DROP 2, INF 5150

Group 5,
November 2005

Vikash Katta (vikashk),
Rune Frøysa (runefr),
Kristoffer Stav (krsta),
Ina Flesvik (inahe)

1. Introduction

This document contains a brief description of the implementation of the Multiple Blind Date system (called BDS in this document), which choices we have made, proof of refinements and a risk analysis with Coras of the implemented system. Section bds describes the implementation. Section security describes the risk analysis of this system.

BDS is a service that organizes events for customers who participate. The customers subscribes for event via text messages (SMS). A reminder message is sent back to the customers prior to the event with a suggested public transport schedule. PATS, a service from Telenor, is used as sms mediator and to find client location, and an online service from Trafikanten is used to find a transport schedule.

2. The BSD implementation

2.1. Introduction

This implementation is based on the drop 1 solution provided by the teachers. We have implemented BlindDate1, which consist of the functionality described in the 'Drop 1 Proposed solution' document, except the RegistrationCustomer feature. We do not attempt to trap user-errors such as atempting to subscribe to a none-existing event.

In the fat-jar file, we have provided an OfflineSMSMediator, which is a GUI application that can act as a replacement for sending and receiving SMSes to PATS. The OfflineSMSMediator is provided in a .jar file, and is enabled simply by replacing no.uio.ifi.pats.client.jar with offlinesmsmediator.jar in your RSM Java-project.

The executable BDS is transformed from UML diagrams, including composite structures and state machines, to plain Java by a JavaFrame RSM-plugin. The composite structure diagrams are not included in this document, since they are already provided in the 'Drop1 proposed solution'. This section includes the state machine diagrams used to build BDS.

2.2. The system

The customer (client) is identified by his / her cellphone number. An event is identified by the concatenation of eventType and eventTime.

A couple of events are provided. These events are built automatically in the creation of the EventHandler with a fixed location; i.a. in the first transition in the EventHandlerSM state machine. This is done in an activity called 'makeEvents'. This system does not provide the functionality of registering new events.

The customers does not have to be registered to join an event. He/she sends an sms to BDS with the text 'STUD1 konto <username> join <event> <time>', where <event> is the name of the event. The first three arguments are ignored by the offline sms mediator. Valid commands with event-id are 'join eventType 60', 'join eventType 90' or 'join eventType 120'. The <time> is the number of seconds until the event is commencing. A join-message could typically look lie this: 'join PlayBingo 60'. When sending this join-message to BDS, the client receives a confirmation message: 'Thank you for subscribing to PlayBingo@60'.

The number of seconds provided in the join-message referes to a TimeTrigger started when the Event was created, which invokes the distribution of reminder-messages to the participants. The reminder message could typically look like this: 'Hi there! You are welcome to participate at event : PlayBingo, at time: 60. How to get there: use line 37 from Sagene kirke (i Arensdalsgt) to Nydalen T at 2005-11-16T17:35:05.000+01:00'

The time provided in this reminder message is the actual departure time from the bus stop, in this case Sagene kirke. We use the DynRoute.getExpectedDepatureTime()-method.

2.3. Limitations

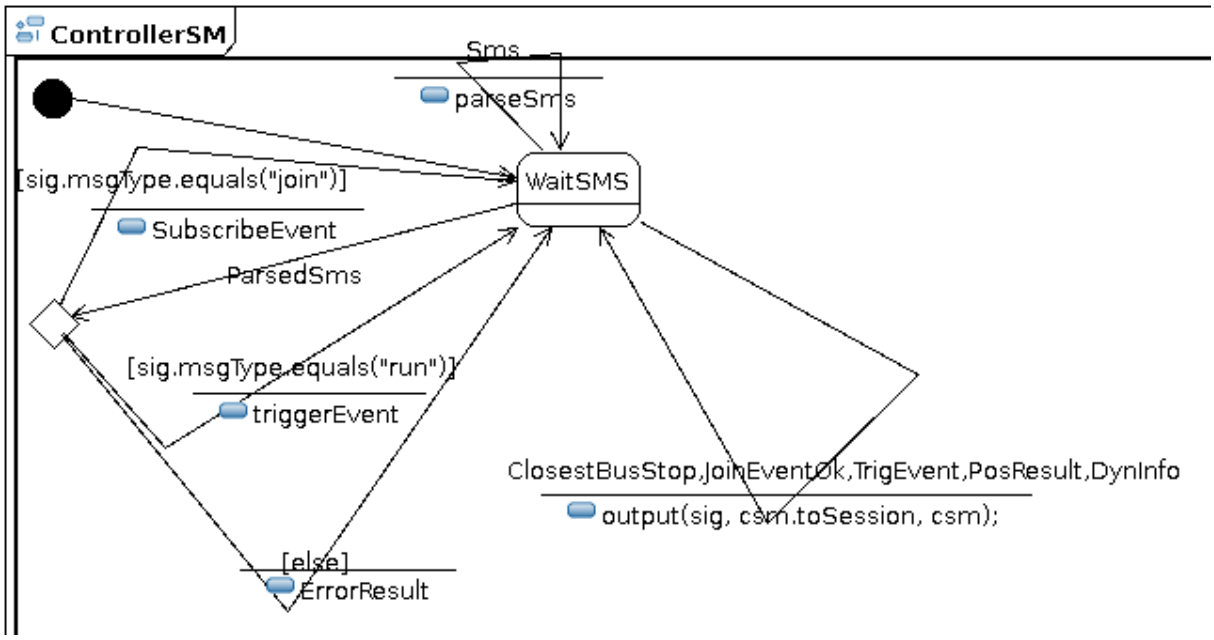
There are some limitations in the implementation both due to time-constraints, but also due to what we feel is relevant in an INF5150 context.

- we have not put much effort into providing helpful messages to the user when receiving bogous input. While the user may receive a syntax error when attempting to join, the message is simply ignored if the event does not exist.
- parsing of join messages is very primitive. The event name cannot contain spaces, and the time is given as the number of seconds from initial event-creation to its happening.
- In the Session state-machine we currently assume that all events occour at the same location. To avoid this, Session would need a way to find the locName for the current event. We could have done this with yet another message. It was skipped due to time constraints.

2.4. Description of implementation

This section contains a rough overview of what happens in the implementation. We only show the state-machines, but not the contents of the various effects. For the full details, please refer to the provided emx file.

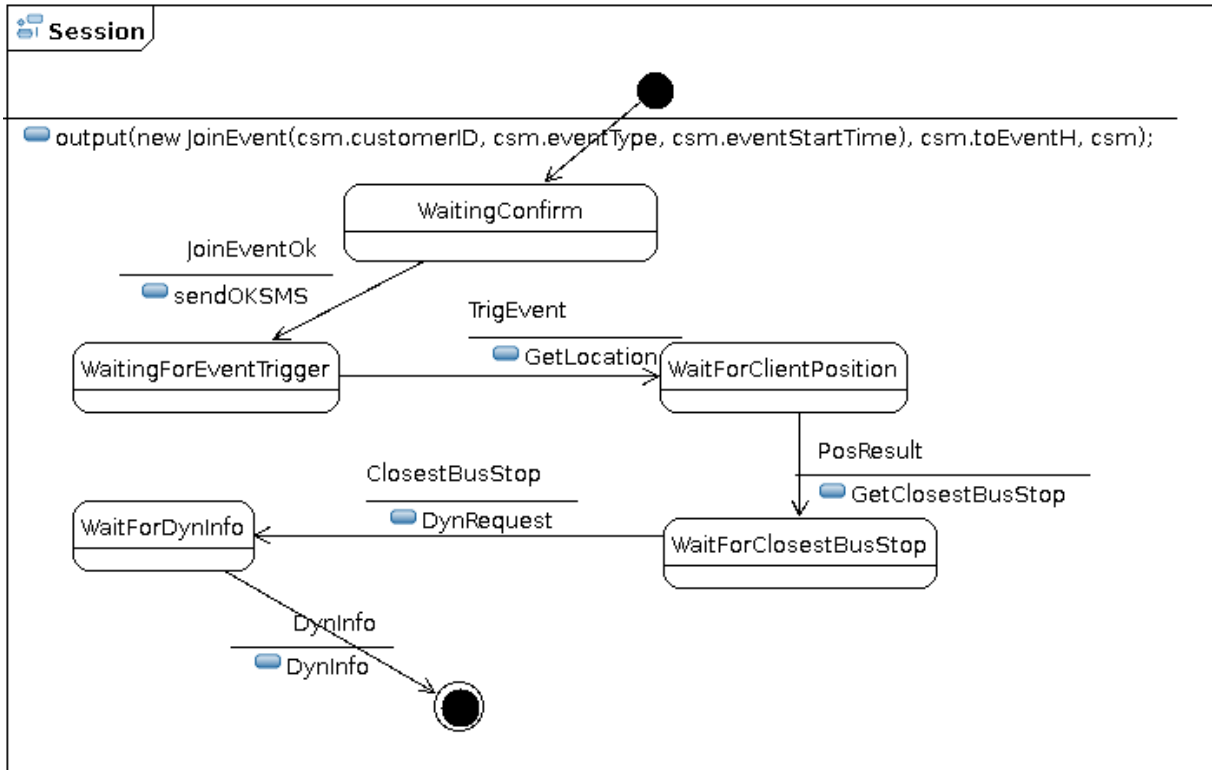
The main state-machine is ControllerSM. Uppon reception of an Sms, it will parse it, and generate a new ParsedSmsSignal. If it is a join message it will create a new Session state-machine. We have also provided support for a special message "run", which is used for debuging to avoid having to wait for the actual timer to trigger.



The ControllerSM

(The black-frame is the result of a RSM bug. If clicking inside the Region it marks it as selected and keeps this marking in the generated gif. If one clicks higher this can be avoided, but then RSM decides to only include parts of the state-machine)

Due to the design of the composites in the drop1 solution, much internal communication can only be done through ControllerSM. Thus several different signals are simply forwarded to the apropriate Session state-machine.

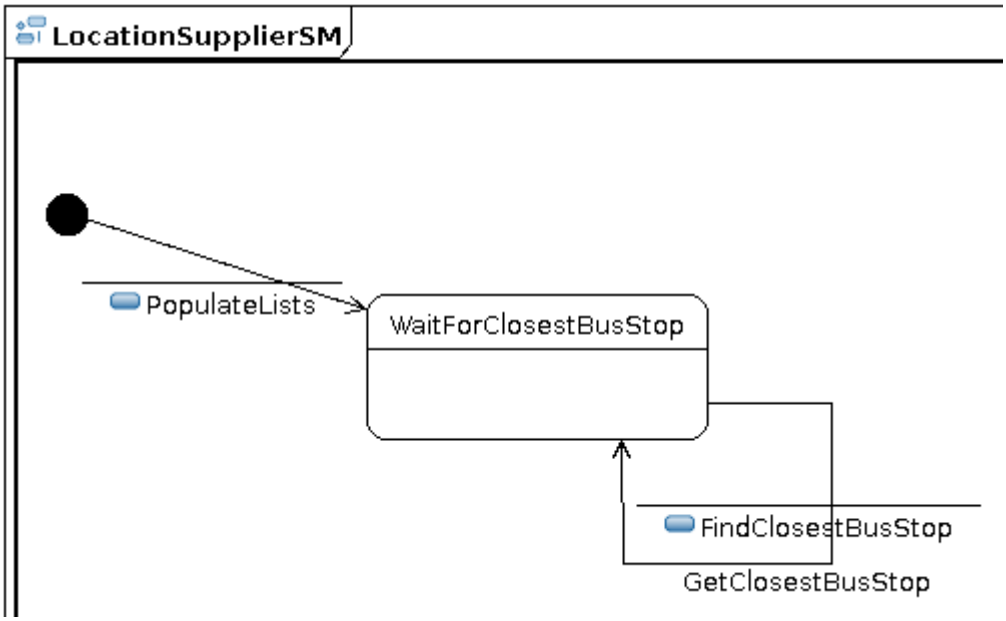


Session

The Session state-machine handles the subscription for one customer for a spesific event. Uppon creation, it sends a JoinEvent to EventHandlerSM (which routes it to Event). After an OK, it will stay in WaitingForEventTrigger until the event happens.

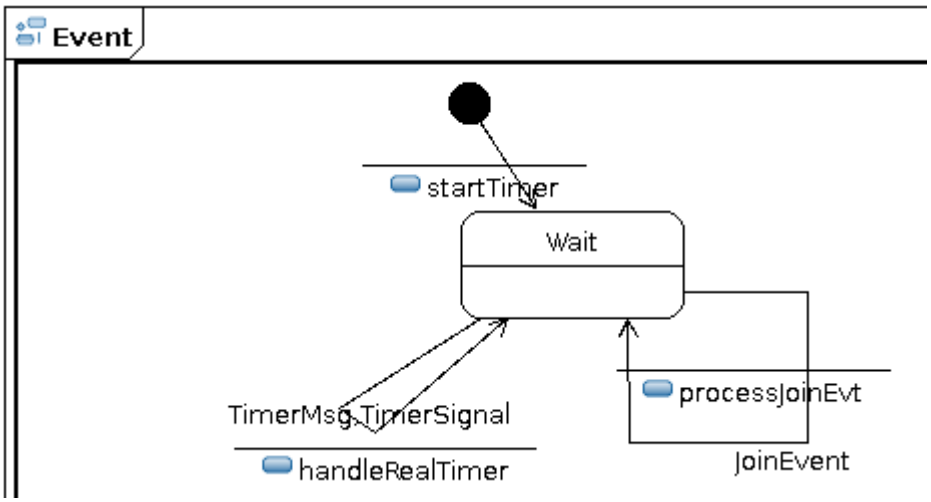
When the trigger happens, we find the clients position, check traveling info with Traffikanten and sends information to the subscriber.

The LocationSupplierSM returns information about what BusStop is closest to a spesific location. "closest" is determined by a fairly primitive algorithm.



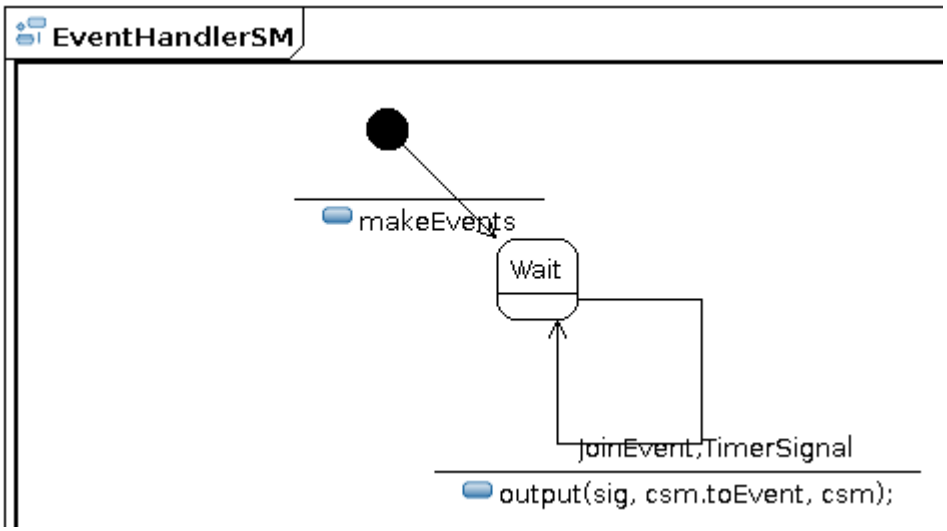
LocationSupplierSM

Uppon creation, the Event state-machine will create a TimerMsg that will trigger when the event should happen. It then waits for a JoinEvent which simply sends out a JoinEventOk (we don't use the participants as we already have one in DynSessionRM). Uppon reception of TimerMsg or TimerSignal (the latter is our debug signal) it will trigger the event.



Event

Uppon creation, the EventHandlerSM will create all the known events. After this its only purpose is to forward messages to Event.



EventHandlerSM

2.4.1. Router mediators

RouterMediators are central to the BDS as we have a number of places where a spesific signal has to reach one or more state-machines depending on information in the signal.

For a signal to get from ControllerSM to the correct Session, we have implemented a SimpleRouterMediator named DynSessionRM. It looks at information in the incoming signal, iterates over its mediatorList and forwards the signal to the apropriate state-machine. Here is an excerpt of its code:

```
public void forward(Message sig) {
   if(sig instanceof JoinEventOk) {
       JoinEventOk evt = (JoinEventOk)sig;
       for (int i=0; i<mediatorList.size(); i++) {
```

```
            fromCS_SM_M mediator = (fromCS_SM_M) mediatorList.get(i);
            if(evt.eventType.equals(mediator.eventType) &&
evt.phoneNo.equals(mediator.customerID) &&
               evt.time == mediator.startTime) {
                 mediator.forward(sig);
            }
       }
```

Currently JavaFrame will throw a NullPointerException if one tries to forward a message to a state-machine that has reached its FinalState. This can happen if a user tries to subscribe to an event that no longer exists. We have decided not to address this issue due to time-constraints.

3. Instructions on usage

Note: our fat-jar is compiled with an OfflineSmsMediator that allows sending sms'es from a simple GUI. A version with the standard mediator can be downloaded from http://folk.uio.no/runefro/tmp/inf5150/drop2/drop2OnlineFat.jar

The BSD is provided as a fat jar. Run it like you run any jar on your operating-system. After starting it, send some messages to it by giving it some commands, for example:

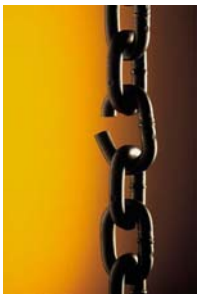
```
STUD1 konto <username> join eventType 60
STUD1 konto <username> join eventType 120
```

You may specify the phone-number for each message that you send. The location used when subscribing will be used when "positioning you" later. N seconds (60 in the first example) after the aplication was started, the event will be triggered. You can also trigger this manually by sending the message `run eventType 60`.

4. Refinement

A state-machine does not have any inconclusive traces. Thus any inconclusive traces in the Drop2 spesifications are now either positive or negative. The state-machines are with some exceptions direct translations of the sequence-diagrams, and the sequence-diagrams for Drop1 does not contains negative traces. Thus we have preserved the positive traces. The exception is for the signals in page 14&15 of the drop1 solutions where signals goes to lifelines that they according to the composite structures cannot reach. In our BDS these traces are now negative, while we have created new positive traces for the same features.

5. Risk analysis



5.1. Context identification

The target that was analyzed is the Blind Date System (BDS) application. We did not consider the trafikanten and customer mobile as they are external to the system (environment).

Following are the assumptions made during the security risk analysis:

1. The security of the BDS is unknown as the application was created using the Java frame. Since we did not have control on the applications internal structure and communication, we assume that the security aspects of the application to be weak.
2. The values for the frequency and consequence in the value definition table are assumed.
3. The main security aspects considered during SRA are confidentiality and availability.

5.1.1. Target of evaluation table

Type: Table
Name: Target of evaluation table
Short description: Defines the area of concern
Concern: Target of evaluation

Table 1: Target of Evaluation Table

Category	Value
Target	The BDS
Client	System owner
Service/Function	(RegisterCustomer), JoinEvent and NotifyCustomers
Quality aspects	Confidentiality of the information used by the BDS and availability of the BDS

5.1.2. Value definition table

Type: Table
Name: Value definition table
Short description: Show the value of the different values
Concern: Target of evaluation

Table 2: Value Definition Table

Type	Allowed values	Description
Asset	Very low, Low, Medium, High, Very high	Difficult to put a value on the different assets, because they are more worth for the business than the actual cost of it
Frequency	Rare, Unlikely, Possible, Likely, Certain	Rare: less than once per ten years Unlikely: less than once a year Possible: about once a year Likely: 2-5 times a year Certain: more than 5 times a year
Consequence	Insignificant, Minor, Moderate, Major, Catastrophic	Insignificant: no impact on system Minor: minor delays Moderate: loss of some profit Major: loss of customer -> loss of profit Catastrophic: out of business
Risk value	Low, Moderate, Major, Extreme	Low: accept the risk Moderate: for some assets - monitor the risk, for other - treat the risk Major: treat the risk Extreme: treat the risk

5.1.3. Risk matrix

Type: Table
Name: Risk matrix
Short description: Shows the connection between consequence and frequency
Concern: Target of evaluation

Table 3: Risk Matrix

Frequency	Insignificant	Minor	Moderate	Major	Catastrophic
Certain	Moderate	Major	Extreme	Extreme	Extreme
Likely	Moderate	Major	Major	Extreme	Extreme
Possible	Low	Moderate	Major	Major	Extreme
Unlikely	Low	Low	Moderate	Major	Major

Frequency	Insignificant	Minor	Moderate	Major	Catastrophic
Rare	Low	Low	Low	Moderate	Major

5.1.4. Asset table

Type: Table
Name: Asset table
Short description: Shows the assets at stake
Concern: Assets

Table 4: Asset Table

Asset ID	Description	Category	Value
BDS	Source code of the system	Software	High
Clientstore	Customers phone numbers and total amount of customer	Information	Medium
Customers satisfaction	How pleased the customer is with the BDS	Human	Very high
Customers trust	How well the customer trust the system will give them right info and don't overcharge them	Human	Very high
Brand value	The company's reputation	Information	High
Hardware	The machine the BDS is installed on	Physical	Low

5.1.5. Risk evaluation criteria table

Type: Table
Name: Risk evaluation criteria table
Short description: Tells which risk that are acceptable for the different assets
Concern: Risk evaluation criteria

Table 5: Risk Evaluation Criteria Table

Criteria ID	Description	Applied for assets
C1	If "Risk level" is equal to "Low" then "Accept the risk"	All
C2	If "Risk level" is equal to "Moderate" then "Monitor the risk"	Hardware, Clientstore, BDS
C3	If "Risk level" is greater og equal to "Moderate" then "Treat the risk"	Customer satisfaction, Customers trust, Brand value
C4	If "Risk level" is "Major" og "Extreme" then "Treat the risk"	BDS, Clientstore, Hardware

5.2. Risk identification

5.2.1. HazOp for BDS

Type: Table
Name: HazOp for BDS
Short description: Shows the threats for the BDS
Concern: Threats

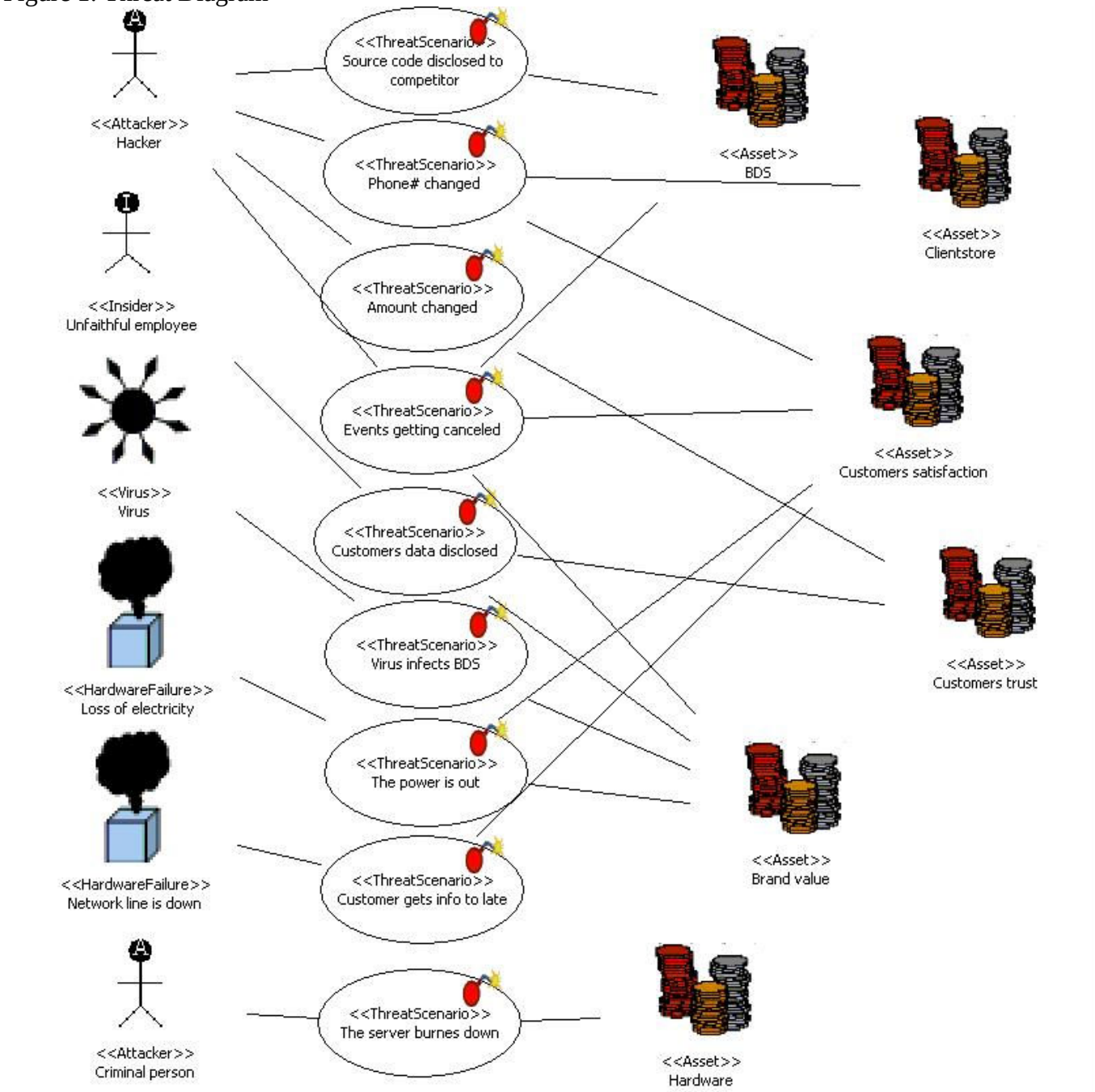
Table 6: HazOp Table

Risk ID	Asset ID	Guideword	Threat	Incident	Scenario
R1	BDS	Disclosure	Hacker	Loss of revenue	Sorce code disclosed to competitor
R2	Customers satisfaction	Disruption	Loss of electricity	Loss of customers	The power is out
R3	Brand value	Disruption	Loss of electricity	Loss of customers	The power is out
R4	Clientstore	Manipulation	Hacker	Loss of customers	Phone# changed
R5	Brand value	Disclosure	Unfaithful employee	Misuse of info	Customers data disclosed to e.g. porn industry
R6	Customers trust	Disclosure	Unfaithful employee	Misuse of info	Customers data disclosed to e.g. porn industry
R7	Customers satisfaction	Disruption	Network line down	Damage BDS reputation	Customer gets info to late
R8	Customers satisfaction	Manipulation	Hacker	Wrong info to user	Phone# changed
R9	Customers trust	Manipulation	Hacker	Customer paid to much or to little	Amount changed
R10	BDS	Manipulation	Hacker	Loss of customers	Events getting cancelled
R11	Customers satisfaction	Manipulation	Hacker	Loss of customers	Events getting cancelled
R12	Brand value	Manipulation	Hacker	Damage BDS reputation	Events getting cancelled
R13	Hardware	Destruction	Criminal person	Loss of customers	The server burnes down
R14	Brand value	Destruction	Virus	Loss of customers	Virus infects system

5.2.2. Threat Diagram

Type: UML Model
Name: Threat Diagram
Short description: Shows the connection between attacker, threat scenario and assets effected
Concern: Threats

Figure 1: Threat Diagram

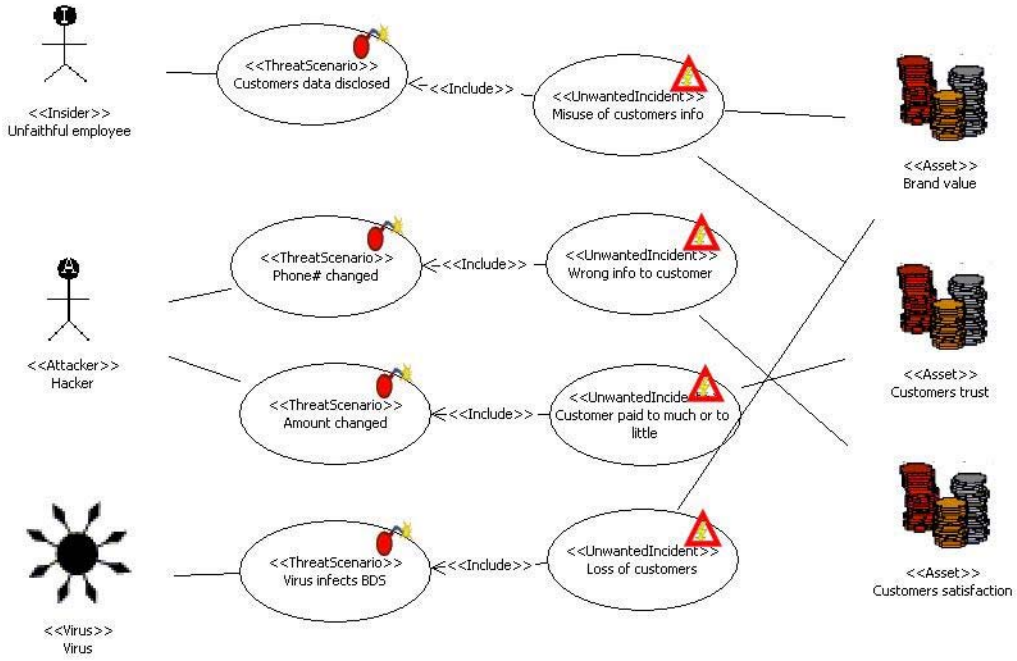


5.2.3. Unwanted incident diagram

Only the diagrams for the unwanted incidents whose risk level is major or extreme (not acceptable) are shown in figure 2.

Type:	UML Model
Name:	Unwanted incident diagram
Short description:	Shows the threats that doesn't satisfy the risk evaluation criteria
Concern:	Threats

Figure 2: Unwanted incident diagram



5.3. Risk analysis

5.3.1. Consequence and frequency table BDS

Type: Table
Name: Consequence and frequency table BDS
Short description: Shows the values of cons and freq of the identified risks
Concern: Consequence

Table 7: Consequence and Frequency Table

Risk ID	Asset ID	Threat	Incident	Scenario	Consequence Value	Frequency Value	Risk Value	Risk Priority
R1	BDS	Hacker	Loss of revenue	Source code disclosed to competitor	Moderate	Unlikely	Low	Accept risk
R2	Customers satisfaction	Loss of electricity	Loss of customers	The power is out	Moderate	Rare	Low	Accept risk
R3	Brand value	Loss of electricity	Loss of customers	The power is out	Moderate	Rare	Low	Accept risk
R4	Clientstore	Hacker	Loss of customers	Phone# changed	Minor	Rare	Low	Accept risk
R5	Brand value	Unfaithful employee	Misuse of info	Customers data disclosed to e.g. porn industry	Major	Possible	Major	Treat risk
R6	Customers trust	Unfaithful employee	Misuse of info	Customers data disclosed to e.g. porn industry	Catastrophic	Possible	Extreme	Treat risk
R7	Customers satisfaction	Network line down	Loss of customers	Customer gets info to late	Moderate	Rare	Low	Accept risk
R8	Customers satisfaction	Hacker	Wrong info to user	Phone# changed	Major	Rare	Moderate	Treat risk
R9	Customers trust	Hacker	Customer paid to much or to little	Amount changed	Major	Rare	Moderate	Treat risk
R10	BDS	Hacker	Loss of customers	Events getting cancelled	Major	Rare	Moderate	Monitor risk
R11	Customers satisfaction	Hacker	Loss of customers	Events getting cancelled	Moderate	Rare	Low	Accept risk
R12	Brand value	Hacker	Loss of customers	Events getting cancelled	Moderate	Rare	Low	Accept risk
R13	Hardware	Criminal person	Loss of customers	The server burnes down	Minor	Rare	Low	Accept risk
R14	Brand value	Virus	Loss of customers	Virus infects system	Major	Likely	Extreme	Treat risk

5.4. Risk treatment

5.4.1. Risk treatment table

Type: Table
Name: Risk treatment table
Short description: Shows how the risks shall be treated
Concern: Treatment

Table 8: Treatment Identification Table

Risk ID/category	Treatment strategy	Description
R5	Avoid	Restriction on personell, give only highly trusted emloyees access to customers info
R6	Avoid	Restriction on personell, give only highly trusted emloyees access to customers info
R8	Reduce consequence	Cryptograp info
R9	Reduce consequence	Cryptograp info
R14	Reduce frequency	Install antivirus program

5.4.2. Risk treatment model

Type: UML Model
Name: Risk treatment model
Short description: Shows treatment of risks
Concern: Treatment

Figure 3: Risk treatment model