UNIVERSITETET I OSLO
Institutt for informatikk

Multiple Blind Date
inf5150 group 4

Aleksander Bai
Frank Petter Larsen
Kristian Furulund
Frank Davidsen
Unni Løland

13. oktober 2005

TABLE OF CONTENTS

1 Introduction

Multiple Blind Date is a new way for users to meet with likeminded people at interesting events.

This document describes our new service in some detail as well as several possible variations and extensions, reflecting that we are still exploring new ideas.

1.1 Overview of the document

Chapter 1 gives a brief overview of the system and provides the foundation for understanding the rest of the document. Chapter 3 describes the basic services to some detail. Chapter 4 explores several variations of the system and discusses these as refinements of the original design.

2 System Overview

The system consists of two major services, which we will now describe briefly and detail throughout the document. The first service allows the user to sign up for upcoming events, while the second service sends reminder messages and gives the user road directions a suitable time prior to the event. The system will direct the user to the nearest meeting place.

2.1 Signing up for events

Users of the service joins meetings by sending an SMS message to our dedicated phone number. As an example, sending the message "Go for a beer today at 8 PM" to our number will sign the user up with other users also wanting to go for a beer at that time today. Notice that the message only needs to include the time the user wishes to go to a meeting, as well as the desired subject. This is sufficient to team the user up with likeminded people.

Several variations of this basic service are explored later in this document. For example, if the user would like to go for a beer at a particularly good bar he knows very well, it is possible to extend the system to allow such user requests.

2.2 Sending reminders

An appropriate time before an event begins, the system will notify every user - called participant in this document - that has signed up for the event by sending SMS messages to them. These messages will not only remind the users of their appointment, but will also give road directions describing the best way to travel to the meeting place from his current location. In order to offer this very personalized service, we will use the users cell phone to track his position and discover the optimal route to the meeting.

This service is offered in cooperation with Telenor, which provides the possibility to locate users by their cell phones, and Trafikanten, which provides all the information for how to best transport the user.

2.3 General assumptions

The precise definition of an event has been a subject for many discussions. But in the end we came to an agreement. For the definitions, let’s use an example. One event with a place

associated with it can look as this: ”coffee Monday 7pm Bislett”. The definitions we have made are:

- eventType: the type of event. The eventType in the example is ”coffee”
- time: the time of the event. In the example it is Monday 7pm
- event: the event is ”coffee Monday 7pm”. Because the place may vary, we have defined the event to be the time + eventType
- place: the place is the location for an event. In the example this is “Bislett”

For the original diagrams, those done before the refinements, we have assumed that one event has only one place associated with it. In the refinements we supplement and open up the possibility for having more than one place associated with one event.

3 Superior sequence diagram

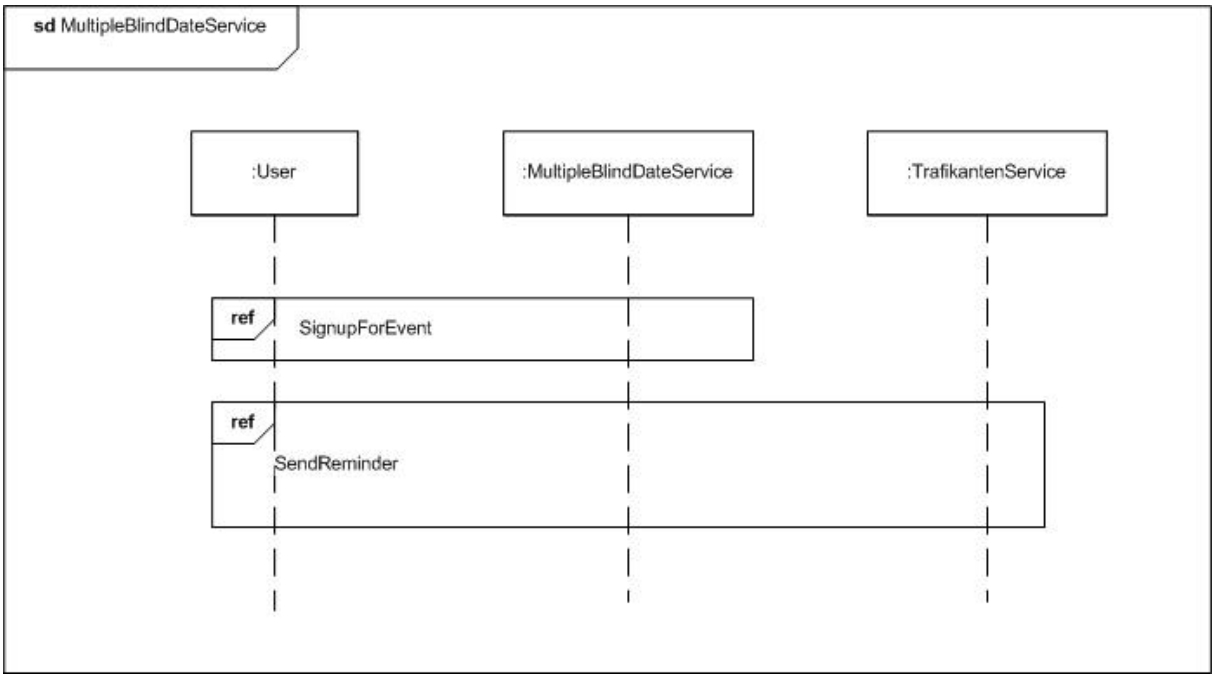


Figure 1 Superior sequencediagram

The superior sequence diagram shows how things relate to each other in the system (see figure 1). From this diagram we have done decomposition for each of the two functionalities; SignUpForEvent and SendReminder. But before we look at these diagrams, we will give an overview of the general assumptions we have done

3.1 SignUpForEvent

For the functionality SignUpForEvent we have done a decomposition for 3 levels.

3.1.1 Level 1 decomposition

See figure 2 for the diagram SignUpForEvent. The main interaction for this functionality is that a user sends a SMS to the Multiple Blind Date Service, and then the user is registered to the given event.

Assumptions for sendReminder level 1

Here we have modeled the "happy day" scenario. This means that the user just send the registration-message, and receives a confirmation.

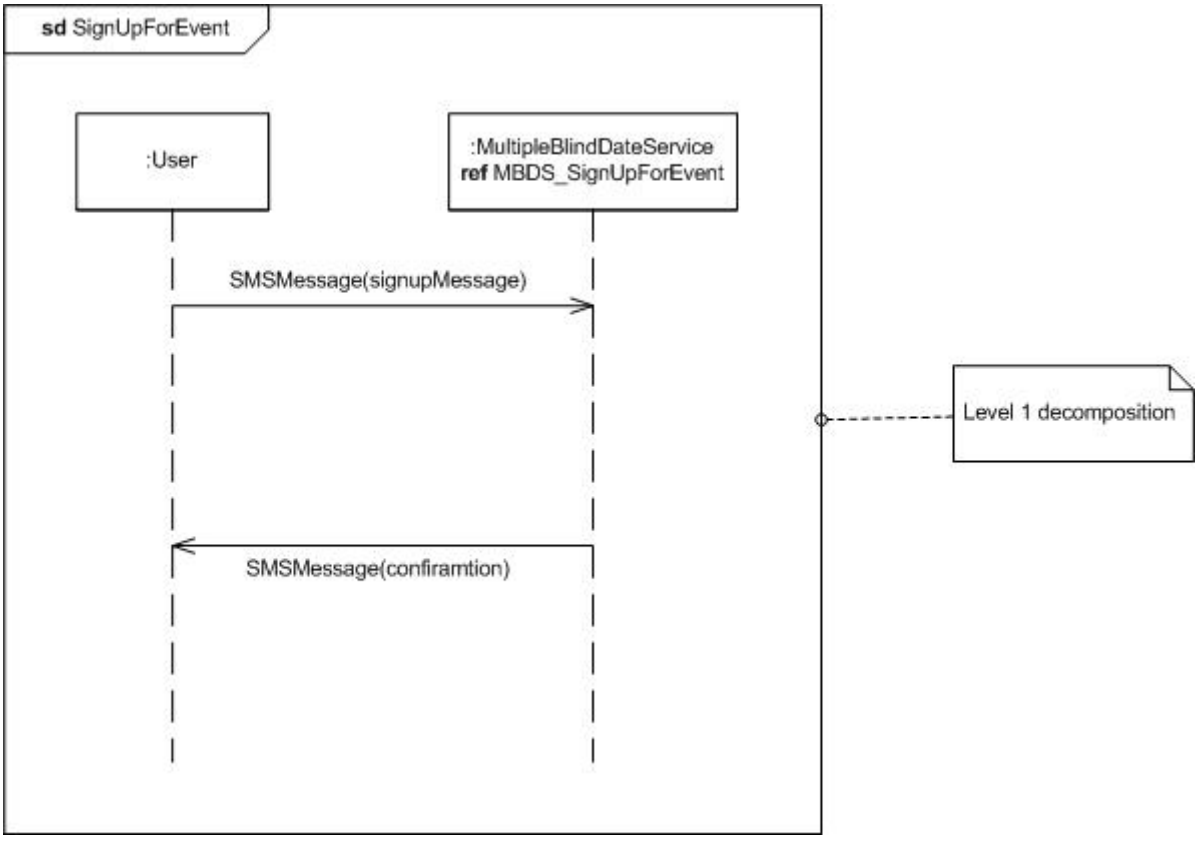


Figure 2

3.1.2 Level 2 decomposition

The level 2 decomposition (Figure 3) shows how the interaction with the MessageService is. The MessageService is an existing service that we use to handle SMS-Messages. From the MessageService we expect to get the phone number from the person who sent the SMS, and of course the message itself.

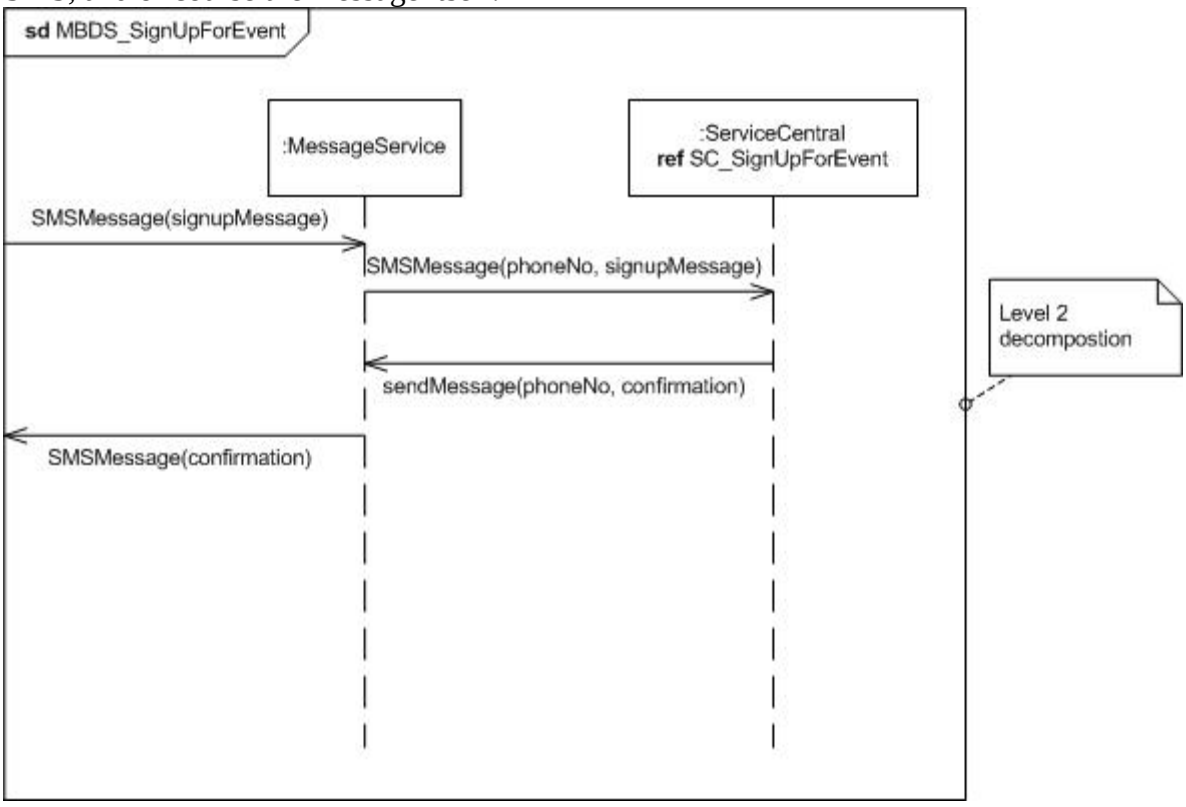


Figure 3

Assumptions for level 2 decomposition

The only thing we assume here is that the MessageService gives us the phone number of the user who sends the SMS, so that we can identify the user in a simple way.

3.1.3 Level 3 decomposition

In level 3 decomposition of SignUpForEvent the actual signup is illustrated (Figure 4). Central is the control object that handles the signupMessage. The signupMessage includes the time of the event and the eventType. Then the central gets the event from the eventRegister, and then signs up the participant for the given event. We have also here only modeled the ”happy day” scenario.

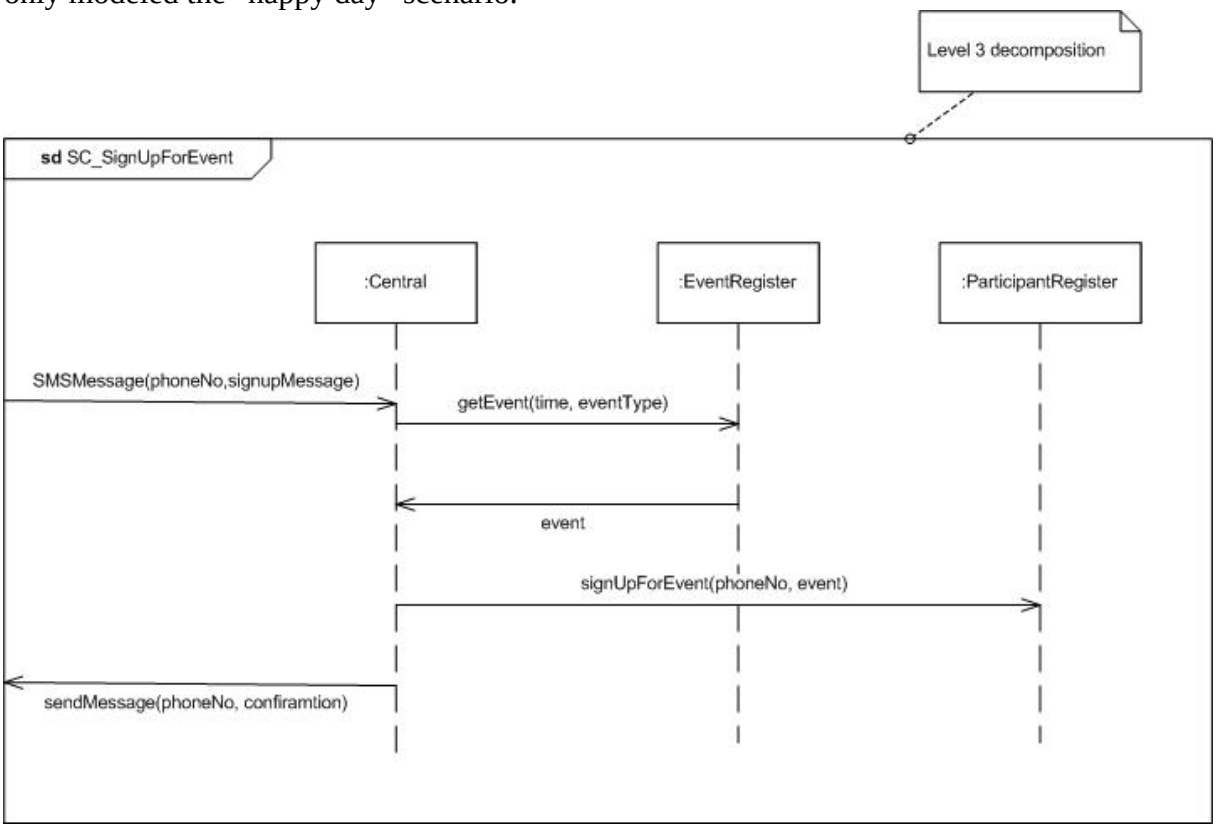


Figure 4

Assumptions for level 3 decomposition

We assume that the signupMessage at this level contains time and eventType, and that the event exists in the eventRegister.

3.2 SendReminder

The SendReminder functionality has been decomposed in three levels.

3.2.1 Level 1 decomposition

At a given time before an event the MultipleBlindDateService asks the TrafikantenService to find the best route between two positions, and then sends a message including this route to the user.

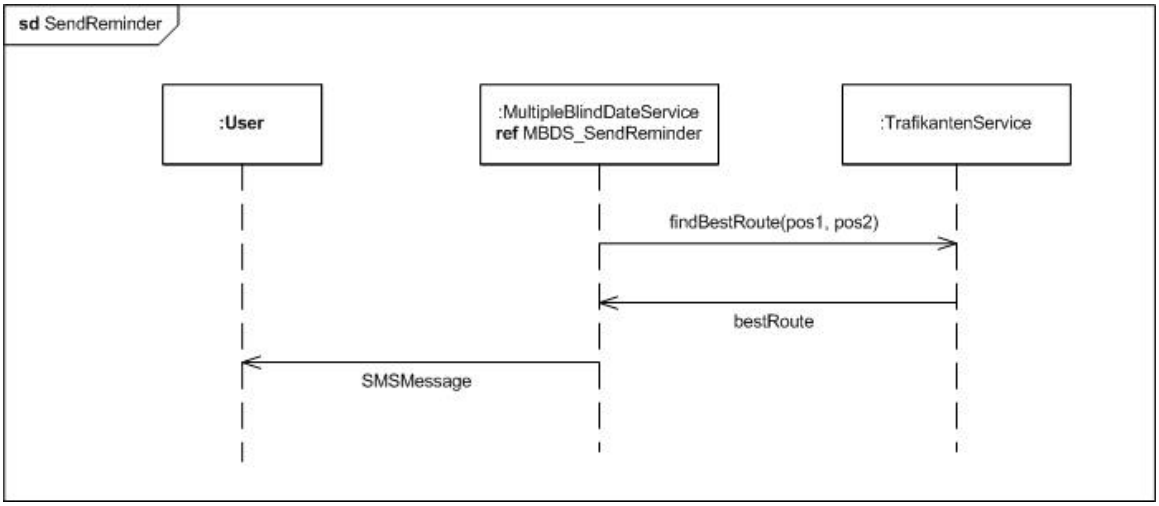


Figure 5

Assumptions for sendReminder level 1

We are assuming that the TrafikantenService has an interface that enables MultipleBlindDateService to get the best route when it provides a position. We have not specified the details of how these positions are represented. Following our happy-day scenario assumption, we assume that there is a route between the two positions.

3.2.2 Level 2 decomposition

At a given time before an event the ServiceCentral asks the PositioningService for the current position of the cell phone with a given phone number. After receiving the best route the ServiceCentral asks the MessageService to send a SMS message containing the best route to a given phone number.

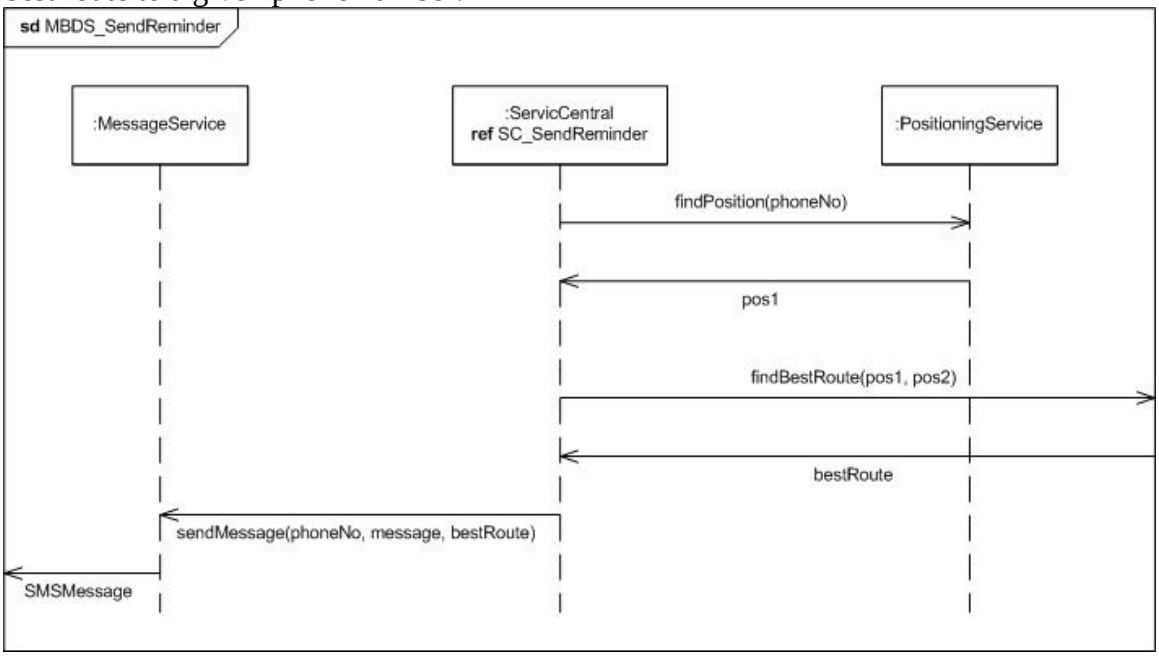


Figure 6

Assumptions for level 2 decomposition

We assume that the cell phone with the given phone number is in a state that makes it possible for the PositioningService to localize it.

3.2.3 Level 3 decomposition

The Scheduler wakes up the Central at a given time (30 minutes before an event occurs) before an event. Then the central retrieves an event from the EventRegister based on the time given by the Scheduler. Central asks ParticipantRegister for a list of participants for the given event. Central gets the place by asking EventRegister. Central now have the event, where the event is located and a list of who wants to take part on the event. The task is then to figure out where the different participants are and provide them with the best route from their current location to the place of the event. This is done within the loop seen in the diagram below, using the concepts described under level 1 and 2.

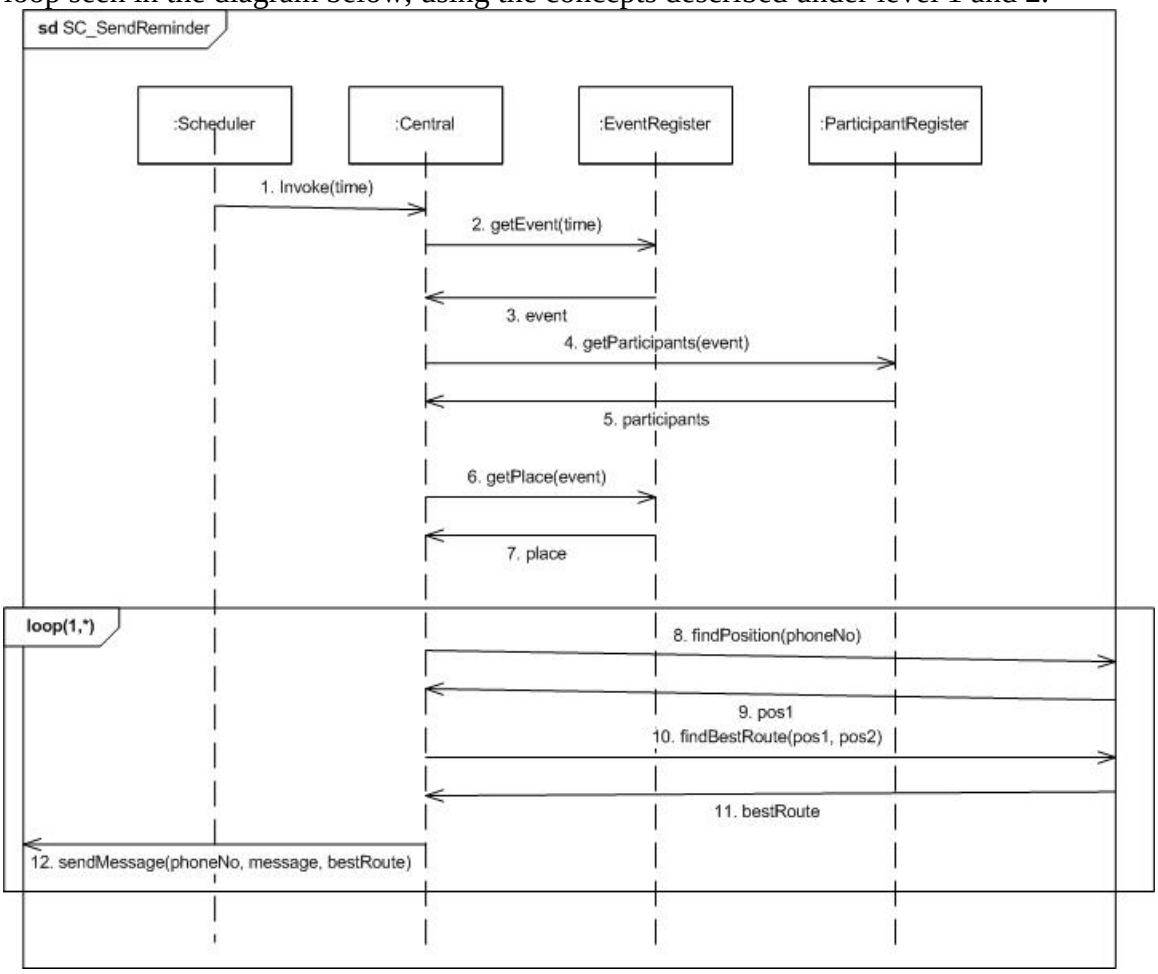


Figure 7

Assumptions for level 3 decomposition

We assume that the EventRegister is capable of returning an event based on time. As mentioned earlier we also assume that there is only one place for an event. Another assumption is that there actually are participants assigned to the given event.

3.3 Composite Structure

In the composite structure diagrams we are reflecting the possible communication between the different parts in the sequence diagrams at the corresponding level.

3.3.1 Level 1

As the sequence diagrams have shown there are two-way communication between User and the MultipleBlindDateService, and between the TrafikantenService and the MultipleBlindDateService. User has no way communicating with the TrafikantenService directly.

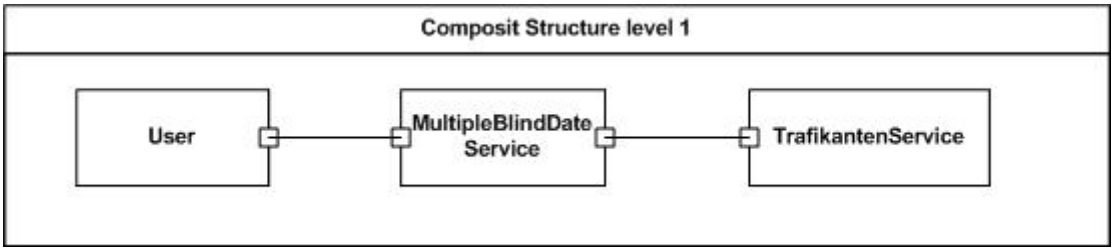


Figure 8

3.3.2 Level 2

This composite structure shows a decomposition of the MultipleBlindDateService part. It consists of a controlling part called ServiceCentral, a MessageService which handles receiving and sending of SMS messages, and a PositioningService. The ServiceCentral has two-way communication with both the other parts, while they have no way of communicating directly with each other.

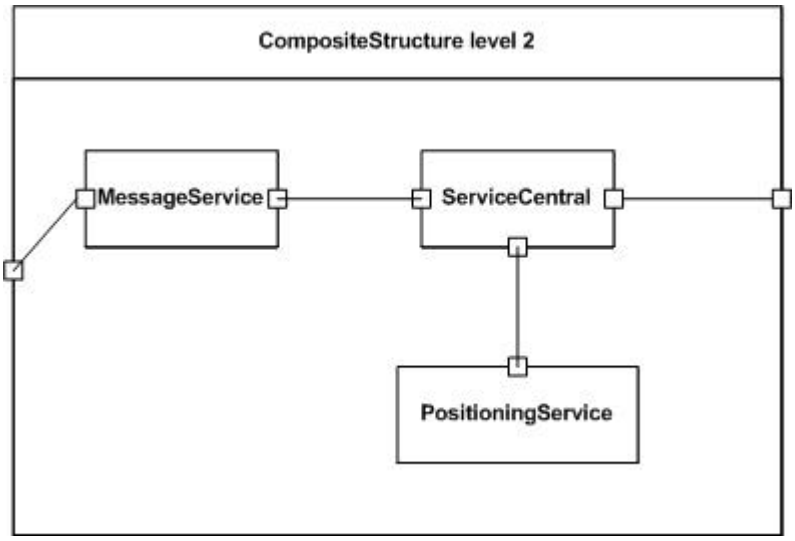


Figure 9

3.3.3 Level 3

This composite structure shows a decomposition of the ServiceCentral part. We have a control part, Central, which has two-way communication with the three other parts Scheduler, EventRegister and ParticipantRegister. These three parts have no way of communicating with each other.

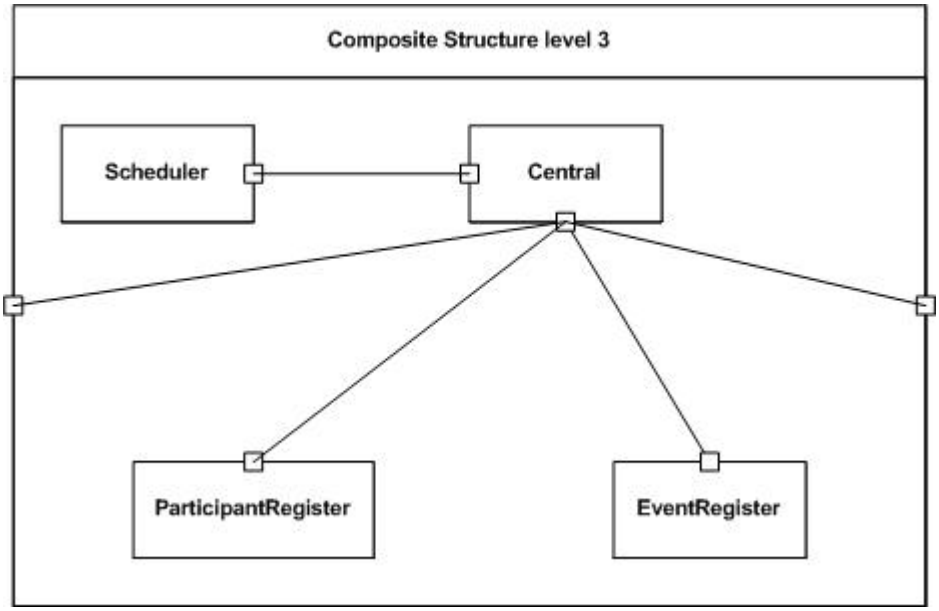
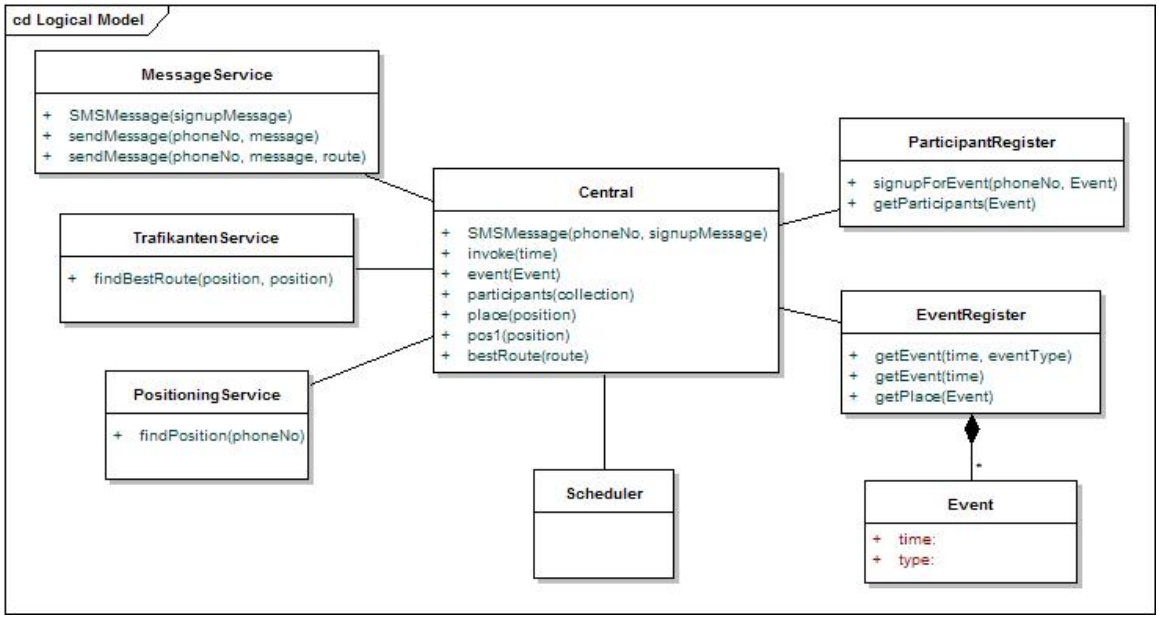


Figure 10

4 Class diagram



Figur 11

5 Refinements

Property refinement can be done mainly in two different ways: narrowing and supplementing. When we do narrowing we move one or more of the positive traces to be negative traces. When we do supplementation we take one or more of the inconclusive traces and make them positive. From our diagrams we have identified the positive traces (shown in Figure). We haven't identified any negative traces. After identifying the existing traces, we found inconclusive traces. Some of them as defined in the mandatory exercise description, and several that we have found on our own.

We have only done supplementing, that is, we have made all of the inconclusive traces positive. All of the refinements are done at level 3 of the decomposition.

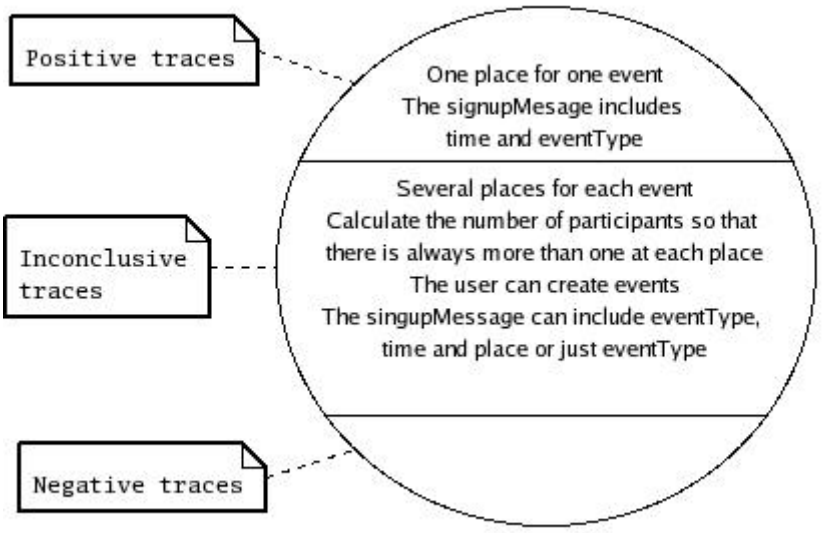


Figure 12

For each of the new diagrams we present a proof that the new diagram is a property refinement of the original diagrams.

5.1 Refinements for SignUpForEvent

The supplementing traces for SignUpForEvent are the traces that have to do with different types of signupMessage and the trace where the user has the opportunity to create new events if they don't exist. The new diagram is shown in figure 13.

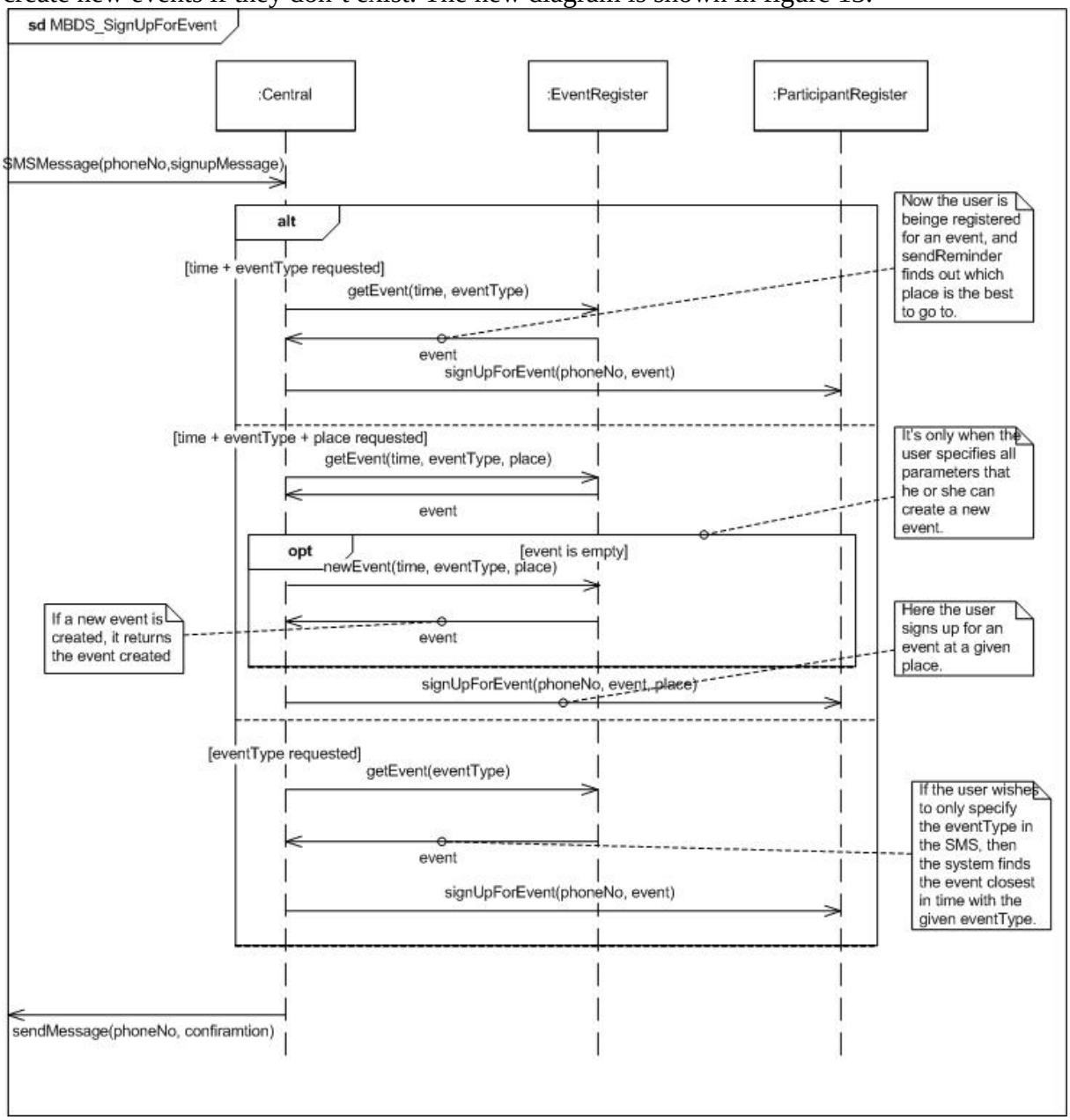


Figure 13

In the sequence diagram above we see that there are introduced three conditional fragments and one optimal fragment. Which alt-box the trace will go into is depended on the signupMessage. If the signupMessage contains time and eventType, than the trace will go through the first alternative. When the signupMessage also includes a place, than the trace will go in the second alternative. In this alternative we have put in an option. If

the event that the user wants to sign up for doesn’t exist, then there will be created a new event with the given time and eventType and the event will be associated with the place that is given from the user. Then the user is signed up for the event, and also connected to his or her given place. If the signupMessage from the user just includes eventType, the trace will go through the third alternative. Here the system finds the event of the given eventType that is closest in time, and then the user is registered for this event. After the user has been signed up for an event, the user gets a confirmation via SMS.

5.1.1 Proof

We now have to prove that the new diagram is a property refinement of the original diagram. We define the diagram in figure 4 be d1 (the original diagram), and the diagram in figure 13 be d_2 (the diagram with refinements). To prove that a diagram is a property refinement of an other diagram we must use the definition of property refinement:

> *An interaction obligation (p_2, n_2) is a property refinement of an interaction obligation (p_1, n_1) if and only if n1 is a subset of or equal to n_2, and p_1 is a subset of or equal to the union of p_2 and n_2*

Let the following represent the messages in sequence diagram d_1 and d_2.
a = SMSMessage(phoneNo, signupMessage)
b1 = getEvent(time, eventType)
b2 = getEvent(time, eventType, place)
b3 = getEvent(eventType)
c = event
d1 = signUpForEvent(phoneNo, event)
d2 = signUpForEvent(phoneNo, event, place)
e = sendMessage(phonNo, confirmation)
f = newEvent(time, eventType, place)

Let p_1 be the set of all positive traces in diagram d_2, and n_2 the set of all negative traces in d_1.
p_1 = { <?a, !b1, ?b1, !c, ?c, !d1, ?d1, !e>,
<?a, !b1, ?b1, !c, ?c, !d1, !e, ?d1> }

n1 = Ø

Let p_2 be the set of all the positive traces in diagram d_2, and n_2 the set of all negative traces in d_2:
p_2 = { <?a> seq (<!b1, ?b1, !c, ?c, !d1, ?d1>
alt <!b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, ?d2>
alt <!b2, ?b2, !c, ?c, !d2, ?d2>
alt <!b3, ?b3, !c, ?c, !d1, ?d1>) seq <!e> }
= { <?a> seq {<!b1, ?b1, !c, ?c, !d1, ?d1>,
<!b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, ?d2>,
<!b2, ?b2, !c, ?c, !d2, ?d2>,
<!b3, ?b3, !c, ?c, !d1, ?d1>} seq <!e> }

= {<?a, !b1, ?b1, !c, ?c, !d1, ?d1, !e> ,
<?a, !b1, ?b1, !c, ?c, !d1, !e, ?d1> ,
<?a, !b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, ?d2, !e>,
<?a, !b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, !e, ?d2>,
<?a, !b2, ?b2, !c, ?c, !d2, ?d2, !e>,
<?a, !b2, ?b2, !c, ?c, !d2, !e, ?d2>,
<?a, !b3, ?b3, !c, ?c, !d1, ?d1, !e>,

<?a, !b3, ?b3, !c, ?c, !d1, !e,?d1> }

= $p_1 \cup$ {

<?a, !b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, ?d2, !e>,

<?a, !b2, ?b2, !c, ?c, !f, ?f, !c, ?c, !d2, !e, ?d2>,

<?a, !b2, ?b2, !c, ?c, !d2, ?d2, !e>,

<?a, !b2, ?b2, !c, ?c, !d2, !e, ?d2>,

<?a, !b3, ?b3, !c, ?c, !d1, ?d1, !e>,

<?a, !b3, ?b3, !c, ?c, !d1, !e, ?d1> }

$n_2 = \emptyset$

Since $p_1 \subseteq p_2 \cup n_2$ and $n_1 = \emptyset \subseteq n_2 = \emptyset$ (neither d_1 nor d_2 has any negative traces), then d_2 is a property refinement of d_1 in accordance to the definition.

5.2 Refinement for SendReminder

We have done the following refinements to the SendReminder functionality.

- There could be more than one place registered for an event.
- The meeting place shall be the most optimal meeting place for the participants.
- Calculate the number of participants so that there is always more than one participant at each place

We understand the most optimal meeting place to be the meeting place closest to the participant's location when the reminder is sent.

In the refinement we do not know if there is more than one place for an event. If there are multiple places for an event we have the possibility to calculate which is the most optimal for a participant. If there is only one place registered for an event there is no need to calculate, as he has to be sent to this location.

If there is more than one place for each event we have to figure out where each participant is located before we can assign him to a place.

The calculation of how many participants there are at each place can not be done before we have done the initial assignment to a place for all participants associated with a given event.

If there are places with only one participant this participant is assigned to a different place.

Whether there were one or multiple places for an event we now have a current position and a destination for each participant, and we can get individual routes for each participant.

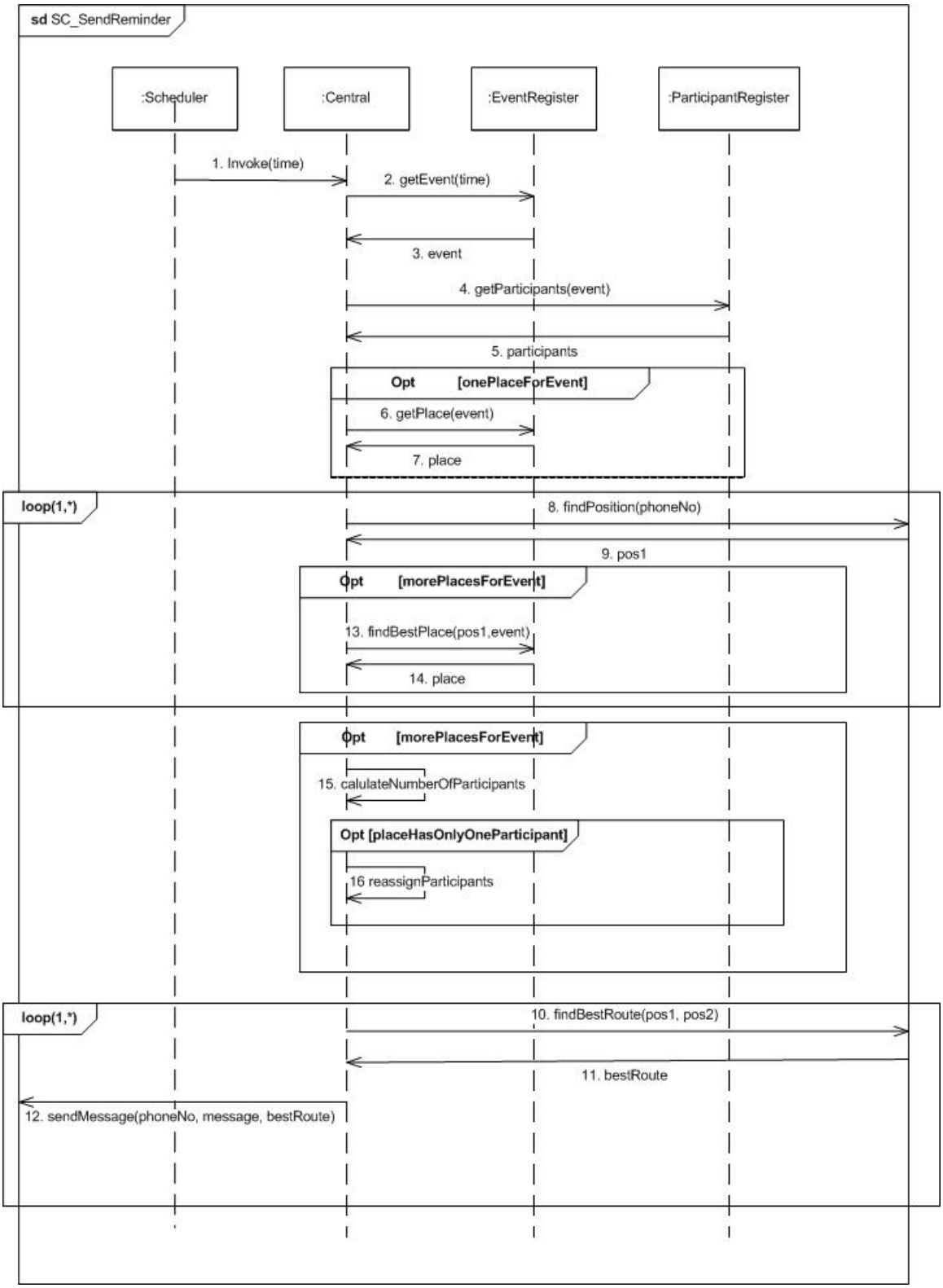

sd SC_SendReminder
:Scheduler
:Central
:EventRegister
:ParticipantRegister
1. Invoke(time)
2. getEvent(time)
3. event
4. getParticipants(event)
5. participants
Opt [onePlaceForEvent]
6. getPlace(event)
7. place
loop(1,*)
8. findPosition(phoneNo)
9. pos1
Opt [morePlacesForEvent]
13. findBestPlace(pos1,event)
14. place
Opt [morePlacesForEvent]
15. calulateNumberOfParticipants
Opt [placeHasOnlyOneParticipant]
16 reassignParticipants
loop(1,*)
10. findBestRoute(pos1, pos2)
11. bestRoute
12. sendMessage(phoneNo, message, bestRoute)

Figure 14

5.2.1 Proof

We now have to prove that the new diagram is a property refinement of the original diagram. We define the diagram in figure 7 be d_1, the original diagram, and the diagram in figure 14 be d_2, the diagram with refinements. Again we have to prove that the two diagrams are related to eachother as per the definition of property refinement on page 18.

Let p_1 be the set of all positive traces in diagram d_1, and n_1 the set of all negative traces in d_1.
p_1 = {(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!6,?6,!7,?7,!8,?8,!9,?9,!10,?10,!11,?11.!12)}
$n_1 = \varnothing$

Let p_2 be the set of all positive traces in diagram d_2, and n_2 the set of all negative traces in d_2.

p_2 = {(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!6,?6,!7,?7,!8,?8,!9,?9,!10,?10,!11,?11.!12),
(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!8,?8,!9,?9,!13,?13,!14,?14,!15,?15,!10,?10,!11,?11.!12),
(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!8,?8,!9,?9,!13,?13,!14,?14,!15,?15,!16,?16,!10,?10,!11,?11.!12}

p_2 = {p_1,
(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!8,?8,!9,?9,!13,?13,!14,?14,!15,?15,!10,?10,!11,?11.!12),
(!1,?1,!2,?2,!3,?3,!4,?4,!5,?5,!8,?8,!9,?9,!13,?13,!14,?14,!15,?15,!16,?16,!10,?10,!11,?11.!12}

$n_2 = \varnothing$

$p_1 \subseteq p_2$ and $n_1 = \varnothing = n_2$

Since $p_1 \subseteq p_2 \cup n_2$ and $n_1 = \emptyset \subseteq n_2 = \emptyset$ (neither d_1 nor d_2 has any negative traces), then d_2 is a property refinement of d_1 in accordance to the definition.