

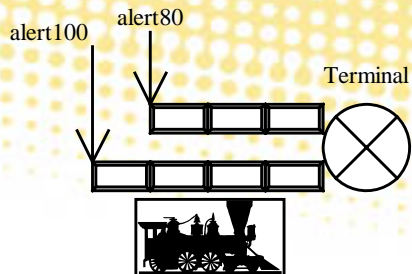
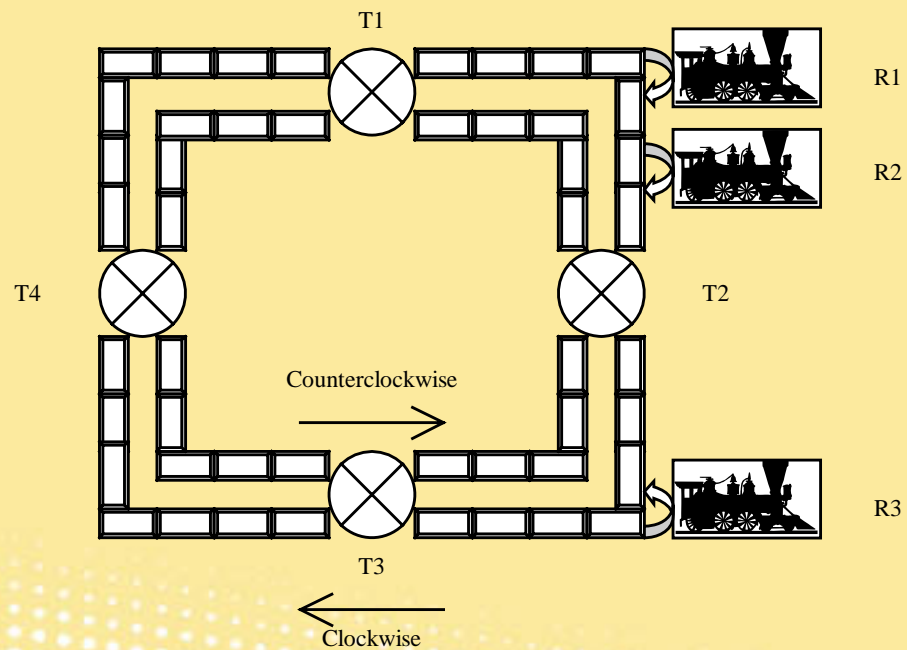
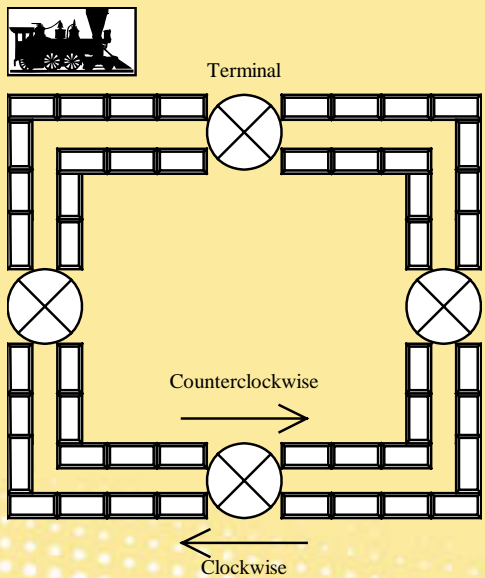
# *“Railcar control system” case study*

Franck Barbier

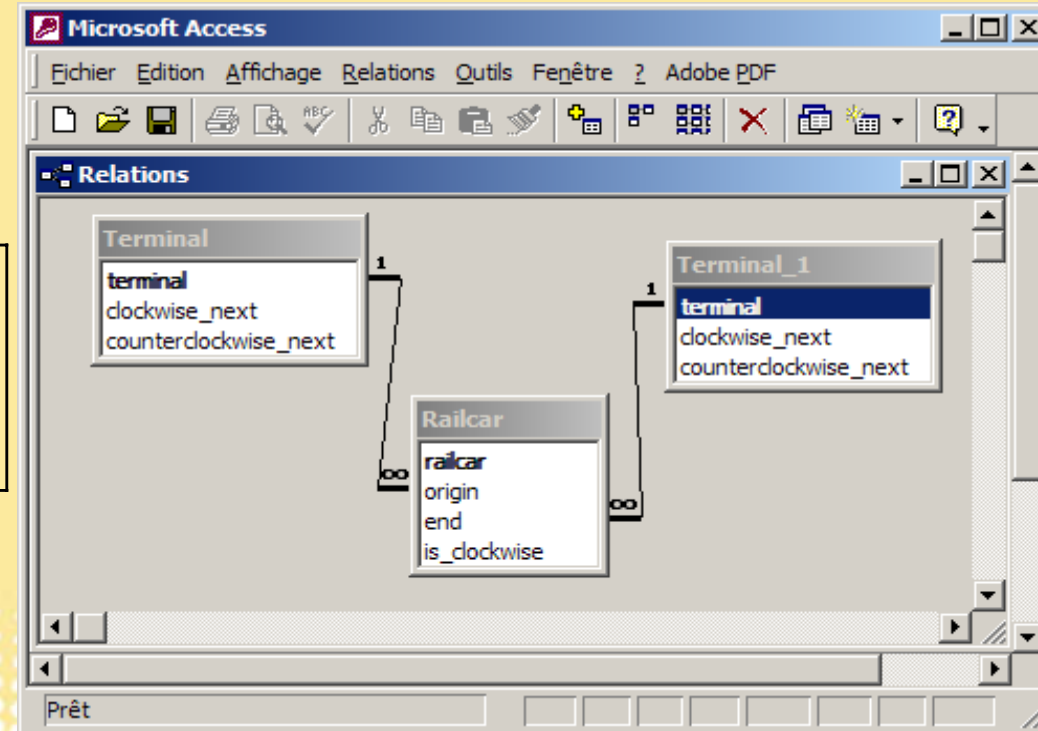
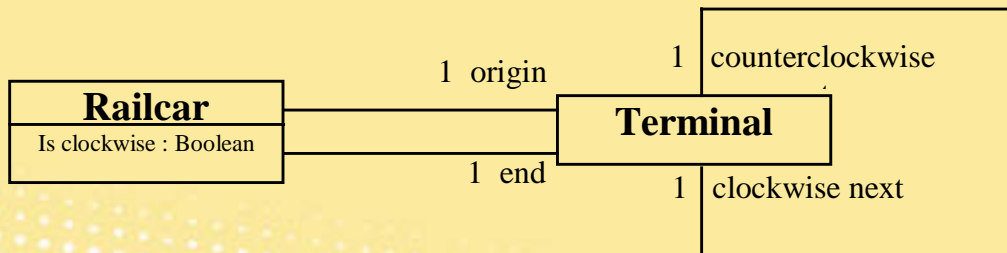
*Université de Pau et des Pays de l’Adour*

Download: [www.PauWare.com](http://www.PauWare.com)

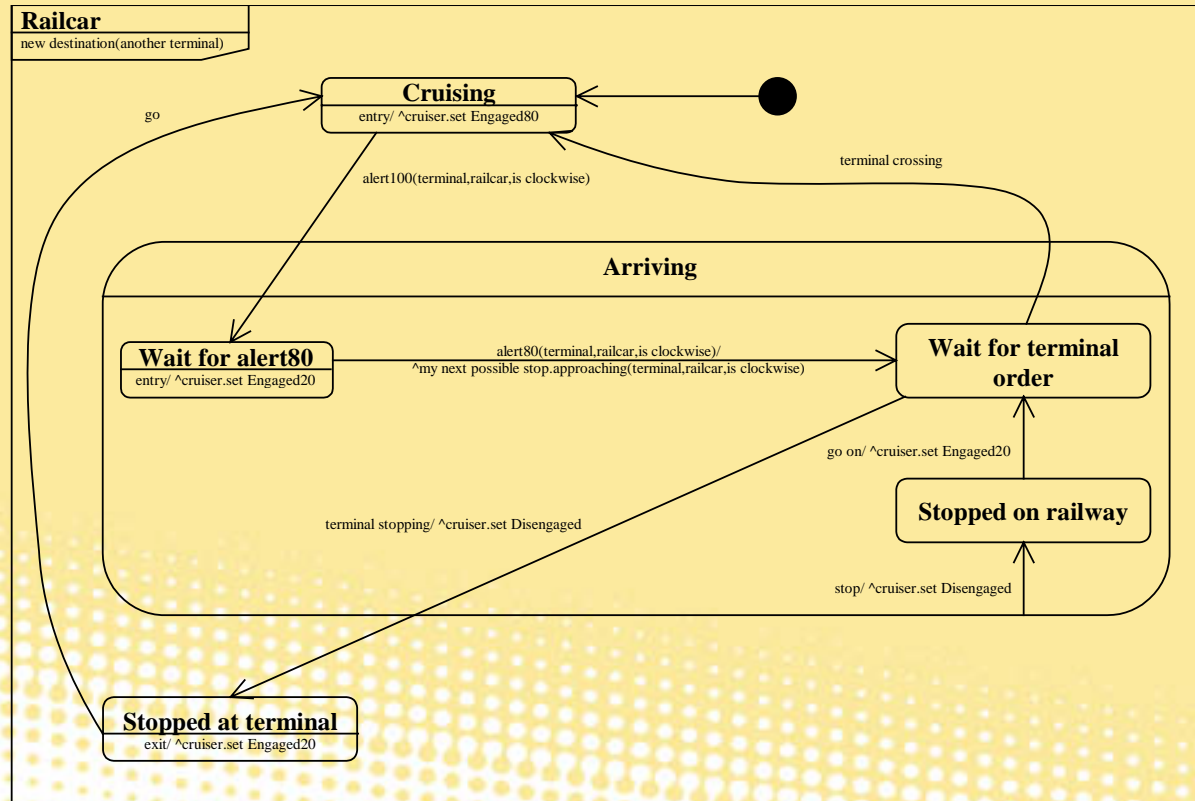
# Railcar control system



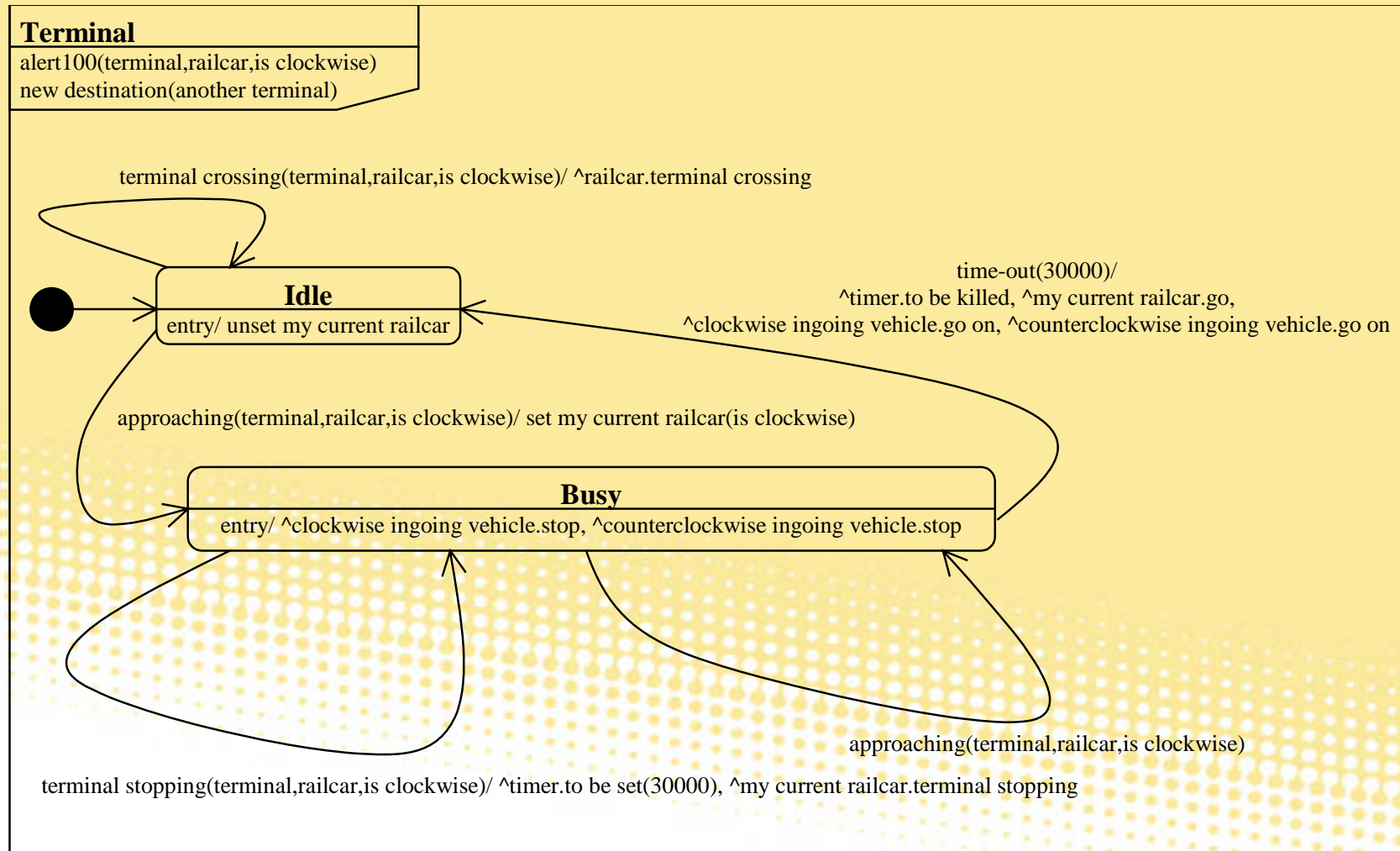
# PIMs, Class Diagram, *topology* & PSMs



# PIMs, Railcar State Machine Diagram



# PIMs, Terminal State Machine Diagram

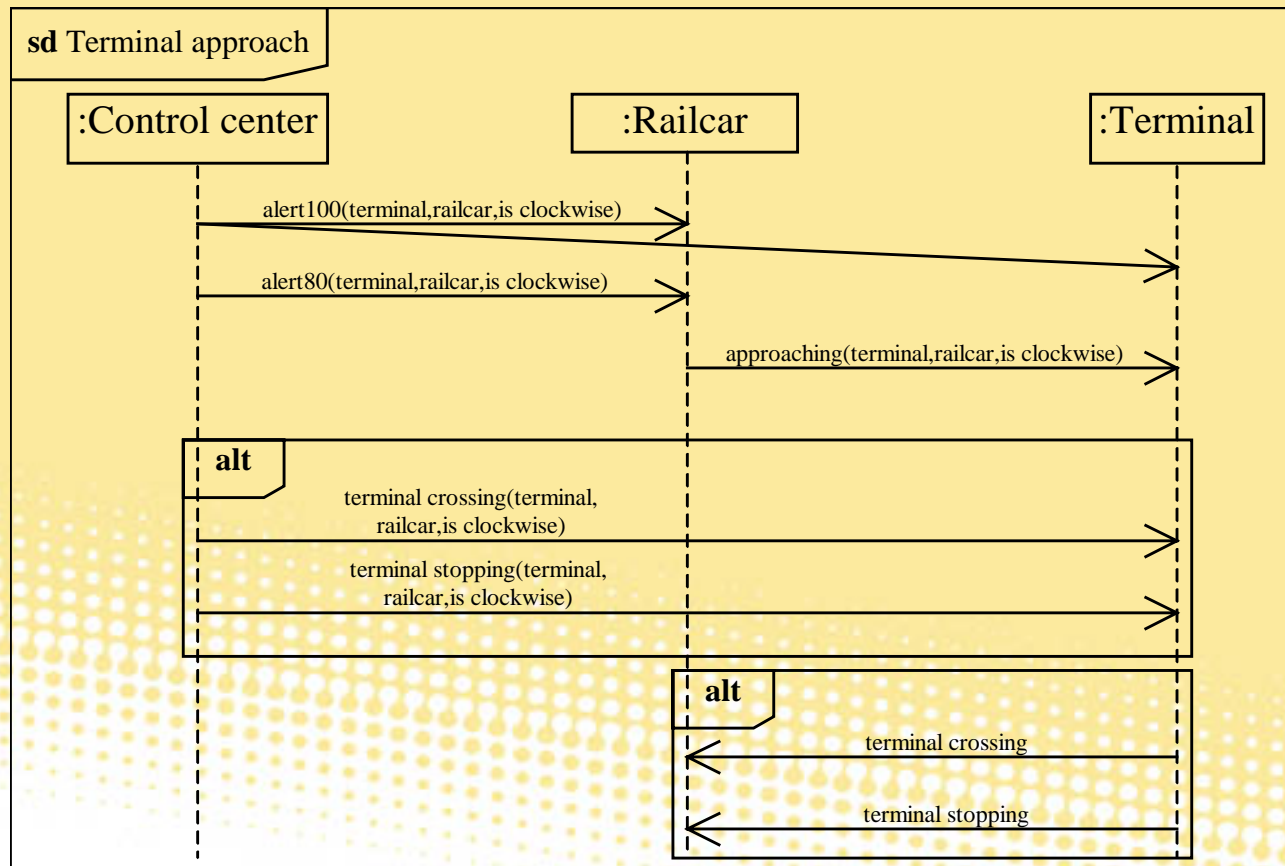


# *PauWare* library-based design

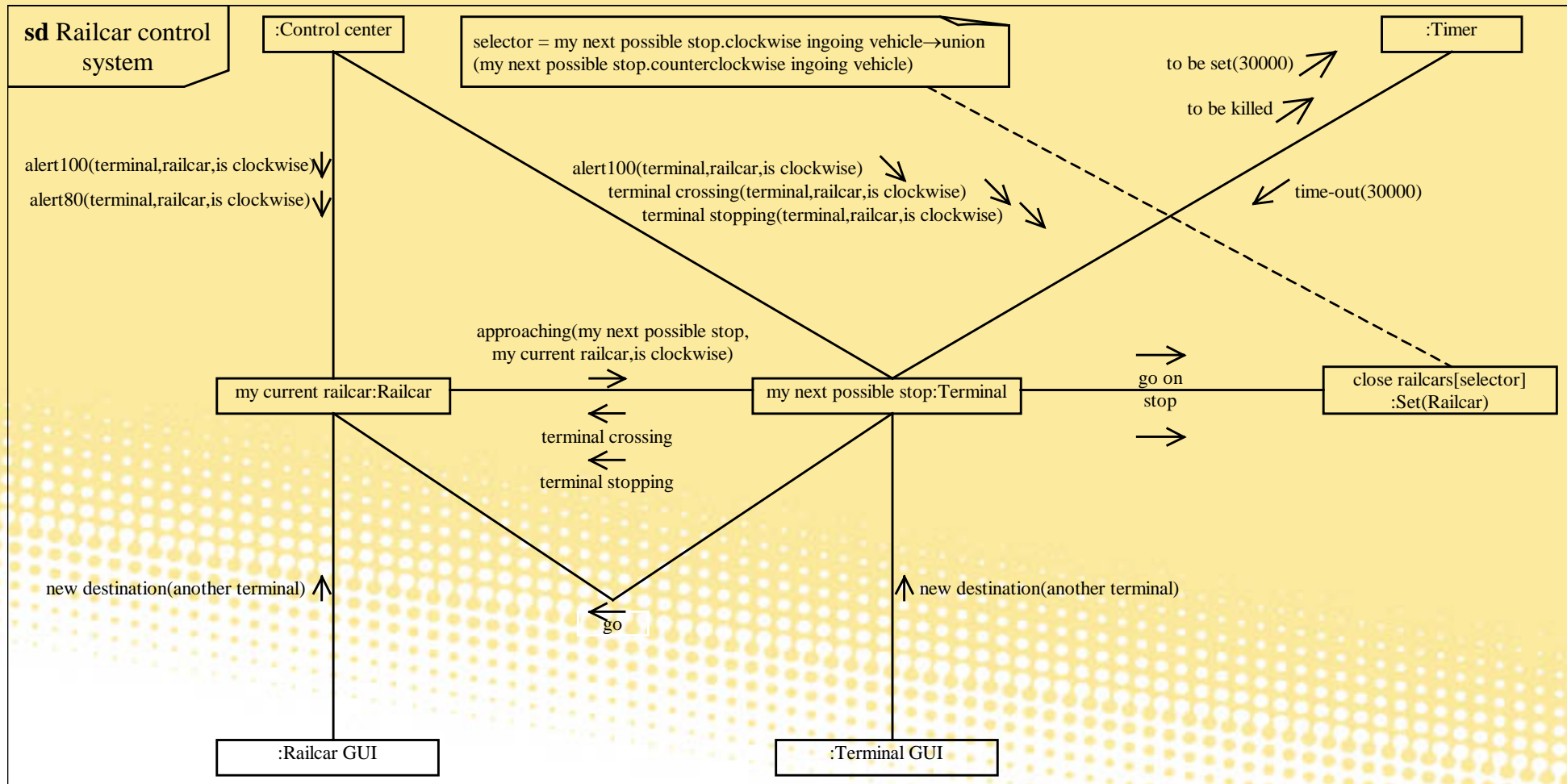
```
protected void init_behavior() throws Statechart_exception {
    _Busy = new Statechart("Busy");
    _Busy.entryAction(this,"clockwise_ingoiing_vehicle_stop");
    _Busy.entryAction(this,"counterclockwise_ingoiing_vehicle_stop");
    _Idle = new Statechart("Idle");
    _Idle.entryAction(this,"unset_my_current_railcar");
    _Idle.inputState();
}

protected void start(String terminal) throws Statechart_exception {
    _Terminal = new Statechart_monitor(_Busy.xor(_Idle),terminal,true);
    Object[] args = new Object[1];
    args[0] = new Long(1000); // value in spec. is 30000
    _Terminal.fires("terminal_stopping",_Busy,_Busy,true,this,"to_be_set",args);
    _Terminal.fires("terminal_stopping",_Busy,_Busy,true,this,"my_current_railcar_terminal_stopping");
    _Terminal.fires("time_out",_Busy,_Idle,true,this,"to_be_killed");
    _Terminal.fires("time_out",_Busy,_Idle,true,this,"my_current_railcar_go");
    _Terminal.fires("time_out",_Busy,_Idle,true,this,"clockwise_ingoiing_vehicle_go_on");
    _Terminal.fires("time_out",_Busy,_Idle,true,this,"counterclockwise_ingoiing_vehicle_go_on");
}
```

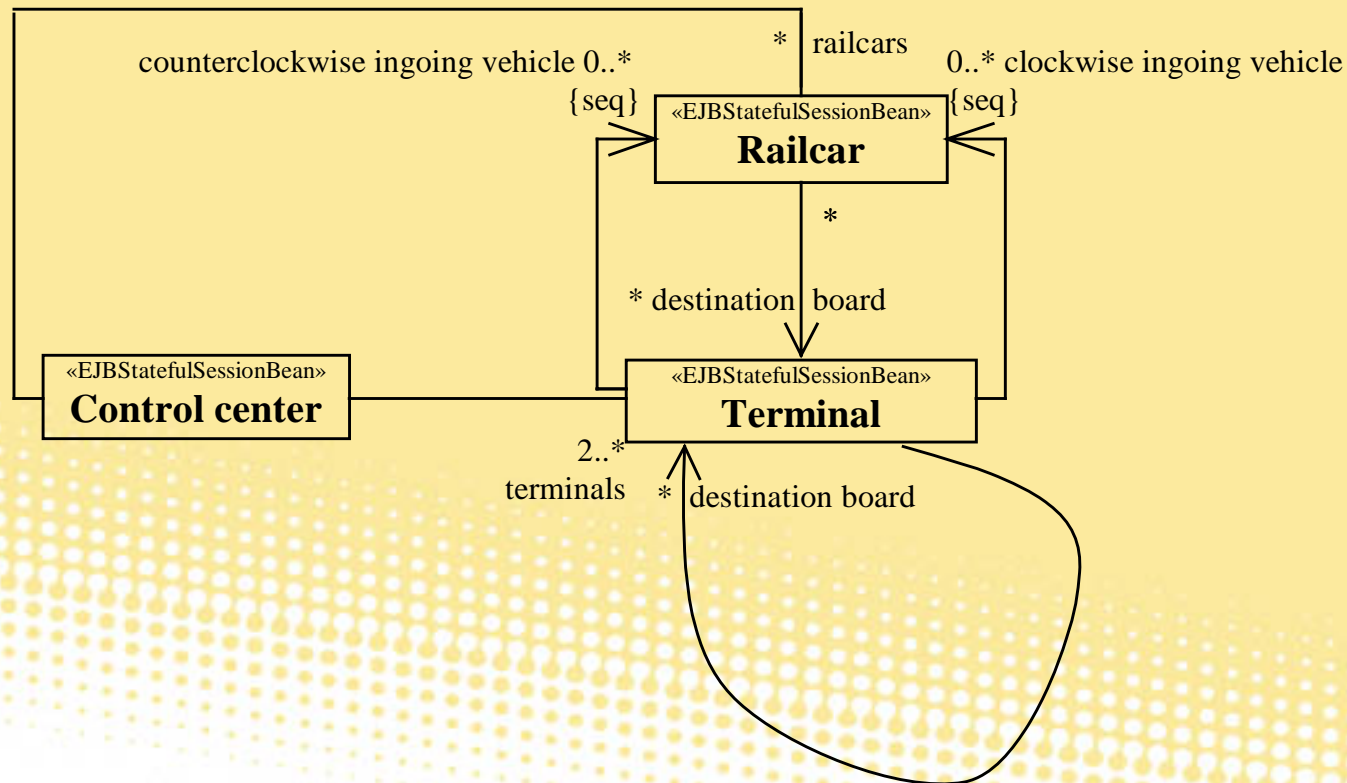
# PIMs, Sequence Diagram (*partial*)



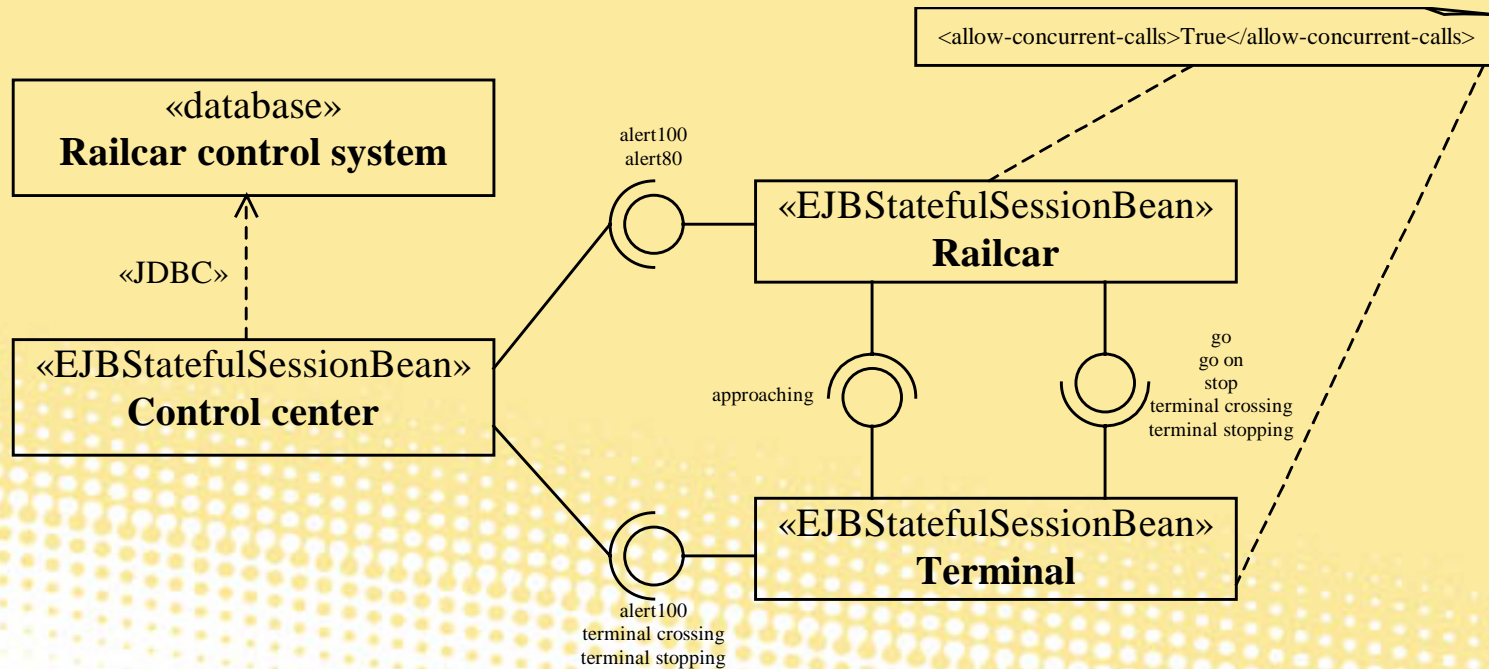
# PIMs, Communication Diagram



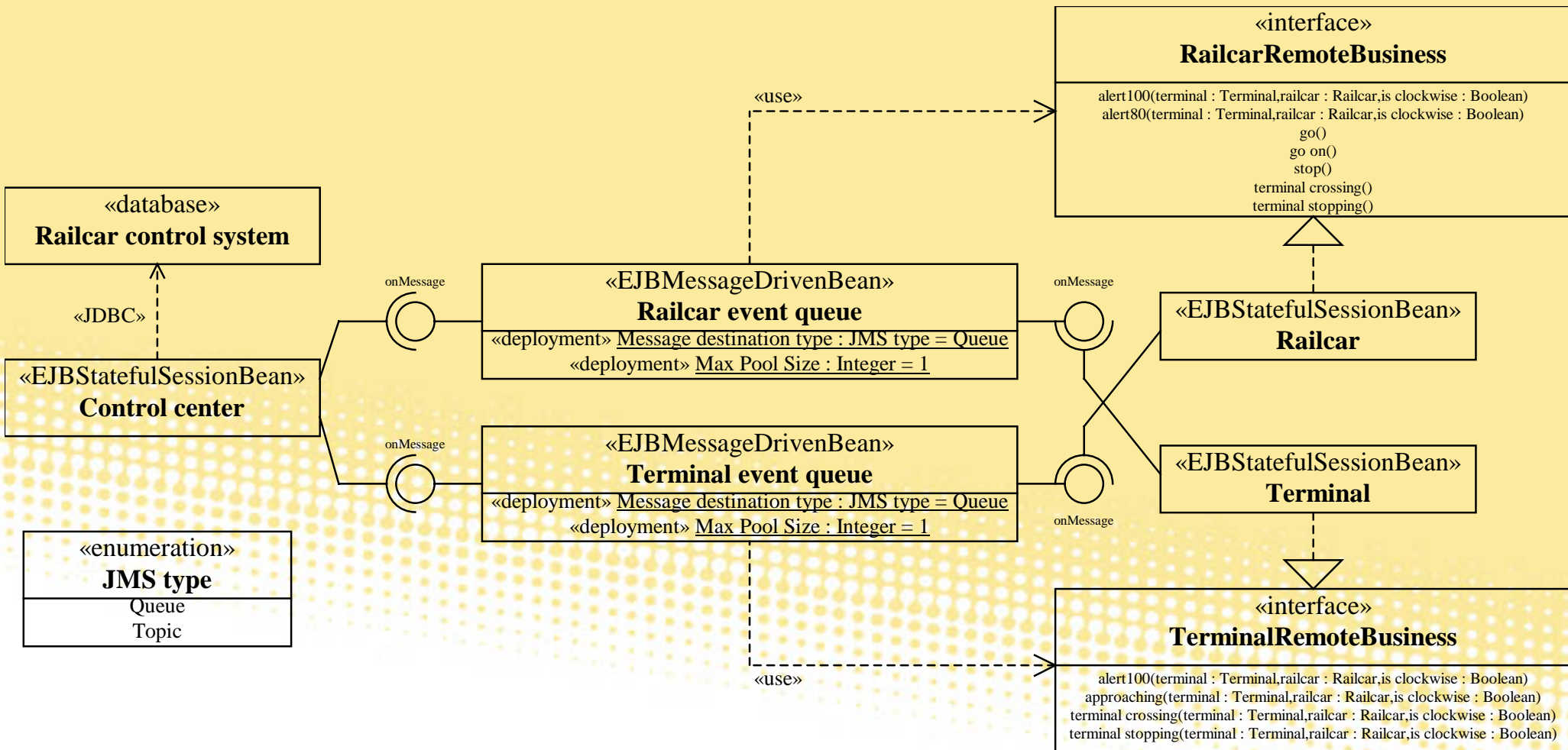
# PSMs, Class Diagram, *situation*



# Proprietary PSM



# EJB 2.1-compliant PSM



# Message routing to terminals

```

public class Terminal_event_queueBean implements MessageDrivenBean,MessageListener {
    ...
    public void onMessage(Message message) {
        ObjectMessage object_message;
        if(message instanceof ObjectMessage) {
            object_message = (ObjectMessage)message;
            try {
                TerminalRailcarIs_clockwise message_data =
                (TerminalRailcarIs_clockwise)((ObjectMessage)object_message).getObject();
                if(message_data._event != null && message_data._event.equals("alert100"))
                ((TerminalRemote)message_data._terminal.getEJBObject()).alert100((TerminalRemote)message_data._terminal.getEJBObject(),(RailcarRemote)message_data._railcar.getEJBObject(),message_data._is_clockwise);
                if(message_data._event != null && message_data._event.equals("approaching"))
                ((TerminalRemote)message_data._terminal.getEJBObject()).approaching((TerminalRemote)message_data._terminal.getEJBObject(),(RailcarRemote)message_data._railcar.getEJBObject(),message_data._is_clockwise);
                if(message_data._event != null && message_data._event.equals("terminal_crossing"))
                ((TerminalRemote)message_data._terminal.getEJBObject()).terminal_crossing((TerminalRemote)message_data._terminal.getEJBObject(),(RailcarRemote)message_data._railcar.getEJBObject(),message_data._is_clockwise);
                if(message_data._event != null && message_data._event.equals("terminal_stopping"))
                ((TerminalRemote)message_data._terminal.getEJBObject()).terminal_stopping((TerminalRemote)message_data._terminal.getEJBObject(),(RailcarRemote)message_data._railcar.getEJBObject(),message_data._is_clockwise);
            } catch(Exception e) {
                throw new RuntimeException(e);
            }
        }
    }
}

```

# Exercice

- Les propriétés *destination board* de *Railcar* et de *Terminal* ne sont pas actuellement utilisées dans l'application
- Logiquement, *Control center* émet l'événement *terminal crossing* si le champ *destination board* de *Railcar* ne contient pas le terminal dont il s'approche et si le champ *destination board* de ce même terminal ne contient pas son terminal adjacent. Dans le cas contraire, *Control center* doit émettre l'événement *terminal stopping*
- Implanter cette logique dans l'application en veillant attentivement à ce que *Control center* ne reçoive des requêtes que d'un seul client