

# Model-Based Software Engineering

## Mini-Project I

Eric Roslin Wete Poaka  
Naira Audi

# Summary

1. Introduction
2. Ecore Model
3. Xtext
4. Sirius
5. Conclusion

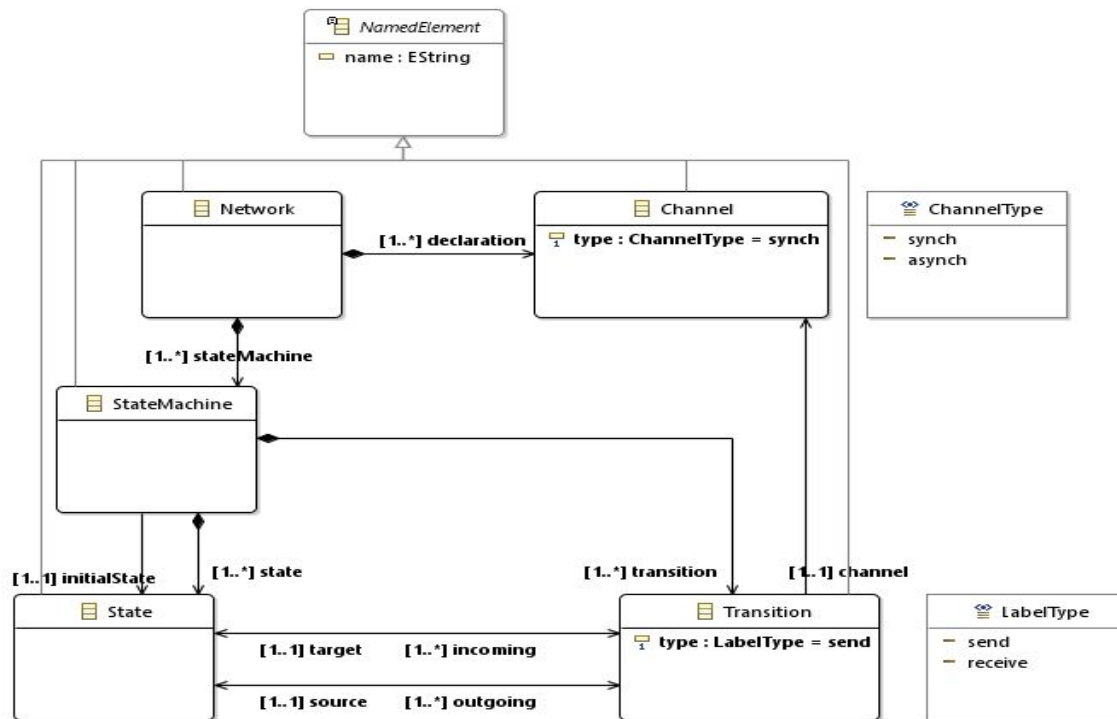
# Introduction

- Purpose:
  - Create an Ecore Metamodel for networks of state machines;
  - Define a textual syntax and build a textual editor using Xtext;
  - Define a graphical syntax and build a graphical editor using Sirius;
  - Model different networks in order to validate the syntax created.

# Introduction

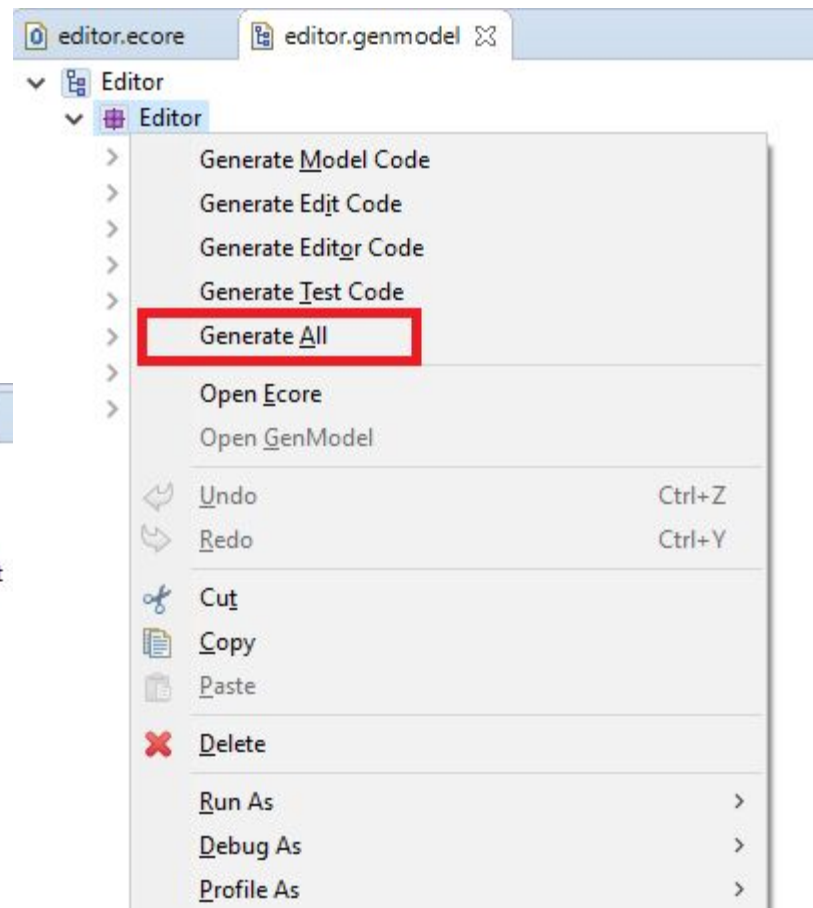
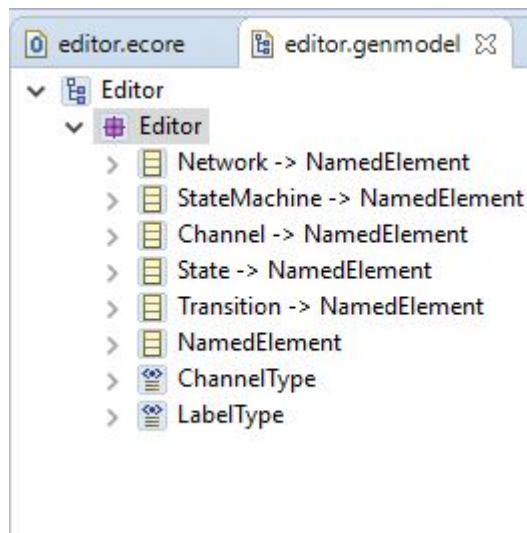
- Examples of state machines:
  - a guest ordering a coffee in a Cafe;
  - a pedestrian pressing a button to cross the street;
  - a person selecting a movie to watch in a catalog;
  - a client withdrawing money from an ATM-machine;
  - ...

# Ecore Meta-Model



# Ecore Meta-Model

- Code generating: java files to be used as classes



# Ecore Meta-Model

- Constraints in .ecore file;
- Avoiding duplicate elements in the same context.

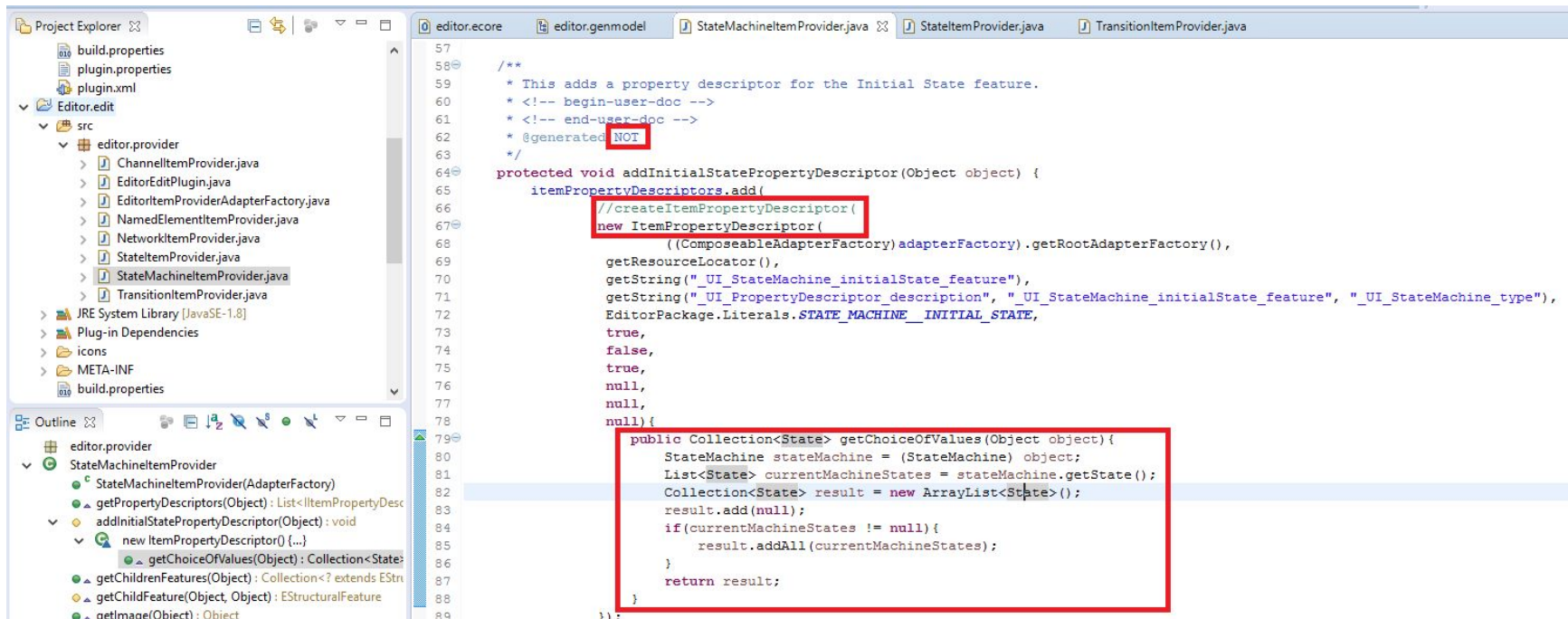
```

1 import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/' ;
2
3 package editor : editor = 'http://www.example.org/editor'
4 {
5     class Network extends NamedElement
6     {
7         property stateMachine : StateMachine[+] { ordered composes };
8         property declaration : Channel[+] { ordered composes };
9         invariant
10         NoDuplicatesDeclarationName('There are more than one channel of the network \'' + self.name + '\'' with de same name.'): self.declaration->forall(d1, d2 |
11             d1 <> d2 implies d1.name <> d2.name);
12         invariant
13         NoDuplicatesStateMachineName('There are more than one state machine of the network \'' + self.name + '\'' with de same name.'): self.stateMachine->forall(sm1,
14             sm2 | sm1 <> sm2 implies sm1.name <> sm2.name);
15     }
16     class StateMachine extends NamedElement
17     {
18         property state : State[+] { ordered composes };
19         property initialState : State[1];
20         property transition : Transition[+] { ordered composes };
21         invariant
22         NoDuplicatesStateName('There are more than one state of the state machine \'' + self.name + '\'' with de same name.'): self.state->forall(st1,
23             st2 | st1 <> st2 implies st1.name <> st2.name);
24     }
25     class Channel extends NamedElement
26     {
27         attribute type : ChannelType[1];
28     }

```

# Ecore

- Changes in .java files to show only own states in a state machine as possible initial state or source and target for transitions by creating dynamic instance.





# Xtext

- Automatically generated from Ecore Meta-Model;
- Customized.

```
42 Transition returns Transition:
43     'Transition'
44     type=LabelType
45     name=EString
46     '(' 'source' source=[State] ',' 'target' target=[State] ',' 'channel' channel=[Channel] ')';
```

# Xtext



The screenshot shows an IDE with two tabs: 'EDsl.xtext' and 'EDslValidator.xtend'. The 'EDslValidator.xtend' tab is active, displaying two Xtext validation rules. The first rule, '@Check checkNoDuplicatesDeclarationName', iterates over all declarations in a network and checks for duplicate names, throwing an error if found. The second rule, '@Check checkNoDuplicatesStateMachineName', iterates over all state machines in a network and checks for duplicate names, also throwing an error if found. Both rules use the 'error' method from 'EditorPackage.Literals' to report the violations.

```
29 @Check
30 def checkNoDuplicatesDeclarationName(Network network) {
31     for (channel1 : network.declaration) {
32         for (channel2 : network.declaration) {
33             if(!channel1.equals(channel2)){
34                 if(channel1.name.equals(channel2.name)){
35                     error('Two declarations of a network should not have the same name.',
36                         EditorPackage.Literals.NETWORK_DECLARATION
37                 );
38             }
39         }
40     }
41 }
42
44 @Check
45 def checkNoDuplicatesStateMachineName(Network network) {
46     for (stateMachine1 : network.stateMachine) {
47         for (stateMachine2 : network.stateMachine) {
48             if(!stateMachine1.equals(stateMachine2)){
49                 if(stateMachine1.name.equals(stateMachine2.name)){
50                     error('Two State Machine of a network should not have the same name.',
51                         EditorPackage.Literals.NETWORK_STATE_MACHINE
52                 );
53             }
54         }
55     }
56 }
```

Xtext  
validations to  
avoid two  
equivalent  
elements with  
the same name

# Xtext - Example Cafe

```

WatchMovies.edsl  Cafe.edsl  PedestrianCrossing.edsl
Network Cafe{
  declaration{
    Channel async orderCoffee ,
    Channel async deliverCoffee,
    Channel async payCoffee
  }
  stateMachine{
    StateMachine Guest{
      initialState waiting
      state{
        State waiting(incoming orderCoffee, payCoffee, outgoing orderCoffee, deliverCoffee),
        State drinkingCoffee (incoming deliverCoffee, outgoing payCoffee)
      }
      transition{
        Transition receive deliverCoffee (source waiting,target drinkingCoffee, channel deliverCoffee),
        Transition send payCoffee (source drinkingCoffee, target waiting, channel payCoffee),
        Transition send orderCoffee (source waiting,target waiting, channel orderCoffee)
      }
    },
    StateMachine Waiter{
      initialState waiting
      state{
        State waiting (incoming payCoffee, outgoing orderCoffee),
        State preparingCoffee(incoming orderCoffee, outgoing deliverCoffee),
        State waitingForPayment(incoming deliverCoffee, outgoing payCoffee)
      }
      transition{
        Transition receive orderCoffee (source waiting, target preparingCoffee, channel orderCoffee),
        Transition send deliverCoffee (source preparingCoffee, target waitingForPayment, channel deliverCoffee),
        Transition receive payCoffee (source waitingForPayment, target waiting, channel payCoffee)
      }
    }
  }
}

```

# Xtext - Example pedestrianCrossing

```

StateMachine CrossingControl(
    initialState initial
    state{
        State initial(incoming ready, outgoing pedestrianRequest),
        State pedestrianRequestReceived(incoming pedestrianRequest, outgoing trafficTurnRed),
        State turningTrafficLightsRed(incoming trafficTurnRed, outgoing trafficTurnedRed),
        State trafficLightsTurnedRed(incoming trafficTurnedRed, outgoing pedestrianTurnGreen),
        State turningPedestrianLightsGreen(incoming pedestrianTurnGreen, outgoing pedestrianTurnedGreen),
        State pedestrianLightsTurnedGreen(incoming pedestrianTurnedGreen, outgoing pedestrianTurnRed),
        State turningPedestrianLightsRed(incoming pedestrianTurnRed, outgoing pedestrianTurnedRed),
        State pedestrianLightsTurnedRed(incoming pedestrianTurnedRed, outgoing trafficTurnGreen),
        State turningTrafficLightsGreen(incoming trafficTurnGreen, outgoing trafficTurnedGreen),
        State trafficLightsTurnedGreen(incoming trafficTurnedGreen, outgoing ready)
    }
    transition{
        Transition receive pedestrianRequest(source initial, target pedestrianRequestReceived, channel pedestrianRequest),
        Transition send trafficTurnRed(source pedestrianRequestReceived, target turningTrafficLightsRed, channel trafficTurnedRed),
        Transition receive trafficTurnedRed(source turningTrafficLightsRed, target trafficLightsTurnedRed, channel trafficTurnedRed),
        Transition send pedestrianTurnGreen(source trafficLightsTurnedRed, target turningPedestrianLightsGreen, channel pedestrianTurnGreen),
        Transition receive pedestrianTurnedGreen(source turningPedestrianLightsGreen, target pedestrianLightsTurnedGreen, channel pedestrianTurnedGreen),
        Transition send pedestrianTurnRed(source pedestrianLightsTurnedGreen, target turningPedestrianLightsRed, channel pedestrianTurnRed),
        Transition receive pedestrianTurnedRed(source turningPedestrianLightsRed, target pedestrianLightsTurnedRed, channel pedestrianTurnedRed),
        Transition send trafficTurnGreen(source pedestrianLightsTurnedRed, target turningTrafficLightsGreen, channel trafficTurnedRed),
        Transition receive trafficTurnedGreen(source turningTrafficLightsGreen, target trafficLightsTurnedGreen, channel trafficTurnedRed),
        Transition send ready(source trafficLightsTurnedGreen, target initial, channel ready)
    }
}

```

# Xtext - Example WatchMovie

```

WatchMovies.edsl
Network WatchMovies{
  declaration{
    Channel synch selectGender,
    Channel synch selectLanguage,
    Channel synch selectMovie,
    Channel asynch playMovie,
    Channel asynch endMovie
  }
  stateMachine{
    StateMachine Viewer{
      initialState initial
      state{
        State initial (incoming endMovie, outgoing selectGender),
        State chooseGender (incoming selectGender, outgoing selectLanguage),
        State chooseLanguage (incoming selectLanguage, outgoing selectMovie),
        State chooseMovie (incoming selectMovie, outgoing playMovie),
        State watchMovie (incoming playMovie, outgoing endMovie)
      }
      transition{
        Transition send selectGender (source initial, target chooseGender, channel selectGender),
        Transition send selectLanguage (source chooseGender, target chooseLanguage, channel selectLanguage),
        Transition send selectMovie (source chooseLanguage, target chooseMovie, channel selectMovie),
        Transition receive playMovie (source chooseMovie, target watchMovie, channel playMovie),
        Transition send endMovie (source watchMovie, target initial, channel endMovie)
      }
    }
  },

```

# Sirius

The image displays the Sirius Specification Editor interface, showing two views of a project.ode design.

**Left View (Tree Structure):**

- platform/resource/graphiceditor.design/description/project.ode design
  - State Machine Specification
    - State Machine Network View
      - State Machine Network Diagram
        - State Machine Network Layer
          - Declaration
            - Square light\_yellow
          - Network
            - Square white
          - Transition
            - Edge Style solid
            - Center Label Style 8
          - StateMachine
            - State
              - Ellipse light\_gray
                - Conditional Style [self=self.eContainer(editor:StateMachine).initialState/]
                  - Ellipse gray
              - Gradient white to light\_gray

**Right View (Detailed View):**

- Edge Creation New Transition
  - Source Edge Creation Variable source
  - Target Edge Creation Variable target
  - Source Edge View Creation Variable sourceView
  - Target Edge View Creation Variable targetView
  - Begin
    - Change Context [source.eContainer()]/
      - Create Instance editor.Transition
        - Set source
        - Set target

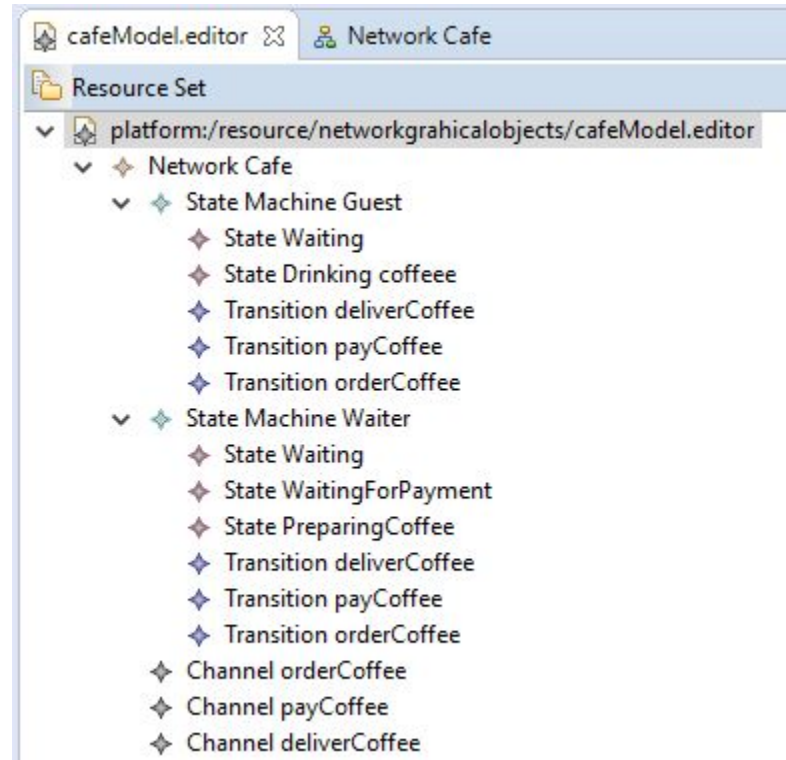
**Third View (Popup Menu State Machine Edit):**

- Popup Menu State Machine Edit
  - Operation Action Set as initial state
    - Container View Variable views
      - Expression Variable currentState
    - Begin
      - Change Context [thisEObject.eContainer()]/
        - Unset initialState
        - Set initialState

.ode design based  
on Ecore Meta-  
Model



# Sirius - Example Cafe: Dynamic instance



# Sirius - Example Cafe

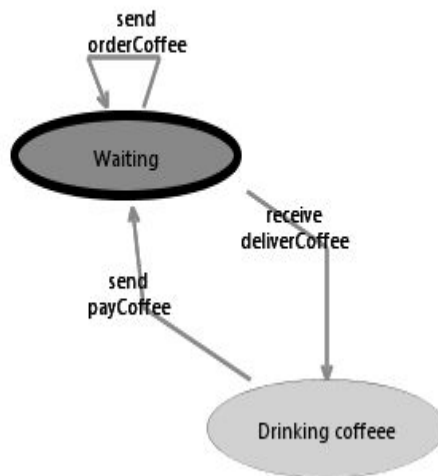
## Network Cafe

asynch orderCoffee

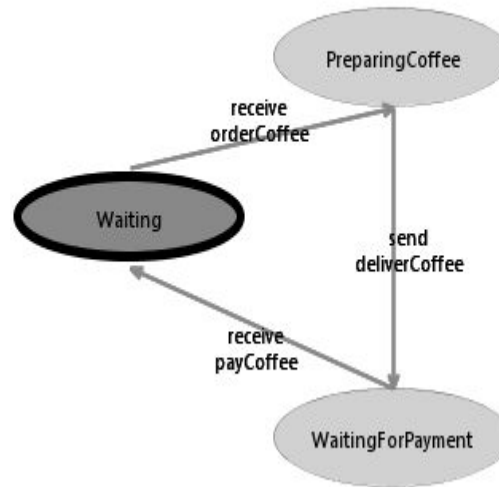
asynch payCoffee

asynch deliverCoffee

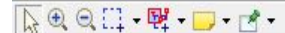
### statemachine Guest



### statemachine Waiter



## Palette

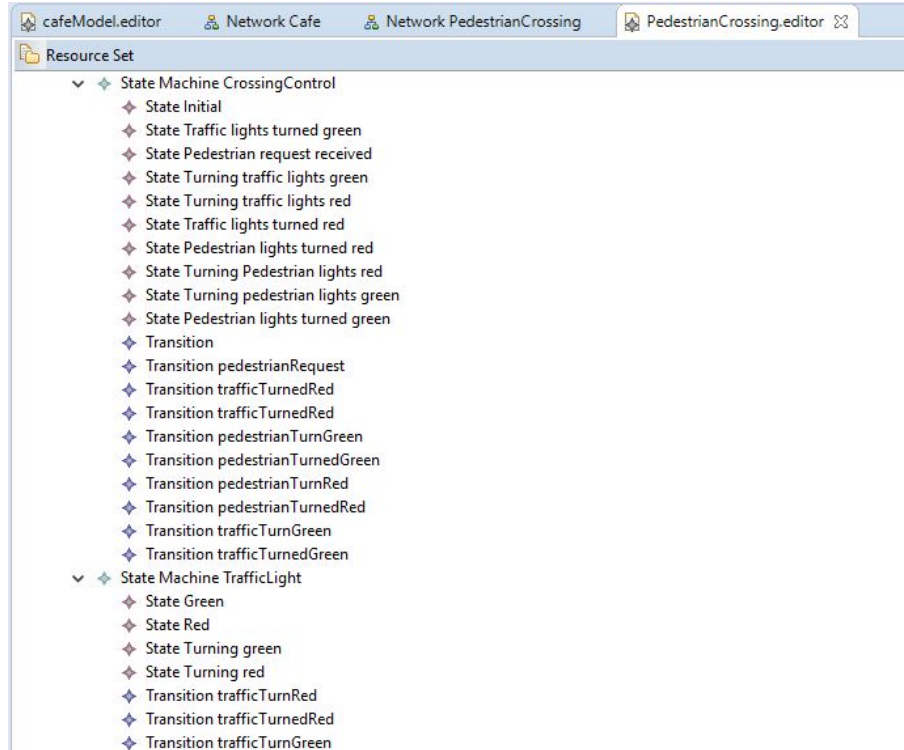


### Create Elements

- ◆ New Channel
- ◆ New State Machine
- ◆ New State
- ◆ New Transition



# Sirius - Example pedestrianCrossing: Dynamic instance

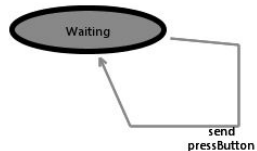


# Sirius - Example pedestrianCrossing

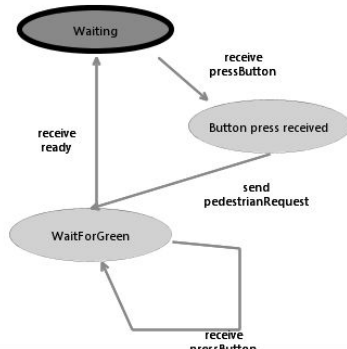
## Network PedestrianCrossing

synchron pressButton  
 asynchron pedestrianRequest  
 synchron pedestrianTurnGreen  
 synchron pedestrianTurnedGreen  
 synchron pedestrianTurnRed  
 asynchron ready  
 synchron pedestrianTurnedRed  
 synchron trafficTurnGreen  
 synchron trafficTurnedGreen  
 synchron trafficTurnRed  
 synchron trafficTurnedRed

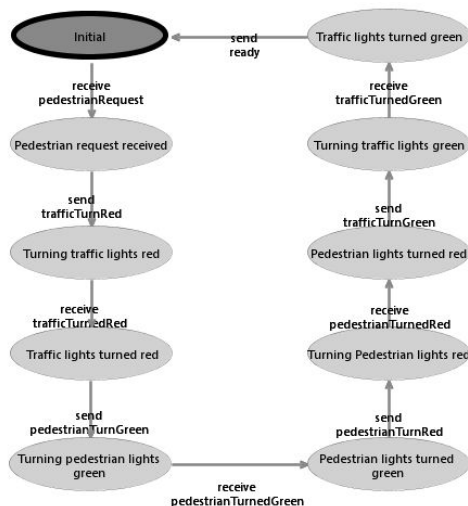
### stateMachine Pedestrian



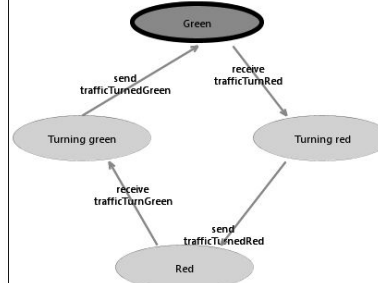
### stateMachine PedestrianUI



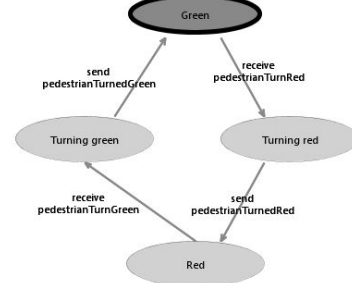
### stateMachine CrossingControl



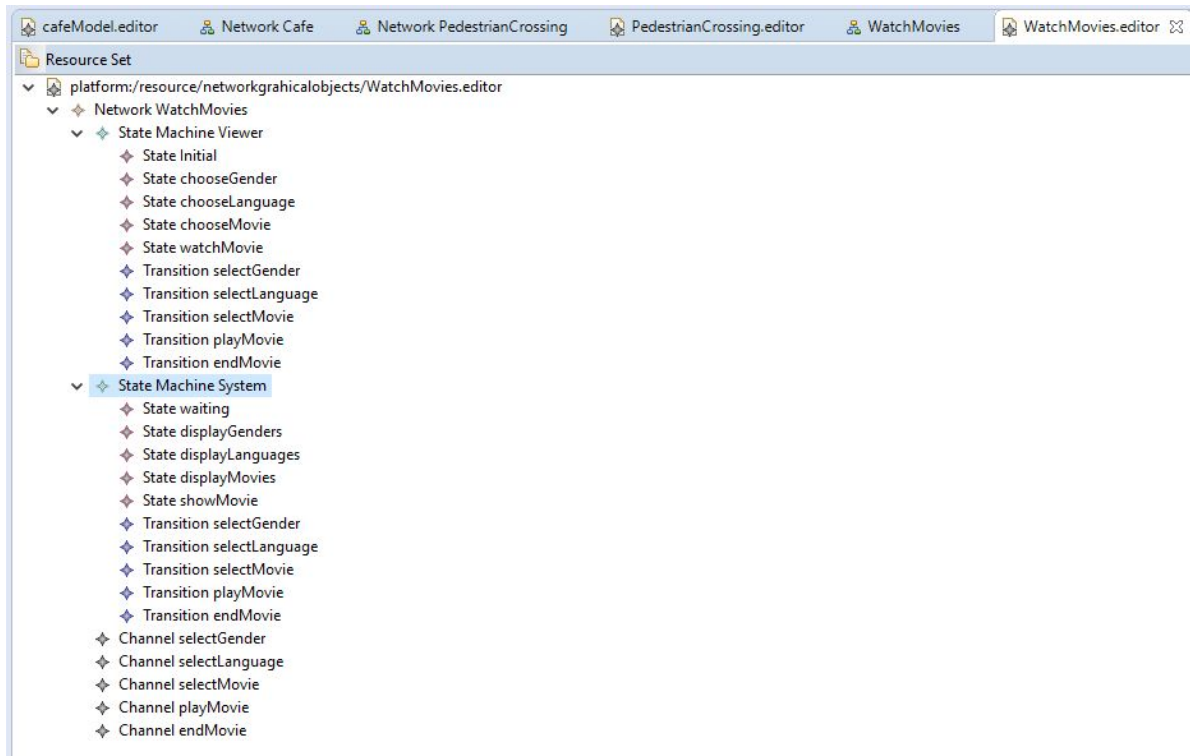
### stateMachine TrafficLight



### stateMachine PedestrianLight



# Sirius - Example WatchMovie: Dynamic instance



# Sirius - Example WatchMovie

## Network WatchMovies

asynch selectGender

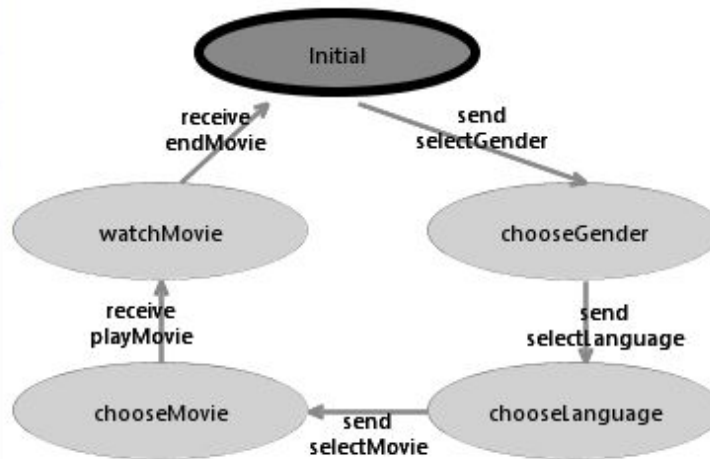
asynch selectLanguage

asynch selectMovie

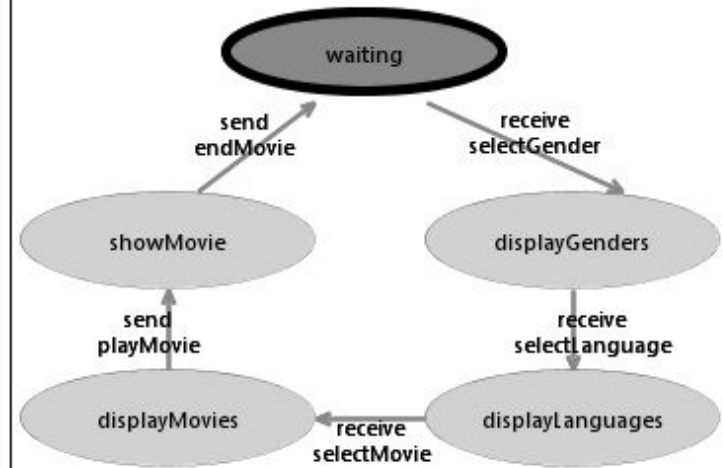
synch playMovie

synch endMovie

### statemachine Viewer



### statemachine System



# Conclusion

- Integrated EMF tools;
- We could not find a way to make some features:
  - print part of the label as bold and the other part as normal;
  - curved arrows.
- Some others could be made, but are not intuitive, such as the restriction of choices in a dropdown list while creating dynamic instance.

# Thank you for your attention!

Do you have any questions?