Solution Proposal

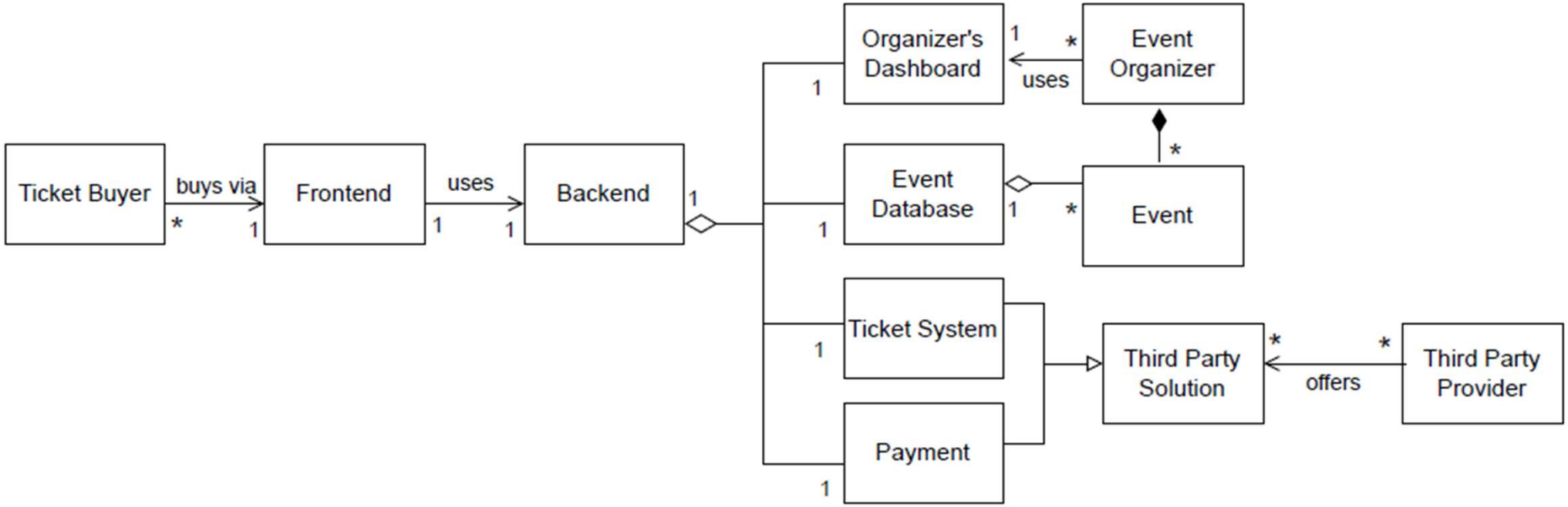
Task I

- Make a class-diagram describing the relevant concepts. The class-diagram should **at least** represent the following concepts: event, event organizer, ticket buyer, third party provider, and the five components on slide 10.
- The class-diagram **must contain** the generalization relation and illustrate correct use of both composition and aggregation. All associations **must** be decorated with multiplicities.

Task I – Class Diagram

Make sure that the diagrams:

- contain everything they should according to the task description
- do not contain a class that is contained in two different compositions (with the black diamond)
- have no multiplicities on the generalization relation (representing inheritance)
- do not contain a composition with a multiplicity different from 1 at the black diamond end (1 is the default value)


Ticket Buyer
buys via
*
1
Frontend
uses
1
1
Backend
1
1
Organizer's Dashboard
1
*
uses
Event Organizer
*
1
Event Database
1
*
Event
1
Ticket System
1
Payment
Third Party Solution
*
*
offers
Third Party Provider

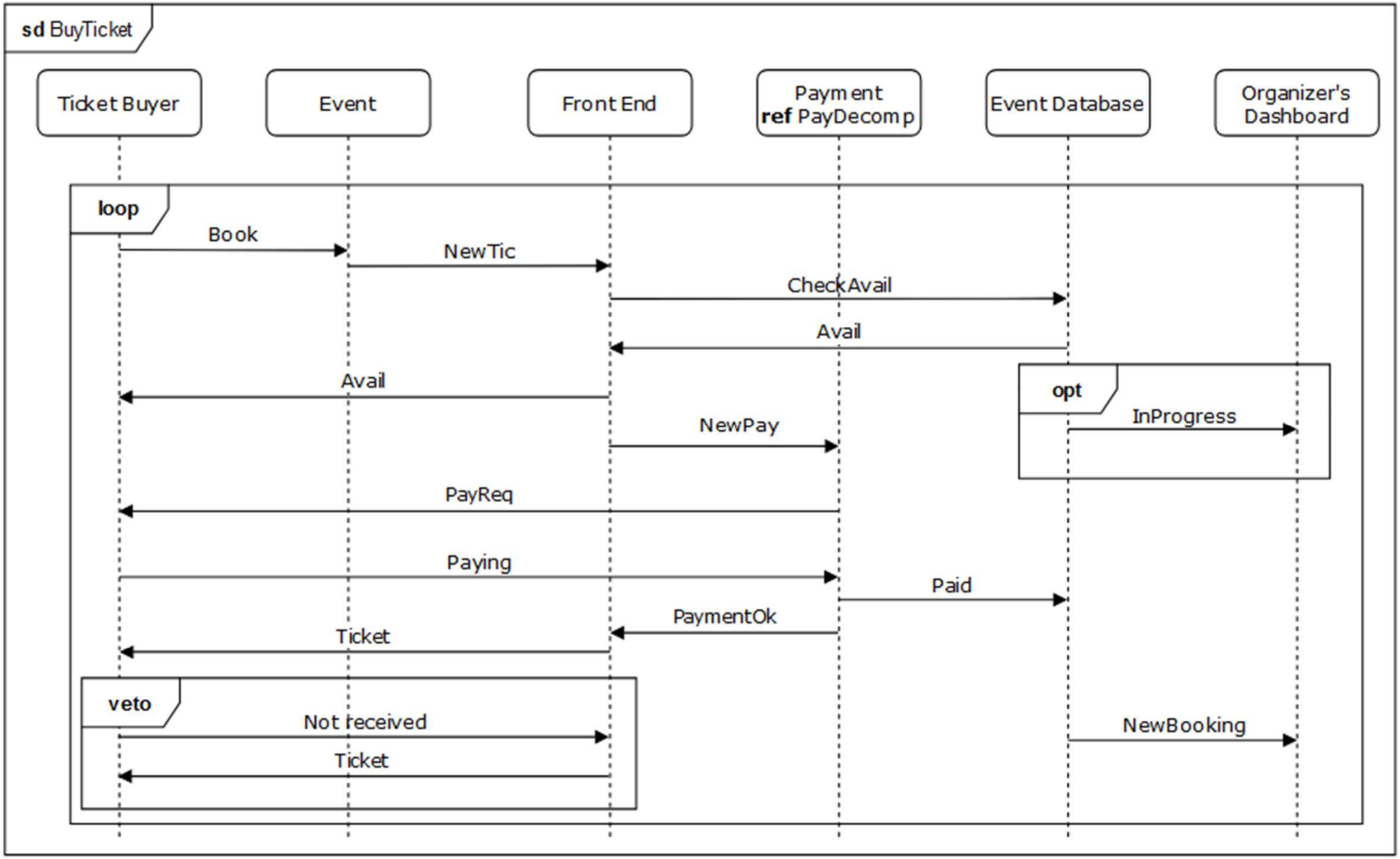
Task II

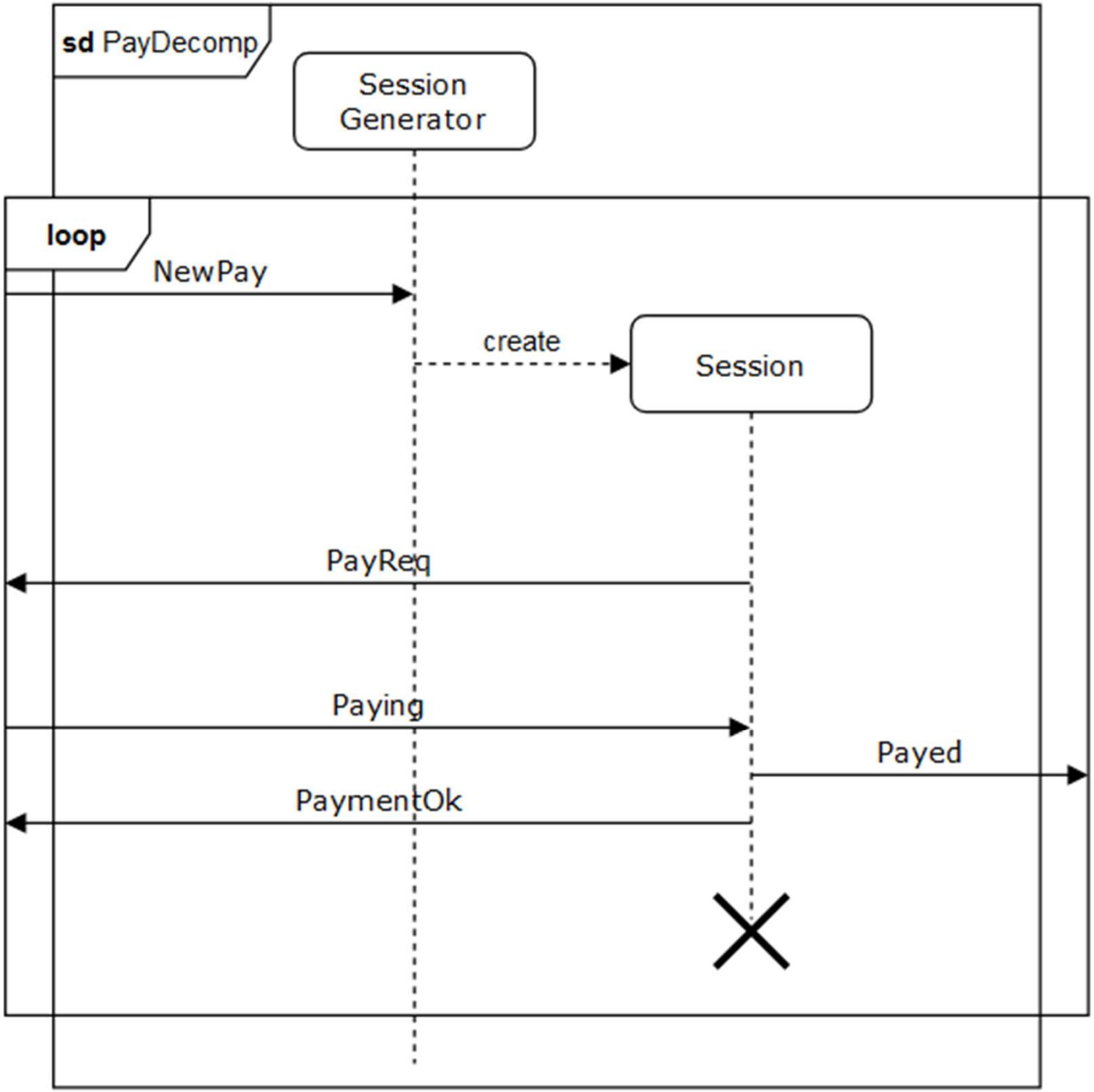
- Make one main sequence diagram describing a booking session. There **must at least** be lifelines for event, ticket buyer, frontend, event database and two of the three other components on slide 10.
- The sequence diagram **must** illustrate correct use of at least referencing (the referenced sequence diagrams **must** also be specified in full), loop, lifeline creation and termination, and opt. The sequence diagram should also describe some negative behaviour. Please avoid excessive complexity. The sequence diagram being made in this task will be used further in the next task.
- Describe the first ten events of at least one positive trace represented by your sequence diagram.

Task II – Sequence Diagram

Make sure that the diagrams:

- contain everything they should according to the task description


sd BuyTicket
Ticket Buyer
Event
Front End
Payment
ref PayDecomp
Event Database
Organizer's Dashboard
loop
Book
NewTic
CheckAvail
Avail
Avail
opt
InProgress
NewPay
PayReq
Paying
Paid
PaymentOk
Ticket
veto
Not received
Ticket
NewBooking



Note that the loop crosses the diagram frame because the loop is global (the loop from the previous diagram)

10 events of one positive trace of BuyTicket

<!Book,?Book,!NewTic,?NewTic, !CheckAvail, ?CheckAvail, !Avail, !InProgress,
?InProgress, ?Avail, ….>

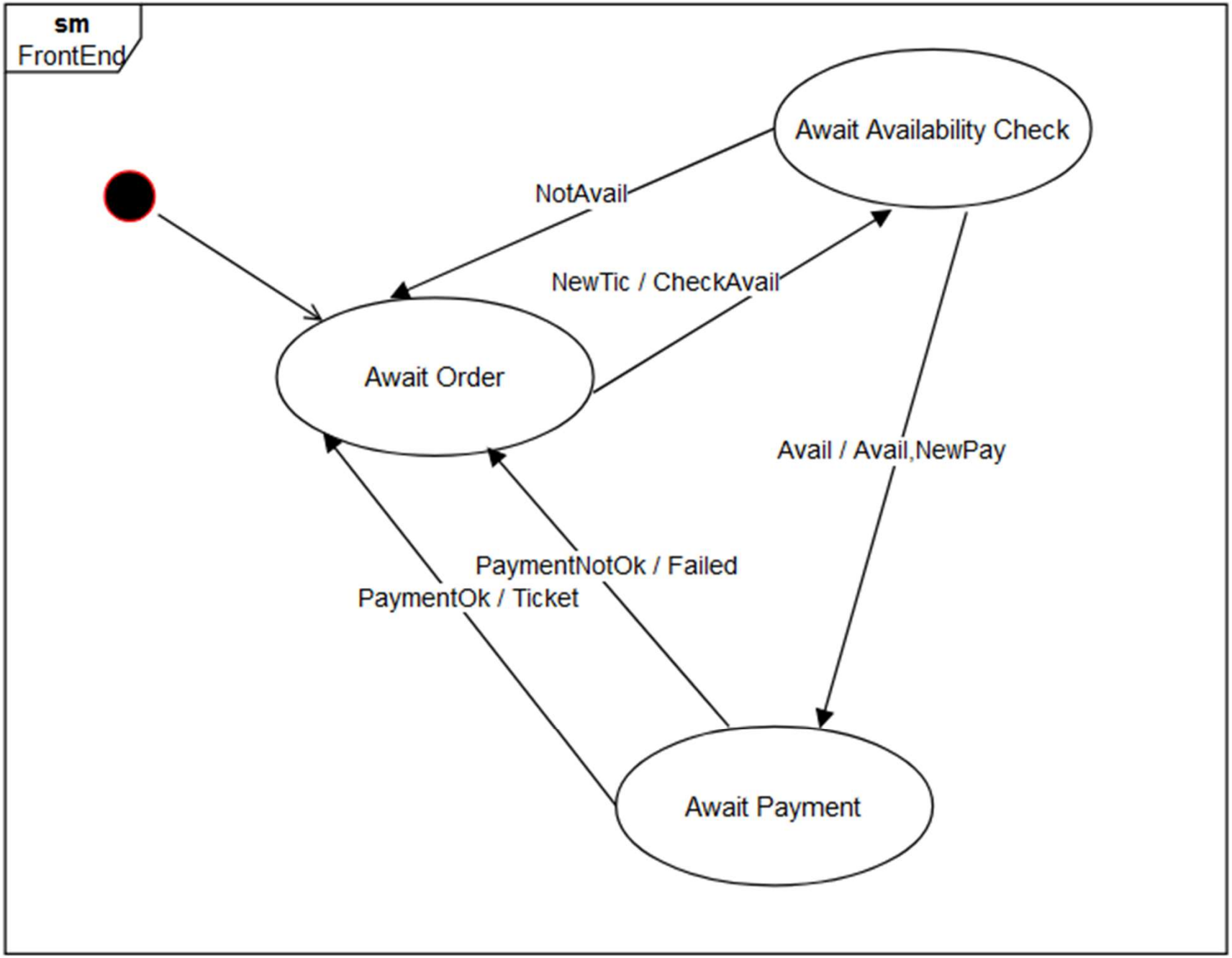
Task III

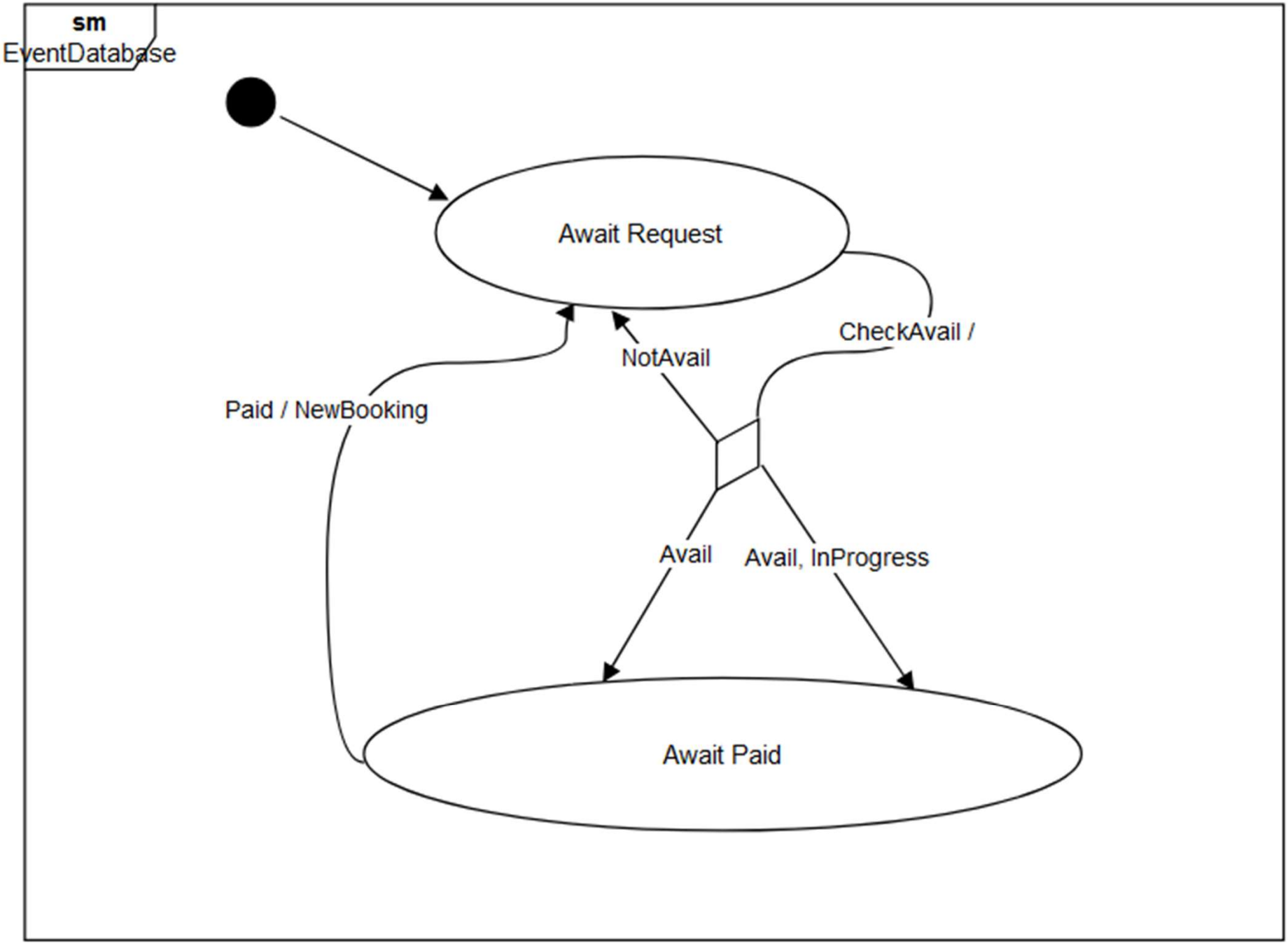
- Make a communicating state-machine that correctly reflects all the traces specified by the sequence diagram for the frontend.
- Make a communicating state-machine that correctly reflects all the traces specified by the sequence diagrams for the event database.
- The two state-machines must allow any positive trace allowed by the respective lifelines in the sequence diagrams but may have additional traces.
- Remember that the machines should communicate the messages specified by the sequence diagrams. Hence, for a lifeline, an input **signal (output signal)** in a state machine should be the **reception (sending)** of a message in the sequence diagram.

Task III – State Machine

Make sure that the machines:

- are reactive (no transition without input)
  - Only the first transition from the initial state (black ball) to the first control state can send without receiving input. This transition cannot receive.
- allow all positive traces allowed by the single lifeline they represent
  - The state machines may allow much more.
- use the same message names as the sequence diagrams
  - The input messages of the lifeline are inputs of its state machine
  - The output messages of the lifeline are outputs of its state machine
- any transition has a decoration of of the form “in/out” unless it only inputs in which case its decoration is of the form “in”
- “in” is a single input message
- the “out” may contain a list of output messages; it may also include actions (e.g. assignments) but the latter is not emphasized in this course


sm
FrontEnd
Await Availability Check
NotAvail
NewTic / CheckAvail
Await Order
Avail / Avail,NewPay
PaymentNotOk / Failed
PaymentOk / Ticket
Await Payment


sm
EventDatabase
Await Request
CheckAvail /
NotAvail
Paid / NewBooking
Avail
Avail, InProgress
Await Paid