Model-based Software Engineering

Summer Term 2016

Mini-Project 1 Assignment:
State machine network language and editor

Note:

- *The deadline for submitting the solutions for the mini-project 1 is Monday, May 23rd, 23:59 CET.*
- *You can work on the mini-project in groups of three to four students. Use the lecture's Stud.IP forum for forming groups if necessary.*
- *Use the lecture's Stud.IP forum for discussion about the project, for asking questions and submitting your solutions. There is a dedicated category for mini-project 1 with an area for discussion and an area for submissions.*
- *Each submission must use its own topic. The first post of your topic must contain (1) the names of all group members, (2) a screenshot of your editor, (3) a download link to a ZIP file that contains the eclipse projects that comprise your solution, and (4) a download link to a PDF file that describes your solution. This description should be in the style of presentation slides and must include a brief description on how to use your editor and an explanation of your solutions to each of the tasks below.*
- *Presentation: The results of this mini-project can be presented in the tutorial session on Tuesday, May 24th. A mini-project can be presented by one member of a mini-project group, who can obtain the presentation bonus by doing so (see bonus point rules explained in the first lecture). If you want to present, send me an email with your presentation slides in PDF format at least by 23:59 on the Monday before the tutorial. Presentations should take between 5 to 10 minutes.*

State machine networks

In this mini-project, you shall *define a language* and *build an editor* for networks of simple communicating state machines. (We choose this simple formalism in order to have relatively few language constructs, but at the same time create a language with which we can have some fun generating code and doing other things in the second mini project.)

In Figure 1 you see a simple network of two state machines that can communicate via channels. In this example, there is one state machine called

Guest and another one called Waiter. They can communicate by sending messages via the asynchronous channels orderCoffee, deliverCoffee, and payCoffee. Initially, a guest and the waiter are waiting. When waiting, the guest can order coffee. The waiter, when waiting, can receive such an order and, after preparing the coffee, can deliver the coffee to the guest. Then the guest will drink coffee and then pay for the coffee. After receiving the payment, the waiter is again waiting for the next coffee order.

As you can see, a network of state machines has a name and consists of multiple state machines and a declaration of a set of channels that can be asynchronous or synchronous. Each state machine has a name and consists of multiple states that can each have a name and where one is the initial state. There can also be transitions between the states. States can have multiple incoming and outgoing transitions, and each transition has one source and one target state. Furthermore, each transition has a label send ⟨channel⟩ or receive ⟨channel⟩ that indicates either the sending or the receiving of a message via a channel.

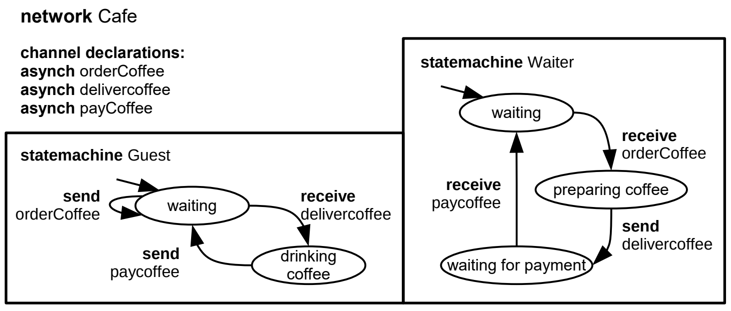

Figure 1: A simple network of two communicating state machines Guest and Waiter

A more complex network is shown in the Figure 2.

1 Model the metamodel with an Ecore class diagram

Create an Ecore metamodel that captures all the concepts of networks of state machines as shown and explained above. (Include a class diagram in your description.)

2 Build a textual editor

Define a textual syntax for the networks of state machines and build a textual editor using Xtext.

3 Build a graphical editor

Define a graphical syntax and build a graphical editor using Sirius. You can either try building an editor that supports editing multiple state machines in one diagram or you can build an editor that supports editing the single state machines that comprise the network. (If you run into problems with this task, also partial solutions are acceptable, but then you have to explain the problems that you ran into in the solution presentation document.)

4 Model examples with the editors

Validate your syntax definitions and editors by showing that you can successfully model the two examples provided here. Also create at least one other example state machine network.

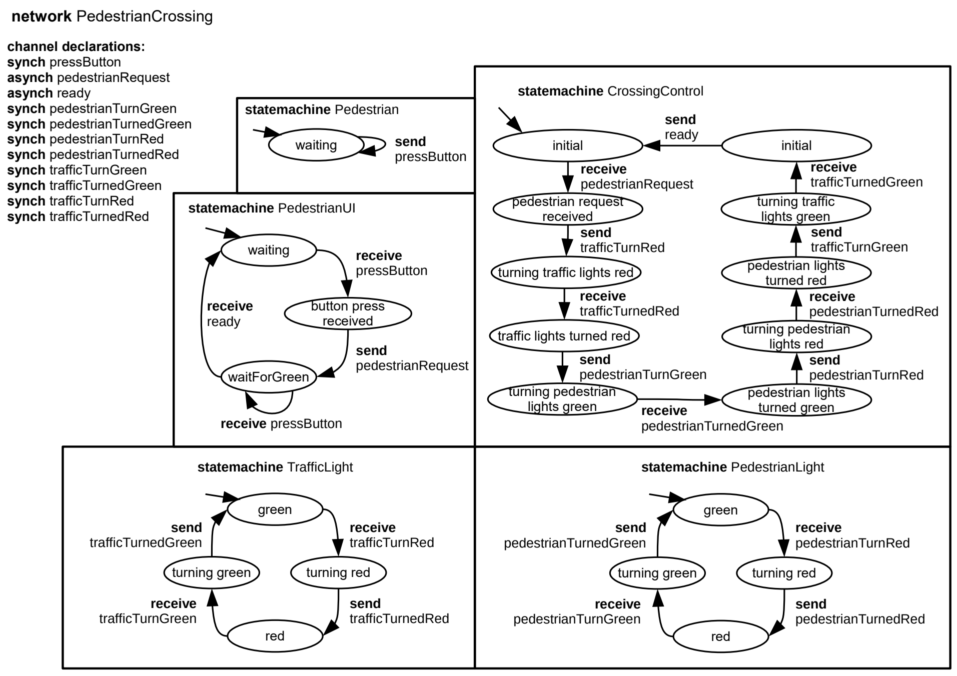
network PedestrianCrossing
channel declarations:
synch pressButton
asynch pedestrianRequest
asynch ready
synch pedestrianTurnGreen
synch pedestrianTurnedGreen
synch pedestrianTurnRed
synch pedestrianTurnedRed
synch trafficTurnGreen
synch trafficTurnedGreen
synch trafficTurnRed
synch trafficTurnedRed
statemachine Pedestrian
waiting
send pressButton
statemachine PedestrianUI
waiting
receive pressButton
button press received
send pedestrianRequest
waitForGreen
receive pressButton
receive ready
statemachine CrossingControl
initial
receive pedestrianRequest
pedestrian request received
send trafficTurnRed
turning traffic lights red
receive trafficTurnedRed
traffic lights turned red
send pedestrianTurnGreen
turning pedestrian lights green
receive pedestrianTurnedGreen
pedestrian lights turned green
send pedestrianTurnRed
turning pedestrian lights red
receive pedestrianTurnedRed
pedestrian lights turned red
send trafficTurnGreen
turning traffic lights green
receive trafficTurnedGreen
initial
send ready
statemachine TrafficLight
green
receive trafficTurnRed
turning red
send trafficTurnedRed
red
receive trafficTurnGreen
turning green
send trafficTurnedGreen
statemachine PedestrianLight
green
receive pedestrianTurnRed
turning red
send pedestrianTurnedRed
red
receive pedestrianTurnGreen
turning green
send pedestrianTurnedGreen

Figure 2: A network of a traffic light for a pedestrian crossing