Sensoria

016004
Software Engineering for Service-Oriented Overlay Computers

D8.0: Case studies scenario description

Lead contractor for deliverable: ISTI-CNR
Author(s): Stefania Gnesi, Maurice ter Beek (ISTI-CNR), Hubert Baumeister (DTU), Matthias Hoelzl (LMU), Corrado Moiso (TILab), Nora Koch (LMU and FAST), Angelika Zobel (FAST) and Michel Alessandrini (S&N)

Due date of deliverable: August 31, 2006
Actual submission date: August 31, 2006
Revision: Final
Dissemination level: 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

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

Contents

1 Executive summary

The purpose of this work package is to provide a context of realistic case studies for developing intuitions that can feed and steer the research process according to the expectations of society and its economy, discussing and communicating ideas among partners and finally communicating research results to and getting feedback from the research community at large, both in industry and academia.

Having in mind the relevance that these areas have in society and the economy, four case studies will be chosen in WP8. Three are coming from industrial applications: telecommunications, automotive and finance plus a common academic case study on distributed e-learning and course management, which we could also use for dissemination and training activities.
The main objective of this work package is to test and verify the applicability of the insights, methodologies and tools developed in the SENSORIA Project WPs more focused on research activities, on the chosen case studies.

During the first period of the project, scenario descriptions and requirements specifications have been provided for all the case studies.

2 Introduction

One main element of SENSORIA are realistic case studies for different important application areas including e-business, telecommunications and automotive to provide a context for developing intuitions that can feed and challenge the research process according to the expectations of society and its economy, discussing and communicating ideas among partners-and finally communicating research results to and getting feedback from the research community at large, both in industry and academia.
Most of the case studies will be defined by the industrial SENSORIA partners as to provide continuous practical challenges for the new techniques of Services Engineering and to serve for demonstrating the research results.
In the following sections we give a scenario description for each of the case study presenting their main characteristics. We moreover provide a number of implicit and explicit links to the Sensoria Ontology (deliverable D1.1.a) [16]
The case studies have been divided into two different classes: demonstrator and experimental case studies.
The demonstrator case studies will be used as demonstrations of the results provided by the other WP.
Experimental case studies will be instead more closely related to fundamental research carried out in SENSORIA.
The development activities related to the case studies will be carried out in parallel with the foundational activities of other WPs. In this way, the case studies can drive and provide grounding to the results produced in those WPs, and give some early feedback to be considered in the refinement of the methodologies and tools developed in the project.
More details on the scenarios description can be found in Sensoria reports [12, 13, 14, 15] added as appendix this deliverable.

3 T8.1 Demonstrator Case Studies: Telecommunications

The telecommunication case study scenario is focusing on the development of applications combining two global computing infrastructures, namely the Internet and Next Generation Telecommunication Networks. In such a context, the applications would be designed, deployed, and executed within an SOA framework that integrates Web Services based computing and a rich set of telecommunication services

and capabilities, including call/session control, messaging features, presence and location features, etc. In particular, issues concerning the composition mechanisms, including semantic and dynamic composition, and asynchronous/synchronous orchestration, etc, to define/create and execute telecommunication services are addressed. We refer to the Sensoria Ontology (deliverable D1.1.a) [12] for detailed explanations of these and other notions related to SOAs. Moreover, the issue concerning the secure and controlled interaction between application components deployed in different domains (e.g., an enterprise domain and a network operator domain) will be considered in the case study.

It is important to be point out that the proposed Telecommunication Case Study mainly addresses issues concerning the evolution of the service infrastructure/platform, and not specific telecommunication services. Some specific services could be used in the description only to exemplify the infrastructural issues.

3.1 Service layers and evolution trends

The Telecommunication Services, i.e., the services that are provided by a telecommunication infrastructure managed by a public network operator, are evolving by considering several aspects of ”convergence”: convergence of media, convergence of terminals, combination of service features and convergence of Telco and Internet/Web worlds.
Moreover, the services should be ”user-centric”, that is their behavior should be personalized according the requirements of single end-users. The possibility of personalizing should be uniform and cover all the features of the service. In particular, the end-users should be seen as single entities even across several different networks, terminals, communication media, and applications/services.
In general telecommunication services are created, executed and managed a ”Service Layer”, which consists of a set of systems implementing the functions needed for the service delivery. The services are defined by a ”business logic” that may use and combine a set of Telco features which provide some basic service capabilities.
Most of the current service layer is realized as a set of ”vertical” platforms, named Silos, each of them specialized to provide services involving a specific Telco Feature and a specific network. In general, such platforms integrate in a single system the service execution environments with the Telco features and some supporting functions (e.g., payment, authentication, profiles).Unfortunately in general, the vertical systems deployed in a service layer are loosely integrated.

- it is quite difficult to share a function across different platforms (i.e., to allow that a service deployed on a platform A can access a function implemented on a platform B);
- the same functions are duplicated on several platforms.

Such an organization of a service layer introduces several problems in dealing with the realization of ”converged” services. In order to improve such situation, the service layer is evolving towards a ”horizontal” approach based on:

- the integration among systems for service delivery which are deployed in the Operator infrastructure;
- sharing of and interoperability among functions, enablers and service capabilities.

The Service Layer may be seen as a complex distributed system which must fulfill requirements coming from the telecommunication domain, such as the capability to interact with the network control functions and service enablers, to provide suitable computational throughput, and support high-availability and fault-tolerance. In any case it must be able to interact with the Internet/IT domain in order to provide Telco-IT convergent services.
A possible trend for the evolution of the service layer according to a horizontal approach is the adoption of Web Services and Service Oriented Architecture solutions. At the moment, there are already

several initiatives that consider this possibility, such as the specification of Web Services for controlling Telecommunication services (e.g., Parlay X Web Services jointly specified by ETSI and 3GPP, or the Mobile Web Services under specification by the OMA initiative, Identity Web services specified by Liberty Alliance). Some of the areas where the Web Services/SOA approach can be adopted for the evolution of the Service Layer are:

- adoption of SOA-based principles and techniques (e.g., based on Web Services) to organize the internal structure of the Service Layer, and to define the communication bus among the different services and macro-functions;
- evolution of composition mechanisms for creating and executing Telco Services by adopting solutions based on orchestration/choreography and by introducing the possibility to handle semantic and dynamic compositions of services (see also *View 1: Service Description for Coordination and Composition* of the Sensoria Ontology [16]);
- adoption of Web Services technologies to expose on the "Internet" Telco Capabilties/Features to 3rd party applications (e.g.. Service/Content Providers, Enterprises), and to assemble services provided by 3rd parties (e.g., to interact with payment services, to access contents);
- adoption of Web Services technologies to introduce a uniform interact model among services delivered by the Service Layer and terminal applications.

In order to follow this trend several issues must be addressed and several open points must be solved, in order to evaluate the suitability of the SOA approach to the requirements for the delivery of telecommunication services and to investigate possible improvements and extensions.

3.2 Telco Case Study Issues

The aspects that the TELCO Case Study can cover are the following:

- Policies for Web Services
- "Identified" Web Services
- Service Session capabilities
- Transactional composition of Telco Web Services
- Asynchronous Web Services interactions
- Synchronous/Asynchronous Service Composition
- Semantic Service Description

In the following we report the main characteristics of the issue. All the details can be found in [15].

3.2.1 Policies for Web Services

A recent trend in Telecommunication is to adopt Web Services technologies to expose Capabilities implemented in a Telecommunication network (e.g., call control, sending/receiving messages, access user information such as presence and location) to applications deployed in 3rd party administrative domains (e.g., Enterprises, Service Providers, and, eventually, End-Users). Standards specified some Telco Web Services (e.g., 3GPP/ETSI Parlay X Web Services [1], OMA Mobile Web Services, Liberty Alliance). In such a context, Network Operators and 3rd parties have to define SLA (Service Level Agreement) on the usage of such Web Services: conditions to control the access and the usage of the Web Services, e.g., in order to monitor the usage of the telecommunication capabilities, to charge the usage, etc. [3, 4, 5].

A 3rd party application can use a Service Component (e.g., a Web Service) after a subscription phase, that establishes conditions and configurations that can be specified through a policy. In order to define Policies for Web Services and to allow a negotiation phase between the requests of the 3rd parties and the offerings of the operator, it is necessary to define a language to express them. The WS- Policy and Web Service Policy Language (OASIS-WSPL) seem to be two possible approaches ([3, 2]).
In the context of policies, the main activities that are to develop are:

- identify policies suitable for the Parlay X Web Service context;
- specify an advanced formalism (language) to define and (dynamically) negotiate and verify policies.

We refer to deliverable D5.1.a [10] and *View 4: Business Goals and Policies* of the Sensoria Ontology [16] for a detailed glossary of terminology related to Policies.

3.2.2 "Identified" Web Services

The telecommunication network operator used by an end-user (respectively, a client application) to access a Web application (respectively, a Web Server) is in a privileged position in order to certify the identity of the client, and to provide to the service providers the identities of the end-user (respectively, the client application) accessing to their Web applications (respectively, a Web Servers). In fact, the operator may exploit the authentication/identity information at network level and by translating them, transparently to the end-users, to the application level. By adopting this solution, the end-users/applications can access services without explicitly providing credentials and without the need to have security modules. Moreover, the service providers do not need to deploy functions to deal with direct authentication of end-users. In such a scenario, the telecommunication network operator plays the role of "Identity Provider" for both its internal services and external service providers. In the proposed scenarios some open issues concern the formal evaluation of the protocol implementing the interactions among the involved entities, i.e.: Web Service clients on the terminals, Identity Provider functions, and Web Services.
In particular, the following issues should be considered:

- validation of the security properties of the interactions, and identification of possible threats;
- characteristics of the identity tokens.

The formal analysis could originate an improvement of the protocols and of the interaction flows among the involved actors. Moreover, it could be interesting investigate how to generalize the approach of adding identity credentials to service interactions in a SOC environment.

In this context we also refer to deliverables D3.1.a [8] and D.3.3.a [9].

3.2.3 Service Session capabilities

This issue enhances the one concerning the policies for Telco Web Services. In fact, it deals on the constraints in combining several Telco Web Services to define the business logic of an application.
In particular, even if an application is allowed to use some Telco Web Service, e.g., a User Location Web Service, it may happen that it can not use it to require location of end-users (at any time and for any end-user), but the usage could be limited in the context of a service session.
A service session could start when an end-user require the service, or subscribe it.

A session may be associated to some capabilities; examples of capabilities are:

- access rights (i.e., in read, write, etc. mode) to personal data, such as the possibility to access the location or charge the account of an end-user; for instance, a session is allowed to access only the location of the end-user A that activated the session or those of the end-users that authorized A to get their location (e.g., as it is member of the same community or of her/his buddy list);

- number of invocations of the Telco Web Services in a session; for example, the application can only send a limited number of SMS to an end-user in the context of a session;
- possibility to invoke the Telco Web Services in a session according to some orders; for instance, the application can invoke a Charge Telco Web Service only at the end of the session;
- duration of the session, in order to allow the application to invoke the Web Services only for a limited time;
- possibility to transfer the session ownership to another application.

Therefore, when the agreement on the usage of a Telco Web Service requires that it can be used only in the context of a service session, it is necessary to verify, at execution time, that the requests to such Telco Web Service are compliant with the capabilities associated to the service session. Moreover, it is necessary to check that the associated service session is still valid and that the invoking application is allowed to use it (e.g., it is the current session owner).
Open issues concern the investigation of:

- the formalism to express the capabilities associated to a Service Session;
- the formalism to define rules for the session lifecycle (start, termination, etc.);
- tools for generating applications compliant with the service session capabilities and/or to verify its compliance at development time;
- mechanisms to efficiently verify such capabilities at run time.

3.2.4 Transactional composition of Telco Web Services

In the definition of a telecommunication service, a lot of the effort is devoted in handling the failure cases. The same applies also in the case the service logic is defined as a composition of Web Services, e.g., by means of some orchestration formalisms.
Therefore, it could be useful to find some formalism/mechanism that may improve the handling of failure cases and exceptions. A potential solution could be the adoption of transactional mechanisms. We refer to *View 3: Non-Functional Properties of Services* of the Sensoria Ontology [16] for a detailed description of the notion of transactionality.
Due to the nature of telecommunication services, which interact with network resources, involve end-users, etc., in case of failure it is necessary to perform a ”logical” compensation of the part of the business logic whose execution caused an exception or a failure. The service definition formalisms should be enhanced in order to easily deal with the definition of the compensations. Moreover, the definition of Telco Web Services could be enhanced in order to provide ”compensation” operations.
Open issues of the usage of transactions in the definition and execution of telecommunication services defined as combination of Web Services are:

- the transactional model for handling compensations in telecommunication services, by considering requirements coming from the telecommunication domain;
- the formalism to introduce transaction definition/requirements in a composition language for telecommunication services;
- extension of the specification of Telco Web Services with the definition of the compensations for their operations.

Moreover, it could be interesting verify whether the identified transactional model can be mapped on the specifications supporting transactions for Web Services, such as:

- Business Transactions Protocol (BTP) specified by OASIS;
- Web Services Coordination and Transactions (WS-C e WS-Tx) defined by WS-*.

3.2.5 Asynchronous Web Services interactions

The telecommunication services are characterized by the need to handle asynchronous interactions among service components.Therefore, it is relevant for a SOA-based service layer to be able to deal with asynchronous interactions, in particular, such interactions must be considered in:

- the communication infrastructure supporting the interaction among the services and the functions on the service layer;
- the definition of the interfaces of Telco Web Services;
- the composition models for the definition and execution of the services.

All the already existing approaches operate at ”application level”: in fact, they define specific operations and interface to deal with asynchronous interactions. An alternative approach could be enhanced the communication protocol. Moreover, the type of the interface exposed by a service does not include the definition of the notifications that are emitted by the service. Therefore, these types do not represent the ”whole” contract between the service and the clients.
Therefore, some of the open issues to consider here are:

- investigate asynchronous protocols suitable for handling long computations and event notifications;
- analyze with formal methods properties of proposed protocols;
- extend the types of the service interfaces in order to includes the definition of the notification emitted by the service.

3.2.6 Synchronous/Asynchronous Service Composition

Telecom operator and service providers needs tools and environment to create and execute services composed of atomic and reusable service components (or building block, Web Service), described and stored in a repository. Each composed service can be also seen as a component. The execution environment must support both process oriented and event based services as Telco services are mainly event-based in order to deal with events coming from the network. Current standard based approaches are:

- BPEL and alike designed to run processes, but do not fit real-time asynchronous requirements of TELCO;
- JAINSLEE (Service Logic Execution Environment) designed to support event based asynchronous services.

Service composition can be done at several stages:

- design time (service designer graphical editor);
- run time (execution engine);
- automatic and dynamic.

Traditional approaches rely on a XML based service descriptor as a result of the composition designer, which is executed by the engine. The objective here is to study and develop or adopt a suitable language for composition of Telco components (e.g. Web Services), in order to cover also the asynchronous communication aspects.

3.2.7 Semantic Service Description

Telco operators are expecting an increasing number of services available to end-users in the near future and for this reason services need to be appropriately described to facilitate the discovery of services of interest and compose them into more complex ones. In order to automate the service design and composition process it is necessary to describe services and service components in a machine friendly way:

- to define a semantic description of components (Categories, cost) and their behavior (Input, Output, Precondition Effects);
- to develop a reasoner that is able to process semantic description of components based on specific template and compose them.

Semantic description is also necessary to allow users to discover or get notified about service of interest. In this context, Semantic Web technology has been focusing on the definition of languages for describing items, and in particular services, and how to define rules to reason on these. The main activities regarding semantic service description are:

- Define an ontology for TELCO services and service components (e.g. OWL-S based);
- Simplify creation of semantic description of services (adopting tools);
- Improve reasoning when discovering services;
- Improve reuse of composed services by storing them as new services.

The Service On Demand Engine is related to the Synchronous/Asynchronous Service Composition of the previous chapter.

4 Demonstrator Case Studies: Automotive

Much of the research and development cost in vehicle production is due to automotive software. This leads to increased importance of software engineering technology in the automotive domain. We provide a case study for the automotive sector based on realistic automotive scenarios that encompass the full complexity of service oriented overlay architectures.

When a new vehicle is developed in an automotive company, 80% of the cost of the innovation is due to software systems. Of the total cost of the final vehicle, 25% is due to software, and each vehicle contains more than 70 ECU's (electronic control units) that require their own specific software [6]. By nature, automotive software is very service-oriented, as e.g. many units have to interplay in different constellations (service orchestration) and units will be composed by other units (service composition) (see Sensoria Ontology for service terminology [16]). Therefore, there is great potential for service-oriented software techniques in the automotive world.

A vehicle that leaves the assembly line today is equipped with a multitude of sensors and actuators that provide the driver with services that assist in conducting the vehicle more safely, for example ABS systems or vehicle stabilisation systems etc. Driver assistance systems kick in automatically when the vehicle context renders it necessary, and more and more context is taken into account (road condition, vehicle condition, driver condition, weather conditions, traffic conditions etc.). Due to the advances in mobile technology it is possible to take connectivity to the car: telephone and internet access in vehicles are possible today, giving rise to a variety of new services for the automotive domain - this is the focus of the work reported in the following.

After giving some examples of automotive scenarios, we provide in the following sections a classification of automotive scenario types. After detailing a selected scenario, we discuss non-functional requirements for automotive services and conclude by providing an initial automotive service architecture. Our next steps will consist of the detailed specification of selected scenarios.

4.1 Scenarios in the Automotive Domain

We analyzed the characteristics of many scenarios in the automotive domain, result of a brainstorming session with automotive experts. An overview of the functionalities of some typical automotive scenarios is given in this section. Most of these functionalities are integrated already in modern vehicles; others might be available to drivers in the near future. In the next section a classification of the scenarios is presented.

4.1.1 Road Sights Scenario

The driver has subscribed to the dynamic sight service offered by the car company. The vehicles GPS coordinates are automatically sent to the dynamic sight server at regular intervals, so the vehicles location is known within a specified radius. Based on the driver's preferences that were given at the beginning of the trip, the dynamic sight server searches a sight seeing database for appropriate sights and displays them on the in-car map of the vehicles' navigation system. The driver clicks on sights he would like to visit which results in the display of more detailed information about this specific sight (e.g. opening times, guidance to parking etc.).

4.1.2 Low Oil Level Scenario

During a drive, the vehicle's oil lamp reports low oil levels. This triggers the in-vehicle diagnostic system to perform an analysis of the sensor values. The diagnostic system reports a problem with the pressure in one cylinder head, and that the car is no longer driveable, and sends a message with the diagnostic data as well as the vehicle's GPS data to the repair server. Based on availability and the driver's preferences, the service discovery system identifies an adequate repair shop in the area, whose phone number and address are displayed on the in-vehicle display. The driver then makes an appointment with the repair shop; the diagnostic data is automatically transferred to the repair shop which is then able to identify the parts needed to perform the repair. After the appointment with the shop has been completed, the service discovery system identifies a towing service, providing to both the GPS data of the stranded vehicle. The driver makes an appointment with the towing service, and the vehicle is towed to the shop. This scenario is used as example in Wirsing et al. ([7]) which presents an overview of some of the techniques and methods developed in the SENSORIA approach.

4.1.3 Airbag Scenario

A driver has subscribed to the accident assistance service available for all vehicles of the car manufacturer. Due to a head on collision, the vehicles airbag is deployed which triggers an automated message to the accident assistance server that contains the vehicle's GPS data, the vehicle identification number and a collection of sensor data like for example the number of seatbelts in use and impact sensors for critical vehicle parts. The accident assistance server places a call to the driver's mobile phone. Due to his injuries, the driver is unable to answer the call. Based on the sensor data available to the accident assistance server, the severity of the accident is assessed and emergency services (police, ambulance) are alerted and provided with the vehicles location. Approaching vehicles are warned about the accident ahead through wireless messages from the vehicle involved in the accident. Accident information is successively passed on to the next approaching vehicles. This scenario is used as example in Wirsing et al. ([7]) to illustrate quantitative analysis of service orchestration.

4.1.4 Work on the Way to Work

Ms. Smith has to drive through the entire city of Munich to commute to work. Most of the time she is stuck in traffic - so her car is equipped with a computer, internet access to the computers at the company,

and even a printer, that is built into the back rest of the driver's seat. While her husband drives her to work, she is able to use the time on the back seat to get started on her daily work.

4.1.5 Tired Driver

John Doe has been driving for about 8 hours without a break. It is late at night, and the road is long and straight. The driver surveillance camera installed on the visor keeps track of John's eye movements and detects that the eyelid closes for more than just a flash while the car pulls over to the right. Since this means that John has dozed off, the vehicle stabilization system is automatically activated, and the vehicle is steered automatically. Simultaneously, a computer voice issues a wake up sound and suggests taking a break.

4.1.6 Parking Assistance

Ms. Jones is looking for a appropriate parking space at Munich's central station to pick up her husband from his business trip. To save money she wants to find a parking space on a nearby street. As she activates the parking assistant the passing parking spaces are scanned and measured by on-vehicle sensors for the correct length. After hundred meters a appropriate parking space is detected and Ms. Jones is alerted. She starts to back into the parking space with activated assistant, which breaks automatically at the correct position and supports her through automatic steering.

4.1.7 Dinner Break

Paul is very hungry since he has been driving without any food for 5 hours. So he activates the dinner service and enters a pizzeria as desired restaurant type and a price range between five and ten euros per meal. The navigation system displays a collection of nearby restaurants which match the preferred settings. Paul chooses the option to check for available seats in the participating local restaurants and as a result the map displays only restaurants with available tables. Paul chooses "Tony's Pizza" and gets his reservation acknowledged. The way to the restaurants parking lot is now displayed on the navigation map.

4.1.8 Route Planning

Steven and John and their families are on their way to Italy in two separate cars. John has entered the destination into his navigation system which is in "autonomous"-mode and thus is calculating and providing the best route during the travel. To make sure both cars take the same route, Steven's navigation system is set to "convoy"-mode, which means that the vehicle's route is guided by another vehicle (in this case John's). Steven's navigation system just receives route planning information from John's instead of performing route planning itself. Due to a collision on the highway some kilometres in front both the "autonomous"-mode and the "convoy"-mode get disabled. They get overridden by the "detour"-mode, in which Highway Emergency System takes control of the user interface and sends a warning and an alternative route to avoid the scene of accident. This example is used to illustrate the *modes approach* for software architecture presented in deliverable D6.1.a (Base Deployment Mechanisms) ([11]).

4.2 Classification of Scenarios

In a further step, a classification of scenarios is presented, making a distinction based on their "digital difference", that is, based on the different techniques required to implement and run them. Our classification is based on the following scenario characteristics:

Type of reaction: There can be a fully automated reaction, which happens merely based on sensor information, like for instance the invocation of the ABS braking assistance. Other reactions require explicit input from the driver like for instance in the road sights scenario, it is also possible that a combination of

driver input and sensor information and / or location triggers a half-automated sequence of events. The type of reaction (fully, half or non-automatic) typically requires different implementation techniques.

Communication patterns: Communication is complex in automotive service oriented software, because communication happens within the vehicle (intra vehicle), between vehicles in the vicinity (inter vehicle) and between the vehicle and the environment, for example with traffic signs or with a server through an Internet gateway (vehicle-environment). In some scenarios, all three types of communication must be invoked to allow the requested services to operate properly.

Type of information: The type of information that is processed during execution of a scenario has substantial influence on the non-functional requirements for the participating services. An example: if the information sent via a network is non-critical (as it is in the Dinner Break scenario) then security aspects are much less important than in the Rear Office example, where confidential data has to be protected. Another example would be varying bandwidth requirements depending on the type of information that is processed (video, music, voice).

Human interaction involved: Human interaction can occur through the driver or through a person outside of the vehicle, like for instance the dispatcher who tries to contact the driver via the mobile phone in case of a suspected accident.

Based on the differences in the scenario characteristics introduced above, automotive scenarios are sorted into five categories: driving assistance, accident assistance, car repair, infotainment, and rear office.

- Driving assistance supports the driver by automated vehicle reaction upon problematic conditions.
- Accident assistance provides automated or requested reaction to any event indicative of an accident.
- Car repair services deliver a reaction requested through human interaction in case of vehicle failure.
- Infotainment provides location based information or entertainment upon driver's request.
- Rear office services provide a vehicle equipped as an office and/or internet access.

4.3 Automotive Service Requirements

Non-functional service properties like accuracy, accessibility, availability, inter-operability, performance, scalabiliy, among others, are extremely important in security critical scenarios like accident assistance and driving assistance. Such services must be available and reliable at all times to all subscribers, regardless of the context. Note that scalability is extremely important for accident assistance, as it must be guaranteed that this assistance is available to each and every subscriber, regardless of the number of vehicles that require accident assistance simultaneously. For scenarios of type infotainment the service requirements could vary greatly depending on the specific information that is needed: there are no requirements other than the network being fast enough, when an entertainment service (such as a movie, music etc.) is requested. For rear office, only a network that is fast and secure enough to allow a safe internet connection is important; the requirements for car repair services are similar to those for infotainment, however the situations in which car repair services are required are in general uncomfortable, so the availability issue is more important here. For a detailed description of service properties we refer the reader to the Sensoria Ontology (deliverable D1.1.a) [16].

4.4 An Automotive Service Architecture: Initial Approach

We present a first approach to an automotive service architecture. It consists of the following modules (see also Fig.1 for a simplified car architecture):

- *Vehicle:* Models a device node which represents the physical entity of a vehicle.

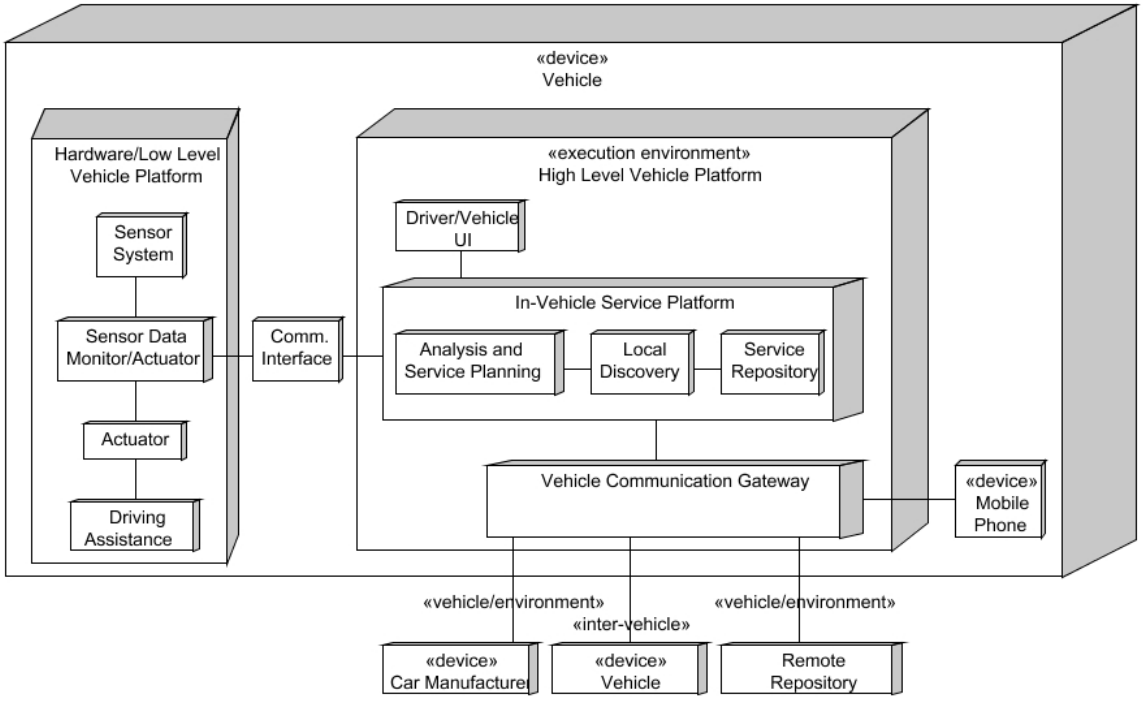

Figure 1: Simplified car architecture

- *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 System:* Contains all relevant sensors necessary to observe the vehicle's status. Each sensor has to provide information to the Sensor Data Monitor.

- *Sensor Data Monitor/Actuator:* 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 Analysis and Service Planning node.

- *Driving assistance:* Contains all low-level driving assistance functionalities (e.g. ABS) which are triggered by the Actuator Software.

- *Actuator Software:* Fully automatic triggering of the on-vehicle (low-level) driving assistance systems like ABS, anti slipping assistance and stability assistance.

- *Communication Interface:* Enables communication between low- and high-level platform of the vehicle.

- *High Level Vehicle Platform:* Models a computing platform (e.g. Java) for the higher computational functionalities of the vehicle.

- *Analysis and Service Planning:* Analyzes events reported by the Sensor Data Monitor, Driver/Vehicle UI or the Incoming Communication Handler and plans the corresponding actions while typically human interaction and/or complex communication patterns are involved. The Internal Service Repository is prompt for an appropriate service for specific actions.

- *Driver/Vehicle UI:* Models the communication interface between driver and vehicle. The driver receives information from the active services and can enter commands to trigger, stop or customise them.
- *Incoming Communication Handler:* Receives incoming communication messages from other vehicles (inter-vehicle communication) or from external entities (vehicle/environment communication) from the Vehicle Communication Gateway, transforms them if necessary and forwards them to the Analysis and Service Planning node.
- *Vehicle Communication Gateway:* Abstracts sending of a message to an external component (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.
- *In-Vehicle Services:* Models the platform for the in-vehicle discovery and execution of high-level services (e.g. Parking assistance, diagnostics). It must provide mechanisms for lookup (of the service repository), discovery and registration of services.
- *Service Communication Interface:* Models an abstraction of the vehicle's communication capabilities that is provided to services with external communication.
- *Mobile Phone:* Represents a 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.
- *Internal Service Repository:* Models a service repository, where in-car services can register themselves, are discovered, and used by the Analysis and Service Planning node and/or other services.

Further information on the automotive case studies scenarios is available in [12].

5 Demonstrator Case Studies: Finance

In a global world with high business performance it will become more and more important to get overall business information on demand. In cases of management decisions in investment and strategic business development depend on the immediately availability and quality of financial information. Like loan advisors customer consultants of financial institutions need to limit there risks by getting detailed business information of there customers. Particularly for mid cap companies the costs for publication of these information are high.
We provide two important case studies of the finance sector. The first study "Self Service" mainly aims to terminals like cash dispensers or statement printers those enable the customer to do his bank transactions without any bank advisor. The points of interest related to SENSORIA are located in the area of security (encryption and authorisation), availability, performance, QoS/SLA constraints and execution semantics (financial transactions). This case study is of special interest because of its scalability to huge amount of nodes and suitability for simulation.
The second case study "Credit Portal" processes the complete loan workflow including interactions with the bank customer, the bank advisor, the accountant and so on. Besides the descriptions of these case studies we provide in the following sections, we have generated a list with requirements on the various areas of service oriented architectures like security, semantic, service orchestration, etc. The points of interest related to SENSORIA are located in the area of orchestrating complex workflows. Additionally user and role based security, encryption and general security constraints, availability, performance and QoS/SLA constraints are also relevant. These facts are related to definitions of business processes and the non-functional properties of services in the service terminology (see Sensoria Ontology [16]). A special point of interest may be the ambitious configuration needs and the demands of flexibility related to integration into existing systems. Even the feature of adapting or changing the business workflows over an administrative interface are of interest.

5.1 Self Service

5.1.1 Business Background

Since long ago, self service systems are an accepted and esteemed part of our daily life. Nevertheless, the today's use and advantage of those systems is far away from the real possibilities and abilities. This is basically caused by the fact that the software technology the currently deployed self service systems are equipped with is only designed for a restricted range of use.
The frameworks currently used for creating self service applications are usually focused on a specific hardware, as they are provided by hardware vendors, or on the environment of a certain group of customers or a certain type of applications as the systems are normally used there. Here, the banking environment plays the most dominant role.
Regarding these background conditions, it is explainable why it is very often so difficult to set up a self service based solution in other environments although there are also many business processes that could be perfectly mapped on self service systems. To break this barrier, CETIS has been developed. CETIS - Customer and Employee Transaction and Information Service - is an integrated, flexible framework for creating Webbased self service applications that can run on ATMs as well as on transaction an information terminals or statement printers.
CETIS integrates the self service system in the customer specific processes and infrastructure conditions. It generates a strong added value by supporting the optimization of business processes and reducing the costs. This is done by providing new interactive functions to existing self service infrastructures, e.g. supporting multi channel strategies in the banking environment, as well as enabling other branches to provide information and transaction processing to new user groups, e.g. employee management in large companies or citizen services in public government.
Openness and the ability of integration in any environment are the main architecture principles of CETIS and fundamental for developing and providing future oriented self service solutions. An efficient usage of actually established technologies and standards, above all from the Open Source background, guarantees maintenance and support in an active Life Cycle Managements as well as the flexibility and economic viability of the solutions in all their components.
CETIS is independent from vendors as well as from hardware and system platforms. The interfaces between the different CETIS components and also the interfaces to backend systems and other externally integrated processes are absolutely open. Future changes in technology or special customer requirements concerning infrastructure or implementation can be accommodated in a flexible way. CETIS is based on available and accepted WEB and Internet standards and Open Source products. Following this strategy, the CETIS development can participate on the further development and enhancements in the technologies it is based on.
Self service applications developed with CETIS are high scalable, flexible extensible and provide a great synergy effect between internet development and the self service solution.

5.1.2 Architectural Overview

The central component of a CETIS infrastructure is the CETIS main-service. This service manages all components that represent a CETIS application: business logic engine (BLE) and user interface.
The CETIS main-service comprises of six main components. The "Business Logic Engine" (BLE) processes business logic based on interpretation rules and thereby integrates CETIS client-services and backend services. The "UI Generator" is responsible for the dynamic generation of user interfaces that are transported to the client-service to be displayed there. The "Backend connectors" are responsible for integration of the backend systems that are the (usually already existing) operative services that own the business data and do the transactional work. The "Security" component is responsible for identifying clients.
As the CETIS main-service is the central instance in the CETIS infrastructure it is able to provide systems management information. The CETIS main-service also allows to protocol all technical or logical

information.

The CETIS client-service offers a high level abstraction between the involved components ”server interface”, ”user interface” and ”device control”. Internally the client-service uses an own, system neutral protocol that connects the interfaces via the CETIS client-service kernel. This protocol is specified on a functional base and is not designed on technical characteristics of the device to process a task like specifies or hardware control issues. This becomes notably apparent at device controls, which are usually implemented by the help of a self service manufacturer delivered framework.

5.1.3 Identified Services

Within the CETIS client-service there are a few other services used. The ”Device access service” must be able to send events to the application at any time. The application must be able to receive and process these events at any time. So, a bidirectional communication is required. The ”System management service” collects system information and sends them to the ”System management server”. The ”Application service” provides user functionality. Within the CETIS main-service there is the ”System management service”, which is the counter part of the ”system management service” of the client-service. The ”Authorization service” is specific to the banking environment. The ”Account service” represents the access to the bank’s account management system. The ”Prepaid card balance service” is responsible for the access to the prepaid card provider. For detailed information about the general and explicit self service requirements, see [14]

5.2 Credit Portal

5.2.1 Business Background

Today collecting all the business information and providing it to the customer advisor in most cases is a manual process. In addition the same data will be edited multiple times in different applications. Also and very often the workflow is characterised by media discontinuity. This results in a time-consuming and error-prone process which stands in contradiction to an efficient loan and investment decision.
To solve these problems an optimised overall workflow is needed. The credit process portal supports the financial information supply chain to get proper information on time - for both sides: company and financial service provider. It installs a service and process oriented ”communication bridge” between financial service providers and mid cap companies. For the customer advisor in the bank the portal improves sales process and customer support. On the customer side there are additional valuable services, e.g. validation of balance sheet, business reporting, external credit and company rating.

5.2.2 Architectural Overview

The credit portal is build as a web application using a web server framework with portal features. The credit portal includes a security layer with user and role management, a data pool, a business logic engine with management and a reporting engine.
Further more the architecture supports connections to external systems, for example the operative data of the bank (booking system) and an external rating service. Please note that this architecture figure shows a logical view and simply points out that systems external to the web server are involved. In other words the credit portal handles local and remote services and/or will use local services to handle remote data or interfaces. A service orientated implementation will of course abstract from the service location.
The framework used as a base for the credit portal will be Cocoon, services will be represented by the web service standard and the business logic engine will probably be based on BPEL. In the focus of SENSORIA other business logic engines (or SENSORIA layers based on BPEL) may be used or additionally supported depending on the SENSORIA framework and tool decisions.
The business process section in the above architecture figure is titled as ”Ł management”. This implies

that the credit portal should not only provide a business logic engine. The administrative interface should allow adaptations and changes to the business workflows of the credit portal. Therefore we need tools which can be integrated into the applications it self.

5.2.3 Identified Services

In the following we report the identified services of the credit portal:

- Data storage and retrieval - Data storage and retrieval is needed in most of the workflows within the portal. Additional services to check the provided data against the banks operative data may be useful for validation and/or user friendly pre filled forms.
- Status and assignment handling - The overall workflow is very complex and will be divided into several sub workflows with multiple users or user roles. To implement of such kind of workflows a status and assignment handling is needed and should be provided as services.
- Financial rating - A bank normally supports different rating modules for different types of customers which are provided by different departments with the same signature. Unfortunately every module may rate every customer without reporting errors. But the resulting rating may be very different (wrong!) if module and customer type does not match. This situation results in requirements on ontology and service matching to allow handling of minimal differences in functional service descriptions. Therefore it may be necessary to use methods of artificial intelligence (i.e. expert systems or neuronal nets) with involving user feedback to implement the matching algorithm.
- Credit assessment - The loan request containing several business data of a customer which will be aggregated by this service to an offer.
- Statement handling - Services for direct entry of statements including validation and import can be integrated in the portal by using services. In this way a common look and feel over different applications can be provided. In the case of the credit portal the look and feel can even be shared with the customer (eventually providing a simplified mode). This common view between customer and bank employee already during capture of statement data may result in a better quality of data.
- Read/Write access to operative data - The credit portal can interact by a service with third party programs, which handling operative data. Other services are: user authentication, finished report generation (e.g. pdf, html or excel), third party services

Further information on the requirements is available in [14].

6 Experimental Case Study: Distributed E-Learning and Course Management System

6.1 Overview

Today's academic environment poses several challenges for administration, faculty, and students. Administration has to provide services for more and more students while the numbers of specialised subjects (e.g., bio-informatics and media informatics in addition to traditional computer science) increases steadily and hence more courses need to be scheduled. Faculty is facing a higher workload and increasing demands from students, e.g., with regards to the availability of homework sheets, course slides and additional teaching aids. Furthermore students are expected to spend parts of their studies in foreign countries without delaying their exams. To manage these problems efficiently and cheaply, universities are beginning to use computerised course-management and e-learning systems. Some of the functionalities performed by typical university software systems are:

- Management of curricula, i.e., the university has to decide which courses it offers, which requirements and how many credit points each course has, etc.
- Management of students by the university
- Management of single courses by teaching personnel
- Management of degrees by students
- E-learning

The increased mobility of students between universities means that different course-management systems will need to be able to exchange data, even in different countries. The Sensoria experimental case study builds a prototypical service-oriented distributed e-learning and course management system that satisfies these requirement and is used to guide Sensoria research efforts.

6.2 Management of curricula

The management of curricula is performed at the university level. The university

- decides which courses it offers in the various subjects
- defines which courses are needed for a degree
- defines courses for individual term (this includes generating the timetable for courses and exams and scheduling of rooms)

In general the first two points are done rather infrequently and require significant amounts of discussion between departments; they are not particularly suited to computerisation. Managing the timetables and exams for a term, on the other hand, can be automated to a large degree. Here the computer system has to take into account the interests and preferences of both faculty and students, e.g., by minimising the number of conflicts between lectures in common combinations of major and minor subject.

6.3 Management of degrees by students

The system also has to provide support to students for the management of their degree. This includes providing information about available subjects, and admission procedures and restrictions to prospective students who are not yet enrolled as well as the degree management by students.

A student has to plan which courses she wants to attend during the course of her studies. This needs to be done for each individual term, but also take into account the planning of the whole study time as some courses may not be available in later terms. To this end the software should provide proposals for curricula (over the duration of the studies, for a term, including courses from other universities). A course management system also has to take into account the availability of lab places and prerequisites for courses, etc.

It is particularly important to provide support for planning exchange terms with foreign universities: they may not offer the same courses, have the same prerequisites, or assign the same number of credit points as the home university and this should be reflected in the study plan.

When the student choses a bachelor or masters thesis the system should provide her with a list of the available thesis topics which are suitable for the specialities she chose during her studies.

Realising the planning software as a service-oriented system which is dynamically orchestrated at run-time makes it possible to satisfy some of these requirements which would be difficult to achieve otherwise. For example, planning has to take into account information from several different universities; by using services to provide this information and to compute local constraints, a planner can easily integrate data from different universities.

6.4 Management of single courses

Students and faculty taking part in a course can be divided into several roles: *students* taking part in the course, *tutors*, i.e., students or faculty teaching exercise groups, *correctors*, i.e., students or faculty grading homeworks, *tutorial manager*, i.e., the person responsible for determining which exercise groups exist and who is responsible for these groups, *lecturer*, i.e., the person holding the lectures. Furthermore we have an *administrator* who assigns the roles to persons in the system. There are various use cases for each of these roles:

- Students need to register for their courses, submit homework, and access their personal data (submitted homeworks, points, address data, ...). Furthermore they need to be provided with access to course material which is not publicly available.
- Tutors need to be able to access teaching aids provided for their exercise groups.
- Correctors need to be able to grade homeworks (retrieve homework, correct homework, assign marks, upload) and discuss homeworks with other correctors.
- Tutorial managers need to be able to assign tutor and corrector roles to persons and assign tutors to tutorials. They need to be able to assign homework to tutors, review graded homeworks, upload and change course material and teaching aids, prepare homeworks and decide when it is due.
- Lecturers need to prepare course material.
- Administrators can manage courses and assign administrative roles such as tutorial managers and lecturer.

6.5 E-learning

The e-learning system should provide students with additional training material or enable the integration of remote courses into the university's offers.

The e-learning system has to manage copyright restrictions on the materials, e.g., by limiting distribution of e-learning material, and disallowing unauthorised use of e-learning material as well as unauthorised copying and modification of e-learning material.

Furthermore it is desirable to integrate interactive tutors into the system which can substitute for some personal supervision which is not always possible in the context of modern universities. One example for such a tutor is the *Student Modelled Exercising on the Web (SmexWeb)* system developed at LMU. SmexWeb provides an interactive tutor for ENBF grammars that adapts to the student's capabilities. By providing these kinds of tutors as services they can be adapted to and integrated into the courses of different (cooperating) universities.

6.6 Architecture

Each functionality of the system is supposed to be provided as a service. Depending on the role and the usage scenario a dynamically collect view of available services is provided to the user. To this end, each service not only provides a functionality but also a description of itself that can be used within an HTML page, i.e., each service is a generalised kind of portlet.

The discovery services contains a semantic description of the service. A simple such description is a hierarchy of usage scenarios, e.g. `Course management/Marks computation/Economics`, the Sensoria ontology might be used to develop more sophisticated descriptions of services and their functionality.

One reason for this architecture is that it allows individual departments of the university to easily customise the system without having to rely on a central authority. For example, new interactive tutors can be integrated by simply starting a tutor-service and registering it with the service repository. Another

purpose of this architecture is to easily add and remove services without having to restart the whole system.

6.7 Security

The design of an application as a dynamically orchestrated set of services poses some interesting challenges for trustworthiness and security. When a service wants to execute an operation it is required that the user who calls the operation and the service itself have the appropriate capabilities; in particular if untrusted users start services we want to prevent these services from accessing or modifying sensitive data. Some scenarios that illustrate this point are the following:

- A student starts service requesting personal information from another service. Even when called by a person or service with appropriate access rights, the service should not be able to get the personal information of other students since the service might pass this information to the student who does not have access permission.
- A Service started with appropriate rights should not provide a student with personal information of other student then himself. We want to ensure that the service can access student data only if it is called by a person or service with the appropriate access rights. If the person has not sufficient access rights, the security system ensures that personal data is not given to the service. On the other hand a person or service with access rights to personal data cannot delegate his access rights to the service if the service itself does not have the appropriate capabilities (e.g. the service is not trustworthy regarding both, the person or service who calls the service and the service who maintains the personal data.

7 Relevant Sensoria Publications and Reports

This deliverable is based on the following SENSORIA publications and reports:

- Maurice H. ter Beek, Stefania Gnesi, Franco Mazzanti and Corrado Moiso, Formal Modelling and Verification of an Asynchronous Extension of SOAP. To appear in *Proceedings of the 4th IEEE European Conference on Web Services (ECOWS'06), Zurich, Switzerland* (A. Bernstein, T. Gschwind and W. Zimmermann, eds.), IEEE Press, 2006. Also available as Technical Report 2006-TR-19, ISTI–CNR, 2006. `http://fmt.isti.cnr.it/WEBPAPER/TRasoap.pdf`
- Martin Wirsing, Allan Clark, Stephen Gilmore, Matthias Hölzl, Alexander Knapp, Nora Koch and Andreas Schroeder. Semantic-Based Development of Service-Oriented Systems. In *Proceedings 26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06), Paris, France* (E. Najn et al., eds.), *Lecture Notes in Computer Science* 4229, Springer-Verlag, 2006, 24–45.
- Techniques for Security and Trust for Services (deliverable D3.1.a), Sensoria Report, 2006.
- An Overview of Techniques for Behavioural Properties (deliverable D3.3.a), Sensoria Report, 2006.
- Process calculi and coordination languages with costs, priority, and probability (deliverable D5.1.a), Sensoria Report, 2006.
- Base Deployment Mechanisms (deliverable D6.1.a), Sensoria Report, 2006.
- Description of Scenarios for Automotive Case Study, Sensoria Report, 2006. `http://www.pst.ifi.lmu.de:8080/automotivecasestudy`

- E-learning Case Study Definition and Requirements, Sensoria Report, 2006.
  `http://www.pst.ifi.lmu.de:8080/elearningcasestudy`
- Sensoria S&N Finance Case Study Definition and Requirements, Sensoria Report, 2006.
  `http://www.pst.ifi.lmu.de:8080/FinanceCaseStudy`
- Sensoria Telecommunication Case Study, Sensoria Report, 2006.
  `http://www.pst.ifi.lmu.de:8080/telcocasestudy`
- Sensoria Ontology - Prototype language for service modelling: ontology for SOAs presented through structured natural language (deliverable D1.1.a), Sensoria Report, 2006.

References

[1] 3GPP, Open Service Access (OSA) - Parlay X Web Services (Release 6), TS 29.199 v6.x.y.

[2] OASIS, eXtensible Access Control Markup Language (XACML) TC.

[3] IBM, Microsoft et al., Web Services Policy Framework (WS-Policy), 2004.

[4] Open Mobile Alliance, Policy Evaluation, Enforcement and Management Requirements - Draft Version 1.0, September 2004.

[5] 3GPP, Open Service Access (OSA) - Application Programming Interface (API) (Release 6) - Part 13: Policy Management, TS 29.198 v6.x.y.

[6] A. Saad: Java-based Functionality and Data Management in the Automobile. Prototyping at BMW Car IT GmbH. Java Spectrum, March 2003.

[7] Martin Wirsing, Allan Clark, Stephen Gilmore, Matthias Hölzl, Alexander Knapp, Nora Koch and Andreas Schroeder. Semantic-Based Development of Service-Oriented Systems. In *Proceedings 26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06), Paris, France* (E. Najn et al., eds.), *Lecture Notes in Computer Science* 4229, Springer-Verlag, 2006, 24–45.

[8] Techniques for Security and Trust for Services (deliverable D3.1.a), Sensoria Report, 2006.

[9] An Overview of Techniques for Behavioural Properties (deliverable D3.3.a), Sensoria Report, 2006.

[10] Process calculi and coordination languages with costs, priority, and probability (deliverable D5.1.a), Sensoria Report, 2006.

[11] Base Deployment Mechanisms (deliverable D6.1.a), Sensoria Report, 2006.

[12] Description of Scenarios for Automotive Case Study, Sensoria Report, 2006.
`http://www.pst.ifi.lmu.de:8080/automotivecasestudy`

[13] E-learning Case Study Definition and Requirements, Sensoria Report, 2006.
`http://www.pst.ifi.lmu.de:8080/elearningcasestudy`

[14] Sensoria S&N Finance Case Study Definition and Requirements, Sensoria Report, 2006.
`http://www.pst.ifi.lmu.de:8080/FinanceCaseStudy`

[15] Sensoria Telecommunication Case Study, Sensoria Report, 2006.
`http://www.pst.ifi.lmu.de:8080/telcocasestudy`

[16] Sensoria Ontology - Prototype language for service modelling: ontology for SOAs presented through structured natural language (deliverable D1.1.a), Sensoria Report, 2006.