentType
public class CashDesk {

  @Statemachine
  CashDeskStateMachine cashDeskStateMachine;

  @Reference
  BarcodeScanner barcodeScanner;

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

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

}
```

```
@ComponentType
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

  March 2018. PhD Thesis

  https://mkonersmann.de/perm/publications/Konersmann2018Codeling_PhD.pdf

- https://www.codeling.de

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