INF 5150 Drop 2.

Sven-Jørgen Karlsen
Gorm Johnsen
Bjørnar Solhaug
Astri Ek Larsen
Ida Margrethe Heyerdahl

November 19, 2005

Contents

1 Introduction

This document specifies an implementation of the blind date system as specified by the sequence diagrams RegisterCustomer, JoinEvent and NotifyCustomers referred to from BlindDate1. The UML design consists of composite structures and state machines.

2 Composite Structures

2.1 Composite Structure of BDSystem

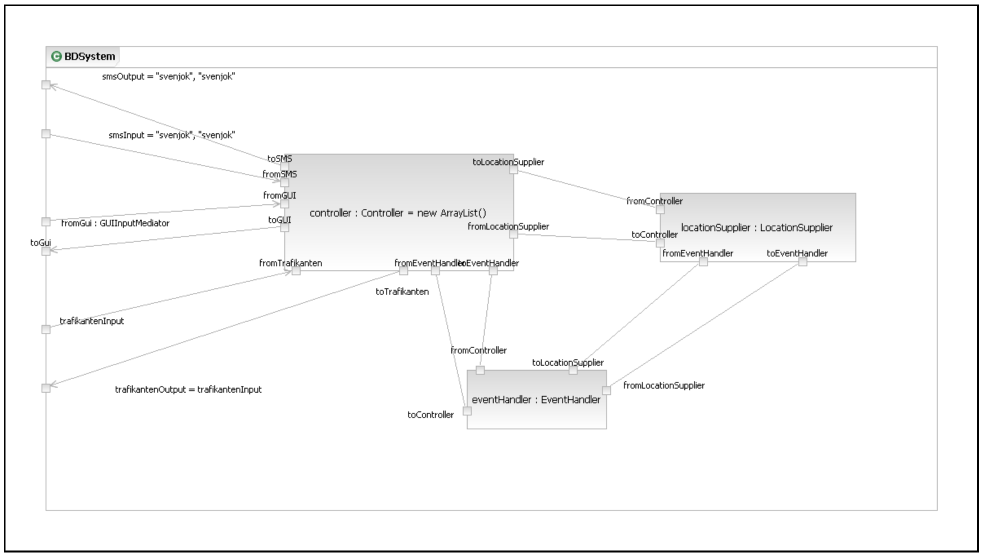


Figure 1: The Composite Structure of the BDSystem class

2.2 Composite Structure of the Controller class

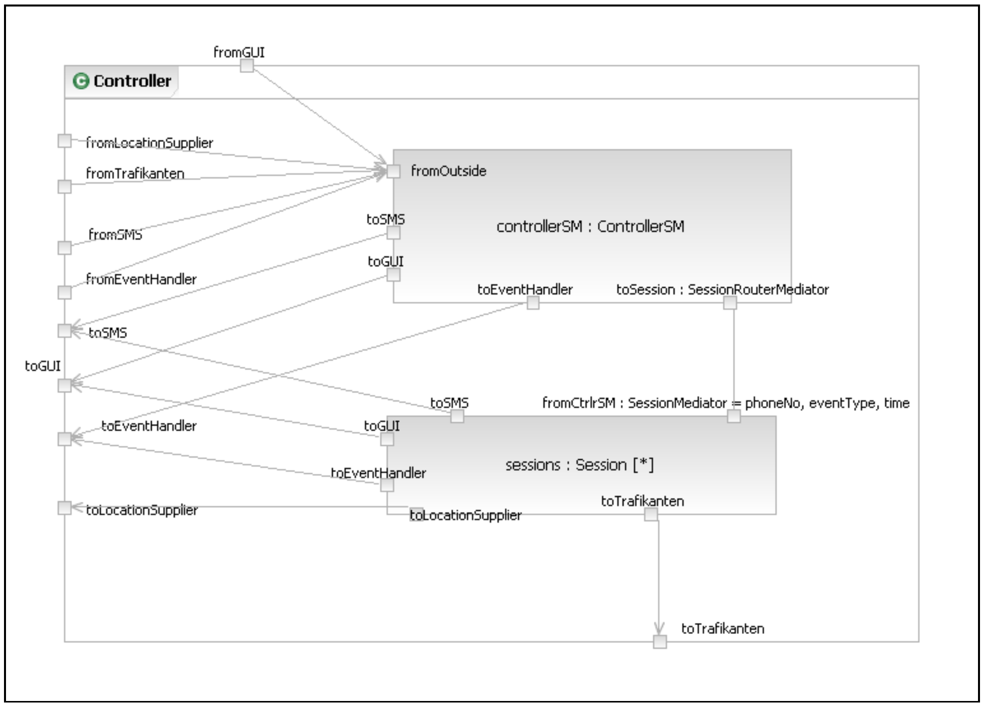


Figure 2: The Composite Structure of the Controller Class

2.3 Composite Structure of EventHandler

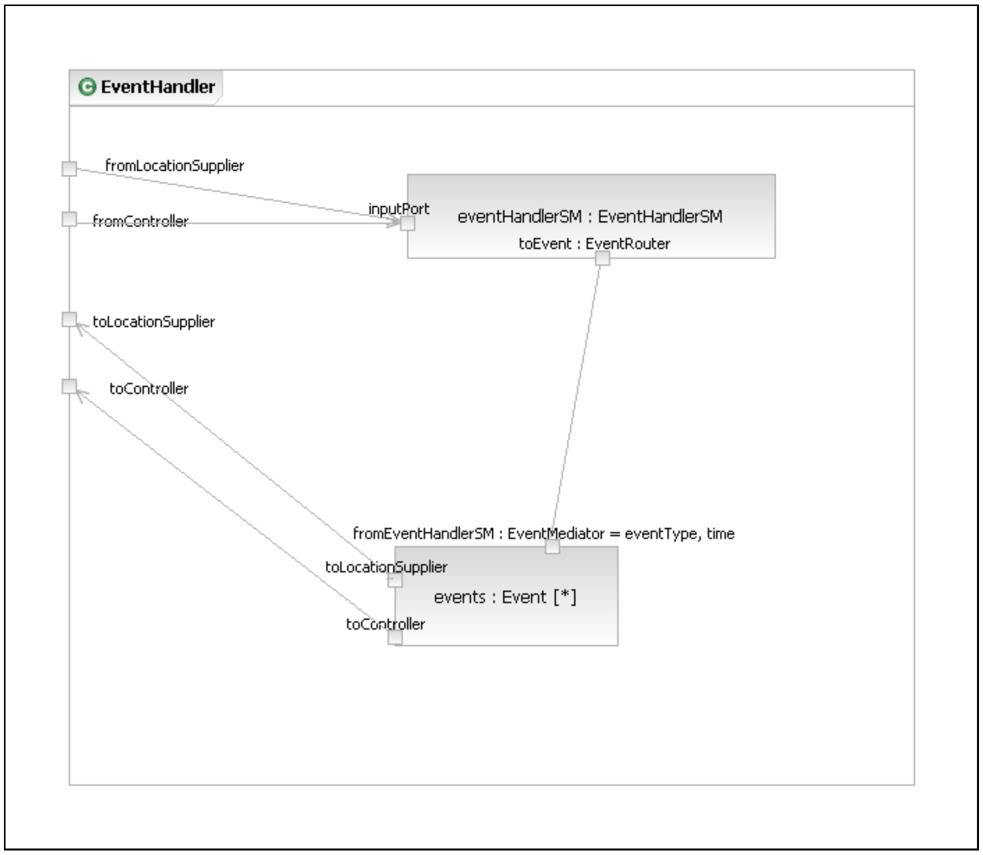


Figure 3: The Composite Structure of the EventHandler class

3 Class diagram of BDSystem

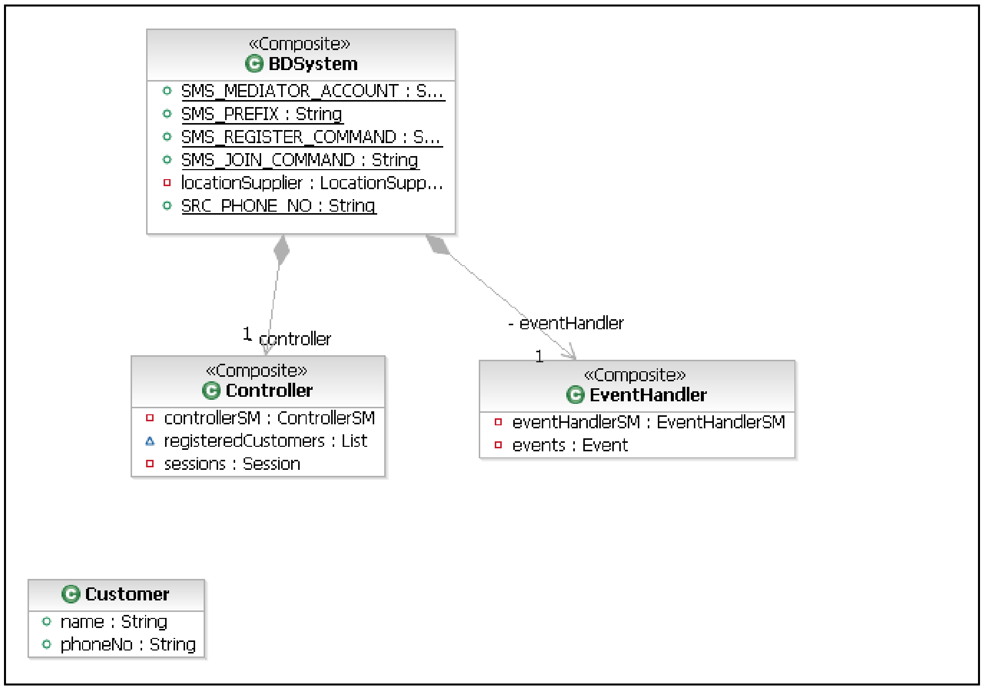


Figure 4: Class diagram of BDSystem

4 State Machines

4.1 Controller

4.1.1 ControllerSM

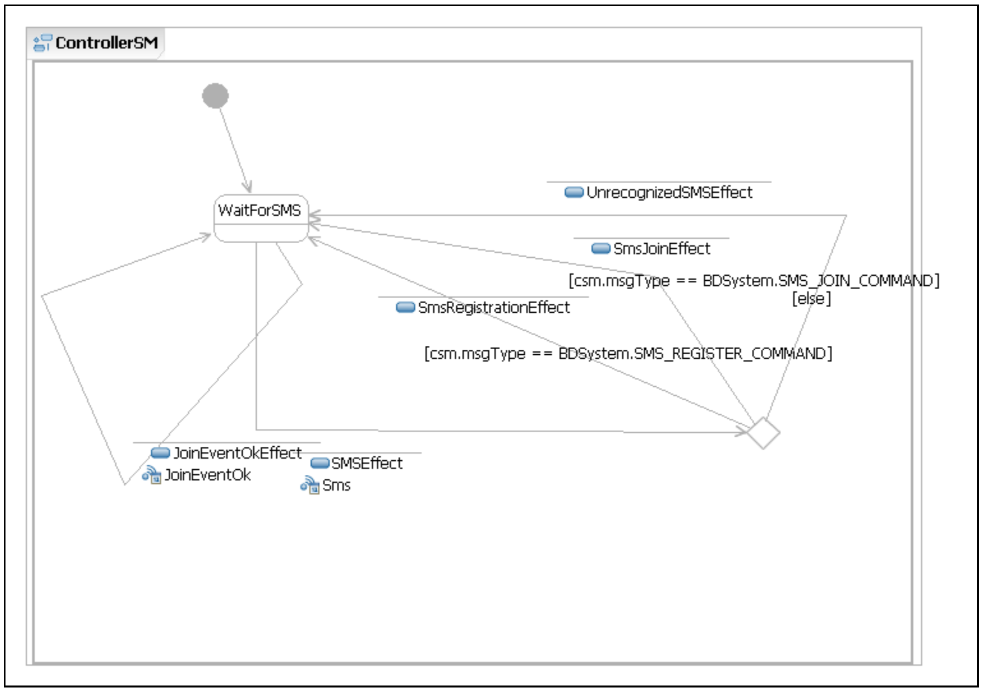


Figure 5: The state machine for the ControllerSM class

4.1.2 Session

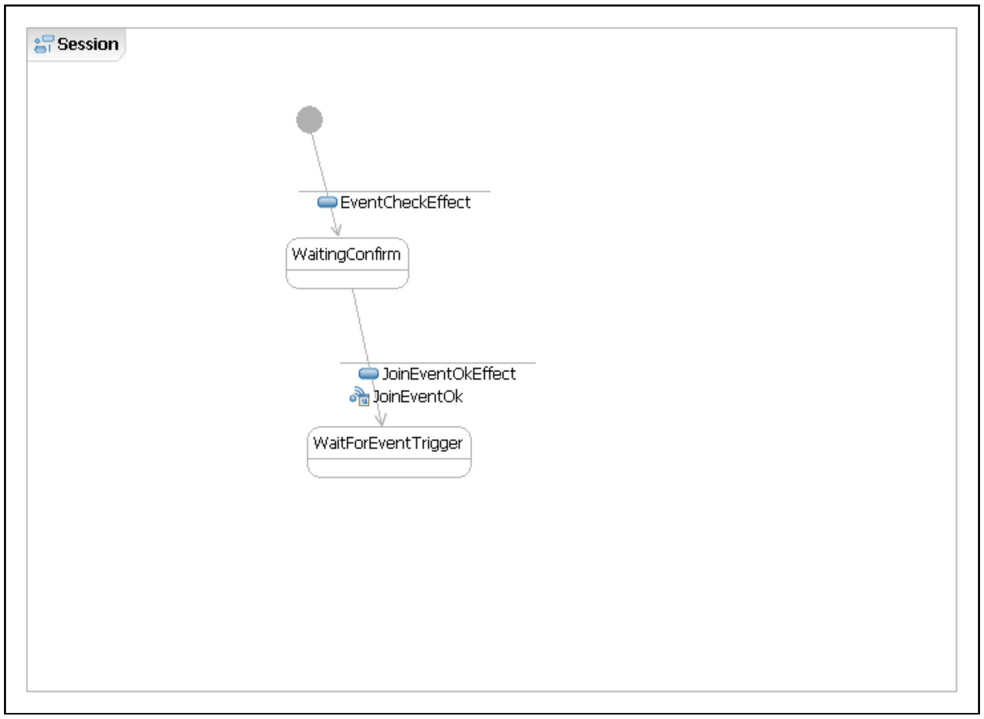
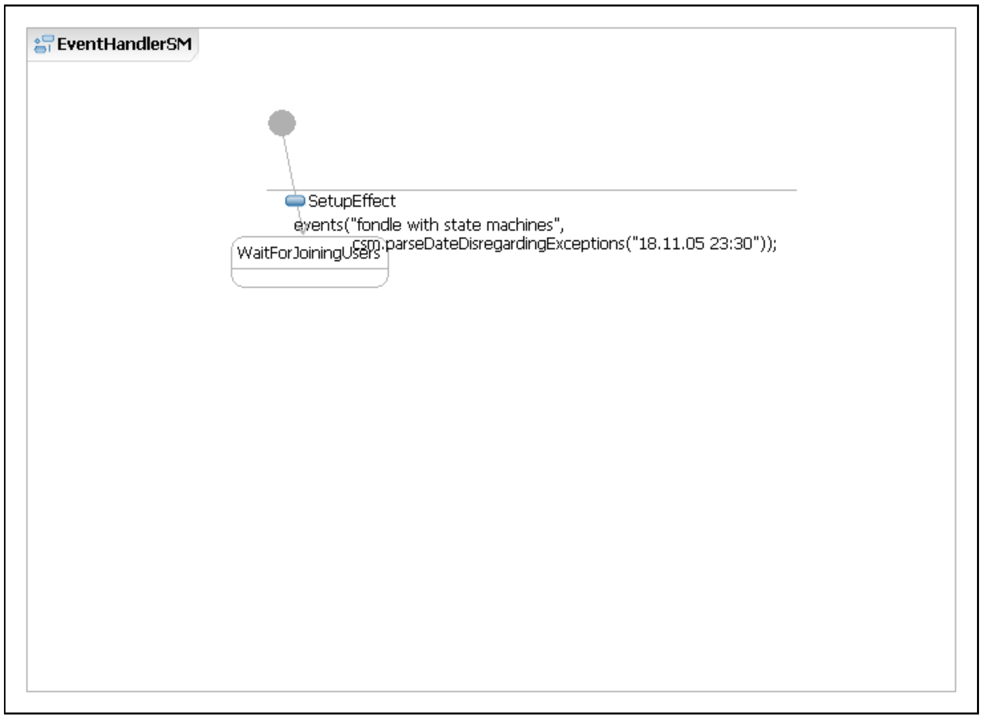


Figure 6: The state machine for the Session class

4.2 EventHandler

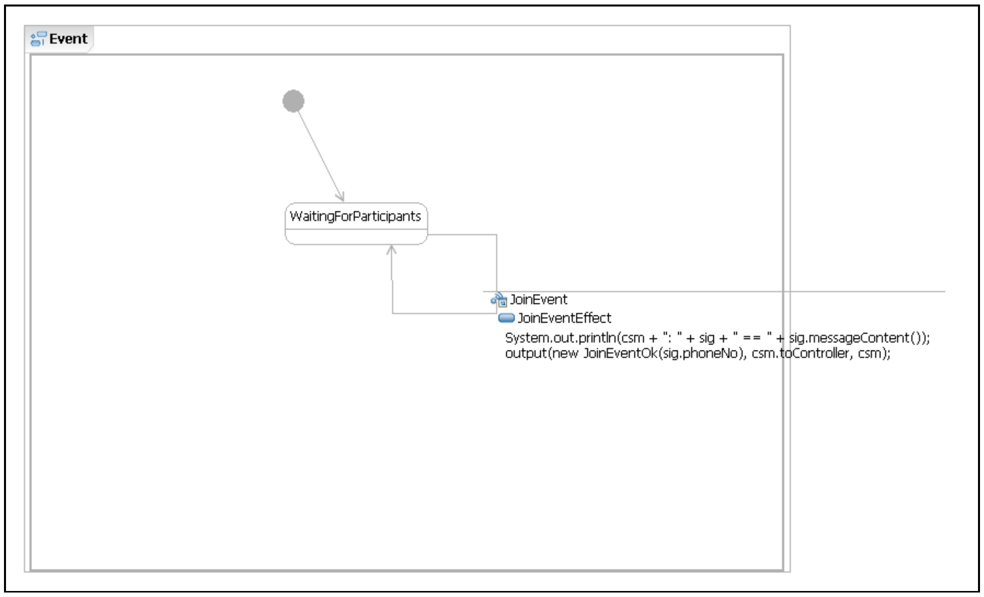
4.2.1 EventHandlerSM



```
events("fondle with state machines",
        csm.parseDateDisregardingExceptions("18.11.05 23:30"));
```

Figure 7: The state machine for the EventHandler class

4.2.2 Event



```
System.out.println(csm + ": " + sig + " == " + sig.messageContent());
output(new JoinEventOk(sig.phoneNo), csm.toController, csm);
```

Figure 8: The state machine for the Event class

5 Running the Implementation

How to test the system: Send SMSs to 2034 and
1. To register a user: "STUD1 konto svenjok register Sven-Jørgen Karlsen"
2. To join an event: "STUD1 konto svenjok join fondle with state machines at 18.11.05 23:30"

6 Refinement argument

Our design is based on the general specification of the Blind Date System, i.e. BlindDate1, as given by the sequence diagram **BlindDate1**. For the purpose of arguing that the design is a refinement of BlindDate1, we first need to identify the set of interaction obligations in the semantics. Since there is no occurrence of the *xalt* operator in the specification, the semantics consists of a singleton set of interaction obligations, $\{(p, n)\}$, where p denotes the positive traces and n the negative ones.

BlindDate1 is specified as the sequential composition of the three sequence diagrams **RegisterCustomer** (**RC**), **JoinEvent** (**JE**) and **NotifyCustomers** (**NC**), where **RC** is optional. Let $\mathsf{pos}(S)$ and $\mathsf{neg}(S)$ denote the set of positive and negative traces, respectively, of any sequence diagram S. The set of positive traces of **BlindDate1** is then given by $\mathsf{pos}(\mathbf{RC}) \succeq \mathsf{pos}(\mathbf{JE}) \succeq \mathsf{pos}(\mathbf{NC}) \cup \mathsf{pos}(\mathbf{JE}) \succeq \mathsf{pos}(\mathbf{NC})$, where $\succeq$ denotes sequential composition of trace sets and interaction obligations. As there are no use of nor *neg* nor *assert* in the specification of **BlindDate1**, $\mathsf{neg}(\mathbf{BlindDate1}) = \emptyset$.

Formally, for the design of the implementation to be a refinement of **BlindDate1**, there must exist an interaction obligation (p', n') in the semantics of the implementation such that $n \subseteq n'$ and $p \subseteq p' \cup n'$. In terms of traces, an implementation contains no inconclusive traces. The positive traces correspond to the possible runs, and the complement to the positive traces forms the set of negative traces. Hence, since $n = \emptyset$, $n \subseteq n'$ is satisfied. Moreover, since $p' \cup n'$ consists of all traces, $p \subseteq p' \cup n'$ is also satisfied. The design of the implementation is thus a refinement of **BlindDate1**.

In the general case, any interaction obligation o is a refinement of an interaction obligation (p, n) in which $n = \emptyset$ since the condition for o refining (p, n) is trivially satisfied.

More generally, since state machines do not classify traces as inconclusive, a system specification given by state machines is a refinement of a specification given by sequence diagrams if the negative traces of the latter remain negative in the former. If the inconclusive ones are not reclassified as positive, they will be negative; in any case, there has been a supplementing. If the positive traces are reclassified as negative, there has been a narrowing. Since positive traces may remain positive and negative remain negative in the refinement, it only remains to verify that there exists no negative trace that has been reclassified as positive.