Distributed Real-time Architecture for Mixed Criticality Systems

Preliminary Assessment Report Related to Improving or Calibrating the Technological Results

D 7.3.1

Project Acronym	DREAMS	**Grant Agreement Number**		FP7-ICT-2013.3.4-610640	
Document Version	2.0	**Date**	13-05-2016	**Deliverable No.**	D 7.3.1
Contact Person	David González	**Organisation**		IKERLAN	
Phone	+34 943 71 24 00	**E-Mail**		dgonzalez@ikerlan.es	

Contributors

Name	Partner
David González	IKERLAN
Ibai Sarasola	IKERLAN
Javier Coronel	FENTISS
Anton Trapman	ALSTOM
Albert Rosado	ALSTOM

Table of Contents

Figure Index

Table Index

Glossary

BVR	Base Variability Resolution
BSP	Board Support Package
COTS	Commercial Off-The-Shelf
DREAMS	Distributed REal-Time Architecture for Mixed Criticality Systems
DoW	Description of Work [1]
E/E/PES	Electrical/Electronic/Programmable Electronic Safety-related Systems
FCR	Fault-Containment Region
FPGA	Field Programmable Gate Array
FSoE	Fail Safe over EtherCAT
GL	Germanischer Lloyd
GPOS	General Purpose Operating System
HFT	Hardware Fault Tolerance
HMI	Human-Machine Interface
HW	Hardware
I/O	Input / Output
KPI	Key Performance Indicator
MS	Milestone
OS	Operating System
PCIe	Peripheral Component Interconnect Express
PFH	Probability of Failure per Hour
PL	Performance Level
QoS	Quality of Service
RTOS	Real-Time Operating System
SCL	Safety Communication Layer
SIL	Safety Integrity Level
SW	Software
SWOT	Strengths, Weaknesses, Opportunities y Threats
SCADA	Supervision, Control And Data Acquisition
SCPU	Safety CPU
TSP	Time and Space Partitioning
UC	Use Case
WCET	Worst Case Execution Time
WP	Work Package

Executive Summary

The wind power use case is one of the three DREAMS project demonstrators (along with the avionics and healthcare use cases). This use case describes a distributed mixed criticality system, which combines safety, real-time and non real-time functionalities. It is inspired in the current supervision and control solution for wind turbines, which is enhanced by the inclusion of DREAMS technologies.

This document presents a preliminary assessment report of the wind power demonstrator, with the objective of improving or calibrating the technological results. For that purpose, some of the initially defined KPIs will be calculated or estimated (based on available information), and the degree of fulfillment of project objectives will be evaluated.

1 Introduction

1.1 Context

Alstom Renovables (formerly Alstom Wind and Ecotècnia) is a company, which designs, manufactures, deploys and maintains wind turbines all over the world. With a great knowledge of the wind power market and trends, ALSTOM is facing the market push towards off-shore operation. The road to off-shore introduces new technological challenges, stringent safety requirements and new standards to comply with.

ALSTOM is a key partner in the evaluation of DREAMS project, since it leads one out of the three demonstrators where the technology developed in the project will be used, validated and showcased. The experience accumulated through the development process, as well as the results obtained at the end of the way, will allow calibrating the potential of the DREAMS contribution in the wind market.

This document presents a preliminary evaluation of the wind power demonstrator based on the framework defined in deliverable D7.1.2 [2] to both monitor and evaluate the demonstrator from technological and business perspectives.

1.2 Revisiting wind power use case

As defined in deliverable D7.1.1 [3], the wind power demonstrator is based on the supervision and control system of the off-shore wind turbines. The original system implements two groups of functionalities:

- Control and supervision.
- Human-Machine Interface (HMI) and communications with the SCADA.

The system is executed in the GALILEO platform, and requires several inputs and outputs that are connected through an EtherCAT field bus. GALILEO is a real-time platform developed by ALSTOM and used mainly for the supervision and control system, though it may support other real-time applications such as wind farm control. The last version of this product is GALILEO V5, which is based on a commercial hardware (industrial PC APC 910 [4]) and customized at operating system and software levels. It is based on an x86 dual core processor.

The protection system is in charge of maintaining the wind turbine in a safe state. The main functionality of the protection system is to assure that the design limits of the wind turbine are not exceeded. The protection functions shall be activated as a result of a failure of the control function (running in the supervisory system) or of the effects of an internal or external failure or dangerous event. It should be activated in cases such as:

- Over-speed.
- Generator overload or fault.
- Excessive vibration.
- Abnormal cable twist (due to nacelle rotation by yawing).

Currently, the protection system is implemented in an external module integrated in the EtherCAT ring. This solution lacks the flexibility to implement complex logics since it is only able to handle digital inputs and outputs, and it is mainly a commercial hardware based system.

The demonstrator aims at achieving a higher degree of integration between the supervisory system and the protection system, thus making the overall solution more robust, maintainable, and flexible, while keeping in mind safety and non safety requirements. The demonstrator will integrate the protection system in the GALILEO platform by means of the harmonized platform of the DREAMS project. The demonstrator is defined in such a way that it provides great benefit with respect to the state of the art solution in terms of dependability, it allows validating as many project requirements as possible (but only those relevant to the wind power domain), and it reuses the maximum hardware and software components of the current solutions in the wind power domain from ALSTOM.

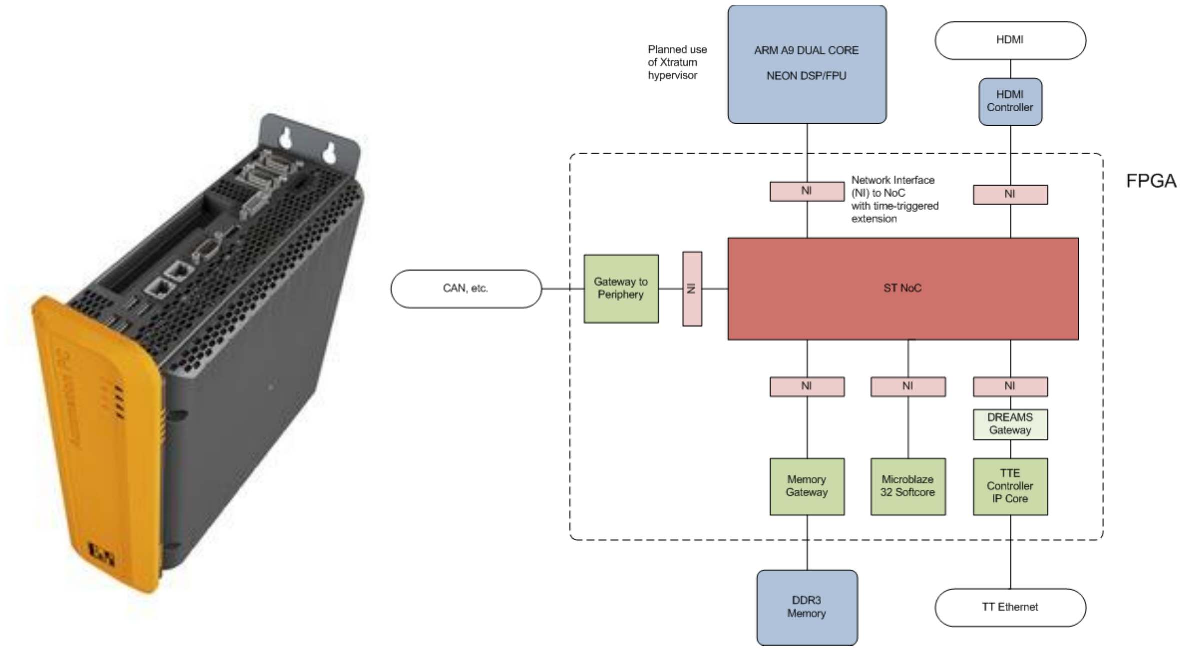


Figure 1: GALILEO V5 and Harmonized Platform

Figure 1 shows the GALILEO V5 platform currently used by ALSTOM for the supervision and control system, along with a diagram of the harmonized platform with the implemented peripherals and services. Both platforms will be interconnected via a PCIe interface.

The harmonized platform is the ZynqTM-7000 board [5], which consist of two ARM Cortex A9 cores and an FPGA where DREAMS technologies and services will be implemented.

As shown in figure 2, the current solution comprises the supervision and control platform (GALILEO), the distributed I/Os connected through EtherCAT, and the protection system, also connected to the field bus by using Safety over EtherCAT protocol (FSoE, Fail Safe over EtherCAT [6]). This solution provides a hardware fault tolerance (HFT) of 0, which means that one failure may cause the loss of the safety function.

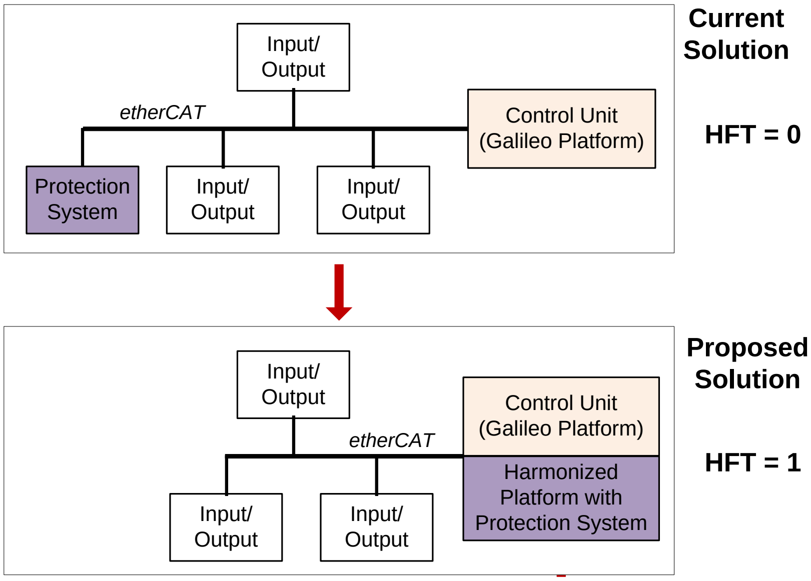


Figure 2: Current solution and proposed solution based on harmonized platform

To achieve higher hardware fault tolerance, the proposed solution integrates the control unit (GALILEO platform) and the harmonized platform via PCIe, keeping the communication with several I/O modules based on EtherCAT field bus. This solution, where the protection system is implemented in the harmonized platform, may be used to achieve heterogeneous redundancy, thus providing an HFT of 1. This increase of HFT is theoretical, and the requirements for on-chip redundancy detailed in IEC-61508-2 Annex E [7] must be met in order to achieve certification.

1.3 Position of the deliverable in the project

This document constitutes the third deliverable in WP7. The status of the whole list of deliverables is detailed in Table 1.

Number	Deliverable title	Status	Delivery date
D7.1.1	Wind power use case specifications	Delivered	M10
D7.1.2	DREAMS wind power evaluation and monitoring plan	Delivered	M24
D7.2.1	Wind power demonstrator	In progress	M42
D7.3.1	Wind power preliminary assessment report	Delivered	M30
D7.3.2	Wind power assessment report	Not started	M45

Table 1: Status of WP7 deliverables

As shown in the table above, the evaluation and monitoring plan (D7.1.2 [2]) was submitted in month 24. This document explains the methodology and indicators to perform monitoring and evaluation of the demonstrator, both during development and after validation.

The design details of the demonstrator, which are required in order to build the necessary arguments to evaluate some of the indicators, are contained in document D7.2.1, which is in progress. This document will be delivered at month 42, but most of the content is already clear for demonstrator developers (at least information related to architecture and design, since development is still being performed and validation will be executed afterwards).

1.4 Relationship to other DREAMS work packages

Figure 3 shows the relationship between the technology work packages and the wind power demonstrator.

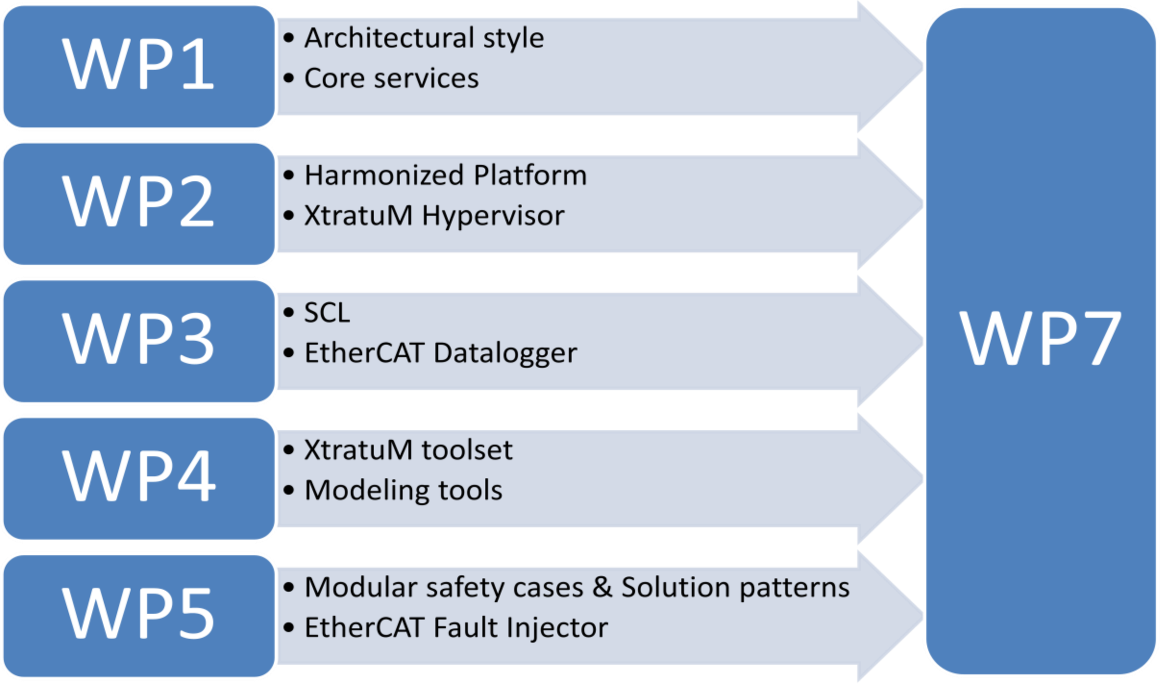


Figure 3: WP7 dependencies with technology WPs

The wind turbine case study integrates a subset of DREAMS technologies listed as follows:

- “WP1: Architectural Style”
  - Subset of core services of the “architectural style” (D1.1.1 [8], D1.2.1 [9])
- “WP2: Multicore Virtualization Technology”
  - Harmonized platform (D2.3.1 [10])
  - XtratuM hypervisor [11] that supports the ‘harmonized platform’, current GALILEO V5 platform and Windows Embedded CE (D2.4.1 [12])
- “WP3: Mixed-Criticality Network”
  - Safety Communication Layer (SCL) (D3.3.1 [13], D3.3.2 [14]).
  - EtherCAT Datalogger (D3.4.1 [15])
- “WP4: Tooling, Scheduling and Analysis”
  - XtratuM toolset is required to design, develop, verify and validate the case study (D4.1.1 [16], D4.1.2 [17])

    - Subset of modelling meta-models and tools (D4.1.1 [16], D4.1.2 [17])
    - SW component model (D4.1.2 [17])
    - HW platform model (D4.1.2 [17])
    - Hypervisor and partition model (D4.1.1 [16], D4.1.2 [17])
    - Safety model (D4.1.2 [17])
    - Variability model (BVR) (D4.3.1 [18])
    - Variability management (BVR), design space exploration (AF3/DSE) and safety-constraint checker (D4.3.2 [19])
- "WP5: Certification, Validation and Verification"
    - Modular safety cases for hypervisor (D5.1.1 [20])
    - COTS multicore device (D5.1.2 [21])
    - Mixed-criticality network (D5.1.3 [22])
    - Solution patterns (D5.3.1 [23])
    - EtherCAT fault injector (D5.2.3 [24])
    - Product line validation strategy (D5.5.2 [25])
    - Product line certification strategy (D5.5.3 [26])

1.5 Objectives of the document

The evaluation plan defined in D7.1.2 [2] focuses on monitoring that requirements are fulfilled in technical work. The plan includes definition of the measures for success of DREAMS project, such as the following (mentioned in Description of Work [1]):

- Assessment of compliance to relevant standards and norms of the proposed solutions.
- Level of dependability and maintainability of the developed building blocks.
- Increased level of time and space separation between virtual partitions
- Reduction of new applications developing time
- Level of cost-effectiveness in the development of prototypes
- Level of reusability of the developed building blocks
- Level of extensibility of developed building blocks

The objectives of this document are:

- To monitor the progress of the wind power demonstrator.
- To apply the evaluation methodology in order to assess intermediate project results. Innovations included in the prototype must be evaluated, and traceability between the technologies developed in DREAMS and the features of the applications exercising each of them must be ensured.

1.6 Structure of the document

The document is organized as follows. Section 1 provides an introduction to the wind power case study and the document itself. Section 2 presents the status of the demonstrator based on the results of the monitoring plan, comparing them with project objectives and milestones. Section 3 contains the results of the preliminary evaluation, presenting the values of the KPIs that are due to be evaluated during development. Finally, section 4 draws conclusions and establishes next steps.

2 Demonstrator monitoring

2.1 Current status of wind power demonstrator

The implementation of the wind power demonstrator is in progress. Once the different elements that compose the demonstrator are available (provided by technology developers), the integration starts and the implementation of demonstrator specific blocks (e.g. application layer) is being scheduled.

The most important achievement in the integration process so far is the deployment of partitions on top of XtratuM Hypervisor in Galileo platform. In February 2016, a meeting was held in Mondragón in order to integrate XtratuM Hypervisor with Galileo platform. The hypervisor was configured in a way that two partitions could run on APC910 hardware: Control partition and Communications partition. Control partition is designed to perform supervision and control tasks while Communications partition will be used as a general purpose execution environment with data servers and communication stacks.

Hardware resources have been assigned as follows:

- Control partition:
  - 3 x Intel 8255 Ethernet Controllers
  - Intel 82574L Ethernet Controller
  - SRAM memory device
  - B&R ADI
  - IDE secondary channel
- Communications partition:
  - Intel 82579 Ethernet Controller
  - USB ports
  - IDE primary channel
  - VGA controller

Control partition requires a Real-Time Operating System (RTOS) to run control algorithms, while Communications partition needs a General Purpose Operating System (GPOS) so that integration of third party communication software is easier. However, for the purpose of the demonstrator, Windows Embedded CE 6.0 will be used in both partitions. This operating system is used by Alstom in the real supervision and control system, and will enable an easy porting of control software to the new platform. It is considered an RTOS, but it supports a subset of Win32 API and there is a decent offer of third party communication software available.

Preliminary images have been created for each one of the partitions. These images have been created by using custom Board Support Packages (BSPs) developed by Fentiss, which are based on CEPC and have been modified so that XtratuM Hypervisor is supported.

These Windows Embedded CE 6.0 images have been compiled along with XtratuM hypervisor into a single image file. This file has been placed together with GRUB bootloader in the storage media device assigned to Control partition (CFast card). Then, the BIOS of the APC910 platform has been configured to boot from the corresponding IDE channel assigned to Control partition (secondary). As a result, XtratuM Hypervisor is properly launched and Control and Communications partitions are loaded.

Connectivity with both partitions has been tested using telnet and FTP protocols. Other preliminary validations tests have also considered graphics support and IDE behaviour. Control partition is headless but Communications partition has been granted graphics support. Therefore, mouse, keyboard and screen usability have been tested. Behaviour of IDE channels has also been validated: they are correctly working and they are only accessible from the configured partition.

At this point, the Windows Embedded CE 6.0 images are in a preliminary stage. They have been used to check the integration of XtratuM Hypervisor with APC910 platform. Nonetheless, these images have to be customized so that application software targeted to this demonstrator can run successfully.

2.1.1 Work in progress

The following table summarizes the components involved in the wind power demonstrator and their current availability. The items listed below include tools, hardware and software components that are required as inputs for the demonstrator.

Component	**Availability**	**Version**
Galileo platform	Yes	APC910 Galileo V5
Harmonized Platform (HW)	Yes	Zynq-7000 ZC706
Harmonized Platform (Bitstream)	Yes	V1
EtherCAT node (HW)	Yes	Beckhoff el9800
EtherCAT node (SW)	No	-
EtherCAT Datalogger (HW)	Yes	-
EtherCAT Datalogger (SW)	Yes	V1
EtherCAT Fault Injector (Board)	Yes	xc7z020clg484-1
EtherCAT Fault Injector (Design)	Yes	V1
XtratuM x86	Yes	XM-X86-VMX-0.1.0
XtratuM ARM	Yes	XM-ARM-2.0.6-DREAMS
Win CE 6.0 BSP for XtratuM x86	Yes	V1
SCL Software Host	No	-
SCL Software Device	No	-
Control application software	Yes	sw-0500rev10

Table 2: Availability of components required for demonstrator (at M30)

Once XtratuM integration with APC910 platform has been accomplished, other tasks have been undertaken. None of them can be marked as fulfilled but they are listed below so that current wind power demonstrator status can be best evaluated.

Windows Embedded CE 6.0 image targeted to the Control partition is being customized so that applications used by Alstom in Galileo platform can run successfully. These applications perform control and supervision tasks, and require some modifications at BSP level and driver-wise.

Modifications at BSP level:

- Unnecessary components that were linked to the BSP have been excluded.
- RAM memory percentage assigned to file system purposes has been updated to 12.5% of the total available memory.
- FTP server has been modified in order to allow downloading through a FTP client a file that is in use (opened) in the device.
- Interruption time base has been changed so that 1ms periodic interrupt is generated.
- Reboot commands from user space have been enabled.

Several drivers have been added:

- ADI driver: it enables access to specific functions of B&R devices through Automation Device Interface (ADI).
- Backwards compatibility drivers: they guarantee backwards compatibility regarding applications created for previous Galileo versions.
- SRAM driver: it enables access to SRAM device.
- 1ms time base driver: it offers applications a 1ms periodic interrupt.
- Watchdog driver.
- EtherCAT driver: EtherCAT master software stack.

Regarding the harmonized platform, the bit stream containing the design for Zynq-7000 ZC706 platform FPGA adapted to the needs of the demonstrator has been provided by USIEGEN. In a first phase of integration, some tests are being performed in order to establish a PCIe link between APC910 platform and Zynq board.

The PCIe interface is going to be used in order to communicate the harmonized platform and Galileo. The harmonized platform will be just another device in the PCI bus and thus it has to be assigned to one of the partitions running on top of APC910 platform (Control partition). This is achieved by updating the XtratuM configuration file and re-generating the binary containing XtratuM and Windows Embedded CE 6.0 images again.

The PCIe link will be used mainly to exchange data between the EtherCAT ring and the harmonized platform, thus allowing safety relevant values to reach their destination. The Control partition owns the EtherCAT interfaces and is responsible to forward relevant data through the PCIe (this is the reason why PCIe device must be assigned to Control partition).

The following diagram shows data flows between Control partition in the Galileo side and ARM and uBlaze processors in the harmonized platform going through the PCIe IP.

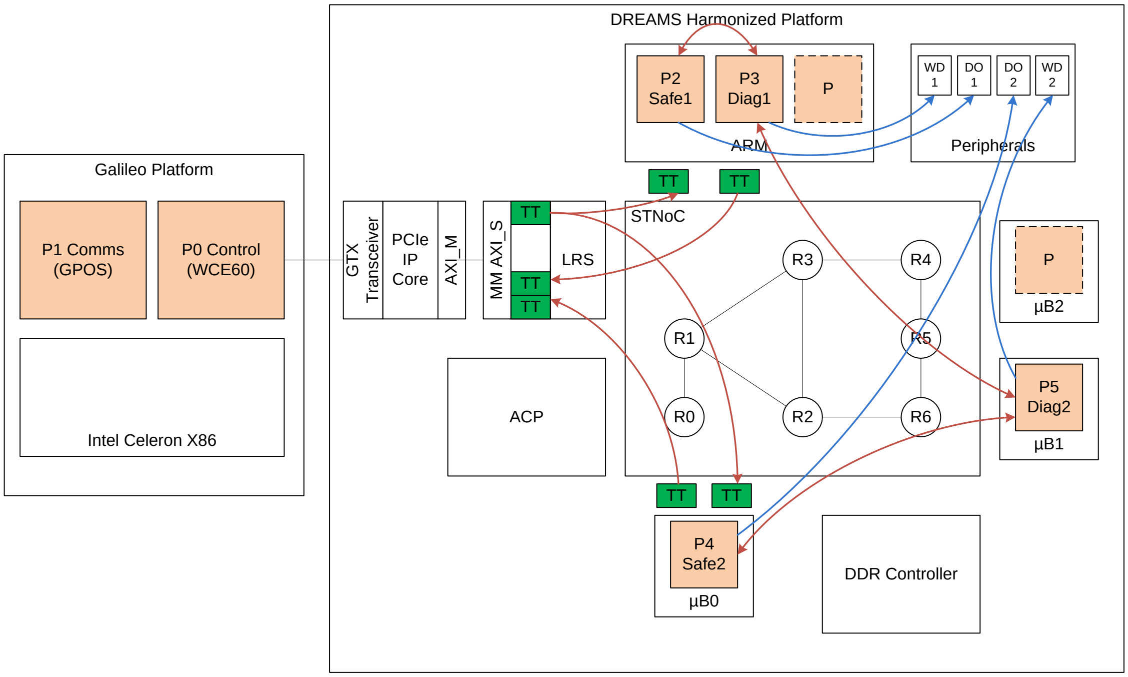


Figure 4: Data flow through PCIe IP and on-chip mixed-criticality network

The evaluation of the harmonized platform does not fit inside the frame of Galileo industrial PC and therefore a PCIe extender has been used to connect both devices. This assembly is illustrated in Figure 5.

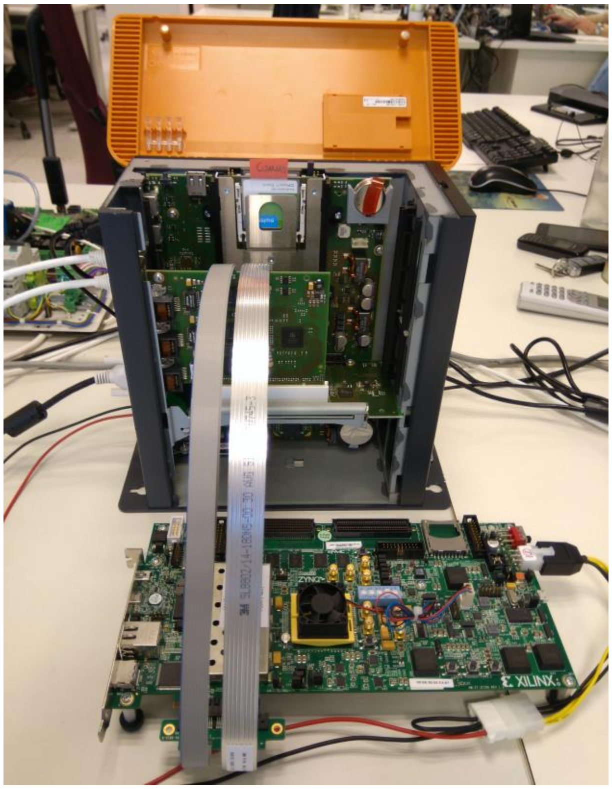


Figure 5: Current status of wind power demonstrator

2.2 Monitoring plan

As explained in deliverable D7.1.2 [2], there are two aspects to be monitored. The first one is the progress of the development process of the demonstrator (described in section 2), which needs to follow a strict schedule in order to respect project deadlines and provide necessary information to other activities in different work packages. The different phases of the development are shown in Figure 6. At this point, the implementation phase is being executed.

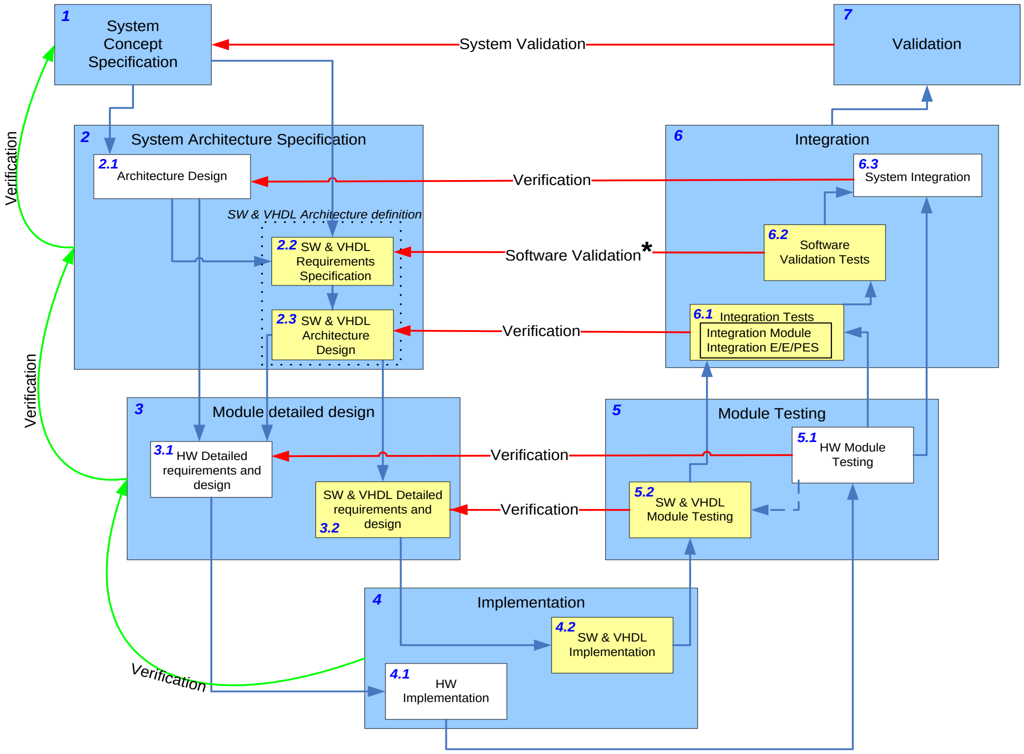


Figure 6: V-model realization according to Ikerlan's IEC-61508 SIL3 FSM [27]

The second aspect to be monitored is the degree of fulfillment of objectives, measures for success and KPIs, presented in section 3. The sooner a deviation is detected, the earlier it can be corrected while maximizing the possibilities to still achieve expected results.

2.3 Development monitoring

Figure 7 shows the main milestones in the development of the wind power demonstrator that shall be monitored to check the correct progress. This timeline is a simplified linear representation of the V-Model development process shown in Figure 6.

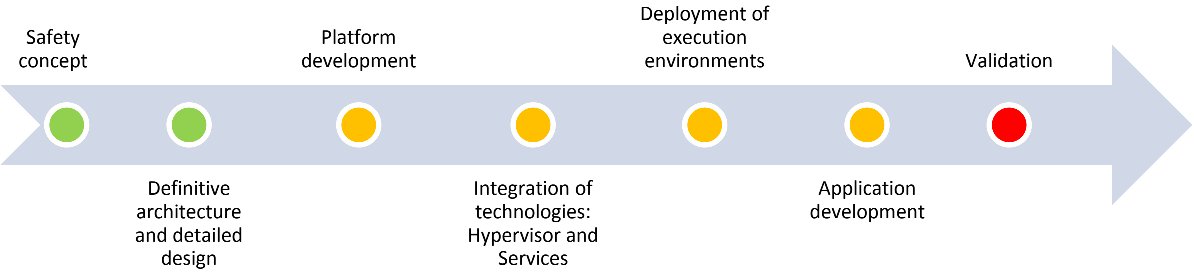

Safety concept
Definitive architecture and detailed design
Platform development
Integration of technologies: Hypervisor and Services
Deployment of execution environments
Application development
Validation

Figure 7: Milestones to be monitored

The first milestone in the development process of the demonstrator is the specification of the safety concept. This document presents the safety argumentation and outlines a first draft of the system architecture, which is refined into a detailed document to obtain the definitive architecture and design, reaching the second milestone. As it can be observed in Figure 7, these two milestones have been reached (they are shown in green colour).

The third milestone consists on developing the core hardware, comprised of the DREAMS harmonized platform properly integrated into GALILEO platform. This is where the development of the DREAMS wind power demonstrator is currently focused. Hardware-wise, APC910 platform is ready and Zynq board FPGA bit stream has been already developed. Once the harmonized platform is integrated, this milestone can be marked as accomplished.

Although some tasks, which belong to the third milestone, have not been finished, a number of activities that are part of the fourth, fifth and sixth milestone have been undertaken.

The fourth milestone focuses on the integration of DREAMS technologies (e.g. virtualization environment, core services, network drivers, SCL, etc.). Integration of APC910 platform with XtratuM Hypervisor has already been performed. This virtualization layer allows creating partitions defined in system architecture to allocate mixed-critical system functionalities with the required properties and resources.

The fifth milestone consists on creating all partitions and successfully deploying execution environments on them. At this point, the application software reused from current supervision and control system is allocated in the corresponding partitions, and new application software will be implemented for the safety protection system. Partitions for Galileo platform have already been created, and uBlaze processors are ready to be programmed. However, partitions on the ARM side of the harmonized platform have not been created yet.

Regarding application layer, the execution environments of the Galileo system are being adapted so that software from current supervision and control system can be ported. The sixth milestone will be reached when application development and porting is completed. This milestone includes migration of current supervision and control applications to the new execution environments, which is a task already in progress However, activities related to new software development, both for Galileo and for harmonized platform, have not started yet.

Finally, the verification and validation plans will be executed to complete the demonstrator development (milestone seven).

Table 3 summarizes the situation previously described.

Milestone	Description	Achieved	Comments
MS1	Safety concept	Yes	Draft Deliverable 7.2.1
MS2	Definitive architecture and detailed design	Yes	Draft Deliverable 7.2.1
MS3	Platform development (hardware)	No	Platform development is in progress. APC910 platform is ready and harmonized platform hardware design has already been delivered. However, harmonized platform integration has not been completed yet.
MS4	Integration of technologies: Hypervisor and Services	No	Integration is in progress. XtratuM Hypervisor has been integrated in APC910 platform but not in ARM Cortex-A9 processor of the harmonized platform. Services such as Safety Communications Layer have not been integrated.
MS5	Deployment of execution environments	No	Deployment of execution environments in progress. Windows CE environment is being customized and no development has been performed regarding ARM/Microblaze execution environments
MS6	Application software development	No	Application software development is in progress. Alstom applications are being migrated to the new execution environment in the control partition. As for new developments, no activity has been started.
MS7	Verification and validation	No	Verification and validation phase has not been started

Table 3: Achievement of wind power demonstrator milestones

2.4 Level of integration of DREAMS technologies

Table 4 shows the degree of integration of DREAMS technologies in the wind power demonstrator. The integration status can be:

- Not started
- In progress
- Completed
- Validated

The table includes the month of the project when the integration of each technology is expected. The time to integrate some of the technologies has already expired according to the initial planning, due to delays in some deliveries. Additionally, it must be taken into account that the development of some technologies can be concluded later than the integration date expected for those technologies in the wind power demonstrator. In those cases, a preliminary version should be integrated to allow continuing with the demonstrator development plan. If this is not possible, an alternative plan should be proposed to minimize the impact of the delays in the demonstrator.

Technology	**Expected at**	**Status**	**Comments**
Architectural style and core services	M26	Completed	Architectural style reflected in the architecture and detailed design of the demonstrator, currently documented in the draft of deliverable D7.2.1. Core services implemented in harmonized platform recently delivered by Siegen University (integration and testing is pending).
Harmonized platform	M26	In progress	Integration of harmonized platform is in progress. FPGA design already developed and delivered, but PCIe connection with APC910 is under development.
XtratuM Hypervisor	M26	In progress	Completed the XtratuM Hypervisor integration in APC910 platform. However, integration in ARM Cortex-A9 of harmonized platform is in progress.
Windows Embedded CE 6.0	M33	In progress	A first version of Windows Embedded CE 6.0 images has been created for Control and Communications partitions. Some details need to be customized for Control partition, and this work is in progress.
Safety Communication Layer	M26	Not started	SCL integration not started either in EtherCAT slave node or in the harmonized platform. The technology is almost ready but needs some adaptations for the specific needs of the wind power demonstrator.
EtherCAT Datalogger	M33	Not started	Datalogger software is ready but the integration in the demonstrator has not started yet, since the EtherCAT communication is still not running.

XtratuM toolset	M30	Completed	XtratuM tools integrated in the demonstrator in order to generate required Hypervisor files for Galileo platform.
Modelling meta-models and tools	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
SW component model	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
HW platform model	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
Hypervisor and partition model	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
Safety model	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
Variability model (BVR)	M34	In progress	Almost completed, pending integration of platform specific models (task T1.6 to be completed in month 34).
Variability management (BVR), design space exploration (AF3/DSE) and safety-constraint checker	M34	In progress	Initial application of methodology in D4.3.2 [19], to be refined in D4.3.3.
Modular safety cases for hypervisor	M22	In progress	Almost completed, waiting for WP4 tools.
COTS multicore device	M26	Completed	The APC910 platform is already integrated in the demonstrator, with XtratuM properly configured to properly handle multicore capabilities.
Mixed-criticality network	M26	In progress	On-chip network is already implemented (though not tested). Off-chip network is not.
Solution patterns	M24	In progress	Some solution patterns are still being developed (e.g. shared memory diagnostic) whereas other ones have already been integrated (e.g. TTEL).
EtherCAT fault injector	M36	Not started	Fault injector development is finished. However, integration with the demonstrator has not started yet, since EtherCAT communication is still not running.
Product line validation strategy	M40	In progress	Pending integration of tasks T4.3.3 and T5.5.3
Product line certification strategy	M40	In progress	Defining GSN diagrams for product line.

Table 4: Status of integration of technologies

3 Preliminary evaluation

3.1 Evaluation methodology

The diagram below shows the whole process regarding wind power demonstrator evaluation. An evaluation plan was defined at M24, being documented in deliverable D7.1.2 [2]. That report aimed at defining the details and criteria of the whole process. It created the basis to monitor the implementation of the wind power demonstrator and also to produce an intermediate evaluation at M30.

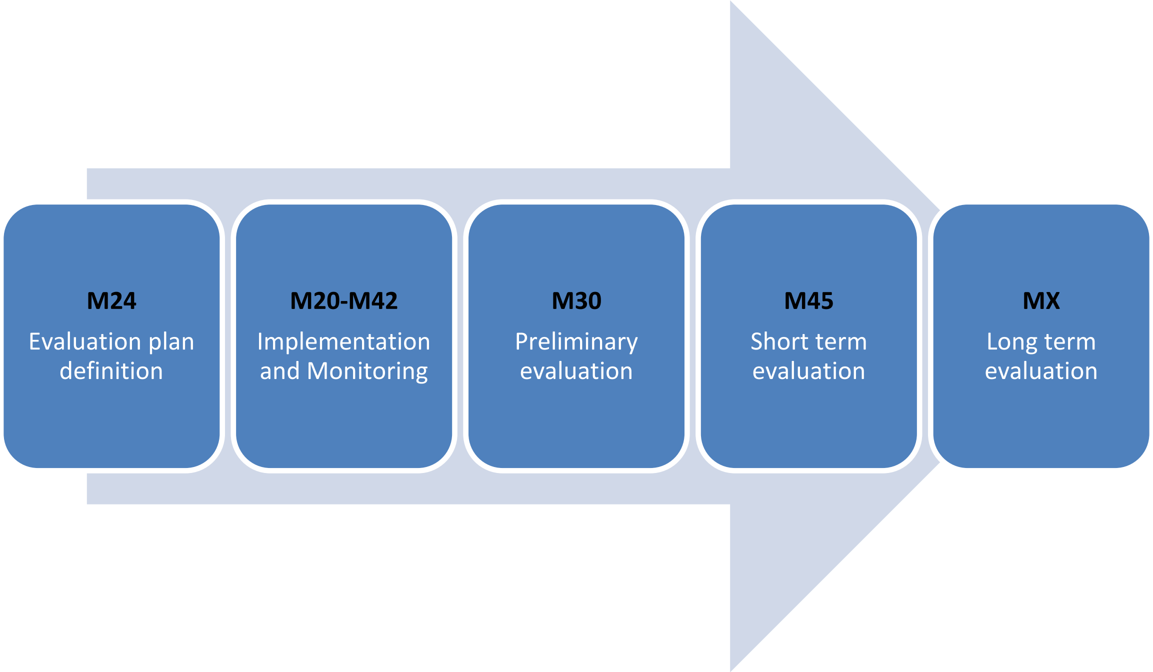
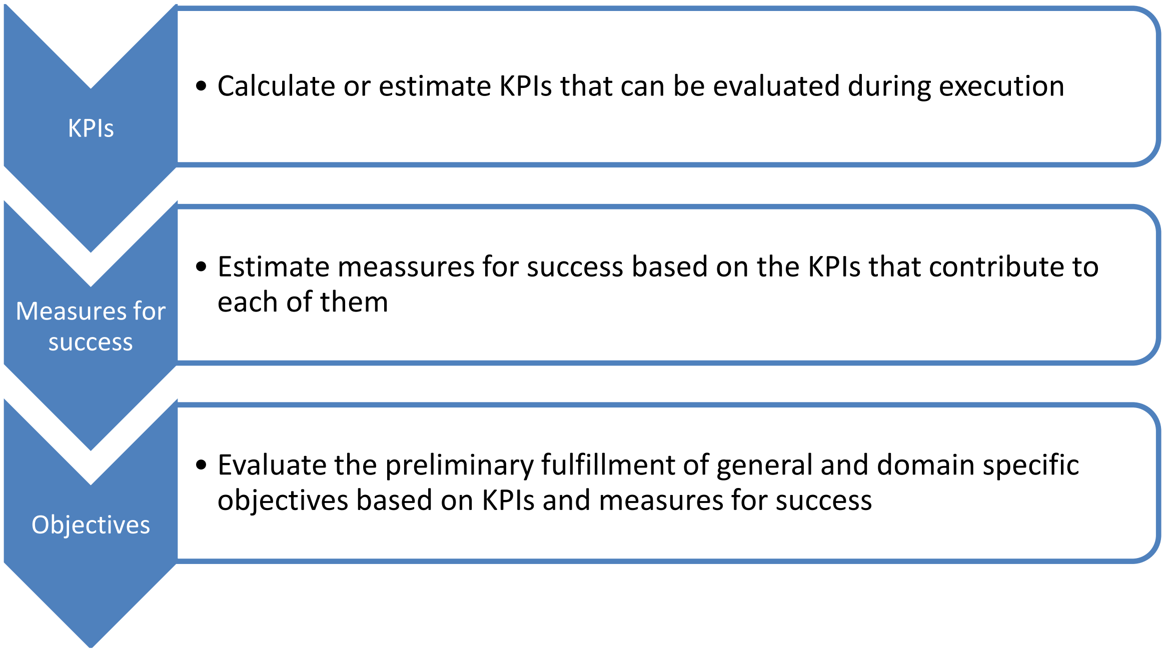


Figure 8: Evaluation process workflow

Deliverable D7.1.2 [2] proposes three evaluation points:

- Preliminary evaluation. This assessment is contained in this document, and it is based on the Key Performance Indicators (KPI) presented in section 3.3. Only KPIs that can be measured during the execution of the demonstrator are included. This evaluation is going to be provided as feedback to the technology work packages for incorporation into the final version of the DREAMS architecture and services. This feedback is going to be used to improve the technological results in WP1-WP5.
- Short term evaluation. This assessment will be started when the demonstrator is completed at M42 and will be based on KPIs used for the preliminary evaluation and KPIs that can be measured by the end of the demonstrator. The objective of the final evaluation is to compare results to expected criteria and to produce a report giving a general picture of project technologies interesting for wind power. This includes overall comments on DREAMS technologies both focusing on the selected use cases and beyond. This report is due to be delivered at M45.
- Long term evaluation. Some KPIs (especially business oriented ones), will need to stress technology and test it in the field, and therefore will not be available right after the demonstrator is finished. That is why a long term evaluation is also proposed.

The objective of the current deliverable is to gather the results of the preliminary evaluation. In order to perform the intermediate evaluation of the wind power demonstrator, a three step process has been defined. First, KPIs that can be evaluated during the execution of the demonstrator are going to be calculated or measured. Secondly, the fulfillment of the measures for success is going to be checked based on the results of the KPIs that contribute to each of them. Finally, the preliminary accomplishment of general and domain specific objectives will be assessed. So as to perform this last step, measures for success that support each objective are going to be checked. Figure 9 summarizes the assessment process defined for the preliminary evaluation of the wind power demonstrator.



- Calculate or estimate KPIs that can be evaluated during execution
- Estimate meassures for success based on the KPIs that contribute to each of them
- Evaluate the preliminary fulfillment of general and domain specific objectives based on KPIs and measures for success

Figure 9: Activities to perform in the preliminary evaluation

3.2 Preliminary assessment of processor timing isolation

Section 3.3 presents the results of different KPIs. There are KPIs related to different technologies involved in the wind power demonstrator. Some of these indicators are specifically related to XtratuM Hypervisor behaviour and the temporal isolation property that must be provided.

Mixed-criticality has increased the interest of researchers and industry for conceptualization and use of multiple components with different dependability, real-time and certification assurance levels (e.g. safety-critical and consumer functionality), which are integrated into a shared multicore computing platform. The development of this demonstrator is an example.

However, in a multi-core system, the access time to hardware resources can be highly variable depending on the concurrent activities of the cores trying to use shared hardware resources of the system. Examples of typical shared resources are the storage resources (such as shared caches and shared dynamic random-access memories) and bandwidth resources such as high-speed buses (e.g. PCIe).

All these factors affect the temporal isolation property of Time and Space Partitioning (TSP) systems. Temporal non-interference would make it possible to support mixed-criticality integration even in the case of safety critical systems with real-time requirements. In mixed criticality systems, this could not be a problem for the non-critical applications but it is for the critical/safety application where the Worst Case Execution Time (WCET) must be deterministic. Although the hypervisor can guarantee the invocation time of partitions, in the case of multi-core hardware temporal non-interference cannot be guaranteed. Additionally, when COTS hardware is used, the level of inter-core interferences is dependent of the specific architecture hardware and electronics components that integrate the platform. Therefore, it is necessary to perform a study on the specific platform to be used in the demonstrator and cover the assessment of KPIs.

With the test presented in the following sections, the effects of parallel execution of two partitions allocated on different cores within the virtualization layer will be assessed. The level of inter-core interference will be analysed to determine how the critical partition should be scheduled, and at the same time, how it can take advantage of the multicore computing performance.

3.2.1 Goals

In this section the software and hardware used in the assessment of inter-core interferences is presented. Some building blocks were not available at the time this assessment was performed; consequently, they were not taken in account in this preliminary analysis. However, these components can be easily integrated into future assessments.

Table 5 presents the hardware and software tools used for the experiment:

Name	Type	Version	Description
Xilinx Zynq 7000	HW	ZC706	Hardware platform based on a dual-core ARM Cortex-A9.
LRS/STNoC/PCIe	HW/SW	Not available	FPGA program in the Zynq-Board. It includes the building blocks for the harmonized platform.
Xilinx Vivado	SW	2014.3	Xilinx development environment for the Zynq Board. It includes GCC 4.8.3, tools to load and debug application and bit streams in the Zynq Board.
XtratuM XM-ARM	SW	2.0.5-DREAMS	Hypervisor and XtratuM development tools.
DRAL	SW	2.0.0	DREAMS Abstraction Layer
Xoncrete	SW	2.6.1	Configuration tool of the system for the generation of the scheduling plan.

Table 5: Resources used in the assessment experiments

Temporal interference is produced when partitions in different cores use shared resources. This evaluation is focused on the temporal impact that a target partition (e.g. safety partition) suffers when another partition (e.g. non-critical partition) is executed in other core and performs intensive access to memory.

The scenario is defined with 3 partitions. Partition 1 (P1) is the target of evaluation and it will be considered as the safety partition. Partition 2 (P2) is considered as non-critical partition, performs accesses to memory and generates the interference on P1. Partition 3 (P3) is in charge of measuring the results. The pseudo code of each partition is presented below.

P1 pseudo code:

```
Defines and initializes variables
Forever
    Access elements of a matrix by columns.
    Increment a counter
End loop
```

P2 pseudo code:

```
Defines and initializes variables
Forever
    Access elements of matrix by rows.
    Increment a counter
End loop
```

P3 pseudo code:

```
Read P1 counter
For (niters)
    Read P1 counter
    Calculate counter increment
    Wait next partition activation
If last plan
    Show the results
Else
    Change scheduling plan
```

The experiment measures the impact of the memory interference when the partition P2, in core 1, overlaps the execution of partition P1.

3.2.2 Experimental setup

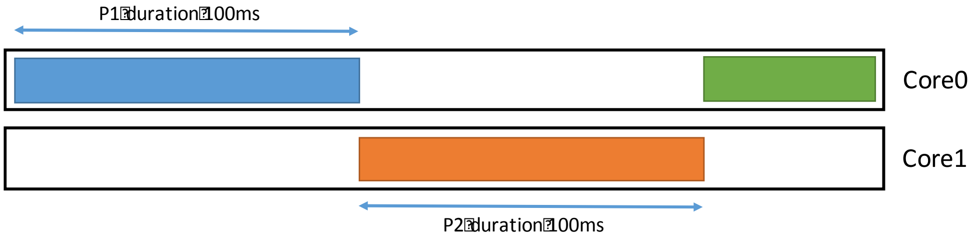
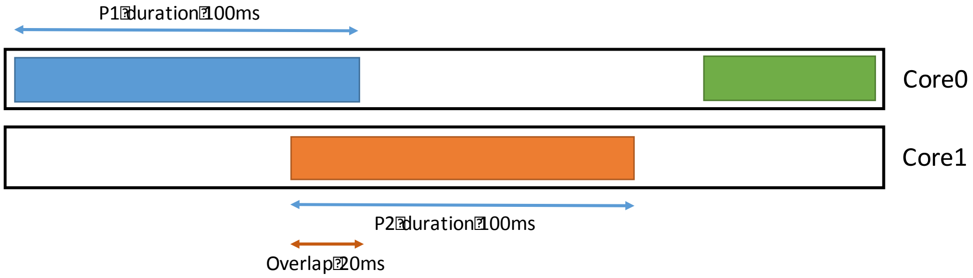
The experiment consists of 4 scenarios with different memory configuration and each scenario defines different scheduling plans.

The four scenarios are:

- Scenario 1 (SC1): P1 and P2 are configured as uncatchable. This forces the maximum and constant impact of the memory interference.
- Scenario 2 (SC2): P1 is configured as cacheable, P2 is configured as uncatchable.
- Scenario 3 (SC3): P1 is configured as uncatchable, P2 is configured as cacheable.
- Scenario 4 (SC4): P1 and P2 are configured as cacheable.

P3 calculates the results after the partition 1 and partition 2 have finished the execution.

Six different scheduling plans are defined for each scenario, where the percentage of overlap of P2 (non-critical) and P1 (critical partition) is increased progressively. Table 6 presents the scheduling plans for all the scenarios.

Plan 1 Reference: S0	
Comments	In this plan, there is not overlap of both partitions. This permits to measure P1 without interference. Partition overlap 0%.
Plan 2 Reference: S20	 P1 duration 100ms Core0 Core1 P2 duration 100ms Overlap 20ms
Comments	In this plan, P2 scheduling is shifted to overlap 20% with P1.

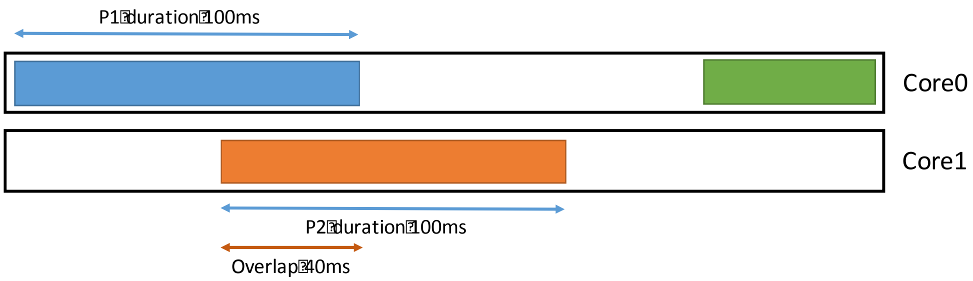
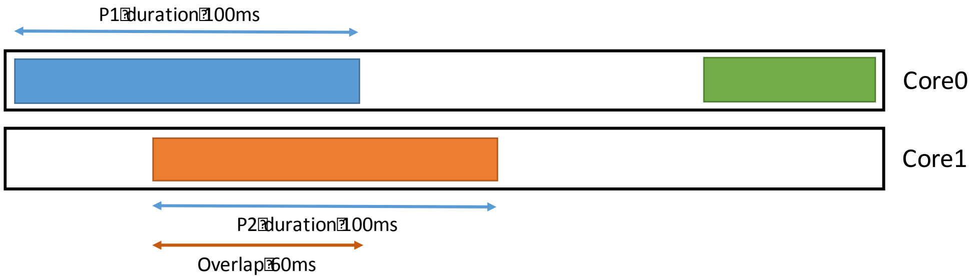
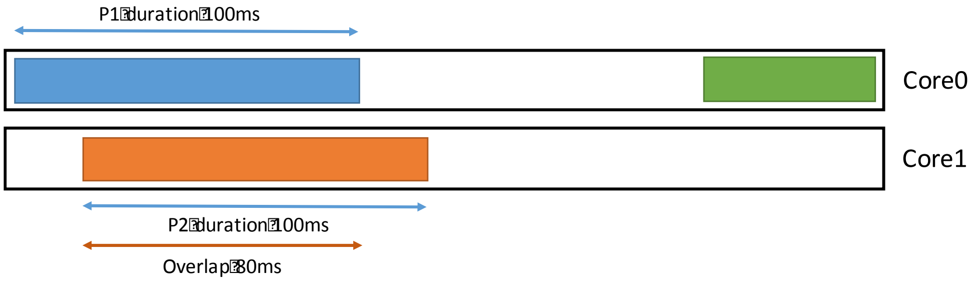
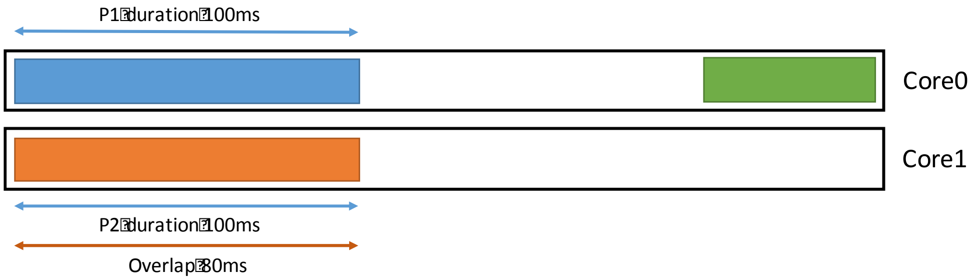
Plan 3 Reference: S40	 P1 duration 100ms Core0 Core1 P2 duration 100ms Overlap 40ms
Comments	In this plan, P2 scheduling is shifted to overlap from 60 ms with P1. It represents 40% of the payload.
Plan 4 Reference: S60	 P1 duration 100ms Core0 Core1 P2 duration 100ms Overlap 60ms
Comments	In this plan, P2 scheduling is shifted to overlap from 40 ms with P1. It represents 60% of the payload.
Plan 5 Reference: S80	 P1 duration 100ms Core0 Core1 P2 duration 100ms Overlap 80ms
Comments	In this plan, P2 scheduling is shifted to overlap from 20 ms with P1. It represents 80% of the payload.
Plan 6 Reference: S100	 P1 duration 100ms Core0 Core1 P2 duration 100ms Overlap 80ms
Comments	In this plan, P2 and P1 are scheduled in parallel. It represents 100% of the payload.

Table 6: Scheduling plans

3.2.3 Experiment metric

The measurements are performed in P1 (safety partition). This partition is constantly accessing to memory in order to perform operations over a matrix and increments a counter in every access. This counter is used to measure the performance loss of P1. The percentage of the differences between the counter values when executed in isolation and when executed concurrently with P2, is the metric used to measure the interference for each of the sub-tests.

To improve the results, the measures are done in intervals. In each interval, P3 reads the number of accesses of P1 and calculates the average.

3.2.4 Results

The scenarios described above have been executed directly on hardware using XtratuM Hypervisor and DRAL. Figure 10 summarizes the impact of inter-core interferences on P1 (safety partition) as result of concurrent access to memory from other partitions. In Figure 10, the X-axis indicates the percentage of overlapping of a non-critical partition with the critical partition, and this percentage goes from 0% to 100%. The Y-axis shows the percentage of overhead in the WCET for the critical partition.

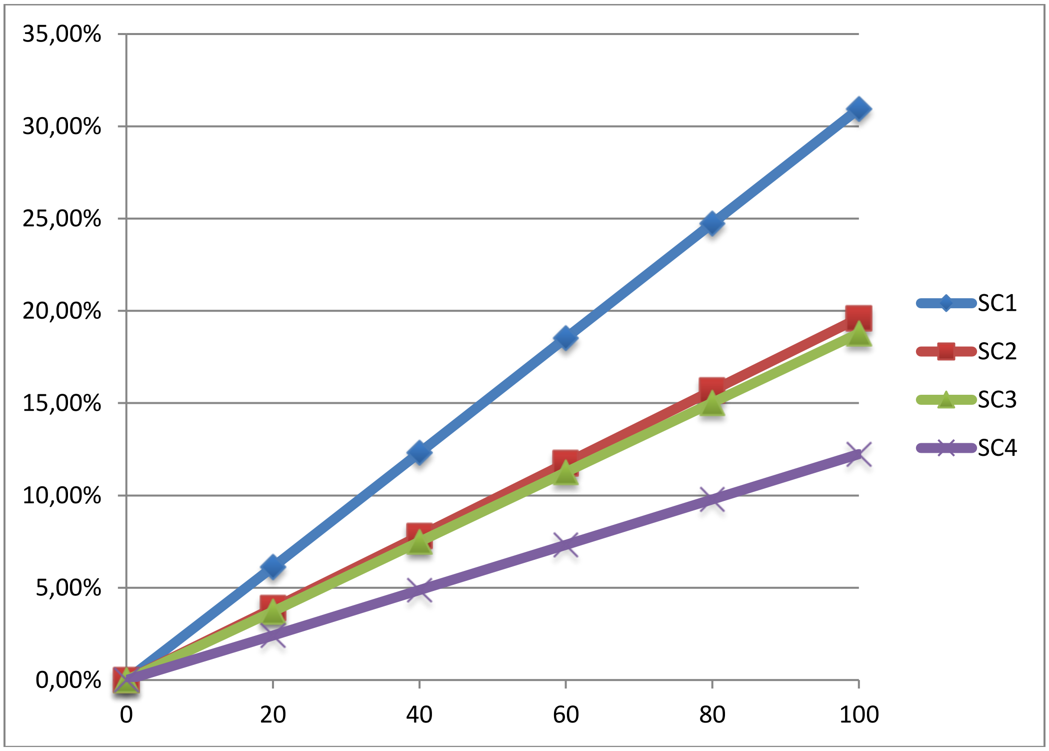


Figure 10: Performance loss caused by memory interference

It is important to point out that this experiment tries to show the existence of the interference and perform an initial evaluation of it.

	SC1	SC2	SC3	SC4
S0	0,00 %	0,00 %	0,00 %	0,00 %
S20	6,12 %	3,88 %	3,72 %	2,42 %
S40	12,33 %	7,82 %	7,49 %	4,88 %
S60	18,53 %	11,75 %	11,26 %	7,33 %
S80	24,74 %	15,68 %	15,03 %	9,78 %
S100	30,95 %	19,61 %	18,79 %	12,24 %

Table 7: Memory interference results

Table 7 shows the nominal percentage of the effect of the interference in the execution time of the safety partition (P1). For instance, in the scenario SC1 an overlap of 40 % (S40) produces an increment in the execution time of P1 of 12,33 %.

3.2.5 Conclusions and recommendations

The results show a lineal relationship between the WCET overhead and the percentage of overlapping in each scenario. This lineal relation is due to the fact that the workload and access to memory in P1 is constant in order to figure the worst case.

Additionally, the higher impact of the interferences occurs in the scenario 1 (SC1), where both partitions have disabled the cache memory and this is translated in several concurrent accesses to the memory bus on almost each execution. On the other hand, the effect of inter-core interference has a reduced impact in the scenario SC3, where both partitions have enabled the cache. In this latter scenario, the access to memory has a different behaviour than SC1, which depends on read and write policies of the cache architecture and it reduces the probability of concurrent access to the shared bus. SC2 and SC3 show an intermediate impact in the WCET of the safety partition, in which scenarios only one of the partitions has the cache enabled.

Although a specific analysis of the safety and non-critical partitions is needed for each specific platform and application, this experiment shows that the interference could be modelled in some conditions and included in the worst case analysis of partition code. Parallel execution of partitions on different cores should be avoided if a safety application has hard real-time requirements. In that case, serialized schedule should be used with no overlapping among partitions. However, in some cases the parallel execution of critical and non-critical partitions could be allowed with a limited overlapping, as long as a detailed execution analysis is performed. This bounded overlapping would allow improving the multicore computing performance.

When whole building blocks are available in the project, a new interference analysis should be performed because new interference sources could appear, e.g. concurrent access to DREAMS ports through FPGA. It should also include real applications running in the partitions.

3.3 Key Performance Indicators (KPIs)

Key Performance Indicators (KPIs) are regarded as a collection of metrics for quantifying the objectives of the project, monitoring its activity progress and assess the expected results.

The KPIs presented in this section are expected to be:

- Objective: it shall be possible to measure them objectively.
- Measurable: it shall be possible to quantify them.
- Relevant to the project: the partners shall confirm their interest.
- Comparable: to the situation of the application use case before using DREAMS approach and technologies.

The performance indicators defined in the following tables will be traced to one or more measure for success. In this preliminary evaluation, they will provide quantitative information to support the qualitative evaluation of every measure for success. Some of the measures for success are not traced to any KPI, since there may be no quantitative data that could support the conclusion.

The KPIs are classified into three subsets: KPIs measurable at any time during the execution of the project, KPIs only measurable at the end of the project, and KPIs that may only be obtained years after the project.

Some examples of KPIs, which could be calculated during the project, are:

- Number of supported core architectures.
- Number of supported operating systems.

KPIs to be calculated at the completion of the project could be such as the following:

- Demonstrator development effort/cost.
- Percentage of DREAMS building blocks used by the demonstrator.

Examples of KPIs to be calculated years after the project could be:

- Time-to-market reduction of a mixed-criticality system based on DREAMS architecture and technologies.
- Cost reduction in variability management of a product developed by using DREAMS architecture and technologies.

Table 8 lists and describes all KPIs of the project, and traces all of them to the measures for success they aim at providing arguments for evaluation. The last column indicates when this metric can be obtained:

- D: During the development of the demonstrator. The KPIs marked with ‘D’ can be evaluated in the preliminary and final reports.
- E: When the development is finished (by the End of the project). These KPIs can only be evaluated in the final report.
- A: After some experience with the technology (After the project). These KPIs cannot be objectively evaluated at the end of the project, since some experience with the technology is needed. Estimation will be provided in the final report.

ID	KPI	Description	Measure for Success	Time
1	Achievable Performance Level	Maximum achievable Performance Level (e.g. PLd, PLe) according to ISO-13849 [28] [29]	1.1., 6.1, 6.2	D
2	Achievable Safety Integrity Level	Maximum achievable Safety Integrity Level (e.g. SIL2, SIL3) according to IEC-61508 [30] [7] [31]	1.1., 6.1, 6.2	D
3	Achievable Hardware Fault Tolerance	Maximum achievable Hardware Fault Tolerance based on DREAMS architecture	1.1., 6.1, 6.2	D
4	Validated support for key real-time OS	(Boolean) The platform supports integration of Windows Embedded CE 6.0 to be used as the OS for the supervision and control system	1.2, 8.1	D
5	Minimum closed-loop cycle time	Minimum period to execute real-time threads of the supervision and control system, containing closed-loop regulation algorithms	1.2	D
6	Minimum field bus cycle time	Minimum period to obtain input values and apply output values in the field bus modules, in both non-safety and safety data (safety data needs an additional software layer)	1.2	D
7	Maximum jitter	Bounded value for jitter in the execution of the most critical real-time thread	1.2	D
8	Fault containment by construction	(Boolean) The certification body accepts evidences to demonstrate fault containment by construction	1.3,1.1	D
9	Percentage of integrated core services	Percentage of core services of DREAMS integrated in the wind power demonstrator	1.6	D
10	Percentage of domain services portable to new architecture	Percentage of services of the subsystems that are going to be integrated in the demonstrator which are either ported or portable to the new platform (ideally 100 %)	1.6	E
11	Percentage of system architecture/design modelled	Percentage of the system architecture and design that is able to be modelled with the tools developed in DREAMS	1.8	D
12	Percentage of software application modelled	Percentage of the application software that is able to be modelled with the tools developed in DREAMS	1.8	D
13	Models complexity	(Boolean) Wind power domain experts appreciate an easier complexity management by using modelling tools and methods	1.9	D
14	Temporal and spatial isolation by construction	(Boolean) The safety concept (supported by the verification plan) demonstrates that the architecture provides temporal and spatial isolation of partitions by construction	2.1	D
15	Bounded temporal interference (network)	Delay introduced in the safety-related communications when heavy non-safety traffic is generated in the network		E

16	Bounded temporal interference (processing)	Delay introduced in the critical thread of the safety-related partition when heavy processing load is generated in neighbouring non-safety partitions	2.1	E
17	Bounded temporal interference (resources access rate)	Delay introduced in the access to resources (memory) by the safety-related partition when heavy resource consumption is required by neighbouring non-safety partitions	2.1,2.2	E
18	Resources access rate penalty	Access rate penalty measured in the access to resources (memory) by the safety-related partition when heavy resource consumption is required by neighbouring non-safety partitions	2.2	E
19	Percentage of out of the box gateways	Percentage of gateways required to connect on-chip and off-chip networks that are provided “out of the box” and not specifically developed for demonstrator	2.3	D
20	Sensor-to-partition latency	Latency between a value is read at the sensor and delivered at the partition where it is going to be processed	0	E
21	Sensor-to-partition jitter	Jitter in the time between a value is read at the sensor and delivered at the partition where it is going to be processed	2.5	E
22	Development time reduction	Reduction in development time of the mixed-criticality system in comparison with the development time of equivalent conventional systems	4.1	E/A
23	Percentage of development steps covered by tools in demonstrator	Percentage of development steps where DREAMS tools provide support in the demonstrator, in one or more of the following aspects: safety, timing, energy, variability	0	D
24	Percentage of development steps potentially covered by tools in wind power	Percentage of development steps where DREAMS tools could potentially provide support in a wind power solution, in one or more of the following aspects: safety, timing, energy, variability	0	E
25	Percentage of automatically executable transformations	Percentage of automatically executed transformations between consecutive development steps provided by tools	4.3	E
26	Effort reduction for addition or modification of features	Estimation of the variation in the assessment effort when changing the safety integrity requirement	5.1	A

27	Effort reduction for replacement of components	Estimation of the re-use of integration evidences, and required additional assessment effort	5.2	D
28	Broadening of the design space	Cost analysis for the rework needed to integrate more abstract descriptions of the components, in order to improve portability and product line evolution	5.3	A
29	Pre-certifiable patterns for aspect features	Percentage of replaceable patterns that provide the safety features	5.5	A
30	Adaptability to evolution of product and standards	(Boolean) The approach provides required adaptability for evolution of product and standards	5.6	A
31	Cost reduction in development of prototype	Cost reduction in the development of the prototype of the demonstrator, compared to the sum of the cost of prototyping the subsystems now integrated	6.3	E
32	Reduction of prototype development time	Development time reduction in the prototype of the demonstrator, compared to the prototyping of the subsystems now integrated	6.3	E
33	Reusability of building blocks in other power generation domain	Percentage of demonstrator building blocks that are straightforward reusable in other domains, and percentage of building blocks that are reusable with small adaptations	6.4, 6.5	E
34	Percentage of public information coming from the demonstrator	Percentage of the contents in website and repository that are part or use demonstrator material	7.1	E
35	Percentage of training material coming from the demonstrator	Percentage of the training material that are part or use demonstrator material	7.2	E
36	Percentage of support for relevant OS (RTOS and GPOS)	Percentage of operating systems that are relevant for the wind power domain (they must be listed) that are available to be deployed on the demonstrator platform	8.1	D
37	Scalability gap	Available resources to scale up the demonstrator to support additional features (in terms of free cores, network throughput, etc.)	8.2	E
38	Reduction in certification cost	Cost reduction in certification due to certification facilities provided (modular safety cases, reference architecture, compliant items, etc.)	9.1	A
39	Reduction in re-certification cost	Cost reduction in re-certification due to certification facilities provided (modular safety cases, reference architecture, compliant items, etc.)	9.2	A

40	Reduction in criticality level up	Cost reduction in the certification of a function which is integrated in the system as a non-safety component and shall be certified (e.g. supervision and control)	9.3	A
41	Safe data availability	Number of variables that the safety partition can safely handle through the mixed-criticality network to use in the safety functions	10.1	D
42	Safe algorithm programming flexibility	(Boolean) The programming of the algorithms of the safety functions does not have any limitation in terms of number of sentences, inputs, outputs, etc.	10.1	D
43	Network flexibility and scalability	(Boolean) Mixed-criticality network allows adding or removing elements and scale the number of nodes	11.1	D
44	Network validation supported by tools	(Boolean) The validation of the mixed-criticality network can be done by using the tools provided in DREAMS	11.2	E
45	Percentage of compatible development steps	Percentage of development steps defined in the DREAMS methodology that are compatible with current process in wind power domain	12.1	E
46	Percentage of variability sources successfully handled	Percentage of variability sources that have been successfully handled through DREAMS methods and tools	12.2	E
47	Reduction of variability adaptation time	Reduction in adaptation time to deliver the system after applying a variability point	12.2	A

Table 8: Key Performance Indicators

Table 9 collects the values of the KPIs that can be evaluated at this point of the project (those marked with 'D' in Table 8). Some values have been calculated while others have been estimated. Additional information is provided in the comments column.

ID	KPI	Goal	Value	Comments
1	Achievable Performance Level	PLe	PLe	Heterogeneous on-chip redundancy of safety relevant logic. Category 3 requirements according to ISO-13849-1 [28] section 6.2.6. $MTTF_d$ high and DC_{avg} medium to high. Full details in safety concept described in D7.2.1.
2	Achievable Safety Integrity Level	SIL3	SIL3	Heterogeneous on-chip redundancy of safety relevant logic. Hardware Fault Tolerance of 1 and SFF medium (90 % -< 99 %). Techniques and measures to control systematic and random failures. Full details in safety concept described in D7.2.1.
3	Achievable Hardware Fault Tolerance	1	1	Heterogeneous on-chip redundancy of safety relevant logic in ARM and uBlaze. This HFT is theoretical, and the requirements for on-chip redundancy detailed in IEC-61508-2 [7] Annex E must be met in order to achieve certification.
4	Validated support for key real-time OS	Yes	Yes	The support for Windows Embedded CE 6.0, which is the key real-time OS for wind power demonstrator is preliminary validated, though the final images are still being created (some adjustments are necessary).
5	Minimum closed-loop cycle time	1ms	1ms	In the Galileo side, the BSP of Windows Embedded CE 6.0 has been modified so that 1 ms interrupts are generated, thus allowing a time base of 1 ms to schedule tasks. In the harmonized platform, shorter times are possible since lighter operating systems are to be used, or even no operating system at all.
6	Minimum field bus cycle time	1 ms	1 ms	The minimum achievable cycle time with the EtherCAT master software stack is 1 ms.
7	Maximum jitter	10 us	5 us	Isolated executions of critical partition guarantee not exceed this value. It must consider the recommendations from section 3.2.
8	Fault containment by construction	Yes	Yes	Specific documentation for each building block and mainly, the deliverables D2.3.1 [10], D5.1.1 [20] and D5.1.2 [21] could represent enough evidences to consider fault containment by construction.

9	Percentage of integrated core services	50 %	75 %	Based on core services listed in D1.2.1 [9] (page 18), three out of four core services have already been implemented in the harmonized platform. The one missing is the core service defined as “Integrated resource management for time and space partitioning”. However, none of them has been tested in the demonstrator since the harmonized platform integration has not been completed.
11	Percentage of system architecture/design modelled	70 %	75 %	The following meta-model elements from D1.4.1 [32] have been used (a) SW Components: Application Components, Virtual Ports, Virtual Channels (b) HW elements: Cluster, Nodes, Tiles, Cores, RAM/ROM, Comm. Networks, Watchdog, Clocks (c) System SW: Hypervisors, Partitions (d) Deployment Components Real-time and energy consumption models have not been used so far.
12	Percentage of software application modelled	50 %	0 %	Functional modelling of the application software is outside the scope of WP4 toolset. However, all entities of DREAMS software components defined in D1.4.1 [32] (component, ports, channels, etc.) have been used in the corresponding phase of wind power demonstrator.
13	Models complexity	Yes	Yes	With WP4 Toolset the expert has an integrated view of all models.
14	Temporal and spatial isolation by construction	Yes	Yes	Spatial isolation is guaranteed by the hypervisor and these evidences can be extracted from specific documentation of the virtualization layer and D2.3.1 [10] and D5.1.1 [20]. Temporal isolation could also be guaranteed by the hypervisor, if the conclusions of the preliminary assessment presented in section 3.2 are taken into account in the building of the system.
19	Percentage of out of the box gateways	50 %	0 %	The only gateway required is PCIe IP in the FPGA, and it has been specifically integrated for the wind power demonstrator.
23	Percentage of development steps covered by tools in demonstrator	60 %	50 %	Almost all development steps completed so far (50 % of the V-cycle) have been supported by tools coming from WP4, except application software design and development.

27	Effort reduction for replacement of components	30 %	0 %	This KPI cannot be properly estimated at the moment. The use of models will clearly help to replace components and calculate the impact in the rest of the system, but this effort reduction can hardly be estimated with currently available information.
36	Percentage of support for relevant OS (RTOS and GPOS)	75 %	75 %	The following relevant operating systems are supported: Windows Embedded CE 6.0 (as RTOS and GPOS), Partikle (RTOS) and XAL. However, Windows Embedded Standard 7 is not supported, which would be the preferred option for the Communications partition. Therefore, 3 out of 4 relevant operating systems are supported.
41	Safe data availability	>10	64 bytes	Up to 64 bytes of safety relevant data is encapsulated in every frame. Then, the maximum number of safety relevant variables to be transmitted depends on the number of bytes consumed by the variable types. The total number of bytes could be increased if necessary.
42	Safe algorithm programming flexibility	Yes	Yes	There is no limitation regarding algorithm complexity in the programming of ARM and uBlaze safety partitions.
43	Network flexibility and scalability	Yes	Yes	The off-chip mixed-criticality network allows adding or removing elements and scaling the number of nodes. However, the on-chip network is not that flexible since a new FPGA design needs to be provided.

Table 9: KPIs evaluated at M30

3.4 Measures for success and objectives

The following tables present the progress towards the completion of measure for success and project objectives by analyzing available information at this point of the project. The measures for success are marked with green color if the progress is positive, orange if there is not enough information to evaluate it, and red if the progress is negative.

Objective 1: Architectural style and modelling methods based on waistline structure of platform services			
Measure for success	KPIs	Evaluation	
1.1 Safety	1,2,3,8	The safety objectives of the project are expected to be fulfilled. However, theoretical conclusions obtained so far shall be implemented in the demonstrator in order to build solid certification arguments.	🟩
1.2 Real-time	4,5,6,7	The relevant RTOS are supported and the timing requirements are met according to tests carried out in this preliminary evaluation. Therefore real-time objectives will be achieved.	🟩
1.3 Fault containment	8	Some evidences to argument fault containment property are already provided in other deliverables as detailed in KPI 8. The objective is achieved.	🟩
1.4 Timely adaptation	▨	▨	▨
1.5 Security	▨	▨	▨
1.6 Domain-independent core services	9,10	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
1.7 Modular architecture		The architectural services have been successfully customized and refined for the wind power demonstrator.	🟩
1.8 Models with fine grained analysis/scheduling	11,12	The range of defined models covers the majority of the development process. However, some aspects (e.g. real-time, energy consumption) have still not been modelled, so this measure for success will be assessed in the final evaluation.	🟧
1.9 Models complexity	13	According to KPI values obtained in the preliminary evaluation (M30), this measure for success is expected to be fulfilled.	🟩
1.10 Models completeness		The range of defined models covers the majority of the development process. However, some aspects (e.g. real-time, energy consumption) have still not been modelled, so this measure for success will be assessed in the final evaluation.	🟧
Objective evaluation			
The preliminary evaluation of this objective is very positive, but there is some information missing. The final evaluation will cover it in detail.			

Table 10: Template for evaluation of objective 1

Objective 2: Virtualization technologies to achieve security, safety, real-time performance as well as data, safety, energy and system integrity networked multi-core chips			
Measure for success	KPIs	Evaluation	
2.1 Isolation	14,15, 16,17	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
2.2 Reduced bank conflicts	17,18	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
2.3 Gateways	19	With the information available at this point of the project, this measure for success is not fulfilled.	🟥
2.4 Reduction of latencies	20	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
2.5 Reduction of jitter	21	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
2.6 Reconfiguration	▨	▨	▨
2.7 Security	▨	▨	▨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is very little information at this point. Testing of the final demonstrator will be required in order to measure some of the proposed KPIs.			

Table 11: Template for evaluation of objective 2

Objective 3: Adaptation strategies for mixed-criticality systems to deal with unpredictable environment situations, resource fluctuations and the occurrence of faults			
Measure for success	KPIs	Evaluation	
3.1 Variability		Testing over the final demonstrator will be required to validate that faults occurring in other partitions/applications do not compromise the safety level of the highest criticality application (protection system).	🟧
3.2 Criticality spectrum		The architecture and technology provide the coverage of the required criticality levels for the wind power demonstrator.	🟩
3.3 Applicability		This measure for success cannot be evaluated yet.	🟧
3.4 Efficiency	▨	▨	▨
3.5 Scalability		Scalability is not going to be evaluated at implementation level. However, the tools defined in WP4 allow scalability at modelling level.	🟩
3.6 Portability		Many of the drivers used in the supervision and control system will be used in the demonstrator with minor adaptation. In this sense, portability can be positively assessed.	🟩
Objective evaluation			
The preliminary evaluation of this objective is positive, but there is some information missing. The final evaluation will cover it in detail.			

Table 12: Template for evaluation of objective 3

Objective 4: Development methodology and tools based on model-driven engineering

Measure for success	KPIs	Evaluation	
4.1 Development process	22	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
4.2 Development steps covered by tools	23,24	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
4.3 Automatically executable transformations	25	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 13: Template for evaluation of objective 4

Objective 5: Certification and mixed-criticality product lines			
Measure for success	KPIs	Evaluation	
5.1 Modular safety-case	26	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
5.2 Safety-case modularity	27	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
5.3 Architectural support	28	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
5.4 Configuration optimization	▨	▨	▨
5.5 Variability	29	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
5.6 Domains and market features	30	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 14: Template for evaluation of objective 5

Objective 6: Feasibility of DREAMS architecture in real-world scenarios			
Measure for success	KPIs	Evaluation	
6.1 Separation	1,2,3	According to KPI values obtained in the preliminary evaluation, the level of time and space separation obtained in the demonstrator is enough to perform certification.	🟩
6.2 Standard compliance	1,2,3	The preliminary safety concept that will be described in D7.2.1 presents the arguments to demonstrate certifiability according to the relevant standards.	🟩
6.3 Cost	31,32	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
6.4 Reusability	33	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
6.5 Extensibility	33	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
Objective evaluation			
Some of the measures for success cannot be evaluated at this point. However, available data suggests a positive progress towards the completion of this objective.			

Table 15: Template for evaluation of objective 6

Objective 7: Promoting widespread adoption and community building			
Measure for success	KPIs	Evaluation	
7.1 Community infrastructure	34	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
7.2 Training material	35	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
7.3 Standardization		This measure for success cannot be evaluated yet.	🟨
7.4 Roadmap		This measure for success cannot be evaluated yet.	🟨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 16: Template for evaluation of objective 7

Objective 8: Enable higher integration of mixed-criticality systems providing scalability and composability			
Measure for success	KPIs	Evaluation	
8.1 Support for integration of criticality levels	4,36	The demonstrator successfully integrates three groups of functionalities with different criticality levels. The assessment of this measure for success is positive, but must be confirmed in the final evaluation.	🟩
8.2 Demonstrator scalability	37	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
Objective evaluation			
The preliminary evaluation of this objective is positive, but there is some information missing. The final evaluation will cover it in detail.			

Table 17: Template for evaluation of objective 8

Objective 9: Reduce certification / effort for safety protection system			
Measure for success	KPIs	Evaluation	
9.1 Certification cost	38	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
9.2 Re-certification cost	39	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
9.3 Criticality level up	40	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟧
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 18: Template for evaluation of objective 9

Objective 10: Increase capabilities and programming flexibility of the safety protection system			
Measure for success	KPIs	Evaluation	
10.1 Safe data availability	41,42	The solution proposed in DREAMS allows overcoming limitations usually introduced by commercial hardware regarding number of variables to handle and programming flexibility.	🟩
Objective evaluation			
The preliminary evaluation of this objective is very positive.			

Table 19: Template for evaluation of objective 10

Objective 11: Incorporate mixed-criticality networks and means for validation			
Measure for success	KPIs	Evaluation	
11.1 Mixed-criticality networks	43	Flexibility of the mixed-criticality network is very good at off-chip level, but there are some limitations at on-chip level.	🟩
11.2 Networks validation means	44	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is very little information at this point.			

Table 20: Template for evaluation of objective 11

Objective 12: Obtain a complete methodology to manage system complexity and variability			
Measure for success	KPIs	Evaluation	
12.1 Methodology compatibility	45	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
12.2 Variability management	46,47	This measure for success cannot be evaluated yet. Some of the KPIs it depends on cannot be assessed in the preliminary evaluation.	🟨
Objective evaluation			
Preliminary evaluation of this objective is not possible since there is no information at this point.			

Table 21: Template for evaluation of objective 12

4 Conclusions

This document presents the preliminary evaluation of the concepts, technologies and tools developed in DREAMS by means of the wind power demonstrator. Key Performance Indicators have been calculated or estimated in order to provide quantitative, objective and measurable information to later evaluate measures for success and fulfillment of project objectives.

The final evaluation will try to minimize the number of estimations and maximize the number of objectively calculated indicators to increase the credit of the document. However, in this preliminary report some of the indicators needed to be estimated.

Regarding monitoring of the demonstrator presented in section 2, the status is slightly delayed with respect to the foreseen schedule. The main reason are the delays in the delivery of some of the technologies, especially the harmonized platform adapted to the demonstrator needs, which has been provided in late April 2016. The progress in the Galileo side of the demonstrator is very promising, since all technologies have been successfully integrated and the platform is ready to host application specific processes (porting is ongoing). As already mentioned, the adaptation of the harmonized platform has been concluded recently, and most of the demonstrator requirements have been implemented successfully. However, it has not been tested yet because of some issues with the integration of the PCIe gateway, which is not always detected by Galileo platform. This is considered a normal situation taking into account the complexity of the development. Solving this problem is a priority for demonstrator planning, in order to enable integration and testing of the two main parts of the demonstrator: Galileo and the harmonized platform.

Section 3.2 presents a preliminary assessment of the processor timing isolation in the ARM cores of the harmonized platform. The conclusion is that absolute temporal isolation is not possible if shared resources are being used, but the temporal interference can be estimated and bounded. If a safety critical partition is to be placed in one of the ARM cores (which is the case in the wind power demonstrator), there are two possibilities in order to guarantee temporal properties. The first one is to avoid parallel execution of partitions on different cores when there is a safety application with hard real-time requirements. This is the most conservative approach, but the potential of the multicore architecture is not fully exploited. The second possibility is to bound temporal interference generated in the safety partition, allow some margin in the temporal behaviour so that this interference does not lead to a failure, and enable mechanisms to detect occasional temporal violations and drive the system to the safe state. This approach allows taking advantage of the multicore potential but may impact the availability of the system if temporal interferences are not correctly bounded. For the purpose of the demonstrator, the second approach is to be used.

In section 3 the preliminary evaluation is presented. A reduced number of KPIs has been considered, since the demonstrator development is still in an early phase. However, most of the KPIs that have been able to be evaluated at this point, have been positively assessed and match the expected values. Some remarkable conclusions are:

- The safety objectives of the projects can be achieved theoretically. However, there is still a lot of work (far beyond this project) in order to transform theory into a real certification.
- Timing objectives are also met.
- The project provides most of the building blocks required by the wind power demonstrator “out of the box”.

- The tools and models provided by the project simplify development and reduce complexity in many ways, while improving flexibility.

The wind power demonstrator will contribute in the evaluation of a high percentage of project measures for success and objectives. The alignment of the demonstrator with the project vision is very high, and this will allow an extensive use of the demonstrator for dissemination activities. However, the evaluation of the measures for success and objectives at his point is very superficial, and needs more evidences which will be hopefully collected in the final evaluation, when the demonstrator is up and running.

5 Bibliography

1. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Description of Work*, in *DOW*2014. p. 260.
2. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Wind Power Evaluation and Monitoring Plan*, in *D7.1.2*2015. p. 56.
3. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Wind Power Use Cases Specifications*, in *D7.1.1*2014. p. 51.
4. B&R. *Automation PC 910*. 2015 [cited 2015 June]; Available from: http://www.br-automation.com/en/products/industrial-pcs/automation-pc-910/.
5. XILINX, *ZYNQ-7000 All Programmable SoC: Technical Reference Manual*, in *UG585*, 2014
6. EtherCAT, *Safety over EtherCAT*, 2011
7. IEC, *IEC 61508-2: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems*, in *Requirements for electrical/ electronical / programmable electronic safety-related systems*2010, IEC.
8. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Architecture Conceptualization: Requirements, Terms and Principles*, in *D1.1.1*2014. p. 295.
9. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Architecture Style of DREAMS*, in *D1.2.1*2014.
10. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: XtratuM support for enhanced hypervisor layer services: description and interfaces*, in *D2.3.1*2015. p. 92.
11. Fent Innovative Software Solutions, *XtratuM Hypervisor for INTEL x86*, in *fnts-xm-um-x86-21c*, 2013
12. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: System Platform Integration*, in *D2.4.1*2016.
13. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: High-Level Design of Cluster-Level Safety and Security Services*, in *D3.3.1*2014. p. 49.
14. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: First Implementation of Cluster-Level Safety and Security Services*, in *D3.3.2*2015. p. 80.
15. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: Integration and Support Report*, in *D3.4.1*2017.
16. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Initial Collection of Offline Adaptation Strategies for Mixed-Criticality*, in *D4.1.1*2014. p. 107.
17. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: Definition of Offline Adaptation Strategies for Mixed-Criticality and Initial Implementation*, in *D4.1.2*2015. p. 102.
18. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: Variability Analysis and Testing for Mixed-Criticality Systems*, in *D4.3.1*2015. p. 54.
19. DREAMS, *Distributed Real-time Architecture for Mixed Criticality Systems: First implementation and improvement of variability analysis and testing techniques for mixed critical systems*, in *D4.3.2*2015. p. 27.
20. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Modular Safety Case for Hypervisor*, in *D5.1.1*2015.

21. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Modular Safety Case for COTS processor*, in *D5.1.2*2015.
22. DREAMS, *Distributed Real-Time Architecture for Mixed-Critiality Systems: Modular Safety Case for Mixed-Criticality Network*, in *D5.1.3*2015.
23. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Cross domain mixed-criticality patterns*, in *D5.3.1*2016.
24. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Fault injection framework*, in *D5.2.3*2015.
25. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Validation Techniques for Product-lines of Mixed Criticality Systems*, in *D5.5.2*2015.
26. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Method for certifying mixed-criticality product lines*, in *D5.5.3*2016.
27. IKERLAN, *Certificate: Functional Safety Management Structure - IEC-61508*, 2013.
28. ISO, *ISO 13849-1: Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design*, 2006. p. 94.
29. ISO, *ISO 13849-2: Safety of machinery — Safety-related parts of control systems — Part2: Validation*, 2012. p. 86.
30. IEC, *IEC 61508-1: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General Requirements*, in *Requirements for electrical/ electronical / programmable electronic safety-related systems*2010, IEC.
31. IEC, *IEC 61508-3: Functional safety of electrical/electronical/programmable electronic safety-related systems - Part 3: Software requirements*, in *Requirements for electrical/ electronical / programmable electronic safety-related systems*2010, IEC.
32. DREAMS, *Distributed Real-Time Architecture for Mixed-Criticality Systems: Meta-models for Application and Platform*, in *D1.4.1*2015. p. 124.

Terminology

Access control

Access control includes authorization, identification and authentication (I&A), access approval, and audit. Authorization specifies what a subject can do, e.g., read, write or execute a file. Access approval grants or rejects access to the requested resource. Audit records the access to a resource. For Identification and authentication please refer to the topic on authentication.

Assurance Level

The assurance level is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system.

Authenticity

Authenticity ensures that data is genuine and that the actual origin of the data is the same as the claimed origin.

Authentication of data origin

Authentication of data origin ensures that the actual origin of the data is the same as the claimed origin.

Authentication of a communication partner

Authentication of a communication partner ensures that the actual communication partner is the same as claimed.

Availability

If an Information or access to a service is needed, it must be available. Additionally, it must also function correctly.

Behaviour

The behaviour of a subsystem is the sequence of message (i.e., intended and unintended) that is produced by the subsystem at its LIF.

Channel

A channel serves for the exchange of messages between ports. A channel is associated with a communication topology, a data-direction (e.g., unidirectional or bidirectional), temporal properties and dependability properties.

Cluster

A cluster is a physically distributed computer system that consists of a set of nodes interconnected by a physical network. Each node can be a multi-core chip with multiple IP cores interconnected by a network-on-a-chip. A cluster can be connected to another cluster using a gateway.

Compliant Item

A compliant item is any item (e.g. an element) on which a claim is being made with respect the clauses of IEC 61508 series.

Component

A component is a constituting element of an application subsystem and forms the basic unit of work. It interacts with other components through the exchange of messages across LIFs in order to work towards a common goal and provide the application services.
A component is regarded as a self-contained building block that can be used in the design of a larger system. The component can have a complex internal structure that is neither visible, nor of concern, to the user of the component. In the context of embedded real-time systems, it is essential that the component behaviour can be specified in the domains of value and time.

Composability

Composability is a concept that relates to the ease of building systems out of subsystems. A system, i.e., a composition of subsystems, is considered composable with respect to a certain property (functional or non-functional) if this property, given that it has been established at the subsystem level, is not invalidated by the integration. Examples of such properties are timeliness or certification.

For example, some embedded systems closely interact with their environment and they have to produce intended results at intended points of time. Temporal composability is a prerequisite for the feasible construction of such temporally predictable systems of high complexity. In architectural styles that support temporal composability, determining the emergent temporal behaviour of the resulting system is eased by the fact that the individual subsystems retain their temporal properties after integration.

Confidentiality

Confidentiality ensures the privacy of information. Only authorized users can read the data. This includes the data stored in memory as well as the data transferred over a network.

Core Platform Service

Core platform services (or core services for short) are mandatory in every instantiation of the architecture style. The core platform services provide the foundation for higher-level, optional platform services. For instance, a message-based communication service is a core service. At any given integration level, the core services form a waist that can be realized using a multitude of implementation choices. In addition, they form the starting point for the domain-customization using optional services. Exemplary categories of core services are communication services, execution services, time services and resource management services.

Criticality System

Mixed-criticality is the concept of allowing application subsystems that must meet different assurance levels (e.g., ranging from DAL A to DAL E in RTCA DO-178B, SIL1 to SIL4 in EN ISO/IEC 61508) to seamlessly interact and co-exist on the same networked distributed computational platform.

Design Pattern

A Design Pattern is a general reusable solution to a commonly occurring problem within a given context. It is a description or template for how to solve a problem that can be used in many different situations. Patterns are formalized best practices.

Dependability Patterns

Design patterns that focus on finding common links on dependability as a measure of a system's availability, reliability, and its maintainability.

Determinism

A model behaves deterministically if and only if, given a full set of initial conditions (the initial state) at time t0, and a sequence of future timed inputs, the outputs at any future instant t are entailed.

Development Methodology

The development methodology is a framework consisting of a development process, a set of methods, techniques and tools for mixed-criticality systems based on networked multi-core chips.

End-to-End Channel

An end-to-end channel is a channel that can include on-chip and off-chip communication links over hierarchical, heterogeneous and mixed-criticality networks. Gateways enable the horizontal integration at the cluster-level across different off-chip communication networks with different protocols (e.g., TTEthernet, EtherCAT, etc.), different reliabilities (e.g., fault-tolerant networks with media redundancy and active star couplers, low-cost field bus networks). Gateways between NoCs and off-chip networks enable the vertical integration through the seamless communication in hierarchical networks respecting mixed-criticality safety and security requirements.

Error

An error is that part of the system state, which is liable to lead to a subsequent failure. A failure occurs when the error reaches the service interface.

Event

“An event denotes a distinct form of state change in a running system, taking place at distinct points in time called occurrences of the event. That is, a running system can be observed by identifying certain forms of state changes to watch for, and for each such observation point, noting the times when changes occur. This notion of observation also applies to a hypothetical predicted run of a system or a system model — from a timing perspective, the only information that needs to be in the output of such a prediction is a sequence of times for each observation point, indicating the times that each event is predicted to occur.” – TIMMO-2-USE

Fail-operational System

A fail-operational system is able to tolerate one or several faults. Fail-operational systems send correct messages despite the failure of their subsystems.

Fail-safe System

If a fail-safe system one or more safe states can be reached in case of a system failure. Fail-safeness is a characteristic of the controlled object, not the computer system. In fail-safe systems the computer system must have high error-detection coverage.

Fault

A fault is the adjudged or hypothesized cause of an error. Faults can be internal or external of a system.

Examples of types: An external fault (e.g. a malicious attack) causes an error, and possible a subsequent failure. An internal fault (i.e. vulnerability) allows an external fault to harm the system and has to pre-exist in the system.

Fault-Containment Region

A Fault Containment Region (FCR) is a subsystem that operates correctly regardless of any arbitrary logical or electrical fault outside the region.

Fault Hypothesis

The fault hypothesis is the specification of the faults that must be tolerated without any impact on the essential system services. The fault hypothesis states the assumptions about units of failure (see Fault Containment Region), failure modes, failure frequencies, failure detection, and state recovery.

Failure

A failure occurs when the delivered service deviates from fulfilling its specification.

Integration Level

The integration level denotes the layer in a system-of-systems at which it is composed out of its components. Different integration levels can be distinguished in embedded systems including the chip-level, the cluster-level and the core-level.

Integration Level: Chip-Level

The chip-level is an integration level where IP cores are integrated using an on-chip network.

Integration Level: Cluster-Level

The cluster-level is an integration level where multiple chips are interconnected to a cluster using one or more off-chip communication networks (e.g., ´TTEthernet, EtherCAT). Thereby, applications can be supported that need more resources than are available on a single SoC. In addition, a distributed system with multiple SoCs is a prerequisite for implementing safety-critical application subsystems, because today’s semiconductor technology does not support the manufacturing of chips with a reliability that is suitable for ultra-dependability.

Integration Level: Core-Level

The core-level is an integration level where components are integrated using a hypervisor.

Integrity

Data integrity means that the data cannot be modified unnoticeably. Every intended and unintended modification of the data should be detectable.

Mixed-Criticality Architecture

A mixed-criticality architecture is an architecture that provides platform services and a development methodology supporting mixed-criticality (e.g., temporal and spatial partitioning, modular certification methods).

Optional Platform Services

The optional platform services, which are built upon the core platform services, can be generic in the sense that they can be used in multiple application domains or specific for a focused domain. These services are optional in the sense that they are not required in every instantiation of the architecture. If needed, developers can pick them out of the architectural style, which includes a set of existing, validated component libraries for the different integration levels. For instance an encryption service could be a generic optional service.

Partition

A partition is the execution environment for a component with corresponding resources (e.g., processor, memory, communication, input/output). The resources for a partition are protected by temporal partitioning and spatial partitioning in order to avoid unintended feature interaction and fault propagation between components.

Periodic Message

Periodic messages are specified by a period and phase, which can be expressed with respect to a system-wide synchronized global time base.

Periodic messages can be exchanged using *time-triggered communication*, where the instants of periodic message transmissions are specified by an a priori planned conflict-free communication schedule. For time-triggered communication, the communication infrastructure is deterministic and guarantees temporal properties such as latency, latency jitter, bandwidth, and message order.

Platform

A platform is the hardware/software foundation for the execution of applications. The platform instantiates the architectural style and implements generic services for the development of applications, which are denoted as platform services (see core platform services and optional platform services).

Platform Services

Platform services facilitate the development of applications subsystems and separate the application functionality from the underlying platform technology to reduce design complexity and to enable design reuse. We differentiate between two different types of platform services: core platform services and optional platform services.

Platform-Independent Model

A Platform Independent Model (PIM) is a model of a system that is independent of the specific technological platform used to implement it.

Platform-Specific Model

A Platform Specific Model (PSM) is a model of a system that is linked to a specific technological platform used in implementation.

Reliability

Reliability is the ability of an application subsystem to perform its required functions under stated conditions for a specified period of time.

Safety manual for compliant items

Safety manual for compliant items is a document that provides all the information relating to the functional safety of an element, in respect of specified element safety functions, that is required to ensure that the system meets the requirements of IEC 61508 series.

Secure End-to-End Channel

Using a secure end-to-end channel means that the communication is uninterruptedly protected between two communicating parties, e.g., PGP (e-mail), ZRTP (VoIP), etc.

Secure Point-to-Point Channel

Using a secure point-to-point Channel means that the communication is uninterruptedly protected between two points/nodes in a network, e.g., VPN, MACsec, IPsec etc.

Security Mechanisms

Security mechanisms are used to provide security services, e.g., encryption is used to ensure confidentiality.

Security Services

Security services define different classes to protect a system against attacks. Security services include authentication, access control, confidentiality, integrity and non-repudiation.

Spatial Partitioning

Spatial partitioning ensures that the service in one partition cannot alter the code or private data of another partition. Spatial partitioning shall also prevent a partition from interfering with control of external devices (e.g., actuators) of other partitions.

Sporadic Message

Sporadic messages establish rate-constrained data-flows with maximum bandwidth use, which helps to guarantee bounded latencies. Successive transfers of sporadic messages belonging to the same rate-constrained dataflow are guaranteed to be offset by a minimum duration (also called minimum inter-arrival time of sporadic messages).

The temporal behaviour of sporadic messages can further be specified by sporadic repetition constraints.

State

The state enables the determination of a future output solely on the basis of the future input and the state the system is in. In other word, the state enables a "decoupling" of the past from the present and future. The state embodies all past history of the given system. Apparently, for this role to be meaningful, the notion of the past and future must be relevant for the system considered.

State Recovery

State recovery is the action of re-establishing a valid state in a subsystem after a failure of that subsystem.

Subsystem

A subsystem is a part of a system that represents a closure with respect to a given property.

System

A system is a set of subsystems.

Temporal Partitioning

Temporal partitioning ensures that a partition cannot affect the ability of other partitions access shared resources, such as the network or a shared CPU. This includes the temporal behaviour of the services provided by resources (latency, jitter, duration of availability during a scheduled access).

Timing Event

Timing Events are identifiable state changes that are possible to constrain with respect to timing. Examples of timing events are: Message Sent, Message Arrived, Task Activation, Task Execution End, Frame Instantiation, Frame Transmission Start, Frame Transmission End.

The most common timing constraints are Latency constraint, Repetition Constraint, Synchronization Constraint.

Task Activation (Event)

A Task Activation is a Timing Event that describes the fact that a recurring task has entered the scheduling queue, i.e. will be considered by the scheduler for allocation of the processing unit.

Task Activations may occur for example periodically, with a certain jitter (see also Repetition Constraint).

Task Execution End (Event)

A Task Execution End is a Timing Event that describes the fact that a recurring task has executed all its instructions and is therefore removed from the scheduling queue.

Synchronization Constraint

A Synchronization constraint describes how tightly the occurrences of a group of events follow each other. This is typically expressed by a temporal window, i.e. an upper bound on the temporal distance between the occurrences of the events of the group.

An example is the reading of input data from different sensors, which must occur in a small time window to ensure a temporally consistent view of the environment.

Worst Case Execution Time (WCET)

The Worst Case Execution Time is the maximal delay needed to execute all instructions of a task, excluding interruption or pre-emption delays.

Worst Case Response Time (WCRT)

The Worst Case Response Time is the worst delay between the occurrence time of the Task Activation and the occurrence time of the Task Execution End. With respect to the WCET, it includes interruption/pre-emption or initial blocking delays (non-pre-emptive scheduling).

Worst Case Traversal Time (WCTT)

The Worst Case Traversal Time is the worst delay between the occurrence time of the Frame Instantiation and the occurrence time of the Frame Transmission End.