Project acronym:	OVERSEE
Project title:	Open Vehicular Secure Platform
Project ID:	248333
Call ID:	FP7-ICT-2009-4
Programme:	7th Framework Programme for Research and Technological Development
Objective:	ICT-2009.6.1: ICT for Safety and Energy Efficiency in Mobility
Contract type:	Collaborative project
Duration:	01-01-2010 to 30-06-2012 (30 months)

Deliverable D1.4:
Functional Requirements Analysis

Editor:	Florian Friederici (Fraunhofer FOKUS)
Reviewers:	Rafael Grote, David Linner (TU Berlin) Alfons Crespo (Universidad Politécnica de Valencia)

Dissemination level:	Public
Deliverable type:	Report
Version:	2.1
Submission date:	12 July 2011

Abstract

This is the revised version of the functional, dependability and security requirements document. It contains the overview on the functional and protection requirements for the OVERSEE platform. The requirements are derived from the use cases, OVERSEE will be designed for and are backed up by more generic requirements from the integrated hypervisor and the requirements originated by standards. The requirements are specified according to the vision of the OVERSEE consortium, to build the upcoming platform for vehicular applications.

Contents

List of Figures

List of Tables

List of Abbreviations

API	**A**pplication **P**rogramming **I**nterface
ARINC	**A**eronautical **R**adio **Inc**orporated
CAN	**C**ontroller–**a**rea **N**etwork
CPU	**C**entral **P**rocessing **U**nit
DSRC	**D**edicated **S**hort-**r**ange **C**ommunications
ECU	**E**lectronic **C**ontrol **U**nit
GPOS	**G**eneral **P**urpose **O**perating **S**ystem
GPRS	**G**eneral **P**acket **R**adio **S**ervice
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile Communications
HMI	**H**uman **M**achine **I**nterface
ID	**I**dentity
IT	**I**nformation **T**echnology
ITS	**I**ntelligent **T**ransportation **S**ystem
OEM	**O**riginal **E**quipment **M**anufacturer
OS	**O**perating **S**ystem
OSEK	**O**ffene **S**ysteme und deren Schnittstellen für die **E**lektronik im **K**raftfahrzeug
OVERSEE	**O**pen **Ve**hicula**r** **Se**cur**e** Platform
POSIX	**P**ortable **O**perating **S**ystem **I**nterface (for Uni**x**)
PP	**P**rotection **P**rofile
RTOS	**R**eal-**t**ime **O**perating **S**ystem
UC	**U**se **C**ase
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
USB	**U**niversal **S**erial **B**us
V2I	**V**ehicle-to-**I**nfrastructure
V2V	**V**ehicle-to-**V**ehicle
V2X	**V**ehicle-to-**X**
WAVE	**W**ireless **A**ccess for **V**ehicular **E**nvironment
Wi-Fi	**Wi**reless **Fi**delity

Document History

Version	Date	Changes
2.1	28-06-2011	Included review comments from UPVLC and TUB. Final version.

1 Introduction

The Open Vehicular Secure Platform (OVERSEE) project has produced this deliverable [4]; therefore it contains contributions from all partners even if Fraunhofer FOKUS is the main contributor.

The present document is the final version of the document about functional, dependability and security requirements. It replaces the initial version of this document, D1.2 [2].

Specification of non-functional requirements and constraints is not part of Task 1.2. Task 1.3 evaluates them, resulting in the deliverables D1.3 [3] and D1.5 [5].

1.1 Definitions

The terms and definitions given in D1.1 [1] apply for this document, if not otherwise noted. Additionally the following definitions apply:

- "**OVERSEE**" is the term for the platform under development.
- Demonstrator or "**OVERSEE-demonstrator**" is the proof-of-concept implementation.
- "**Facilities**" is the term for commonly used services.
- Facilities which will be available to guest-OS and applications on top of OVERSEE will be accessible through the "**OVERSEE-API**".

Utilising the terms above, the boundaries of the OVERSEE platform are defined as follows:

- OVERSEE system boundaries **included** parts:
  - a hypervisor[1]
  - the OVERSEE-API
  - a set of facilities
- OVERSEE system boundaries **excluded** parts:
  - hardware
  - applications
  - guest operating systems

This distinction is shown in Figure 1 and is required for the further process within the project, to clarify which parts of the OVERSEE System have to be developed by the consortium. The OVERSEE-demonstrator however will of course consist of hardware with OVERSEE, a guest-OS and applications on top. Hardware requirements are discussed in chapter 4.4.

[1] See Chapter 3.3 for more details.

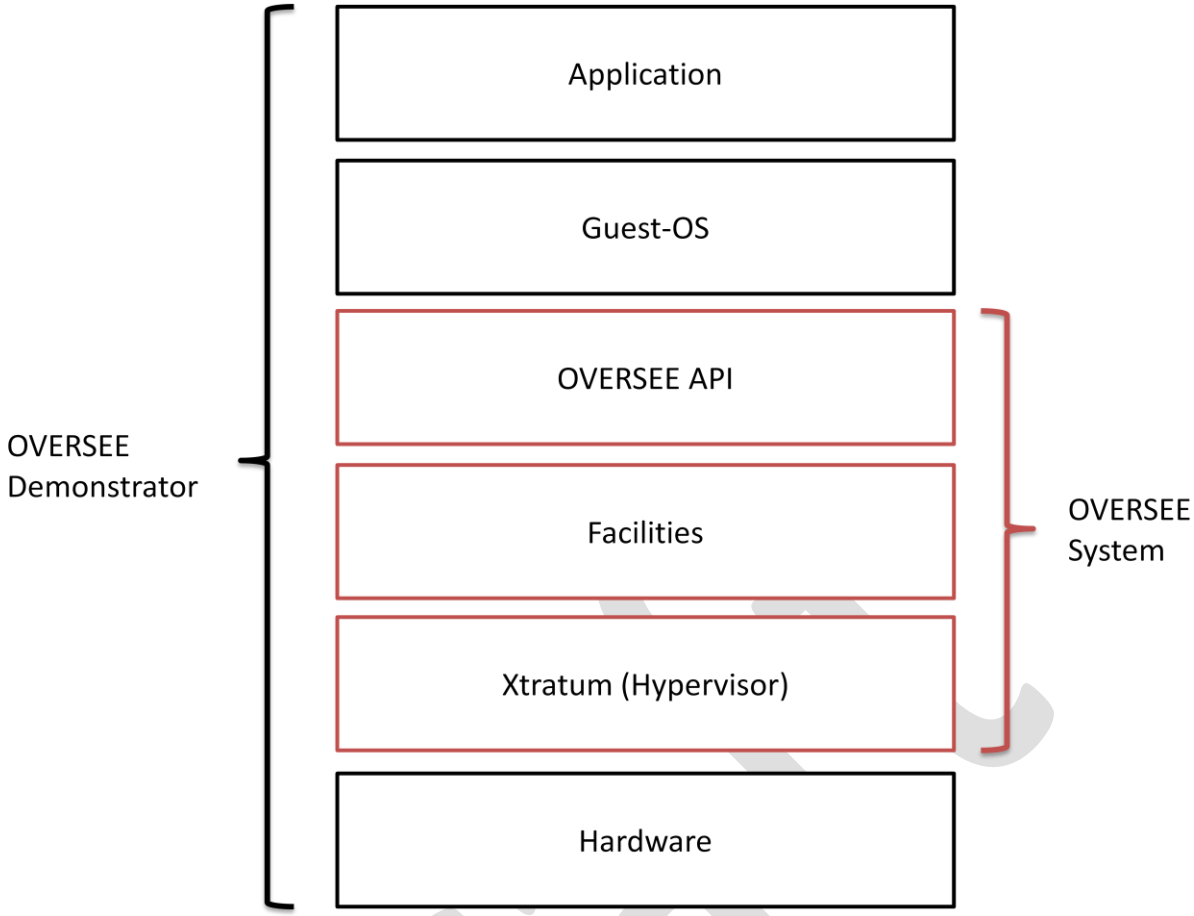


Figure 1: OVERSEE System

1.2 Scope

The present document provides the functional and protection requirements for the use cases defined in [1]. The scope and purpose is limited to:

- Functional requirements
- Dependability requirements
- Security requirements

Derived from:

- The use case analysis
- The underlying core component XtratuM
- Applicable standards
- Matching protection profiles

The addressed audience of the document is the WP2 team (design of platform) of the OVERSEE project, to start the platform design. It is not the objective of the present document, to cover all **possible** requirements of the OVERSEE platform, rather it is meant to outline the requirements **important** for the design team. These are:

- What OVERSEE requires from the excluded parts

- What the excluded parts require from the included parts (OVERSEE)

The specification of the OVERSEE-demonstrator is out of scope of the document. The demonstration will show a possible OVERSEE implementation on real hardware and selected use cases. However, in task 1.2 we concentrate on the platform itself and in particular on the implementation independent basic requirements of OVERSEE.

1.3 Document Outline

The document is structured as follows: Section 2 introduces the method and procedures used, in order to accomplish the work within task 1.2, for the current deliverable. Section 3 outlines the predefined OVERSEE architecture vision. In Sections 4 and 5 requirements originated by the work of others are discussed. Finally, section 6 explains how the project is going to proceed. Appendix A shows a tabular summary and lists all current requirements in order of their category.

2 Requirements Inquiry Method and Procedure

2.1 Relevant Use Cases

In D1.1 [1] a number of use cases relevant to OVERSEE were identified. These use cases reflect a broad variety of possible applications out of the following categories: *Operation*, *Driver Assistance Systems*, *Convenience*, *Mobility*, and *Security*. The OVERSEE consortium selected 20 use cases to be considered for the design of the OVERSEE platform. Table 1 shows the selected use cases. The selection represents each of the categories named above.

No.	ID	Use Case Name	ETSI Mapping
1	UC-TOLL	E-Toll	C.2.9
2	UC-eC	E-Call	C.3.10
3	UC-PLR	Parking Lot Reservation	C.3.2
4	UC-SCoND	Secure Integration of Nomadic Devices	C.3.9
5	UC-LDW	Meta-UC: Hazard Warning between Vehicle	C.1.4
	UC-EVSP 1/2	Emergency vehicle signal pre-emption (1 & 2)	C.1.2
6	UC-SVT	Stolen vehicle tracking	C.3.11
7	UC-TIE	Traffic Information between Entities	C.2.3
8	UC-ABR	Safety Reaction: Active Brake	C.1.1
9	UC-PTC	Personalize the Car	n.a.
10	UC-CF	Car Finder (e.g., via mobile phone)	n.a.
11	UC-ELP	Electronic License Plate	n.a.
12	UC-DTM	Dynamic Traffic Management	C.2.4
13	UC-BDC	Break-Down-Call-Live-Check	C.3.12
14	UC-RCC	Remote Car Control (e.g. door opener)	n.a.
15	UC-PAYD	Pay as You Drive (Road Safety & Eco)	n.a.
16	UC-IA	Install Applications	n.a.
17	UC-VRE	Remote Vehicle Rental	C.3.4
18	UC-PSS	Parking Sensor System	n.a.
19	UC-ELB	Electronic Driver Logbook	n.a.
20	UC-WEB	Web 2.0 for Cars	C.3.8

Table 1: Use case selection of D1.1

Table 1 includes a mapping between OVERSEE use cases and use cases from the ETSI TC ITS basic set of applications [12]. More than half of the relevant use cases are typical vehicular communication use cases. However, OVERSEE will not only be capable of running plain communications use cases, but also the more local, vehicle bound use cases. The high number of vehicular communications use cases requires us to take the current standardisation work by ETSI TC ITS into account. Most important are the requirements document [10] and the threat, vulnerability and risk analysis [11] by ETSI TC ITS.

2.2 Virtualisation Layer Requirements

The Virtualisation layer is the core component of the OVERSEE platform. It provides virtual execution environments. The OVERSEE virtualisation layer is based on the XtratuM hypervisor [6].

XtratuM is a hypervisor specifically designed for real-time embedded systems which will be used in aerospace for a new generation of satellites. European Space Agency (ESA) and Centre Nationale d'Etudes Spatialles (CNES) have launched several projects for the adaptation of XtratuM to LEON processors that are used in space systems. The goal of this ESA and CNES program ("IMA for Space") is to define a standard for space applications, adapting previous work done in the aeronautic sector that defined a concept as Integrated Modular Avionics (IMA) based on partitioned kernels. A standard as ARINC-653 has been defined for partitioned systems in avionics.

In this framework, both organisations have defined a set of requirements to fulfil the needs of the platform. The requirement list generated shows more than 200 requirements covering functional and non-functional aspects. Next list shows different aspects covered by these requirements:

- Core virtualisation elements: Refers to the hardware components that have to be virtualised:
  - Processor
  - Memory
  - Interrupts
  - Clocks and Timers
  - Floating Point Unit
  - Devices
  - Input Output
- Partitioning: Specifies the temporal and spatial isolation properties and the actions that partitions can perform on the global state.
  - Partition execution: How partitions are executed enforcing temporal isolation.
  - Scheduling properties: How partitions are scheduled.
  - Memory allocation of partitions: How partitions are allocated in memory preserving the spatial isolation.
  - Partition actions on the own state: How a partition can change its state.

    - Partition attributes: How partitions with specific rights can perform some actions on other partitions.
    - Partition and memory access: How partitions with specific rights can access to other memory areas.
    - Partition and resources: How partitions can access to the virtualised resources, devices and IO.
- Clock and Timers: Specifies the clock and timer services to be used by partitions.
    - Clock definition: Global and local (to the partition) clocks are defined.
    - Timers: Types and attributes of timers.
    - External synchronisation: How the clock is referred to an external signal from Earth.
- Inter-partition communication: Specifies the concepts and mechanisms used by partitions to exchange information. It is based on ARINC-653 inter-partition mechanisms.
    - Sampling ports: Specifies the behaviour of communication through ports that have no buffering.
    - Queuing ports: Specifies the communications through ports that have buffering.
    - Shared memory: Specifies the communication between partitions through shared memory.
- Health Monitoring: Details the requirements associated to the Health Monitor (HM) events
    - HM Mechanisms: Specifies how the system detects and handles system traps.
    - HM Actions: Specifies the set of actions to be taken depending on the trap generator.
    - HM Log: Specifies how the HM events are logged.
- Initialisation: Details how the system is initialised, states and actions.
- Monitoring: Details the format of the traces generated by partitions and hypervisor.
- Auto-tests: Defines the auto-test to be executed by the hypervisor in order to inform about the internal status.
- Modes: Specifies the different modes of the system and the protocol to be applied when a mode change is produced.
- Performance: Details the internal measurements to be performed and the mechanisms to provide this information to the partition with appropriated rights.
- Quality: Specifies the procedures to be applied to develop and validate the software.
- Configuration: Defines the requirements to be applied to identify and configure the parameters related to the different components.
    - Processor: Frequency, number of processors, etc.

- Memory regions: Initial address, size, rights, flags.
- Communication channels: Input, output, message length, number of messages, etc.
- Scheduling plan: Mode, slots, duration, etc.
- Devices: IO addresses, number of registers, etc.
- Partitions: Name, identifier, list of memory regions, list of IRQs, list of devices, etc.

2.2.1 Selection Criteria

This set of requirements shows very detailed requirements about all the aspects that have to be validated. The goal of the requirement analysis has been to analyse and select a subset of requirements that:

1. Cover all the aspects of the virtualisation layer.
2. Avoid excessive details or group similar functionalities.
3. Remove specific formats and implementation details for space applications.

In order to explain these selection criteria, some examples are shown:

- Criteria 1.
  - We have included requirements covering the aspects considered previously.
- Criteria 2.
  - In the original requirements, it can be found several requirements as:
    - REQ: PAR_030: A partition with system rights shall start any other partition.
    - REQ: PAR_040: A partition with system rights shall stop any other partition.
    - REQ: PAR_050: A partition with system rights shall suspend any other partition.
    - REQ: PAR_060: A partition with system rights shall resume any other partition.
    - REQ: PAR_070: A partition with system rights shall shutdown any other partition.
  - Proposed requirement: *Requirement Platform-Part1 Partitioning*
    - "System partitions" can control the system execution (partition security, robustness, etc.) and take actions to start/stop/resume/reset/shutdown partitions.
- Criteria 3
  - In the original requirements, it can be found detailed:

- REQ: MON_040: Event traces shall provide the information of operation code (opcode); partition (module) and additional information.
- REQ: MON_050: The opcode value 0 shall be reserved for “no error”.
- REQ: MON_060: The hypervisor shall be the modules numbered 0x01.

2.3 Requirement Analyses Method

The objective of this document is to get a preferably complete overview of requirements for the OVERSEE platform. This can be achieved at best by analysing the requirements for all selected use cases. These use cases are supposed to cover most requirements since they reflect all possible categories of applications. Additionally, platform requirements are derived from the analysis of the used platform *XtratuM [6]*, which has been done earlier. Figure 2 outlines the workflow between the task in work package 1 and the output towards work package 2.

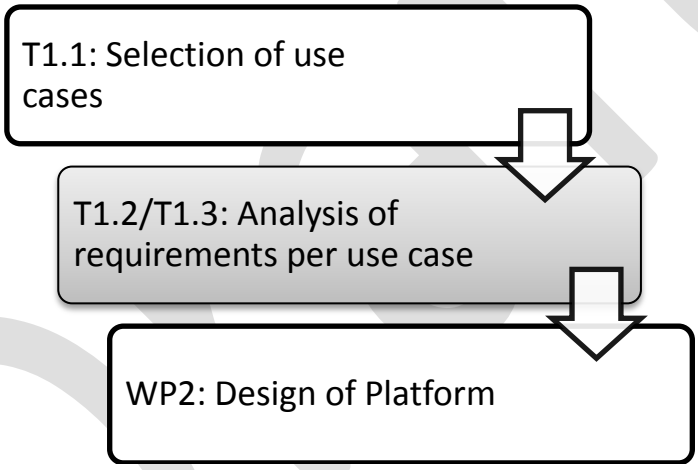


Figure 2: OVERSEE Workflow

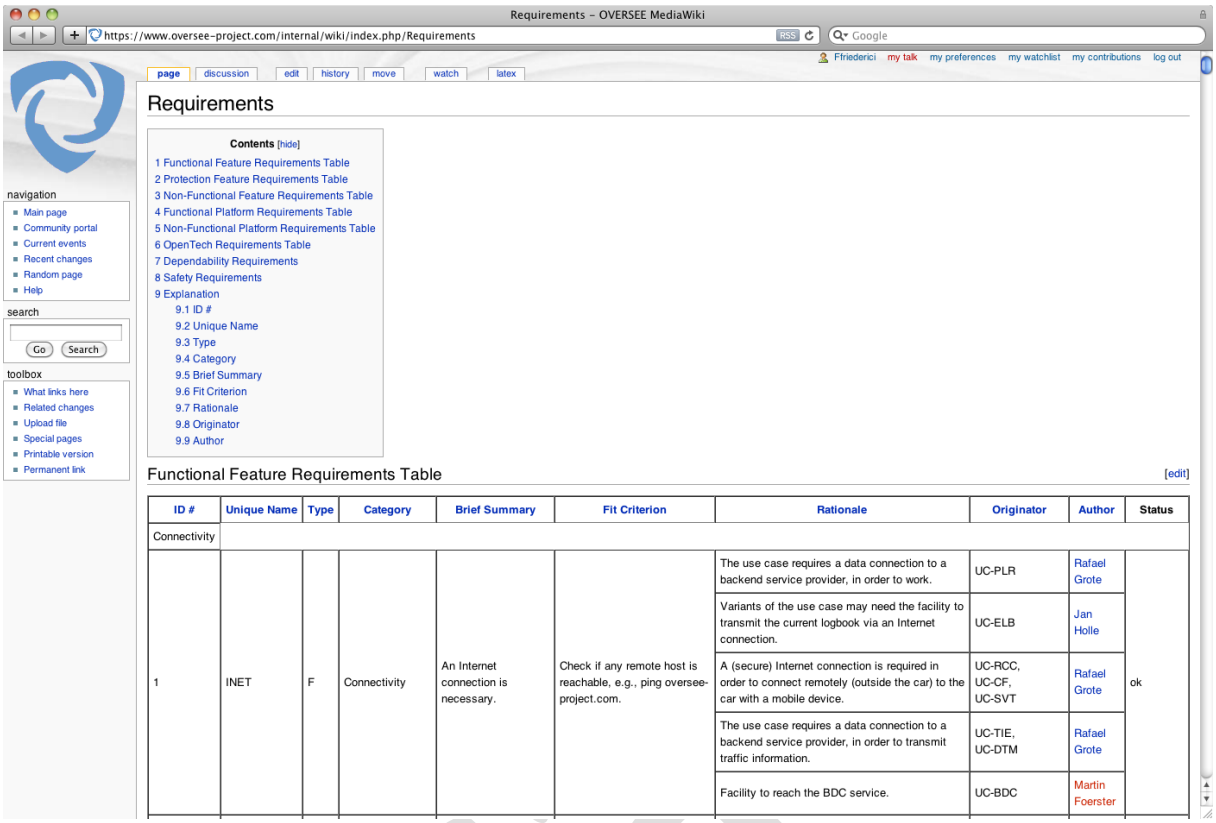
2.3.1 Requirements Filtering and Selection

The process from the collection to the deliverable output contained different steps, which are outlined in the following.

- Requirements collection
- Sorting and filtering
- Editorial review
- Automatic generation of word template

Each project partner has been assigned a number of use cases to analyse. The resulting requirements where collected within the internal OVERSEE project wiki. In that manner, doubling of requirements was avoided. Additionally, the most recent requirements are always available to the consortium, which is especially important with regard to the

overlapping schedule of work package 2. Figure 3 shows a screenshot of the internal requirements list.



ID #	Unique Name	Type	Category	Brief Summary	Fit Criterion	Rationale	Originator	Author	Status
Connectivity									
1	INET	F	Connectivity	An Internet connection is necessary.	Check if any remote host is reachable, e.g., ping oversee-project.com.	The use case requires a data connection to a backend service provider, in order to work.	UC-PLR	Rafael Grote	ok
						Variants of the use case may need the facility to transmit the current logbook via an Internet connection.	UC-ELB	Jan Holle	
						A (secure) Internet connection is required in order to connect remotely (outside the car) to the car with a mobile device.	UC-RCC, UC-CF, UC-SVT	Rafael Grote	
						The use case requires a data connection to a backend service provider, in order to transmit traffic information.	UC-TIE, UC-DTM	Rafael Grote	
						Facility to reach the BDC service.	UC-BDC	Martin Foerster	

Figure 3: OVERSEE Internal WIKI Requirements

After the initial collection was finished, the requirements where sorted into categories and the requirement type was checked. An editorial review of all collected requirements was done, to ensure the consistency.

The deliverable output of the requirements is automatically generated with the assistance of scripts. Only the requirements which are marked with the status “ok” in the WIKI, were included in the deliverable. Duplicates and requirements with flaws were dropped through this mechanism.

2.3.2 Collected Information

For each requirement, a set of data is collected and documented. This data includes the information described in the following. Every requirement is fully specified by the sum of all elements, however can be identified by only the unique name or ID #.

2.3.2.1 Unique Name

A human readable and recognizable name to identify the requirement is contained in italic letters in the headline of each requirement.

2.3.2.2 ID

Additionally, a unique numeric id identifies each requirement.

2.3.2.3 Type

The type of requirements may be one of these:

- F - Functional
- P - Protection
- N - Non-functional

The type is separated from the name and id, because it would be easier to change the type afterwards if we find that some requirement should have another type.

2.3.2.4 Category

Requirements are assigned to one of the following categories:

- Connectivity – Required hardware interfaces of the OVERSEE system
- API-Method – Required API calls
- Configuration – Required configurable behaviour of the OVERSEE system
- Security – Required security features
- External-API – Required external APIs to connect to
- Communication – Required communication protocols
- Virtualisation – Required virtualisation features
- Partitioning – Required partitioning features
- Interface – Required interfaces

The List of Requirements in section 3 is divided into subsections for each category. Requirements are ordered by means of this category field.

2.3.2.5 Brief Summary

This section gives a short description of the requirement. Here we expect a simple sentence, giving a clear statement what is required. The requirement itself is fully described by the "Brief Summary". The other fields (Fit Criterion and Rationale) are only additional information to the requirement itself.

2.3.2.6 Fit Criterion

The fit criterion describes a how to measure if the requirement is met. Even if we do not use it for validation purposes, it is necessary to make sure, that we only define requirements that can be validated.

2.3.2.7 Rationale

The rationale justifies the requirement. If the requirement originates from multiple use cases (or other origins) various rationales may be given. The corresponding origins are mentioned in brackets after each rationale. An origin might be a use case, a person, legal regulations, etc.

3 OVERSEE Prerequisites

This section outlines the vision of OVERSEE. This vision has a major impact on the high-level requirements of OVERSEE and should be taken into account.

3.1 Idea of OVERSEE

The idea of OVERSEE can be split in two main parts: first, the *open and secure platform* for the execution of OEM and non-OEM applications and second, the *secure single point of access* to ITS communications. However, only the combination of these two aspects will offer the potential for a wide range of new automotive applications.

The automotive applications running on OVERSEE will be executed in protected runtime environments for maximum dependability and security. Applications are prevented from influencing each other, the OVERSEE platform, or communications on the connected networks - especially on the vehicle internal networks. To achieve this goal, virtualisation is one of the main concepts of OVERSEE. The applications will be executed in runtime environments that abstract from the physical hardware. They are controlled by the virtualisation system. The concepts of virtualisation are well known in the IT domain but not applied to automotive applications yet. Virtualisation will be the solution to offer a temporal and spatial partitioning platform to execute several execution environments (applications) on one physical OVERSEE-ECU with very low overhead while increasing the reliability of the applications.

The improvements will be possible because of the intensive consideration of security, dependability and reliability issues within the development of OVERSEE. The API for developing OVERSEE applications will be publicly available to increase the quantity of available applications in the short term.

The communication interface of OVERSEE is be based on existing standards, and those currently under development, e.g. in ETSI TC ITS. Thus, it is possible to connect most recent and new vehicle internal and external networks with only small effort. As security issues are an integral part of OVERSEE, connecting new networks would be possible without the fear of creating new backdoors for attackers.

The security of communication via OVERSEE and with the applications executed on OVERSEE will be based on a small and well-defined message and command set. The message and command set will be defined in an early stage of the project. The access of applications executed on OVERSEE to the communication interfaces as well as the incoming interfaces of OVERSEE will be protected by a message filtering firewall. The OVERSEE firewall will be customisable by user policy rules. The format of the policy rules and the firewall implementation will be developed within this project. The policy format will be based on publicly available standards.

In order to provide a basis for secure and trustworthy communication via the connected networks, OVERSEE will provide a cryptographic API (e.g. encryption and decryption, signing and verify of signatures), as well as the interfaces for platform identity management to establish trust in large networks. By using OVERSEE and the provided APIs, developers shall

be brought into the situation to quickly and efficiently develop new automotive applications and integrate security and dependability aspects right from the start.

As another core aspect for establishing secure and dependable applications within an ITS environment, the capabilities to validate future OVERSEE compliant open platform implementations will be created. This enables automotive suppliers and future projects to offer OVERSEE compliant platforms and applications easily. Openness and ease of use for security and dependability mechanisms will promote the OVERSEE platform and make it widespread. Economics of scale as well as available tools will reduce the costs for developing OVERSEE compliant ECUs and applications.

3.2 OVERSEE Vision and Objectives

The vision of the OVERSEE project is to provide an open platform for secure and dependable vehicular applications as well as the tools, services and their integration into ITS standards necessary for its widespread acceptance.

The consortium agreed on the following objectives for the OVERSEE project:

- Providing a generic and open source platform for spatial and temporal partitioning of secure simultaneous execution of multiple innovative automotive applications on one single OVERSEE-ECU
- Providing a secure and dependable runtime environment
- Create an open and standardized secure single point of access to in-vehicle networks
- Providing a standardized API for accessing security and dependability services
- Providing validation support capabilities and tools
- Providing the capabilities of secure and non-deniable recording

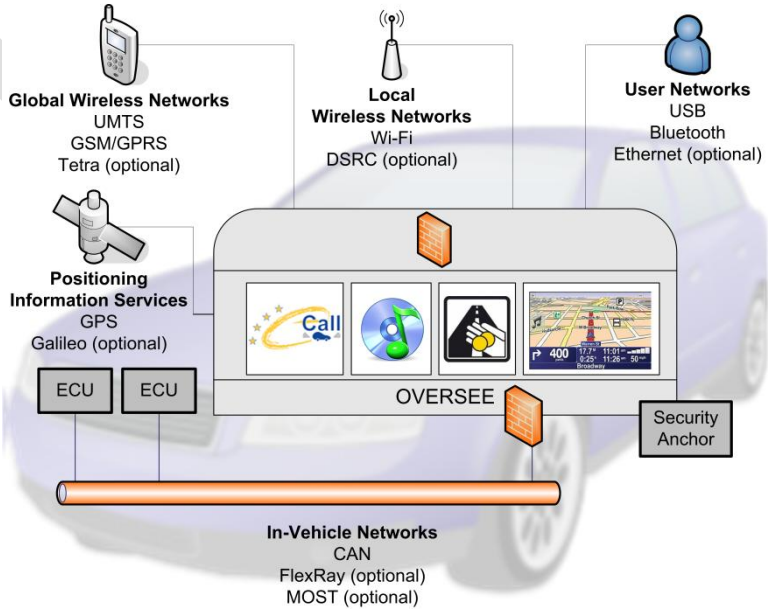


Figure 4: OVERSEE Environment

3.3 Predefined Framework Architecture of OVERSEE

The Figure 5 shows the general architecture concept for OVERSEE.

In this figure, three main layers can be identified:

- **Hardware layer.** It includes the hardware devices: CPU, memory, peripherals, clocks and timers, hardware support for secure key storage and cryptographic services, etc.
- **Virtualisation Layer.** It virtualises the hardware resources and offers virtualised services to the execution environments. It also guarantees the temporal and spatial partitioning of the system resources. The main virtualised resources are CPU, memory, interrupts, clock, timers and security module. The Health Monitor included in this layer performs an earlier detection of possible errors and reacts to anomalous events or states trying to solve or isolate the faulting subsystem in order to avoid or reduce the possible consequences. The main issues in the design of this layer for dependable and secure real-time embedded systems, should consider:
    - Spatial isolation. Partitions should be isolated from others avoiding the access from them.
    - Temporal isolation. Partitions should be executed under real-time scheduling policies that can guarantee the real-time constraints independently of other partitions.
    - Resources virtualisation: basic hardware components as clock and timers, interrupts, memory, cpu, time, serial i/o, need to be virtualised to partitions
    - Efficient and deterministic system services.
    - Efficient and secure inter-partition communication.
    - Cryptography services to partitions
    - Health monitoring
    - Low overhead and footprint.
- **Application Layer.** It includes the execution environments (partitions) where the applications are executed. A partition is composed by an operating system and the applications. Partitions can be built using a specific operating system according to the application needs (real-time, secure real-time or general-purpose operating system).

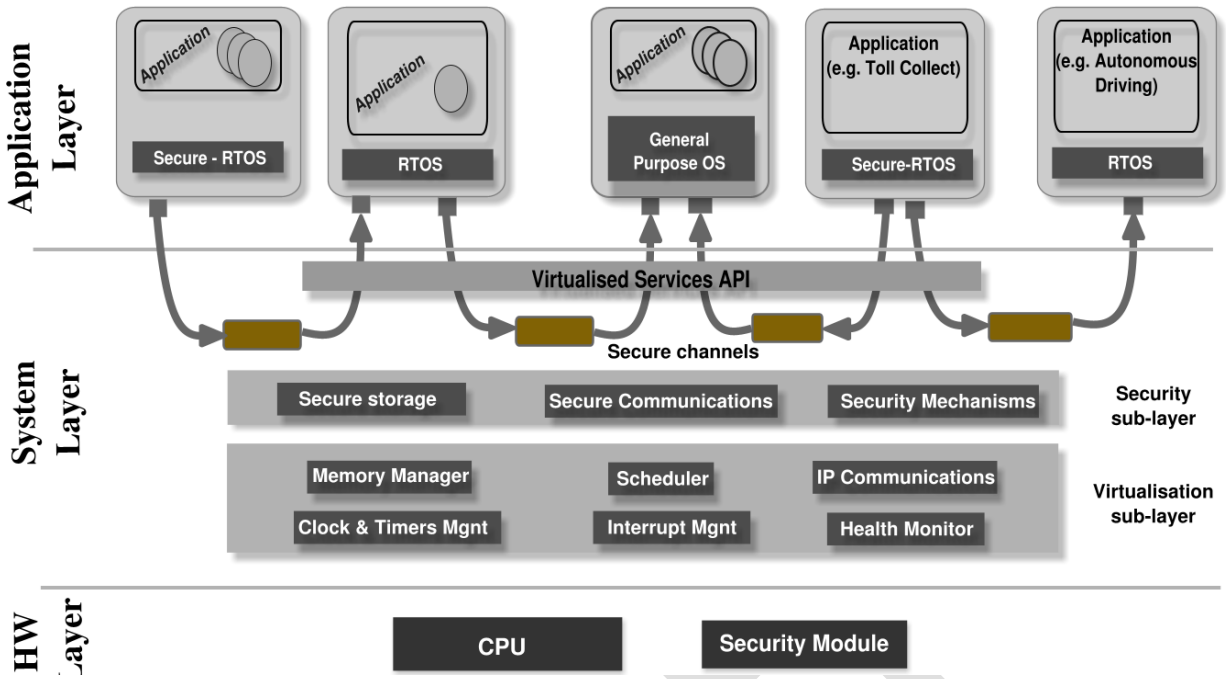


Figure 5: OVERSEE Architecture

4 Standard Requirements

Standards have a great impact on the requirements of OVERSEE, since OVERSEE shall offer access to standardised communication systems. On the other hand, OVERSEE shall define an API for applications to utilise all of OVERSEE's facilities. To achieve the goals defined in Chapter 3.1, it will be necessary to bring parts of OVERSEE into standardisation as well.

This section outlines which national-, international- and industry-standards shall be taken into account in the design of OVERSEE. According to the definition of scope and OVERSEE vision, standards for operating systems, virtualisation, communications and hardware are evaluated.

4.1 Operation Systems and Applications

Part of the OVERSEE concept are so called partitions, including the combination of an operation system and one or many applications. This section discusses issues around the content of those partitions.

4.1.1 Supported Operation Systems

- Real-Time Operation Systems (RTOS)
- General Purpose Operation Systems (GPOS)

The support for Real-Time Operation Systems is required, to enable OVERSEE to run legacy applications on the one hand, as well as newly developed time critical applications. One of the most famous RTOS within the automotive domain is OSEK, see [7] and [8]. Alongside they are other RTOS which may run on OVERSEE, however the OVERSEE design must at least support OSEK guests.

Alongside AUTOSAR is a concept developed within the automotive industry, to advance standards around electronic control units, see [13], [14]. The design of OVERSEE shall consider these developments.

Most applications without hard real-time constraints will require some kind of GPOS. The demand is mainly to support POSIX compliant systems like GNU/Linux. Most infotainment applications in the automotive domain are however developed to be executed either on (embedded) Microsoft Windows systems or on top of a JAVA/OSGi framework. Therefore OVERSEE should also be able to execute those.

4.1.2 OVERSEE facilities and API

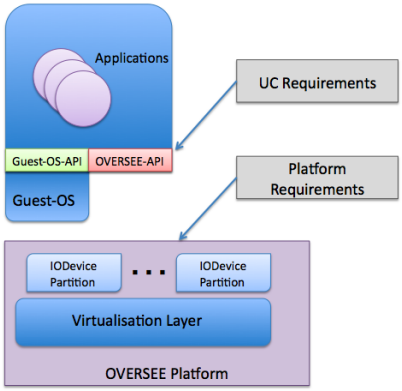


Figure 6: OVERSEE architecture from D1.3

An application running within a guest on top of OVERSEE may use the API of the corresponding guest OS, as well as the OVERSEE-API, cf. Figure 6. The OVERSEE-API shall grant access to all OVERSEE facilities independently of their concrete implementation and location in the overall OVERSEE framework.

The design of the OVERSEE-API shall be driven by the requirements collected from the use-cases, while the platform design shall be driven by the platform requirements, see Annex A.

4.1.3 Requirements from OS and Applications to OVERSEE

Since the operation system is nowadays in charge of abstracting the hardware to the applications, the interface between OVERSEE and the guest OS is the only part where requirements arise. OVERSEE shall offer a virtualized hardware to every guest partition wherever possible, so that only minimal or in the best case no changes to the guest OS are required.

4.2 Virtualisation

Since Virtualisation is the main architecture concept of OVERSEE, possible standards should be carefully examined.

4.2.1 Hypervisor

OVERSEE will build upon the XtratuM hypervisor, which features a scheduling policy based on ARINC-653 cyclic scheduling.

4.2.2 Virtual facilities

Facilities are most likely to be implemented in a separate service partition of OVERSEE and are accessible through the OVERSEE-API. The demand of facilities is specified by the requirements of the use-cases and should be designed like in the related projects. Within the standardisation of ETSI TC ITS, so called SAPs (Service Access Points) are defined between

the different layers of an ITS-Station. The OVERSEE design should align with the already available SAPs, wherever possible.

4.3 Communications

Enabling paravirtualised partitions to access the OVERSEE communication channels is of major importance to the vision of OVERSEE. The requirements of the single communications technologies are defined by the individual standards of the corresponding medium.

The access to those can be divided in the following parts:

- Physical device
- Device driver
- Communications stack
- OVERSEE-API access

Connecting the physical device to an OVERSEE implementation (like the demonstrator) is discussed in chapter 4.4. The corresponding device driver may run inside the guest OS or within the hypervisor layer, depending if the device will be available to only a single or multiple guest partitions. The communications stack will be within the guest OS in any case. Some shared communications media will require accessibility through the OVERSEE-API.

4.3.1 Required Communication Interfaces

The following communication interfaces are needed as identified by the use-case analysis. They therefore form a minimal set of communication interfaces, which have to be taken into account by the OVERSEE-API design.

- Positioning Systems (e.g. GPS)
- LF Sound in-/output (e.g. speakers and microphone)
- In-vehicle bus via an OEM gateway (e.g. CAN-bus)
- Personal area networks (e.g. Bluetooth, WiFi)
- Smartcard interface (wireless or with contact)
- Wireless ITS communications (European 5.9 GHz ITSG5A/B/C, CEN DSRC)
- Public land mobile operator networks (e.g. UMTS, GSM/GPRS)

4.4 Hardware

Hardware discussions are in principle out of scope of the OVERSEE platform, but a few remarks should be taken into account. Concerning quality requirements:

- There are industry standards for hardware components.
- Automotive-grade equipment has to fulfil requirements regarding the storage and usage temperature ranges and other environment conditions.

Those quality requirements are discussed in D1.5 [5].

Implementation specific functional requirements may arise from:

- Used CPU architecture
- Available interfaces (limiting connectivity)
- Special co-processors (e.g. crypto acceleration hardware)
- Tamper resistant components (e.g. for secure key storage)

4.4.1 Drivers

Some hardware components might be shared among multiple guest operating systems. Therefore, simultaneous access and access rights have to be managed. The proposed mechanism to do so is to bind hardware drivers to a secure IO partition, which is capable of offering virtual drivers to the other guest partitions. This shall be implemented by the inter partition communication mechanisms defined by the virtualisation layer.

5 Security and Dependability Requirements

Considering OVERSEE as the "Open Vehicular **SECURE** Platform", it is very important to carefully look at the functional security in terms of safety and dependability, as well as the information security in terms of IT-security and communications security.

This chapter will outline the different types of security objectives for the system under development. Best practices for the design of the platform will be highlighted and the severity of failures sketched.

5.1 Objectives

OVERSEE will be designed to cover the following objectives:

- Safety: Functional correctness of the system behaviour
- Security: Protection against unauthorized information change or extraction
- Protection: Protection against unauthorized access to system resources
- Privacy: Protection of individual private data
- Reliability: Enduring functional correctness
- Dependability: Combination of safety and reliability

However, there will be restrictions on the coverage of safety, outlined in section 5.1.2.

5.1.1 Security

The obejctives of Security, Protection and Privacy are summarized as security aspects. They are all about preventing misuse. The summary below outlines what issues are covered.

The main assets from the security perspective are:

- Confidentiality
- Integrity
- Authenticity

5.1.2 Safety

The objectives of Safety, Reliability and Dependability are taken together as safety aspects. The OVERSEE consortium agreed on excluding safety in the specification of the OVERSEE platform. Therefore, we will not go into detail on safety, as well on dependability. The reason for this is explained below.

While an automotive platform most obviously also has some safety related demands on it - notably when interacting with system busses in a vehicle, the prime focus of the OVERSEE platform as described in the DOW is security. Nevertheless, it was the intention of the consortium to take safety issues into consideration with respect to not making the results

unusable in the context of safety related systems. With the given time frame and resources, it is though not realistic to provide a platform with stringent security capabilities while fully adapting a suitable safety process along side. In this respect, we might add that the issue of merging the sometimes-conflicting requirements of safety and security are not even resolved at the research level in part due to a largely disconnected safety and security community.

On the technical side, it is to be noted that while security properties can be treated in isolation in automotive systems due to the current systems not communicating with the outside world directly (ignoring a few cases where this is not the case like the wheel pressure sensors, and known bus related vulnerabilities, i.e. ODB), safety is an inherently system level property and it would not be realistic to interconnect a research platform like OVERSEE to safety related internal busses to directly control safety functions in a car. That though is not stating that OVERSEE is not extendible in the future to accommodate these capabilities. Basically the architecture of OVERSEE, as noted above, did take safety needs into consideration in the high-level design (i.e. utilization of a separation kernel, centralized configuration of resources, restricted inter-partition communication semantics allowing to achieve certification in principle). To verify the feasibility of such an extension based on the current architecture design, a mapping to the predominant automotive architecture standards, OSEK/VDX and AUTOSAR were conducted in the early design phase of OVERSEE, and no discrepancies at the architectural level were discovered. Further to allow development of safety related applications in a suitable restricted environment, allowing to verify properties at the formal level, as well as ease migration of automotive applications to the OVERSEE platform, a OSEK partition was added to the OVERSEE effort, with the intent to provide a minimum OSEK RTE on top of the separation kernel. With these components in place extending OVERSEE to the arena of functional safety in the context of IEC 26262 seems doable with minor amendments to the current architecture, though serious investment in the tooling and procedural side would seem mandatory.

The de-scoping of safety as a target of OVERSEE was a decision by the consortium in the first Bochum meeting (March 2010) due to the unanimous conclusion that the resources available would not allow to produce results in the context of OVERSEE that are suitable for certification against the upcoming IEC 26262 (Note that IEC 26262 was not yet formally released as of March 2010). This is in part due to the standard not yet being fully adopted in Industry and no suitable (at least not published) strategies being available how to tackle certification of pre-existing components in the context of this functional safety standard.

5.2 Target of evaluation

To do a security analysis on OVERSEE prior its design is impossible. What we however can achieve is, to give the right directions, to create a secure by design system. The vulnerable parts of OVERSEE are:

- External Interfaces
    - Communications interfaces
- In-car system
    - Physical access to OVERSEE (hardware)

- OVERSEE platform
    - Logical access to OVERSEE (software)

While the first item, external communication interfaces is fully covered by the TVRA of the ETSI TC ITS security working group [11] and the security of hardware access is out of scope of this specification. The biggest issue remains the logical security of OVERSEE.

The OVERSEE platform shall offer secure communication channels between partitions and at the same time, ensure the enforcement of predefined policies.

- Protect and isolate partitions against each other
- Limit partition capabilities according to a predefined policy
- Offer secure (confidential, integer and authentic) communication between partitions

5.3 Severity of security breaches and dependability loss

The design of the OVERSEE system shall include reasonable security functionalities. Dependability is de-scoped, c.f. section 5.1.2. However, it is of cause possible, that the security concept is broken afterwards. The impact of any breach can range from serious, enabling attackers to overtake the whole system and maybe connected devices to rather low, when only parts of the system get affected.

The selection of countermeasures for possible vulnerabilities requires therefore an evaluation of the severity. OVERSEE must be designed according to [11] and [15], to guarantee the validity of the corresponding analysis.

6 Next Steps

Task 1.2 ends with the finalization of this deliverable. The OVERSEE project will now enter the design phase. Based on the results of the first work package, the platform will be planned. The second work package consists of the following tasks:

- Specification of interfaces
- Design of the information flow
- Specification of required security services
- Definition of internal building blocks
- Design of secure communication
- Technical Requirements validation support

Along with the second work package, the first task of the third work package will start.

- Selection for reuse of existing building blocks

Annex A: List of collected requirements

Tabular Summaries

The following Table 2 summarises from where the requirements originate. It shows only the use case claimed requirements. Use Cases may also induce more requirements than shown in the table.

	UC-TOLL	UC-eC	UC-PLR	UC-SCoND	UC-LDW	UC-EVSP	UC-SVT	UC-TIE	UC-ABR	UC-PTC	UC-CF	UC-ELP	UC-DTM	UC-BDC	UC-RCC	UC-PAYD	UC-IA	UC-VRE	UC-PSS	UC-ELB	UC-WEB
Connectivity Requirements																					
INET			x				x	x			x		x	x	x					x	
PS	x	x	x		x	x		x			x		x	x						x	
MIC		x																			
Speaker		x																			
TIME		x																			
BUS-READ								x	x	x						x	x				x
BUS-SEND								x	x	x							x				x
PAN				x																	
CEN-DSRC	x																				
GWN	x																				
DEVICE				x																	
ITS5.9GHz					x	x															
USR-IDENT		x	x				x			x	x	x			x	x		x		x	x
API-Method Requirements																					
MILEAGE																				x	
HMI	x	x	x	x	x	x										x				x	
SIGN.1	x				x	x										x				x	
ExpFile																				x	
SPEED																				x	
Airbag		x																			
VERIFY.1	x				x	x															

	UC-TOLL	UC-eC	UC-PLR	UC-SCoND	UC-LDW	UC-EVSP	UC-SVT	UC-TIE	UC-ABR	UC-PTC	UC-CF	UC-ELP	UC-DTM	UC-BDC	UC-RCC	UC-PAYD	UC-IA	UC-VRE	UC-PSS	UC-ELB	UC-WEB
ACTUATOR															x						
PART-Comm													x								
CAR-AUTH.1														x							
SERVICE-AUTH.1														x							
Configuration Requirements																					
InactiveGWN		x																			
MODE-STOLEN							x														
Communication Requirements																					
ISO-TP														x							
UDS														x							
USR-AUTH		x	x				x			x	x	x			x	x		x		x	x
Security Requirements																					
BUS-S-SEND					x			x	x	x							x				x
S-INET							x				x					x					
S-PAN				x											x						
DEV-AUTH				x																	
CAR-AUTH.2														x							
SERVICE-AUTH.2														x							
BUS-RESTRICT																	x				
Virtualisation Requirements																					
Partitioning Requirements																					

Table 2: Use Case Requirements

Connectivity Requirements

Requirement *INET*

ID	Type	Category
1	Functional	Connectivity

Brief Summary

An Internet connection is necessary.

Fit Criterion

Check if any remote host is reachable, e.g., ping oversee-project.com.

Rationale

- The use case requires a data connection to a backend service provider, in order to work. (UC-PLR)
- Variants of the use case may need the facility to transmit the current logbook via an Internet connection. (UC-ELB)
- A (secure) Internet connection is required in order to connect remotely (outside the car) to the car with a mobile device. (UC-RCC, UC-CF, UC-SVT)
- The use case requires a data connection to a backend service provider, in order to transmit traffic information. (UC-TIE, UC-DTM)
- Facility to reach the BDC service. (UC-BDC)

Requirement *PS*

ID	Type	Category
10	Functional	Connectivity

Brief Summary

OVERSEE must connect to an indoor and/or outdoor positioning system. For indoor positioning infrared or camera based systems shall be used. In outdoor environments GPS shall be used.

Fit Criterion

Get positioning data (e.g., from GPS) and compare with another positioning device (e.g., GPS).

Rationale

- Positioning is required to find free parking lots near to the vehicle's location. (optional) (UC-PLR)
- Variants of the use case may need to store the geographical position together with the mileage to fulfil legal requirements. (UC-ELB)

- For some variants of the use case it is maybe necessary to gather the geographical position of the vehicle to determine if the vehicle drives along a road for which tolling is required. (UC-TOLL)
- The position is part of the MSD in eCall furthermore the position is needed to determine the direction of the vehicle prior to the accident. (UC-eC)
- The current position of the vehicle is needed to determine the proper reaction on a received hazard warning or to send out a hazard warning including the current position of the vehicle. (UC-LDW1, UC-EVSP, UC-EVSP2)
- The actual position is relevant for traffic information. (UC-TIE, UC-DTM)
- The actual position is relevant for car location. (UC-CF)
- Facility to combine events and vehicle's position to send out roadside assistance or to perform statistical analysis. (UC-BDC)
- Model cars, which will be used to demonstrate long-term use-cases, require indoor positioning. (Model Cars)

Requirement *MIC*

ID	Type	Category
18	Functional	Connectivity

Brief Summary

It is required to connect a microphone to receive voice of driver/occupant.

Fit Criterion

Recording the voice of driver and/or occupants possible.

Rationale

- Because of the requirement of in-band transmission of the MSD in eCall it is necessary to handle also the voice connection to the driver within the platform. (UC-eC)

Requirement *Speaker*

ID	Type	Category
18.2	Functional	Connectivity

Brief Summary

It is required to connect speakers for sound output.

Fit Criterion

Playback of sound possible.

Rationale

- Because of the requirement of in-band transmission of the MSD in eCall it is necessary to handle also the voice connection to the driver within the platform. (UC-eC)

Requirement *TIME*

ID	Type	Category
19	Functional	Connectivity

Brief Summary

Gather local time from real time clock.

Fit Criterion

Gather local time and compare with another device providing the local time.

Rationale

- The local time is part of the MSD in eCall. (UC-eC)

Requirement *BUS-READ*

ID	Type	Category
20	Functional	Connectivity

Brief Summary

Read access to internal bus communication via OEM gateway.

Fit Criterion

Read left mirror setting.

Rationale

- Having the capability to read (certain) information from internal vehicle buses such as CAN to get certain in-vehicle information such as driving parameters or settings. Note there is an additional gateway/firewall - out-of-scope of OVERSEE and hence safely/securely managed by the OEM - that controls which internal bus communication can be read. (UC-TIE, UC-ABR, UC-PTC, UC-PAYD, UC-IA, UC-WEB)

Requirement *BUS-SEND*

ID	Type	Category
21	Functional	Connectivity

Brief Summary

Write access to internal bus communication via OEM gateway.

Fit Criterion

Send message to change left mirror setting.

Rationale

- Having the capability to write/send (certain) information to the internal vehicle buses such as CAN to set certain in-vehicle parameters. Note there is an additional gateway/firewall - out-of-scope of OVERSEE and hence safely/securely managed by the OEM - that controls which internal bus communication can be effectively forwarded to the internal bus communication system. (UC-TIE, UC-ABR, UC-PTC, UC-IA, UC-WEB)

Requirement *PAN*

ID	Type	Category
26	Functional	Connectivity

Brief Summary

The capability to communicate with user devices via personal area networks in a wireless, bi-directional manner is required.

Fit Criterion

Connect user device via Bluetooth or Wi-Fi connection.

Rationale

- Allow nomadic devices to connect via Bluetooth or Wi-Fi. (UC-SCoND)

Requirement *CEN-DSRC*

ID	Type	Category
36	Functional	Connectivity

Brief Summary

DSRC Communication according to CEN TC278.

Fit Criterion

Receive and Transmit of a message via CEN DSRC.

Rationale

- For purposes of toll collection CEN standardized DSRC Communication within CEN TC278. (UC-TOLL)

Requirement *GWN*

ID	Type	Category
37	Functional	Connectivity

Brief Summary

OVERSEE shall enable connections via GWNs (e.g., UMTS, GSM/GPRS).

Fit Criterion

Send and receive data via a GWN connection.

Rationale

- Transmission of data (e.g., summary of trips) for accounting purposes and update of e.g., configuration settings and maps for the tolling application. (UC-TOLL)

Requirement *DEVICE*

ID	Type	Category
39.1	Functional	Connectivity

Brief Summary

It must be possible to register nomadic devices to OVERSEE.

Fit Criterion

Try to connect unregistered device (should fail). Register device and try again (should work).

Rationale

- The use case requires registering and authenticating nomadic devices. (UC-SCoND)

Requirement *ITS5.9GHz*

ID	Type	Category
32	Functional	Connectivity

Brief Summary

Capability to receive and transmit messages in the frequency band for Intelligent Transport Systems. This could be substituted for demonstration purposes with 802.11a.

Fit Criterion

Receive and transmit DEN messages according to the ETSI standards.

Rationale

- Local hazard warnings will be distributed by DEN (decentralized environmental notification) messages hence it is necessary to receive, transmit and process DEN messages. (UC-LDW, UC-EVSP1, UC-EVSP2)

Requirement *USR-IDENT*

ID	Type	Category
25.1	Functional	Connectivity

Brief Summary

Having the capability to uniquely identify vehicle driver (and passengers) by reliable identification interface (e.g. token reader, biometric scanner, user input).

Fit Criterion

To identify current driver (and passengers) and related him (them) to a local profile (if existing), OVERSEE requires a reliable identification interface based for instance on (wireless) token recognition, biometry or other user inputs.

Rationale

- Driver (and passenger) identification is required to assign individual authorizations & resources, individual settings etc. and/or to relate individual actions and responsibilities. Moreover, driver (and passenger) identification is mandatory prerequisite for driver (and passenger) authentication. (In fact all, but at least UC-TOLL, UC-eC, UC-PLR, UC-SVT, UC-PTC, UC-CF, UC-ELP, UC-RCC, UC-PAYD, UC-VRE, UC-ELB, UC-WEB)

API-Method Requirements

Requirement *MILEAGE*

ID	Type	Category
3	Functional	API-Method

Brief Summary

A facility to read current vehicle mileage is required.

Fit Criterion

Read current mileage from vehicle.

Rationale

- The use case requires access to the current mileage of the vehicle since this value has to be stored on start and end of a trip. (UC-ELB)

Requirement *HMI*

ID	Type	Category
5	Functional	API-Method

Brief Summary

Capability to show the driver an input form, to receive additional information.

Fit Criterion

Show form to driver and receive input from the driver.

Rationale

- The use case needs the capability to receive additional information concerning the trips for legal reasons. (UC-ELB)
- Drivers need the capability to specify the destination of their trip (or another location) where a parking lot should be reserved, as well as additional information (vehicle size, handicapped driver, etc.). (UC-PLR)
- One of the legal eCall requirements is providing information on the processing of eCall to the driver. Furthermore the driver must be able to activate eCall manually (e.g., on personal disease without accident). (UC-eC)
- Show the current and archived billing information records as well as the current rates for PAYD or eToll to the driver. (UC-PAYD, UC-TOLL)
- Show hazard warning to the driver. (UC-LDW, UC-EVSP1, UC-EVSP2)
- Nomadic devices need to be registered (and authorized) by the user using the HMI. (UC-SCoND)

Requirement *SIGN.1*

ID	Type	Category
8.1	Functional	API-Method

Brief Summary

Generate signatures for data and messages.

Fit Criterion

Generate and verify a signature.

Rationale

- The logbook must be signed with the platforms private key to prove authenticity and integrity for legal requirements. (UC-ELB)
- The billing information for the PAYD insurance rate or the PAYD tax must be signed with the platforms private key to prove authenticity and integrity. (UC-PAYD)
- The billing information for eToll must be signed to prove authenticity and integrity. Furthermore the communication with enforcement vehicles and road side units have to be authentic and of integrity. (UC-TOLL)
- Hazard warning messages must be signed to verify the authenticity of the sending vehicle and avoid fake messages. (UC-LDW, UC-EVSP1, UC-EVSP2)

Requirement *ExpFile*

ID	Type	Category
12	Functional	API-Method

Brief Summary

It shall be possible to store files to nomadic devices.

Fit Criterion

Write File on a nomadic device.

Rationale

- Variants of the use case may need the facility to store/export the ELB on a nomadic device. (UC-ELB)

Requirement *SPEED*

ID	Type	Category
13	Functional	API-Method

Brief Summary

Get current vehicle speed.

Fit Criterion

Read current vehicle speed and compare it with a reference.

Rationale

- Input Forms should only be shown while the vehicle is stopped, therefore the vehicle velocity is necessary to know. (UC-ELB)

Requirement *Airbag*

ID	Type	Category
16	Functional	API-Method

Brief Summary

Read Airbag Status.

Fit Criterion

Read Airbag Status.

Rationale

- ECall should be triggered automatically; hence it is necessary to read the status of the airbag to trigger automatic eCall. (UC-eC)

Requirement *VERIFY.1*

ID	Type	Category
33.1	Functional	API-Method

Brief Summary

The capability to verify signatures of messages and data is required.

Fit Criterion

Generate and verify a signature.

Rationale

- Hazard warning messages must be signed to verify the authenticity of the sending vehicle and avoid fake messages. (UC-LDW, UC-EVSP1, UC-EVSP2)
- Communication with enforcement vehicles and road side units has to be authentic and of integrity therefore verification of signatures are needed. (UC-TOLL)

Requirement *ACTUATOR*

ID	Type	Category
38	Functional	API-Method

Brief Summary

The capability of controlling actively (non-safety relevant) actuators of the car (e.g. actuators for closing and opening of windows, doors, or similar units) is required.

Fit Criterion

Trigger actuator to open front left window and have an optical check that the window is open.

Rationale

- The use case requires access to actuators in order to control remotely functions of the car via mobile devices. (UC-RCC)

Requirement *PART-Comm*

ID	Type	Category
41	Functional	API-Method

Brief Summary

Send requests or information to other applications or partitions. Receive requests or information from other applications or partitions.

Fit Criterion

Partitions can send a request to other partitions and receive the answer to the request.

Rationale

- Different applications (partitions) can need to exchange information. (UC-DTM)

Requirement *CAR-AUTH.1*

ID	Type	Category
44.1	Functional	API-Method

Brief Summary

The vehicle is able to prove its identity in a privacy preserving way.

Fit Criterion

Authenticate Vehicle.

Rationale

- The BDC service needs to verify and authenticate the vehicle. Implies secure memory for vehicle's private key. (UC-BDC)

Requirement *SERVICE-AUTH.1*

ID	Type	Category
45.1	Functional	API-Method

Brief Summary

The authentication of the Break Down Call (BDC) Service shall be verifiable.

Fit Criterion

The service is able to prove its identity.

Rationale

- The vehicle needs to verify and authenticate the BDC service, e.g. signed by an OEM. Implies secure memory for OEM's public key. (UC-BDC)

Requirement *Platform-Int1 Service access*

ID	Type	Category
124	Functional	API-Method

Brief Summary

The OVERSEE platform will provide its services through a well-specified API. This mechanism is called „hypercall". Hypercall are equivalent to system calls in an operating system.

Fit Criterion

The API library will detail the services provided by the platform.

Rationale

- Partitions need to use the platform services to access to the virtual resources or as support for internal operations. The API will provide the services. (All)

Configuration Requirements

Requirement *InactiveGWN*

ID	Type	Category
15	Functional	Configuration

Brief Summary

GWN module should provide a mode where the GWN module is not registered within the GWN.

Fit Criterion

Activate and Inactivate GWN module.

Rationale

- If the GWN is only used for eCall it is necessary that the GWN module could be disconnected from the GWN and only registers to the GWN if necessary (while still receiving GWN status information). This is a privacy requirement within the eCall specification. (UC-eC)

Requirement *MODE-STOLEN*

ID	Type	Category
43	Functional	Configuration

Brief Summary

The car emits messages to authorised services to inform about its status, if it has detected, that it has been stolen.

Fit Criterion

Trigger the stolen vehicle sensor and test messages that are sent.

Rationale

- The car is able to detect its situation and to inform authorised services. (UC-SVT)

Communication Requirements

Requirement *ISO-TP*

ID	Type	Category
46	Functional	Communication

Brief Summary

Transport segmented data over vehicle's busses is required.

Fit Criterion

Communication over ISO-TP is possible (ISO 15765-2 compliant).

Rationale

- ISO-TP offers the facility to transmit up to 4095 Bytes at one time between two ECUs. This protocol is needed by higher-level communication services. (UC-BDC)

Requirement *UDS*

ID	Type	Category
47	Functional	Communication

Brief Summary

Unified diagnostic service according to ISO 14229-1 is required.

Fit Criterion

OVERSEE is able to read out the instrument cluster's event memory (ISO 14229-1 compliant).

Rationale

- UDS offers the facility to read out the ECU's event memory. (UC-BDC)

Requirement *USR-AUTH*

ID	Type	Category
25.2	Protection	Communication

Brief Summary

Having the capability to securely (e.g., as shown by an adequate security analysis) authenticate vehicle driver (and passengers).

Fit Criterion

To authenticate current driver (and passengers), i.e. to verify the claimed identity (cf. USR-IDENT), OVERSEE requires a secure authentication interface together with secure authentication mechanisms (e.g., authentication protocols).

Rationale

- Driver (and passenger) authentication is required to verify that the individual authorizations & resources, individual settings, actions, responsibilities are related only to the real / genuine person or role they are intended for. (In fact all, but at least UC-TOLL, UC-eC, UC-PLR, UC-SVT, UC-PTC, UC-CF, UC-ELP, UC-RCC, UC-PAYD, UC-VRE, UC-ELB, UC-WEB)

Requirement *Platform-Comm1 Interpartition communication*

ID	Type	Category
115	Functional	Communication

Brief Summary

The OVERSEE platform shall provide robust and secure mechanisms to communicate partitions.

Fit Criterion

The inter-partition communication offers the possibility to exchange partition information.

Rationale

- The platform will implement the mechanisms to guarantee the communication between partitions in a secure and robust way. The communication involves a single source to one or more destinations. (All)

Requirement *Platform-Comm2 Interpartition communication*

ID	Type	Category
116	Functional	Communication

Brief Summary

The inter-partition communication will be performed through ports. A port is a service through a partition can read or write messages from or to other partitions.

Fit Criterion

The partitions can see ports to read or write information.

Rationale

- Interpartition communication is conducted via messages. A message is defined as a continuous block of data of finite length. A message is sent from a single source to one or more destinations. The destination of a message is a partition, and not a thread within a partition. (All)

Requirement *Platform-Comm3 Interpartition communication*

ID	Type	Category
117	Functional	Communication

Brief Summary

The OVERSEE platform shall provide robust and secure channels to link partition ports.

Fit Criterion

Channels are implemented by the hypervisor in a secure way.

Rationale

- The basic mechanism for linking partitions by messages is the channel. A channel defines a logical link between one source and one or more destinations, where the source and the destinations may be one or more partitions. It also specifies the mode of transfer of messages from the source to the destinations together with the characteristics of the messages that are to be sent from that source to those destinations. (All)

Requirement *Platform-Comm4 Interpartition communication*

ID	Type	Category
118	Functional	Communication

Brief Summary

The OVERSEE platform shall provide channels with two transfer modes: sampling and queuing.

Fit Criterion

Channels can permit the broadcast (sampling) or the point-to-point connection (queuing).

Rationale

- Partition can exchange messages considering the nature of the information exchanged. (All)

Requirement *Platform-Comm5 Interpartition communication*

ID	Type	Category
118.1	Functional	Communication

Brief Summary

The OVERSEE platform shall provide sampling port communication.

Fit Criterion

A source partition can sent messages that can be read by several partitions.

Rationale

- In the sampling mode, successive messages typically carry identical but updated data. No queuing is performed in this mode. A message remains in the source port until it is transmitted via the channel or it is overwritten by a new occurrence of the message, whichever occurs first. This allows the source partition to send messages at any time. Each new instance of a message overwrites the current message when it reaches a destination port, and remains there until it is overwritten. This allows the destination partitions to access the latest message. (All)

Requirement *Platform-Comm6 Interpartition communication*

ID	Type	Category
118.2	Functional	Communication

Brief Summary

The OVERSEE platform shall provide queuing port communication.

Fit Criterion

A source partition can sent messages that can be read by one partition in a buffered way.

Rationale

- In the queuing mode, each new instance of a message may carry uniquely different data and therefore is not allowed to overwrite previous ones during the transfer. No message should be unintentionally lost in the queuing mode. The ports of a channel operating in queuing mode are allowed to buffer multiple messages in message queues. A message sent by the source partition is stored in the message queue of the source port until it is transmitted via the channel. When the message reaches the destination port, it is stored in a message queue until it is received by the destination partition. (All)

Security Requirements

Requirement *BUS-S-SEND*

ID	Type	Category
23	Protection	Security

Brief Summary

Secure write access to internal bus communication via OEM gateway.

Fit Criterion

Enforce confidentiality of message to vehicle logbook.

Rationale

- Having the capability to enforce protection of information to internal vehicle buses such as CAN regarding authenticity, integrity, freshness or confidentiality. Note there is an additional gateway/firewall - out-of-scope of OVERSEE and hence safely/securely managed by the OEM - that controls which internal bus communication can be written at all. (UC-TIE, UC-ABR, UC-PTC, UC-IA, UC-WEB, UC-LDW)

Requirement *S-INET*

ID	Type	Category
24	Protection	Security

Brief Summary

There should be an option to have secure Internet connections.

Fit Criterion

Open a secure connection to a remote host, e.g., ssh oversee-project.com.

Rationale

- Having the capability to securely (e.g., via SSL/TLS) communicate with (external) Internet servers in a bi-directional manner. (UC-PAYD, UC-CF, UC-SVT)

Requirement *S-PAN*

ID	Type	Category
27	Protection	Security

Brief Summary

Communication with user devices via personal area networks shall be secure in order to have the capability to enforce protection of information (regarding authenticity, integrity, freshness, or confidentiality).

Fit Criterion

Connect user device via encrypted Bluetooth connection.

Rationale

- Secure integration of nomadic devices requires secure near field communication. (UC-SCoND)
- Secure near field communication is required for remote car control, in order to connect mobile devices from inside the car. (UC-RCC)

Requirement *DEV-AUTH*

ID	Type	Category
39.2	Protection	Security

Brief Summary

It is required that all external devices must authenticate themselves against the OVERSEE platform.

Fit Criterion

Try to connect unregistered device (should fail). Register device and try again (should work).

Rationale

- The use case requires registering and authenticating nomadic devices. (UC-SCoND)

Requirement *CAR-AUTH.2*

ID	Type	Category
44.2	Protection	Security

Brief Summary

Private key material must be stored in a secure way within the vehicle.

Fit Criterion

Secure Memory must conform a later to be defined security standard.

Rationale

- The BDC service needs to verify and authenticate the vehicle. Implies secure memory for vehicle's private key. (UC-BDC)

Requirement *SERVICE-AUTH.2*

ID	Type	Category
45.2	Protection	Security

Brief Summary

Broken Down Call (BDC) Service must offer its authentication.

Fit Criterion

The service must prove its identity.

Rationale

- The vehicle needs to verify and authenticate the BDC service, e.g. signed by an OEM. Implies secure memory for OEM's public key. (UC-BDC)

Requirement *BUS-RESTRICT*

ID	Type	Category
49	Protection	Security

Brief Summary

Access to vehicle's signals has to be restricted.

Fit Criterion

An application running on OVERSEE do not get access to signals which should not be available to that application.

Rationale

- If OVERSEE offers a partition for 3rd party applications, bus access has to be restricted to ensure the application's behaviour and to avoid unauthorised bus read and bus write operations. (UC-IA)

Requirement *Platform-Sec1 Health monitor*

ID	Type	Category
119	Functional	Security

Brief Summary

When a fault is detected, the OVERSEE platform shall execute a health monitor module that will perform a predefined action. The occurrence of the fault will be logged or not according the system configuration.

Fit Criterion

Faults are handled by the hypervisor and, as a consequence of the fault, an action is executed.

Rationale

- The Health Monitor (HM) is the function of the O/S responsible for monitoring and reporting hardware, application and O/S software faults and failures. The HM helps to isolate faults and to prevent failures from propagating. Logs will permit to perform auditing activities. (All)

Requirement *Platform-Sec2 Health monitor*

ID	Type	Category
120	Functional	Security

Brief Summary

The actions to be executed by the OVERSEE platform as consequence of a fault shall be defined at system configuration time.

Fit Criterion

All actions performed by the HM shall be defined in the configuration time. If not defined, a default action specific to each fault shall be assumed.

Rationale

- In a robust and secure environment the actions to be taken when a fault is triggered have to be statically defined. (All)

Requirement *Platform-Sec3 Health monitor*

ID	Type	Category
120.1	Functional	Security

Brief Summary

The actions to be executed by the OVERSEE platform as consequence of a fault include: halt/restart the system, stop/suspend/restart partitions, change to a maintenance mode, ignore and propagate.

Fit Criterion

Depending on the default nature the appropriated action has to be executed when a failure is triggered.

Rationale

- These actions try to avoid chained faults in the system. (All)

Requirement *Platform-Sec6 Health monitor*

ID	Type	Category
121	Functional	Security

Brief Summary

When the fault is generated by a partition, one of the possible actions will be to propagate the fault to the partition.

Fit Criterion

Partition will have the possibility to manage internal errors (i.e. numeric errors).

Rationale

- When a fault is generated by a partition (i.e. numeric error), the platform will capture the fault and propagate it to the partition to be handled internally. (All)

Requirement *Platform-Sec7 Tracing*

ID	Type	Category
122	Functional	Security

Brief Summary

The OVERSEE platform shall provide a trace system to store relevant information of the system operation.

Fit Criterion

The platform requires being audited.

Rationale

- A trace system permits to store event generated by the system or logged as result of the HM actions. These events can be used to audit the system operation. (All)

Requirement *Platform-Sec8 Tracing*

ID	Type	Category
123	Functional	Security

Brief Summary

The OVERSEE platform will provide a specific trace system to each partition allowing storing relevant information of the partition operation.

Fit Criterion

Partitions require to be audited.

Rationale

- A trace system permits to store event generated by the partition. These events can be used to debugging purposes or to audit the partition operation. (All)

Virtualisation Requirements

Requirement *Platform-Virt1 Resource virtualisation*

ID	Type	Category
100	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall be able to paravirtualise the system resources providing a virtual machine to execute building blocks (partitions).

Fit Criterion

Partitions are executed in an independent way.

Rationale

- Several applications have to run in partitions accessing to the system resources (virtualised resources) in an independent way. (All)

Requirement *Platform-Virt2 Startup*

ID	Type	Category
101	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall be able to initialise the system resources and, then, to initiate the partition execution.

Fit Criterion

The system is initialised.

Rationale

- When booting, the system has to initialise the different virtualised resources in order to build the virtual machines. Once the virtualisation layer is initialised, it initialises the partitions. (All)

Requirement *Platform-Virt3 Temporal and spatial isolation*

ID	Type	Category
102	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall be able to execute partition in an isolated way guaranteeing the temporal independence and providing a complete spatial isolation to the partitions.

Fit Criterion

Partitions are executed in an independent way. A partition cannot access to the memory allocated to other partitions. A partition cannot be influenced by the way that other partitions use the system resources (CPU).

Rationale

- The execution of a partitioned system implies a strong temporal and spatial isolation of partitions. Possible failures in a partition are completely isolated and do not affect to others. (All)

Requirement *Platform-Virt4 Interrupt management*

ID	Type	Category
103	Functional	Virtualisation

Brief Summary

Errors shall be captured by the virtualisation layer and handled at the faulty level.

Fit Criterion

Check if errors are handled to maintain the secure state of the platform.

Rationale

- The OVERSEE platform shall manage all the interrupts, traps and exceptions that can be generated by the system resources and take the specified actions to maintain the system in a secure state. One of the actions will be to propagate the interrupt, trap or exception to a faulty partition. (All)

Requirement *Platform-Virt5 Device management*

ID	Type	Category
104	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall be able to allocate the peripheral management to a partition in exclusive mode.

Fit Criterion

Non virtualised devices are handled by partitions.

Rationale

- Devices shall not be managed by the virtualisation layer. These handlers can be allocated to a partition (IOServer) or an application partition. (All)

Requirement *Platform-Virt6 Processor mode*

ID	Type	Category
105	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall be executed in privilege processor mode and the partitions in user processor mode.

Fit Criterion

Partitions cannot directly access to the system resources.

Rationale

- It permits to the hypervisor be the unique component in the system to access to the system resources. Any attempt to access these resources by the partitions will generate a system trap which will be captured by the hypervisor. (All)

Requirement *Platform-Virt7 Clock management*

ID	Type	Category
106	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall provide two monotonic virtual clocks to the partitions. The virtual clocks can be absolute (real-time) or relative to the partition execution.

Fit Criterion

Partitions can manage the virtual clocks internally and take decisions based on any of these virtual clocks.

Rationale

- Partitions require to access to the real clock and the clock associated to its execution in order to know how the global and local time is increased. (All)

Requirement *Platform-Virt8 Clock management*

ID	Type	Category
107	Functional	Virtualisation

Brief Summary

The OVERSEE platform shall provide two timers to the partitions based on the two virtual clocks.

Fit Criterion

Partition can setup as many timers as needed based on these two timers.

Rationale

- Partitions need to take actions depending on the time elapsed in some operations. Take an action after a time, set a deadline for a specific action, etc. These time intervals can have as reference the global or local time. (All)

Partitioning Requirements

Requirement *Platform-Part1 Partitioning*

ID	Type	Category
108	Functional	Partitioning

Brief Summary

The OVERSEE platform shall permit to partitions labelled as „system partitions“ a set of rights to allow the capability to control other partitions.

Fit Criterion

„System partitions“ can manage other partitions.

Rationale

- „System partitions“ can control the system execution (partition security, robustness, etc.) and take action to start/stop/resume/reset/shutdown partitions. (All)

Requirement *Platform-Part2 CPU management*

ID	Type	Category
109	Functional	Partitioning

Brief Summary

The OVERSEE platform shall be able to allocate the CPU to the partitions in a robust way guaranteeing the temporal isolation. The scheduling policy can be cyclic or bandwidth servers.

Fit Criterion

Partitions are scheduled according to the scheduling policy.

Rationale

- In a partitioned system, the system concurrency is controlled by the virtualisation layer at partition level. The scheduling policy determines the time allocated to each partition. When partitions have real-time constraints, the scheduling policy has to satisfy the temporal requirements of the partitions. (All)

Requirement *Platform-Part3 CPU management*

ID	Type	Category
110	Functional	Partitioning

Brief Summary

The OVERSEE platform shall provide to the partitions the services to implement an operating system inside of the partition.

Fit Criterion

A partition that includes an operating system and its applications has to be executed in the OVERSEE platform.

Rationale

- The platform has to include the services to paravirtualise any operating system. These services include to enable/disable interrupts, support for threading, clock and timer management, memory management and low level services to support the thread context switch at partition level. (All)

Requirement *Platform-Part4 CPU management*

ID	Type	Category
111	Functional	Partitioning

Brief Summary

The OVERSEE platform shall provide the ability to a system partition to change dynamically the scheduling plan according to a set of predefined plans.

Fit Criterion

The platform can change the time allocation to the partitions according to external conditions.

Rationale

- The platform can support several execution modes of the system: normal, error, maintenance, or used defined. Every execution mode can require different time allocations to the partitions. (All)

Requirement *Platform-Part5 Memory management*

ID	Type	Category
112	Functional	Partitioning

Brief Summary

The OVERSEE platform shall allocate partitions in independent memory regions. These memory regions will be unreachable to other partitions.

Fit Criterion

Partitions have to be executed in a spatial isolated mode.

Rationale

- The platform has to provide spatial isolation to the partitions in order to guarantee the security and confidentiality of the partitions. Any attempt of a partition to access to a memory region that is allocated to other partition will capture by the platform and raised the appropriated exception. (All)

Requirement *Platform-Part6 Memory management*

ID	Type	Category
113	Functional	Partitioning

Brief Summary

The OVERSEE platform shall allocate memory regions to be shared by partitions. Partitions shall declare the shared regions and the access rights in the specification of the system.

Fit Criterion

Shared memory between partitions permits to communicate partitions through memory based schemes.

Rationale

- Partitions can interchange data using special memory regions that are shared among the partitions involved in the communication. Partitions sharing a memory region have to be specified in the configuration file. (All)

Requirement *Platform-Part7 Memory management*

ID	Type	Category
114	Functional	Partitioning

Brief Summary

The OVERSEE platform shall permit to system partitions the access to all memory regions to be able to manage partitions.

Fit Criterion

A system partition could perform some security actions on other partitions: perform a digest of the code, reinstall the code and load a new partition.

Rationale

- A specific service to access to all memory regions is provided in order to perform security actions on the system. Only „system partitions“ can use this service that break the spatial isolation. (All)

References

[1] OVERSEE Project: D1.1 Use Case Identification. 2010

[2] OVERSEE Project: D1.2 Initial Version of the Functional, Dependability and Security Requirements Document. 2010

[3] OVERSEE Project: D1.3 Initial version of the non-functional requirements and constraints document. 2010

[4] OVERSEE Project: D1.4 Functional Requirement Analysis. 2010

[5] OVERSEE Project: D1.5 Non-functional requirement analysis. 2010

[6] XtratumM, http://www.xtratum.org/

[7] OSEK/VDX, http://osek-vdx.org

[8] OSEK Operating System Specification 2.2.3, http://portal.osek-vdx.org/files/pdf/specs/os223.pdf

[9] OSEK/VDX Operating System Test Plan, http://portal.osek-vdx.org/files/pdf/modistarc/ostestplan20.pdf

[10] ETSI: TS 102 637-1 V1.1.1. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements. Sep. 2010

[11] ETSI: TR 102 893 V1.1.1. Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA). Mar. 2010

[12] ETSI: TR 102 638 V1.1.1. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions. Jun. 2009

[13] AUTOSAR Technical Specification, http://autosar.org/download/R3.1/AUTOSAR_TechnicalOverview.pdf

[14] AUTOSAR Software Requirements, http://autosar.org/download/R3.1/AUTOSAR_SRS_OS.pdf

[15] U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness Version 1.03