Sensoria

016004
Software Engineering for Service-Oriented Overlay Computers

Automotive Case Study: UML Specification of On Road Assistance Scenario

FAST Report # 1, 2007

Author: Nora Koch

Date of preparation: August 27th, 2007
Revision: v0.8
Classification: PU

Contract Start Date: September 1, 2005 Duration: 48 months
Project Coordinator: LMU
Partners: LMU, UNITN, ULEICES, UWARSAW, DTU, PISA, DSIUF, UNIBO, ISTI, FFCUL, UEDIN, ATX, TILab, FAST, BUTE, S&N, LSS-Imperial, LSS-UCL, MIP, ATXT

Integrated Project funded by the European Community under the "Information Society Technologies" Programme (2002—2006)

Executive Summary

This report summarises and describes the main building blocks of the Automotive Case Study defined within the scope of the SENSORIA project. The document includes the specification of the automotive architecture and the *On Road Assistance* scenario (also called *On Road Repair* or *Low Oil Level* scenario). The specification language UML 2.0 is selected for modelling the application. The UML 2.0 is extended using the extension mechanisms provided by the UML and resulting in the SENSORIA UML profile [3], in order to provide to the models service-oriented architecture (SOA) specific semantics. The extension defines stereotypes for specific structural and behavioural aspects of service-oriented computing. It includes concepts like *service*, *service interface*, *service provider* and *requester* for the specification of the structure of a SOA and *send, receive,* and *compensate* for the specification of orchestration of services.

The application is modelled using the CASE tool IBM Rational Software Modeler v7 [7].

Contents

1 Introduction

This document includes the specification of the automotive architecture and the *On Road Assistance* scenario, which in other SENSORIA documents is called “on road repair” or “low oil level" scenario. The specification language UML 2.0 is selected for modelling the application. The UML 2.0 is extended within the scope of SENSORIA using the extension mechanisms provided by the UML, in order to express service-oriented semantics in service-oriented architecture (SOA) models. The extension is defined as a UML profile. For further details, the reader is referred to D1.4a describing the SENSORIA UML profile for service-oriented computing [3]. The extension defines stereotypes for structural and behavioural aspects in SOA models. It includes concepts like *service*, *service interface*, *service provider* and *requester* for the specification of the components of a SOA and *send, receive,* and *compensate* for the specification of orchestration of services.

The application is modelled using the CASE tool IBM Rational Software Modeler v7 [7].

The specification starts providing a specification of the architecture of the automotive case study in Section 2. Section 3 specifies the functional requirements of a system capable to manage on road assistance for vehicles requiring a garage, a tow truck and rent-a-car services. Use cases and a detailed orchestration of services are presented in this section. The last section contains models describing the structural and behavioural aspects of those components of the automotive case study, which are relevant for the *On Road Assistance* scenario. Each component is first modelled separately and the last diagram shows the interplay of the components in this *On Road Assistance* scenario. The report illustrates how the scenarios of the automotive case study that are textually described in deliverable D8.0 [1] and further detailed in the Case Study Scenario Description report [2], can semi-formally be modelled using UML and the SENSORIA UML Profile.

2 Automotive Service-Oriented Architecture

This section presents the main components of the SENSORIA Automotive System. Figure 1 depicts an overview of the Service-Oriented Architecture (SOA).

The architecture is represented as an UML deployment diagram, which shows the distribution of the components within the different physical devices of the vehicle, and the related components belonging to the vehicle environment. In fact, the architecture distinguishes three domains: *vehicle, vicinity-vehicles*, and *environment-infrastructure*. Each domain is briefly introduced; a detailed specification of the components of each domain is given in section 4.

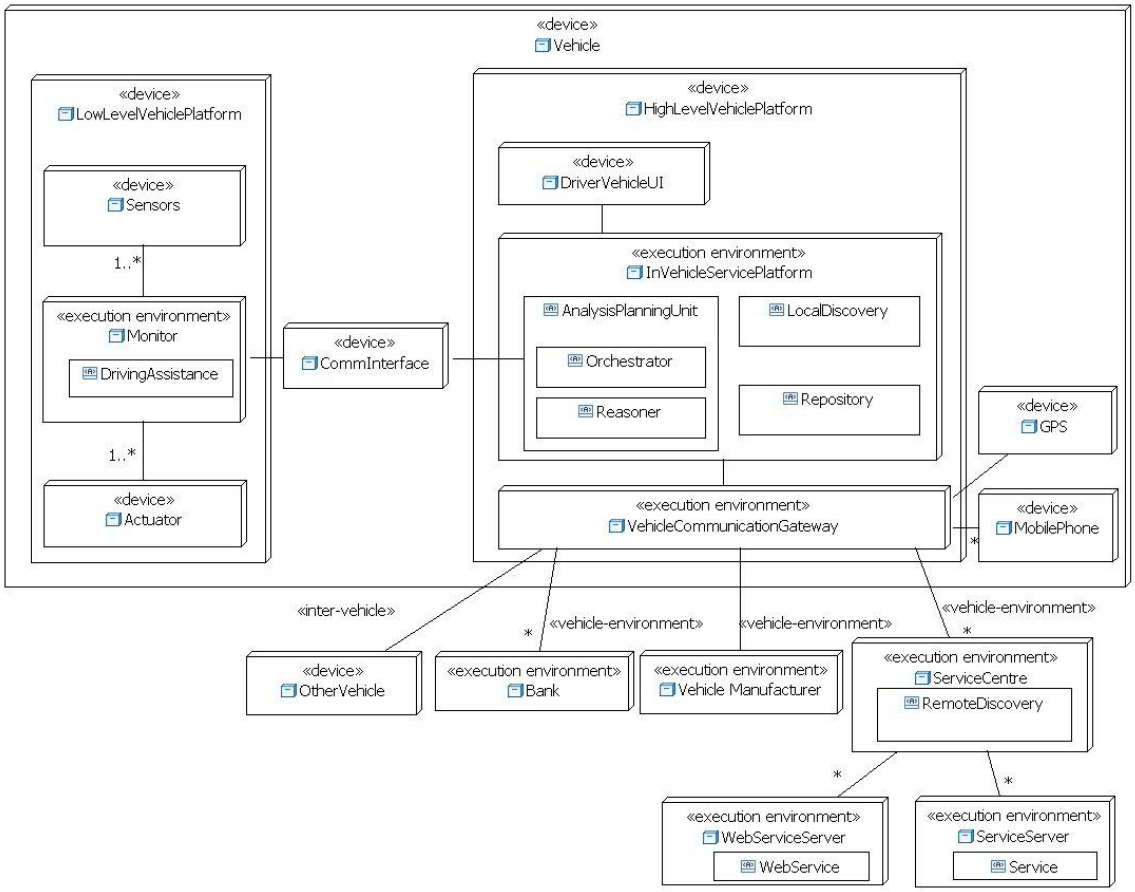

Figure 1: Automotive SOA

The architectural elements the UML provided to build a deployment diagram are nodes and artifacts connected by UML associations and dependencies among others. Artifacts are used to specify the physical pieces of information that are used or produced by a software development process or by deployment and operation of a system. Nodes are computational resources upon which artifacts may be deployed for execution. We use to special types of nodes: device and execution environment. For these nodes the UML provides corresponding stereotypes «device» and «execution environment». A device is a physical computational resource with processing capability upon which artifacts may be deployed for execution. Devices may be complex (i.e., they may consist of other devices). An execution environment is a node that offers an execution environment for specific types of components that are deployed on it in the form of executable artifacts.

The *vehicle domain* comprises all devices placed within the car, which are relevant for the specification of the automotive scenarios. It includes hardware and software integrated in the car. It may also be portable devices, such as mobile phone, PDAs or laptops. We distinguish between a low-level and high-level vehicle platform. For an overview of nodes and artefacts of these platforms, see Table 1.

The *vicinity-vehicle domain* includes other vehicles, which are not further specified, but need to be modelled in order to represent graphically the inter-vehicle communication. The architecture of other vehicles contains similar elements to those described in the vehicle domain.

The *environment-infrastructure* includes only those modules of the environment, which are potential providers of services for the car or the driver. For example, in our case study the vehicle manufacturer, a service centre and a bank. The service centre provides services like service discovery and on road assistance consisting of car repair, tow truck and car rental.

The SENSORIA Automotive System distinguishes three types of communication: communication between modules within the vehicle, vehicle-to-vehicle communication and communication to other entities in the environment. Using the UML extension mechanisms, three stereotypes are defined to ease the specification of the different types of communication: *«intra-vehicle»*, *«inter-vehicle»* and *«vehicle-environment»*. In Figure 1 we specify the internal communication in the car as default communication type, i.e the name of the stereotype «intra-vehicle» is not explicit depicted in the diagram.

Our automotive service-oriented architecture consists of the following elements:

Node	Description
Vehicle	Represents the physical entity that is meant to carry or transport something. In the context of SENSORIA a vehicle contains sensors and is able to determine its geographical position and to interact with the external world. It is modelled as a device node.
Hardware/Low Level Vehicle Platform	Acts as a container for both the hardware and the low-level software components of the vehicle. Critical driving assistance services as ABS or ESP and their related sensor software systems are deployed to this low architectural level to ensure minimal response time.
Sensor	Contains all relevant sensors necessary to observe the vehicle's status. Each sensor has to provide information to the *Monitor*.
Monitor	Manages and monitors all vehicle sensors, alerts in case of a sensor indicating an event. Very critical situations (e.g. wheel spin) detected are reported to the *Actuator* software while other sensor indications (e.g. low fuel or oil level) are submitted to the node *Analysis and Planning Unit*.
Driving assistance	Contains all low-level driving assistance implemented functionalities (e.g. ABS), which are triggered by the *Actuators* and deployed on the *Monitor*.
Actuator	Triggers fully-automatically the on-vehicle (low-level) driving assistance systems like ABS, anti slipping assistance and stability assistance. Note that no complex diagnostics and planning have to be performed by the *Actuator*.
Communication Interface	Enables communication between low- and high-level platforms of the vehicle.
High Level Vehicle Platform	A computing platform to perform the higher computational functionalities of the vehicle.
Driver/Vehicle UI	Communication interface between driver and vehicle. The driver receives information from the active services and can enter commands to trigger, stop or customise them.
In Vehicle Service Platform	Architectural element containing a service repository and software to manage service specific functionalities like service discovery and service orchestration.
Analysis and Planning Unit	Analyses events reported by the *Monitor* or *Driver/Vehicle UI* or the *Vehicle Communication Gateway* and plan the corresponding actions while typically human interaction and/or complex communication patterns are involved. The *Internal Service Repository* is asked for an appropriate service for specific actions. An example is the lookup of an onboard diagnostic service when the *Monitor* reports abnormal situations. It must provide mechanisms for lookup (of the service repository), discovery and registration of services.
Orchestrator	Architectural element that is in charge to achieve a goal by mean of composition of services.
Reasoner	Analyses and selects services based on established criteria.
Local Discovery	Local service for finding appropriate services.
Repository	Internal repository of services, where in-car services or software needed for consuming external services are stored locally.
Vehicle Communication Gateway	Abstracts sending of messages to external components (e.g. another vehicle or a server) from the underlying communication technology and protocol. The best suitable communication technique (e.g. UMTS, GPRS, WLAN) is selected dynamically and transparently to the sender of the message.
Mobile Phone	Device used to build up GPRS or UMTS connections. The mobile phone can be

	built-in the vehicle or provided by the driver.
GPS	Receiver for Global Positioning System information.
Bank	Commercial or state institution that provides financial services.
Vehicle Manufacturer	Manufacturer of the vehicle. In the SENSORIA context, it also provides services, such as remote diagnostic and car repair.
Service Centre	Remote execution platform offering discovery of services and applications supporting the service requests.
Remote Discovery	Remote service for finding appropriate services.
Service Provider	Remote execution platform offering services.
Web Service Provider	Remote execution platform containing services available on the Web.
Service	An abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities (SENSORIA Glossary).
Web Service	Service that is available on the Web.

Table 1: Elements of SENSORIA automotive architecture

3 On Road Assistance Scenario: Functional Requirements

In the *On Road Assistance* scenario, the diagnostic system reports a severe failure in the car engine, for example, the vehicle's oil lamp reports a low oil level. This triggers the in-vehicle diagnostic system to perform an analysis of the sensor values. The diagnostic system reports for example a problem with the pressure in one cylinder head, and therefore the car is no longer driveable, and sends a message with the diagnostic data as well as the vehicle's GPS data to the car manufacturer or service centre. Based on availability and the driver's preferences, the service discovery system identifies and selects the appropriate services in the area: repair shop (garage), tow truck and rental car. When the driver makes an appointment with the garage; the diagnostic data is automatically transferred to the garage, which could then be able to identify the spare parts needed to perform the repair.

The service discovery system identifies as well a towing service, providing the GPS data of the stranded vehicle. The driver makes an appointment with the towing service, and the vehicle is towed to the shop. The selection of services takes into account personalised policies and preferences of the driver. We assume that the owner of the car has to deposit a security payment before being able to order services.

Section 3.1 shows the specification of the functional requirements of a system capable to manage on road assistance for vehicles requiring a bank, a GPS, a garage, tow trucking and rent-a-car services. Section 3.2 presents the orchestration of services.

3.1 Use Case Model

Functional requirements are represented as uses cases. Figure 2 shows the UML 2.0 use cases diagram.

The Driver, the Bank, a GPS, a Service Centre, and On Road Assistance are modelled as actors. To keep the models simple, we assume that this assistance provides the three on road services needed in the case the car cannot be driven, i.e. car repair, tow trucking and car renting. In such a situation, the driver will request the repair of the vehicle, which includes access to the GPS data, discovery of appropriate services, ordering of services (car repair, tow truck and car rental). Cancellation of these orders is also possible, which imply the corresponding compensations activities. Two types of discovery are considered: discovery of services in the local repository and a general remote discovery.

Services are modelled using the stereotype «serviceUseCase» of the SENSORIA UML profile.

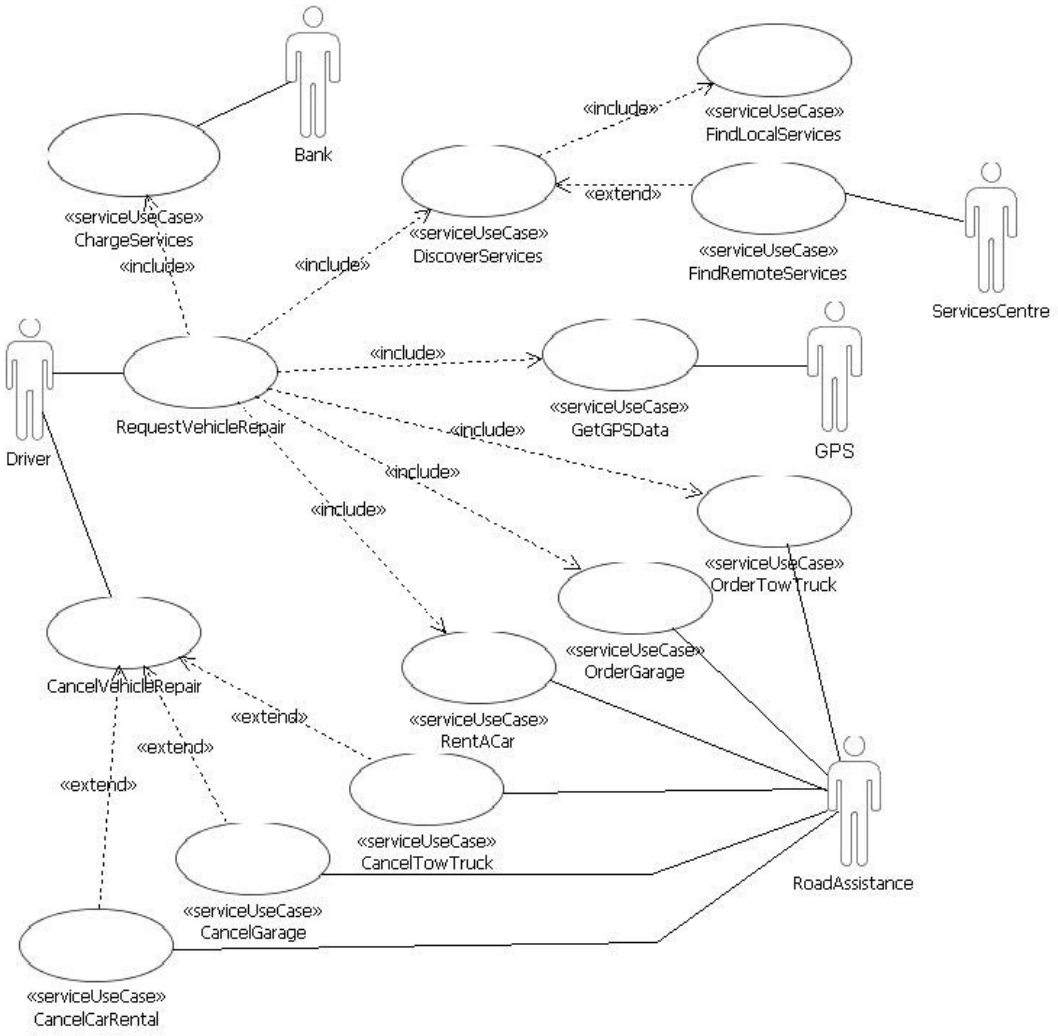

Figure 2: Use case model for the *On Road Assistance* scenario

3.2 Orchestration of Services

One of the most distinguishing aspects when modelling the behaviour of a service-oriented system is the workflow describing the orchestration of services. In the modelled business process of the *On Road Assistance* scenario the orchestration is triggered by an engine failure or a sensor signal like low oil level. Figure 3 shows an UML 2.0 activity diagram for the orchestration of services in the *On Road Assistance* scenario. We use stereotyped UML 2.0 actions indicating the type of interactions: «send», «receive» and a «sendAndReceive». In addition, stereotypes are used to model compensation of long running transactions: «compensate» and «compensationEdge». Actions match operations of required and provided interfaces of the services, which are defined as ports of UML 2.0 components. For the SENSORIA UML 2.0 profile the reader is referred to D1.4a [3].

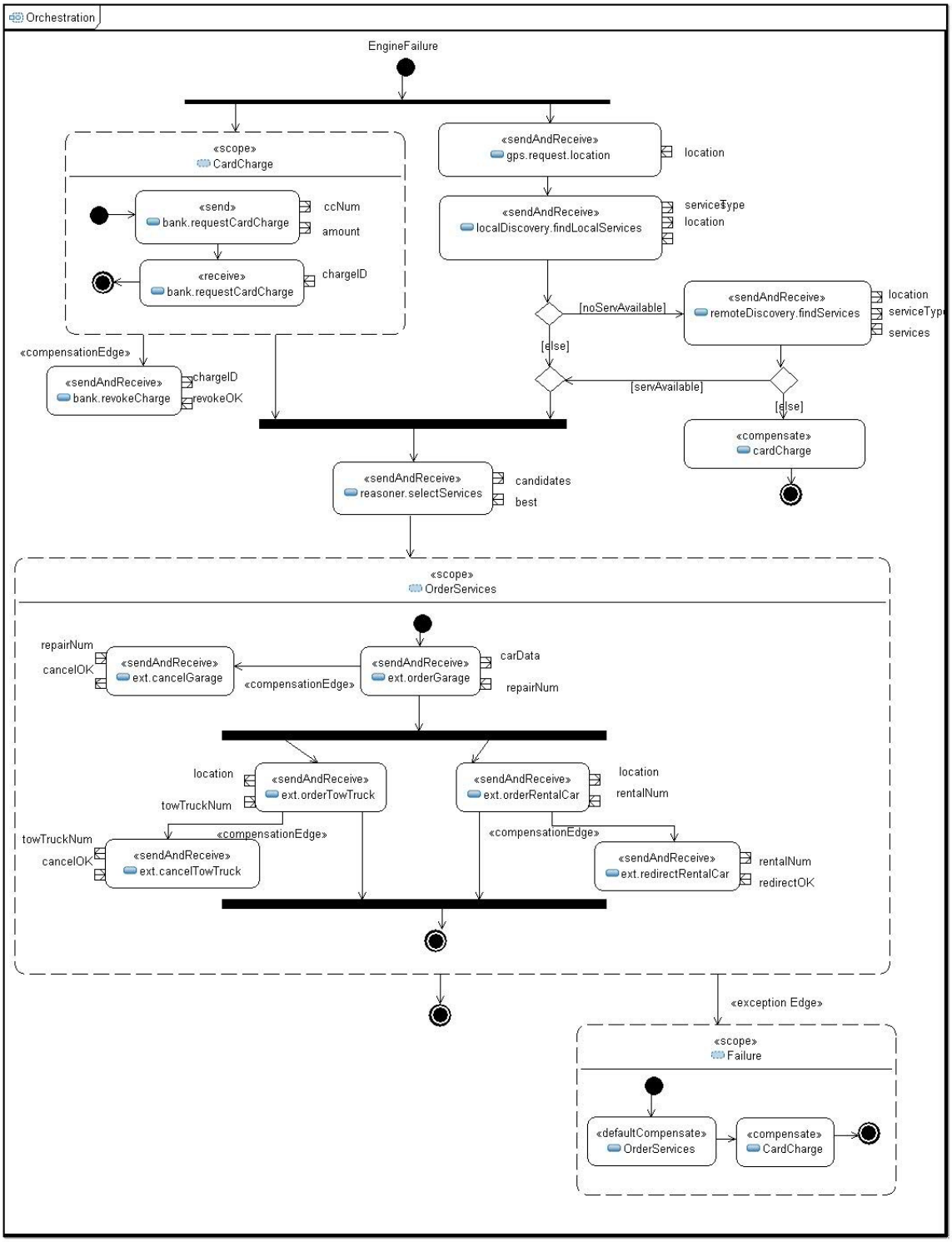

Figure 3: Orchestration of services

The process starts with a request from the *Orchestrator* to the *Bank* to charge the driver's credit card with the security deposit payment, which is modelled by an asynchronous UML action *RequestCardCharge*. To charge the credit card the card number is provided as an input parameter of the UML stereotyped call action. In

general, to determine input and output parameters, the following assignment is used: Input pins have an arrow pointing towards to the action; output pins have an arrow pointing away from the action.

In parallel to the interaction with the bank, the orchestrator initiates a synchronous interaction to get the current position of the car from the *GPS* service. The current location is modelled as input of the send and receive action and subsequently used by the *FindServices* action which retrieves a list of services. In case no local services are available, an action *FindServices* on the *RemoteDiscovery* is started. If services cannot be found an action to compensate the credit card charge will be launched.

For the selection of services, the *Orchestrator* synchronises with the *Reasoner* to obtain the most appropriate (best) services. Service ordering is modelled by the UML actions *OrderGarage, OrderTowTruck* and *RentalCar* following a parallel and sequential process, respectively.

The other focus when modelling services lies on the specification of an appropriate transactional business process. Such a business process contains both forward actions and compensations. As UML 2.0 does not provide explicit elements for the modelling of such compensations, we defined a set of modelling primitives and corresponding stereotypes for UML 2.0 activity diagrams with a Saga-like semantics. The extension comprises a compensable area named *Scope,* a specific *Composition* edge for the control flow and a specific *Compensate* action. For the new element types corresponding stereotypes *«scope», «compositionEdge»* and *«compensate»* are defined. With these extensions, the orchestration for the car repair scenario can be compactly formulated (see Figure 3). In the modelled business process, the driver's credit card is charged with the security deposit payment, which will be revoked if ordering the services failed. A garage appointment is searched first and a tow truck is ordered in parallel to renting a car. The appointment with the garage will give coordinates to tow the broken down car to, and a location constraint that restricts the car rental agency that may be ordered. If ordering the car rental fails, the overall process does not fail, as the activity is enclosed in a sub-transaction. However, if ordering a tow truck fails the garage appointment has to be cancelled as well. For this reason, the *Orchestrator* will try to order a tow truck service until either no more service offers are found or the ordering succeeds. If ordering a tow truck fails the rental car delivery will be redirected to the driver's actual location.

4 On Road Assistance Scenario: Components

This section describes the automotive case study components that play a relevant role in the *On Road Assistance* scenario. The component itself, the ports, the interfaces and the classes, which implement the interfaces, specify the structure of a component. Therefore, components, such as Sensor and Mobile Phone are not modelled in detail. In addition, only main operations of the interfaces are specified. A simple UML 2.0 state machine that specifies the possible set of states and the transitions between the states models the behaviour of each component.

4.1 Bank

The bank represents an institution that provides financial services. The bank operations that are relevant for the *On Road Assistance* application are the charge of a credit card and the revoke of a charge. Figure 4 depicts the Bank component, the *«service»* port *cardTransaction* and the class *CreditCard,* which implements a provided *CardCharge* interface. The state machine describing the behaviour of the component *Bank* is shown in Figure 5.

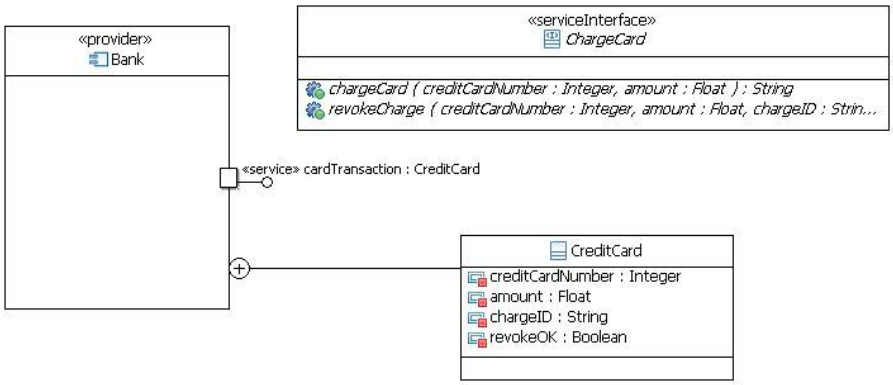

Figure 4: Structure of component Bank

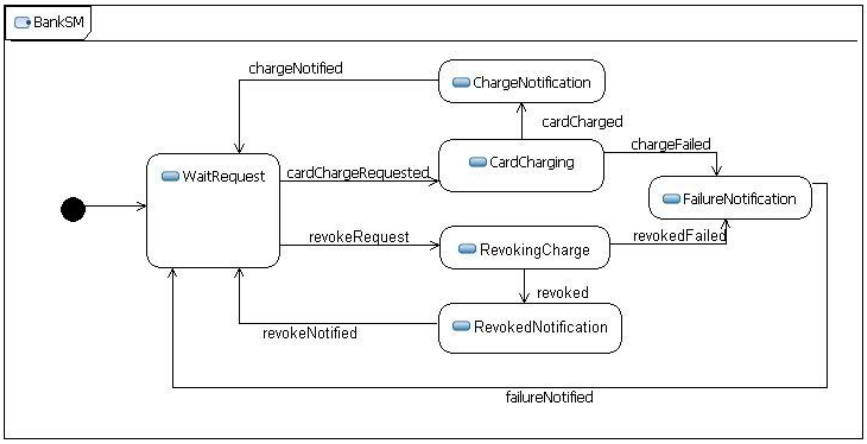

Figure 5: Behaviour of component Bank

4.2 GPS

The *GPS* is provider of data from the Global Positioning System. The service relevant for the *On Road Assistance* application is the request of position, i.e. the current position of the car. Figure 6 depicts the GPS component, the *«service»* port *GPSData* and the class *Location,* which implements the provided *Location* interface. The state machine describing the behaviour of the component *GPS* is shown in Figure 7.

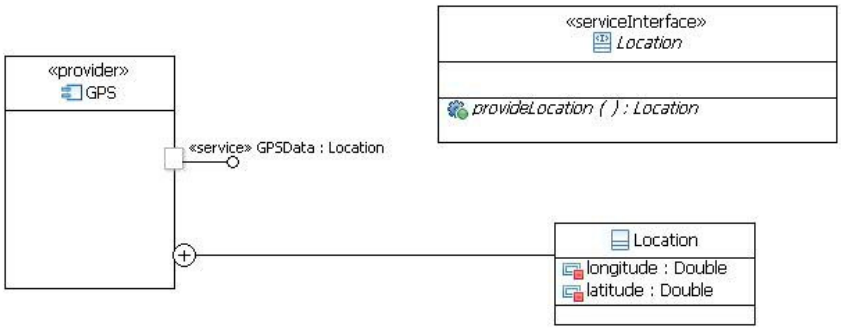

Figure 6: GPS structure

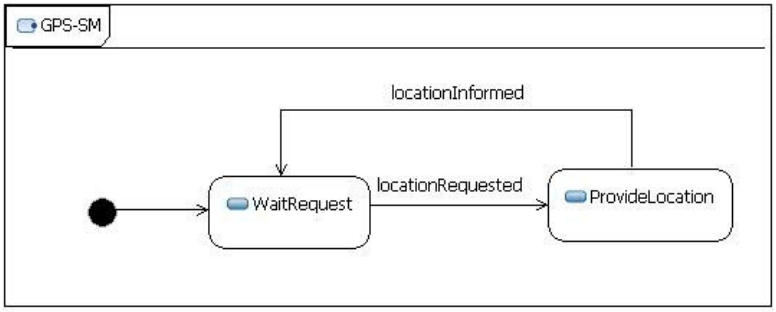

Figure 7: Behaviour of component GPS

4.3 Local Discovery

The *Local Discovery* looks for appropriate services in the local repository. The *Local Discovery* component is shown in Figure 8 together with the interface *LocalService* depicted as interface and the class that implements the service. The state machine describing the behaviour of the *LocalDiscovery* component is shown in Figure 9.

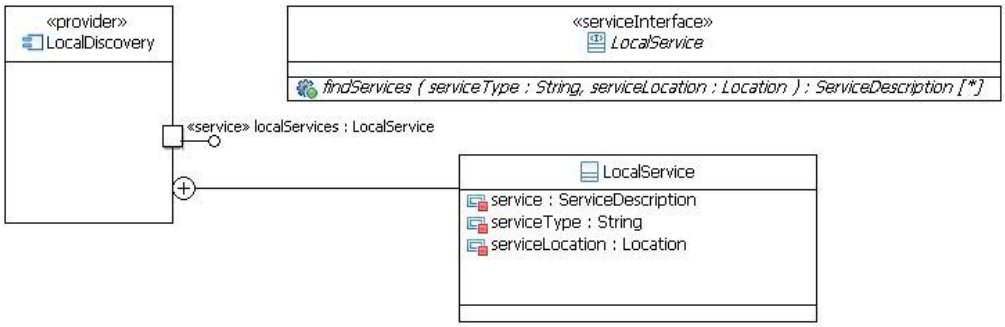

Figure 8: LocalDiscovery structure

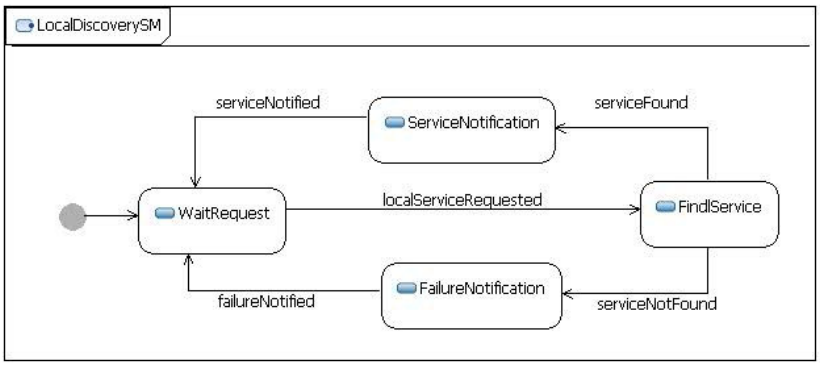

Figure 9: Behaviour of component LocalDiscovery

4.4 Orchestrator

The *Orchestrator* is an architectural element that is in charge to achieve a goal by means of composition of services. Figure 10 shows all components that the orchestrator needs to communicate with in order to orchestrate services in the *On Road Assistance* scenario. These components are on the one hand the Reasoner and the Local Services, which are needed to discover services (*findLocalServices*) and select services (*bestServices*) respectively. On the other hand, dynamic binding will be performed with external service providers such as *Garage*, *a TowTruck* and a *RentACar,* which have been selected in the previous step according to the results of the discovery and reasoning process. For a more simple graphical representation, these services are accessible through the *Vehicle Communication Gateway*. A simplified state machine describing the behaviour of the component *Orchestrator* is shown in Figure 11.

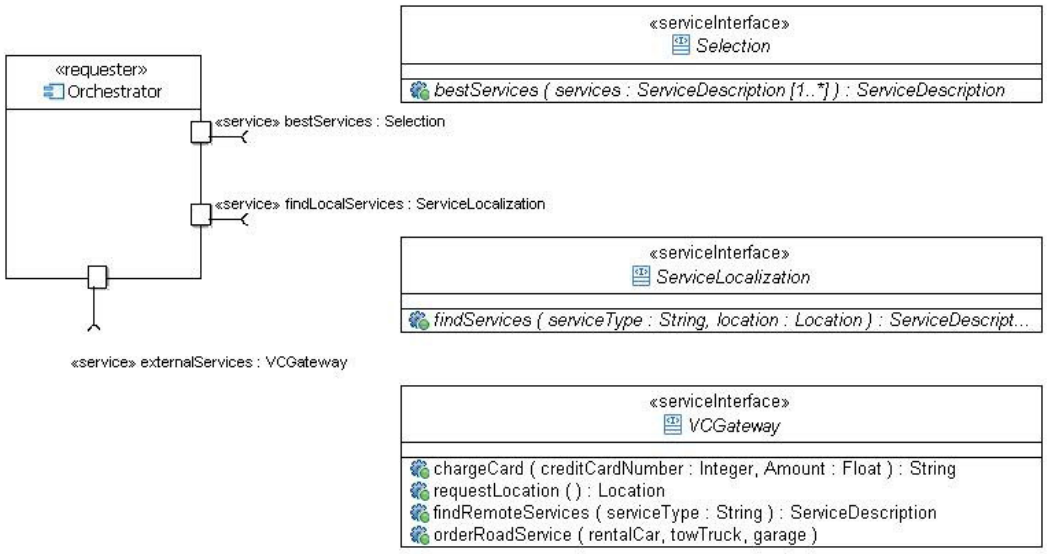

Figure 10: Orchestrator structure

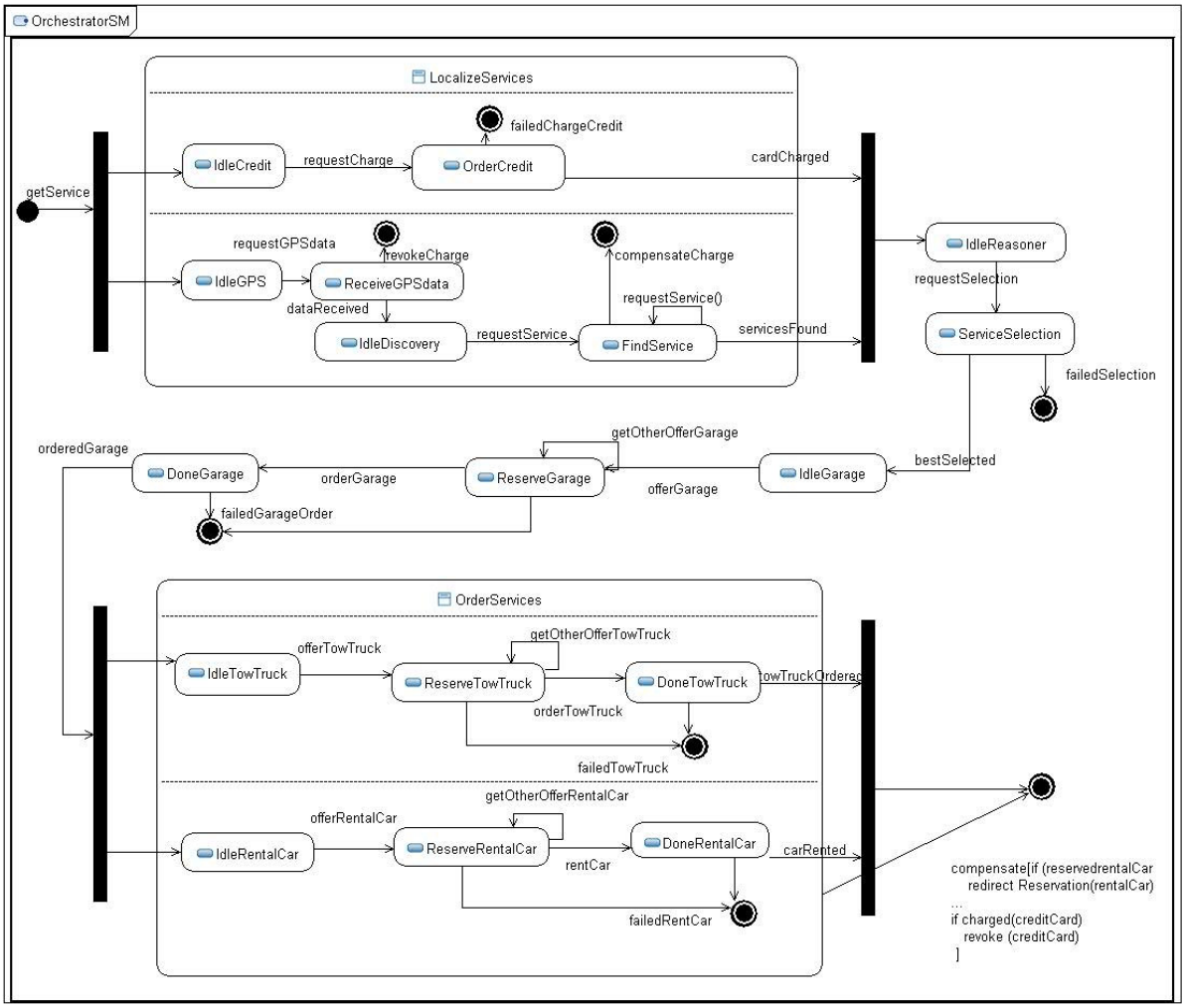

Figure 11: Behaviour of component Orchestrator

4.5 Reasoner

The *Reasoner* service provider analyses and selects services based on established criteria. The *Reasoner* component is shown in Figure 12 together with the service port *selectServices*, *ServiceSelection* depicted as service interface and the class that implements the service selection of best services. The state machine describing the behaviour of the *Reasoner* component is shown in Figure 13.

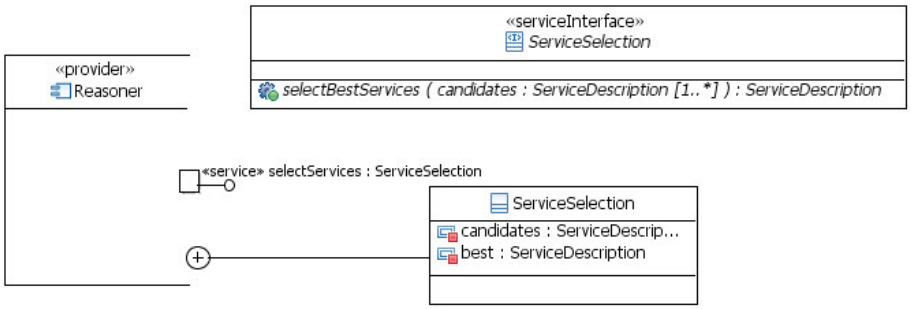

Figure 12: Reasoner structure

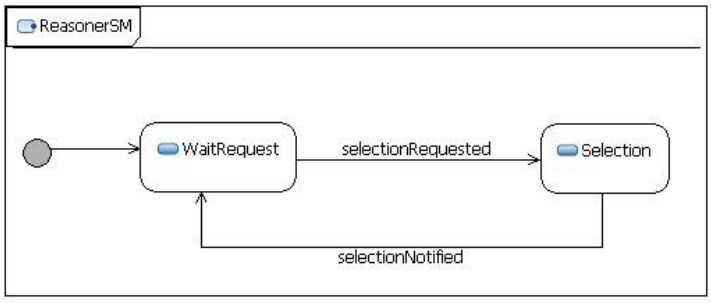

Figure 13: Behaviour of component Reasoner

4.6 Remote Discovery

The *Remote Discovery* searches for appropriate services in the remote repository. The *Remote Discovery* component is shown in Figure 14 together with the service *remoteDiscovery* modelled as port, the a provided service interface *ServiceDiscover* and the class that implements the service. The state machine describing the behaviour of the *RemoteDiscovery* component is shown in Figure 15.

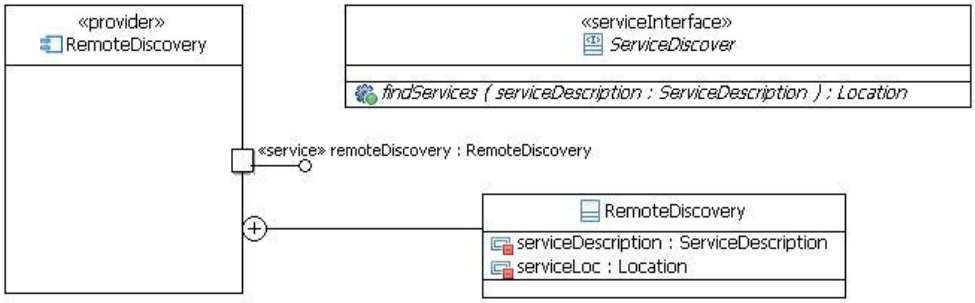

Figure 14: RemoteDiscovery structure

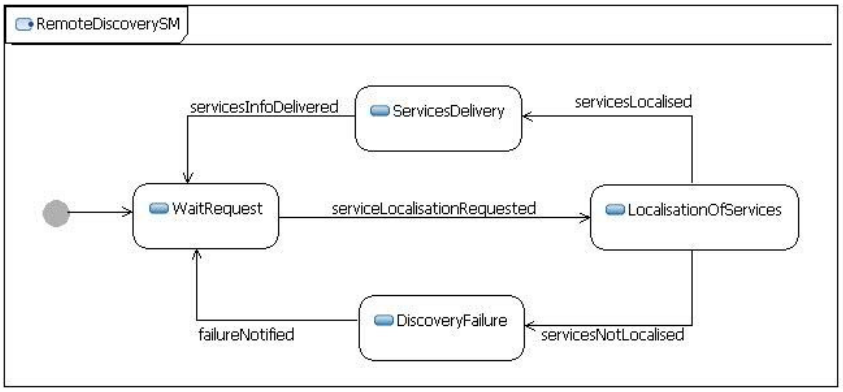

Figure 15: Behaviour of component RemoteDiscovery

4.7 Road Assistance

The *Road Assistance* service provides all required services for car repairing and for arriving on time on the scheduled appointments. The structure of the *Road Assistance* component is shown in Figure 16. The state machine describing the behaviour of the *RoadAssistance* component is shown in Figure 17.

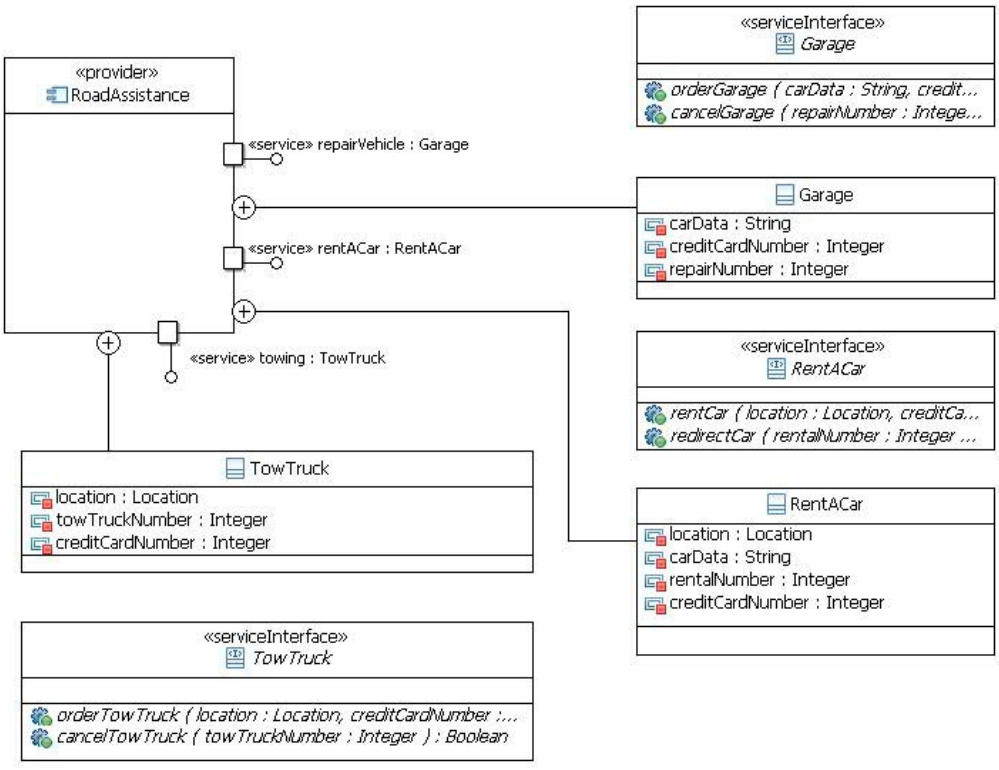

Figure 16: OnRoadAssistance structure

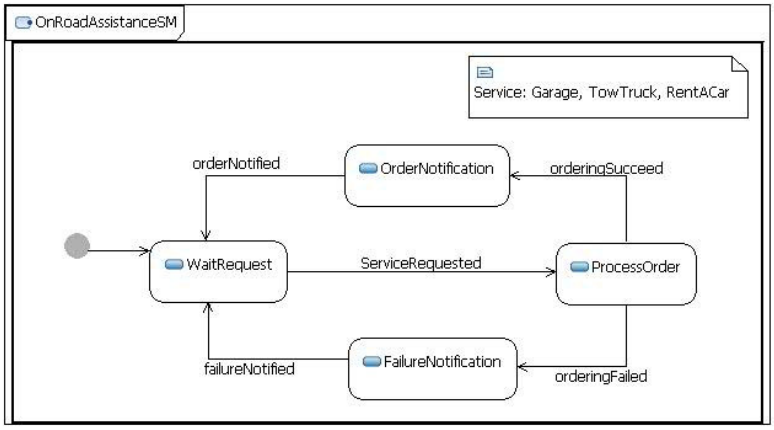

Figure 17: Behaviour of component OnRoadAssistance

4.8 Vehicle Communication Gateway

The *Vehicle Communication Gateway* models sending of messages to external components, such as other vehicles or a server, in a high level of abstraction without any details of the underlying communication technology and protocol. Figure 18 shows all the Vehicle Communication Gateway component and all components, which will be bind in case messages need to be transmitted from the car to such components. These components are the *Bank*, a *RoadAssistance* that will be in charge of finding appropriate services, the selected *TowTruck, Garage* and *RentACar* service providers. The corresponding interfaces are shown as well. A simplified state machine describing the behaviour of the component *Vehicle Communication Gateway* is shown in Figure 19.

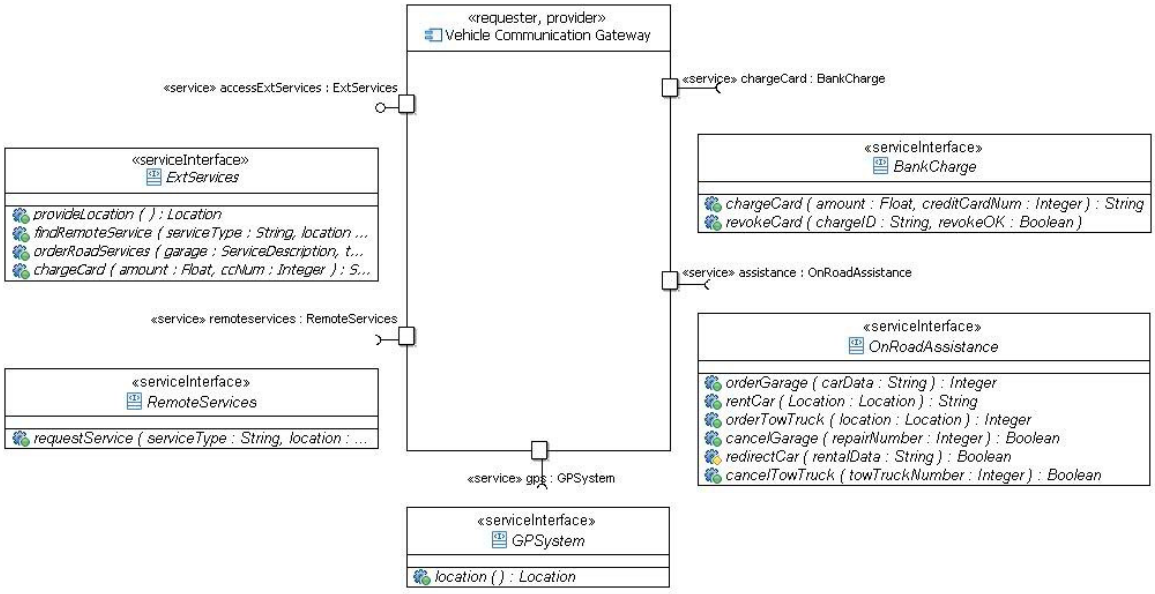

Figure 18: Vehicle Communication Gateway structure

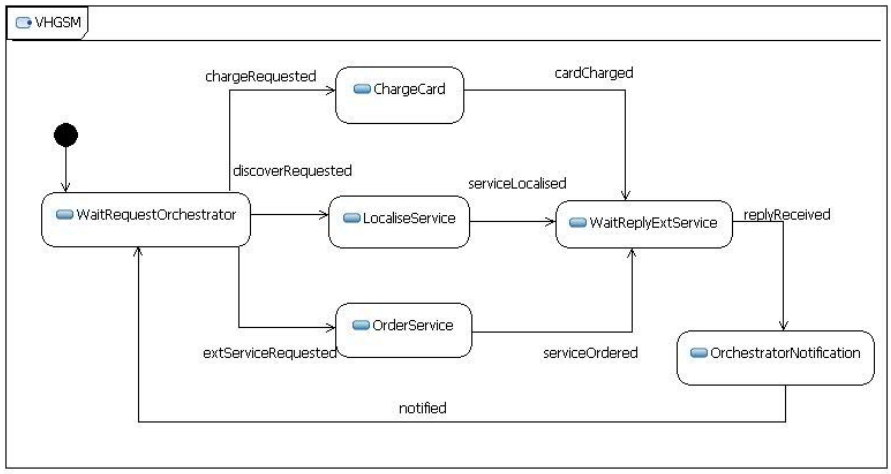

Figure 19: Behaviour of component Vehicle Communication Gateway

4.9 Components Interplay

Communication between components of the *On Road Assistance* Scenario is shown in Figure 20. A UML structure diagram is used to model components, ports, provided and required interfaces.

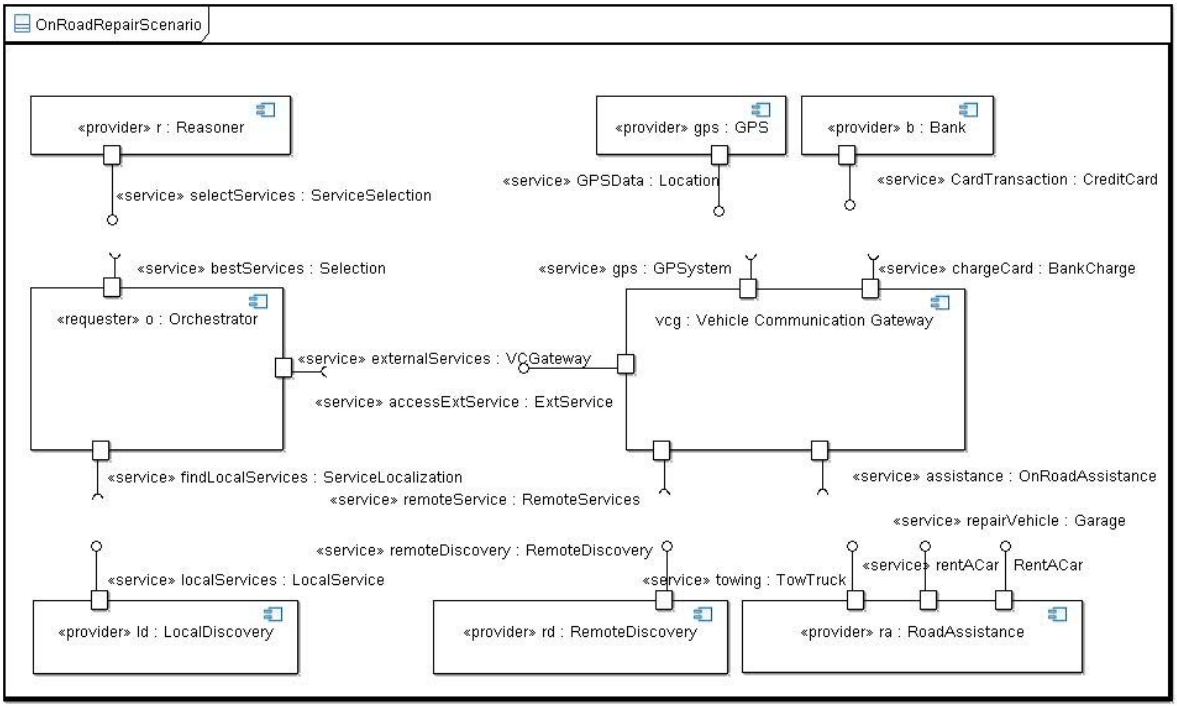

Figure 20: Component diagram of *On Road Assistance* Scenario

5 Outlook

The specification of the *On Road Assistance* scenario presented in this report could be further refined including for example features to model policies and non-functional properties, such as security aspects. Current research in the automotive telematics area will also be used to identify critical issues in the engineering of such services.

Based on the input of SENSORIA partners, the specification of the scenarios will be refined with details required for application of specific techniques developed within the scope of the project, such as model-checking or verification of service behaviour, e.g. with Hugo/RT [4], UMC [5] or PEPA [6]. Other scenarios of the automotive case study, such as Accident scenario, Parking Assistance and Route Planning can be specified similarly to the *On Road Assistance* scenario present in this report.

References

1. SENSORIA D8.0 Case Studies Scenario Description, October 2006.
2. SENSORIA Report Scenarios Description of the Automotive Case Study, September 2006.
3. SENSORIA D1.4a, UML for Service-Oriented Systems, to be published.
4. Hugo/RT, http://www.pst.ifi.lmu.de/projekte/hugo/, last visit August 6th, 2007.
5. UMC, http://fmt.isti.cnr.it/umc/, last visit August 20th, 2007.
6. PEPA, http://www.dcs.ed.ac.uk/pepa/, last visit August 6th, 2007.
7. IBM Rational Modeler, V7, http://www.ibm.com/developerworks/rational/products/rsm/, last visit August 6th, 2007.