Ontological Modelling of E-Services to Ensure Appropriate Mobile Transactions

Vagan Terziyan

Faculty of Information Technology, University of Jyvaskyla, P.O. Box 35, FIN-40351,
Jyvaskyla, Finland, e-mail: vagan@it.jyu.fi

Abstract

The main goal of this paper is to provide simple ontological support to the mobile electronic commerce. The description of ontology-driven Transaction Monitor (TM) for mobile business applications is considered. The approach is based on assumption that the transaction management tool can be implemented in a mobile terminal, allowing integration of different distributed external e-services. We use the ontology-based framework for transaction management so that the TM will be able to manage transaction across multiple e-services and we consider management of distributed location-based services as an example of such ontology-based TM implementation. The core of the approach is ontologies. Ontologies should be "placed" both in mobile terminals and in e-services. They define common multiple clients - multiple services standards and vocabularies for the use of the names, types, schemas, default values for parameters, atomic service actions with appropriate structure of their input and output. In our implementation ontologies help to the TM to deal with multiple services during transactions and simplify appropriate user interface.

1 Introduction

M-commerce refers to e-commerce activities relying on mobile e-commerce transactions.

A mobile e-commerce transaction is any type of business transaction of an economic value that is conducted using a mobile terminal that communicates over a wireless telecommunications or Personal Area Network with the e-commerce infrastructure.

Transaction management was first developed in the database management context. In the modern sense the requirements were set up in the context of relational database management systems. The main targets of transaction management, concurrency control of the simultaneously running applications and recovery in the case of crashes, are taken care of automatically by the database management system. The executions should exhibit "ACID" properties. **A**tomicity means that either all the retrievals, updates and deletions within data are performed or all of its effects are aborted. **C**onsistency means that the transaction program is semantically correct, i.e. it keeps the database in a consistent state if run alone. **I**solation means that the concurrent executions of different transaction programs should behave towards the user and database as if there were no concurrency in running them. **D**urability means that the results of successfully terminated transaction program executions must persist in the state of database irrespective of the other concurrently running transactions or system crashes that might destroy the state of the system produced by the transaction. The results of the earlier studies on transaction management issues are exhaustively represented in [2] and [7].

There are some differences between e- and m-commerce transactions that should be taken into account in transaction management [10]:

- the mobile e-commerce environment is hostile in the sense that the customers or merchants might be traitorous;
- in the mobile e-commerce environment a terminal can easily be stolen and taken in an unauthorised use;
- the terminals in the mobile e-commerce have much less processor, memory and other resources;
- the security mechanisms are different ;
- existing m-commerce infrastructure is not homogeneous, but rather different from country to country;

- "location invalidation", i.e., transaction becoming invalid due to out-dated location information is specific to wireless.

Mobile E-commerce transactions are currently being developed in an industry-led consortium called MeT-forum [5]. The work has produced a public white paper [4] where the opportunities and risks of m-commerce are discussed. Scenarios (business models) for the m-commerce are currently being developed.

Location-based services (LBS) are based on the fact that the terminal has a position on the earth and this is made known to applications running on the infrastructure. The infrastructure can be running on mobile operator's sphere of control or on some external service provider. A typical query is: "Where am I now?", "Where is the cheapest restaurant that is 500 m away?", "Send me a taxi!" There are also another emerging applications [9].

The main goals of this paper are to provide the description of the TM based on assumption that it is an independent mobile terminal application, which can integrate different distributed external e-services by managing appropriate transactional processes. For that we use the ontology-based framework for transaction management so that the TM will be able to manage transaction across multiple e-services and we consider management of distributed location-based services as an example of such ontology-based TM implementation.

2 Ontology-based Transaction Monitor

Here we provide some background and the implementation basic for a TM based on ontologies.

2.1 Concept of Ontology-Based Transaction Management

The implementation of the TM we are basing on the assumption that the highest level of control functions related to transactions will be in hands of a user i.e. they should be implemented at the mobile terminal. This means that a user will be able to decide when to begin new transaction, when to stop or cancel it, when to switch from one service to another during the transaction, which values of parameters to use when submitting queries to a service and so on. Thus TM itself will be placed in a mobile terminal.

In general case we suppose that a user probably will need to contact several e-services to perform one transaction. Service better than user knows its recent offerings and the order of actions, which a user should do to get the service, he needs. This means that all interactions within one service will be better managed by a service itself. Such set of interactions we will call as a subtransaction and left the monitoring of it to a service. However we are leaving to a user the right to cancel active subtransaction and, due to atomicity requirements, return to the state, which was before the subtransaction started.

Mobile terminal should be able to manage situations when the contact with one service inside the transaction might be necessary to get an information, which is required as an input to deal with another service within the same transaction. To handle such cases a user should be sure that the Ids of such cross-parameters are the same for different services. For example, if one service returns to a user as output parameter "terminal location" and after that a user switches to LBS asking for a map around his location, then the LBS should recognize the input "terminal location" by the same way as the first service does.

To make possible to the TM to handle multiple service transactions and standardize e-services for that, we are presenting the concept of *ontology-based transaction management* (Figure 1) for implementation of the TM.

Every client in Figure 1, which in our case is mobile terminal, is equipped with a TM. Monitor was registered to several services and keeps basic *service data* about them, e.g. brief description, Id, contact address, the recent state of the monthly bill for the use of appropriate service and so on.

Client also keeps data about active transaction, e.g. current state of parameters, active subtransaction, last query and so on.

Every service in Figure 1 is equipped with a Subtransaction Monitor, which allows to a service to work with multiple clients and know current state of a subtransaction with each of them. This Monitor manages stored at the service basic data about clients, e.g. logins, passwords, contact addresses, the recent state of an active subtransaction with this client, recent value of monthly bill of this client and so on. Monitoring is based on a *service tree*, which keeps an order of basic service actions, offered by a service to its clients, and appropriate states of possible subtransactions with this service.

The core of the approach is ontologies (Figure 1). Ontologies should be placed both in mobile terminals and in services, which is actually our case, or should be easily accessed by both from a third party. Ontologies define common multiple clients - multiple services standards and vocabularies for the use of the names, types, schemas, default values for parameters, atomic service actions with appropriate structure of their input and output. In our implementation ontologies help to the TM to deal with multiple services during transactions and simplify appropriate user interface.

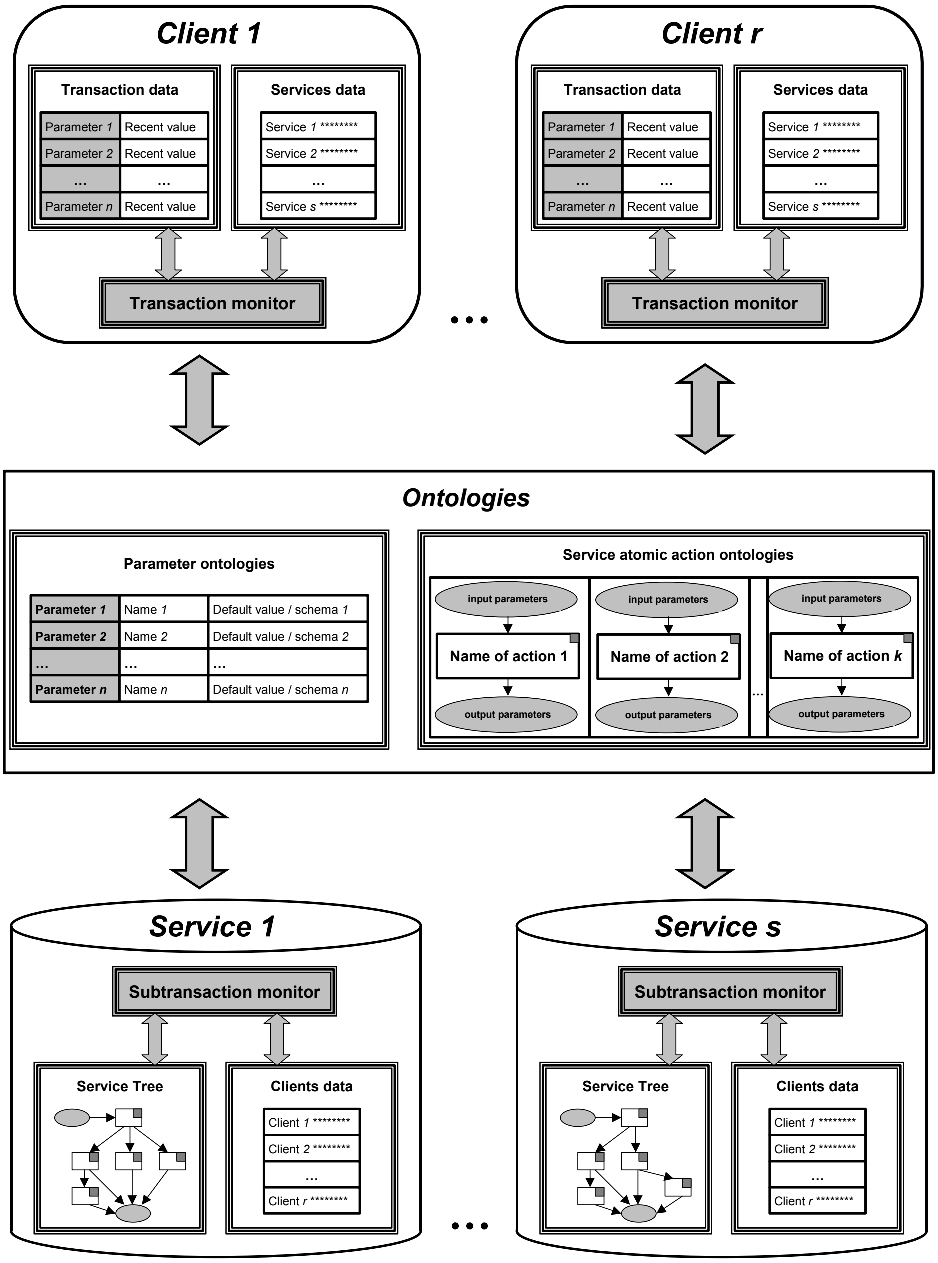


Figure 1. The conceptual scheme of the ontology-based transaction management

2.2 Some basic definitions

Let an *action* be a single client-server query-response session between the mobile terminal (hereinafter - terminal) and the e-service provider (hereinafter - service) as following:

$$A_i(x_1, x_2, ..., x_p, x_{p+1}, x_{p+2}, ..., x_{p+q}),$$

where A_i is action's Id; $x_1, x_1, ..., x_p$ - Ids of p input parameters for the action, which should be specified at the terminal to create a query; $x_{p+1}, x_{p+2}, ..., x_{p+q}$ - Ids of q output parameters of the action, which the terminal receives as the result to its query.

Subtransaction STR_i is a vector of one or more actions between a terminal and the service $Serv_j$ and appropriate states $S_0, ..., S_k$ managed by the service with definitely stated final goal and common memory of parameters:

$$STR_i : \{S_0; LOGIN[S_1], A_1[S_2], A_2[S_3], ..., A_{k-1}[S_k], LOGOUT[S_0]\}_{Serv_j},$$

where $S_0 = 0$ is an initial state of the subtransaction, $A_i[S_{i+1}]$ means that after performing the action A_i the subtransaction will come to state S_{i+1}, *LOGIN* and *LOGOUT* are two obligatory actions, which are marginal for every service.

Transaction TR_l is a vector of one or more subtransactions with the same terminal $Term_f$ and possibly different services managed by the terminal, with definitely stated final goal and common memory of parameters:

$$TR_l : \{STR_1, STR_2, ..., STR_r\}_{Term_f}.$$

Service tree is a structured set of subtransactions, which a service can offer to his clients and which is used by a service to manage subtransactions with clients (Figure 2). A subtransaction is any route in a tree from *Login* to *Logout* nodes, which includes at least one action of interest. *Action of interest*, toned for every subtransaction in the service tree in Figure 2, is such an action, which outcome is in particular interest of a customer and has an economic value.

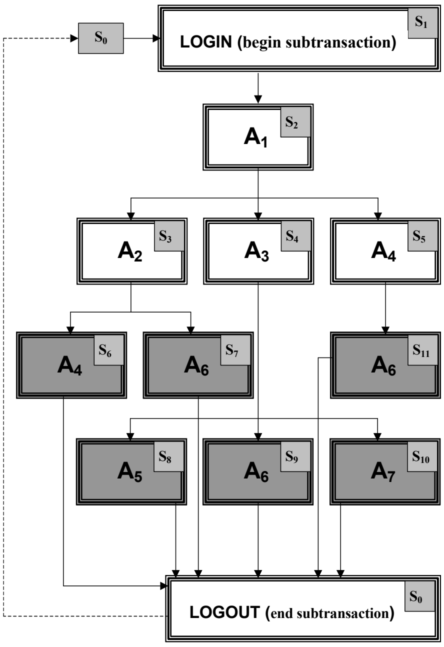


Figure 2. An example of a Service tree as a collection of subtransactions offered by the Service to its customers. In the rectangles together with the Id of an action there is also Id of a state, into which an appropriate subtransaction is coming after performing this action

2.3 Constants, ontologies and variables

In the TM model we consider a group of constants, which are defined by initial settings of the monitor (See Table 1). The group consists of:

1) *basic constants*, which define Ids of the terminal and services used, basic screens for the interface, total numbers of services, actions and parameters, which TM is operating with;
2) *service atomic actions ontologies* define basic actions with their input and output, from which every service can be composed, and which are used as a common procedural language between a client and a service (include always *LOGIN* and *LOGOUT* actions ontologies);
3) *parameter ontologies* describe parameters, which can be used in actions, by providing their Ids, default values and types (or schemas), and which are actually a common declarative language between a client and a service.

Table 1. Basic constants and ontologies of the TM

ID of the Constant	Dimension	Value
Basic constants:		
TERMINAL_ID	1	From settings
TOTAL_NUMBER_OF_SERVICES	1	From settings
TOTAL_NUMBER_OF_ACTIONS	1	From settings
TOTAL_NUMBER_OF_PARAMETERS	1	From settings
SERVICE_ID	TOTAL_NUMBER_OF_SERVICES	From settings
SCREEN_FRAME	16	From settings
Service atomic action ontologies:		
ACTION_ID	TOTAL_NUMBER_OF_ACTIONS	From settings
INPUT_PARAMETERS_FOR_ACTION	TOTAL_NUMBER_OF_ACTIONS × TOTAL_NUMBER_OF_PARAMETERS	From settings
OUTPUT_PARAMETERS_FROM_ACTION	TOTAL_NUMBER_OF_ACTIONS × TOTAL_NUMBER_OF_PARAMETERS	From settings
Parameter ontologies:		
PARAMETER_ID	TOTAL_NUMBER_OF_PARAMETERS	From settings
PARAMETER_DEFAULT_VALUE	TOTAL_NUMBER_OF_PARAMETERS	From settings
PARAMETER_TYPE/SCHEMA	TOTAL_NUMBER_OF_PARAMETERS	From settings

In the TM model we consider three groups of variables:

1) *control variables* (Table 2) have sense only for a TM and are used to manage different states of the terminal during going-on transactions, subtransactions and actions;

2) *working variables* (Table 3) are used to manage parameters' states and provide common memory for different subtransactions, which can be run with different services. PARAMETER_CANCEL_ SUBTRANSACTION_VALUE is used to guarantee atomicity of a subtransaction (if for some reason a subtransaction cannot normally be finished, then the value of each parameter from the very beginning of the subtransaction will be restored);
3) *billing variables* (Table 4) are used to manage billing data in the TM. The terminal will collect bills separately for every service adding online price for appropriate service actions to it, when it is requested.

Table 2. Control variables of the TM

ID of the Control Variable	Dimension	Initial Value
CURRENT_STATE_OF_TRANSACTION	1	0
CURRENT_STATE_OF_SUBTRANSACTION	1	0
LIST_OF_AVAILABLE_ACTIONS	TOTAL_NUMBER_OF_ACTIONS	0
ACTIVE_ACTION_ID	1	0
ACTIVE_PARAMETER_ID	1	0
ATOMICITY_PROTECTOR	1	0

Table 3. Working variables of the TM

ID of the Working Variable	Dimension	Initial Value
PARAMETER_RECENT_VALUE	TOTAL_NUMBER_OF_ PARAMETERS	PARAMETER_DEFAULT_VALUE
PARAMETER_CANCEL_SUBTRANSACTION _VALUE	TOTAL_NUMBER_OF_ PARAMETERS	PARAMETER_DEFAULT_VALUE
SCREEN	1	Screen 1

Table 4. Billing variables of the TM

ID of the Billing Variable	Dimension	Initial Value
BILL_RECENT_VALUE	TOTAL_NUMBER_OF_SERVICES	0
PRICE_FOR_LAST_ACTION	1	0

2.4 Actions: query-response sessions

Service action in our model is a single query-response session. Formats of service queries, which a mobile terminal can submit to a service and appropriate responses are given in Figure 3 (a-b). Being a part of a subtransaction this query-response session change a subtransaction state from one to another, according to a service tree. There are also *control actions* possible to be used to protect the subtransaction atomicity (like "cancel query" or "repeat response", etc.).

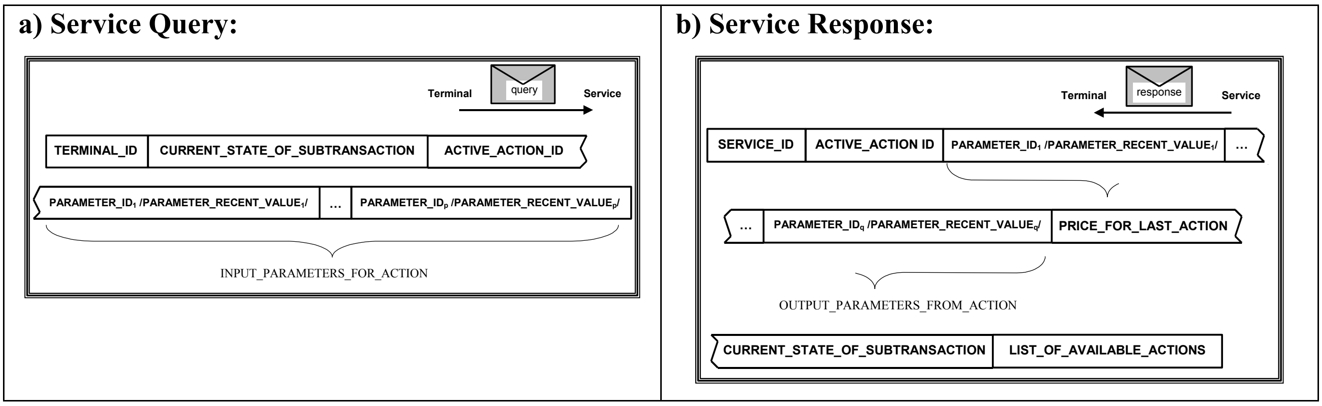

a) Service Query:
Terminal
query
Service
TERMINAL_ID
CURRENT_STATE_OF_SUBTRANSACTION
ACTIVE_ACTION_ID
PARAMETER_ID₁ /PARAMETER_RECENT_VALUE₁/
...
PARAMETER_IDₚ /PARAMETER_RECENT_VALUEₚ/
INPUT_PARAMETERS_FOR_ACTION
b) Service Response:
Terminal
response
Service
SERVICE_ID
ACTIVE_ACTION ID
PARAMETER_ID₁ /PARAMETER_RECENT_VALUE₁/
...
...
PARAMETER_ID_q /PARAMETER_RECENT_VALUE_q/
PRICE_FOR_LAST_ACTION
OUTPUT_PARAMETERS_FROM_ACTION
CURRENT_STATE_OF_SUBTRANSACTION
LIST_OF_AVAILABLE_ACTIONS

Figure 3. Formats for query (a) and response (b) of a service action in terms of constants, ontologies and variables.

An example of two service actions performed is shown in Figure 4.

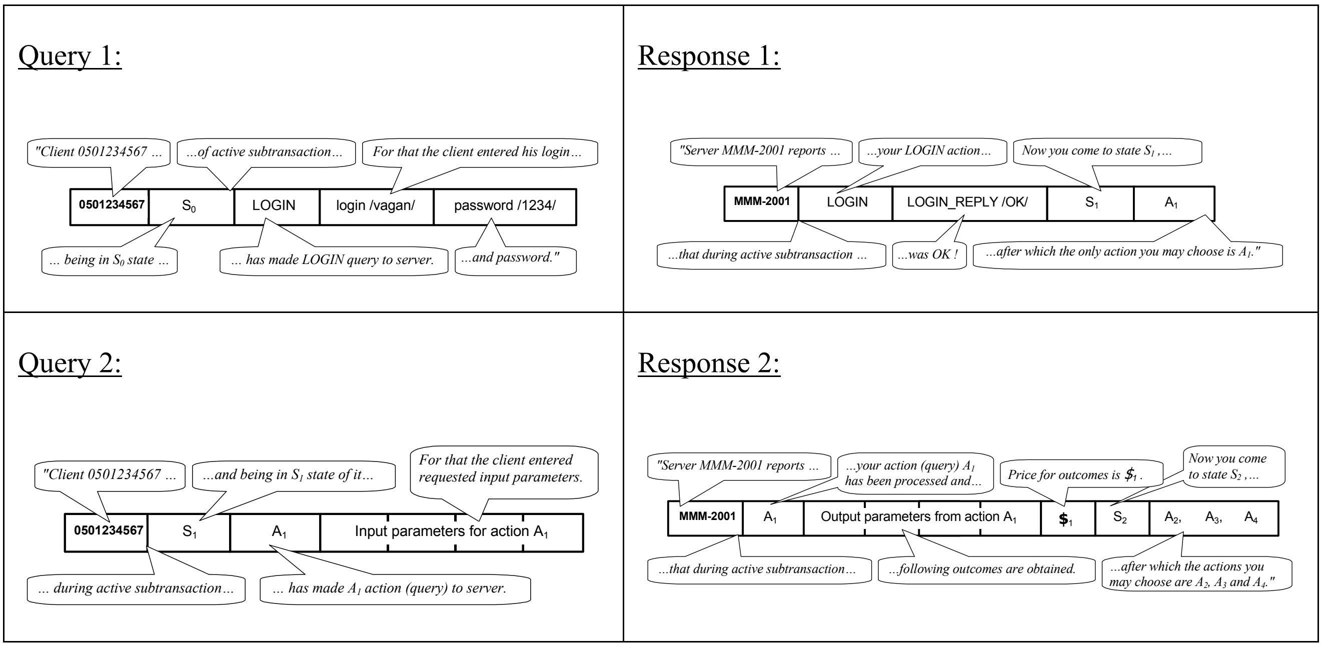

Query 1:
"Client 0501234567 ...
...of active subtransaction...
For that the client entered his login...
0501234567
S₀
LOGIN
login /vagan/
password /1234/
... being in S₀ state ...
... has made LOGIN query to server.
...and password."
Response 1:
"Server MMM-2001 reports ...
...your LOGIN action...
Now you come to state S₁ ,...
MMM-2001
LOGIN
LOGIN_REPLY /OK/
S₁
A₁
...that during active subtransaction ...
...was OK !
...after which the only action you may choose is A₁."
Query 2:
"Client 0501234567 ...
...and being in S₁ state of it...
For that the client entered requested input parameters.
0501234567
S₁
A₁
Input parameters for action A₁
... during active subtransaction...
... has made A₁ action (query) to server.
Response 2:
"Server MMM-2001 reports ...
...your action (query) A₁ has been processed and...
Price for outcomes is $₁ .
Now you come to state S₂ ,...
MMM-2001
A₁
Output parameters from action A₁
$₁
S₂
A₂, A₃, A₄
...that during active subtransaction...
...following outcomes are obtained.
...after which the actions you may choose are A₂, A₃ and A₄."

Figure 4. An example of two performed actions (client-server query-response sessions) between the Terminal and the Service. These are first two actions of the subtransaction according to the Service Tree from Figure 2

3 Ontology-based Transaction Monitor for m-commerce location-based services

Here we consider the implementation of the ontology-based TM to manage transactions for location-based services. We consider the model example of two services, LBS and positioning service, across which the TM will perform transactions. For that we will define necessary model ontologies and show basic stages of transactional process with the Monitor.

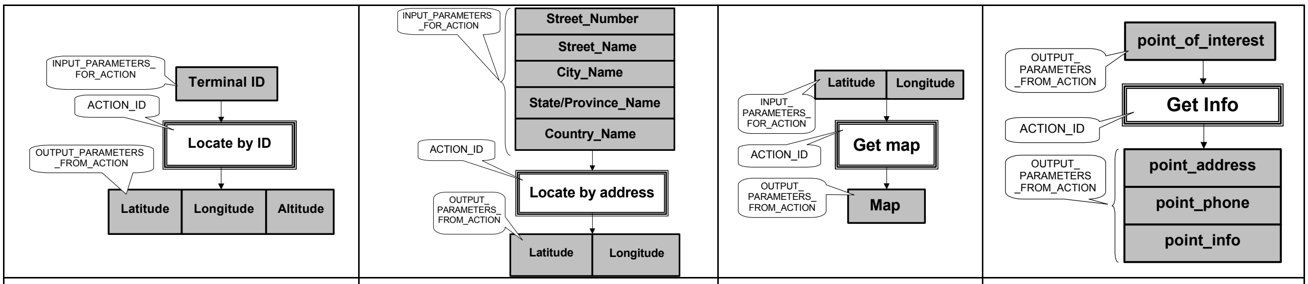
3.1 Constants and ontologies for the LBS model example

We will consider the following basic constants and the subset of parameters, actions and appropriate ontologies to describe the LBS-type services as presented in Table 5 and Figures 5a-d.

Table 5. Constants and ontologies for the LBS

ID of the Constant	Value
Basic constants:	
TERMINAL_ID	From settings
TOTAL_NUMBER_OF_SERVICES	2
TOTAL_NUMBER_OF_ACTIONS	3
TOTAL_NUMBER_OF_PARAMETERS	9
SERVICE_ID [1]	POSITIONING_SERVICE
SERVICE_ID [2]	LOCATION_BASED_SERVICE
Service atomic action ontologies:	
ACTION_ID [1]	LOCATE_BY_ID
ACTION_ID [2]	LOCATE_BY_ADDRESS
ACTION_ID [3]	GET_MAP
INPUT_PARAMETERS_FOR_ACTION [1,*]	TERMINAL_ID
OUTPUT_PARAMETERS_FROM_ACTION [1,*]	LATITUDE LONGITUDE ALTITUDE
INPUT_PARAMETERS_FOR_ACTION [2,*]	STREET_NUMBER STREET_NAME CITY_NAME STATE/PROVINCE_NAME COUNTRY_NAME
OUTPUT_PARAMETERS_FROM_ACTION [2,*]	LATITUDE LONGITUDE
INPUT_PARAMETERS_FOR_ACTION [3,*]	LATITUDE LONGITUDE
OUTPUT_PARAMETERS_FROM_ACTION [3,*]	MAP
Parameter ontologies:	
PARAMETER_ID [1]	LATITUDE
PARAMETER_DEFAULT_VALUE [1]	0 [or optional: latitude of Jyvaskyla RW Station]
PARAMETER_TYPE/SCHEMA [1]	32-bit signed integer
PARAMETER_ID [2]	LONGITUDE
PARAMETER_DEFAULT_VALUE [2]	0 [or optional: longitude of Jyvaskyla RW Station]
PARAMETER_TYPE/SCHEMA [2]	32-bit signed integer
PARAMETER_ID [3]	ALTITUDE
PARAMETER_DEFAULT_VALUE [3]	0 [or optional: altitude of Jyvaskyla RW Station]
PARAMETER_TYPE/SCHEMA [3]	32-bit signed integer
PARAMETER_ID [4]	STREET_NUMBER
PARAMETER_DEFAULT_VALUE [4]	16
PARAMETER_TYPE/SCHEMA [4]	2-byte unsigned integer
PARAMETER_ID [5]	STREET_NAME
PARAMETER_DEFAULT_VALUE [5]	HANNIKAISENKATU
PARAMETER_TYPE/SCHEMA [5]	ASCII text
PARAMETER_ID [6]	CITY_NAME
PARAMETER_DEFAULT_VALUE [6]	JYVASKYLA
PARAMETER_TYPE/SCHEMA [6]	ASCII text

PARAMETER_ID [7]	STATE/PROVINCE_NAME
PARAMETER_DEFAULT_VALUE [7]	CENTRAL_FINLAND
PARAMETER_TYPE/SCHEMA [7]	ASCII text
PARAMETER_ID [8]	COUNTRY_NAME
PARAMETER_DEFAULT_VALUE [8]	FINLAND
PARAMETER_TYPE/SCHEMA [8]	ASCII text
PARAMETER_ID [9]	MAP
PARAMETER_DEFAULT_VALUE [9]	Map around Jyvaskyla Railway Station
PARAMETER_TYPE/SCHEMA [9]	GML file

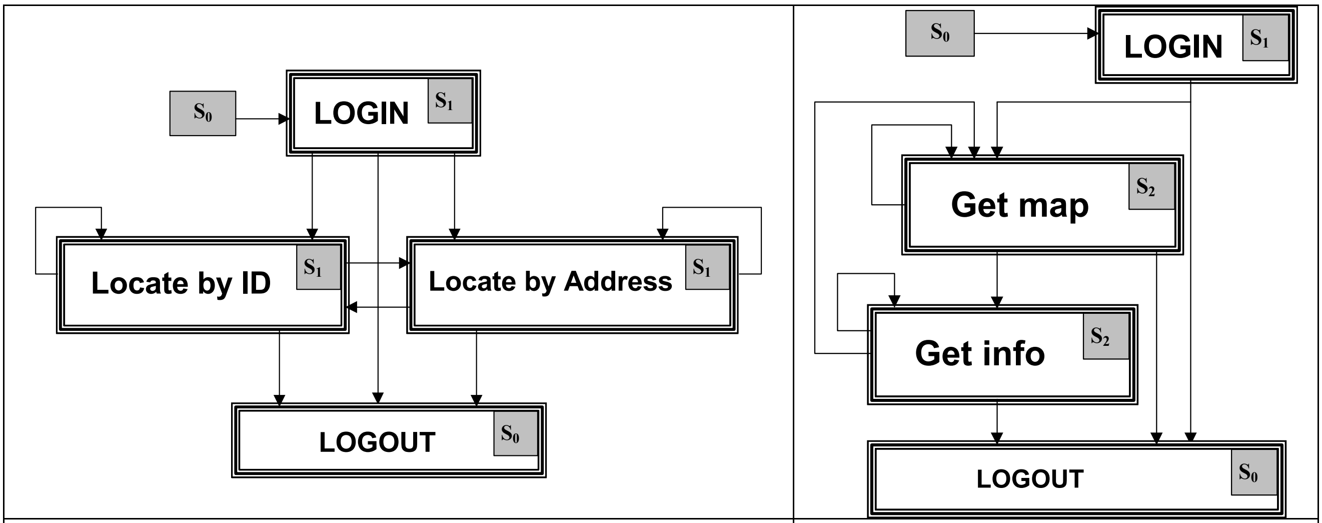


a) "LOCATE_BY_ID" **b)** "LOCATE_BY_ADDRESS" **c)** "GET_MAP" **d)** "GET_INFO"

Figure 5. Action ontologies for the LBS

3.2 Service trees for the LBS model example

We are considering two services: positioning service and location-based service (LBS). Suppose that the positioning service performs two actions - locating the mobile terminal based on its ID or locating a user based on a street address (i.e. transfer submitted street address to the coordinates). Based on location coordinates, the LBS can deliver an appropriate Map to a user's terminal and, if user selects an appropriate point of interest on this map, the LBS delivers appropriate information about selected point to a user's terminal. The appropriate service trees for positioning service and LBS can be presented as it shown in Figures 6 (a) and (b) 11 respectively.



a) Possible service tree of the Positioning Service

b) Possible service tree of the Location Based Service

Figure 6. Possible service tree of the services involved

3.3 Transaction sequence diagram for the LBS model example

Assume that a user of the terminal, which is equipped by the TM, is a registered user of the two above-mentioned services (Positioning Service and LBS).

Consider following scenario. A user (tourist), being in some new for him area, needs to find information about one of a nearest hotels and find the route how to reach it. Assume also that a user is standing in such a place that he is able to see his street address. Let the address be "43 Nokatu, Jyvaskyla, Finland". First the tourist uses the address to get his coordinates from positioning service and then uses the coordinates to get map around his location from LBS. LBS provides requested map showing available points of interests on it (including hotels). The user selects one of nearest hotels from the map and makes request to LBS to deliver information about it. Finally the LBS delivers requested information about selected point including necessary contact details.

Example scenario of the use of the TM to get necessary information by managing transactions across two services are shown in Figure 7 in the form of sequence diagrams.

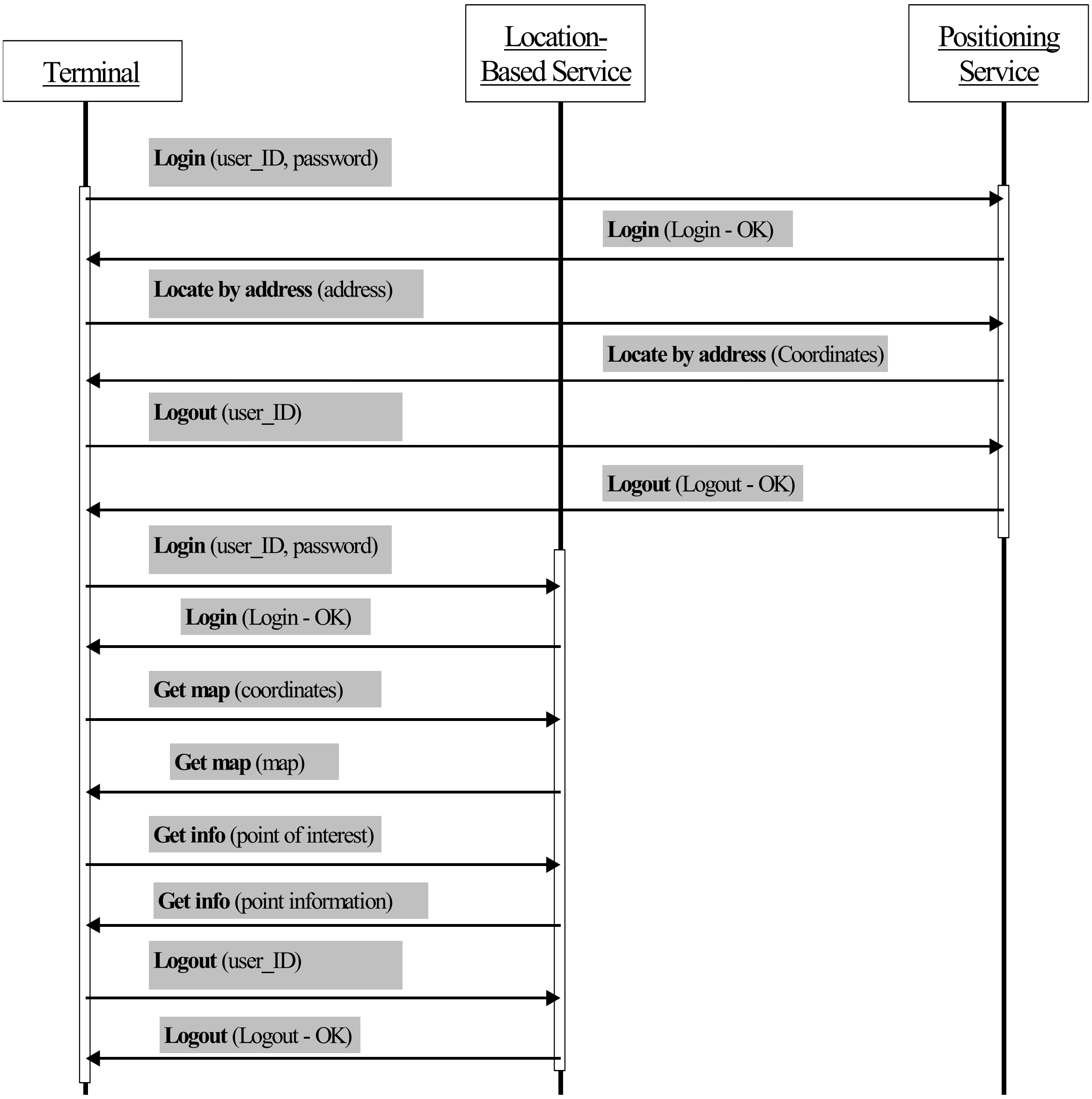
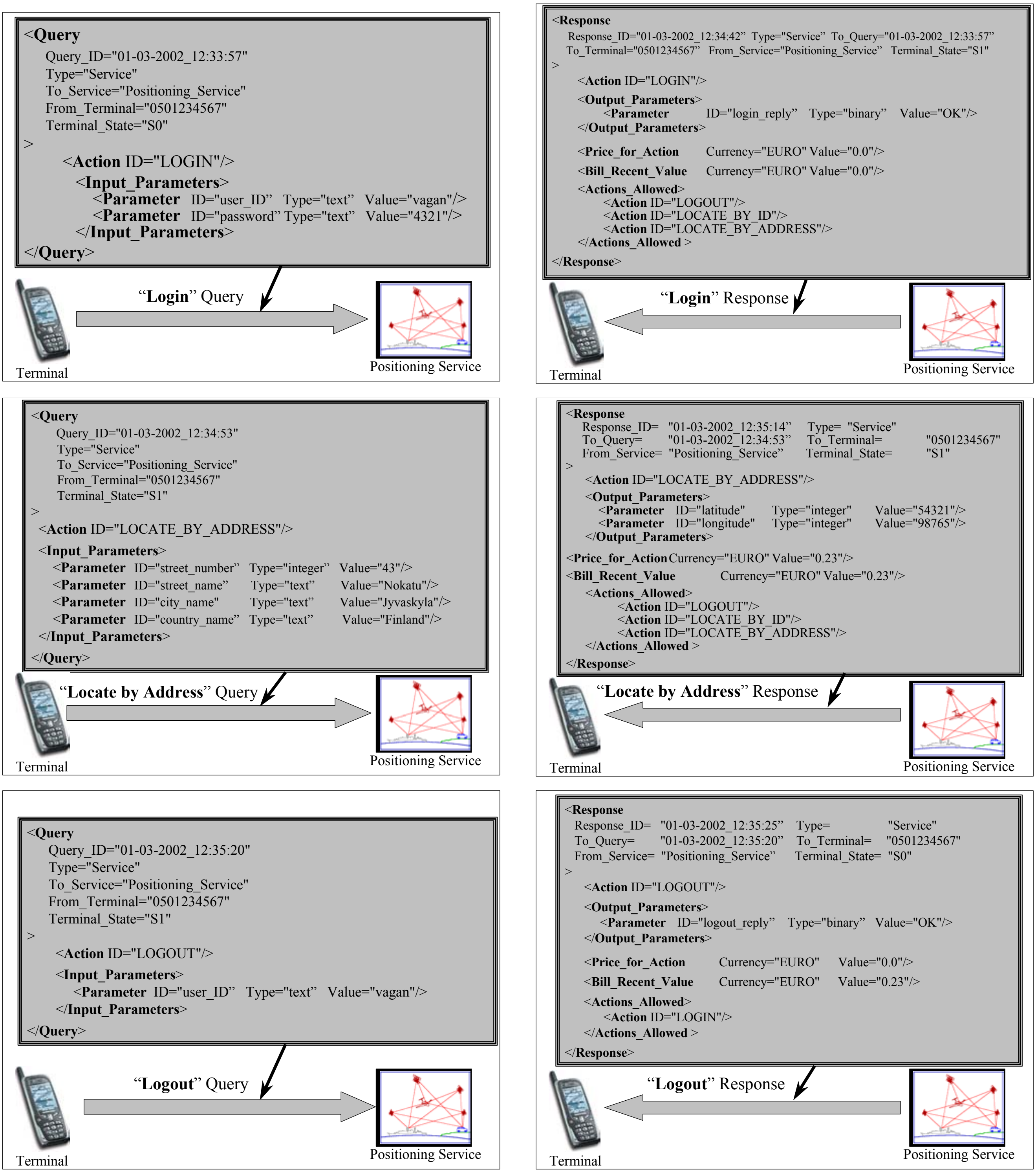


Figure 7. Sequence diagrams for the LBS example

Notice that these are the model examples and for commercial implementation these scenarios as well as TM functionality can be optimised and further developed.

If we consider the whole cycle of actions from sending address of current location until getting information about desirable hotel as a entire terminal-based transaction. It consists of two subtransactions, first one with Positioning Service and second one with LBS. The goal of first subtransaction is to get coordinates of current location and the goal of second subtransaction is to get information about the hotel. In Figure 8 the actions (query-response pairs described in XML) are shown related to the subtransaction with Positioning Services. In Figure 9 the actions related to the subtransaction with LBS are presented with similar description.



```
<Query
    Query_ID="01-03-2002_12:33:57"
    Type="Service"
    To_Service="Positioning_Service"
    From_Terminal="0501234567"
    Terminal_State="S0"
>
        <Action ID="LOGIN"/>
          <Input_Parameters>
             <Parameter  ID="user_ID"  Type="text"  Value="vagan"/>
             <Parameter  ID="password" Type="text"  Value="4321"/>
          </Input_Parameters>
</Query>
```

```
<Response
   Response_ID="01-03-2002_12:34:42"  Type="Service"  To_Query="01-03-2002_12:33:57"
   To_Terminal="0501234567"  From_Service="Positioning_Service"  Terminal_State="S1"
>
     <Action ID="LOGIN"/>
     <Output_Parameters>
          <Parameter        ID="login_reply"  Type="binary"   Value="OK"/>
     </Output_Parameters>
     <Price_for_Action      Currency="EURO" Value="0.0"/>
     <Bill_Recent_Value     Currency="EURO" Value="0.0"/>
     <Actions_Allowed>
          <Action ID="LOGOUT"/>
          <Action ID="LOCATE_BY_ID"/>
          <Action ID="LOCATE_BY_ADDRESS"/>
     </Actions_Allowed >
</Response>
```

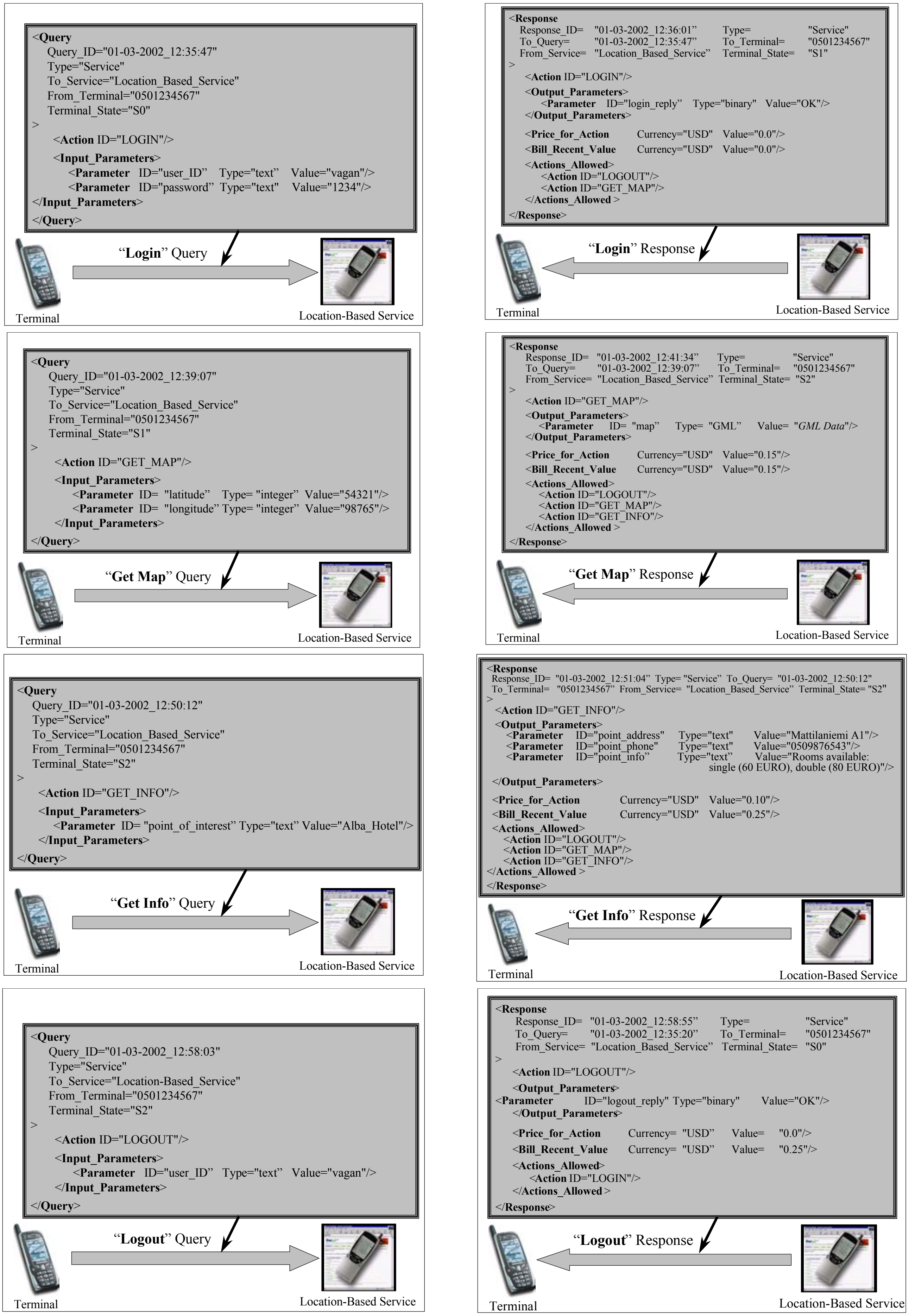
```
<Query
    Query_ID="01-03-2002_12:34:53"
    Type="Service"
    To_Service="Positioning_Service"
    From_Terminal="0501234567"
    Terminal_State="S1"
>
  <Action ID="LOCATE_BY_ADDRESS"/>
  <Input_Parameters>
     <Parameter  ID="street_number"  Type="integer"  Value="43"/>
     <Parameter  ID="street_name"    Type="text"     Value="Nokatu"/>
     <Parameter  ID="city_name"      Type="text"     Value="Jyvaskyla"/>
     <Parameter  ID="country_name"   Type="text"     Value="Finland"/>
  </Input_Parameters>
</Query>
```

```
<Response
   Response_ID=  "01-03-2002_12:35:14"   Type=  "Service"
   To_Query=     "01-03-2002_12:34:53"   To_Terminal=        "0501234567"
   From_Service= "Positioning_Service"   Terminal_State=     "S1"
>
    <Action ID="LOCATE_BY_ADDRESS"/>
    <Output_Parameters>
      <Parameter  ID="latitude"    Type="integer"    Value="54321"/>
      <Parameter  ID="longitude"   Type="integer"    Value="98765"/>
    </Output_Parameters>
<Price_for_Action Currency="EURO" Value="0.23"/>
<Bill_Recent_Value         Currency="EURO" Value="0.23"/>
    <Actions_Allowed>
          <Action ID="LOGOUT"/>
          <Action ID="LOCATE_BY_ID"/>
          <Action ID="LOCATE_BY_ADDRESS"/>
    </Actions_Allowed >
</Response>
```

```
<Query
    Query_ID="01-03-2002_12:35:20"
    Type="Service"
    To_Service="Positioning_Service"
    From_Terminal="0501234567"
    Terminal_State="S1"
>
    <Action ID="LOGOUT"/>
    <Input_Parameters>
       <Parameter  ID="user_ID"  Type="text"  Value="vagan"/>
    </Input_Parameters>
</Query>
```

```
<Response
  Response_ID=  "01-03-2002_12:35:25"   Type=             "Service"
  To_Query=     "01-03-2002_12:35:20"   To_Terminal=   "0501234567"
  From_Service= "Positioning_Service"   Terminal_State=  "S0"
>
    <Action ID="LOGOUT"/>
    <Output_Parameters>
       <Parameter  ID="logout_reply"  Type="binary"  Value="OK"/>
    </Output_Parameters>
    <Price_for_Action      Currency="EURO"    Value="0.0"/>
    <Bill_Recent_Value     Currency="EURO"    Value="0.23"/>
    <Actions_Allowed>
      <Action ID="LOGIN"/>
    </Actions_Allowed >
</Response>
```

Figure 8. XML-based query-response sessions with Positioning Service in the LBS example



```
<Query
    Query_ID="01-03-2002_12:35:47"
    Type="Service"
    To_Service="Location_Based_Service"
    From_Terminal="0501234567"
    Terminal_State="S0"
>
    <Action ID="LOGIN"/>
    <Input_Parameters>
        <Parameter  ID="user_ID"   Type="text"   Value="vagan"/>
        <Parameter  ID="password"  Type="text"   Value="1234"/>
</Input_Parameters>
</Query>
```

```
<Response
   Response_ID=   "01-03-2002_12:36:01"      Type=            "Service"
   To_Query=      "01-03-2002_12:35:47"      To_Terminal=     "0501234567"
   From_Service=  "Location_Based_Service"   Terminal_State=  "S1"
>
    <Action ID="LOGIN"/>
    <Output_Parameters>
        <Parameter  ID="login_reply"  Type="binary"  Value="OK"/>
    </Output_Parameters>
    <Price_for_Action      Currency="USD"   Value="0.0"/>
    <Bill_Recent_Value     Currency="USD"   Value="0.0"/>
    <Actions_Allowed>
        <Action ID="LOGOUT"/>
        <Action ID="GET_MAP"/>
    </Actions_Allowed >
</Response>
```

```
<Query
    Query_ID="01-03-2002_12:39:07"
    Type="Service"
    To_Service="Location_Based_Service"
    From_Terminal="0501234567"
    Terminal_State="S1"
>
    <Action ID="GET_MAP"/>
    <Input_Parameters>
        <Parameter ID=  "latitude"   Type= "integer"  Value="54321"/>
        <Parameter ID=  "longitude"  Type= "integer"  Value="98765"/>
    </Input_Parameters>
</Query>
```

```
<Response
    Response_ID=   "01-03-2002_12:41:34"      Type=            "Service"
    To_Query=      "01-03-2002_12:39:07"      To_Terminal=     "0501234567"
    From_Service=  "Location_Based_Service"  Terminal_State=  "S2"
>
    <Action ID="GET_MAP"/>
    <Output_Parameters>
       <Parameter    ID=  "map"    Type=  "GML"    Value=  "GML Data"/>
    </Output_Parameters>
    <Price_for_Action      Currency="USD"   Value="0.15"/>
    <Bill_Recent_Value     Currency="USD"   Value="0.15"/>
    <Actions_Allowed>
       <Action ID="LOGOUT"/>
       <Action ID="GET_MAP"/>
       <Action ID="GET_INFO"/>
    </Actions_Allowed >
</Response>
```

```
<Query
    Query_ID="01-03-2002_12:50:12"
    Type="Service"
    To_Service="Location_Based_Service"
    From_Terminal="0501234567"
    Terminal_State="S2"
>
    <Action ID="GET_INFO"/>
    <Input_Parameters>
       <Parameter  ID= "point_of_interest" Type="text" Value="Alba_Hotel"/>
    </Input_Parameters>
</Query>
```

```
<Response
 Response_ID=  "01-03-2002_12:51:04"  Type= "Service"  To_Query=  "01-03-2002_12:50:12"
 To_Terminal=   "0501234567"  From_Service=  "Location_Based_Service"  Terminal_State= "S2"
>
  <Action ID="GET_INFO"/>
  <Output_Parameters>
     <Parameter   ID="point_address"   Type="text"     Value="Mattilaniemi A1"/>
     <Parameter   ID="point_phone"     Type="text"     Value="0509876543"/>
     <Parameter   ID="point_info"      Type="text"     Value="Rooms available:
                                                   single (60 EURO), double (80 EURO)"/>
 </Output_Parameters>
 <Price_for_Action          Currency="USD"   Value="0.10"/>
 <Bill_Recent_Value         Currency="USD"   Value="0.25"/>
 <Actions_Allowed>
    <Action ID="LOGOUT"/>
    <Action ID="GET_MAP"/>
    <Action ID="GET_INFO"/>
</Actions_Allowed >
</Response>
```

```
<Query
    Query_ID="01-03-2002_12:58:03"
    Type="Service"
    To_Service="Location-Based_Service"
    From_Terminal="0501234567"
    Terminal_State="S2"
>
    <Action ID="LOGOUT"/>
    <Input_Parameters>
        <Parameter  ID="user_ID"  Type="text"  Value="vagan"/>
    </Input_Parameters>
</Query>
```

```
<Response
     Response_ID=  "01-03-2002_12:58:55"      Type=            "Service"
     To_Query=     "01-03-2002_12:35:20"      To_Terminal=     "0501234567"
     From_Service=  "Location_Based_Service"  Terminal_State=  "S0"
>
    <Action ID="LOGOUT"/>
    <Output_Parameters>
<Parameter       ID="logout_reply" Type="binary"     Value="OK"/>
    </Output_Parameters>
    <Price_for_Action      Currency=  "USD"    Value=    "0.0"/>
    <Bill_Recent_Value     Currency=  "USD"    Value=    "0.25"/>
    <Actions_Allowed>
       <Action ID="LOGIN"/>
    </Actions_Allowed >
</Response>
```

Figure 9. XML-based query-response sessions with Location-Based Service in the LBS example

4 Related work

According to MeT "Consistent User Experience" framework [3] the user interface should allow to people to transfer their knowledge and skills from one application to any other application. Consistency of visual interface and terminology helps people to learn and then easily recognize the "language" of the interface. The TM, due to implementation of the concept of ontology-based transaction management, offers such a consistent standardize user interface with multiple services.

E-speak is an open software platform designed specifically for the development, deployment, intelligent interaction, and management of globally distributed e-services [12]. E-Speak makes services capable to interact with each other on behalf of their users, and compose themselves into more complex services. The E-Speak Service Engine [12] actually is a TM software that performs the intelligent interaction of e-services. The TM in the hands of user is a good example of an E-Speak engine, which expands the E-Speak internet-based framework to wireless.

OntoWeb [6] - Ontology-based information exchange for knowledge management and electronic commerce is the IST Project Thematic Network of 64 academic and industrial partners inside and outside Europe. The goal of the OntoWeb Network is to bring together researchers and industrials promoting interdisciplinary work and strengthening the European influence on Semantic Web standardisation efforts such as those based on RDF and XML. The implementation of the concept of ontology-based transaction management for mobile terminal allows expanding a target for the Semantic Web framework also to m-commerce.

DAML-S [1] - semantic markup for Web services is one of the Semantic Web community efforts to enable not only content but also services on the Web. It will enable users and software agents to automatically discover, invoke, compose, and monitor Web resources offering services, under specified constraints. The DAML-S coalition is developing ontology of services, service profiles and appropriate process model. The use of DAML (DARPA Agent Markup Language) supposes that artificial agents will do most of future transactions across multiple web services. The mobile terminal-based TM is one step towards agent based future of e-services in a mobile environment.

5 Conclusion

In this paper we assume that TM is an independent mobile terminal application, which can integrate different distributed external e-services by managing appropriate transactional processes. For that we use the ontology-based framework for transaction management so that the TM will be able to manage transaction across multiple e-services and we consider management of distributed location-based services as an example of such ontology-based TM implementation.

Another possible approach, which was described in [8] was based on the assumption that some specific terminal-based application is already exists, which supports certain transactions with certain services. In that case the TM was used as a tool to guarantee basic transactional properties of that transactions, i.e. protect the application's data from terminations related to the specifics of a mobile device. TM in this case will be fully controlled by the application.

With the increasing market of electronic commerce it becomes an interesting aspect to use autonomous mobile agents for electronic business transactions. Being involved in money transactions, supplementary security features for mobile agent systems have to be ensured. In [11] architecture was presented for a mobile agent system which offers fault tolerance for the whole agent system at a high level. This architecture pretends to guarantee security for the host as well as security for the agent. To handle these issues for mobile agents we use various encryption mechanisms and a novel method was applied for mobile agent systems by using distributed transactions in the architecture. Due to this security architecture an agent will be enabled to carry out money transactions.

All three approaches seem to be reasonable to integrate to be used in the appropriate context.

Acknowledgements

The economic support of the Multimeetmobile project by the National Technology Agency of Finland (TEKES), Nokia, Hewlett Packard Finland, and Yomi Vision under contract 40599/99 is highly appreciated. Useful discussions and suggestions from Prof. J. Veijalainen and other colleagues from the Multimeetmobile project team are also highly appreciated.

References

1. A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng, DAML-S: Semantic Markup for Web Services, May 2002, available in: http://www.daml.org/services/daml-s/2001/05/daml-s.html.
2. P. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addision-Wesley, 1987.
3. MeT Consistent User Experience, Version 1.0, 21 February 2001, available in: http://www.mobiletransaction.org.
4. MeT Overview White Paper, Version 2.0, 29 January 2001, available in: http://www.mobiletransaction.org/pdf/White%20Paper_2.0.pdf.
5. Mobile Electronic Transactions Forum, available in: http://www.mobiletransaction.org/.
6. OntoWeb: Ontology-Based Information Exchange for Knowledge Management and Electronic Commerce, 2000, available in: http://www.ontoweb.org.
7. C. Papadimitriou, The Theory of Database Concurrency Control, Computer Science Press, 1986.
8. V. Terziyan, J. Veijalainen, M-Commerce Transaction Model Implementation at a Mobile Terminal, Multimeetmobile Project Report, TITU, Univ. of Jyvaskyla, 9 May 2001, 56 pp.
9. V. Terziyan, J. Veijalainen, H. Tirri, Mobile e-Commerce Transaction Model, Multimeetmobile Project Report, TITU, Univ. of Jyvaskyla, 18 December 2000, 48 pp.
10. J. Veijalainen, Transactions in Mobile Electronic Commerce, LNCS, Vol. 1773, Springer, 1999, pp. 208-229.
11. H. Vogler, T. Kunkelmann, M.-L. Moschgath, Distributed Transaction Processing as a Reliability Concept for Mobile Agents, 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97), October 29-31, 1997.
12. What is E-Speak? Product Information, 2001, Hewlett Packard Company, available in: http://www.e-speak.hp. com/product/overview.shtm.