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 D1.5:
Non-Functional Requirements Analysis

Authors: Alfons Crespo (UPV)
Salvador Peiró (UPV)

Reviewers: Jan Holle (UniSiegen)
André Groll (UniSiegen)
Hakan Cankaya (escrypt)

Dissemination level: PU

Deliverable type: Report

Version: 1.6

Submission date: 01.07.2011
Abstract

This document is a revised version of the D1.3 “Non-functional requirements and constraints”. It contains an analysis of the non-functional requirements for the OVERSEE platform. The analysis performs a detailed description of the conditions and constraints associated to each requirement. The requirements are specified according to the vision of the OVERSEE consortium, to build the upcoming platform for vehicular applications.

This document is an extension of document D1.4 "Functional requirements analysis".
## Contents

Abstract.................................................................................................................................................. ii
Contents.................................................................................................................................................... iii
List of Figures........................................................................................................................................... vi
List of Abbreviations ............................................................................................................................... vii
Document History .................................................................................................................................. viii

1 Introduction.......................................................................................................................................... 1
   1.1 Definitions ....................................................................................................................................... 1
   1.2 Scope .............................................................................................................................................. 2
   1.3 Document Outline ......................................................................................................................... 3

2 Non-Functional Requirements ............................................................................................................ 4
   2.1.1 Requirement Sources and Selection ...................................................................................... 4
   2.1.2 Requirement Analysis ............................................................................................................. 5

3 OVERSEE Non-Functional Requirements: Use Case Requirements ...................................................... 6
   3.1 Time Requirements ....................................................................................................................... 6
   3.1.1 Scheduling impact ................................................................................................................... 6
   Requirement TRESP ........................................................................................................................... 8
   3.2 Communication Requirements .................................................................................................... 8
   Requirement MSDTransTime ............................................................................................................. 9
   Requirement DSRC-LAT .................................................................................................................... 9
   3.3 Resource Allocation Requirements ............................................................................................. 10
   3.3.1 Overhead of the Virtualization Layer .................................................................................... 10
   3.3.2 Overhead of the Guest Operating System ........................................................................... 10
   Requirement CRYPT-ACC ................................................................................................................ 11
   3.4 Platform Requirements ............................................................................................................... 11
   3.4.1 Booting Process ....................................................................................................................... 11
   Requirement FAST-BOOT ................................................................................................................. 12
   3.4.2 Data Validation ....................................................................................................................... 12
   Requirement DT-DL.2 ......................................................................................................................... 13
   3.5 Testing Requirements ................................................................................................................... 13
   Requirement V&V-PCO/COM ............................................................................................................ 13

4 OVERSEE Non-Functional Requirements: Platform Requirements .................................................. 15
3.1 Configurability & Flexibility ........................................................................................................... 15
   4.1.1 Virtualization Layer Configuration ......................................................................................... 16
       Requirement Platform-Conf0 ....................................................................................................... 16
   4.1.2 System configuration .............................................................................................................. 17
       Requirement Platform-Conf4 ....................................................................................................... 20
       Requirement Platform-Conf1 ....................................................................................................... 20
       Requirement Platform-Conf1.1 ................................................................................................... 21
       Requirement Platform-Conf1.2 ................................................................................................... 21
       Requirement Platform-Conf1.3 ................................................................................................... 22
       Requirement Platform-Conf1.4 ................................................................................................... 22
       Requirement Platform-Conf1.5 ................................................................................................... 23
       Requirement Platform-Conf1.6 ................................................................................................... 23
       Requirement Platform-Conf1.7 ................................................................................................... 24
       Requirement Platform-Conf5 ....................................................................................................... 24
   4.2 Deployment Requirements .......................................................................................................... 25
       4.2.1 Configuration Analysis and Validation .................................................................................. 25
           Requirement Platform-Dep1 ..................................................................................................... 26
       4.2.2 System generation .............................................................................................................. 27
           Requirement Platform-Dep2 ..................................................................................................... 27
       4.2.3 System loader .................................................................................................................... 28
           Requirement Platform-Dep3 ..................................................................................................... 28
   4.3 Performance Requirements ......................................................................................................... 29
       4.3.1 Predictable Services ........................................................................................................... 29
           Requirement Platform-Perf1 .................................................................................................... 29
       4.3.2 Metric for the Virtualization Layer ...................................................................................... 30
           Requirement Platform-Perf2 .................................................................................................... 30
       4.3.3 Virtualization Layer Overhead ........................................................................................... 31
           Requirement Platform-Perf3 .................................................................................................... 31
       4.3.4 Scalability Requirements .................................................................................................... 31
           Requirement Platform-Scal1 ..................................................................................................... 32
       4.3.5 Reliability Requirements .................................................................................................... 33
           Requirement Platform-Rell1 ..................................................................................................... 33
           Requirement Platform-Rell2 ..................................................................................................... 34
5  Summary and Next Steps .................................................................................................................. 35
List of Figures

Figure 1. Real time chain 6
Figure 2. Scheduling plan 7
Figure 3. Virtualization layer configuration 16
Figure 4. Configuration vector 26
Figure 5. System generation 27
List of Abbreviations

API      Application Programming Interface
APIC     Advanced Programmable Interrupt Controller
ARINC    Aeronautical Radio Incorporated
C2CCC    Car to Car Communication Consortium
CPU      Central Processing Unit
DENM     Decentralized Environmental Notification Message
DSRC     Dedicated Short-range Communications
DoW      Description of Work
ECDSA    Elliptic Curve Digital Signature Algorithm
ELF      Executable and Linking Format
FPU      Floating Point Unit
GPOS     General Purpose Operating System
GPS      Global Positioning System
HW       Hardware
ID       Identity
MAF      Major Time Frame
MSD      Minimum Set of Data
OS       Operating System
OSEK     Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
OVERSEE  Open Vehicular Secure Platform
PCO      Point of Control and Observation
PCS      Partition Context Switch
POSIX    Portable Operating System Interface (for Unix)
PSAP     Public Safety Answering Point
RTOS     Real-time Operating System
SKPP     Separation Kernel Protection Profile
SRS      Software Requirements Specification
SW       Software
UART     Universal Asynchronous Receiver-Transmitter
UC       Use Case
V2V      Vehicle to Vehicle
VGA      Video Graphics Adapter
XMCF     XtratuM Configuration File
XML      Extensible Markup Language
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>01.07.2011</td>
<td>Included comments. Final Version.</td>
</tr>
</tbody>
</table>
1 Introduction

The Open VEhiculaR SEcurE platform (OVERSEE) project as planned in DoW [1] has produced this deliverable, therefore it contains contributions from all partners.

The present document is the final version of the non-functional requirements and constraints document. It replaces the initial version of the non-functional requirements and constraints requirements document D1.3 [4].

Final versions of requirement documents D1.4[5] and D1.5 (this document) share definitions, approaches and criteria to select use cases and platform requirements. In order to avoid repetitions, the sections detailing these issues will be described in D1.4. In this sense, this document is an extension of D1.4 for the non-functional requirements and constraints.

1.1 Definitions

In this document, we use some concepts defined in the next list:

- **Boot-loader**: Software component that is executed after a reset and load the system to be executed in memory and jumps to its entry point.
- **Bounded service**: A service with a limited amount of code to be executed. The limitation refers to the capacity to execute a piece of code that the number of instructions can be analysed and it is constant or linear with some parameter. If the code contains loops, the limits of the loops are statically defined.
- **Communication channel**: Mechanism defined in ARINC-653 to perform the communication between partitions.
- **Configuration file**: File describing the static configuration of a system.
- **Configuration vector**: Binary representation of the configuration file once it has been analyzed and validated.
- **Container**: File system that contains a set of binary components.
- **Cyclic scheduling**: Scheduling policy based on a predefined table of time slots attached to executable entities (generally tasks or processes).
- **Guest OS**: Operating system running in a partition.
- **Health monitor**: Software component in charge of fault management.
- **Hypercall**: Mechanism provided by the virtualization layer to invoke a service.
- **Hypervisor**: Software layer that provides a virtual machine.
- **Overhead**: Additional CPU load occurring as consequence of the virtualization layer.
- **OVERSEE platform**: The architecture defined in the OVERSEE project.
- **Partition**: Executable entity in a partitioned system.
- **Partition context switch**: Time required to switch from one partition to the next one to be executed.
### Deliverable D1.5: Non-functional Requirements Analysis

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictable service</td>
<td>Service given by a real-time operating system or hypervisor whose worst case execution time can be calculated</td>
</tr>
<tr>
<td>Queuing channel</td>
<td>Channel with a buffer to store messages.</td>
</tr>
<tr>
<td>Sampling channel</td>
<td>Channel with a unique register to store a message. New messages overwrite previous ones.</td>
</tr>
<tr>
<td>System mode</td>
<td>Mode of operation of the system. I.e. Initialization, Normal, Maintenance, ...</td>
</tr>
<tr>
<td>System mode change</td>
<td>Mechanism that permits to switch from one mode to another.</td>
</tr>
<tr>
<td>Tool-chain</td>
<td>A set of software utilities used to perform an operation. For example in software development the tool chain will include: compiler, assembler, linker and debugger.</td>
</tr>
<tr>
<td>Virtualization layer</td>
<td>See hypervisor.</td>
</tr>
<tr>
<td>XML parser</td>
<td>Software that permits to analyze a XML file.</td>
</tr>
</tbody>
</table>

#### 1.2 Scope

The goal of this document is to analyze the non-functional requirements and other effective constraints such as corresponding organizational, legacy, or certification aspects (e.g., liability/responsibility aspects, potential dependability/security validation requirements, and organizational security requirements).

The current document performs the analysis of the non-functional requirements and constraints initially defined in D1.3 [4] and presents them together.

This document provides the non-functional requirements for the use cases defined in [1].

The scope and purpose is limited to non-functional requirements that include:

- Configurability and flexibility
- Performance
- Reliability and robustness
- Scalability
- Portability

Requirements are derived from two main sources:

- Use cases detailed in D1.1 [2]
- Platform requirements derived from previous requirements of XtratuM [7].

The description of these requirements, selection criteria and analysis method is detailed in D1.4 [5]. In order to avoid repetition of the same text, we omit these details in this document.

The intended audience of this document is the WP2 "Design of Platform" team of the OVERSEE project, to start the platform design. It is not the intention of the present document, to cover all possible requirements of the OVERSEE platform, rather it is meant to outline the requirements important to the design team.

Out of scope of the document, is the specification of the OVERSEE-demonstrator. The demonstration will show a possible OVERSEE implementation on real hardware and selected...
use cases. In the current task we concentrated on the platform itself, which means the implementation independent basic requirements of OVERSEE.

1.3 Document Outline

This document is organized in the following way: Section 2 presents a short introduction and scope of the non-functional requirements and constraints of the system. Section 3 analyses the requirements derived from the use cases. Section 4 details the requirements derived from the platform. Finally, section 5 summarizes the work carried out and presents the next steps.
Non-Functional Requirements

Non-functional requirements define the criteria that can be used to judge the operation of a system, in contrast to functional requirements that define specific behavior or functions.

Categories of non functional requirements can be:

- **Configurability & Flexibility**: Software is configurable if it has the ability to handle a wide variety of system configuration sizes. On the other hand, flexibility is applied when the software intends to increase or extend the functionality after its deployment.

- **Performance**: The performance constraints specify the timing characteristics of the software. Efficiency specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth, etc.

- **Reliability & Robustness**: Reliability specifies the capability of the software to maintain its performance over time. A robust system is able to handle error conditions gracefully, without failure. This includes a tolerance of invalid data, software defects, and unexpected operating conditions.

- **Availability**: A system’s availability or “uptime” is the amount of time that it is operational and available for use.

- **Portability**: Portability specifies the ease with which the software can be installed on all necessary platforms and the platforms on which it is expected to run.

- **Usability**: Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users.

- **Quality**: About procedures, test plan, validation and verification.

Security requirements can be considered as functional or non-functional depending on the importance of these functionalities. In this project, we consider the security as a nuclear issue and they will be considered in the functional requirement document.

Each of these categories are detailed below, with respect to the OVERSEE platform analyzing the implications and constraints.

### 2.1.1 Requirement Sources and Selection

As detailed in D1.4 [5] many requirements are derived from the relevant use cases derived in Task 1.1. Additionally, platform requirements are also derived from the virtualization layer or platform specification. These requirements were initially defined by CNES (Centre Nationale d’Etudes Spatiales, France) for XtratuM.

Furthermore the experiences and expertise of the OVERSEE partners have served to create a comprehensive set of requirements.

The selection criteria of use case and platform requirements has been detailed in D1.4 [5]. These criteria are applicable to functional and non-functional requirements. See D1.4 [5] for a detailed description of the criteria used to select the considered requirements.
2.1.2 Requirement Analysis

Whereas functional requirements are related to functional aspects that can be understood without a detailed description, non-functional requirements require, in general, an introduction and clear definition of the aspects to be around the issue to be specified. For instance, there are some issues that are impacted by the scheduling policy or the communication mechanisms. For this reason, we present the requirements in a different way as has been done in D1.4.

For each requirement, a set of data is collected and documented. The methodology used to present and analyze each requirement is:

- A presentation of the issues related to a specific category with the implications on other topics
- A list of requirements that specify the expected behavior related to this category
- For each requirement, the next information is provided:
  - ID: requirement identifier (unique numeric identifier)
  - Type: requirement type. Possible values are functional, protection and non-functional
  - Category: nature of the requirement
  - Requirement: requirement definition, following a more formal pattern
  - Brief summary: short informal textual explanation
  - Fit Criterion: basic criteria to be validated
  - Rational: justification of the requirement

This document is focused on the requirements of the following aspects:

- Time
- Communication
- Platform
- Configuration
- Deployment
- Performance
- Scalability
- Reliability
3 OVERSEE Non-Functional Requirements: Use Case Requirements

These requirements have been extracted from the use cases defined in document D1.1 [1]. They specify some non-functional requirements from the point of view of the applications.

3.1 Time Requirements

Time requirements deal with the need of bounded and known response time of the OVERSEE platform to external stimuli. This is characteristic of real-time systems that have to guarantee the response time.

In a partitioned system, the predictability impacts all the software layers. Therefore, applications with temporal constraints have to be executed in a partition with a real-time operating system. On the other hand the virtualization layer has to be also predictable.

Figure 1 shows the chain of real-time constraints to be considered by the different components.

If the external stimuli is generated by a sensor that is managed by a I/O partition, it also has to be included in the real-time chain.

In order to guarantee the response time, as required in requirement TRESP, of 500 milliseconds the scheduling policies of the hypervisor and partition operating system have to be considered. At hypervisor level, if the scheduling policy is cyclic, it has to be provided a sufficient periodic time window to allocate the partition resources to achieve the temporal constraint. Additionally, at partition level, the partition operating system has to guarantee the response time of the application by using the appropriated priority.

3.1.1 Scheduling impact

To perform a temporal analysis, it has to be considered that two partitions are involved in the data flow:
Deliverable D1.5: Non-functional Requirements Analysis

- **I/O partition**: should be executed in a periodic way (Period = $P_{I/O}$). Every $P_{I/O}$, it reads the sensor (or several sensors) and writes the values to the communication channels to be read by other partitions.

- **Application partition**: if this application has to take decisions considering the periodic values, it should be also periodic (same period).

The scheduling plan could be seen as detailed in figure 2.

So, the *worst case latency* between the sensor value is acquired by the I/O partition and it is processed by the application partition depends on:

1. The duration of the temporal slot of I/O partition
2. The duration of the temporal slots of “other partitions”
3. The duration of the temporal slot of application partition

The conclusion is that the response time depends on the predictability of the platform and the scheduling plan defined by the system integrator.
Deliverable D1.5: Non-functional Requirements Analysis

Requirement \textit{TRESP}

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Non-functional</td>
<td>Time</td>
</tr>
</tbody>
</table>

\textbf{Requirement}

The \textit{OVERSEE} platform shall provide bounded time response to the applications.

\textbf{Brief Summary}

Timed response time to the application needs depends on the different software layers from the application to the hardware. The bounded response time depends on the real-time operating system where the application is executed and the virtualization layer.

\textbf{Fit Criterion}

A periodic application on top of a real-time operating system in a partition will read a sensor. The period of the application will be 500 milliseconds. No sensor reads are missing.

\textbf{Rationale}

The use case requires that an incoming message from a sensor must be treated and answered in hard/weak real-time way, e.g., the response must be done in less than 500 milliseconds.

Note: 500 milliseconds can be considered as good practice period for periodic activities in the use cases. However, this value could be stressed to analyze the worst case and limit the periodic activities.

\subsection{3.2 Communication Requirements}

Communication requirements involve data exchange between partitions and time response constraints. From the time response point of view, the analysis is similar to the previous study on time requirements. It means that the CPU allocation to the architecture components have to be granted by the scheduling policies.

Additionally, there is a constraint about the communication between partitions involving the application and the services. The time required to send a message from a partition to another partition depends on the next factors:

- Length of the message. Large messages require more time to perform a copy of the message from the source partition to the destination. In order to guarantee the message delivery, messages are copied from the partition scope to the kernel scope and then to the destination partition scope.

- Communication type. Messages can be delivered under a synchronous or asynchronous mechanism. The scheduling policy of the hypervisor has a strong impact on the mechanism. Under a cyclic scheduling, the appropriated mechanism is the asynchronous.
Scheduling policy. In a partitioned system there exist two scheduling policies: global scheduler (at hypervisor level to schedule partitions) and local scheduler (at partition level to schedule applications). Each partition can have different local schedulers. With respect to the partition scheduling, there are several techniques to deal with the communication and time response constraints. One of these techniques is cyclic scheduling, which is mainly used in safety and hard real-time systems. This cyclic plan determines at which time windows the partitions will be executed. The system designer is responsible of the scheduling plan, according to the temporal requirements of the applications. The time distance between the execution of the source and the destination partitions imposes latency delay in the communication.

The time specified in this requirement has to be analyzed in the worst case situation.

**Requirement MSDTransTime**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Non-functional</td>
<td>Communication</td>
</tr>
</tbody>
</table>

*Requirement*

The OVERSEE platform shall guarantee a MSD transmission time lower than 4 seconds.

*Brief Summary*

Timed response time to the application needs depends on the different software layers from the application to the hardware. Time of MSD transmission has to be less than 4 seconds.

*Fit Criterion*

Measure time between connection to PSAP and finalized transmission of MSD under different circumstances.

*Rationale*

Transmission time for MSD in eCall must not be longer than 4 seconds.

**Requirement DSRC-LAT**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>Non-functional</td>
<td>Communication</td>
</tr>
</tbody>
</table>

*Requirement*

The OVERSEE platform shall provide bounded time response to the applications.

*Brief Summary*

Latency Time for DEN messages transmitted via ITS-G5

*Fit Criterion*

Measure Latency of DEN messages under different circumstances
### 3.3 Resource Allocation Requirements

The amount of processing of a partition strongly depends on the amount of time scheduled for this partition. The scheduling plan determines the time intervals where the CPU will be allocated to a partition. Some aspects have to be considered to determine the time allocated to a partition.

#### 3.3.1 Overhead of the Virtualization Layer.

The virtualization layer will execute the scheduling plan and allocate the resources to a partition, according the scheduling plan. The overhead of the virtualization layer is mainly due to the partition context switch (PCS). The effective time (ET) for a slot can be determined as:

\[
ET_i = SD_i \cdot PCS
\]

Where \(ET_i\) is the effective time for slot \(i\) that have a duration of \(SD_i\).

Considering the performance loss as the relation between the \(ET_i/SD_i\), it depends on the duration

\[
\frac{ET_i}{SD_i} = 1 - \frac{PCS}{SD_i}
\]

which means that shorter slot duration produces higher performance loss. The performance loss is modeled with the second term of the previous formula.

For explanation purposes, it can be assumed a PCS around 10 microseconds, and according to the previous formula, it is not the same if a partition is scheduled in 1 slot of 100 milliseconds, which can generates a performance lost of 0.01%, or 100 slots of 1 milliseconds, which generates a performance lost of 1.0%.

Note: the PCS strongly depends on the CPU type and frequency.

#### 3.3.2 Overhead of the Guest Operating System

An application running on a guest operating system (for instance Linux) is executed concurrently with other applications under the guest operating system scheduling policy. The effective execution time of an application depends on the scheduling policy and some application parameters (priority in most cases). The number and the priority of all applications (processes) determine the amount of resources allocated to an application.

In general, for the requirements specifying an amount of computation, the scheduling plan design is critical and can have a strong impact on the time required to execute it. In the case
of the CRYPT-ACC requirement, specifying an amount of cryptographic signatures per period, it should be

1. computed it in the same operating system without virtualization layer,
2. determined the amount of time required,
3. allocated the required time considering the performance loss due to the virtualization layer

**Requirement CRYPT-ACC**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>Non-functional</td>
<td>Communication</td>
</tr>
</tbody>
</table>

**Requirement**
The OVERSEE platform shall provide acceleration for cryptographic functions.

**Brief Summary**
Acceleration for cryptographic functions to enable secure V2V communication

**Fit Criterion**
OVERSEE is able to verify 200 ECDSA signatures per second.

**Rationale**
In worst case scenarios, Car-2-Car applications (e.g. active break) need to communicate with 200 fresh vehicles per second.

The system designer has to define a scheduling plan that guarantees to the cryptographic generator component sufficient resources per period to compute the needed signatures.

### 3.4 Platform Requirements

#### 3.4.1 Booting Process

Requirement FAST-BOOT is related to the booting process of the system and, consequently, the applications included on it. The booting process in a virtualized environment can be summarized in the next steps:

1. All software components are stored in a permanent device (flash memory) and ready to be loaded into memory. A boot loader shall copy all these binary components (virtualization layer, configuration vector and partitions) in memory according to the configuration vector specification. Then the boot loader shall give the control to the virtualization layer.
2. The virtualization layer initializes the internal data structures and performs the initial checks to guarantee the initial secure state of the system. Once this is achieved, the scheduling plan defined in mode 0 is executed.

3. This execution mode, shall execute the partitions as specified in the scheduling plan. In this mode, it is assumed that partitions will be initialized. If Linux partitions have been defined, the initialization of these partitions will be possible when the virtual disk service is started by a system partition in charge of the virtual resources (network, disk, console, etc.).

4. A system partition shall be in charge of detecting when all partitions have been initialized and then switch to an operational mode.

The booting time, strongly depends on the type of partition that has to be booted. If several Linux partitions are defined, this time could be large.

### Requirement FAST-BOOT

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Non-functional</td>
<td>Platform</td>
</tr>
</tbody>
</table>

**Requirement**

The OVERSEE platform shall provide the mechanisms for fast booting.

**Brief Summary**

An application has to be available after clamp 15 is switched on

**Fit Criterion**

An application running on OVERSEE starts signal processing within 500 ms after clamp 15 is switched on.

**Rationale**

Some applications, e.g. parking sensor system, have to provide their functionality immediately to the vehicle’s user when clamp 15 is switched on.

### 3.4.2 Data Validation

This requirement involves two aspects to be considered: the authentication process and the data validation.

The authentication process requires the use of internal or external services in order to download a chunk of data. Once the data is downloaded, it should be validated. Validated can be understood as an initial analysis in order to detect inconsistencies and incompatibilities.
Requirement **DT-DL.2**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.2</td>
<td>Non-functional</td>
<td>Platform</td>
</tr>
</tbody>
</table>

**Requirement**

The OVERSEE platform shall provide mechanisms to authenticate the source of incoming data and validate the data for integrity and authenticity.

**Brief Summary**

Central applications receiving critical data have to have the ability to authenticate the source of data and the authenticity and integrity of the data.

**Fit Criterion**

Source of downloaded software has to be authenticated and the data has to be validated.

**Rationale**

Some applications like the software download module receive security critical data. In such a case the server of source should be able to authenticate itself (e.g. with a certificate). The downloaded data package has to provide integrity with a secure hash process and authenticity with a signature. The key data used in the validation process has to be trusted. The usage of data (in our case the installation process) starts only after a successful validation.

3.5 **Testing Requirements**

Testing requirements impact on the design of the interfaces and the way they are reachable for authorized users.

The virtualization layer shall provide a mechanism to permit the partitions to store information about its state or significant events. This mechanism is the trace service provided by the virtualization layer that permits to a partition write “traces” that shall be accessed by an authorized partition in order to know the internal state of the partition or analyze a set of events that permit the partition observation. In partitioned systems, a specific partition shall be in charge of the global state of the partitions and will provide points of observation to perform a remote validation.

Requirement **V&V-PCO/COM**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>Non-functional</td>
<td>Testing</td>
</tr>
</tbody>
</table>

**Requirement**

The OVERSEE platform shall provide points of control and observation (PCOs) to perform a
validation.

Brief Summary
Platform provides points of control and observation (PCOs) on each communication interface.

Fit Criterion
An application is able to access to the PCOs and analyze this information

Rationale
Each communication interface in the OVERSEE platform have to be reachable for an external and authorized user to allow verification and validation. It means that some PCOs must been reachable for an authorized tester. This requirement will allow one to perform a wide range of functional validation
4 OVERSEE Non-Functional Requirements: Platform Requirements

Non-functional requirements define the criteria that will be used to judge the operation of a system, rather than specific behavior. This should be contrasted with functional requirements that define specific behavior or functions. As introduced in section 2, categories of non-functional requirements can be:

- Configurability & Flexibility: Software is configurable if it has the ability to handle a wide variety of system configuration sizes. On the other hand, flexibility is applied when the software intends to increase or extend the functionality after its deployment.
- Deployment specifies how the system is generated and executed
- Performance: The performance constraints specify the timing characteristics of the software.
- Efficiency specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth, etc.
- Reliability & Robustness: Reliability specifies the capability of the software to maintain its performance over time. A robust system is able to handle error conditions gracefully, without failure. This includes a tolerance of invalid data, software defects, and unexpected operating conditions.
- Availability: A system’s availability, or “uptime”, is the amount of time that it is operational and available for use.
- Portability: Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run.
- Quality: About procedures, test plan, validation and verification,

Security requirements can be considered as functional or non-functional depending on the importance of these functionalities. In this project, we consider the security as a nuclear issue and they will be considered in the functional requirement block.

Each of these categories is detailed below with respect to the OVERSEE platform analyzing the implications and constraints.

3.1 Configurability & Flexibility

The OVERSEE platform is an embedded system that has to be configured and deployed for each specific target. As target, it is understood an instantiation of the OVERSEE platform for a specific hardware configuration.

Configurability deals with the ability to define the configuration parameters to adapt the software to the specificities of the hardware and applications. Two levels of configurability have to be considered:
1. **Virtualization layer**: This configurability level shall permit to adapt the virtualization layer to the available hardware.

2. **System**: A configuration file will specify the resources to be used by A partitions in a specific deployment configuration.

   Flexibility can be considered as a property of the configuration. The configuration of the system has to be able to permit different system configurations in a flexible way.

### 4.1.1 Virtualization Layer Configuration

The configuration of the virtualization layer considers the following elements:

- **Processor**: in the OVERSEE platform the target processor is x86.
- **Stack size for the partitions**: This shall permit to define partitions with different OSs needs.
- **VGA support**
- **UART support**: Serial port
- **Support for partition accounting**: Provides the mechanisms to compute statistic information about the partition execution.

The compiled result of this configuration shall be a binary code of the virtualization layer to be used to deploy applications. Figure 3 illustrates this process.

![Figure 3. Virtualization layer configuration](image)

**Requirement** *Platform-Conf0*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>131</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

*Requirement*

*The virtualization layer shall be configured with a set of parameters to customize it for a specific target.*

**Brief Summary**
A set of parameters related to the platform will be used to obtain a compiled version (binary version) of the virtualization layer to be used for the deployment.

**Fit Criterion**
The virtualization layer shall be statically defined with a set of parameters depending on the used hardware. This configuration process will detail the processor, stack size and support for different devices.

**Rationale**
A configuration of the virtualization layer shall permit to adapt the virtualization layer to the underlying hardware in order to use the hardware capabilities and permit to adapt it to different guest OSs.

### 4.1.2 System configuration

Each embedded system deployment involves several partitions to be executed on top of the virtualization layer. The configuration file will detail how the systems resources are allocated.

The virtualization layer handles a set of resources that are specified in the configuration file. In this configuration file all resources and operations have to be explicitly defined.

Exported resources that can be configured are:

- **Processor.** Resources that are exported as a whole during a time interval.
  - CPU: There is only one CPU. It includes the internal registers (user and control registers). The scheduling plan identifies "which" partitions will use this resource and "when".
  - FPU: There is only one FPU. It is exported jointly with the CPU if the subject has specified its use in the XMCF.

- **Memory areas.** Memory is not exported as a whole. Memory areas are regions of memory that are directly exported as resource to subjects. Each subject is permitted to directly access to the memory areas that has specified in the
  - Memory layout. It defines the whole memory available in the system. All exported memory areas have to be independent (no overlapping) areas of the memory layout.
  - I/O memory area. It defines an I/O memory region as exported resource. These exported resources are assigned exclusively to a subject (no sharable).

- **Basic peripherals.** These devices (UART, VGA, etc.) are basic components of the system that are exported as resources. They require an explicit definition and subjects using this resource have to explicitly declare. The access to these devices is provided via hypercalls.

- **Inter-process communication.** It defines the inter-partition communication needs. It is based on channels and ports.

- **Time allocation.** It specifies when the resources are available for the partitions. It is defined by means of modes, plans and slots.
- **Traces.** are exported resources that permits to authorized subjects to register events (traces/audit records).

- **Health Monitor:** It specifies the actions to be executed as result of faults.

Listing 1 shows an example of a configuration file:

```xml
<SystemDescription xmlns="http://www.xtratum.org/xm-3.x" version="1.0.0" name="example">
  <HwDescription>
    <ProcessorTable>
      <Processor id="0" frequency="50Mhz">
        <CyclicPlanTable>
          <Plan id="0" majorFrame="25ms">
            <Slot id="0" start="0ms" duration="10ms" partitionId="0"/>
            <Slot id="1" start="15ms" duration="5ms" partitionId="1"/>
          </Plan>
          <Plan id="1" majorFrame="10ms">
            <Slot id="0" start="0ms" duration="5ms" partitionId="0"/>
            <Slot id="1" start="5ms" duration="5ms" partitionId="2"/>
          </Plan>
        </CyclicPlanTable>
      </Processor>
    </ProcessorTable>
    <MemoryLayout>
      <Region type="stream" start="0x40000000" size="4MB"/>
      <Region type="sdram" start="0x60000000" size="16MB"/>
    </MemoryLayout>
    <XMHypervisor console="Uart">
      <PhysicalMemoryAreas>
        <Area start="0x40000000" size="512KB"/>
      </PhysicalMemoryAreas>
    </XMHypervisor>
    <PartitionTable>
      <Partition id="0" name="Partition1" flags="system" console="Uart">
        <PhysicalMemoryAreas>
          <Area start="0x40100000" size="256KB"/>
          <Area start="0x40300000" size="128KB" flags="shared"/>
        </PhysicalMemoryAreas>
        <TemporalRequirements duration="25ms" period="10ms"/>
      </Partition>
      <PortTable>
        <Port name="writerQ" type="queueing" direction="source"/>
        <Port name="writerS" type="sampling" direction="source"/>
      </PortTable>
    </PartitionTable>
    <Trace device="Trace1"/>
  </HwDescription>
</SystemDescription>
```
<HealthMonitor>
  <Event action="XM_HM_AC_HALT" log="yes" name="XM_HM_EV_PARTITION_ERROR" />
</HealthMonitor>

<HwResources>
  <Interrupts lines="4" />
</HwResources>

<Partition id="1" name="Partition2" flags="fpu" console="Uart">
  <PhysicalMemoryAreas>
    <Area start="0x40180000" size="256KB" />
    <Area start="0x40300000" size="128KB" flags="shared" />
  </PhysicalMemoryAreas>
  <TemporalRequirements duration="25ms" period="5ms" />
  <PortTable>
    <Port name="readerQ" type="queuing" direction="destination" />
    <Port name="readerS" type="sampling" direction="destination" />
  </PortTable>
  <Trace device="Trace2"/>
</Partition>

<HwResources>
  <IoPorts>
    <Range base="0x80000080" noPorts="4" />
    <Range base="0x80100110" noPorts="15" />
    <Restricted address="0x80100200" mask="0x60" />
  </IoPorts>
  <Interrupts lines="/7" />
</HwResources>

<Partition id="2" name="Partition3" console="Uart">
  <PhysicalMemoryAreas>
    <Area start="0x40200000" size="256KB" />
  </PhysicalMemoryAreas>
  <TemporalRequirements duration="25ms" period="5ms" />
  <PortTable>
    <Port name="readerS" type="sampling" direction="destination" />
  </PortTable>
</Partition>
</PartitionTable>

<Channels>
  <QueuingChannel maxMessageLength="512B" maxNoMessages="10">
    <Source partitionId="0" portName="writerQ" />
    <Destination partitionId="1" portName="readerQ" />
  </QueuingChannel>
  <SamplingChannel maxMessageLength="512B">
    <Source partitionId="0" portName="writerS" />
  </SamplingChannel>
</Channels>
Listing 1. Example of a configuration file

```
<Devices>
  <MemoryBlock name="SystemTrace" start="0x40380000" size="128KB"/>
  <MemoryBlock name="Trace1" start="0x403C0000" size="64KB"/>
  <MemoryBlock name="Trace2" start="0x403E0000" size="64KB"/>
  <Uart id="0" baudRate="115200" name="Uart"/>
</Devices>
</SystemDescription>
```

### Requirement **Platform-Conf4**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

#### Requirement

The configuration file will be specified in a XML notation.

#### Brief Summary

The configuration file requires to be analyzed to evaluate the conformance with respect to the specified format. A schema is defined.

#### Fit Criterion

The configuration file can be parsed using a XML parser.

#### Rationale

XML notation provides facilities to be imported/exported and analyzed using standard tools.

### Requirement **Platform-Conf1**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

#### Requirement

The system parameters shall be allocated in a configuration file.

#### Brief Summary

This configuration file will detail the scheduling policy, memory areas, resources allocated to partitions, communication channels and devices

#### Fit Criterion
The virtualization layer will allocate, execute and permit the use of resources according to the configuration file.

**Rationale**
A complete description of the system resources will permit to achieve a trusted system.

### Requirement *Platform-Conf1.1*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.1</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**

The system shall be configured with a processor description specifying how the CPU is allocated to the partitions

**Brief Summary**

The CPU resource is allocated to the partitions concerning a cyclic scheduling plan, specified in the configuration table. The processor description details the modes and the concrete plan in each mode. A plan is specified as a set of slots detailing the temporal windows associated to each partition

**Fit Criterion**

The virtualization layer will execute, depending on the selected mode, the slot sequence. When the MAF is completed, the sequence is executed again.

**Rationale**

The plan specifies a fixed scheduling policy where partitions are executed in a cyclic way. It guarantees the temporal isolation of partitions.

### Requirement *Platform-Conf1.2*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.2</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**

The system shall configure a memory layout for the board. The memory layout corresponds to the available memory on the board where the system will be deployed.

**Brief Summary**

Each board can have several memory chips with different sizes and technology. The memory layout defines the memory available on each board which will be allocated to the virtualization layer and the partitions.

**Fit Criterion**

The memory layout defines the memory areas available on the board. All memory areas allocated to partitions shall be included in the memory layout definition.

**Rationale**

The memory layout of a board has to be coherent with the memory area allocation in the
Deliverable D1.5: Non-functional Requirements Analysis

**Requirement Platform-Conf1.3**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.3</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

*Requirement*

*The system shall be configured with a set of channels that permits communication between partitions.*

*Brief Summary*

Channels are links between ports which are the mechanisms used by the partitions to read or write information in the channels. A partition specifies a set of ports to communicate with other partitions. These ports shall be defined in the channel configuration section.

*Fit Criterion*

The virtualization layer will guarantee that a partition can use only the ports specified in the channel definition.

*Rationale*

Partitions will use only the ports specified under the constraints configured in the channel definition in the configuration file. The virtualization layer will grant the static configuration and will not permit any other communication among partitions.

---

**Requirement Platform-Conf1.4**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.4</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

*Requirement*

*The system shall allow to configure the channel type (sampling/queueing), maximum size of the message, number of messages to be stored (queueing channels) and a pair \(<\text{partition}, \text{port}>\) for each channel.*

*Brief Summary*

A channel specification consists of the type (sampling or queueing), maximum message size to be sent, number of messages to be queued in a queueing channel and the partition-port reference.

In sampling channels no queuing is performed. A message remains in the source port until it is transmitted by the channel or it is overwritten by a new occurrence of a message.

Queueing channels 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 by the channel.

*Fit Criterion*

The virtualization layer will guarantee that a partition can only use the ports specified in
the configuration parameters configured in the channel definition.

**Rationale**
Partitions will use only the ports specified in the constraints configured in the channel definition in the configuration file. The virtualization layer will grant the static configuration and will not permit any other communication among partitions.

**Requirement Platform-Conf1.5**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.5</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**
The system shall be configured concerning a set of devices that are available on the board. Each device shall be defined according to its properties.

**Brief Summary**
Devices are hardware peripherals that can be used by partitions. Devices can define a memory or I/O address where the controller is allocated or a memory area or other parameters like baud rate.

**Fit Criterion**
A device can be allocated to a partition. The configuration file shall permit to allocate the different devices to the partitions. A partition only can use those devices configured in the partition specification.

**Rationale**
Partitions require the use of external devices. Devices are not virtualized due to the existing amount of drivers. Also, the inclusion of drivers developed by suppliers should not be included in the virtualization layer for security reasons. Partitions will handle directly devices.

**Requirement Platform-Conf1.6**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.6</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**
The configuration file shall specify the set of actions to be executed by the health monitor associated to the faults generated by the virtualization layer

**Brief Summary**
The virtualization layer is configured in the configuration file. The configuration will specify the set of actions that will be executed as consequence of a trap or an health monitor event. If no action is specified, the virtualization layer will apply the default action for each event.

**Fit Criterion**
The virtualization layer will execute the specified actions associated to the health monitor events when the fault is generated at the virtualization layer.

**Rationale**
In order to guarantee the trust ability of the system, faults have to be handled according the user needs specified in the configuration file.

---

### Requirement *Platform-Conf1.7*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>132.7</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**
The configuration file shall specify the set of actions to be executed by the health monitor associated to the faults generated by partitions

**Brief Summary**
Partitions are configured in the configuration file. The configuration will specify the set of actions that will be executed as consequence of a trap or an health monitor event generated by the partition code. If no action is specified, the virtualization layer will apply the default action for each event.

**Fit Criterion**
The virtualization layer will execute the specified actions associated to the health monitor events when the fault is generated by the partition’s code.

**Rationale**
In order to guarantee the trust ability of the system, faults have to be handled according the user needs specified in the configuration file.

---

### Requirement *Platform-Conf5*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>Non-functional</td>
<td>Configuration</td>
</tr>
</tbody>
</table>

**Requirement**
The system shall be configured with a set of partitions in the configuration file. Each partition shall specify the rights, memory areas and their rights, ports used to communicate with other partitions, interrupts handled and devices used.

**Brief Summary**
Partitions are configured in the configuration file. Each partitions defines the memory areas and its rights (read/write), the partition type (normal/system), the temporal requirements (bandwidth), the health monitoring actions associated to the health monitor events (if not defined default values are assumed), the communication ports and their
Deliverable D1.5: Non-functional Requirements Analysis

<table>
<thead>
<tr>
<th>rights (source/destination), the interrupts handled directly by the partition, the trace device and the devices used.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fit Criterion</strong></td>
</tr>
<tr>
<td>The virtualization layer will guarantee that the resources configured for each partition are allocated under the rights given to the partition during the temporal windows defined in the plan.</td>
</tr>
<tr>
<td><strong>Rationale</strong></td>
</tr>
<tr>
<td>The partition configuration specifies the resources allocated to the partitions during its execution. Memory areas specify the spatial isolation properties that will be granted by the virtualization layer.</td>
</tr>
</tbody>
</table>

4.2 Deployment Requirements

These requirements define the tools to deploy a system based on the OVERSEE platform. In order to obtain an executable file to be uploaded in an embedded system, a set of tools are required. The set of tools is known as tool-chain and permits to facilitate the compilation and generation of the different pieces of software and to build a final executable file. The tools included in the tool-chain are:

- **Configuration file analysis and validation**: these tools will guarantee that the configuration file is coherent and correct. They will produce a binary representation of the configuration file.

- **System generation**: the system generation shall permit to pack the different software components (hypervisor, configuration file and partitions) in a single executable file. The final package includes a system loader which will permit to copy in the memory areas all the software components and start the execution.

- **System loader**: will permit to upload the executable file in the embedded system

4.2.1 Configuration Analysis and Validation

The configuration vector is the result of the compilation process of the configuration file. The aim is to generate a binary representation of the configuration file that defines how the system will work.

This process involves several operations that are represented in figure 4.

- **XML parser**: The configuration file is described in XML format. The first step is to parse this file and analyze the coherence and constraints of the configuration file. The result of this process is a representation of the XML description in C code where a set of data structures are generated. In this file, all non-defined parameters are set to its default values.

- **Compilation of the C code representation of the configuration file**: This process generates a binary representation of the configuration file (configuration vector) which shall be included in the final deployment.
Requirement *Platform-Dep1*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>Non-functional</td>
<td>Deployment</td>
</tr>
</tbody>
</table>

**Requirement**

*The system shall include the configuration vector which will describe the system configuration. The configuration vector is a binary representation of the configuration file.*

**Brief Summary**

The configuration file shall be parsed and compiled to obtain a binary representation. It will be used by the virtualization layer to guide the system execution. The virtualization layer will guarantee that the configuration vector can not be modified during the execution.

**Fit Criterion**

The binary representation shall represent the configuration file of the system. Any partition can access (read or write) this configuration during the execution. The information in the configuration file shall be provided by the virtualization layer through specific hypercalls.

**Rationale**

The configuration vector defines the resource allocation in the system. This configuration file is specified by the system integrator and included in the deployment. The configuration vector guides the system execution.
4.2.2 System generation

An OVERSEE system includes the following components:

- virtualization layer binary
- configuration vector
- partition’s binaries

All these components have to be joined in a single file to be loaded in the target. The file generated is a file (“system container”) that has an internal structure compatible with execution file format (ELF) in order to be executed in an emulation tool. Figure 5 shows this process.

![Figure 5. System generation](image)

**Requirement Platform-Dep2**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>Non-functional</td>
<td>Deployment</td>
</tr>
</tbody>
</table>

Requirement
A packer tool shall take as input the different binaries of an embedded system and generate a container to be deployed in the target board. The container shall follow the ELF standard.

**Brief Summary**

All the components in a partitioned embedded system need to be loaded in the target board; this can be solved in an individual or global approach. The packer tool permits to build a system container to be deployed at once.

**Fit Criterion**

The system container is a file system that can be analyzed internally to verify the content and extract the different components.

**Rationale**

The system container permits to handle binary distributions of a partitioned embedded system. Although the system container includes all the components, they can be extracted and substituted in a container.

### 4.2.3 System loader

Once the container is in the target (memory, flash memory, or other device), a boot loader has to understand the container format (ELF) and navigate in the file system to extract the components and load them in the memory to be executed. Then, the boot loader gives the control to the virtualization layer.

**Requirement Platform-Dep3**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>138</td>
<td>Non-functional</td>
<td>Deployment</td>
</tr>
</tbody>
</table>

**Requirement**

A system loader shall extract the components in a container and copy them in memory areas as specified in the configuration file.

**Brief Summary**

All the components in a partitioned embedded system need to be loaded in the target board. It can be solved in an individual or global approach. The packer tool permits to build a system container to be deployed at once.

**Fit Criterion**

Once the container is loaded in a board, the boot loader will load the components in memory and start the execution of the virtualization layer.

**Rationale**

A boot loader is the first software that is executed after a restart. It is loaded in ROM or in a specific device.
4.3 Performance Requirements

Performance evaluation is the analysis of a software behavior using information gathered during its execution to extract relevant information for the users. With respect to a virtualization layer for real time analysis the measurements are addressed in different ways:

- Measuring the services provided by the virtualization layer. These services are called hypercalls.
- Generic metric for a virtualization layer.
- Overheads introduced by the virtualization layer.

4.3.1 Predictable Services

In order to guarantee the response time of the applications, the system has to provide predictable services. These services can be classified according to their costs:

- Constant time cost: the set of services which not dependent on the parameters in the service call. These services are
  - Partition management services
  - Time management services
  - Interrupt management services
  - System status services
  - Trace and health monitor services
- Variable time cost: These services are related to the communication between partitions and depending on the amount of data sent or received. For these services the cost has two components: 1) cost of the service; 2) cost of the data transfer. The first component shall be constant whereas the second shall be linear with the amount of data.
  - Communication services through sampling or queuing ports

 Requirement Platform-Perf1

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>139.1</td>
<td>Non-functional</td>
<td>Performance</td>
</tr>
</tbody>
</table>

Requirement

The virtualization layer shall provide the time cost of all services.

Brief Summary

The services provided by the virtualization layer for real-time systems should have a predictable execution time.

Fit Criterion

A measurement of these services shall provide their costs of maximum, minimum and
average execution time. The possible variation of the cost depends on the cache conditions. In the case of services involving messages, these measurements shall be performed depending on the amount of data transferred.

**Rationale**
In a real-time system all services have to have a predictable execution time in order to guarantee the temporal constrains of the applications.

### 4.3.2 Metric for the Virtualization Layer

In real-time operating systems there is a consensus about the metrics to be used for the evaluation of the performance. However, the metric for virtualization layers has is under definition. There are some actions by the Embedded Microprocessor Benchmark Consortium to define a metric, but currently, it is a project that has not produced results.

Taking into account the experience in real-time operating systems and the new services, the following aspects can be considered as relevant for the evaluation of a virtualization layer:

- **Interrupt latency**: Cost related to the maximum latency when an external stimuli is produced and propagated to a partition.
- **Partition context switch**: The time required by the hypervisor to change the active partition.
- **Mode switch cost**: Time required to change the active mode.
- **Stability of the execution plan**: MAF duration after a long execution interval.
- **Performance loss due to the number of partitions**.
- **Performance loss due to the number of slots in the scheduling plan**.
- **Performance loss due to the number of modes defined**.

#### Requirement *Platform-Perf2*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>139.2</td>
<td>Non-functional</td>
<td>Performance</td>
</tr>
</tbody>
</table>

**Requirement**

*Virtualization layer metrics shall provide a measurement of those mechanisms that impact in the system performance.*

**Brief Summary**

A measurement of critical aspects related to the system execution will be measured. These aspects include: Interrupt latency, partition and mode context switch, plan stability and performance loss due to scheduling issues.

**Fit Criterion**

A measurement of these aspects shall provide their maximum, minimum and average costs.
Rationale
It is required to provide a cost of the internal activities of the virtualization layer to be considered in the design and the worst case execution time of the applications.

4.3.3 Virtualization Layer Overhead

Partitioned systems offer clear benefits from the view of separation of concerns. It allows the integration of applications from several developers in the platform. On the other hand, it has to be analyzed the impact to an application in terms of performance comparing its execution on a virtual machine or a native machine. In other words, it is relevant to analyze the impact of the virtualization layer.

Requirement Platform-Perf3

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>139.3</td>
<td>Non-functional</td>
<td>Performance</td>
</tr>
</tbody>
</table>

Requirement
The overhead introduced by the OVERSEE platform shall be measured with respect to the applications.

Brief Summary
From the application point of view, the overhead introduced by a partitioned system corresponds to the difference between its execution in a native or virtual machine.

Fit Criterion
An acceptable overhead shall be in the range of 2% to 5%.

Rationale
It is difficult to measure the overhead introduced by an operating system when comparing an application running on top of an operating system or without operating system. However, results of 5% to 10% can be considered acceptable. An additional overhead of 2% to 5% due to the partitioned system seems reasonable.

4.3.4 Scalability Requirements

Scalability requirements define the need to deploy the OVERSEE platform for several partitions.

In some sense, these requirements could be considered as part of the configuration and flexibility requirements. The focus of these requirements is the number of the components (partitions, channels, devices, etc.) that can be considered.

The main limitation is the amount of memory available on the board and the partition’s sizes. As the system is statically defined in the configuration file, the size of the virtualization layer depends on the number of components defined.
Additionally, the overhead of the virtualization layer should not be impacted by the number of components.

Next considerations should be taken into account:

- **Partitions:** It is difficult to give maximum number of partitions to be considered. For instance, it is not easy from the complexity design and integration point of view, to design a partitioned system with more than 10 Linux partitions.
  - partitions without operating system should have low memory needs (<512Kb). In these cases, the number of partitions could be extremely high (>50).
  - partitions with a real-time operating systems should have medium needs (<1Mb). The number of partitions could be high (<30).
  - partitions with a general purpose operating system (e.g., Linux) have strong memory needs (>4Mb). The number could be low (<10).

- **Channels:** Channels require an amount of memory that depends on the message size and the number of messages (in the case of queuing channels). The size of the internal data structures for the channels impacts the virtualization layer size, which includes these data structures. Again, the more channels and ports the more complex system will be, due to the number of communication paths.

- **Devices:** Devices have a strong dependency on the type.
  - Memory block devices: the constraint is related to the amount of memory used.
  - Hardware devices: the limitation is the number of interrupts available.

The number of partitions is limited by the partition’s sizes and the memory available on the board. A number of 20 partitions is considered as an appropriated maximum value to fulfill this requirement.

**Requirement Platform-Scal1**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>Non-functional</td>
<td>Scalability</td>
</tr>
</tbody>
</table>

**Requirement**
The virtualization layer shall allow the design of a system with a high variability of components. Components are: partitions, channels and devices.

**Brief Summary**
In general, the virtualization layer does not limit the number of components in the system. The limitation is imposed by the hardware available. The amount of memory available in an embedded system is the limiting aspect.

**Fit Criterion**
An acceptable number of these components depend in general on their memory needs.

**Rationale**
It is difficult to define a maximum number that will be acceptable. The initial approach is that the virtualization does not limit the number of components.

4.3.5 Reliability Requirements

These requirements are related to the reliability of the OVERSEE platform. It is expected that it can run during long time periods while maintaining the performance and reliability over the time.

A second aspect considered is the robustness with respect to possible faults during the execution. The platform has to handle a wide range of faults at hypervisor and partition level.

A summary of the faults to be handled are:
- Temporal isolation faults
- Spatial isolation faults
- Illegal instruction execution at partition level
- Access from a partition to a non accessible resource
- Numeric error at partition level
- Stack overflow at partition level
- Privileged instruction execution at partition level

**Requirement Platform-Rell1**

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>Non-functional</td>
<td>Reliability</td>
</tr>
</tbody>
</table>

**Requirement**
The virtualization layer shall maintain the same performance independently of the time the system is under execution.

**Brief Summary**
Embedded systems have to be under execution during long periods. The expected behavior is that the performance does not decrease during the execution period.

**Fit Criterion**
In space application, the execution period can cover several years, in avionics this period is weeks, in automotive this period could be days.

**Rationale**
Embedded systems running for long periods have to maintain the performance.
Requirement *Platform-Rell2*

<table>
<thead>
<tr>
<th>ID</th>
<th>Type</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>142</td>
<td>Non-functional</td>
<td>Reliability</td>
</tr>
</tbody>
</table>

**Requirement**

*The virtualization layer shall handle the errors generated during the execution through the health monitor*

**Brief Summary**

The health monitor is in charge of the error management during the execution. The actions to be done when an error occurs are statically specified in the configuration file.

**Fit Criterion**

After an error, the virtualization layer will execute the action specified in the configuration file. If no action is specified, a default action, depending on the component that has generated the error, is executed.

**Rationale**

Errors must be handled. All errors have to be detected and handled according the user needs.
5 Summary and Next Steps

In this document, we have analysed the non-functional requirements and constraints of the OVERSEE platform. For each set of requirements, we have provided the scope and the issues that directly or indirectly can impact on them.

Task 1.3 finishes with this deliverable. D1.4 and D1.5 (previously D1.2 and D1.3) are will used as input in the design and the validation phases.

WP2 is in charge of the design the OVERSEE platform. It defines the next 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 WP2, the first task of WP3 will start.
- Selection for reuse of existing building blocks
References

[1] OVERSEE Project. Description of Work. 2010