Drop 1 i inf5150

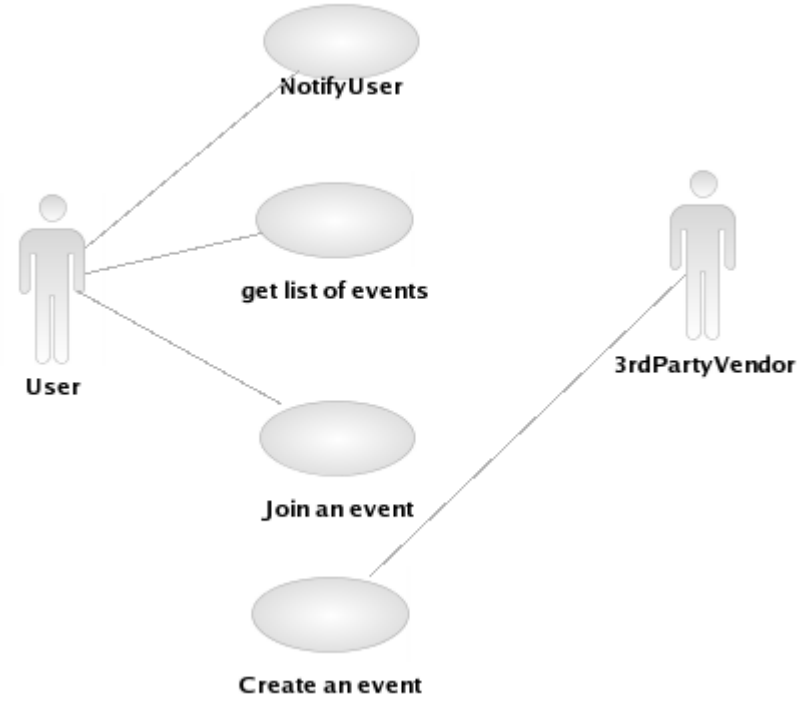
Gruppe 6

Frode Jensen
Jenny Hougen
Geir Skjøtskift
Erik Nilsen Haga

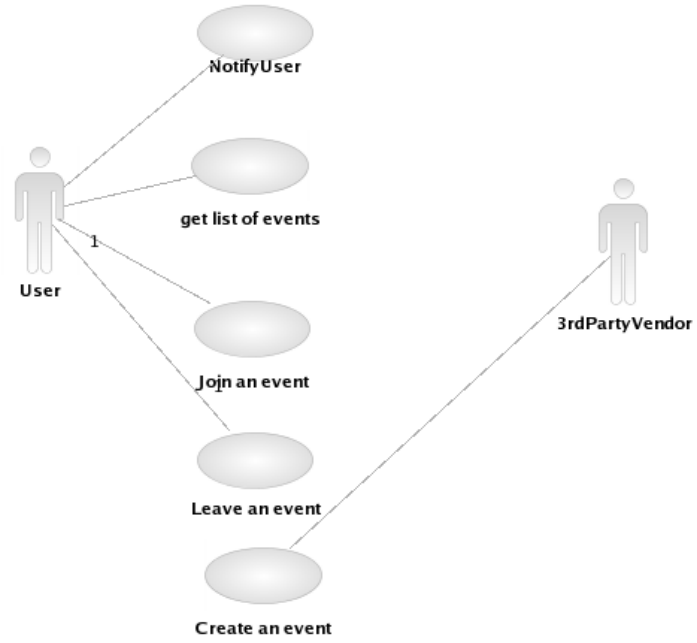
14.oktober 2005

Multiple Blind Date

1. Use case diagram



This is the usecase diagram for the base case.



This is the usecase diagram for the refined (modified) case.

2. Introduction

We are creating a system that is going to handle events for different groups. The user can get a list of available events by sending a SMS to the system. The user can then join any of these events by sending another SMS specifying what group to join. One user is treated as separate persons in each event. New events are created by the owners of the system, by sending a SMS to the system. The owners of the system is in this case us. When it is 30 minutes until the event is to take place the system gets the position for each user, gets routes from these positions and to the place of the event, and sends this information to the corresponding user including the details of the event.

- Create events: Events are created by a third party vendor by sending an SMS to the system.
- Join event message: The system receive SMS from people that want to join an event.
- Notify event message: The system is sending an SMS with time, place and details on how to get to the event.
- List events: The user sends an SMS to the system to get an list of possible events to join.

3. Assumptions

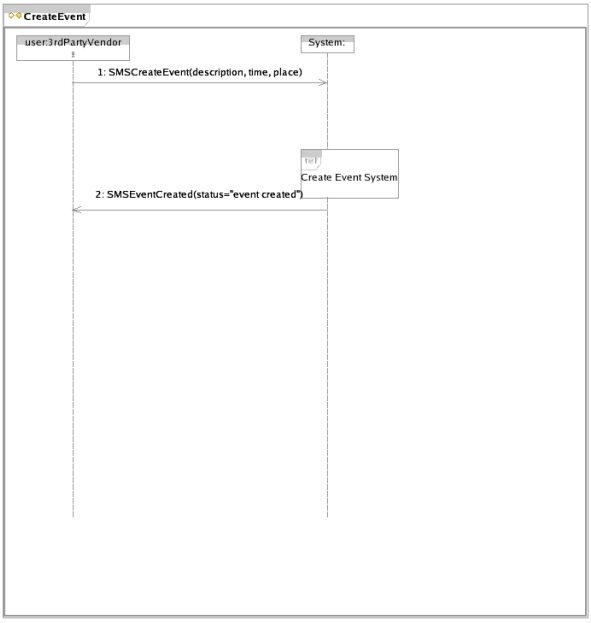
It is several aspects that are not defined in the exercise explanation, e.g. could a person belong to several groups? Hence we need to presume the following for our system:

- One person will only have one telephone number in our system and this is the unique identifier for the person.
- The system sends a notify about route (bus stop, bus number) and place 30 min before the event is taking place.
- The only people we consider as third party vendors is us.

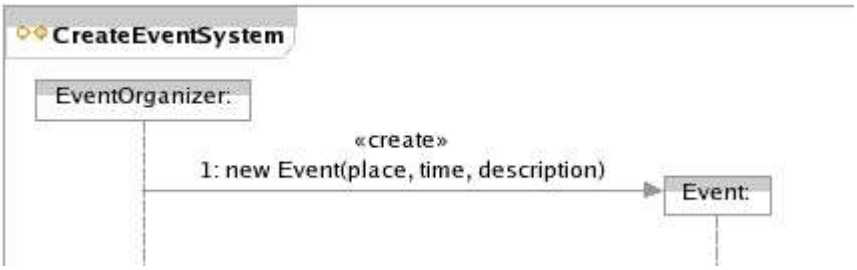
4. Original MBD system

The original MBD system is the system as specified in the original assignment text. The system will not decide time and place for the event. The time and place is decided by the event creator. When the system receive an join SMS from a person, the system will find the persons position and find bus stop and bus nr from that position. Then the system will send an SMS to the person with all the information. The system will:

- Create event with time and place.
- Add a person to a event group
- Find the persons position
- Find the bus and nearest bus stop for the person.
- Send an SMS to the person with the time, place, description, bus stop and bus nr to the person.



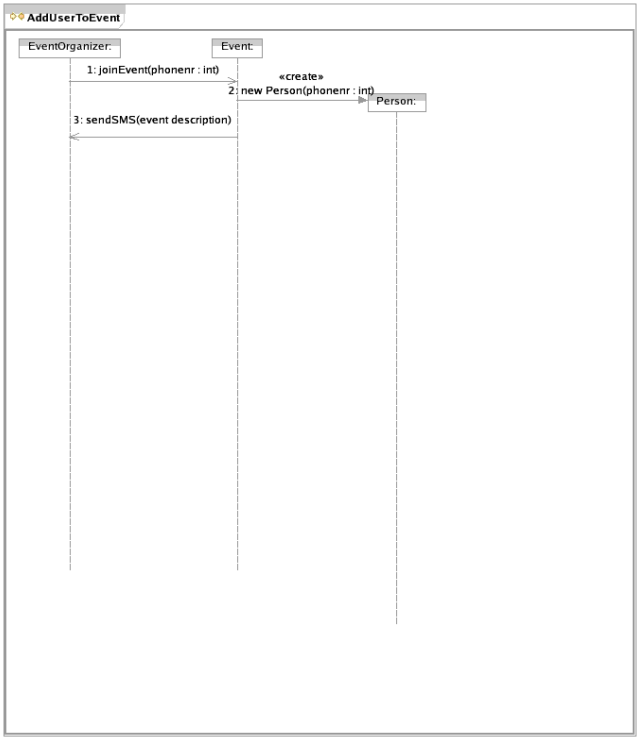
This diagram shows the basic messages to and from the 3rd. party vendor (us) when creating an event. We send an event to the system, the system does it's magic, and notifies us by sms of the status of the event. No validation of user vs. administative rights is done.



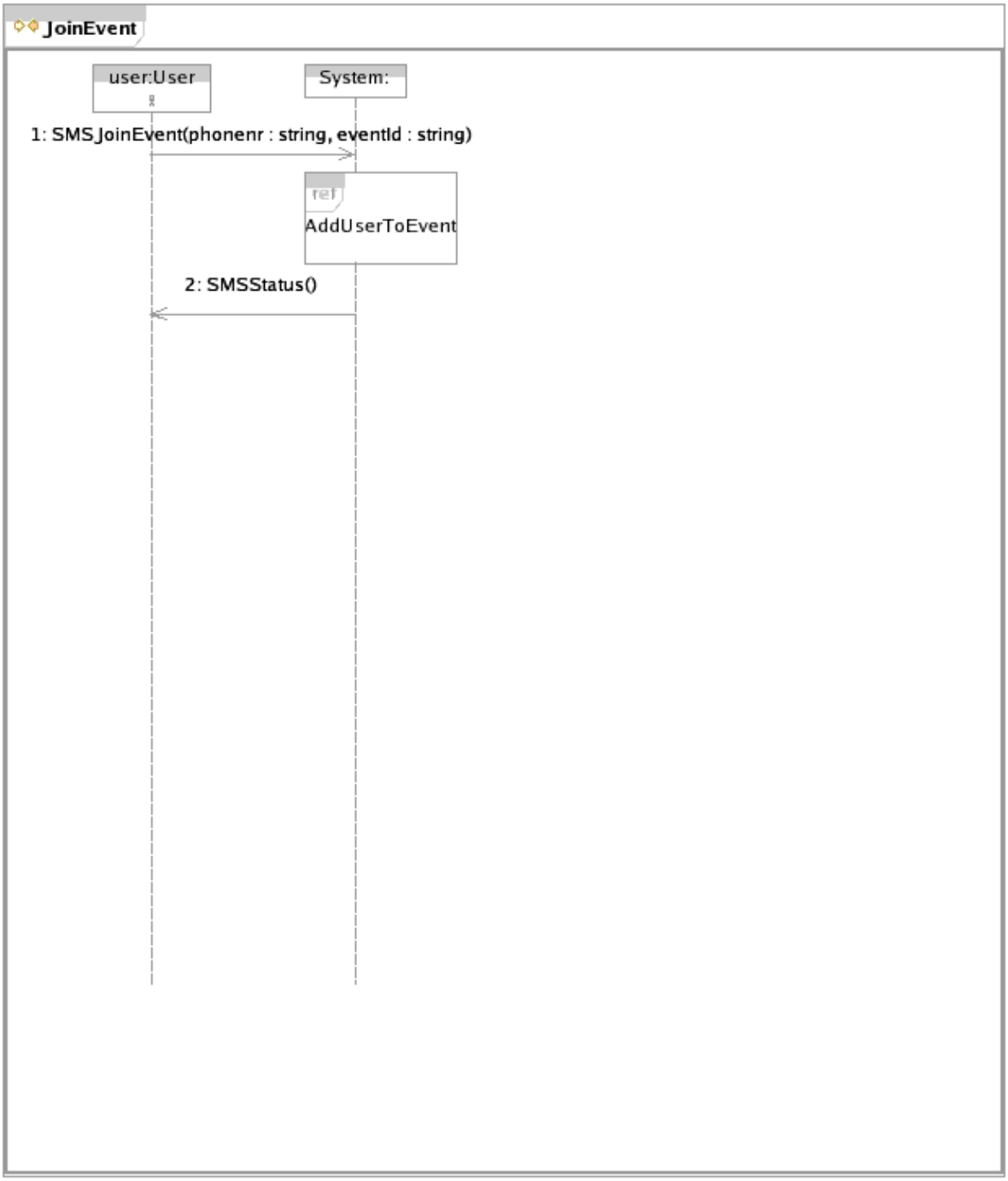
This diagram describes the basic trace when the EventOrganizer actually creates the event. This concludes the diagrams describing the actual creation of an event.

The next couple of diagrams describes the joining of an event by a user.

This diagram describes an overview of the messages sent to an from the system by the user when joining an event. The user sends a “join event #xx” to the system, the systems again does it's magic and returns an Join successful. In the base case no error handling is done concerning non existing events. We assume the preconditions is correct.

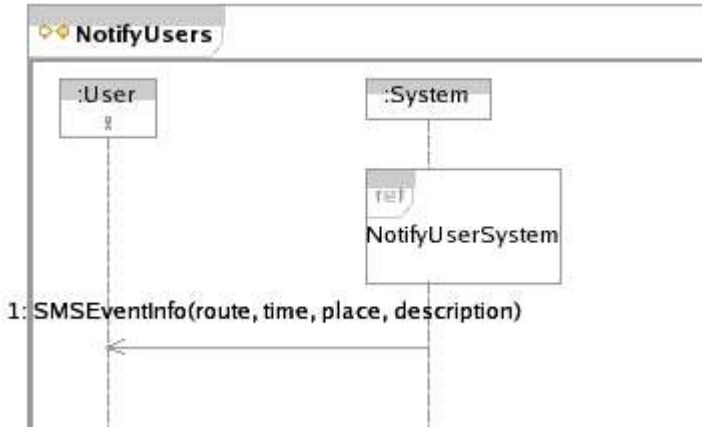


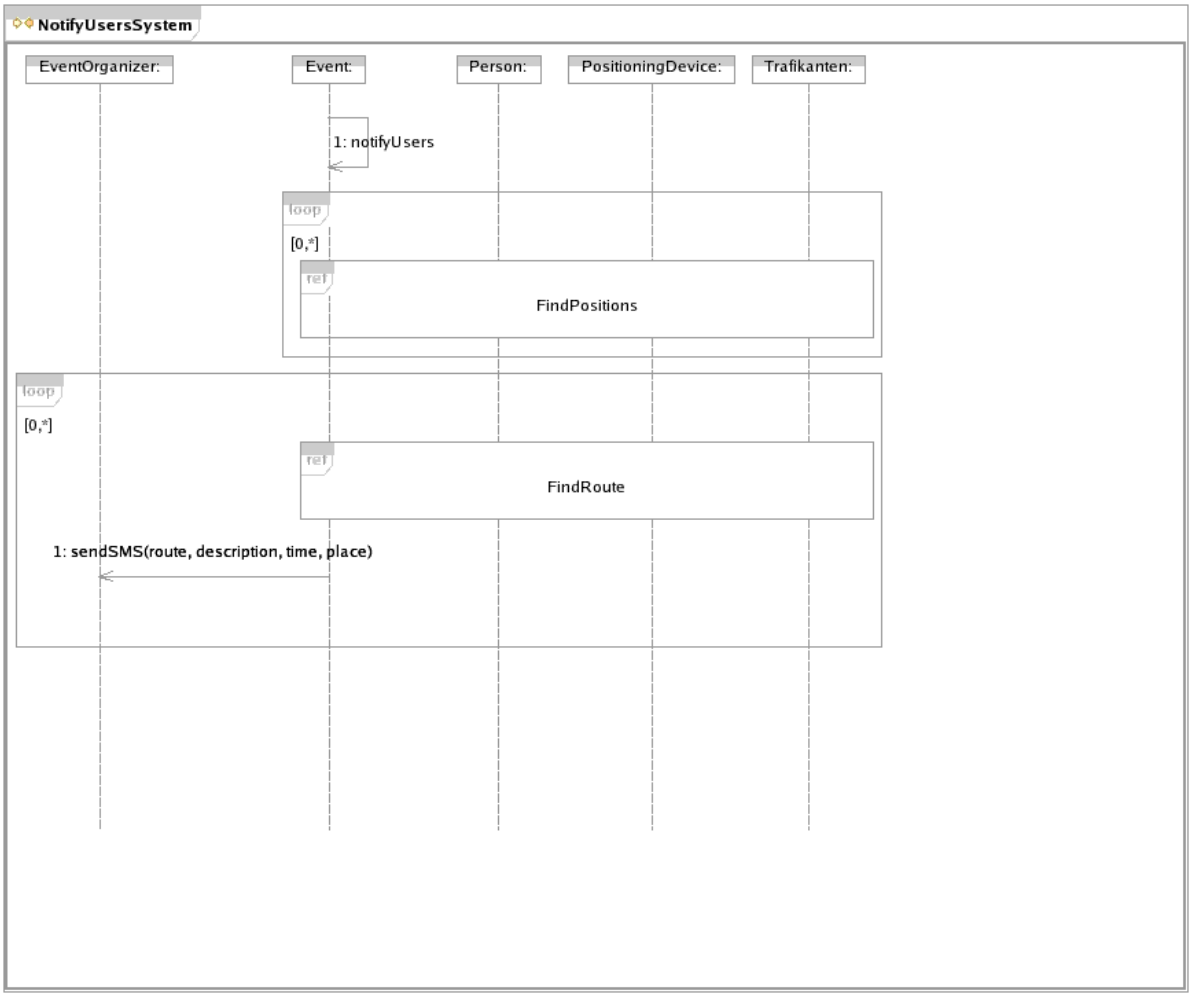
This diagram describes the internal messages when actually adding the user to the system. The event organizer will send a message to the event with the users phone number. The event will then create a new person corresponding with this number and keep a reference to this user in it internal data structure for later use. The event will then send a message back to the event organizer referring the status of the operation.



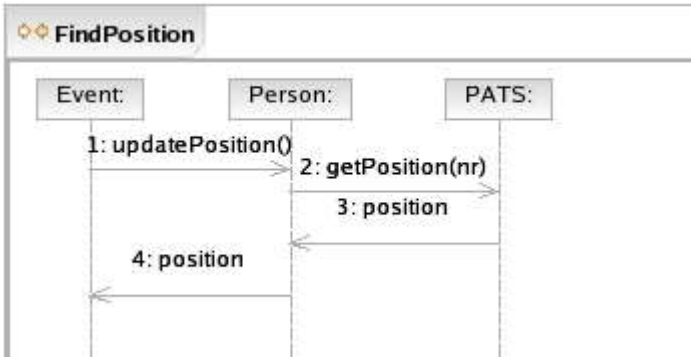
This concludes the adding of a user to the system.

The next couple of diagrams describes the functionality of the system when the time to leave has expired.
This diagram describes the overall communication of the system when it is time to leave. The event will trigger, find the routes from the users position to the event place and send this message to the users.

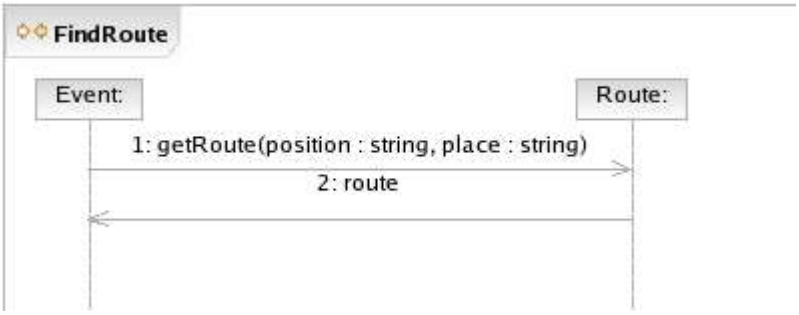

NotifyUsers
:User
:System
ref
NotifyUserSystem
1: SMSEventInfo(route, time, place, description)


NotifyUsersSystem
EventOrganizer:
Event:
Person:
PositioningDevice:
Trafikanten:
1: notifyUsers
loop
[0,*]
ref
FindPositions
loop
[0,*]
ref
FindRoute
1: sendSMS(route, description, time, place)

This diagram describes the overall internal messages and parts of the system when the time to leave for the event has arrived. The system will first of all find the current position of the users, and the call the “route-finding-system” to retrieve the routes for the users.



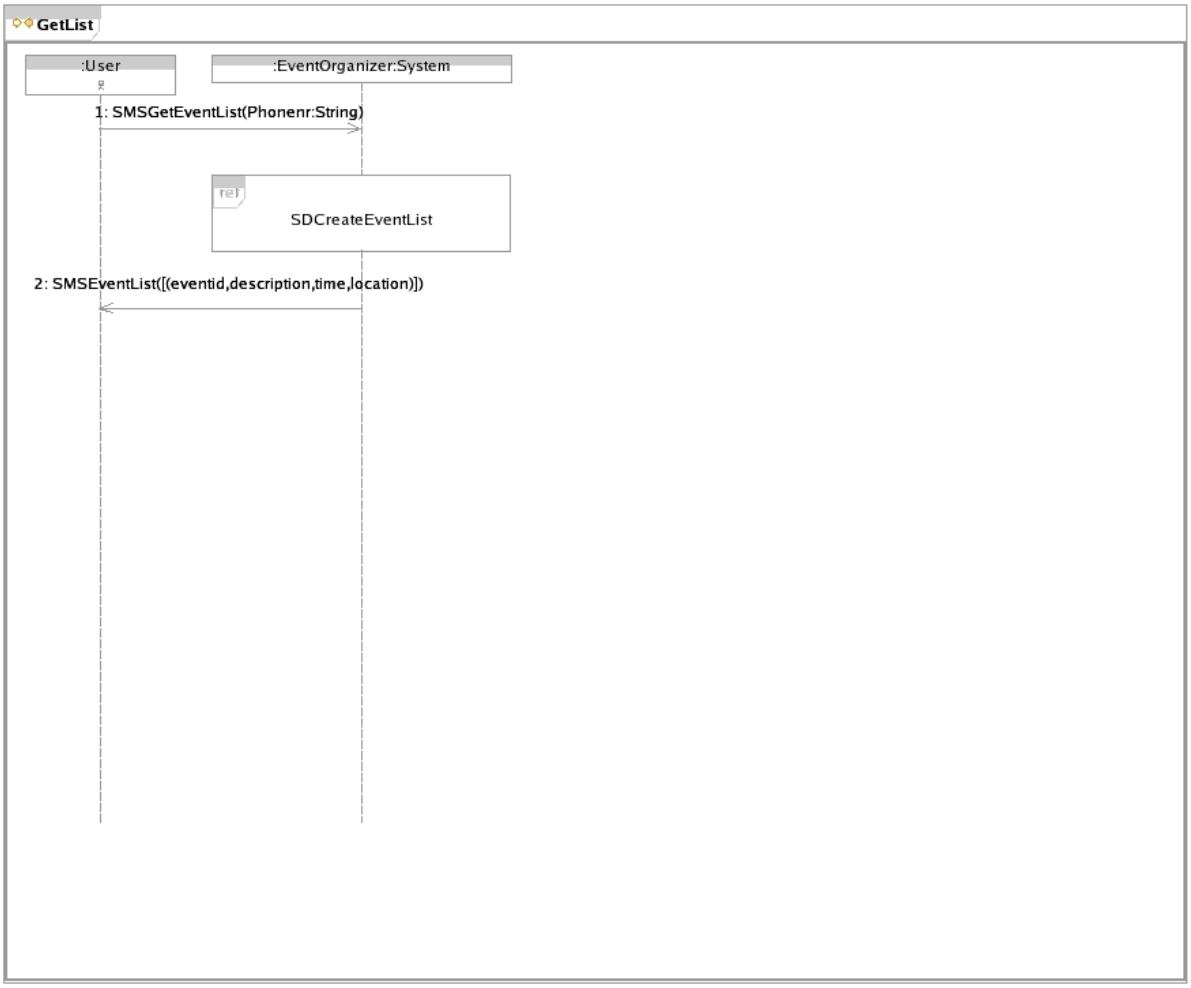
This diagram describes the internal system calls required to find the position of a user. This diagrammed is looped in the preceding diagram. The system makes use of PATS to do the actual positioning.



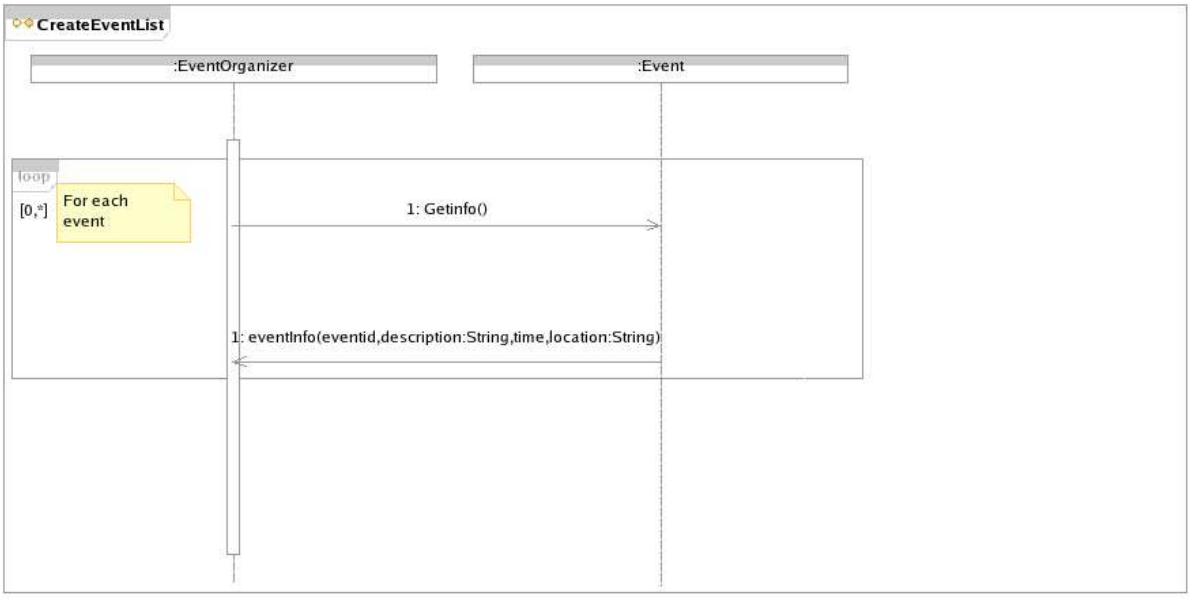
This diagram describes the internal system calls necessary to retrieve the routes the user needs to get from one place to another. (in our case, from his current position to the place of the event). The event calls the route (in our case “trafikanten”) to get the desired result.

This concludes the calls necessary to give all users a route to the location of the event.

The next diagrams describes the listing of event. The user will use this list to retrieve the corresponding number for the event to which he want to take part of.



This diagram describes the actual basic messages with the system necessary for the user to get a list of events.



This diagram describes the internal working of the system when returning a list of events to the user. The event organizer loops through all events in the system. Retrieves the information of the event and creates a message for the user based on this data. If the event has not received a place yet, the place will be returned as TBA ("To be announced").

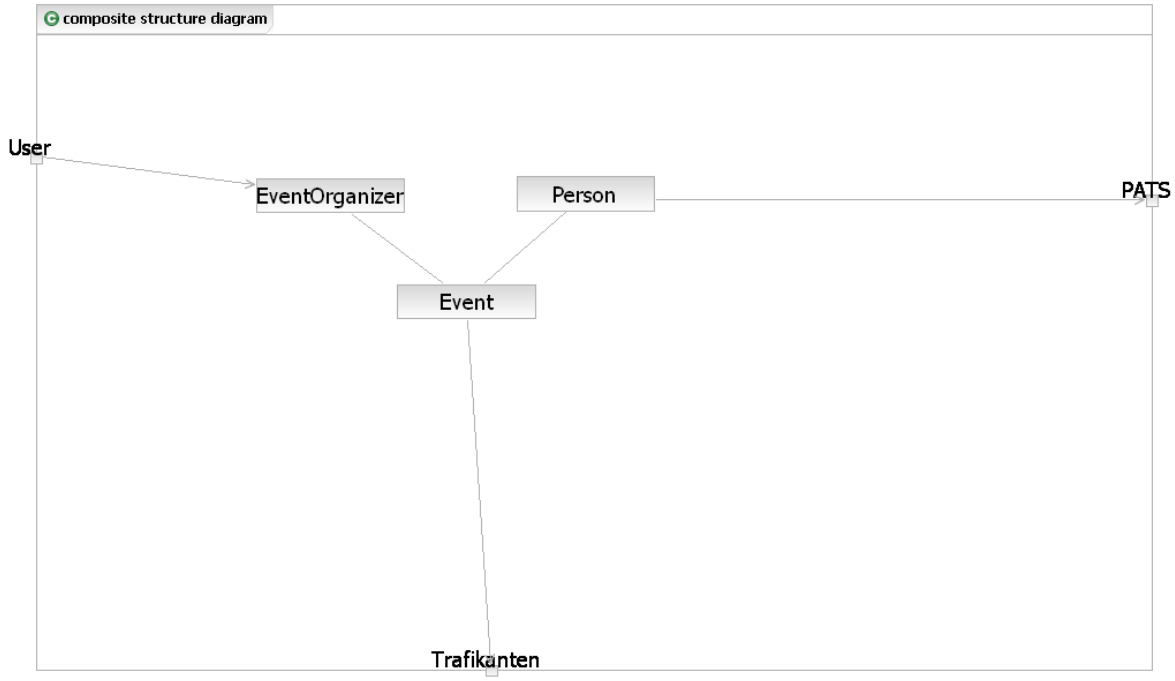
This concludes the diagrams necessary to return a list of events.

4. Class diagram

The class diagram consists of 5 classes:

- EventOrganizer: This is the controller of the system. It receives the SMS from the persons and it sends SMS to persons.
- Event: This class adds new events to the system and trigger the event when it is time to send an SMS to persons in a group.
- Person: This class is responsible to update its position when the event asks for the persons position.
- PositionFinder: This class is called by the person and finds the current position for the person. (It could be the PATS system).
- Route: This is responsible for find the bus stop and bus number from a position. It uses Trafikanten to do this.

5. Composite structure diagram



Figur 1 Composite structure diagram of MBD

This is a composite structure diagram that describes the communication between the system components and the external environment. We use PATS to find the users position. We use Trafikanten to find the nearest bus stop and bus nr for the user.

6. Sequence diagram (Modified system)

1. Create event
   1.1. Create event with time and place: The event creator manage the place and time for event. The time and place for the event is set by the event creator.
   1.2. Create event without place: The system sets the place of the event. Place is found by the most optimal meeting place for the participants in the group.
   1.3. Create event without time:
2. Join event
3. Notify user about event
   3.1. Send SMS now
   3.2. Send SMS later

3.3. Find meeting place

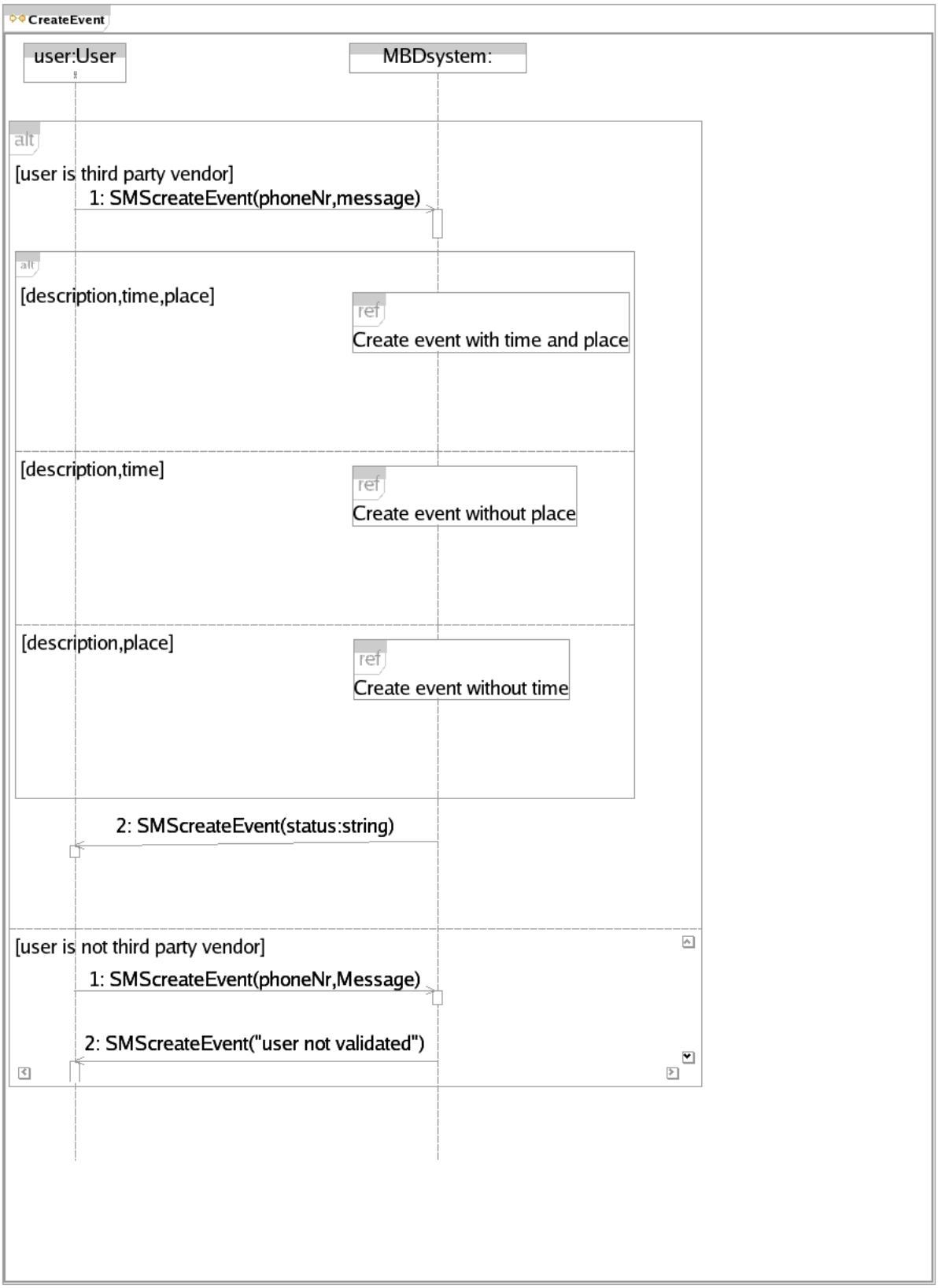


Figure 2 Overview create event

Figure 2 gives an overview of the create event functionality in the MBD system. Before the system can create a new event it has to authenticate the user as a third party vendor. One possibility to authenticate the user is to validate the phone number against a list of legal phone numbers.

When the event organizer receives the SMScreateEvent it interprets the message and finds out which createEvent it should trigger. The message could contain three possibilities:

- description, time and place
- description, time
- description, place.

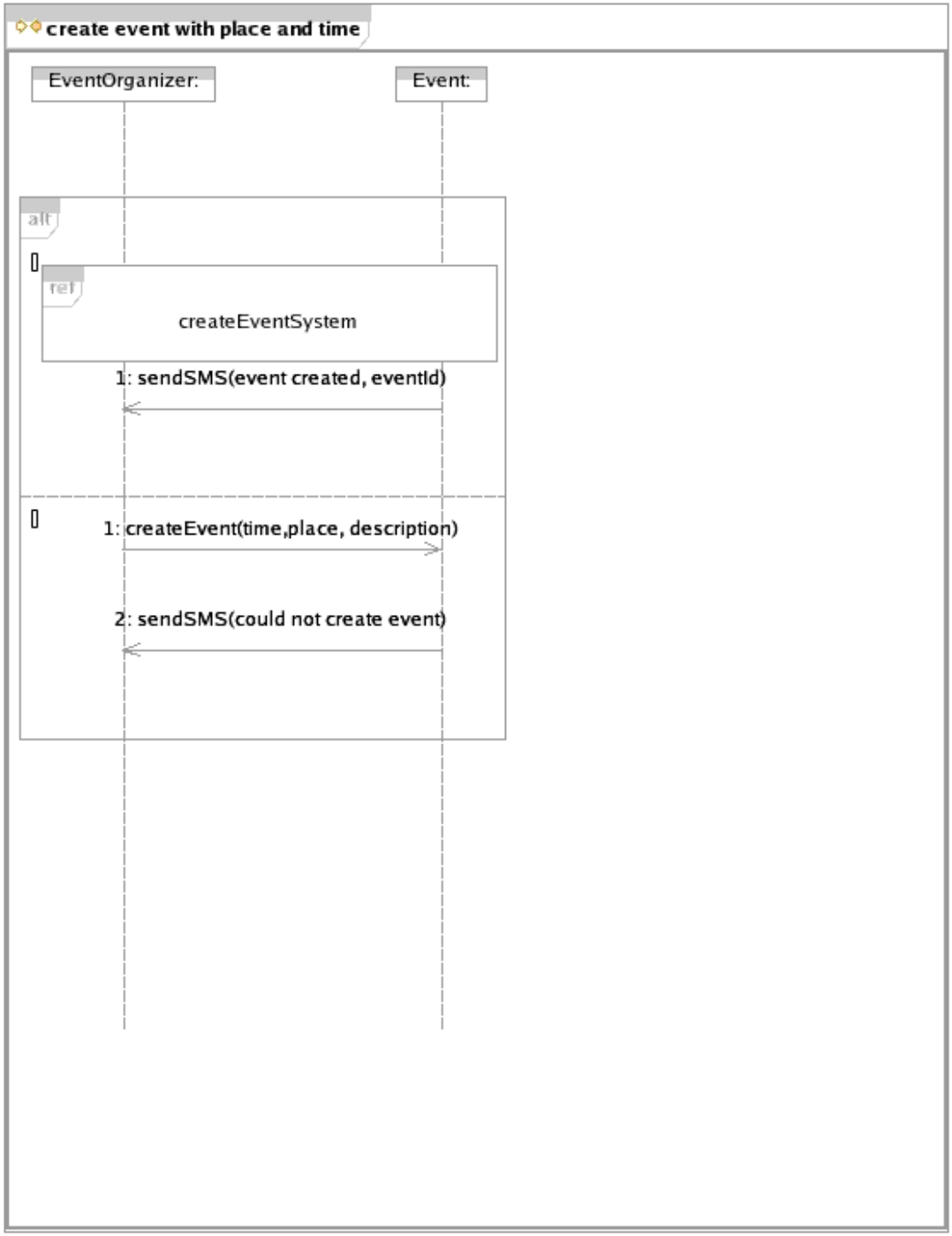


Figure 3 Create event with time and place

Figure 3 describes how to create an event when time and place is decided of the third part vendor. It has two alternatives the system could create the event or the system was not able to create the event.

Figure 4 Create event without place

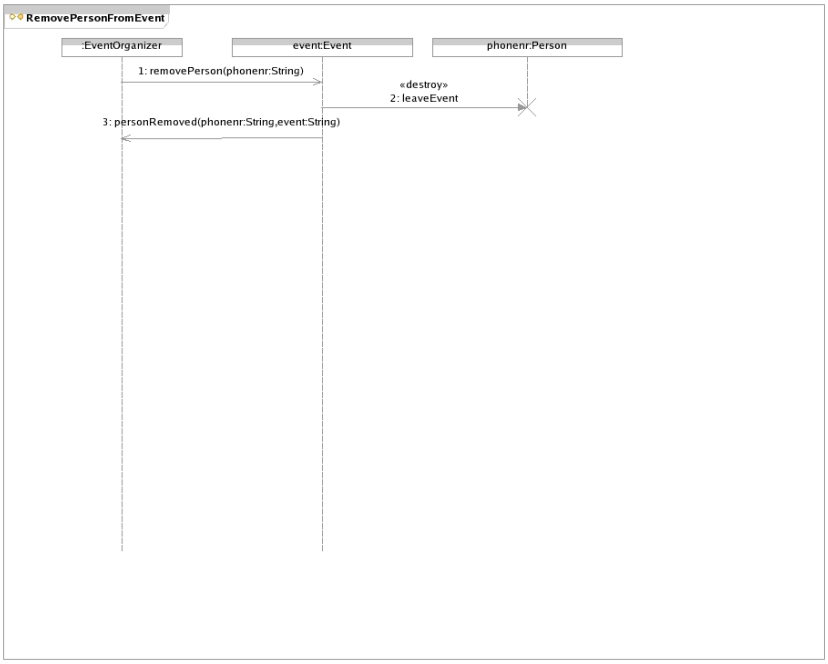
Figure 4 describes how to create an event when place is not decided by the third party vendor. The system should then find the best place for the event, but this is done when the

system shall notify the user about the route. This case has also to alternatives as mentioned in figure 3.

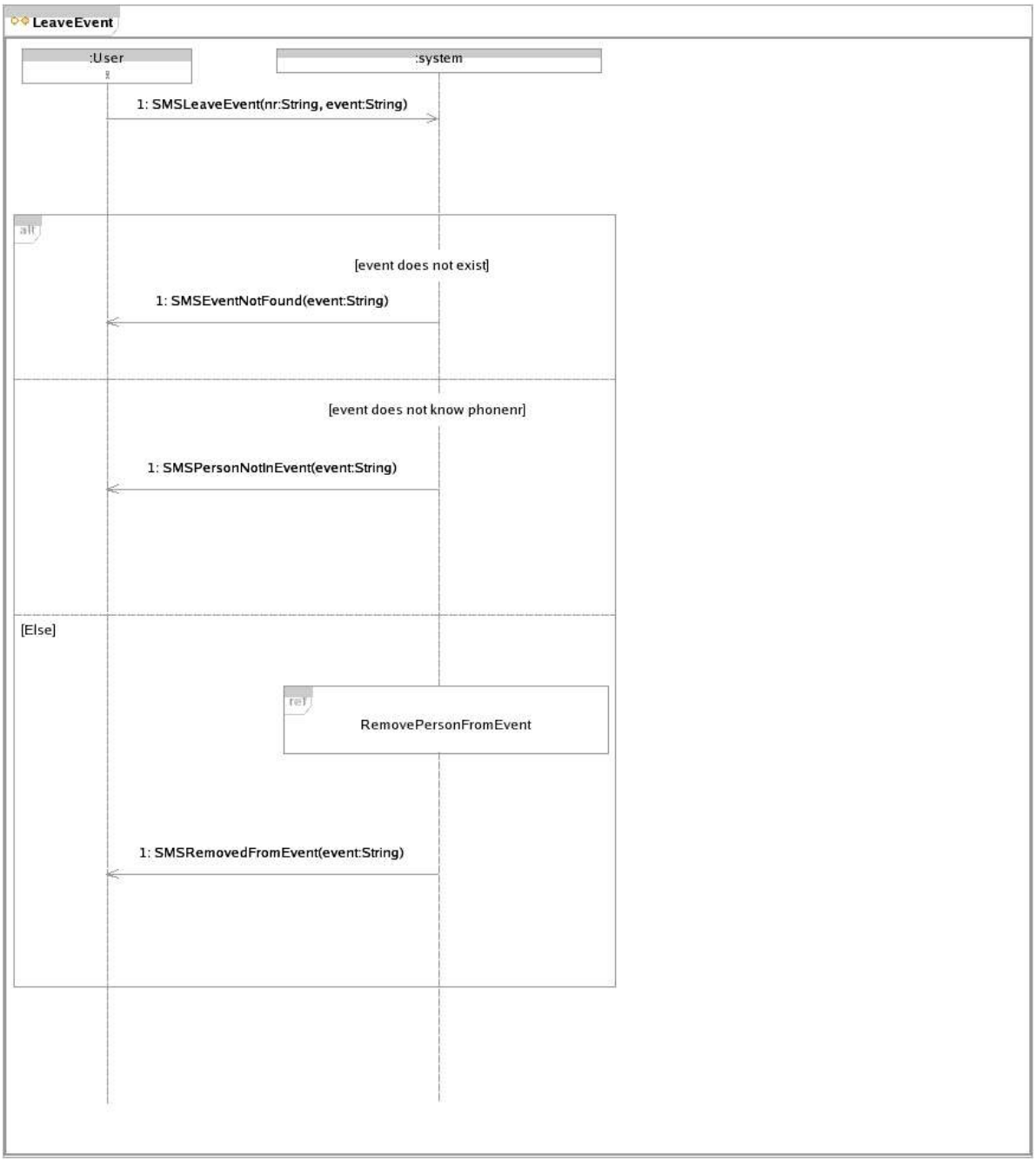
Figur 5 Create event without time

Figure 5 describes how to create a new event when the time is not given from the third part vendor. The time of the event should then be sat to the current time(NOW) + 30 min. This figure has two alternatives; able to create the event or not able to create the event.

The next couple of diagrams will describe the internal working of the system when a user attempts to withdraw from an event.

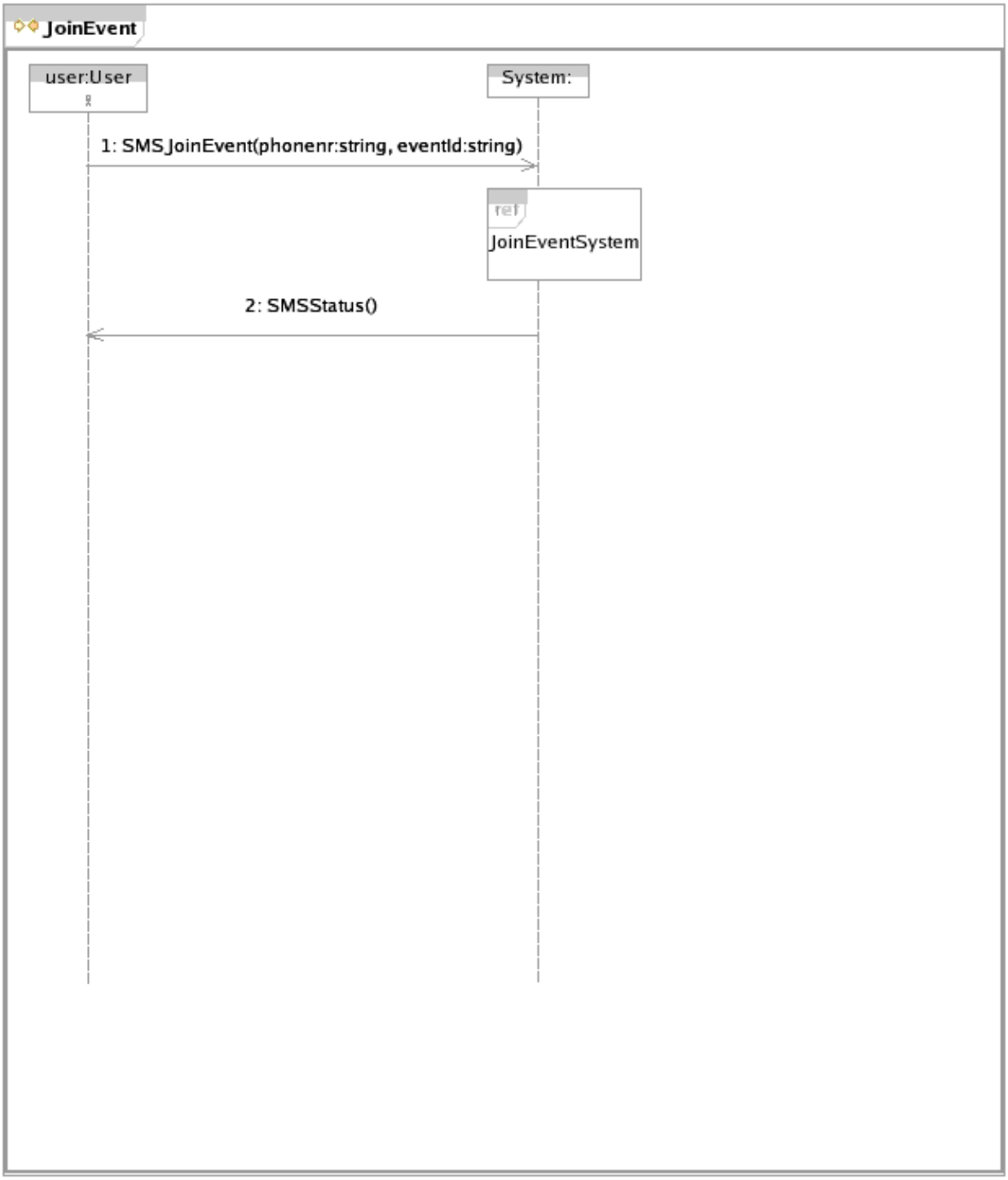


This diagram describes the messages sent to and from the system when a user withdraws from the event. If the event the user wants to withdraw from does not exist or the event from which the user wants to withdraw does not know of the user, this will negates the actual withdrawal. The system will remove the user from the event and send a status message to the user in question. No one else will be notified of this case.

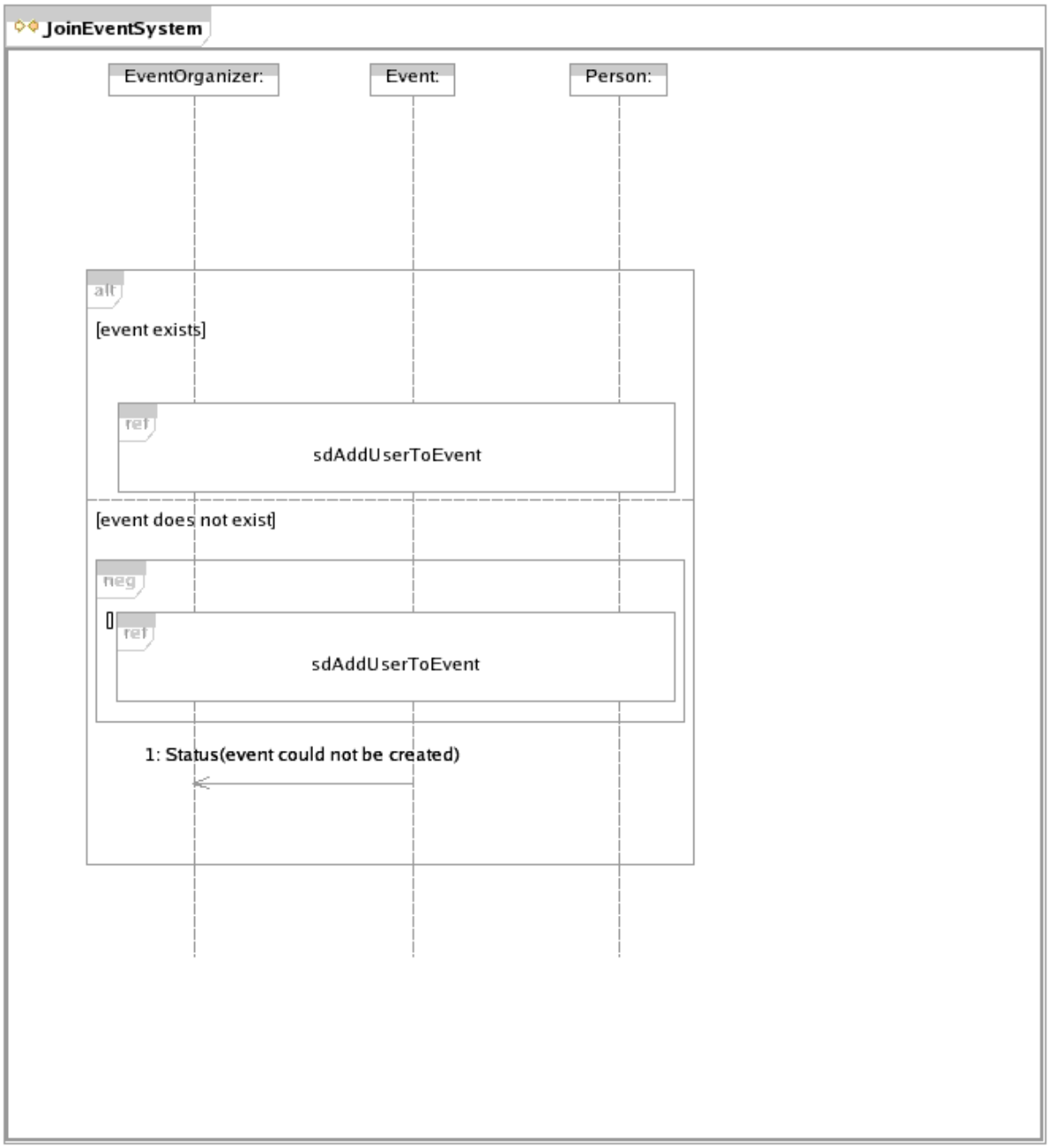


This diagram describes the internal messages of the system when removing a user from the event. The EventOrganizer will tell the event do disband the user. The event will then destroy the person object and detach its reference to this object. The event will the notify the event organizer that this is done and the EventOrganizer will in turn notify the user.

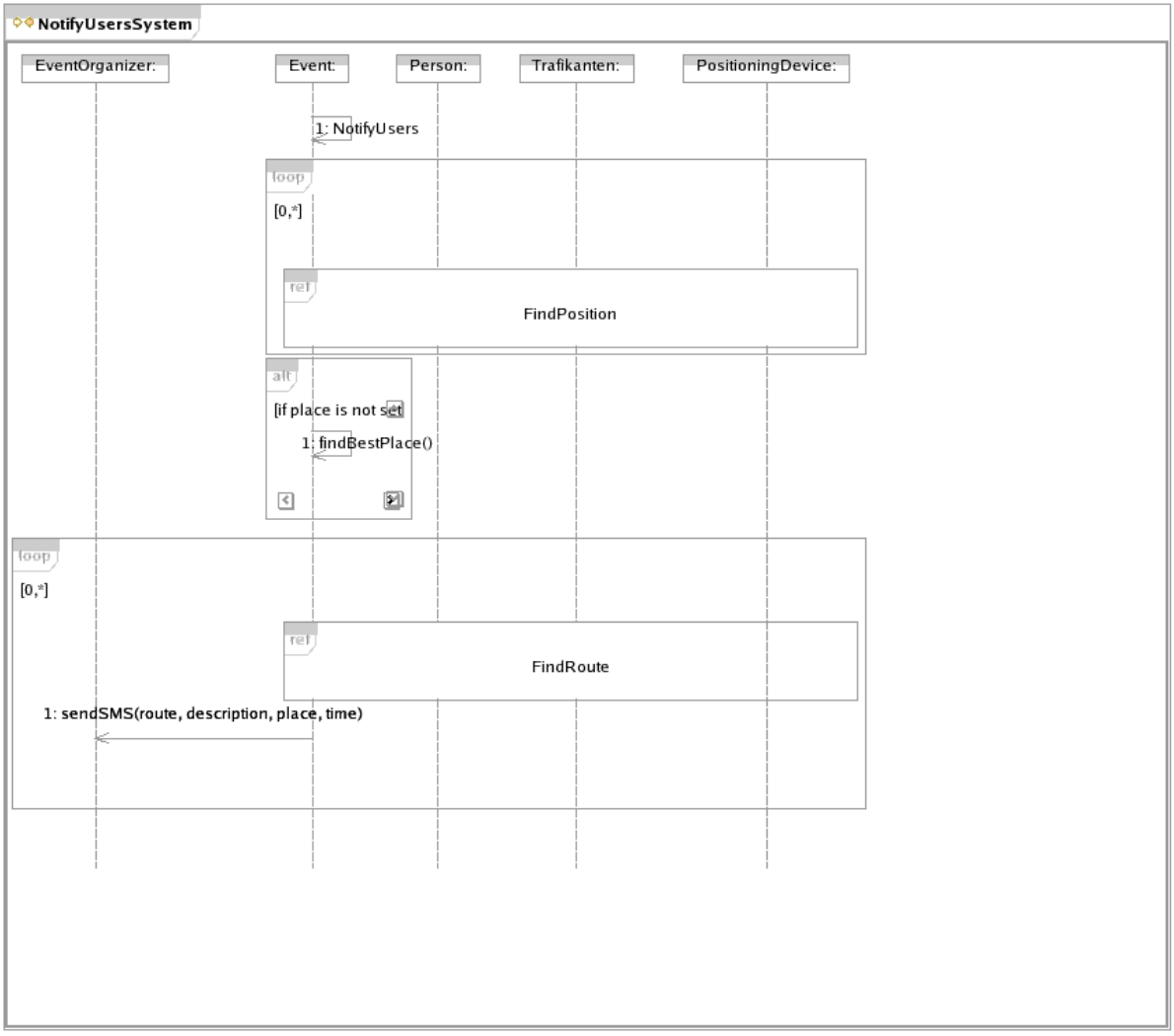
The next diagrams is a refinement of Join events



This diagram shows the overview of the messages passing to and from the system when a user joins an event.



This diagram shows that there is the possibility that the event that the user is trying to enter does not exist. In this case the user will of course not be added. If, however, the case exists the user will be added to the event.



This diagram describes the chance that the place is not set the place most suitable to the group of participants will be chosen. In the case of an event taking place in less than 30 minutes, the place must be given when the event is created, since there are no users to choose a “best place” for.

7. **The modified system as a refinement of the base system**.

Attachment (written in latex, the rest in oo.)

Proving that our refinement actually is a refinement of the basic system

14th October 2005

In the following proof SD should be as read sequence diagram.

Let S_1 be the basis system and S_2 the modified system. Let t be a token that will be used as a placeholder for traces that is not of interest for this proof.

S_1 contains the SD's CreateEvent, GetList, JoinEvent, and NotifyUsers. S_1 also includes the SD's refered by these. Let CE, GL, NU, and JE respectivly be these diagrams and pos(s) be the positive traces of SD's. Likewise let neg(s) be negative traces of s.

In order to describe the system we choose to consider seperate Sequence diagramms as joined in diffrent part of an alt construct. Since S_1 does not contain any xalt constructs its sematic $[\![S_1]\!]$ is given by a simple interaction obligation $\{(p_1, n_1)\}$ where

- $p_1 = pos(CE) \cup pos(GL) \cup pos(NU) \cup pos(JE)$
- $n_1 = neg(CE) \cup neg(GL) \cup neg(NU) \cup neg(JE)$

S_2 contains the SD's CreateEvent, GetList, JoinEvent, NotifyUsers and LeaveEvent. S_2 also includes the SD's refered by these. Let CE', GL', NU', JE' and LE' respectivly be these diagrams. As S_1 S_2 does not contain any xalt constructs. $[\![S_2]\!]$ is given by a simple interaction obligation $\{(p_2, n_2)\}$ where

- $p_2 = pos(CE') \cup pos(GL') \cup pos(NU') \cup pos(JE') \cup pos(LE')$
- $n_2 = neg(CE') \cup neg(GL') \cup neg(NU') \cup neg(JE') \cup neg(LE')$

Because $[\![S_1]\!]$ and $[\![S_2]\!]$ both contains a singel interaction obligation we have that $S_1 \rightsquigarrow S_2$ iff $(p_1, n_1) \rightsquigarrow (p_2, n_2)$. By definition this is equivalent to

$$n_1 \subseteq n_2 \wedge n_1 \subseteq p_1 \subseteq p_2 \cup n_2 \qquad (1)$$

Since $GL = GL'$ including the SD's referred from these SD's $pos(GL) = pos(GL')$ and $neg(GL) = neg(GL')$.(2) Although NU=NU' SD's referred from these documents is changed. We will prove that(3);

- $pos(CE) \subset pos(CE') \wedge \text{neg(CE)} \subset neg(CE')$
- $pos(NU) \subset pos(NU') \wedge \text{neg(NU)} \subset neg(NU')$
- $pos(JE) \subset pos(JE') \wedge \text{neg(JE)} \subset neg(JE')$

(2) and (3) is sufficent to satisfy (1) and $S_1 \rightsquigarrow S_2$ will follow.

$pos(NU) \subset pos(NU') \wedge \text{neg(NU)} \subset neg(NU')$

These SD's refer the SD NotifyUsersSystem which is changed in the refined system. Let NUS refer to the SD in the basis case and NUS' the SD in the refinement. Because NUS is equal to NUS' except for the inclusion of an alt construct, $pos(NUS') = pos(NUS) \cup t$. Likewise for the negative traces. $pos(NU) \subset pos(NU') \wedge \text{neg(NU)} \subset neg(NU')$ therfore holds.

$pos(JE) \subset pos(JE') \wedge \text{neg(JE)} \subset neg(JE')$

JE refers to addusertogroup wheras JE' refers to JoinEventSystem JoinEventSystem contains a loop construct that refers to addusertogroup and has not got any messages outside the construct. We have therfore $pos(JE') = pos(JE) \cup t$ and the same for negative traces. $pos(JE) \subset pos(JE') \wedge \text{neg(JE)} \subset neg(JE')$ follows.

$pos(CE) \subset pos(CE') \wedge neg(CE) \subset neg(CE')$

CE' contains an alt construct at the point where CE refers to Createeventsystem shortened CES. The alt construc in CE' contais an option that refers to CreateEventwithtimeandplace witch only contains an alt structrue that has an option that refers to CES. As above it follows that $pos(CE) \subset pos(CE') \wedge neg(CE) \subset neg(CE')$.

QED.