Trusted Computing Engineering for Resource Constrained Embedded Systems Applications

Deliverable 6.1

Specification of Platform

Project:	TERESA
Project Number:	IST-248410
Deliverable:	D6.1
Title:	Specification of Platform
Version:	v1.0
Confidentiality:	PP restricted
Author:	escrypt GmbH
Date:	31 May 2010

Part of the Seventh Framework Programme
Funded by the EC - DG INFSO

Table of Contents

1 Document History

Version	Status	Date
V0.1	Draft	31/01/2011
V0.2	Incomplete	16/03/2011
V0.3	To be reviewed	12/05/2011
V0.4	Internal reviewed at escrypt	23/05/2011
V1.0	Final version	31/05/2011

Approval		
	Name	Date
Prepared	Annika Paus	31/01/2011
Reviewed	All partners	23/05/2011
Authorised	Annika Paus	31/05/2011

Circulation	
Recipient	Date of submission
Project partners	31 May 2011
European Commission	31 May 2011

2 Executive Summary

For WP6, the evaluation of the TERESA approach, a specific application will be considered in each domain. Based on the patterns defined in WP4, each domain focuses on different patterns for the application development.

According to the description of work this document should specify the platforms used for the evaluation in the four different domains Automotive, Home Control, Industry Control and Metering, while the description of the application chosen for the evaluation should be covered in deliverable D6.2.

Due to the suggestion of the European Commission after the first TERESA review, we should first concentrate on finding one adequate, representative domain specific use case as prominent candidate for the demonstration and the proof of concept for the TERESA approach. We have therefore redistributed the order of the work.

The redistribution of the work is that we consider only one use case in this document, covering the tasks destined for the deliverables D6.1 and D6.2. The platform and application description for the other domains will then be covered in D6.2.

This allows us to start with D6.3 for the industry control domain at an earlier point.

The consortium figured out the smart metering domain as qualified, but in the metering domain until now, there exists no standardised engineering process for developing smart meter devices. Due to this fact, we decided to firstly concentrate on a use case from the industry control domain.

The types of platforms are chosen depending on adequate, representative domain specific use cases for the evaluation. The specification of the industry control platform provided in this document gives a detailed description including advantages and constraints.

While the pattern and application selection is focused in Chapter 4, Chapter 5 deals with a detailed specification of the application, before the platform specific characteristics are described in Chapter 6.

The integration and evaluation of the patterns for the chosen application will be part of the following deliverables of WP6. The same applies to the investigation for the integration of the repository access tool with the domain's process environment, which will be covered to develop the applications.

3 Measures of Success

In order to proof the TERESA concept we will provide an evaluation of the results in terms of engineering productivity. We assess if the criteria for measures of success of the TERESA objective G and S have been achieved.

TERESA has the following objectives:

- Objective G: Provide guidelines for the specification of sector specific RCES trusted computing engineering. Software process engineers in a given sector use the guideline to define a trusted computing engineering process that is integrated to the software engineering process used in the RCES sector.
- Objective S: Define a trusted computing engineering approach that is suited to a number of sectors: the automotive sector, the home control sector, the industry control sector, the metering sector.

The figure below shows the objectives of the project:

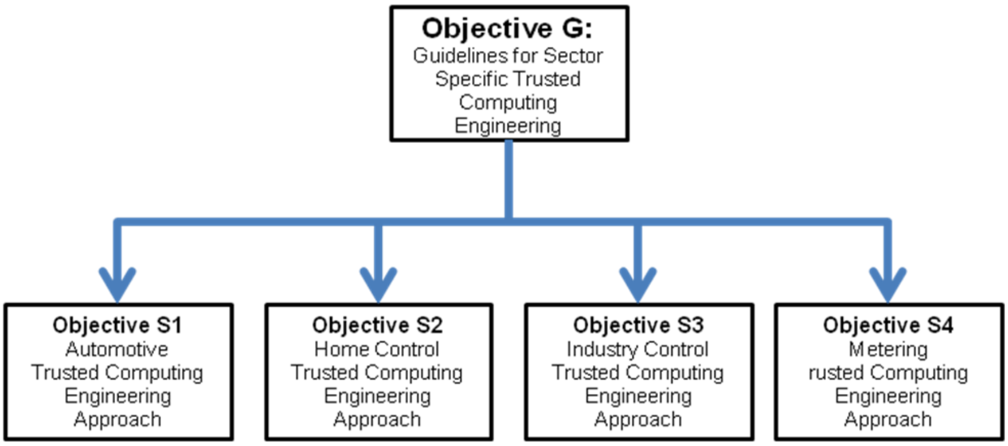

Figure 1 Guideline Objective G and Engineering Approach Objective S

Hence, the main aim of work package 6 is to show that the patterns in the pattern repository can be used for the integration into an application.

Therefore, the precondition is that the pattern repository, which is developed in the TERESA work package 4, possesses the required patterns. Hence, first the necessary patterns have to be introduced by identifying a security or dependability property of an existing application that can be extracted as a pattern.

Figure 2 shows a possible way, how the access to the repository can be realised. The investigation of the integration of repository access tools is task of work package 6. Here, we will test to which extent the provided tools are able to support the pattern integration, resp., assist the engineering process. In this context also the extendibility of the pattern repository with new patterns, as well as the extendibility of existing patterns is observed.

Furthermore, we evaluate how far the patterns are useful to increase engineering productivity. The S&D know-how comprised in a pattern (e.g., in the form of guidelines, source code, et al.) will be observed with respect to its generality. I.e. we will prove if the same guidelines can be used successfully to instantiate the four TERESA sector specific engineering processes, and if they can also be used to instantiate other processes.

We intent to demonstrate, that the TERESA pattern approach leads to a reduced number or to a simplification of the engineering process steps. The guidelines which are provided should support the developer regarding security and dependability issues and reduce the error frequency.

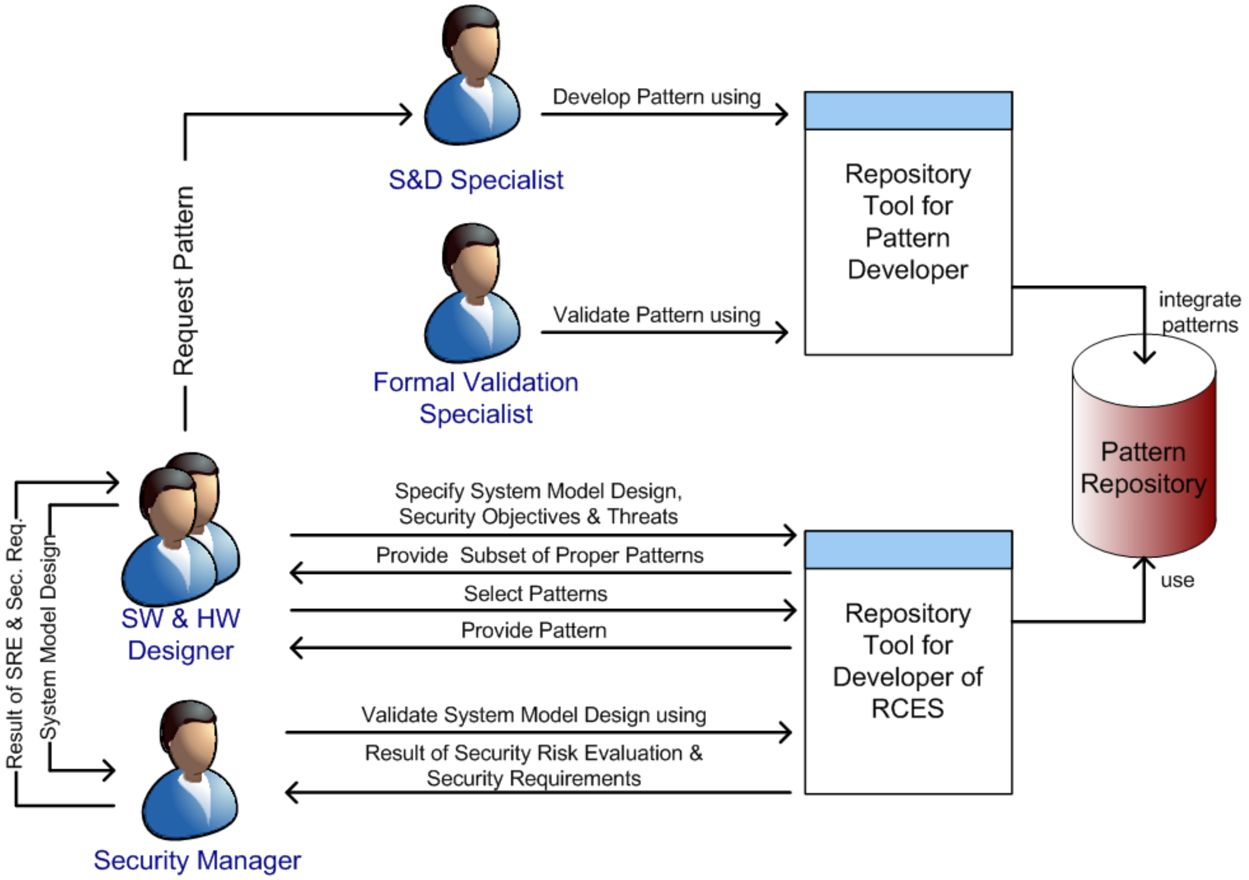

Figure 2 Access to the Pattern Repository

4 Application and Pattern Selection

The selection of applications for evaluation has to be considered as a very important decision for this work package. It has a direct impact on the selection of patterns, and thus on the whole evaluation. In the following, first the applications that have been chosen in the different domains are addressed before the patterns associated with each are described.

4.1 Application Selection

In order to find useful adequate applications for the evaluation, domain specific use cases described in [2] have been analysed. The decision which applications are taken into account in WP6 strongly relies on the selection of patterns that are considered in the TERESA evaluation.

4.1.1 Automotive

For the automotive domain based on an extensive analysis of the domain specific use cases described in Deliverable 2, the use case “Active Brake” has been figured out to be best suited for the evaluation of the TERESA approach.
This use case is a good fit for the TERESA vision due to its combined S&D characteristics. On the one hand it provides the need of a secure communication to prevent malicious activation and on the other hand it provides a severe security impact on its environment considering the people involved. It also features some security influences that have a certain impact on the safety.
The “Active Brake” use case is an excellent choice due to enormous interest in the automotive domain and its character to be up-to-date. It features a huge gain in traffic security but also can cause some unwanted failures which could lead to dependability risks. For this reason it is essential that certain dependability and security properties are met.

4.1.2 Home Control

In the home control domain we decided to choose the application: “Access to a Home device power management”.

The user remotely manages the power states of devices by using a mobile device (a Smartphone for instance). This application is quite simple but relies on widely used mechanisms and is typical for the domain.

The scenario is as follows:

- A mobile device (e.g., a Smartphone) is already owned by a user which means (e.g., keys) of authentication are available
- The mobile device therefore performs a service discovery to gather the available services from the other authenticated devices
- This service discovery is executed via a secure communication
- The various devices provide the requested information

The user is now able to switch off/on the devices using his or her own mobile device.

4.1.3 Industrial Control

For the industry control domain the application “Safe4Rail” has been chosen.

To select the application, the main characteristic that was considered was the need to develop an S&D-related system that can meet a SIL 4 level. This characteristic requires a number of design techniques as redundancy, diversity, monitoring to be taken into account when implementing the application Safe4Rail.

Once the application was analysed, a study about which of these design techniques would be implemented through the patterns identified in TERESA is carried out. As a result, it was detected that redundancy could be a candidate to be implemented with the help of these patterns.

4.1.4 Metering

The target application for the metrology domain will be an electricity meter. Electricity meters play a central role in many kinds of smart metering scenarios. Due to the fact that a reliable power supply is available, smart electricity meters are often equipped with different kinds of interfaces for receiving and transmitting data via public or private networks.

A possible application scenario would be an electricity meter collecting readout values from a nearby gas and/or water meter and sending the data together with the own readout to an appropriate remote readout centre. In this scenario the electricity meter handles all the necessary communication.

4.2 Pattern Selection

While on the one hand a range of different patterns should be covered, on the other hand the reuse of the same pattern should be proven in order to evaluate the use of the same pattern across different domains.

Hence, based on the availability of application implementations and the partner's interest the pattern represented in Figure 3 have been selected.

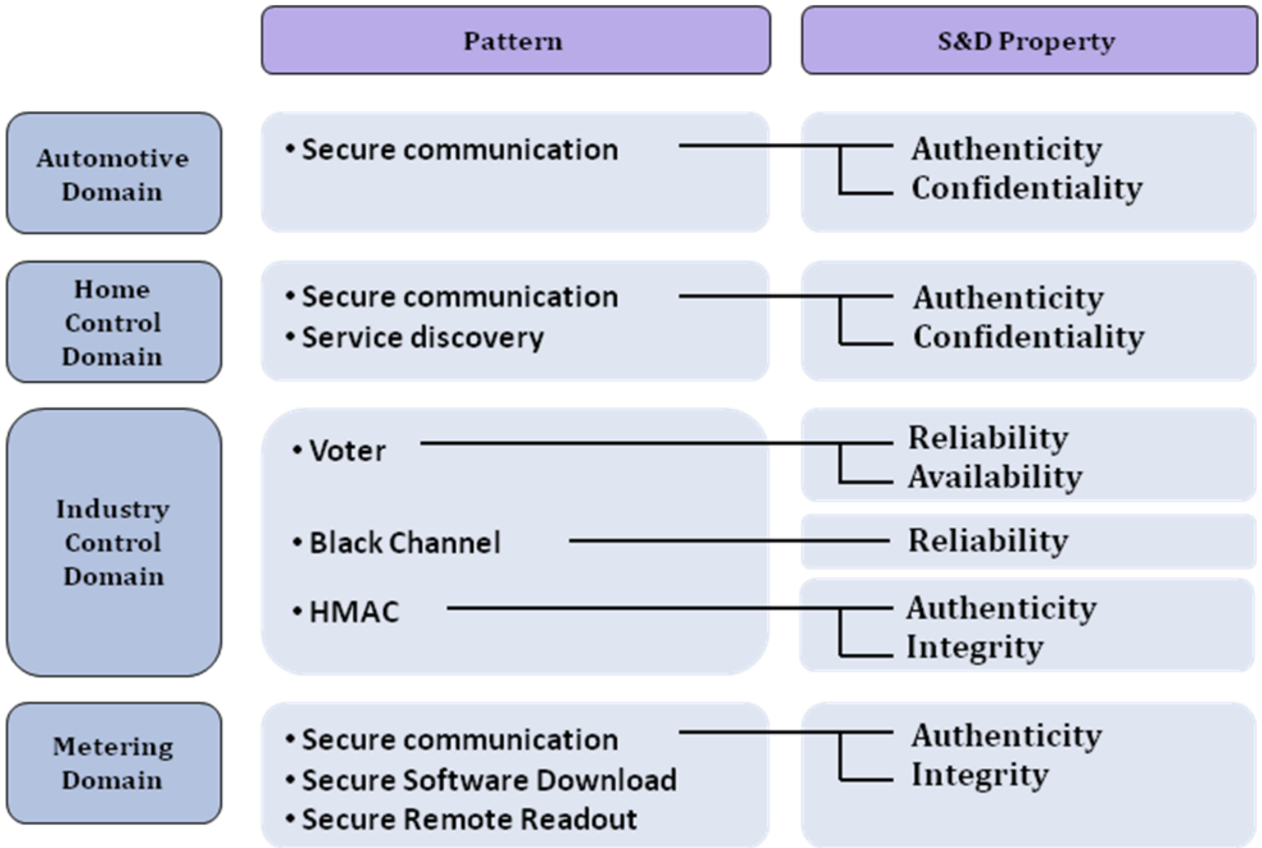

Figure 3: Selected patterns for the evaluation

We decided to choose the **Secure Communication** pattern for three different domains. For this purpose the "Secure Communication" pattern is very suitable for many reasons. It is a fundamental security objective that is of high interest in all domains. It has various application forms with different security goals. In fact it is based on numerous other patterns such as the **Authentication**, **Public Key Infrastructure**, **Key Agreement** and **Encryption** pattern. At the same time it serves as basis for several other patterns such as the **Secure Software Download** and the **Secure Remote Readout** pattern. Furthermore, the **Secure Communication** pattern allows us to evaluate the use of the same pattern across different domains.

Beside the Secure Communication pattern the HMAC pattern is evaluated as a second security pattern.

As dependability patterns on the one hand the **Voter** pattern is focused and on the other hand the **Black Channel** pattern is focused in the industry control domain.

4.2.1 Automotive

Secure Communication: This pattern features the most important properties in the chosen use case. The communication between vehicles should fulfil certain security qualities like authentication or confidentiality. It should be assured that all communication is secure under defined assumptions.

4.2.2 Home Control

Various patterns are involved in this application. The two main ones, already identified in [2], are **Service Discovery** and **Secure Communication**. As described, it is possible to merge both into a new **Secure Service Discovery** pattern.

The authentication will rely on a classical message authentication code based on its private key. A good candidate is the **HMAC** pattern.

All the communications that are take place are using the **Secure Communication** pattern.

4.2.3 Industrial Control

To select the patterns used to implement redundancy in the application Safe4Rail, the restrictions imposed by the redundant systems and by the architecture e.g. guarantee that the system is deterministic, secure and reliable communication between the nodes through and standard mechanism, as the characteristics of the application e.g. multiple input signals of different types coming from different sources, and support of different languages, were considered. The patterns selected are:

- **Voter:** This design technique provides the following characteristics to the application.
    - Early fault detection (At different stages of the application -data acquisition, calculate, decision-).
    - Support interaction between the different developing languages (in the application each channel of the redundant system is developed with a different language).
    - Select the data value that is going to be used to perform the operations in the application.
- **Black channel:** This mechanism provides a dependable communication between the different channels of the redundant system.
- **HMAC:** This mechanism provides a secure communication between the different channels of the redundant system.

4.2.4 Metering

Based on the SYM2-specification for smart electricity meters (www.sym2.org), a smart electricity meter application will be used to implement, integrate and evaluate one of the following patterns as a proof of concept for the TERESA approach for the metrology domain:

- **Secure Software Download:** The use of the Secure Software Download makes it possible to securely remote update an embedded system. This mechanism is of great importance especially for the metrology domain. It may be necessary to update the software of a smart meter in the field in case of correcting errors or adding new functionalities to the meter. In consequence of the lawful regulations, the new meter software needs to be approved by a notified body. The measuring point operator has to ensure that only approved software is downloaded to the meter. Authenticity and integrity of the software image and its origin have to be provided. Compared to other patterns, the Secure Software Download is relatively complex. The mechanism itself consists of the following steps:
    - Enable the download functionality by sending a command
    - Transmit the software image to the embedded (meter) system ensuring authenticity and integrity (e.g. by using digital signatures)
    - Send a command for initiating a system reboot
    - Start the new, downloaded software image
- **Secure Remote Readout:** The Secure Remote Readout is used to read out measurement data from meters installed in the field. Authenticity and integrity of the measurement data needs to be provided.

Both patterns, the Secure Software Download as well as the Secure Remote Readout fit the hyponymy of **Secure Communication**. The considered patterns therefore are based on the underlying **Secure Communication** pattern.

The integration of this pattern is unambiguous to guarantee the security aspects of a secure download but also the aspects of a secure readout.

5 Application Specification

This section focuses on the specification of the domain applications, which are addressed within the evaluation of the TERESA approach, even though, according to the description of work, it is part of deliverable D6.2. Section 5.1 describes which information needs to be provided about the application, while Section 5.2 already gives detailed information about a specific application of the industry domain.

5.1 Application Characteristics

Besides a description, for a well-structured documentation, it is important to state all requirements and assumptions that result from, resp. are related to, each characteristic.

	Application Characteristics	**Description**
General	Application Title	Title of the application.
	Functionality	Short description of the application.
System Model Description	Actors/Roles System entities/ components	If the application has a user interface, the actors and roles, which are related to the application, should be described. Furthermore, the involved system entities and components should be explained to get a detailed overview.
	Data/ Interfaces	For all kind of data that is transferred, the amount of data and its use should be specified. In this conjunction it is important to outline the system components that send, receive, and handle the data. Furthermore, the interfaces should be described, including the outside interfaces, internal interfaces and user interfaces.
	Functions/ Use cases	More detailed description of the application functionality that identifies all relevant functions and use cases. Here, for each function a summary, the involved actors and system components, preconditions, the execution flow, as well as alternative execution flows, e.g. in the case of failure behaviour is described. Furthermore, dependencies between different functions should be outlined.

Security Objectives	Security goals and assets	Necessary security functionalities and security properties required by the application. E.g., • Secure cryptography • Secure implementations • Secure authentication • Secure communication • True random numbers • Secure non-volatile memory • Secure organisation / administration • Secure identification and authentication of users • Secure access control • Secure protection of data and software …
Security Threats	Potential attackers and threats	Identification of potential attackers and threats, i.e., • internal attackers, e.g., malware • Man-in-the-middle, external attacker
	Security environment	For all components involved in the application that have an influence on the security, the requirements or assumptions should be described. This should include all security requirements of hardware and peripherals, software and run-time environment, interfaces and communications. Besides, the security functions implemented by the security-related part of the application should be described.
	Attack paths	Description of all known attack paths for each security objective and identification of possible attacks per attack path.
Security Risk Analysis	Attack potentials	The analysis of the potential attack is specified in an analogous manner as in the research project EVITA [5], Appendix C1.3. The attack potential is well defined in the ISO 15408 standard [3], which is based on the "Common Criteria for Information Technology Security Evaluation", and in [4]. As specified in [3] the attack potential is based on the following values: • Estimated attack duration • Required expertise / skills • Necessary knowledge about the object of attack • Necessary level of access • Necessary equipment As already accomplished in [5], for determining the attack potential of each attack path the appropriate values from Table 2 are summed up before Table 3 is applied to classify the attack potential.

Security Risk Analysis	Potential damages	Analysis of the potential damage resulting from security functionalities For determining the potential damage of a security attack, we proceed in the same way in [5], Appendix C1.2. For the security threats the following aspects are taken into account: • Safety (Human related) - possible health effects (IEC61508/ISO26262) • Privacy - identification and tracking of vehicles or individuals; • Financial - possible financial losses (business models, damages, loss of reputation ...) • Operational - possible operating losses (comfort limitations, loss of service) We have adapted the estimation of the potential damages of EVITA by generalising the automotive specific values and established Table 4. Using this table for each of the four aspects from above a different rating can be assigned, which is required to determine the risk of an attack later on.
	Risk evaluation	An analysis of the security risk is performed according to [1] and [5], Here, the risk is defined as: risk = (probability of an accident) x (expected losses of that accident) This result in a table visualising the application's attack tree augmented with risk analysis parameters including the values shown in the Table 5. A detailed description how this table can be achieved is provided in [5], Appendix C1.4.
Dependability Objectives	Dependability goals and assets	Necessary dependability functionalities and dependability properties according to [6]. E.g. • Reliability • Availability • Maintainability • Safety
Dependability Threats	Failure analysis	Identification of all possible modes of failures of the system's components. This should also include the possible causes of a failure and the possible effects.
	Dependability environment	For all components involved in the application that have influence on the dependability, the requirements or assumptions should be described. This should include all dependability requirements of hardware and peripherals, software and run-time environment, interfaces and communications. Besides, it should be described the dependability functions implemented by the dependability-related part of the application.
Dependability Risk Analysis	Risk evaluation/ Probability of an accident	Analysis of the dependability risk and the probability of potential accidents.

Table 1 Description of application characteristics

Factor	Level	Comment	Value
Elapsed Time	≤ 1 day		0
	≤ 1 week		1
	≤ 1 month		4
	≤ 3 months		10
	≤ 6 months		17
	> 6 months		19
	not practical	The attack path is not exploitable within a timescale that would be useful to an attacker	∞
Expertise	Layman	Unknowledgeable compared to experts or proficient persons, with no particular expertise	0
	Proficient	Knowledgeable in being familiar with the security behaviour of the product or system type	3
	Expert	Familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, principles and concepts of security employed, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc.	6
	Multiple experts	Different fields of expertise are required at an Expert level for distinct steps of an attack	8
Knowledge of system	Public	e.g. as gained from the Internet	0
	Restricted	e.g. knowledge that is controlled within the developer organisation and shared with other organisations under a non-disclosure agreement	3
	Sensitive	e.g. knowledge that is shared between discreet teams within the developer organisation, access to which is constrained only to team members	7
	Critical	e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need-to-know basis and individual undertaking	11
Window of Opportunity	Unnecessary /unlimited	The attack does not need any kind of opportunity to be realised because there is no risk of being detected during access to the target of the attack and it is no problem to access the required number of targets for the attack	0
	Easy	Access is required for ≤ 1 day and number of targets required performing the attack ≤ 10	1
	Moderate	Access is required for ≤ 1 month and number of targets required to perform the attack ≤ 100	4
	Difficult	Access is required for > 1 month or number of targets required to perform the attack > 100	10
	None	The opportunity window is not sufficient to perform the attack (the access to the target is short to perform the attack, or a sufficient number of targets is not accessible to the attacker)	∞
Equipment	Standard	Readily available to the attacker	0
	Specialised	Not readily available to the attacker, but acquirable without undue effort. This could include purchase of moderate amounts of equipment or development of more extensive attack scripts or programs	4
	Bespoke	Not readily available to the public because equipment may need to be specially produced, is so specialised that its distribution is restricted, or is very expensive	7
	Multiple bespoke	Different types of bespoke equipment are required for distinct steps of an attack	9

Table 2 Rating of aspects of attack potential [5]

Values	Attack potential required to identify and exploit attack scenario	Attack probability *P* (reflecting relative likelihood of attack)
0-9	Basic	5
10-13	Enhanced-Basic	4
14-19	Moderate	3
20-24	High	2
≥ 25	Beyond High	1

Table 3 Rating of attack potential and attack probability [5]

Security threat severity class	Aspects of security threats			
	Safety (S_S)	Privacy (S_P)	Financial (S_F)	Operational (S_O)
0	No injuries.	No unauthorised access to data.	No financial loss.	No impact on operational performance.
1	Light or moderate injuries.	Anonymous data only (no specific entity data).	Low-level loss (~€10).	Impact not discernible to entity.
2	Severe injuries (survival probable). Light/moderate injuries for multiple entities.	Identification of entity. Anonymous data for multiple entities.	Moderate loss (~€100). Low losses for multiple entities.	Entity aware of performance degradation. Indiscernible impacts for multiple entities.
3	Life threatening (survival uncertain) or fatal injuries. Severe injuries for multiple entities.	Entity tracking. Identification of entity for multiple entities.	Heavy loss (~€1000). Moderate losses for multiple entities.	Significant impact on performance. Noticeable impact for multiple entities.
4	Life threatening or fatal injuries for multiple entities.	Entity tracking for multiple entities.	Heavy losses for multiple entities.	Significant impact for multiple entities.

Table 4 Generalisation of the rating of potential damages described in [5]

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack potential (*A*)	Asset (attack)	Attack Probability (*P*)
A	$\mathbf{S_A}$	A1	$\mathbf{R_{A1}(S_A,}A_{A1})$	$A_{A1}=\min\{Pa,Pb\}$	a & b	*Pa* *Pb*
		A2	$\mathbf{R_{A2}(S_A,}A_{A2})$	$A_{A2}=\max\{Pd,Pe,Pf\}$	d	*Pd*
					e	*Pe*
					f	*Pf*
B	$\mathbf{S_B}$	B1	$\mathbf{R_{B1}(S_B,}A_{B1})$	$A_{B1}=\max[\min\{Pa,Pd,Pc\},\min\{Pc,Ph\}]$	a & b & c	*Pa* *Pb* *Pc*
					c & h	*Pc* *Ph*
		B2	$\mathbf{R_{B2}(S_B,A_{A2})}$	$A_{B2}=Pg$	g	*Pg*

Table 5 Application attack tree augmented with risk analysis parameters [5]

5.2 Domain Specific Application Specifications

5.2.1 Industry Control

The following railway application is described and the application characteristics of Section 5.1 are taken into account.

5.2.1.1 General

	Application Characteristics	Description
General	Application Title	Safe4Rail
	Functionality	Maintained at all times the speed and distance travelled by the train below the threshold set by the environment. Supervise distance and speed of a train provided by the odometry system, and compare them to the limits imposed by the infrastructure. If the maximum speed is overcome, a warning will be activated in a first term, then the service brake will be applied and finally the emergency brake will stop the train.

5.2.1.2 System Model Description

The System Model Description (SMD) is one of the most important analysing parts when it comes to application specification. An underestimated and improperly done analysis of the System Model can put all consecutive work to waste. An accurate System Model can yield improved performance and thus profitable results. First, the actors and roles are defined.

5.2.1.2.1 Actors and Roles

System Model Description	Actors/Roles	Driver of the railway	
			The actor Driver interacts with the train supervision system by switching between "*Standby*" and *"Supervision"* mode and by releasing the emergency brake.

5.2.1.2.2 System Entities and Components

In addition to the Actors and Roles the specific system entities and components have to be analysed. Figure 4 gives an overview about the whole system design before a description for each subsystem that contributes to the overlaying system is given. Furthermore the list provides the requirements which these components will fulfil.

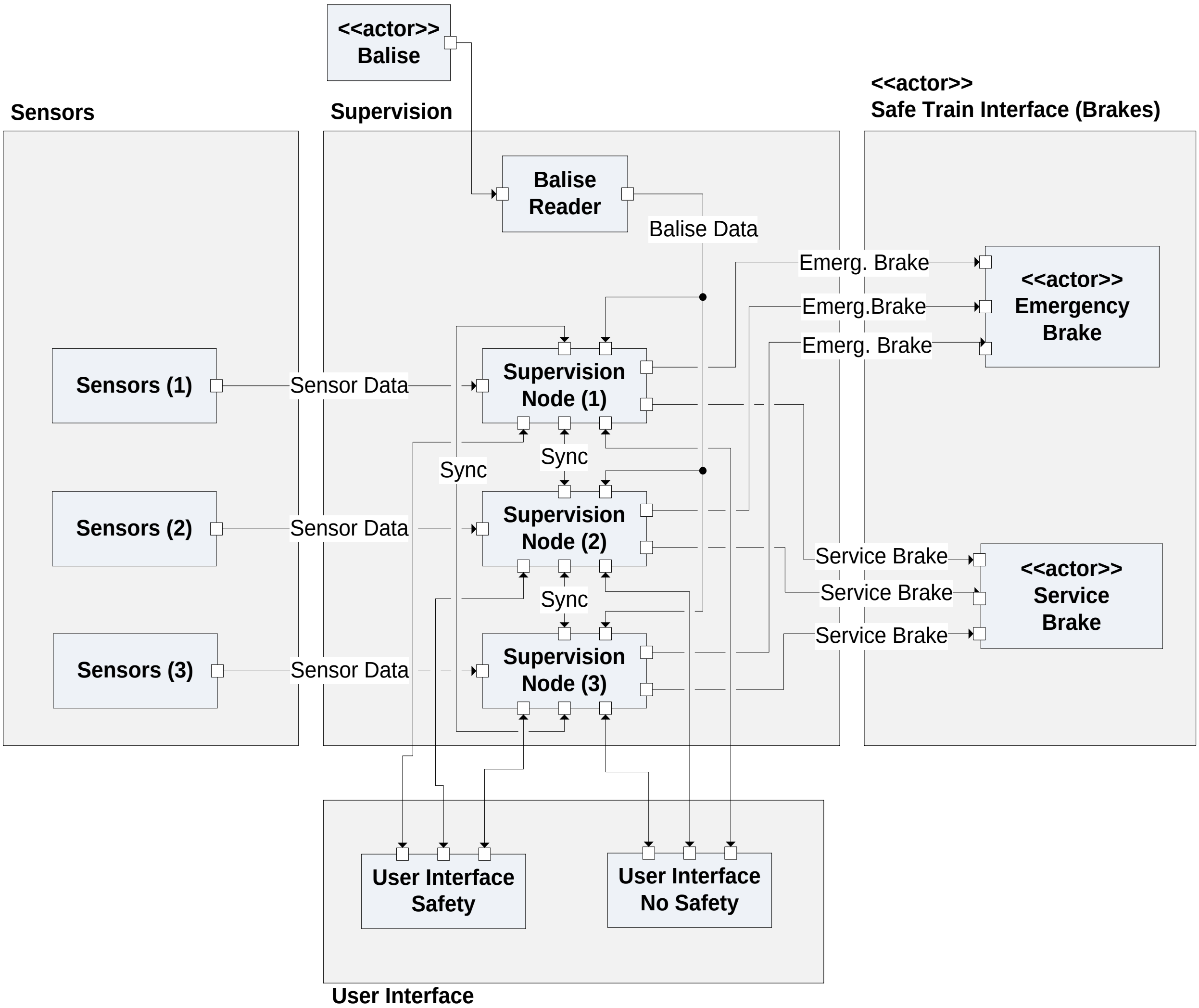

Figure 4 "Safe4Rail" System Entities and Components

System Model Description	System entities/ components	Clock	
			Generates a periodic event which triggers the system to estimate the current position and speed and to supervise that the train complies with the current track restrictions.
		Environmental Conditions	
			Represent the physical interaction between environment (train, track, others) with the sensors of the system.
		Balise	
			Represents a Balise installed on the track which supplies to the train supervision system with new information regarding the current position and the track conditions.
		Safe Train Interface	
			Represents the actuators for the application.

System Model Description

System entities/ components

Supervision System

Sensors. Provide the actual position and speed of the train and the track conditions to the system.

Req. 1.1: Sensors

- **Req.1.1.1:** In order to know the speed, position and current track conditions, different sensors (encoders, accelerometers, balises) to measure the environment conditions are needed.

Supervision. The main component of the system responsible of carrying out the functionality of the system.

Req. 1.2: Functional

- **Req. 1.2.1:** The supervision application is a software system component and is executed by the microprocessor of the Supervision Node (x) system component. The application is responsible for estimating and supervising the current speed and position of the train and the corresponding warnings to the driver and the activation/deactivation of the brakes.

Odometry. Implement the algorithm to calculate the actual speed and position of the train.

- **Req. 1.2.1.1:** The supervision application must estimate the train speed and position.

ModeControl. Control the functional mode of the system.

- **Req. 1.2.1.2:** The system must have the following operating modes:
  - StandBy
  - Supervision

Decisions. Only active in “*Supervision”* mode

- **Req. 1.2.1.3:** Supervision mode should do the supervision of the train speed and position.
- **Req. 1.2.1.4:** This function must decide if the warning or service brake or emergency brake have to be activated or not depending on the user commands and the current speed and position.
  - *BrakeCurveGenerator.* Provides the speed limit at all times.
    - ***Req. 1.2.1.4.1:*** *This function must provide for the current position the speeds when the warning and the service brake are activated, and the maximum allowed speed of the train.*
  - *Decide.* Compare the speed provide by *“Odometry”* and the one provided by the *“BrakeCurveGenerator” and take the right decision*
    - ***Req. 1.2.1.4.2:*** *The Decisions function must compare the estimated speed with the maximum allowed speed provided by the Brake Curve Generation function.*
    - ***Req. 1.2.1.4.3:*** *If the estimated speed is higher or equal to the maximum allowed speed, the emergency brake must be activated. If the estimated speed is lower than the maximum allowed speed, the status of the emergency brake must not be changed.*
    - ***Req. 1.2.1.4.4:*** *Once the emergency brake has been activated, it can only be deactivate when the train has been stopped (estimated speed < 0.1 m/s) and at the same time the function has received the release emergency brake command.*

StandBy. Only active in *“StandBy”* mode. Check the status of the system and keep the emergency brake activated.

- **Req. 1.2.1.5:** StandBy mode should do the necessary checks to ensure that the system works properly and it can carry out the dependability function.

System Model Description	System entities/ components			*BaliseReader.* Detect and read the information provided by the balise on the rail. - **Req. 1.2.1.6:** Once the train passes over a balise, the system must detect it, and read the information provided by the balise (absolute position, ground inclination, position of next balise, others).
			User Interface. The driver interacts with the system through this interface. **Req. 1.3: User Interface** • **Req. 1.3.1:** The following values must be visualised by the driver at the user interface: * Current speed * Current position * Target speed * Target position • **Req. 1.3.2:** The following information must be visualised by the driver at the user interface: * Access into the warning zone * Access into the service brake zone * Use the service brake * Access into the emergency brake zone * Use the emergency brake *Hardware.* Physical characteristics required for the platform implementation. **Req. 1.4: System** **Req. 1.4.1: Physical format** - **Req. 1.4.1.1:** The system is integrated by three nodes. Each node is composed by one carrier and one module. * **Req. 1.4.1.1.1:** The carrier must support the standard COM Express Type 2. * **Req. 1.4.1.1.2:** The carrier must support a Spartan 6. * **Req. 1.4.1.1.3:** The module used must be a COM Express Module. **Req. 1.4.2: Voltage Monitoring** - **Req. 1.4.2.1:** Each carrier must have an independent voltage monitor that will generate a rest if the power supply is not correct. **Req. 1.4.3: Temperature sensor** - **Req. 1.4.3.1:** A temperature sensor must collect the temperature in each carrier. - **Req. 1.4.3.2:** If the temperature sensed is out of range (0ºC – 255ºC), the sensor must send a signal to the Spartan to put the carrier in standby or turn off the carrier. - **Req. 1.4.3.3:** The temperature of the processor and the voltage supply to the processor must be monitored. **Req. 1.4.2: FPGA** - **Req. 1.4.2:** An FPGA must be used to manage the communication between nodes. * **Req. 1.4.2.1:** FPGA selected must support communication through Ethernet. * **Req. 1.4.2.2:** FPGA selected must support communication through PCI Express.	

System Model Description	System entities/ components		
			Req. 1.4.3: Processor - **Req. 1.4.3.1:** The processor must support a Virtualisation Technology. - **Req. 1.4.3.2:** The processor must support an HyperThreading Technology - **Req. 1.4.3.3:** The processor selected must be a low power consuming processor. - **Req. 1.4.3.4:** The processor must support internal thermal management. **Req. 1.4.4: Memory** - **Req. 1.4.4.1:** The carrier will have an external flash memory to store the data and image of the operating system at initialisation. - **Req. 1.4.4.2:** The carrier will have two external flash memories to store the configuration bits of the FPGA. **Req. 1.4.5: Interface** - **Req. 1.4.5.1:** The carrier must provide two Ethernet connections to establish the communication between the FPGA and the processor and between the nodes. - **Req. 1.4.5.2:** The carrier must support a PCI Express connection to establish the communication between the FPGA and the processor. - **Req. 1.4.5.3:** The carrier should provide two FMC connections to support extra modules. - **Req. 1.4.5.4:** The carrier must provide an IDE connection to establish the communication between the Compact flash and the Processor. - **Req. 1.4.5.5:** The carrier should provide an USB connection to ease the verification of the platform and to provide an alternative way to upload an image of an operating system in the platform. - **Req. 1.4.5.6:** The carrier should provide GPIO connections. - **Req. 1.4.5.7:** The carrier must provide a RS-232 connection to support the input of the sensors to the application. - **Req. 1.4.5.8:** The carrier should provide a LVDS connection to support the graphic interface.

5.2.1.2.3 Data and Interfaces

Additionally the used Data and Interfaces come into consideration. The importance of gaining a complete overview of the working data allows a deep analysis of security critical proceedings. Firstly we will deal with the interfaces that communicate with the outside.

System Model Description	Data/ Interfaces	Data (amount)
		Outside Interfaces
		Encoder (6) / Accelerometer (3). • *"Sensors"* block captures the information of the environment conditions and sends it to the *"Supervision"* block • *"Supervision/Odometry"* block uses this data to calculate the actual speed and position.
		Balise data (1). • *"BaliseReader"* block receives the information of the balise and sends it to the "*Supervision"* block. • *"Supervision/Odometry"* block uses this data to calculate the actual speed and position. • "Supervision*"* block uses this data to calculate the speed and position and takes the decision of activate/deactivate the emergency and service brakes.
		ServiceBrake (3)* • *"Supervision/Decision/Decide"* block sends this data to the service brake. • Activate/deactivate the service brake.
		EmergencyBrake (3)* • *"Supervision/Decision/Decide"* block sends this data to the emergency brake when Supervision mode is activated. • *"Supervision/StandBy"* block sends this data to the emergency brake when StandBy mode is activated. • Activate/deactivate the emergency brake

Secondly the Interfaces that deal with the inside communication are considered. These Internal Interfaces are mainly the sensors that gather information.

System Model Description	Data/ Interfaces	Internal Interfaces
		Position (3)*. • *"Supervision/Odometry"* block sends this data to the *"Supervision/Decision"* block. • *"Supervision/Decision/BrakeCurveGenerator"* block uses this data to generate the warningSpeed, serviceSpeed and emergencySpeed data. • *"Supervision/Decision/Decide"* block uses this data to take the decision of activate/deactivate the emergency and service brake. • *"Supervision/Decision/BrakeCurveGenerator"* block uses this data to *generate* the data warning, warningSpeed, serviceSpeed, emergencySpeed, and maxPosition).

System Model Description	Data/ Interfaces	Speed (3)*. • *"Supervision/Odometry"* block sends this data to the *"Supervision/Decision"* block. • *"Supervision/Decision/Decide"* block uses this data to take the decision of activate/deactivate the emergency and service brake.
		WarningSpeed (3)*. • *"Supervision/Decision/BrakeCurveGenerator"* block sends this data to the *"Supervision/Decision/Decide"* block. • *"Supervision/Decision/Decide"* block uses this information to take the decision of activate/deactivate the warning signal.
		ServiceSpeed (3). • *"Supervision/Decision/BrakeCurveGenerator"* block sends this data to the *Supervision/Decision/Decide"* block. • *"*Supervision/*Decision/Decide"* block uses this data to activate/deactivate the serviceBrake signal.
		* These data is used by the *"User Interface"* block and by the *"Supervision/Decision"* block.
		Internal Interfaces
		EmergencySpeed (3). • *"Supervision/Decision/BrakeCurveGenerator"* block sends this data to the *Supervision/Decision/Decide"* block. • *"Supervision/Decision/Decide"* block uses this data .to activate/deactivate the emergencyBrake signal.
		ActivateStandby (3) • *"Supervision/ModeControl"* block sends this data to the "*Supervision/StandBy"* block. • Enable/disable the "*Supervision/StandBy"* block.
		ActivateSupervision (3) • *"Supervision/ModeControl"* block sends this data to the *"Supervision/Decision/BrakeCurveGenerator"* block and to the *"Supervision/Decision/Decide"* block. • Enable/disable the *"Supervision/Decision/BrakeCurveGenerator"* and *"Supervision/Decision/Decide"* blocks.

Lastly the interfaces that interact with the user and the system defined roles are considered. These interfaces must deal with user input and are therefore more security- and dependability-critical.

System Model Description	Data/ Interfaces	User Interface
		Position (3)* • *"Supervision/Odometry"* block sends this data to the User Interface. • *"User* Interface*"* block uses this data to inform the user the actual position.
		Speed (3)* • *"Supervision/ Odometry"* block sends this data to the User Interface. • *"User* Interface*"* block uses this data to inform the user the actual speed.

System Model Description	Data/ Interfaces			Warning (3) • *"Supervision/Decision/Decide"* block sends this data to the User Interface. • *"User* Interface*"* block uses this data to inform the user that the actual speed is higher than the warningSpeed.
				ServiceBrake (3)* • *"Supervision/Decision/Decide"* block sends this data to the User Interface. • *"User Interface"* block uses this information to inform the user the actual status of the service brake.
				* These data is used by the *"User Interface"* block and by the *"Supervision/Decision"* block.
			User Interface	
				EmergencyBrake (3)* • *"Supervision/Decision/Decide"* block sends this data to the User Interface. • *"User Interface"* block uses this information to inform the user the actual status of the emergency brake.
				WarningSpeed (3)*. • *"Supervision/Decision/BrakeCurveGenerator"* block sends this data to the *"User Interface"* block. • *"User Interface"* block uses this information to inform the user the actual value of the warningSpeed.
				MaxPosition (3). • *"Supervision/Decision/BrakeCurveGenerator"* block sends this data to the *"User Interface"* block. • *"User Interface"* block uses this data to inform the user the maximum position that the train can reach.
				ChangeMode (3) • *"User Interface"* sends this data to the *"Supervision/StandBy"* block. • Enable/disable the *"Supervision/StandBy"* block.
				ResetEmergencyBrake (3) • *"User Interface"* sends this data to the *"Supervision/Decision/Decide"* block. • *"Supervision/Decision/Decide"* block uses this data to deactivate the emergency brake.
				* These data is used by the *"User Interface"* block and by the *"Supervision/Decision"* block.

5.2.1.2.4 *Functions and Use Cases*

The following section deals with the Functions and Use Cases. This is essential because these are the capabilities the user expects from the system. The system is defined through its functions from the user's point of view. An exact description is unambiguous to determine the security of the overlaying system. In the Safe4Rail use case we have 11 functions. In the following sections we give a look on the communication between subsystems, the risk calculation and the autonomous activation of the emergency brake.

System Model Description	Functions/ Use cases	Function 1		
			Objective	
				Supervise the train speed and position for the service brake.
			Involved actors and system components	
				Clock
			Preconditions	
				A1.1: System must be in supervision mode.
			Execution flow	
				Check the actual speed value. Check the actual position value. Check the value of warningSpeed, and serviceSpeed. Decide to activate or deactivate the warning signal and the service brake. Send the information about the actual status of the safety application to the user (warning, maxDistance, warningSpeed and status of serviceBrake).
			Alternative flow	
				N/A
			Dependencies	
				Function 3: Supervise the current position and speed and activate the warning signal and the service brake accordingly.
				Function 5: Provide the information to the user.
		Function 2		
			Objective	
				Supervise the train speed and position for the emergency brake.
			Involved actors and system components	
				Clock. To guarantee the order execution.
			Preconditions	
				A2.1 System must be in supervision mode
			Execution flow	
				Check the actual speed value. Check the actual position value. Check the value of the emergency speed. Decide to activate or deactivate the service brake. Send the information about the status of the emergency brake to the user.
			Alternative flow	
				N/A
			Dependencies	
				Function 8: Supervise the current position and speed and activate the emergency brake accordingly.
				Function 5: Provide the information to the user.

System Model Description	Functions/ Use cases	Function 3		
			Objective	
				Supervise the current position and speed and activate the warning signal and the service brake accordingly.
			Involved actors and system components	
				SafeTrainInterface
			Preconditions	
				A 3.1: System must be in supervision mode.
			Execution flow	
				Get the warning speed and service speed. Get the actual speed and position. Compare the actual speed with the warningSpeed* and serviceSpeed** corresponding to the current position. Decide to activate/deactivate the warning signal or the serviceBrake signal.
				* The speed value used by the application to decide whether the actual train speed is high enough to activate the warning signal. ** The speed value used by the application to decide whether the actual train speed is high enough to activate the service brake.
			Alternative flow	
				N/A
			Dependencies	
				Function 4: Estimate current position and speed.
		Function 4		
			Objective	
				Estimate current position and speed.
			Involved actors and system components	
				N/A
			Preconditions	
				A4.1: System must be in supervision mode.
			Execution flow	
				Take the data coming from the sensors/balise. Calculate the actual speed. Calculate the actual position.
			Alternative flow	
				N/A
			Dependencies	
				Function 7: Send the information of the balise.
				Function 6: Send the information of the sensors.

System Model Description	Functions/ Use cases			
		Function 5		
			Objective	
				Provide the information to the user
			Involved actors and system components	
				Driver
			Preconditions	
				N/A
			Execution flow	
				Receive the actual value of the status of the system (warning, warningSpeed, emergencyBrake, serviceBrake, speed, position, others) Update the information of the system in the user interface.
			Alternative flow	
				N/A
			Dependencies	
				N/A
		Function 6		
			Objective	
				Send the information of the sensors
			Involved actors and system components	
				Sensors (Encoders)
			Preconditions	
				N/A
			Execution flow	
				Check the environment conditions. Provide the system with the speed measured by the sensors.
			Alternative flow	
				N/A
			Dependencies	
				N/A
		Function 7		
			Objective	
				Send the information of the balise.
			Involved actors and system components	
				Balise
			Preconditions	
				N/A
			Execution flow	
				Check the presence of a balise. Provide the system with the balise's data.
			Alternative flow	
				N/A
			Dependencies	
				N/A

System Model Description	Functions/ Use cases	Function 8		
			Objective	
				Supervise the current position and speed and activate the emergency brake accordingly.
			Involved actors and system components	
				SafeTrainInterface
			Preconditions	
				A5.1 System must be in supervision mode.
			Execution flow	
				Check the status of the resetEmergencyBrake signal. Compare the actual speed with the emergencySpeed* corresponding to the actual position. Decide to activate/deactivate the emergencyBrake signal.
				* The speed value used by the application to decide whether the actual train speed is high enough to activate the service brake.
			Alternative flow	
				N/A
			Dependencies	
				Function 4: Estimate current position and speed.
				Function 9: Send the emergency brake command.
		Function 9		
			Objective	
				Send the emergency brake command.
			Involved actors and system components	
				Driver
			Preconditions	
				A 6.1 The train must be stopped.
			Execution flow	
				The driver requests the deactivation of the emergency brake through the user interface. Send the resetEmergencyBrake signal to the system.
			Alternative flow	
				N/A
			Dependencies	
				N/A

System Model Description	Functions/ Use cases			
		Function 10		
			Objective	
				Change the state between *"StandBy"* and *"Supervision"* modes.
			Involved actors and system components	
				Driver
			Preconditions	
				N/A
			Execution flow	
				The driver request to change the operating mode. The system changes its operating mode.
			Alternative flow	
				N/A
			Dependencies	
				N/A
		Function 11		
			Objective	
				Activate the emergency brake and perform diagnostics
			Involved actors and system components	
				Clock
			Preconditions	
				A11.1 The system must be in *"StandBy"* mode.
			Execution flow	
				The system must keep the emergency brake activated. The system periodically must check the status of all subsystems
			Alternative flow	
				N/A
			Dependencies	
				N/A

5.2.1.2.5 Security characteristics

After the specification of all system entities and functions the Security Characteristics are taken into consideration. This part analyses the predefined security goals the system should provide. In regard to the system description above security threats will be made clear. The threats and their attack potential will be evaluated corresponding to [5]. This is even more important in the industrial domain because an underrated security threat could cause severe health risks for many people.

The communication between nodes is carried out through a standard communication mechanism which is accessible by any attacker.

Security Objectives	Security goals and assets	**Security goals:** • **Authenticity:** Because of the physical separation of the nodes in the train, it is important to assure the authenticity of the information shared between nodes to guarantee that information cannot be manipulated by an external attacker. **Req. 5.1: Security** • **Req. 5.1.1:** The authenticity of both the source of the information and its integrity must be guarantee. **Assets:** • **Communication channel between nodes** • **Assumption:** It is assumed that the nodes (sensors and supervision node) itself are authentic and cannot be attacked.
Security Threats	Potential attackers and threats	• External attacker, e.g. Man-in-the-middle
	Security environment	**Secure Interface:** • **Assumption:** The secret key has to be shared between all the participants of the communication. **Req. 5.1.2: Secure Interface** • **Req. 5.1.2.1:** The application must support the implementation of a cryptographic hash function. • **Req. 5.1.2.2:** The interface between the nodes must support the request/delivery of HMAC packages. **Security function:** • Share critical information between nodes through a standard communication mechanism ensuring the authenticity of both the source of the message and its integrity.

Security Risk Analysis	Attack paths	For this application there exists only one main attack objective:
		Manipulate information and users*:* Any kind of unauthorised manipulation of the message transmitted through a standard communication mechanism. Possible attack methods include:
		• Denial of service
		• Communication interception
		• Corrupt & Fake messages (inject, alter) based on
		  ◦ Replay attack
		  ◦ Cryptographic attack on the security functionality
		  ◦ Brute force attack on the security functionality

Security Risk Analysis	Attack potentials		
Asset (attack)			
Asset (attack) 1: Communication channel between nodes (Denial of service by completely cutting of the communication)			
Elapsed time	Some time is needed to find the correct communication lines (1)		
Expertise	No expertise needed (0)		
Knowledge of system	The system has to be known in a certain way to cut the correct lines (3)		
Window of opportunity	A small window of opportunity is enough to cut through the lines (2)		
Equipment	The equipment is available to all potential attackers (scissor/knife) (0)		
Required attack potential			
Value	6		
Rating	Basic		
Asset (attack) 2: Communication channel between nodes (Denial of service introducing a one bit error for selected messages resulting in an invalid verification)			
Elapsed time	For an attacker with proficient expertise only a little time for planning is required (1)		
Expertise	Proficient expertise is sufficient to figure out the communication protocol and message contents transferred in order to introduce a one bit error of a selected message (3)		
Knowledge of system	For inserting a one bit error an access point of the correct lines must be known (3)		
Window of opportunity	The window of opportunity is a bit stricter to position attack tools (3)		
Equipment	The attack tools that allow to modify a bit of transferred message are require no specialised equipment (0)		
Required attack potential			
Value	10		
Rating	Enhanced Basic		
Asset (attack) 3: Communication channel between nodes (Communication interception by eliminating complete selected messages)			
Elapsed time	This attack requires very few time (0)		
Expertise	No expertise is needed to execute this attack (0)		
Knowledge of system	To intercept by eliminating messages a profound knowledge is sufficient (3)		
Window of opportunity	A moderate opportunity window is enough to get the required access (4)		
Equipment	A certain specialised equipment is required to eliminate selected messages (4)		
Required attack potential			
Value	11		
Rating	Enhanced Basic		

Security Risk Analysis — Attack potentials

Asset (attack) 4: Communication channel between nodes (Causing corrupt and fake messages by replaying messages)

Elapsed time	This attack requires very few time (0)
Expertise	A proficient expertise is needed to perform message replaying (3)
Knowledge of system	A certain knowledge is sufficient to get access to the system (3)
Window of opportunity	The window of opportunity is accordingly moderate (4)
Equipment	A certain specialised equipment is required to replay messages (4)
Required attack potential	
Value	14
Rating	Moderate

Asset (attack) 5: Communication channel between nodes (Causing corrupt and fake messages, e.g. inject or alter a message by attacking the HMAC with a cryptographic attack)

Elapsed time	The HMAC mechanism is expected to be secure for years (19)
Expertise	Expert techniques are at least needed (6)
Knowledge of system	A certain knowledge is sufficient to get access to the system (3)
Window of opportunity	The window of opportunity can be valued as easy if an successful attack can be applied (1)
Equipment	To break the HMAC scheme different types of bespoke equipments are required (9)
Required attack potential	
Value	29
Rating	Beyond High

Asset (attack) 6: Communication channel between nodes (Causing corrupt and fake messages, e.g. inject or alter a message by attacking the HMAC with a brute force attack)

Elapsed time	The HMAC mechanism is expected to be secure for years (19)
Expertise	For a brute force attack no expertise is needed (0)
Knowledge of system	A certain knowledge is sufficient to get access to the system (3)
Window of opportunity	The window of opportunity can be valued as easy if an successful attack can be applied (1)
Equipment	For "efficiently" implementing a brute force attack very specialised hardware is required. (9)
Required attack potential	
Value	32
Rating	Beyond High

Security Risk Analysis	Potential damages	
		The information shared between nodes could be replaced, modified, or removed so the system will not have the right information available to take the decision of activating the emergency brake. We concluded that in the worst case, the environment has to face severe injuries and life threatening injuries (4). Due to the fact that the train is a public transport system the privacy impacts is non-existent (0). In regard to the dependability complications the possible financial losses are accordingly extreme high (4). The operational severity class is quite high but the failure of one train does not impact the functionality of other trains (3).
		Attack objective
		Authenticity of users and integrity of the message
		Safety (S_S) — **4**
		Privacy (S_P) — **0**
		Financial (S_F) — **4**
		Operational (S_O) — **3**

5.2.1.2.5.1 Risk evaluation

In an analogous manner as in the research project EVITA [5], Appendix C1.2 we derived the risk evaluation for the “Safe4Rail” application in the following table.

Attack Objective	Severity (S)	Attack Method	Risk level (R)	Combined attack potential (A)	Asset (attack)	Attack Probability (P)
Manipulate information and users	S_S = 4,C1 S_P = 0 S_F = 4 S_O = 3	Denial of service	R_S = R6 R_F = R6 R_O = R5	5	Asset (attack) 1	5
					Asset (attack) 2	4
		Communication interception	R_S = R5 R_F = R5 R_O = R4	4	Asset (attack) 3	4
		Valid Corrupt & Fake messages (inject, alter)	R_S = R4 R_F = R4 R_O = R3	3	Asset (attack) 4	3
					Asset (attack) 5	1
					Asset (attack) 6	1

Table 6 Risk analysis for “Safe4Rail”

5.2.1.2.6 Dependability characteristics

In Addition to the security analysis the dependability characteristics will be considered accordingly.

<table>
<tr><td>Dependability Objectives</td><td>Dependability goals and assets</td><td>
Diversity:
In dependability-related applications one of the aspects that must be guarantee is the diversity (To reduce the common cause failures in the redundant systems). In this platform it was used software diversity e.g. selection of different operating systems and different programming languages, and hardware diversity e.g. selection of different sensors.
Req. 6.1: Dependability
Req. 6.1.1: The dependability function must guarantee a SIL 4 level.
Req. 6.1.2: Diversity
Req. 6.1.2.1: Different sensor types/manufactures should be used.
Req. 6.1.2.2: Different operating systems should be used.
Req. 6.1.2.3: Different programming languages should be used.
Redundancy:
In order to increase the reliability of the application and at the same time the availability of the dependability function, the dependability-related parts of the application have been triplicate.
Req. 6.1.3: Redundancy
Req. 6.1.3.1: The dependability-related application must be implemented three times in three independent nodes.
Req. 6.1.3.2: All the parts/components of the application that are related to the dependability part of the application must be redundant.
Req. 6.1.3.3: A pair of encoders must be used as an input for each node.
Req. 6.1.3.4: An independent accelerometer must be used as an input for each node.
Independence:
In order to guarantee the right functionality of the dependability-related part of the application, the application has been split in two independent parts, one in charge of the non-dependability part and the other in charge of the dependability part.
Req. 6.1.4: Independence
Req. 6.1.4.1: The independence between the dependability and non-dependability part in the application must be guarantee.
Control Execution:
In order to guarantee the program sequence.
Req. 6.1.5: Program Sequence
Req. 6.1.5.1: The temporal and logical program sequence monitoring must be guaranteed in order to assure the right execution of the application.
</td></tr>
</table>

Dependability Threats	Failure analysis				
		Supervision – Application			
		Function : Supervise Train speed and position for emergency brake			
			Failure modes	**Potential causes:**	**Potential Effect:**
			Incorrect calculation	Failure Software (Systematic)	Emergency Brake not Activated
			No calculations	Failure Software (Systematic)	Emergency Brake not Activated
			Out of time calculations	Failure Software (Systematic)	Emergency Brake not Activated
			Out of order calculations	Failure Software (Systematic)	Emergency Brake not Activated
		Function: Read digital input			
			Failure modes:	**Potential causes:**	**Potential Effect:**
			Incorrect reading	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
			No reading	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
			Out of time reading	Failure Supervision node (Systematic)	Identical to Incorrect Reading
			Out of order reading	Failure Supervision node (Systematic)	Identical to Incorrect Reading
		Function: Write digital output			
			Failure modes:	**Potential causes:**	**Potential Effect:**
			Incorrect writing	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
			No writing	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
			Out of time writing	Failure Supervision node (Systematic)	Identical to Incorrect Reading
			Out of order writing	Failure Supervision node (Systematic)	Identical to Incorrect Reading

Dependability Threats	Failure analysis	**Function:** Send/Receive Ethernet Frame		
		Failure modes:	**Potential causes:**	**Potential Effect:**
		Incorrect frame	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Incorrect frame with correct CRC	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		No frame	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Out of time frame	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Out of order frame	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Supervision – Application		
		Function: Run Software Application		
		Failure modes:	**Potential causes:**	**Potential Effect:**
		Incorrect execution	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		No execution	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Out of time execution	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Out of order execution	Failure Supervision node (Random - Systematic)	Emergency Brake not Activated
		Hardware –Switch Ethernet		
		Function: Route Traffic		
		Failure modes:	**Potential causes:**	**Potential Effect:**
		Frame routed to incorrect destination	Failure Switch (Random – Systematic)	Emergency Brake not Activated
		Incorrect frame	Failure Switch (Random – Systematic) Noise in the communication line	Emergency Brake not Activated
		Incorrect frame with correct CRC	Failure Switch (Random – Systematic)	Emergency Brake not Activated
		No communication	Failure Switch (Random – Systematic)	Emergency Brake not Activated
		Out of time frames	Failure Switch (Random – Systematic)	Emergency Brake not Activated
		Out of order frames	Failure Switch (Random – Systematic)	Emergency Brake not Activated

Dependability Threats	Dependability environment	**Dependability function -Safe4Rail application:** • The application is responsible for estimating and supervising the current speed and position of the train and the activation/deactivation of the emergency brake.
Dependability Risk Analysis	Risk evaluation/ Probability of an accident	If the dependability function detects that one node fails, the application continue working with two nodes, in the case two nodes fails immediately the emergency brake is activated. The average frequency of a dangerous failure of the dependability function for a continuous mode operation, which results in an accident, is the probability of failure per hours (PFH): PFH = $>10^{-9}$ to $< 10^{-8}$ h^{-1} This value comes from the standard IEC 61508 and is assigned to SIL level 4. This value must be derived from the calculation of the failure rate of each component of the platform and with the design techniques and measures selected for the application. Finally you have to check that the value obtained is within the range of allowed values for the PFH according to SIL level to be achieved.

6 Platform specification

Based on a general overview about platform characteristics, which have commonly been identified as important for the platform specification (Section 6.1) this chapter deals with a detailed description of the platform used in the four addressed domains (Section 4.2).

Mainly, the focus is to consider only single platforms. Only in individual cases, systems with multiple platforms are considered.

6.1 Platform Characteristics

In order to get a comprehensive overview about the platforms of the resource constrained embedded systems (RCES) that are used to evaluate the TERESA approach in each domain, we have to take numerous platform characteristics into account. Beside hardware and software characteristics this also includes security and dependability assets. Table 7 lists all characteristics which are taken into account.

The description of the domain's platform specifications can comprise three different kinds of information values related to each of these characteristics:

1. **Associated Application Requirements**

   For each characteristic, all requirements of the application description which are related to it should be referred.

2. **Minimum Platform Requirements**

   The overview about all associated application requirements for a characteristic enables to establish the minimum conditions or requirements necessary for the characteristic. Here, for some characteristics, the description may be empty.

3. **Actual Platform Condition**

   If there does already exist a working platform for the application, the actual condition is described. This is not necessarily the case for the chosen application of all domains, so that the information may not be available. Note, the actual condition can be equal to the nominal condition described within the Minimum Requirements.

For some of the applications chosen by the domains (see Section 4.1) a working platform does already exist. Here the "Actual Platform Condition" can be specified. For the other applications only the "Minimum Platform Requirements" will be described.

	Platform Characteristics	**Description**
Hardware	**Hardware architecture**	Overview about the major hardware components of the platform and component interconnections.
	Processor	For each processor the type, the clock speed, the number of instructions per clock, and the bus width is specified. If internal memory is available, it is stated if there is separated memory for RAM and ROM available, and the corresponding sizes. For more efficient processors, a cache may be integrated.
	Processor type	
	Clock Speed	
	Instructions per clock	
	Bus width	
	Internal memory (RAM/ROM)	Beside the processors, further components such as a Crypto-Co-Processor, external memory and other additional hardware components may be part of the platform.
	Cache	In addition to these characteristics, a figure visualizing the platform components as block diagram is included in the description.
	Instruction set	

Hardware		Crypto-Co-Processor	
Hardware		External memory (RAM/ROM)	
Hardware		Additional required hardware components	
Hardware	**Total working memory size (RAM)**		The platform's overall RAM size.
Hardware	**Total program memory size (ROM)**		The platform's overall ROM size.
Hardware	**Physical Ports**		Specification of all available physical ports and their Bandwidth.
Hardware		Required ports	
Hardware		Optional ports	
Hardware	**Energy consumption**		The platform's energy consumption.
Software	**Logical Interfaces**		Specification of all logical interfaces and of all input and output data and control paths as well as internal data paths.
Software		Data input interface	
Software		Data output interface	
Software		Control input interface	
Software		Control output interface	
Software		Internal Data Interfaces	
Software	**Communication protocols**		Communication protocols used for the different interfaces.
Software	**Operating system**		Specification of the platform's operating systems.
Software	**Languages**		Specification of the used programming languages.
Security	**Security functional components**		Security functional components for instance include mechanisms for secure internal memory, e.g., for a secure storage of private keys, secure boot, and protection mechanisms like debug port security (JTAG, etc.) and read/ write access limitations.
Security	**Physical security**		Physical security mechanisms in order to restrict unauthorised physical access to the contents of the module. E.g. Internal memory (RAM/ ROM), Crypto-Co-Processor, tamper detection mechanisms, etc.
Security	**Attack potential**		Known issues etc.
Dependability	**Dependability functional components**		Security functional components for instance include mechanisms like system redundancy, synchronisation, etc.

Table 7: Platform Characteristics

6.2 Domain Specific Platform Specifications

This deliverable covers the platform specification associated with the industry control application Safe4Rail described in Section 5.2.1. The applications of the other domains are considered in deliverable D6.2.

6.2.1 Industry Control

For the industry control application Safe4Rail a working platform does already exist. The following some figures are given which provide an overview about the platform's hardware components and their ports and interfaces. Furthermore, a table which summarizes the details of the platform components is given that includes information about the associated platform requirements and the actual platform condition.

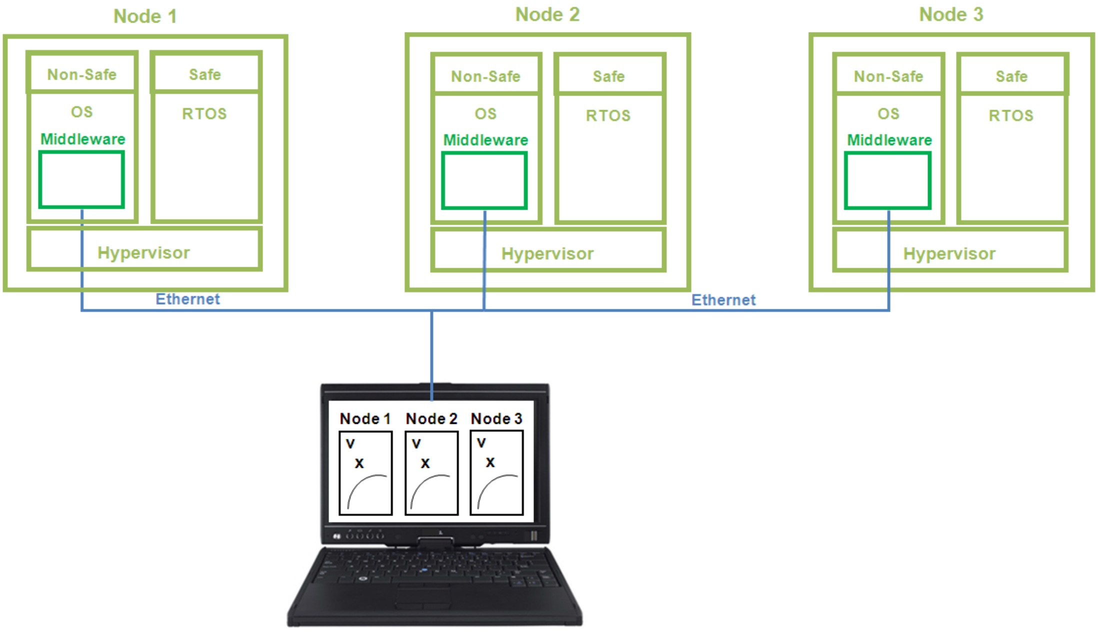

Figure 5 Redundant Supervision Nodes

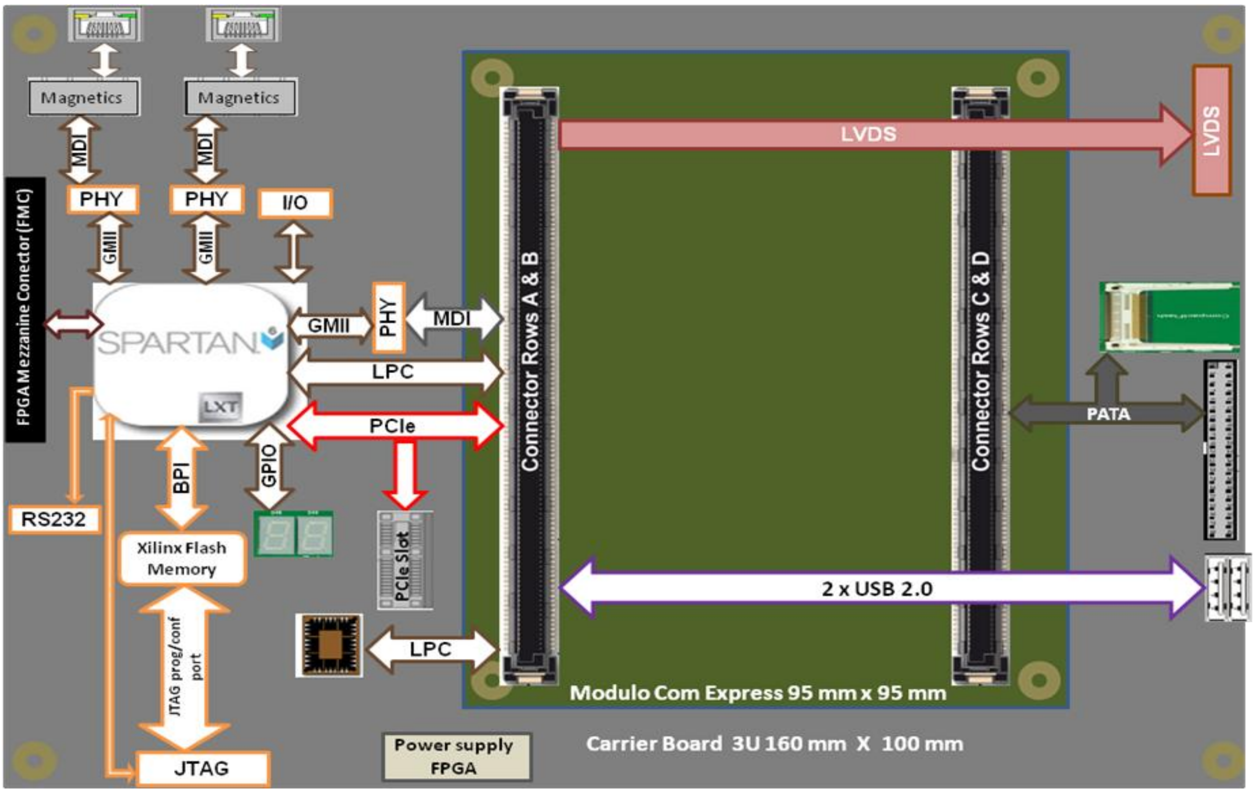

Figure 6 Hardware System Design

	Platform Characteristics			Associated Application Requirements	Actual Platform Condition
Hardware	**Hardware architecture**				
		Processor 1			
			Processor type	**Req. 1.4.3.1** **Req. 1.4.3.2** **Req. 1.4.3.3** **Req. 1.4.3.4** **Req. 5.1.2.1** **Req. 6.1.4.1**	Intel Atom Z530
			Clock Speed	N/A	1.6 GHz
			Instructions per clock	N/A	Maximum two instructions per cycle.
			System controller	To provide the peripherals to the core	System Controller Hub –SCH US15W-
			Bus width	To communicate the Atom with the System controller Hub	533 MHz
			Cache	N/A	512 KB L2 cache
			Instruction set	N/A	IA-32 architecture (Instruction set architecture for Intel's 32 bit architecture)
		Crypto-Co-Processor		N/A	N/A
		Carrier		**Req. 1.1.5.1** **Req. 1.1.5.2**	Proprietary carrier *Support COM Express Module Type 2* *Support Spartan 6 FPGA*
		Module		**Req. 1.1.5.3**	COM Express Module - Conga CA 630707
		External memory			
			RAM	Storage of data and operating system at execution	DDR2 RAM, 1GB
			Flash	**Req. 1.4.4.1**	Compact Flash, 8GB
			Flash	**Req. 1.4.4.2**	Flash, 32 MB Flash, 8 MB

Hardware	Additional required hardware components					
		Programmable Hardware			To establish the communication between nodes To establish external communication	
			FPGA type		**Req. 1.4.2** **Req. 1.4.2.1** **Req. 1.4.2.2**	Spartan 6 LX150T
			Clock Speed		N/A	Oscillator – 100 MHZ
			Internal memory			
				Block RAM	Storage of data at execution	268 blocks of 18Kb
				Distributed RAM	Storage of data at execution	1355 Kb
				Configuration memory	Storage of configuration bits at execution	33.8 Mb
			Communication			
				PCI Express	**Req. 1.4.2.1**	Block for PCI Express – 1
				Ethernet	**Req. 1.4.2.2**	GTP Low- power Transceivers – (3)
		Sensors				
			Speed and position sensors		**Req. 1.1** **Req.1.1.1** **Req. 6.1.3.3** **Req. 6.1.3.4**	Encoders (6) Accelerometers (3) Balise (1)
			Temperature sensor - Carrier		**Req. 1.4.3.1** **Req. 1.4.3.2**	External temperature sensor – LM95245
			Voltage Monitor -Carrier		**Req. 1.4.2.1**	External voltage monitor – ADM1184
			Temperature sensor and Voltage monitor - Module		**Req. 1.4.3.3**	External temperature sensor and monitor voltage – ADT7476 Controller
		Watchdog			**Req. 6.1.5.1**	External microcontroller: ATmega 165p

Hardware	**Total working memory size (RAM)**	Storage of data and operating system at execution	1 GB
	Total program memory size (Flash)	**Req. 1.4.4.1** **Req. 1.4.4.2**	8 GB for processor 40 MB for FPGA
	Physical Interfaces		
	Required Interfaces		
	Internal Interfaces	Communication between RAM DDR2 and the processor.	**Memory Bus**
		Communication between SCH and the processor.	**FSB**
		Communication between BIOS and the processor / Control signal from the processor to FPGA.	**LPC**
		Req. 1.4.5.2	**PCIe**
		Communication between configuration memory (Flash memories) and FPGA	**SPI**
	External Interfaces	**Req. 1.4.5.4**	**IDE**
		Req. 1.4.5.8	**LVDS**
		Req. 1.4.5.1 **Req. 5.1.2.2**	**Ethernet**
		Req. 1.4.5.7	**RS-232**
	Optional Interfaces	**Req. 1.4.5.3**	**FMC**
		Req. 1.4.5.5	**USB**
		Req. 1.4.5.6	**GPIO**
	Energy consumption		**Processor** **Normal mode : 2.3 W** **Sleep mode: 80 – 100 mW**

Software	**Logical Interfaces**		
	Data input interface	To define the dependability inputs for the application	**Data inputs for safety application:** Encoder_1 Encoder_2 Radar_Speed Acceleration Balise_Position Ground_Inclination Next_Balise_Postion **Data inputs for non-safety application:** All the data input for the non-safety application comes from the dependability-related part. For more information see Internal Interfaces section.
	Data output interface	To define the dependability / non-dependability outputs for the application	**Data output for safety application** All the data outputs for the safety application is consider as a control output. To control the emergency brake and to control the status of the safety application. **Data output for non-safety application** O_Max_Speed_Voted O_Max_Distance_Voted O_Position O_Speed
	Control input interface	To define the control dependability / non-dependability inputs for the application	**Control inputs for safety application** Emergency_Brake_Reset Set_Mode New_Balise **Control inputs for non-safety application** All the control input for non-safety application comes from the dependability-related part. More information see Internal Control Interfaces section.
	Control output interface	To define the control dependability / non-dependability outputs for the application	**Control output for safety application** Emergency_Brake_Voted Node_Failure **Control output for non-safety application** Service_Brake_Voted Warning_Voted

Software		Internal data interface	To define the interfaces between nodes	**Internal Data Interfaces between nodes – dependability-related** **Message Control / Black channel** CRC Cycle Source Time_Stamp **Safety Application** VI_Traction_Wheel_Pulses VI_No_Traction_Wheel_Pulses VI_Acceleration VI_Radar_Speed VI_New_Balisse VI_Actual_Balisse_Position VI_Ground_Inclination VI_Next_Balisse_Position VI_Reset_Emergency_Brake VI_Set_Mode VO_Speed VO_Position VE_Emergency_Brake_Local **Internal Data Interface between nodes – non-dependability-related** **Message Control / Black channel - optional** CRC Cycle Source Time_Stamp **Safety Application** VS_Max_Speed VS_Max_Distance VS_Activate_Service_Brake VS_Activate_Warning **Internal Data Interface between dependability and non-dependability part** **Estimated data for position and speed** EB_SEstimated EB_VEstimated

Software	Internal control interface	To define the control interfaces between nodes	**Internal Control data between nodes – dependability-related** **Detect its own failure** Mon_Node_Failure – Self monitoring **Communicate the failure of a node to the others** Mon_NodeB_Failure – Reciprocal monitoring Mon_NodeC_Failure – Reciprocal monitoring **Internal Control data between dependability and non-dependability part** **Communicate the operation mode from the dependability part to the non-dependability part** EB_Set_Mode
	Communication protocols		**Ethercat** – Communication between nodes
	Operating system	To take into account software diversity in the application	**Linux RT** **Windows CE** **On time RTOS_32** **Integrity** Different configuration will be used in the implementation of each node. More information see Dependability Characteristics field.
	Languages	To take into account software diversity in the application	**SySML** **SCADE** **Simulink** A different approach will be used for the implementation of each node. More information see Dependability Characteristics field **VHDL** – Configure interfaces.
Security	**Security functional components**	**Req 5.1.1**	An HMAC function is used to encrypt the message in order to guarantee the integrity of the information and a set of privates keys are used to identify the participants of the communication.
	Physical security	The security analysis revealed that no further physical security measures have to be met. The ratio between additional costs and gained benefit is not sufficient to justify additive physical security.	N/A

Dependability	**Dependability functional components**		
	Processor	**Req. 1.4.3.1** **Req. 1.4.3.2** **Req. 6.1.1**	Intel Atom Z530 • Virtualisation technology – Hypervisor type 1 / Bare metal • HyperThreading
	Operating System	**Req. 1.4.3.1** **Req. 6.1.1** **Req. 6.1.2.2** **Req. 6.1.4.1** To reduce the common cause failure in redundant systems	With Hypervisor • Windows CE + On time RTOS 32 • Linux RT + On time RTOS 32 Without Hypervisor • Integrity Note: Support MMU (Memory management Unit) to partition of memories and to control the access to them.
	Language	**Req. 6.1.1** **Req. 6.1.2.3** To reduce the common cause failure in redundant systems	**SySML –** Modelling and manual and automatic code generation – Non-certified code **SCADE –** Modelling and automatic code generation – Certified code **Simulink –** Modelling and manual and automatic code generation
	Application	To guarantee that the system is determinist	**Input agreement** • Same data type • Same magnitude **Order of execution** • Sequential code • Cyclical execution • Do not use threads for the implementation **Synchronisation (between nodes)** Execution and communication are independent

7 Conclusions

The first step towards an evaluation of successful pattern integration is made in this deliverable. Unlike a separate analysis of the application and platform specification this paper combines the description of both. This is especially benefiting because of the clearly arranged overview of the system and its integration in the whole.

In particular, the industry domain and its selected application “Safe4Rail” are taken into consideration. The security and dependability characteristics as well as a separation of security and dependability aspects are examined.

This deliverable covers the first aspects of hardware and software requirements that are later used to demonstrate the correctness of the pattern integration process. The model driven development allows us to automatically create solutions for each desired system. Its realisation and the attestation of functionality are unambiguous connected to the requirements.

This step is essential for the successive evaluation process. The application specifications need to be verified in regard to the benefit of using patterns. Integration and evaluation of the voting and black channel patterns for the chosen application will take place.

The Deliverable D6.2 applies this study accordingly. Thus, all the aspects that are taken into consideration in this deliverable for the industry control domain are also taken into consideration in the next papers in respect to the home control, automotive and metering domain.

8 Annex A: Biography

[1] M.Scheibel, M.Wolf: "Security Risk Analysis for Vehicular IT Systems – A Business Model for IT Security Measures", in 7th Embedded Security in Cars Workshop (escar 2009), Düsseldorf, Germany, November 24-25, 2009

[2] TERESA, "Deliverable 2.1 Use Cases Application Viewpoint"

[3] ISO/IEC 15408, "Information technology – Security techniques – Evaluation criteria for IT security", (3 parts).

[4] ISO/IEC 18045, "Information technology – Security techniques – Methodology for IT security evaluation".

[5] EVITA E-safety vehicle intrusion protected applications, "Deliverable 2.3 Security requirements for automotive on-board networks based on dark-side scenarios", Version 1.1, December 2009

[6] TERESA, "Deliverable 3.2 Common Engineering Metamodels"