1 Traffic Light Controller

As illustrated by Figure 1, the traffic light controller controls four lights, namely, two lights for the pedestrians and two for the cars.

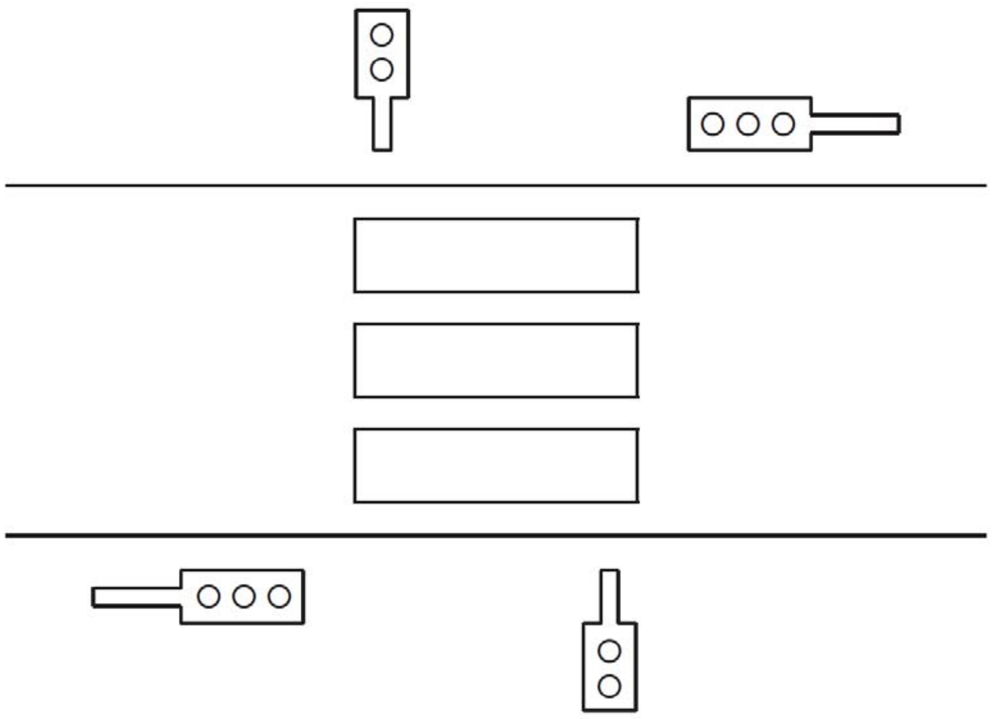

Figure 1 Pedestrian road crossing

Initially, the traffic light is red for the pedestrians and green for the cars. Each of the two traffic lights for the pedestrians has a button. If a pedestrian presses one of these buttons while the lights for the cars are green and the lights for the pedestrians are red, the traffic lights turn yellow and then red for the cars and finally green for the pedestrians. After a while the lights turn back to their initial positions, but in the opposite order. We make use of the following four types of signals to describe the controller:

$$
\begin{aligned}
\text{type } Col &= green \mid yel \mid red \\
\text{type } Par &= car \mid ped \\
\text{type } Con &= set(c \in Col, p \in Par) \\
\text{type } Act &= req \mid ack(n \in Col, p \in Par)
\end{aligned}
$$

The interaction between the controller and the traffic light is assumed to be acknowledgement driven.

Define a UML state machine for the traffic light controller ignoring timing aspects, and the handling of exceptions and malfunctions.

2 Tourist Guides on Demand

At the Folk Museum in an imaginary city there are a number of *tourist guides* supplied with mobile phones moving around in the museum. Our application can be used to get their attention in different ways. Furthermore the Folk Museum has a number of static *tourist spots* of special interest. Our application will make it possible for the *tourist* to get information by SMS of such a tourist spot.

We are going to augment the exercise with more functionality. We introduce an advanced service for the tourist, named “ASK” with an explicit question following the command. This question will be transmitted to all guides. They may respond either by an “ANSWER” or a “PASS”. The system will decide that one of those that respond “ANSWER” will physically walk to the asking tourist. See Figure 2 for details.

In Figure 2 there is a sequence diagram specifying a scenario for the ask&answer protocol.

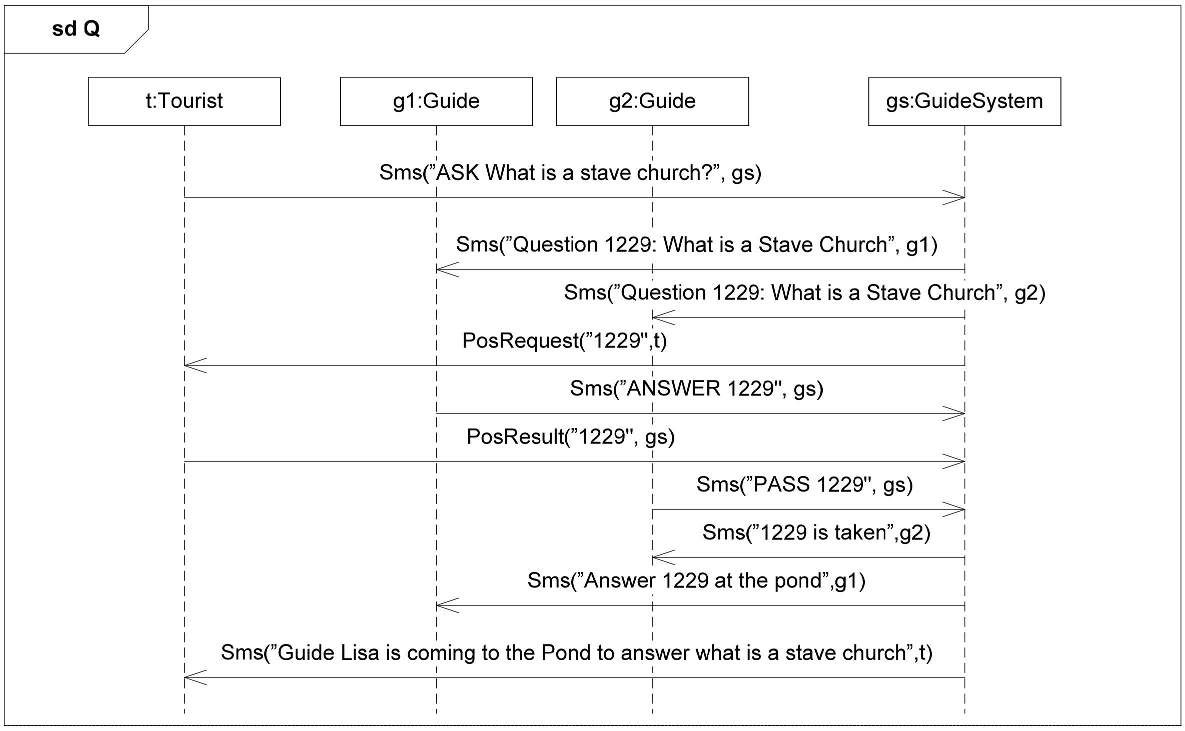
sd Q
t:Tourist
g1:Guide
g2:Guide
gs:GuideSystem
Sms(”ASK What is a stave church?”, gs)
Sms(”Question 1229: What is a Stave Church”, g1)
Sms(”Question 1229: What is a Stave Church”, g2)
PosRequest(”1229",t)
Sms(”ANSWER 1229", gs)
PosResult(”1229", gs)
Sms(”PASS 1229", gs)
Sms(”1229 is taken”,g2)
Sms(”Answer 1229 at the pond”,g1)
Sms(”Guide Lisa is coming to the Pond to answer what is a stave church”,t)

Figure 2One scenario for advanced asking and answering

We assume that the system has hotspots similar to ICU such that the tourist can be placed at the Pond from the positioning result. The number 1229 is arbitrarily chosen and represents the idea that there is a unique identifier generated by the *gs:GuideSystem* associated with this particular question. The second parameter of the *Sms* signal indicates the target of the signal.

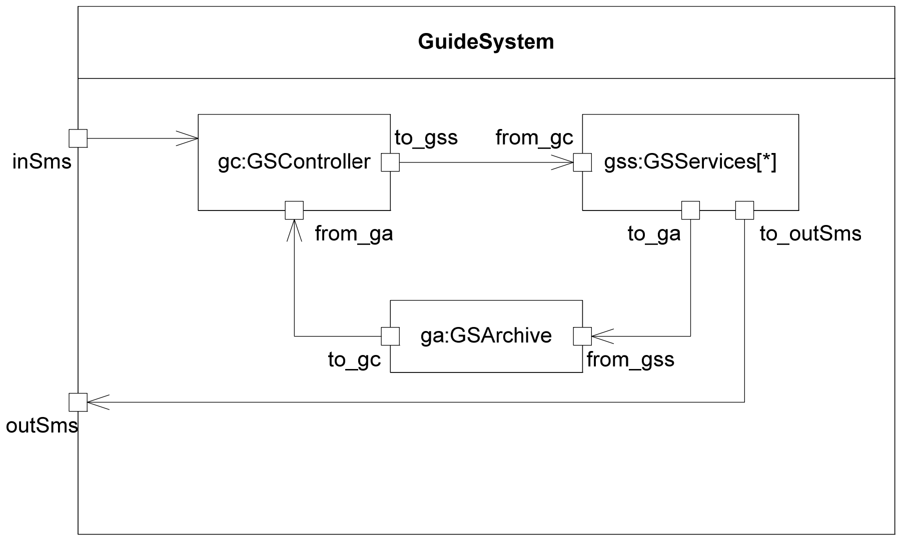

Figure 3 Composite structure of GuideSystem

2 a)

Define a sequence diagram *gs-Q* representing a decomposition of lifeline *gs:GuideSystem* for sequence diagram *Q* given in Figure 2 for the composite structure given in Figure 3. Every *Sms* signal to *ts* should create a new session. The port *to_gss* on *gc:GSController* is a routing port.

2 b)

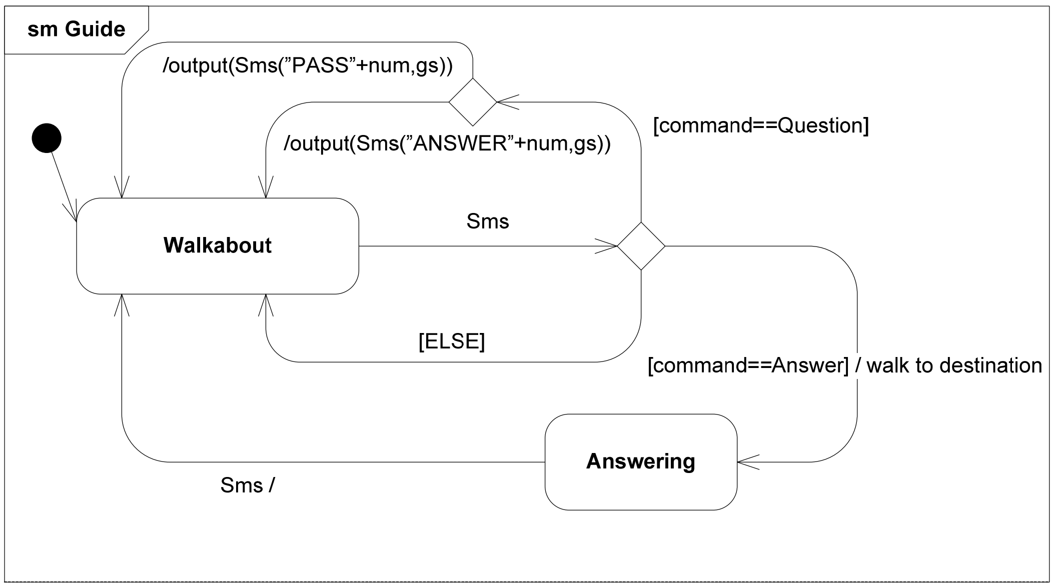
sm Guide
/output(Sms(”PASS”+num,gs))
[command==Question]
/output(Sms(”ANSWER”+num,gs))
Walkabout
Sms
[ELSE]
[command==Answer] / walk to destination
Answering
Sms /

Figure 4 State Machine for a Guide

Assume that we want to simulate a tourist guide e.g. for the purpose of testing our *GuideSystem*. In Figure we show a state machine as a definition of a *Guide*. We assume that the *Guide* will parse the text of the *Sms* and interpret the first word as a ‘command’. We furthermore interpret the output of an *Sms* such that *Sms(“ANSWER”+1229, gs)* will yield an *Sms* with text *“ANSWER 1229”* sent to *gs*.

Explain whether the state machine in Figure 4 is consistent with the sequence diagram in Figure 2.

2 c)

Discuss if there are any problems with the definition of *Guide* in Figure 4 and how you would modify the definition to cope with the problems.

2 d)

How many reception events take place on the lifeline *gs*? Explain your answer.

2 e)

What is (are) the first event(s) of Figure 2? (If there is more than one possibility, list all the possibilities.) Explain your answer.

2 f)

What is (are) the last event(s) of Figure 2? Again explain your answer and if necessary list more than one event.

2 g)

Explain how to update the diagram in Figure 2 by adding only message arrows to have exactly three possible first events and two possible last events?