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

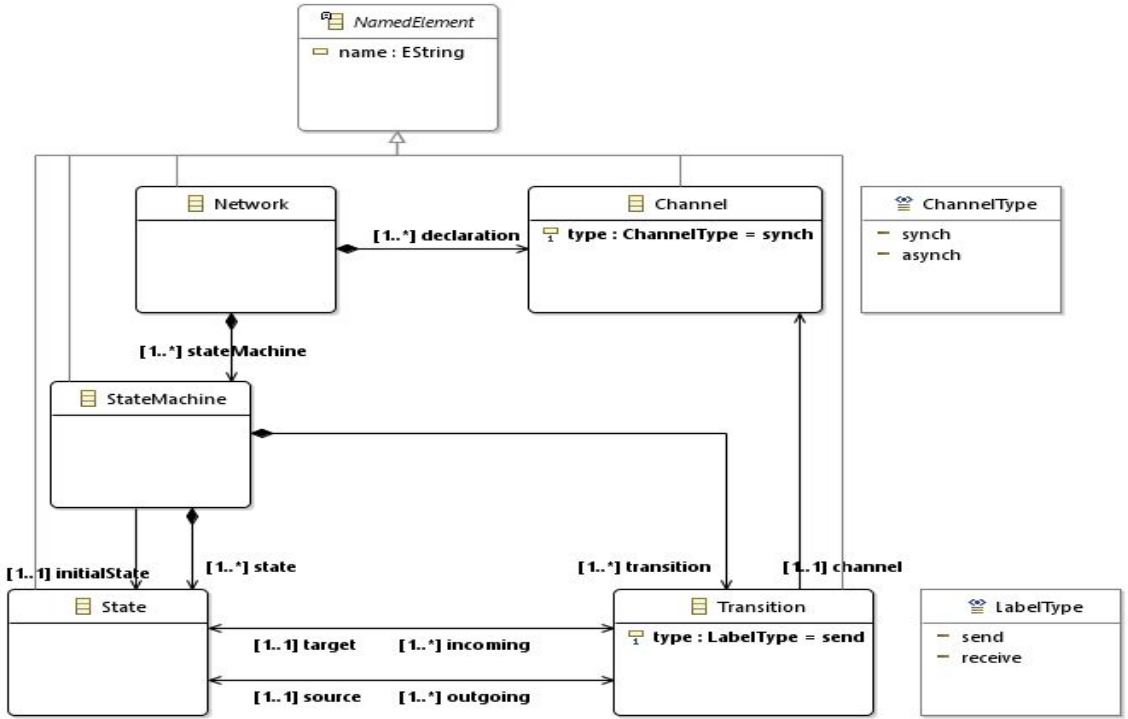
Introduction

- 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

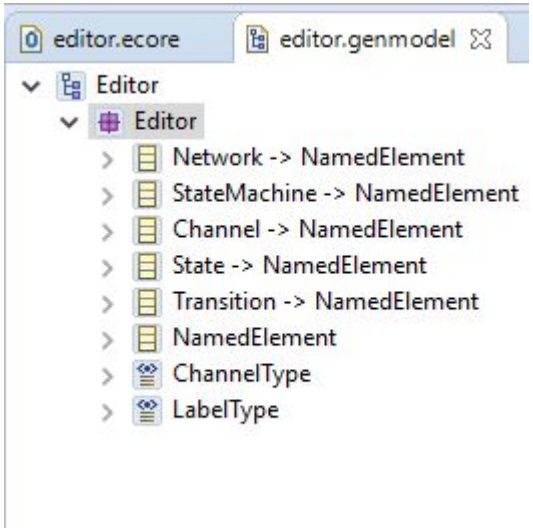
- Examples of state machines:
  - a guest ordering a coffee in a Cafe;
  - a pedestrian pressing a button to cross the street;
  - a person selecting a movie to watch in a catalog;
  - a client withdrawing money from an ATM-machine;
  - …

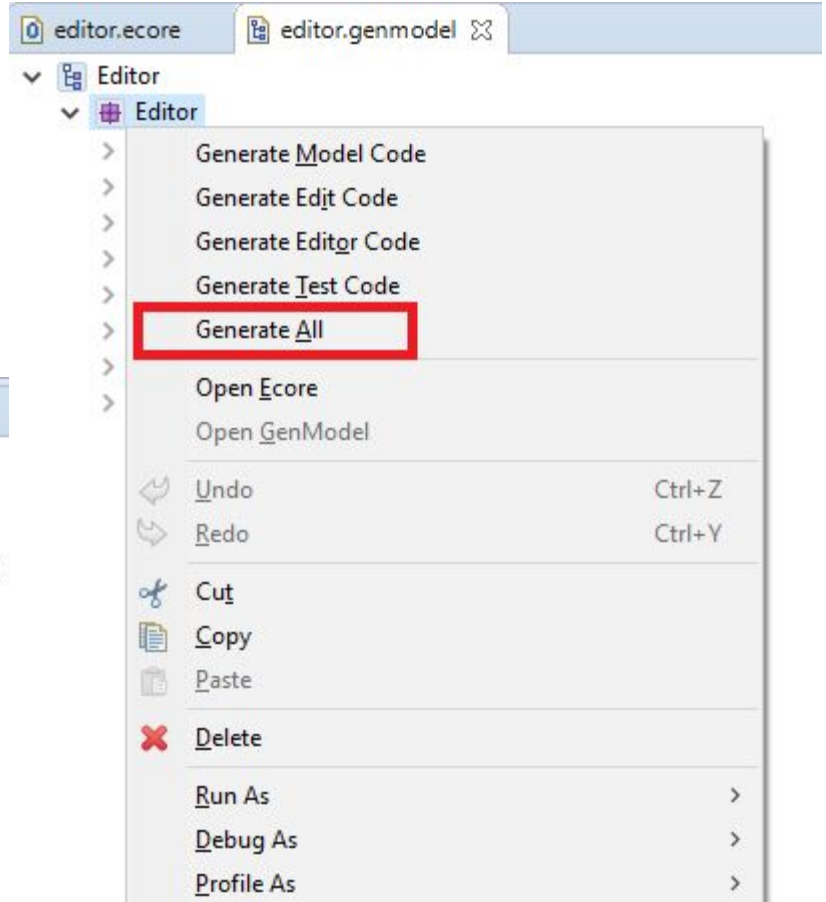
Ecore Meta-Model



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.

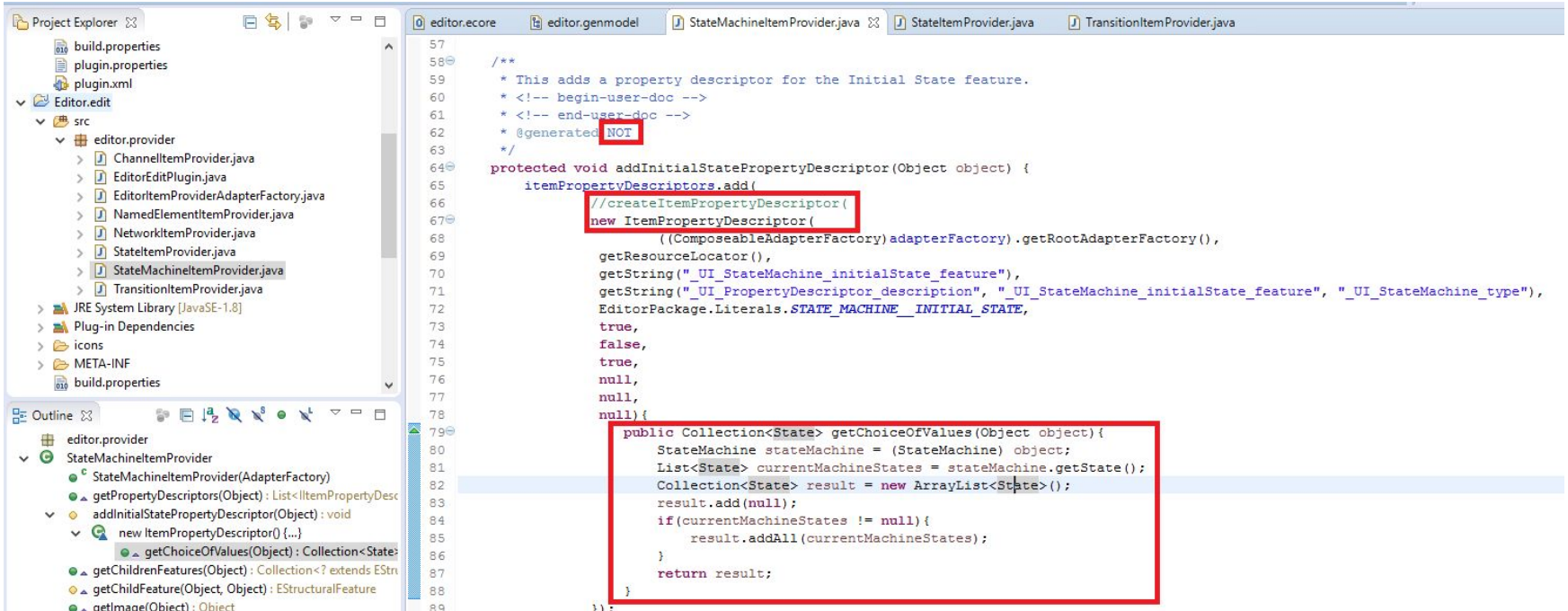
editor.ecore

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/' ;

package editor : editor = 'http://www.example.org/editor'
{
    class Network extends NamedElement
    {
        property stateMachine : StateMachine[+] { ordered composes };
        property declaration : Channel[+] { ordered composes };
        invariant
        NoDuplicatesDeclarationName('There are more than one channel of the network \'' + self.name + '\' with de same name.'): self.declaration->forAll(d1, d2 |
                d1 <> d2 implies d1.name <> d2.name);
        invariant
        NoDuplicatesStateMachineName('There are more than one state machine of the network \'' + self.name + '\' with de same name.'): self.stateMachine->forAll(sm1,
                sm2 | sm1 <> sm2 implies sm1.name <> sm2.name);
    }
    class StateMachine extends NamedElement
    {
        property state : State[+] { ordered composes };
        property initialState : State[1];
        property transition : Transition[+] { ordered composes };
        invariant
        NoDuplicatesStateName('There are more than one state of the state machine \'' + self.name + '\' with de same name.'): self.state->forAll(st1,
                st2 | st1 <> st2 implies st1.name <> st2.name);
    }
    class Channel extends NamedElement
    {
        attribute type : ChannelType[1];
    }
```

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

EDsl.xtext | EDslValidator.xtend

```
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    }
43
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 asynch orderCoffee ,
        Channel asynch deliverCoffee,
        Channel asynch 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

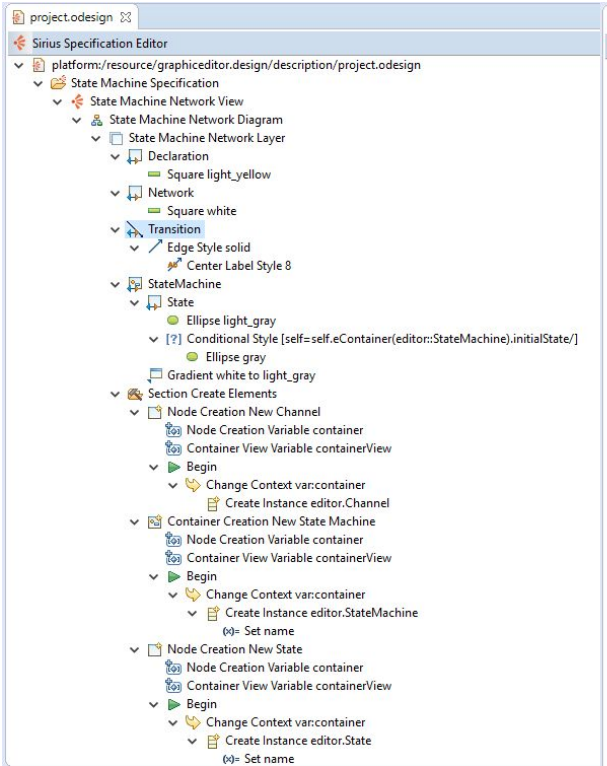
```
StateMachine CrossingControl{
    initialState initial
    state{
        State initial(incoming ready, outgoing pedestrianRequest),
        State pedestrianRequestReceived(incoming pedestrianRequest, outgoing trafficTurnRed),
        State turningTrafficLightsRed(incoming trafficTurnRed, outgoing 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)
    }
}
}
}
```

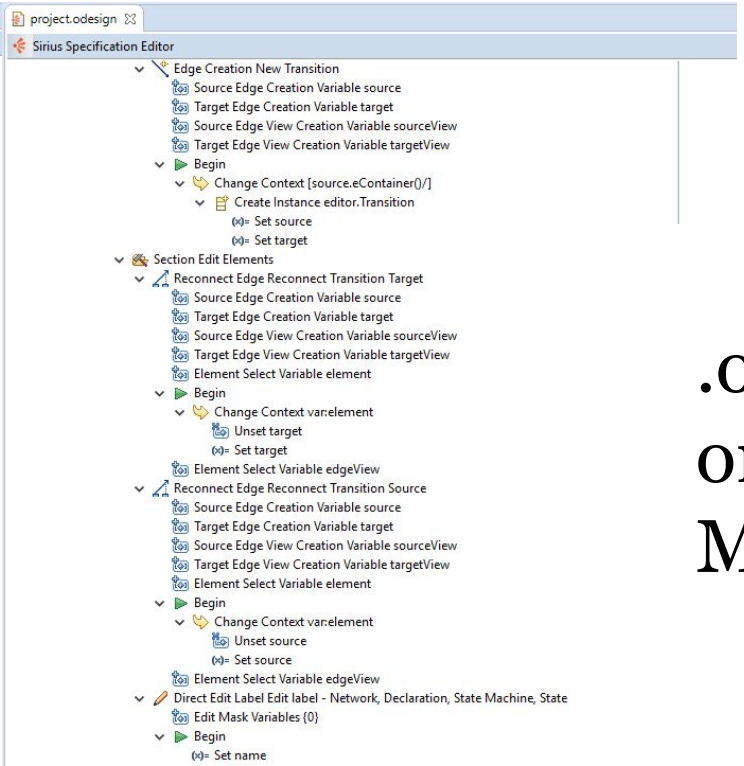
Xtext - Example WatchMovie

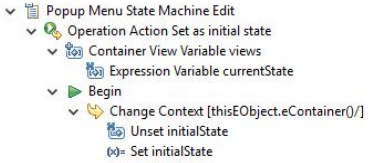
WatchMovies.edsl

```
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

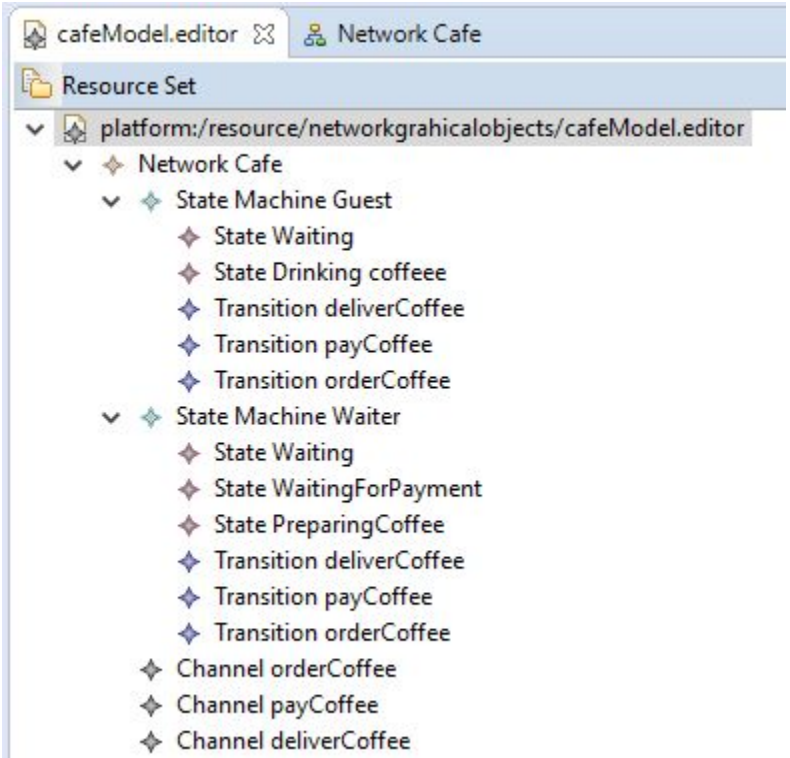




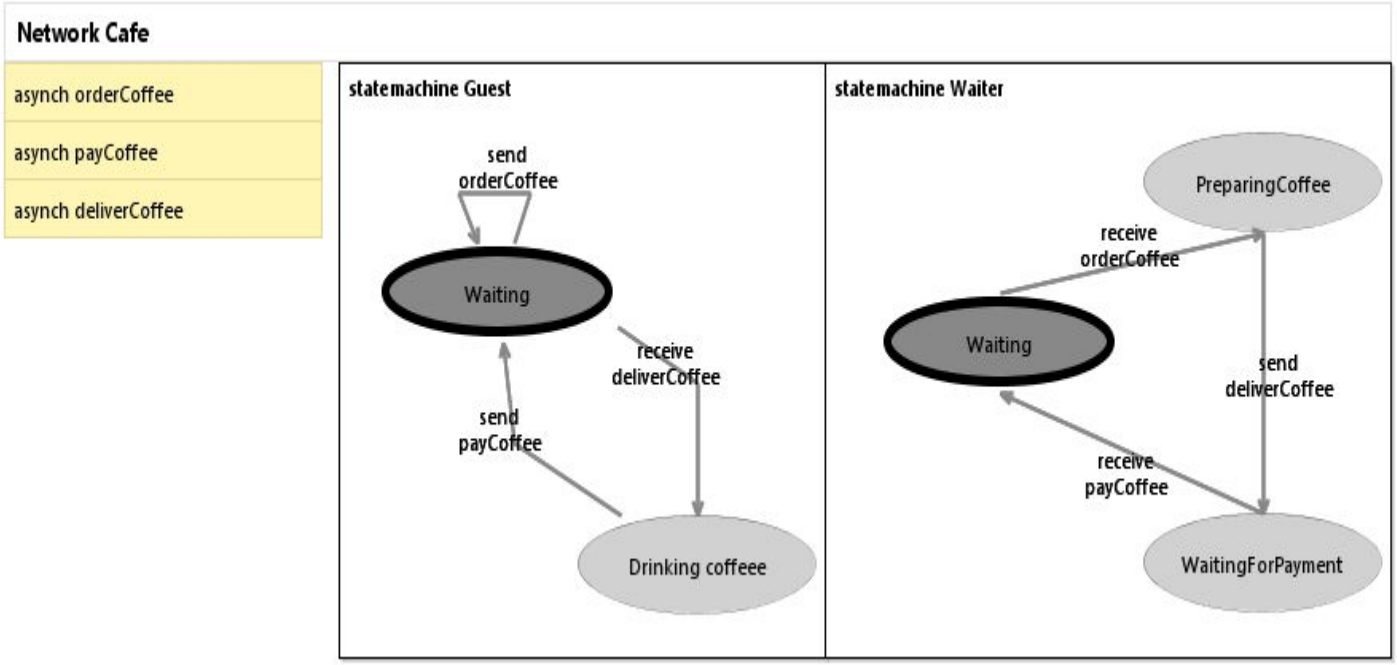


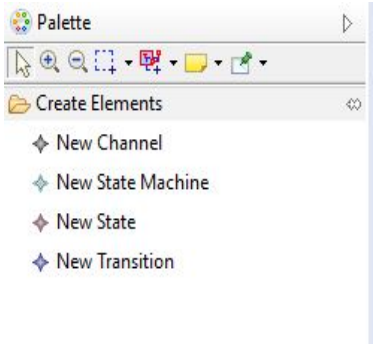
.odesign based on Ecore Meta-Model

Sirius - Example Cafe: Dynamic instance

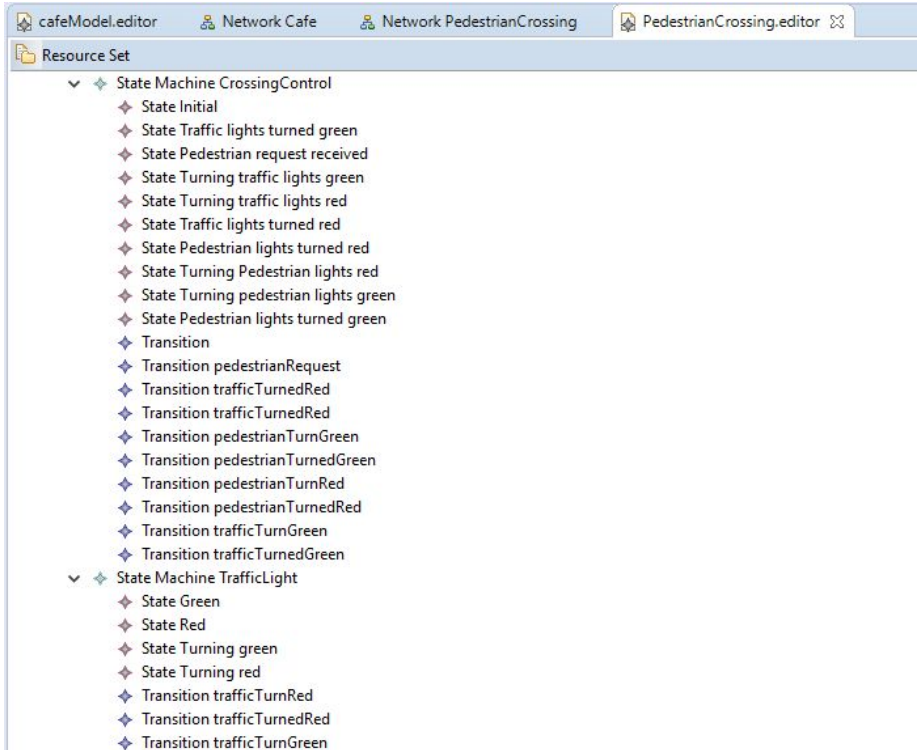


Sirius - Example Cafe

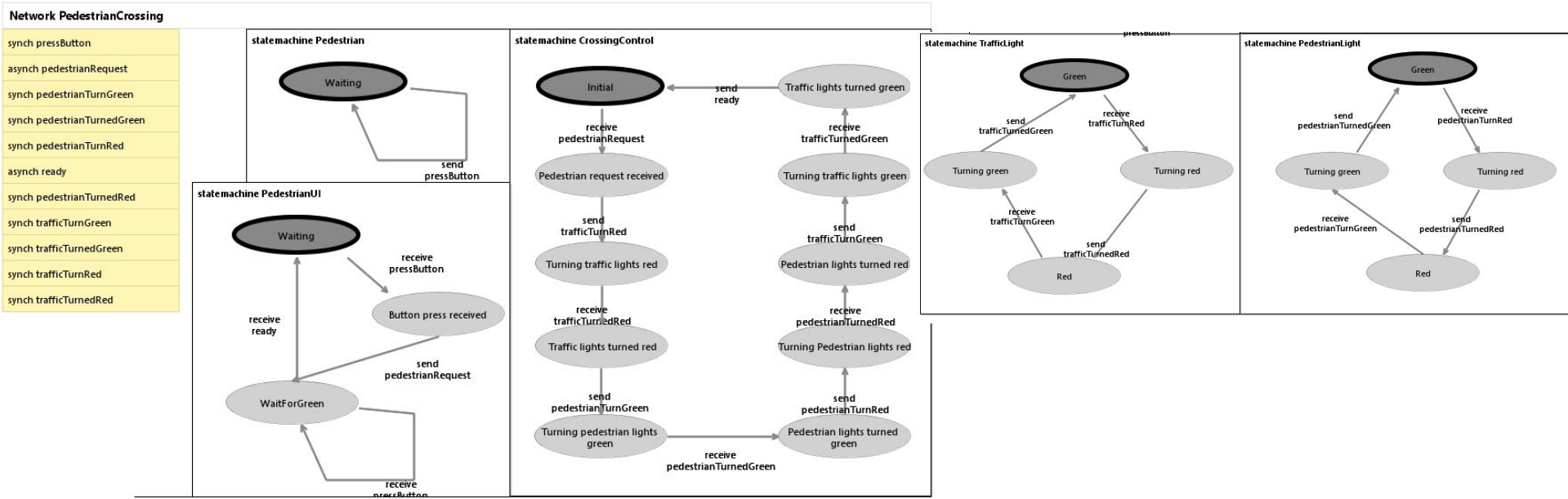




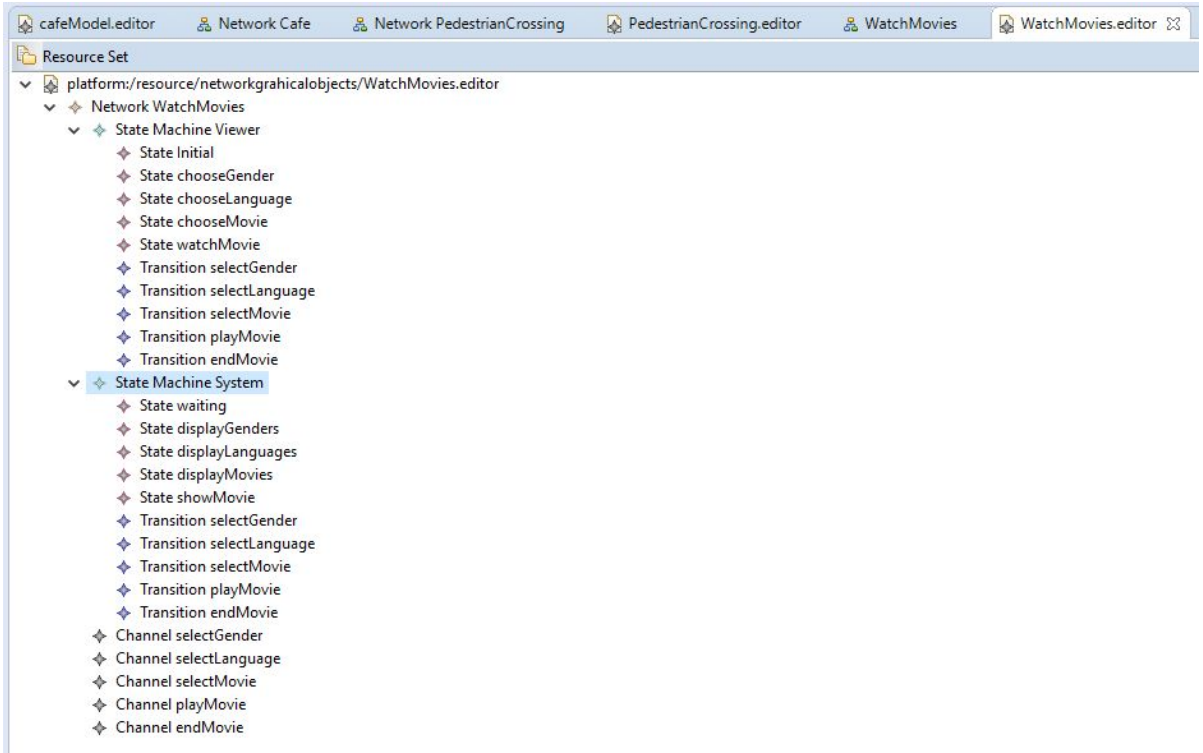
Sirius - Example pedestrianCrosisng: Dynamic instance



Sirius - Example pedestrianCrossing



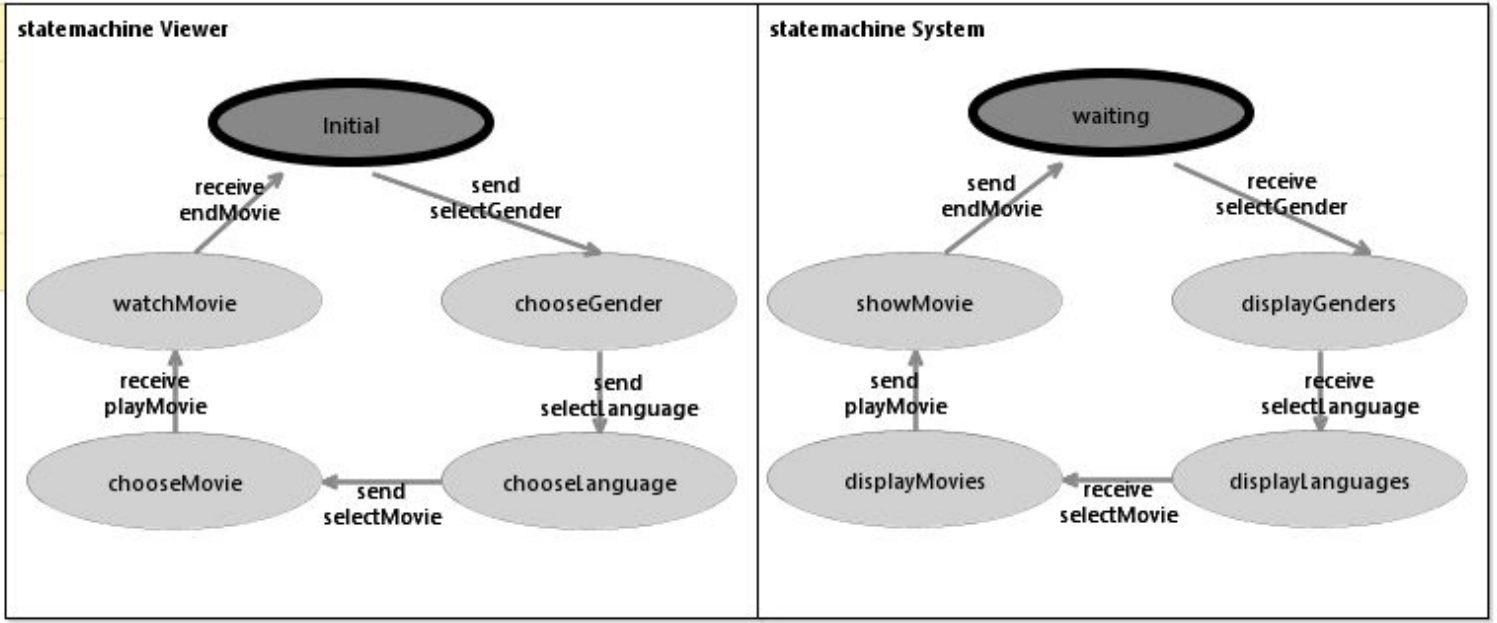
Sirius - Example WatchMovie: Dynamic instance



Sirius - Example WatchMovie

Network WatchMovies

- asynch selectGender
- asynch selectLanguage
- asynch selectMovie
- synch playMovie
- synch endMovie



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?