Distributed Real-time Architecture for Mixed Criticality Systems

Report on final assessment and support for demonstrators D 1.8.2

Project Acronym	DREAMS	**Grant Agreement Number**		FP7-ICT-2013.3.4-610640	
Document Version	1.0	**Date**	2017-09-30	**Deliverable No.**	D 1.8.2
Contact Person	Hamidreza Ahmadian	**Organisation**		USIEGEN	
Phone	+49 271 740 2547	**E-Mail**		hamidreza.ahmadian@uni-siegen.de	

Contributors

Name	Partner
Hamidreza Ahmadian	USIEGEN
Roman Obermaisser	USIEGEN
Thomas Koller	USIEGEN
Simon Barner	FORTISS
Mohammed Abuteir	TTT
Andreas Eckel	TTT
Jörn Migge	RTAW
Tiziana Mastroti	RTAW
Ibai Sarasola	IKL
Imanol Martínez	IKL
María Cristina Zubia	IKL
Daniel Gracia Pérez	TRT
Gautam Gala	TUKL
Ton Trapman	ALSTOM
Kevin Chappuis	VOSYS
Jeremy Fanguède	VOSYS
Marcello Coppola	ST
Michael Soulie	ST
Alfons Crespo	UPV
Javier Coronel	FENTISS
Gebhard Bouwer	TUV
Miltos Grammatikakis	TEI
Youcef Bouchebaba	ONERA
Durrieu Guy	ONERA
Franck Chauvel	SINTEF

Table of Contents

Executive Summary

This deliverable summarizes the final assessment of the DREAMS architecture in the avionic, wind power and healthcare domains. The technological building blocks with updated plans include the DREAMS meta-models, the platform configuration file generators, the on-chip network communication, the composition of schedules, the safety communication layer, the modular safety cases, the resource management services, the virtual platform, the security services, and the gateways. Moreover, it is discussed how the defined project objectives are fulfilled by the technologies as well as the demonstrators. In addition, it is analyzed which technological building block can be shut down during the demonstration to highlight the contribution of the DREAMS solutions. Finally, the demonstrator support activities of the technology work packages are summarized. The goal of the demonstrator support is to promote and support the usage of WP1-5 results by the demonstrators in WP6, WP7 and WP8.

1 Introduction

1.1 Context of the Deliverable

This deliverable is the second outcome of the task T1.8 (support for demonstrators and consolidation of assessment) in work package WP1, whose goal is to support and promote the usage of the work-package results (e.g., application and platform models) by the demonstrators.

The support includes also help on the identification of opportunities to apply the results. In addition, T1.8 consolidates the assessment from the demonstrators in WP6, WP7 and WP8. The final assessment examines the result of the final integration (T1.7) and benefits from the valuable feedbacks from the intermediate assessment in the three application domains; and it is also used to evaluate the technological building blocks in WP1-WP5.

The purpose of the final assessment is to quantify significant properties based on the identified metrics (e.g., reliability, timing, fault containment, performance, energy) facilitating the exploitation of the DREAMS architecture.

1.2 Methodology

The main purpose of this deliverable is to provide a summary of the final assessment from the avionics demonstrators of WP6, the wind power demonstrator of WP7, and the healthcare demonstrator of WP8.

Furthermore, a feedback loop is created in order to provide recommendations on how to improve the technological results of the work packages 1-5.

This document consists of the following three major parts:

- Fulfillment of the project objectives based on the technologies and the KPIs
- Summary of results from the demonstrators assessments (avionics, wind power, healthcare)
- Possibility and expected effect of shutting down the DREAMS services during the demonstration
- Demonstrator support by WPs 1-5

2 Fulfillment of the Project Objectives, considering Measures for Success

The DREAMS project aims to provide a flexible platform and associated design tools for embedded applications where subsystems of different criticality, executing on networked multi-core chips, can be integrated seamlessly. The vast range of the DREAMS technological building blocks address a number of project objectives. This section[1] elaborates on how the project objectives are fulfilled by the DREAMS technologies.

2.1 Consolidation and extension of architectural concepts from previous projects (e.g., RECOMP, GENESYS, ACROSS, ARAMIS) towards a new architecture style for the seamless virtualization of networked embedded platforms ranging from multi-core chips to the cluster level with support for security, safety and real-time performance as well as data, energy and system integrity

Architectures from previous projects served as a starting point for consolidation, integration and extension in DREAMS. For example, the GENESYS architectural style and corresponding time-triggered architectures from ACROSS and INDEXYS provided a starting point for the communication and time services. Hypervisors and certification concepts from MULTIPARTES were analyzed to define the DREAMS execution services. The results of the input projects FRESCOR and ACTORS were considered for the definition of the resource management services. Although these inputs served as a starting point, substantial contributions and extensions beyond these prior results were provided in order to establish the DREAMS architectural style for networked multi-core chips. In particular, the DREAMS architectural style supports hierarchical end-to-end virtualization and resource management in networked multi-core chips using interpartition communication of the hypervisors, on-chip networks, off-chip networks and gateways. Time/space partitioning is addressed at all considered integration levels, while supporting heterogeneous application models (e.g., time-triggered and event-triggered, message passing and shared memory). The development of mixed-criticality applications for this architectural style is supported by the model-based DREAMS development methodology and modular safety concepts.

2.2 Waistline architecture with domain-independent platform services that can be successively refined and extended to construct more specialized platform services and application services. The platform services shall provide a stable foundation for the development of applications and enable the safe and secure composition of mixed-criticality systems out of components

The DREAMS architectural style defines domain-independent platform services as a stable baseline for mixed-criticality applications. The domain-independent platform services include time services, communication services, execution services and resource management services. The realization of these services is supported on a variety of different implementation technologies, thereby ensuring high exploitation potential and avoiding technology obsolescence. For example, as part of the demonstrators the platform services were realized using Xilinx ZYNQ-7000, Juno ARMv8, Intel x86 and PPC T4240. Likewise, the domain-independent platform services of DREAMS can serve as a basis for different application services. In the project, we have demonstrated the refinement of the platform

[1] This section was added based on a review recommendation.

services towards healthcare, avionics and windpower applications. The result is a bigger market for platform providers to leverage the economies of scale.

2.3 Models of hierarchical platforms comprised of networked multi-core chips to enable MDE

The development process defined in DREAMS covers the phases defined in V-shape top-down development approaches that are required by safety standards. The process defined in the course of the project focuses on design, implementation, integration and verification activities and is entirely based on models. This enables to implement seamless tool support in form of an integrated tool chain, and to ensure the consistency of the different artifacts created in the course of the process.

The pivot of the DREAMS approach is the architectural style [1] that provides the blueprint of an architecture and the set of services that is required to enable the safe integration of mixed-criticality applications on virtualized networked multicore chips.

In order to support this architecture in a model-driven engineering process, the project developed an adequate modeling approach to describe mixed-criticality applications and instances of the DREAMS platform. In the following, we assess the fulfillment of this objective based on the criteria defined in [2]:

1. *The models should contain sufficient details to support fine-grained analysis/scheduling of mixed-criticality systems in WP4 as well as the development of use-cases in WP6-WP8.*

   The DREAMS model-based development process [3] and the resulting tool-chain [4] uses the DREAMS meta-model [5] [6] as a backbone. This metamodel has been developed to support the modeling of MCS based on the principles defined in the DREAMS architectural style. As pointed out in the WP4 reports, [4] in particular, this tool-chain includes *scheduling tools* for the resources at the different levels of the DREAMS hierarchy. In the following, we list these scheduling tools, and also include further *automated design tools*:

   - AutoFOCUS3/Design Space Exploration
   - RTaW-Timing / Timing Decomposition
   - RTaW-Timing / On-chip TT Sched
   - TTEPlan
   - Grec
   - MCOSF
   - Xoncrete
   - BVR

   Further, [4] also describes also the following *analysis tools*:

   - RTAW-Timing / Evaluation
   - Mixed Criticality Product Line Editor (Safety Checker)

   [4] points out, how each of the tools is connected to the DREAMS meta-model, and how the integration of the overall tool-chain is based on this common data format (c.f. sub-sections *"Data Exchange"* in the tool-specific sections of [4]).

   The application demonstrators ([D6.3.2] [7], Section 3.6; [D7.3.2] [8], Table 11; [D8.3.2] [9], Table 6) positively assessed the applicability of the model-based engineering process and tool-chain, and hence also the DREAMS modeling approach. For the Avionics demonstrator in WP6, [10] provides initial models that have been revised and refined for the final version of the demonstrator (see [18]). For the windpower demonstrator in WP7, [10] also contains an initial model, that has already been refined in [11], [12], [13] during the experimental evaluation of the variability exploration approach and that have been published in [17]. A model of the WP8 healthcare demonstrator is presented in [16].

2. *The complexity of the models established in the use-cases will be assessed against experienced data acquired from domain experts from the demonstrator work-packages.*
   The applicability of the approach has been positively assessed in ([D6.3.2] [7], [D7.3.2] [8], [D8.3.2] [9]), which conclude that the use of model-driven engineering (MDE) has contributed to a reduction of development time.

3. *The completeness of the implementation will be assessed against the requirements collected in the initial phase of the project.*
   D1.1.1, Section 9 defines the requirements for the DREAMS modeling approach. In Table 1 below, we analyze for each of these requirements if it has been satisfied, and trace it to the corresponding deliverable that describes the solution developed in the context of the project, as well as the means applied to validate the its satisfaction. In addition to the references listed in the "traces" column, the videos available in the mixed-criticality forum[2] and the video on the DREAMS tool-chain demonstrate the usage of the DREAMS metamodel.

ID	Name	Status	Traces	Comments
R9.1.1	Separation of concerns	Satisfied	[D1.4.1], Sec. 2 [D1.6.1], Sec. 2	Introduction of dedicated viewpoints providing metamodels for different concerns Resulting architecture of DREAMS metamodel
R9.1.2	Adequate degree of abstraction (Measure of success)	Satisfied	[D6.3.1], Sec. 3.5 [D7.3.1], Sec. 2.4, [D8.3.1], Sec. 4.1 [D1.8.1], Sec. 2 [D4.4.1]	Evaluation by demonstrator use cases in progress (applicability of approach has already been positively assessed). Analysis of initial and final assessment by demonstrators Successful integration of DREAMS toolchain
R9.1.3	Domain-independence (measure of success).	Satisfied	[D1.8.1], Sec. 2	Analysis of initial and final assessment by demonstrators
R9.2.1	Platform-independent Application Meta-Model	Satisfied	[D1.4.1], Sec. 4.5 [D1.5.1], Sec. 4.1.2, [D1.7.1], Sec. 2.9.3 [D4.4.1] [D1.8.1], Sec. 2	Module test: synthetic example models Integration test: 1) Preliminary demonstrator models, 2) Use of modeling and tooling approach in demonstrators, 3) Successful integration of DREAMS toolchain

[2] http://www.mixedcriticalityforum.org/about/model-driven-engineering/ (videos at the bottom of the page)

				Acceptance test: Analysis of initial and final assessment by demonstrators
R9.2.2	Precise execution semantics	Partially satisfied	[D1.4.1], Sec. 4	Module test: Logical viewpoint defines component architecture, state and mode automata and provides synthetic examples. Integration / acceptance test: For the demonstrators, only architectural models whose timing behavior is described by means of the temporal viewpoint is used.
R9.2.3	Support for modeling memory requirements	Satisfied	[D1.4.1], Secs. 4.3.1, 5.2.5, 5.2.6, 5.3 [D1.4.1], Sec. 4.5, 5.2.5.2, 5.2.6.2 [D1.5.1], Sec. 4.1.2, [D1.7.1], Sec. 2.9.3 [D4.4.1] [D1.8.1], Sec. 2	Models for memory requirements of logical architecture, and memory configuration of system software and physical architecture Module test: synthetic example models Integration test: 1) Preliminary demonstrator models, 2) Use of modeling and tooling approach in demonstrators, 3) Successful integration of DREAMS toolchain Acceptance test: Analysis of initial and final assessment by demonstrators
R9.3.1	Platform architecture	Satisfied	[D1.4.1], Section 5 [D1.4.1], Sec. 5.2.X.2 [D1.5.1], Sec. 4.1.2, [D1.7.1], Sec. 2.9.3 [D4.4.1]	Technical architecture modeling approach Module test: synthetic examples for the different levels of the DREAMS architecture Integration test: 1) Preliminary demonstrator models, 2) Use of modeling and tooling approach in demonstrators, 3) Successful

			[D1.8.1], Sec. 2	integration of DREAMS toolchain Acceptance test: Analysis of initial and final assessment by demonstrators
R9.3.2	Platform services	Satisfied	See R9.3.1	See R9.3.1
R9.4.1	Representation of Deployed Applications	Satisfied (acceptance test pending)	[D1.4.1], Sec. 6, [D1.6.1], Sec. 3.1, 3.2, 3.3, 4. [D4.4.1]	Deployment viewpoint: mapping model, platform-specific parameters Platform-specific model: Resource utilization MM (virtual link MM, reconfiguration MM) and Service configuration Viewpoint Module test: [D1.4.1] and [D1.6.1] provide synthetic examples Integration test: Successful integration of DREAMS toolchain Acceptance test: final implementation of demonstrators pending
R9.5.1	Latency constraints	Satisfied	[D1.4.1], Sec. 7.1 [D1.4.1], Sec. 7.3 [D1.5.1], Sec. 4.1.2, [D1.7.1], Sec. 2.9.3 [D4.4.1] [D1.8.1], Sec. 2	Reaction constraint Module test: Synthetic timing model example Integration test: 1) Preliminary demonstrator models, 2) Use of modeling and tooling approach in demonstrators, 3) Successful integration of DREAMS toolchain Acceptance test: Analysis of initial and final assessment by demonstrators
R9.5.2	Repetition constraints	Satisfied	[D1.4.1], Sec. 7.1, Validation: See R9.5.1	Periodic constraints (and sporadic constraint) Validation: See R9.5.1
R9.5.3	Synchronization constraints	Satisfied	[D1.4.1], Sec. 7.1, Validation: See R9.5.1	Delay constraint

R9.5.4	Coverage of tools and demonstrators	Satisfied	[D6.3.1], Sec. 3.5 [D7.3.1], Sec. 2.4, [D8.3.1], Sec. 4.1 [D1.8.1], Sec. 2 [D4.4.1]	Assessment of modeling approach by demonstrator applications Successful integration of DREAMS toolchain
R9.6.1	Policies according to IEC-61508	Satisfied	[D1.4.1], Sec. 8.1 [D1.4.1], Sec. 3.4 [D1.5.1], Sect. 4.1.2.2 [D1.8.1], Sec. 2	IEC 61508 and Diagnostic Techniques and Measures metamodel and Safety Compliance metamodel. Module test: Example models Integration test: Example models Acceptance test: Analysis of initial and final assessment by demonstrators
R9.6.2	Criticality levels	Satisfied	See R9.6.1	Safety integrity level, see R9.6.1
R9.6.3	Traceability	Satisfied	[D1.4.1], Sec. 6, 7, 8.1, 9 [D1.6.1], Sec. 4.1	Deployment provides links between logical and technical architecture, external models (i.e., safety, timing and variability model) point to logical architecture Reference to system model element in base class representing configuration
R9.7.1	System-level NoC static/dynamic power consumption analysis model	Satisfied	[D1.4.1], Sec. 8.3	System level analytical interconnect IP power model
R9.7.2	System-level energy / Power requirements meta-model	Satisfied	[D4.1.2], Sec. 3.1.2.5 [D4.3.2], Sec. 4.4, [D4.3.3], Sec. 6.2	DSE specification model (energy minimization objective) Evaluation of DSE on wind power demonstrator
R9.8.1	Security Meta-Model for Data Confidentiality	Satisfied	[D1.4.1], Sec. 8.2.2 [D1.4.2], Sec. 8.2.4	Confidentiality annotation Example models

R9.8.2	Security Meta-Model for Data Integrity	Satisfied	See R9.8.1	Integrity annotation, see R9.8.1
R9.8.3	Security Meta-Model for Authentication	Satisfied	See R9.8.1	Authenticity annotation, see R9.8.1
R9.9.1	Separate variability description	Satisfied	[D1.4.1], Sec. 9.1 [D1.4.1], Sec. 9.3 [D4.1.2], Sec. 3.2.2.3 [D4.3.2], Sec. 4.3, [D4.3.3], Sec. 6.1	BVR meta-model that is agnostic of the underlying product space metamodel Example models Example based on DREAMS metamodel to illustrate product exploration and fragment substitution Evaluation of variability approach on wind power demonstrator
R9.9.2	Layered and modularized variability description	Satisfied	See R9.9.1	Model elements to structure variability specifications (packages and compound objects), see R9.9.1
R9.9.3	Flexible variability resolution	Satisfied	[D4.3.1, D4.3.2, D4.3.3]	Specification as well as initial and final implementation of variability resolution process, including experimental validation based on the wind power use case
R9.9.4	Flexible variability implementation platform	Partially satisfied	[D1.4.1], Sec. 9.1 [D4.3.1, D4.3.2, D4.3.3]	BVR meta-model that is agnostic of the underlying product space metamodel (i.e., also the DREAMS platform metamodel). Specification as well as initial and final implementation of variability resolution process. While that process is able to address the exploration of platform architectures (as demanded in this requirement), in the course of the wind power use case experiments the exploration of safety architecture variants has been investigated.

Table 1 Assessment of fulfillment of requirements for DREAMS modeling approach.

2.4 Certifiable *platform services for virtualization and segregation of resources* at cluster and chip-level (e.g., I/O virtualization, message-based networks and memory architectures, dynamic resource management)

In the context of DREAMS, several ST, TEI and Virtual Open Systems have developed several technologies that enables to support certifiable services and segregation of resources. These technologies are implemented either in as Software or as Hardware.

The technologies developed are:

- Firmware Mmonitor layer
- MemGuard
- STNoC Memory interleaving

MemGuard performs dynamic memory bandwidth management at CPU-level by using hardware performance counters to monitor periodically the number of last-level cache misses (or equivalently accesses to the shared bus). In DREAMS, an extension to MemGuard algorithm (called MemGuardXt) has been developed. This extension provides a way to provide hard guarantee on the traffic rate which is especially important for real-time applications and to improve its adaptivity for predicting bandwidth. In addition, to improve modularity, MemGuardXt algorithm has been used directly in either user- or kernel-space, in one or more instances. Using this methodology, it is possible two kernel modules: a kernel module running the MemGuardXt algorithm and a new network regulation module (called NetGuardXt) running over netfilter which uses a similar algorithm to MemGuardXt.

Memory Interleaving is a way to segregate DDR memory resources to enable mixed time critical application to meet their requirements. Multiprocessor chip have several memory controllers where DDR memory is plugged. Each memory controller provides a channel of information flow from/to the DDR memory. Before DREAMS, these channels were managed by the memory controllers and the network on chip without considering the specific real time requirement (Hard, Soft or Best Effort).

A channel is accessed by specific transaction address, through a memory map. Using the STNOC with interleaving extension is possible to give to the software how to use/ to balance the load among memory channels. This feature enables applications with HARD hard realtime (such as Software define radio) or Soft soft realtime (such as Media decoder) properties to meet their requirements. In particular, with this extension the predictability through precomputed memory patterns can be analyzed, and then performed by the applications.

The firmware monitor layer [D2.3.2], which enables the native concurrent execution of a safety critical Real-Time Operating System (RTOS) along with a rich Operating System (OS) with the option to use virtualization extensions, such as Linux-KVM, in order to instantiate a variety of different Virtual Machines (VMs). The monitor layer is the highest secure operating mode available on ARM processors, designed with the hardware security extension ARM TrustZone, which manages the interaction between two execution worlds. In this context, the firmware monitor layer ensures the isolation of each world by using ARM TrustZone and provides, at the same time, functions to enable a safe and secure communication between them. Therefore, the resources (e.g., memory, peripherals, interrupts, CPU, etc) of the safety critical RTOS, running in secure world, are totally isolated from non-critical applications executing in the normal world. In addition, the main advantage of such an implementation is to guarantee safety, security, and latency predictability, while enabling dynamically cores sharing to be shared dynamically between both OSes by prioritizing the safety critical RTOS. This

scheduling policy combined with an optimized context switch mechanism offers a close to native performance for real-time applications.

Concerning safety requirements, the firmware monitor layer has been designed and developed in order to meet the stringent requirements of the ISO 26262 certification. Indeed, the small footprint of this software component (~6000 Lines of code) aims to ease the certification process as well as reducing the certification cost. The ISO 26262 international standard for functional safety of electrical and/or electronic systems in production of automobiles is an adaptation of the Functional Safety Standard IEC 61508. ISO 26262 is a risk based safety standard, where the risk of hazardous events is assessed and safety measures are defined accordingly. The main goals are:

- Provides an automotive-specific risk-based approach for determining risk classes (Automotive Safety Integrity Level, ASIL).
- Uses ASIL for specifying safety requirements to reach an acceptable residual risk.
- Provides an automotive safety lifecycle (management, development, production, operation, service, decommissioning)
- Covers functional safety aspects of the entire development process (e.g., requirements specification, design, implementation, integration, verification, validation, and configuration)
- Provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety is achieved

To sum up, the firmware monitor layer provides a certifiable solution, which aims to consolidate mixed-criticality systems on a common hardware platform by isolating a secure instance of a safety critical RTOS with ARM TrustZone, while virtualization can be used as an option in the Normal world, such as Linux-KVM, giving the possibility to implement an additional layer of isolation between normal world applications.

In this context, Linux-KVM has been extended to provide close to native real-time performance for applications, which run in virtual machines. A mechanism of hierarchical and coordinated scheduling has been applied to the kernel of both host and guest systems in order to limit the overhead due to virtual machines. This technology affects tasks and I/O scheduling by optimizing latencies, execution time, and memory management where the minimal memory bandwidth can be guaranteed.

The following table gives an overview of the different measures for success, which are addressed by the technologies previously described.

ID	KPI	Description	Measure for Success
1	Achievable Safety Integrity Level	Maximum achievable Safety Integrity Level (e.g. ASIL-B, ASIL-C) according to ISO 26262 for the secure monitor firmware layer	1.1, 6.1, 6.2
2	Validated support for key real-time OS	(Boolean) The ARM JUNO development platform supports integration of FreeRTOS to be used as the OS for the supervision.	1.2
3	Maximum jitter induced by the secure monitor layer	Bounded value for jitter in the execution of the most critical real-time thread	1.2
4	Maximum overhead during the RTOS boot	Bounded value for overhead induced by the secure monitor firmware layer during the boot of the RTOS	1.2
5	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

6	Maximum latency overhead of applications inside a KVM virtual machine	Percentage of the overhead of the latency of KVM virtual machine on loaded system. Latency is measured with Linux tool “cyclictest” inside a virtual machine with and without CPU workload. The overhead is the difference between those two measurements.	2.1, 2.4
7	I/O latency inside KVM virtual machine is not affected by the I/O workload	(Boolean) The I/O latency of application inside virtual machine, on a systems with I/O workloads, is about the same value than on a system with idle medium.	2.1
8	Memory bandwidth isolation by construction	(Boolean) The architecture provides a memory bandwidth isolation between tasks	2.1
9	Memory bandwidth reservation for highest criticality level application	(Boolean) The architecture provides a memory reservation feature to preserve memory bandwidth of highest critical applications	3.1

Table 2 Overview of the different measures for success by the technologies

2.5 *Gateways for end-to-end segregation* as means for integration of mixed criticalities at chip-, network- and cluster-level

The gateways developed in DREAMS provide an interference-free communication mechanism in order to operate applications of different criticality on the same network without them being aware of each other. End-to-end segregation is ensured by all logical components on the network (switches and nodes), ensuring safety. By means of completely deterministic behavior, the cluster-level communication provides real-time operation. Security is ensured by means of implementation of authentication, confidentiality and integrity functions in hardware at gateways and in the network switches.

2.6 *Resource managers achieving virtualization for heterogeneous applications and platforms:* Mixed-criticality systems typically consist of subsystems with different programming models (e.g., message passing vs. shared memory, time-triggered vs. event-triggered) and have different requirements for the underlying platform (e.g., trade-offs between predictability, certifiability and performance in processors cores, hypervisors, operating systems and networks)

In a complex system fulfilling, or even recognizing, system wide, high level constraints, such as end-to-end deadlines, or reliability, is not possible by individual resources in isolation, but needs a system wide view and requires system wide decisions to be made. DREAMS provides services for system-wide adaptability of mixed-criticality applications consuming several resources via global integrated resource management.

The *resource management services* are realized by a *Global Resource Manager (GRM)* in combination with local building blocks for resource management. There exist three local building blocks for resource management: Local Resource Managers (LRMs), Local Resource Schedulers (LRSs) and Resource Monitors (MONs). These local resource management building blocks are located on each node. The GRM and LRMs are implemented as XtratuM partitions. These implementations remain similar for various different platforms.

The LRSs performs the runtime scheduling of resource requests (e.g., execution of tasks on processor) according to requirements. The MONs monitors the resource availability. Resource monitors also observe the timing of components (e.g., detection of deadline violations). The LRS and MON implementations are heavily dependent with the hardware and hence are specific to node architecture (different implementation for PPC and DHP platforms).

2.7 Support for *monitoring and dynamic reconfiguration of virtualized resources* as foundation for integrated resource management

The DREAMS Monitoring Services (MON) provide means for the runtime system to gather information from the system and the applications so the DREAMS Local Resource Manager (LRM) can take reconfiguration or adaptation actions to satisfy safety and performance requirements. MON are in charge to detect the different failures and are defined to satisfy the DREAMS requirements: (1) MON for core failures, (2) MON for deadline overrun and (3) MON for quality of service (QoS).

Hardware failures is one of the aspects controlled by the MON and more concretely the core failures, i.e., a core stopped working. For that purpose the MON executes a service regularly in each core to detect if the core is working correctly or not. If the core is working right, the service writes to a shared structure that it is in the nominal mode. Otherwise, if the core has failed, the service is not activated and is not able to update the shared structure. For an efficient detection, this operation must be accurately timed and ordered so each core performs this action at a predefined known time.

The MON for deadline overrun service extends the deadline warning detection method described in the literature. The idea is that each critical application monitors its execution and checks if the application is in danger of overrunning its deadline. If it is the case, then the MON service signals to the LRM that a deadline overrun will probably occur.

The MON for QoS service allows the Local Resource Manager to be kept informed of the level of resource usage of the application programs in each partition. When combined with the Monitoring for deadline overrun it allows the Local Resource Manager to take action more efficiently to avoid deadline overruns, thus improving the overall utilization of the system usage and the performance of non-critical applications.

2.8 *Integrated resource management for mixed-criticality systems* with monitoring, runtime control and virtualization extensions recognizing system wide, high level constraints, such as end-to-end deadlines and reliability

Resource management is a core service provided in the DREAMS middleware for system wide adaptability of mixed criticality applications. The main goals of the integrated resource management are:

1. Reconfiguration of a mixed-criticality system upon foreseen and unforeseen changes in its operational and environmental conditions.

2. Adaptability mechanisms for securely reconfiguring the system without interrupting or interfering with its execution.

Practically, the resource management services are realized by a Global Resource Manager (GRM) in combination with a set of Local Resource Managers (LRM). The GRM gathers information from the LRMs and provides new configurations for the virtualization of resources (e.g., partition scheduling tables or resource budgets). The GRM configuration can include different precomputed configurations of resources (e.g., time-triggered schedules) or parameter ranges (e.g., resource budgets).

Local resource management services consist of three major parts: Resource Monitors (MONs), Local Resource Schedulers (LRSs) and Local Resource Managers (LRMs). The MON monitors the resource availability and timing of components (e.g., detection of deadline violations). The LRS performs the

runtime scheduling of resource requests (e.g., execution of tasks on processor, I/O requests) based on the configuration set by the LRM. The LRM either adopts the configuration from the GRM to particular resources (e.g., processor core, memory, I/O) or selects a new configuration from the ones available and reports state of the resource (from MON) to the GRM.

DREAMS middleware relies on time and space partitioning principles. We consider that those principles are implemented at the chip-level by the XtratuM hypervisor, which is a technology involved in the project. Therefore, applications will be executed by a set of partitions. A partition is defined by one or multiple slots, each with a start time and a length. Inside a slot, several tasks can be executed.

2.9 Integration of *offline and online scheduling* algorithms, providing segregation of activities of different criticalities in a flexible way

DREAMS resource management services include QoS management and deadline overrun management to handle multiple criticality tasks in a segregated way. Segregation is further increased by using a hypervisor to statically partition resources for different application features. The flexibility in the schedule for integration of offline and online critical activities is achieved using job-shifting algorithm implemented by the partition local LRS. Furthermore, transition modes (with blackout intervals) are defined between nominal modes to reduce the mode switching delay and keep the system schedulability during a mode switch.

The job-shifting algorithm, defined as part of Task 4.1, estimates the free resources in an offline schedule and stores this information for online admission of aperiodic activities. This enables quicker aperiodic response times and allows efficient use of resources without reserving a processor bandwidth for the aperiodic activities. The algorithm is efficient and incurs very small overheads (i.e. in the order of a few microseconds on DHP). As all the critical partitions are responsible for implementing job-shifting independent of the activities in other partitions, the approach is highly scalable and efficient.

A transition mode is defined as an intermediate mode (between two nominal modes) which executes at most once after a mode switch request. For most of the industrial applications, the MaC/MAF of a node is usually large, due to which a mode switch request (released just after the start of MaC/MAF) might be fulfilled quite late (i.e. at the end of MAF) without transition modes. Transition modes define an intermediate schedule between the switching and to-be-switched mode to perform other tasks, for instance initiating new activities and changing configuration. These modes can be switched as soon as the mode-change blackout is over. In contrary to the system MAF, mode-change blackout intervals are very small, and therefore may lead to smaller mode-switching delay. Since these mode changes are implemented locally by each DREAMS node, the approach is very efficient and scalable.

2.10 Combination of *global strategies* with local resource monitoring and local management schemes

DREAMS provides services for system-wide adaptability of mixed-criticality applications consuming several resources via global integrated resource management. The approach in DREAMS is based on providing the system view on an abstract level to reduce the overhead of disseminating the global system state and only provide information requiring a system wide reconfiguration (as provided by WP4).

Practically, the resource management services are realized by a Global Resource Manager (GRM) in combination with a set of Local Resource Managers (LRMs) as explained in Task 3.2. The LRM sends its local configuration to GRM every Major Cycle (MaC), and can additionally ask for a global reconfiguration if it is unable to resolve the failure locally. The GRM gathers information from the

LRMs, and provides new configurations for the virtualization of resources. The GRM configuration includes different pre-computed configurations for nodes in the system.

When the MON informs the LRM of a core failure and the LRM finds no local reconfiguration that can recover from this situation, it locally reconfigures the critical tasks in priority if possible (pre-computed configuration) and some best effort applications may be removed. Then, the LRM informs the GRM that some applications cannot be hosted any longer on the node and requests the GRM to make a global reconfiguration. If the GRM finds a new global configuration that can assign some or all of the remaining tasks on new nodes, then it sends an order message with new local configuration to the LRMs whose nodes are involved in the global reconfiguration. The application requirements state that an application cannot split onto different multi-core. Thus, when a global reconfiguration must be performed, complete applications are reconfigured on different cores.

In the current implementation, LRM takes a local reconfiguration decision at the end of the MaC by collecting information from all monitors. This entails that several failures may happen during a MaC and decision could consider multiple failures. To avoid non-deterministic decisions, we impose reconfiguration graphs to be symmetric. The GRM must be informed via an update message of the node current configuration. From this information, it applies global reconfiguration changes if required and if possible. This entails for local reconfiguration graphs to be complete, i.e. from any configuration, the GRM must be able to deduce all failed cores (this constraint must be taken into account by the off-line scheduling tools.

The GRM and LRMs communicate regularly so as to efficiently combine local and global management schemes. There are two type of communication exchanges between the LRM and the GRM:

1. Update messages: These messages are sent by the LRM to the GRM every MaC. They can either be simple update informing the GRM of the current configuration of the node (corresponding to the LRM) or a global reconfiguration request so as to reassign some or all of the tasks that cannot be scheduled locally due to a failure.
2. Order message: This message is sent by the GRM to the relevant LRMs in case of a global reconfiguration.

2.11 Real-time *fault recovery* strategies based on dynamic reconfiguration

The fault recovery strategies have been described in deliverables D2.2.2 [17], D2.3.4 [18].

If a MON informs the LRM of a core failure, the LRM has two possibilities:

1. A local reconfiguration is possible according to its local reconfiguration table. In that case, it asks the LRS to change the plan at the end of the MaC to the new configuration and just sends an update to the GRM informing of the change in configuration.
2. No local reconfiguration can recover from the situation. In that situation, the critical tasks are locally reconfigured in priority if possible (pre-computed configuration) and some best effort applications may be removed. Then, the LRM informs the GRM that some applications cannot be hosted any longer on the node and requests the GRM to make a global reconfiguration

Reconfigurations can also occur due to internal deadline overrun of critical applications. More precisely, maximal deadlines of each task executing in a partition slot may be specified by the user. In that situation, MON is in charge of monitoring those internal deadlines and if there is an overrun, the LRM immediately stops the best-effort applications. It is described in detail in deliverable D2.3.4. The deadline overrun service can potentially severely degrade the performance of non-critical applications. To reduce that effect the DREAMS run-time is extended with a QoS service. The QoS service speculatively stops during short periods non-critical applications to avoid the deadline overrun service to take action, i.e., to stop non-critical partitions.

It is also important to deal with failures of the resource managers themselves. An LRM partition executes on each core at the end of the MAF/MaC and all LRM are synchronous. One among them is the Master. In case if the core executing the master LRM fails, another LRM is assigned the role of master based on an offline defined sequence. The GRM must receive an update message from the master LRM every MaC. Incase if the GRM doesn't receive a update from the LRM, it may trigger a global reconfiguration considering the node corresponding to non-responsive LRM has failed. The GRM is fail silent. GRM only makes global reconfiguration decisions when necessary, but it is not required for the continuous operation of the system. Thus, in case of GRM failure, the overall system dependability is not compromised as the system will still keep on executing; just no new global reconfigurations will be possible. Thus, this failure is not considered.

2.12 *Development process* ranging from modelling and design to validation of mixed-criticality systems

In order to achieve the project goals, the DREAMS development process, defined in D1.3.1 [3], is based on the IEC 61508 Safety engineering process, with additional steps for safety and timing. The focus is mainly on the development of the software part of the system.

It is furthermore a model driven development process for which metamodels have been defined (see D1.4.1 [5] and D1.6.1 [6]) that allow to model the system at the different design steps and abstraction levels. Further, to increase speed and reliability of development, tool support has been developed (D4.4.1 [4]) to support model transformations steps that correspond to variability management (see D4.3.*), scheduling and resource management (see D4.1.*) and configuration file generation (D4.2.*).

The assessment of the applicability of the model-driven development process comes to the conclusion that the DREAMS MDE approach is applicable to the application demonstrators, mitigates the complexity of the advanced DREAMS HW/SW platform and speeds of the development of DREAMS-based systems (see [D6.3.2], [D7.3.2], [D8.3.2]).

The collected in Section 9 of D1.1.1 is described in the following table.

ID	Name	Status	Traces	Comments
R 2.12.1	Definition of Model-to-model transformations.	Satisfied	[4.3.*] [4.1.*] [4.2.*]	*Variability Management (T4.3):* A two step process for binding variability (business and technical) has been defined, which represents the following Model-to-model transformations: • PIM with variability -> PIM • PM with variability -> PM *Scheduling design tools (T4.1):* A set of allocation and scheduling algorithms have been defined, which represent the following of Model-to-model transformations • PIM, PM -> PSM *Config. File Generators (T4.2):*

				The specifications of the configuration file generators contain a mapping between the PSM and the configuration model or the configuration file format.
R 2.12.2	Definition of implementation artefacts	Satisfied	[D1.3.1], Sec. 3.6, [D4.4.1], Sec. 3 [D1.6.1].	Implementation artifacts are the final products of the DREAMS development process, i.e. deployable applications on the one hand, and the corresponding platform configuration on the other hand. The initial draft of the collection of these artifacts [D1.3.1] has been refined in the description of the DREAMS toolchain [D4.4.1]. The platform-specific model [D1.6.1] defines the configuration artifacts of the DREAMS platform.
R 2.12.3	Offline mapping and real-time scheduling methods for MC systems	Satisfied	[4.1.3]	Mapping and scheduling algorithms have been defined. Their evaluation by the demonstrators is going on.
R 2.12.4	Consideration of online adaptation and resource management strategies	Satisfied	[4.1.3], Section 9,10,11	Recovery strategies are based on off-line designed and verified resource allocation configuration for which changes are applied online under predefined conditions.
R 2.12.5	Design-space exploration	Satisfied	[4.1.3], Section 3	DSE algorithm for product line exploration and evolutionary product optimization have been developed.
R 2.12.6	Timing analysis	Satisfied	[4.1.3], Section 5	End-to-end timing analysis of timing chains that cover the scheduling mechanism foreseen for the on-chip and off-chip communication and partition and task scheduling.
R 2.12.7	Meet-in-the-middle development methodology	Satisfied	[D1.3.1], Sec. 2.1.1.1, 2.2.1.1, 2.3.2.2	The usage of already developed sub-systems in a new development has been described for the different aspects: safety, security, timing.

R 2.12.8	Consideration of Certification, Validation and Verification	Satisfied	[D5.6.1]	Using a IEC61508 based Functional Safety Management process defines in a precise way the verification and validation procedures that are going to be applied.
R 2.12.3	The development (design, verification, validation) process shall foresee the definition of application timing requirements.	Satisfied	[D1.4.1], Sec. 7 [D1.3.1], Sec. 2.3. [D4.4.1], Sec. 3	Timing requirements can be described with the Timing View Point of the meta-model. The definition of timing requirements is foreseen by the development process.
R 2.12.9	Development of safety related parts	Satisfied	[D1.3.1], Sec. 2.1	The DREAMS development process includes a safety approach based on IEC 61508.
R 2.12.10	V-shape development process	Satisfied	[D1.3.1], Sec.1.4	The IEC 61508, which is used as based for the DREAMS development process, uses V-Models.
R 2.12.11	Requirement traceability support	Satisfied	[D1.3.1], Sec.1.4	The IEC 61508, which is used as based for the DREAMS development process, foresees requirements traceability.
R 2.12.12	Consideration of time and space partitioning mechanisms	Satisfied	[4.1.3]	The selected platform building blocks or scheduling mechanism support time and space partitioning: hypervisors, time triggered scheduling for partitions, tasks and the TT traffic class in the on-chip and off-chip network.
R 2.12.13	Reliability analysis / methods for active redundancy	Satisfied	[D4.1.2], Sec. 3.1.3.2, [D4.1.3], Annex, Sec. 2.2, [D4.3.2], Secs. 3.3, 3.4 [D4.3.3], Sec. 4	The design-space exploration allows exploring redundant mappings of components to execution of the platform (in conjunction with voting). Further, it supports exploring different safety architectures with varying numbers of redundant channels and implementation options (e.g., diagnostic units, diverse software component implementations).
R 2.12.14	DREAMS solutions integration in current	Satisfied	[D5.4.1]	The tools that support the development process have been successfully mapped to

	development processes			steps of the IEC 61508 safety engineering process. The evaluation of their actual integrability is part of the ongoing demonstrator assessments.
R 2.12.15	Process for variability resolution	Satisfied	[4.3.*]	In T4.3 has been defined a method for binding variability. It has been successfully applied to the WP7 demonstrator.

Table 3 Completeness of the model driven development process for mixed criticality systems with respect to the specific requirements

2.13 A *methodology and prototypes of tools* for mapping mixed-criticality applications to heterogeneous networked platforms including algorithms for scheduling and allocation, analysis of timing, energy and reliability

The goal of WP4 was to developed methods and tools that support the transformation steps of the model driven development process:

- variability management (see D4.3.*),
- scheduling and resource management (see D4.1.*)
- configuration file generation (see D4.2.*).

All transformation steps are covered by at least one tool, but can also be performed manually. The initial description of the applications, the hardware, the constraints and the variability must of course be performed manually.

In D4.4.1 are provided step by step descriptions of the usage of the tools in three different use cases. As explained in D5.4.1, the tool (prototypes) developed or extended in DREAMS fall into the following categories:

- Model Editors
  - AutoFOCUS3: logical and platform architecture
  - Timing Model Editor
  - Safety Model Editor
  - BVR: variability model editor
- Design Tools
  - Variability Management
    - BVR (product generator)
  - Design Space Exploration
    - AutoFOCUS3: DSE plug-in
  - Scheduling design
    - RTaW-Timing / Timing Decomposition
    - RTaW-Timing / On-chip TT Sched
    - TTEPlan
    - Grec
    - MCOSF
    - Xoncrete
- Verification Tools
  - IKL Safety Constraint Checker
  - RTaW-Timing/ Evaluation

  - Virtual Platform
- Platform Configuration File Generators
  - Xtratum CFG
  - TTE-Plan
  - On-Chip Ni CFG
  - Resource Management Service CFG Generator
  - Virtual Platform CFG

All data exchange needed for being able to use the different tools of the tool chain have been automated, to avoid error prone manual editing, and have been assessed positively by the demonstrators (see [D6.3.2], [D7.3.2], [D8.3.2]).

The completeness of the methods and tools with respect to the specific requirements collected in Section 4 of D1.1.1 is described in the following table.

ID	Name	Status	Traces	Comments
R 2.13.1	Generation algorithms of resource allocation configurations	Satisfied	[4.1.3], Section 4	A heuristic allows decomposing end-to-end latency constraints on timing chains into sub-constraints. If local schedules allow to satisfy the local constraints then also the global constraints are satisfied.
R 2.13.2	Criticality spectrum: combination of offline and online scheduling	Satisfied	[4.1.3], Section 9,10,11	Recovery strategies are based on off-line designed and verified resource allocation configurations for which changes are applied online under predefined conditions
R 2.13.3	Response time analysis algorithms	Satisfied	[4.1.3], Section 5	End-to-end timing analysis of timing chains that cover the scheduling mechanism foreseen for the on-chip and off-chip communication and partition and task scheduling.
R 2.13.4	Design-space exploration	Satisfied	[4.1.3], Section 3	DSE algorithm for product line exploration and evolutionary product optimization have been developed.
R 2.13.5	Performance	???	[D6.3.2]	The evaluation by demonstrator of the efficiency of the generated schedules is in progress.
R 2.13.6	Automation of configuring DREAMS systems	Satisfied	[4.2.*] [4.3.*]	We coupled the Base Variability Resolution technology (BVR) with the DSE to partly automate the design and optimization of products within a product-line. The

				resulting models are then input to configuration files generators.
R 2.13.7	Explicit configuration definition	Satisfied	[4.2.*] [4.3.*]	The DREAMS model-driven approach allow designer to either specify explicitly the system or to use design-space exploration to automatically explore a pre-existing product-line. Regardless, the resulting model can then be fed into the configuration generators.
R 2.13.8	Formal definition of Real-time faults detection and recovery strategies	Satisfied	[4.1.1], Section 6 [4.1.2], Section 7	Definition of faults. Definition of the recovery strategies for core failure and internal deadline overrun.
R 2.13.9	Tool chain	Satisfied	[4.4.1]	Most of the design activities of the DREAMS development process are supported by tool.
R 2.13.10	Continuous data flow through the tool chain	Satisfied	[4.4.1], Section 4	The exchange of design data with the central model has been automated through the implementation of importers and exporter for most of the tools.
R 2.13.11	Cross-domain applicability of methods and tool	Satisfied	[6.3.1], [8.3.1],	Not all platform services or mechanism are used in all demonstrators. But if used, then the corresponding tool is available applicable, as determined during the preliminary demonstrator assessments.
R 2.13.12	Configuration file generators	Satisfied	[4.2.1], [4.2.2]	Configuration file generators that automatically created configuration files out of a verified model of the system, have been developed for all building blocks of the DHP.

Table 4 Completeness of the methods and tools with respect to the specific requirements

2.14 *Test bed* for validation, verification and evaluation of extra-functional properties

The goal of T5.2 was to develop a framework for simulating and verifying the behaviour of a mixed-criticality system based on the DREAMS architecture.

The following tools have been developed in T5.2:

- Simulator

  - Virtual platform: simulator of multi-core chips with on chip-network, connected by an off-chip network.
  - Off-chip network simulator with fault-injection
  - low-level simulator of TTEthernet hardware components for deriving tests for the actual implementation
- Physical Fault Injection Tools
  - EtherCat fault injection tool
  - STNoC fault injection tool
- Formal verification
  - Formal verification framework STNoC components

The completeness of the tools with respect to the specific requirements collected in Section 5 of D1.1.1 is described in the following table.

ID	Name	Status	Traces	Comments
R 2.14.1	Gatway simulation building block between off-chip and on-chip networks	Satisfied	[5.2.2], Sec. 3.5	Implemented as GEM5 simulation model, which forward messages between the on-chip and off-chip network according to the rules that apply to TT and RC traffic classes.
R 2.14.2	Simulation building blocks for off-chip networks	Satisfied	[5.2.1], Sec. 3.1.1	Implemented as Opnet simulation model.
R 2.14.3	Simulation building blocks for on-chip networks	Satisfied	[5.2.2], Sec. 3.3	Implemented as modification of a GEM5 Garnet Fixed Pipeline Model simulation model.
R 2.14.4	Simulation building for execution service	Satisfied	[5.2.2], Sec. 4	Implemented as OVPSim simulation model.
R 2.14.5	Configuration interfaces to integrate the simulation environment into the DREAMS development process	Satisfied	[5.2.2], Sec. 6	Configuration interfaces have been defined for several simulation modules but not all: off-chip network, on-chip/off-chip gateway, off-chip network, nodes with a core.
R 2.14.6	Fault injection at communication networks	Satisfied	[5.2.3], Sec. 5 [5.2.3], Sec. 2 [5.2.3], Sec. 3	Simulated fault-injection for the off-chip network (OPNET). Real-time fault-Injection tool for EtherCat. Emumlator based fault-Injection Framework for the on-chip network.

R 2.14.7	Simulation building blocks for fault injection at chip level	Partially Satisfied.	[5.2.3], Sec. 3	An emulator based framework for the on-chip network has been developed instead, see R.6.7
R 2.14.8	Configuration interface of fault injection mechanisms (e.g., fault containment units, failure modes, failure rates)	Satisfied	[5.2.3], Sec. 5 Sec.2	Fault injection can be configured through the GUI of the simulation model for the off-chip network. Fault injection can be configured through the GUI of the real-time fault injection tool for EtherCAT
R 2.14.9	Analysis and evaluation of simulation results with respect to timing and reliability	Partially Satisfied	[5.2.2], Sec. 6	Trace format has been specified for timing events, which can be imported into a tool that can present the results in different ways, but the simulation modules do not produce traces.
R 2.14.10	Multi-source simulation building block with reusability of test cases	USIEGEN	[5.2.2], Sec. 6	Multi-source simulation is able to read the generated file that includes the specification of the timing, contents and control information for time-triggered, rate-constrained and best-effort messages.
R 2.14.11	Transfer networking test results from simulation to physical systems	Satisfied	[5.2.3] Sec. 8.	A corresponding framework has been developed TTEthernet.
R 2.14.12	Support for formal verification		[5.2.2], Sec. 8	Formal verification of SVA properties of STNoc design.

Table 5 Completeness of the tools with respect to the specific requirements

Certification is a formal procedure in which a certified body assesses and verifies the attributes, characteristics of a system, subsystem or element in accordance with established standards. In the safety domain, safety certification aims to assess the compliance of a system to the requirements of a safety standard (e.g., IEC 61508 and ISO 26262). The traditional approach to certification relies on the certification of the whole system, where if a safety aspect of the system changes, the entire system shall be re-certified.

Embedded computing platforms commonly follow a federated architecture paradigm in which each Distributed Application Subsystem (DAS) is implemented on its own standalone distributed hardware base with a well-defined functionality. However, the soaring demand for high performance and increasing functionality challenge the viability of this approach, leading to the increasing trend of moving towards integrated architectures. As a consequence, system engineers aim at the integration of multiple applications with different criticality levels (e.g., safety, security and real-time) on the same embedded computing platform. A system that combines applications of different criticality is often referred to as a mixed-criticality system.

From a product line perspective, individual products (e.g., mixed-criticality systems) are typically not independent but belong to families of products. The development and certification of product families are usually affected by the variability approach that leads to significant and potentially unacceptable increase of engineering and certification costs. Variability is the ability to change (customize/extend/configure) the HW or the SW for a particular context. For example, in a wind turbine product family that is composed of a set of supervision and control units and [1:N] wind turbines, the variability can be applied in the HW (e.g., change of platform) and/or the SW (e.g., change of programming language).

This technology group paves the way towards the competitive development and certification of mixed-criticality embedded computing platforms, providing solutions to manage variability and complexity, increase re-usability and reduce the engineering and certification time and cost.

2.15 *Modular safety-case* for mixed-criticality systems

A 'safety case' "represents an argument supporting the claim that the system is safe for a given application in a given environment". It provides I) arguments to demonstrate that the safety properties are satisfied and the risk has been mitigated, II) a notation mechanism that is often required as a piece of the certification process and III) interoperability among different standards and domains (e.g., avionic, automotive, railway). A well partitioned safety case limits the impact of changes to a reduced area of the safety case, and enables the reusability of these parts. On this basis, the implementation of modular safety cases enables the reusability of predefined modules, reducing the overall complexity (simplification strategy) and supporting the limitation of change impacts to specific modules.

FP7 DREAMS project contributes with the definition of modular safety cases for selected 'building-blocks' of mixed-criticality systems: safety hypervisor and partition(s) (D5.1.1), multicore COTS processor(s) (D5.1.2) and mixed-criticality network(s) (D5.1.3). IEC-61508 is selected as the safety reference standard because results could potentially be extended in the future to different domain specific standards that take IEC-61508 as a reference standard (e.g., railway, automotive, vertical transportation, machinery, etc.). Within the DREAMS project this standard only applies to the wind power demonstrator. The different modular safety cases have been integrated within the wind power demonstrator by means of the toolchain developed in WP4, where recommendations on how to proceed with the integration and the needed documentation to face a certification process have been suggested.

The assessment of the modular safety cases has been done in two different phases. The first assessment by the certification authority TÜV Rheinland was done for each modular safety case, where the wording was updated based on the recommendations. In a second phase the assessment have been done within the wind power use case.

There are no real measures of the improvement obtained by using the modular concept against a global safety case concept. These numbers can be obtained by following a real product lifecycle where after the first certification the product needs a recertification. When doing this modular recertification, the experience obtained from other global certification processes can deal to real numbers regarding the impact of modularity against global certification processes.

2.16 Architectural support for the eased definition of *mixed-criticality product lines* with certification support across product lines

Mixed-criticality systems considered in DREAMS have one special property that affect certification, the high degree of re-configurability and re-configurability when working with product lines. To deal

effectively with this property, we have demonstrate how explicitly modeling product-lines of MCS makes it possible to generate detailed and product specific arguments for any individual product certification.

The DREAMS toolset supports the semi-automated design-space exploration (DSE), seeking for the best possible product configuration for a given set of requirements, also including safety requirements. During the design stage, these tools automatically select alternative candidate deployments of the logical components on the target hardware, resolving the variability models and going through a number of verifications in a safety evaluator component: the Safety Compliance & Rules Checker. As for the certification, this toolset also automatically assembles an argumentation model for each product sample. The outcome after completion of the DSE is a partial argumentation model, which will be mapped to a set of certification documents according to the Functional Safety Management process chosen by the DREAMS user. These documents are generated in a semi-automated fashion, taking into account that part of the final evidences would come from verification and validation (V&V) activities to be carried out at later stages of the project, and some of these could not be available at DSE phase.

2.17 *Variability* in applications and platforms as another architectural dimension to handle different criticalities and domains

In DREAMS, variability modeling is supported by the BVR tool. Specific extra-functional analysis techniques are tailored for MCS and have been integrated in the DREAMS model-driven process: safety, timing, and reliability. The DREAMS project approaches design-space exploration at the business and technical levels, using variability models and an evolutionary optimization. At the business level, the variability models capture the design decisions that govern what functionality will be offered by the system. At the technical level, the evolutionary DSE algorithm explores alternative engineering decisions resulting in contrasted trade-off between different extra-functional system properties, safety, timing and reliability. Although complementary, the final products yielded by both approaches remains within the limits of the design problem.

2.18 *Different domains and market features*, and also optimal selection and configuration of components and platform services for mixed-criticality systems

The DREAMS project results provide a structured methodology to ease the certification of mixed-criticality product lines built upon the certified and certifiable components of the DREAMS harmonized platform. This methodology has been applied to different domains and markets, Energy (wind power use case WP7), Healthcare (WP8) and Avionics (WP6). The level of integration of the methodology in the demonstrators is different but it can be said that all the results are applied in one way or another. All the results (safety cases, toolchain integration, variability, product line) are used within the wind power demonstrator. IEC-61508 is selected as the safety reference standard and it is integrated in all the toolchain elements as the de-facto standard. This way results could potentially be extended in the future to different domain specific standards.

2.19 Demonstrators of avionic, industrial, and healthcare mixed-criticality applications that integrate the concepts and tools of DREAMS

The DREAMS conceptual outcomes, including the meta-models from WP1, the chip-level technologies from WP2, the cluster-level technologies from WP3, the development methods from WP4, and the certification/validation methods from WP5, that were integrated through the final integration have been deployed in all of the three demonstrators of the DREAMS project. For more information please see the respective deliverables, i.e., D6.2.2 for avionics, D7.2.1 for windpower and D8.2.1 for heal-care.

2.20 Practical demonstration of cross-domain applicability of the developed framework and methodology

In the frame of the project, the toolchain and the underlying metamodels have been validated in three application demonstrator that represent a broad range of mixed-criticality systems that differ in the various dimensions such as the following:

- Real-time requirements ranging from hard real-time systems with varying cycle times and performance demands (WP6, WP7) to soft real-time systems (WP8).
- Fault-tolerance requirements varying from fail-operational systems with resource management (WP6) over fail-safe (WP7) to not safety-critical (WP8) systems.
- Security requirements, covered in WP6 (integrity and authenticity of resource management) and WP8 (also including confidentiality).

The successful application of the MDE methodology to these representative application demonstrators, in particular to synthesize the required configuration artifacts for the building blocks of the cross-domain HW/SW platform, allows to conclude that the DREAMS modeling and tooling approach is also applicable to mixed-criticality systems in other domains such as automotive or factory automation. If needed for a particular domain, established domain-specific metamodels can be used to describe inputs of the workflow, such as the (architecture of) applications (e.g., using IEC 61311 models for industrial control applications, or AUTOSAR for automotive applications). The open DREAMS metamodel [D1.4.1, D1.6.1] and development process [D1.3.1] provides the basis for sustainable investments into model-transformations that translate domain-specific representations into the generic logical component models used in DREAMS. Thanks to the layered architecture of the platform-specific model (PSM) that is used to capture the results of the DREAMS toolchain in a tool- and platform-independent manner, additional back-ends for the generation of domain-specific configuration artifacts can easily be integrated. Further, this PSM-approach also allows to integrate adapters interfacing the DREAMS toolchain into established domain-specific toolchains that are deployed in large quantities in a number of industries (e.g., automotive).

2.21 Establish and support a *sustainable DREAMS community for system integrators and component developers*, who will use the project results for developing mixed-criticality applications and foster the future refinement and extension of DREAMS building blocks

A Mixed Criticality Forum has been set up that allows related projects to exchange results, ideas and developments (see i.e. D9.1.2).

2.22 Produce *technical training ma*terials as part of an overall strategy to encourage uptake of results, while at the same time gathering feedback for the refinement of technical concepts

A roadmap of the DREAMS technologies was the basis for the design of a set of training videos that could present the project results in an easy and understandable way. A specific DREAMS channel in YouTube system has shown to be effective. The number of visits and the statistics of the video produced has been important to gather feedback of the technology used for the training activities. This material is also planned to be used in the Summer School as complementary material of the lectures.

2.23 Actively pursue standardization of the DREAMS architecture. The community will liaise with standardization bodies and provide a single point of contact for interested stakeholders

Multiple bodies for standardization were and are permanently approached in order to communicate the results relevant to specific standards within the working groups (See i.e. D9.2.1.).

2.24 Develop a European innovation roadmap for research in mixed-criticality systems and provide a community infrastructure

Significant efforts have been put aligning research and industry by developing a research and innovation roadmap on the topic of mixed criticality to achieve critical mass and facilitate breakthrough innovations in the medium and long-term. A Scientific workshop for developing innovation road map was held in Jan 2015 in Vienna where researchers from academia and industry had participated. Specific points for in-depth discussion were identified in this workshop. During the cluster workshop session (HIPEAC, 2016) discussion regarding defining mixed criticality, the role and influence of standards, role for modular certification, MCS Methods, Tool evolvement, Business models and eco-system impact, etc., were held in small groups. Co-operation has also been established with other projects in area of mixed criticality research. A convergence workshops was held at HIPEAC'2017. Another workshop is planned towards the end of the project together with the summer school. The innovation roadmap will be finalized by M45, and concluded with a White paper on mixed-criticality research and innovation in M48.

3 Objectives vs. Demonstrators

In the previous section, it was shown how the project objectives are fulfilled with the DREAMS technological results. This section[3] provides an overview on how the fulfillment of the project objectives are evaluated by the KPIs in the demonstrator assessments.

Number	Title of Objective	KPI ID in WP6	KPI ID in WP7	KPI ID in WP8
1	Architectural Style und Modelling Methods based on Waistline Structure of Platform Services			
1.1	Consolidation and extension of architectural concepts from previous projects (e.g., RECOMP, GENESYS, ACROSS, ARAMIS) towards a new architecture style for the seamless virtualization of networked embedded platforms ranging from multi-core chips to the cluster level with support for security, safety and real-time performance as well as data, energy and system integrity	11		
1.2	Waistline architecture with domain-independent platform services that can be successively refined and extended to construct more specialized platform services and application services. The platform services shall provide a stable foundation for the development of applications and enable the safe and secure composition of mixed-criticality systems out of components	11	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	1, 2, 3, 4, 10, 22, 31, 32, 33, 34, 35, 36
1.3	Models of hierarchical platforms comprised of networked multi-core chips to enable MDE	11	11, 12, 13	11, 12
2	Virtualization Technologies to Achieve Security, Safety, Real-Time Performance as well as Data, Energy and System Integrity in Networked Multi-Core Chips			
2.1	Certifiable platform services for virtualization and segregation of resources at cluster and chip-level (e.g., I/O virtualization, message-based networks and memory architectures, dynamic resource management)	1,5	14, 15, 16, 17, 18	5, 6, 7, 8, 15, 16, 17
2.2	Gateways for end-to-end segregation as means for integration of mixed criticalities at chip-, network- and cluster-level	10	19	13, 14
2.3	Resource managers achieving virtualization for heterogeneous applications and platforms: Mixed-criticality systems typically consist of subsystems with different programming models (e.g., message passing vs. shared memory, time-triggered vs. event-triggered) and have different requirements for the underlying platform (e.g., trade-offs between predictability, certifiability and performance in processors cores, hypervisors, operating systems and networks)	2, 3, 6, 7	20, 21	18

[3] This section was added based on a review recommendation.

2.4	Support for monitoring and dynamic reconfiguration of virtualized resources as foundation for integrated resource management	2, 6		
3	Adaptation Strategies for Mixed-Criticality Systems to Deal with Unpredictable Environment Situations, Resource Fluctuations and the Occurrence of Faults			
3.1	Integrated resource management for mixed-criticality systems with monitoring, runtime control and virtualization extensions recognizing system wide, high level constraints, such as end-to-end deadlines and reliability	2, 6		
3.2	Integration of offline and online scheduling algorithms, providing segregation of activities of different criticalities in a flexible way	1, 2, 5, 6		5, 9, 37
3.3	Combination of global strategies with local resource monitoring and local management schemes	3, 7, 9		
3.4	Real-time fault recovery strategies based on dynamic reconfiguration	4, 8		
4	Development Methodology and Tools based on Model-Driven Engineering			
4.1	Development process ranging from modelling and design to validation of mixed-criticality systems	9, 10, 11	22	19, 20
4.2	A methodology and prototypes of tools for mapping mixed-criticality applications to heterogeneous networked platforms including algorithms for scheduling and allocation, analysis of timing, energy and reliability	2, 6, 9, 10, 11	23, 24, 25	
4.3	Test bed for validation, verification and evaluation of extra-functional properties	11		
5	Certification and Mixed-criticality Product Lines			
5.1	Modular safety-case for mixed-criticality systems		26, 27	
5.2	Architectural support for the eased definition of mixed-criticality product lines with certification support across product lines		28	
5.3	Variability in applications and platforms as another architectural dimension to handle different criticalities and domains		29	
5.4	Different domains and market features, and also optimal selection and configuration of components and platform services for mixed-criticality systems		30	21
6	Feasibility of DREAMS Architecture in Real-World Scenarios			
6.1	Demonstrators of avionic, industrial, and healthcare mixed-criticality applications that integrate the concepts and tools of DREAMS		1, 2, 3	
6.2	Practical demonstration of cross-domain applicability of the developed framework and methodology		31, 32, 33	1, 5
7	Promoting Widespread Adoption and Community Building			
7.1	Establish and support a sustainable DREAMS community for system integrators and component developers, who will use the project results for developing mixed-criticality applications and foster the future refinement and extension of DREAMS building blocks		34	
7.2	Produce technical training materials as part of an overall strategy to encourage uptake of results, while at the same time gathering feedback for the refinement of technical concepts		35	

7.3	Actively pursue standardization of the DREAMS architecture. The community will liaise with standardization bodies and provide a single point of contact for interested stakeholders			
7.4	Develop a European innovation roadmap for research in mixed-criticality systems and provide a community infrastructure			

Table 6 Assessment of the project objectives by KPIs in three demonstrators

4 Summary of Results from Assessment

The following subsections provide a short summary of the demonstrator assessment documents D6.3.2 (WP6, avionics), D7.3.2 (WP7, wind power) and D8.3.2 (WP8, health care).

The evaluated results include the DREAMS architectural style and meta-models (WP1), the STNoC (WP2), LRS (WP2), XtratuM (WP2), the secure firmware monitor layer (WP2), resource management with a focus on fault recovery tolerance (WP2, WP3), off-chip communication services (WP3), the DREAMS toolchain (WP4), the safety communication layer SCL (WP5), the EtherCAT data logger and fault injector (WP5) and the virtual platform (WP5).

4.1 Avionics Demonstrator (WP6)

We demonstrated the applicability of the DREAMS architecture and tooling (AF3 with its extensions, GRec, MCOSF, RTAW-Timing, Xoncrete and the network configuration tools) by creating a distributed demonstrator composed of three multi-core nodes (2 PPC platforms and 1 DHP) interconnected through a real-time switch. On each node one or two critical applications were deployed. These critical applications communicate through the network or shared memory depending on their deployment, which could change at runtime due to core failure events.

The critical applications were deployed on top of the DREAMS resource management solutions, which themselves were in-turn deployed on top of the XtratuM hypervisor with the DREAMS Abstration Layer (DRAL) interface. The demonstrator was further completed with non-critical applications, to demonstrate the capabilities of the DREAMS solutions to handle mixed-critical systems.

The evaluation demonstrated that thanks to the DREAMS developed solutions:

- Interferences suffered by critical applications in the multi-core nodes were controlled when deploying non-critical applications running in parallel in the same node. Deadlines requirements were reduced when compared to a deployment when interferences were not managed, additionally the observed maximum and median execution time of the critical applications was also reduced when compared to the same interference deployment.
- The non-critical applications were able to exploit the nodes performance, extracting between 70% and 90% of the available performance of the multi-cores on the time windows of the schedule with the critical applications.
- Availability of the system was improved with the core failure resource management solutions implemented by the GRM, and LRM and MON modules, and configuration generated by through the GRec tool. Thanks to the XtratuM DRAL interface critical applications did not require to be aware of their deployment (i.e., communications local or through the network shared the same implementation, DRAL managing the underlying communication mechanism, memory or network), and communication channels were automatically reconfigured depending on the deployment selected by the resource management. Furthermore, the implementation of the GRM and the LRMs modules implementation ensured bounded reconfiguration times.

In addition, other project solutions were exploited in the demonstrator, like (1) the security library used for the critical applications and resource management communications, (2) the job-shifting technique and the MCOSF tool to enhance the scheduling and execution of critical applications with aperiodic tasks, and (3) the usage the simulation tools to study the resilience of the demonstrator to communication (network) faults.

4.2 Wind Power Demonstrator (WP7)

DREAMS wind power demonstrator is 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.

The demonstrator has achieved 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.

47 KPIs were defined when the assessment plan was defined. These indicators include measured that evaluate different aspects of the demonstrator: architectural design, modeling and tools, performance, certifiability, etc.

Out of 47 KPIs, 44 have equaled or exceeded the threshold initially defined. This translates into a big percentage of originally established objectives being fulfilled.

Mapping between KPIs and objectives is done through defined Measures for Success. Each objective is itemized in different Measures for Success. Likewise, each Measure for Success is linked to one or several KPIs.

There are seven objectives that are common to the three demonstrators developed within DREAMS. In addition, there are also another five objectives which are specific to the wind power demonstrator.

Common objectives are related to architectural style and modeling; virtualization technologies; adaptation strategies for mixed-criticality systems; development methodology and tools; certification; feasibility of DREAMS architecture in real-world scenarios and community building. Based on the KPIs obtained in the assessment process, these objectives have been fulfilled. One of the main important consequences of the fulfillment of the objectives stated above is that DREAMS tools and technologies offer time and cost reduction regarding processes such as the development steps, certification and re-certification procedures.

As far as wind power demonstrator specific objectives are concerned, they are related to higher integration of mixed criticality systems; reduced certification effort for safety protection system; increased flexibility in the protection system; incorporation of mixed criticality networks; system complexity and variability management. The results of the KPIs calculated in the assessment have been positive and the state that the integration of the safety protection system with the control and supervision system has been positive. One of the main benefits of this integration is the increase in capabilities and programming flexibility of the protection system.

4.3 Healthcare Demonstrator (WP8)

The Healthcare demonstrator has been an important challenge to address since all the technologies, integrated in the final demonstrator have been developed in the context of DREAMS project in order to achieve the isolation of mixed-criticality healthcare and entertainment data (e.g., ECG and video streaming) on a single wired network, while ensuring the isolation and prioritization of ECG's critical traffic. In this context, three main technology blocks have been integrated, namely the DREAMS Harmonized Platform (DHP) with XtratuM hypervisor to route the ECG critical data through the Time-Triggered network to the ARM Juno server where Linux-KVM combined with the Secure Monitor Firmware ensure the safety co-execution of the ECG application along with the video streamer for the STM32 Smart display. In addition, an Odroid board is used to receive ECG packets from the Bluetooth device called ST BodyGateway.

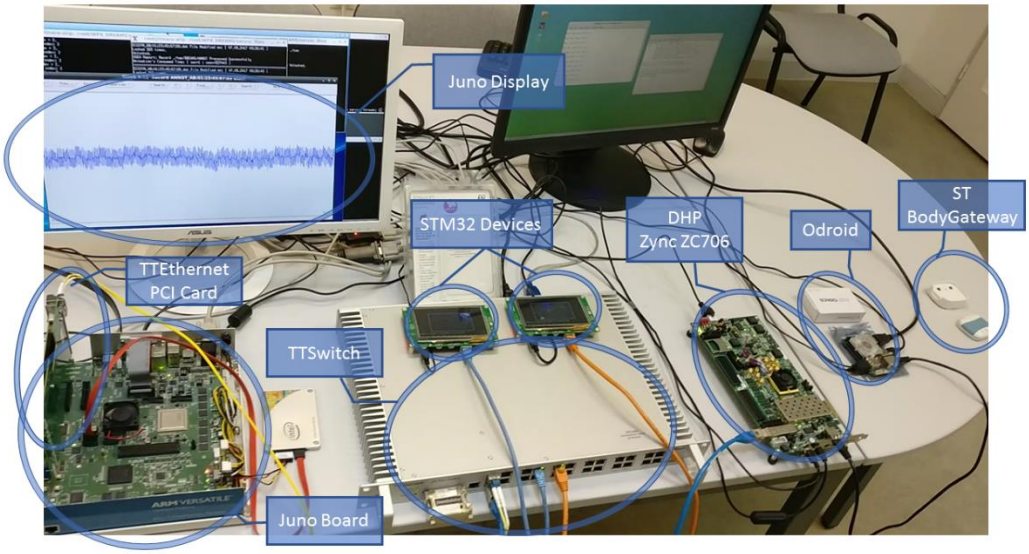

Figure 1 Healthcare demonstrator overview

The assessment of the healthcare demonstrator was reported by a number of KPIs in D8.3.2. Each KPI is described with its ID, its name, the target value and the assessed value. A last column indicates to which project objectives the KPI evaluation provide input to.

Overall, all the KPIs have been evaluated with success. Indeed, the DREAMS technologies integrated in the demonstrator allow a strong isolation between the critical data (e.g., ECG) and the non-critical one (e.g., video streaming), while keeping latency for mission-critical tasks low enough to meet the real-time constraints. However, two points for improvements have been identified during the evaluation:

- The Juno platform, which is used as a Hospital server in the Healthcare demonstrator, is divided in two partitions. Indeed, the Secure Monitor Firmware implemented on the Hospital server enables the co-execution of a trusted partition, which executes a Real-Time Operating System (i.e., FreeRTOS), along with a Non-Secure one running Linux/KVM. For integration facilities, the TTEthernet device connected to the Juno board has been mapped to the Non-Secure partition, which contains Linux/KVM, since TTEthernet drivers are already available for Linux kernel Intel/x86. However, it is important to notice that the TTEthernet device should be mapped to the Trusted partition for a final product in order to ensure a full isolation of the ECG critical data. Regarding the demonstrator, this point has been mitigated by creating a shared memory between Linux/KVM and FreeRTOS. Therefore, FreeRTOS is able to monitor the execution of the ECG application as well as the correct reception of Time-Triggered data.

- The number of ST Body Gateway devices that can be simultaneously connected to the healthcare demonstrator is six. However only four devices can be visualized (ECG display and cardiac disease detection) at the same time in the Hospital Server due to a lack of computing power of the server.

5 Possibility of Shutting Down the DREAMS Services

A vast range of technological building blocks have been addressed in DREAMS. Each result provides a service and addresses one of the project objectives. In order to highlight the contribution of the technologies, the possibility of shutting down of each of the technologies has been analyzed to be demonstrated during the final demonstration. The below table lists the DREAMS technologies and discusses the effect of shutting down of individual DREAMS technology during the demonstration. Moreover, it discusses whether it can be shut down to demonstrate the effect of not having that specific technology[4].

5.1 Shutting Down of Services

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
Archi-tecture	Architectural Style including waistline structure of services and hierarchical system structure	Without the DREAMS architectural style, there will be no support for cross-domain usability and an independent development of platform services (secure and fault-tolerant global time base, timely and secure communication services for time and space partitioning, timely and secure execution for time and space partitioning, integrated resource management for time and space partitioning).	As the architectural style is used at the design time, there will be no possibility of shutting down this technology.	Yes	Yes	Yes
DREAMS Model-Based Development and Tooling	DREAMS model-based development process (from application model, platform model to platform-specific model and generated configurations for the platform)	The *DREAMS model-based development process* is essential for the efficient development, deployment and integration of mixed-critical applications onto the DREAMS platform. Without this process and the tools supporting it, the developing the demonstrator applications and creating the corresponding configuration artifacts for the DREAMS platform building	The configuration yielded by the *DREAMS model-based development process* and tool-chain will be an intrinsic integrated part, i.e. it is not possible to disable or remove this design-time result during a presentation of the demonstrator. However, the size of the complexity of the generated configuration can be estimated	Yes	Yes	Yes

[4] This section was added to this deliverable based on a review recommendation.

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		blocks (e.g., NIs of NoC, hypervisor, etc.) would be much more complex and time consuming, especially considering the different iterations that are typically required to obtain the final demonstrator version. On the one hand, several design problems addressed by the DREAMS model-based development process are NP-hard (application to platform mapping, scheduling, etc.). On the other hand, manually creating the configuration artifacts is tedious and error-prone since consistency is very hard to achieve manually (e.g., scheduling entries for involved execution and communication resources of a single event chain) and requires expert knowledge of the devices and services to be configured that is encapsulated in the configuration generators.	by inspecting the corresponding configuration artifacts. Further, it could be discussed which steps and which in-depth knowledge of the involved building blocks of the DREAMS platform are necessary to manually generate the artifacts required to develop and deploy a sample application.			
	Application and platform model based on DREAMS meta-models (PIM, PM, PSM)	The selected model-based development process is based on a *common meta-model*, which ensures the consistency of different artifacts and enables to trace artifacts created in different phases of the development process. Further, it enables to more efficiently integrate tools based on a common interface where at most one import and one export module is required for every tool. Compared to the approach implemented by this result, direct tool-to-tool integration would have the following disadvantages: 1) more data-exchange modules likely to be required 2) solution is not sustainable, but tied to concrete tool-	Removing the *application and platform model based on the DREAMS meta-model* from a demonstrator has the same effect as not applying the result *DREAMS model-based development process* (see corresponding discussion how to shut down that result). Further, it is certainly not feasible to construct a development process and a tool-chain that is not based on a common meta-model only for the purpose of evaluating the selected approach. However, the number of transformation and import/export modules that would have been required to	Yes	Yes	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		implementations, which is inconsistent with the DREAMS approach that encourages to instantiate the architectural style into different implementations (and hence also different configuration tools).	obtain an alternative implementation of the DREAMS development process (i.e., with other tool-integration approaches) can be calculated.			
	Multi-objective optimization techniques for design space exploration (DSE)	Without the *MOEA-DSE* result, no tool support is available for the following tasks in the design and dimensioning process of DREAMS systems: • Calculation of component-to-partition mapping that satisfies the constraints defined by the system integrator (e.g., temporal constraints, safety constraints, etc.) and that is Pareto-optimal w.r.t. the selected criteria (e.g., energy consumption, cost). • Resolution of technical variability, i.e., selection of variants of logical components with different characteristics (e.g., WCET, memory consumption) Selection of safety architectures (e.g., 1oo2, 1oo2D, etc.) that enable to achieve a demanded SIL on a given platform.	It should be noted that the problems addressed by *MOEA-DSE* have a strong interdependency: While the selection of the safety architecture and the resolution of component variants affects the set of logical components to be passed to the mapper, the deployment to the physical architecture determines the SIL that is actually achievable. Without tool-support, engineers may manually construct single solutions that satisfy the safety constraints based on their experience and existing designs. However, exploring different variants that take into account further characteristics (e.g., energy consumption) causes to much effort.	No	Yes	No
	Model-to-model and model-to-text transformation framework	The M2M and MT2 framework constitutes the backend of the *DREAMS model-based development process* that is used to generate the actual building block-specific configuration. I.e., removing this result would have the same	See discussion provided for *DREAMS model-based development process*.	Yes	Yes	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		effect has not applying the DREAMS model-based development process.				
	Algorithm for the generation of the time-triggered communication in the on chip NI	The scheduling mechanisms of the ST-NoC do not prevent resource contention through time partitioning, but solve them by scheduling rules (FIFO, priority). In that case, a worst-case timing analysis algorithm is needed to verify that the latency constraints are met. But for wormhole routing, such an algorithm does not exist. To overcome this problem, an algorithm has been developed that defines transmission offsets (admission control in the Ni at the input of the NoC) such that no resource contentions may occur for TT flows. As a result, the NoC traversal times are always equal to the minimal traversal time. Based on this fact and the defined transmission offsets, it is possible to compute upper bounds on network traversal times and to verify and guarantee delay constraints. Without the algorithm that generates transmission offsets that induce predicable delays, it is not possible to verify network traversal times of TT flows. Only simulation can be used to do statistical prediction, but without the possibility to identify worst-case delays and thus without the possibility to provide absolute guarantees.	Instead of the transmission offsets defined by the algorithm that allows guaranteeing timing constraints, one could use manually created or randomly generated transmission offsets. These alternative configurations may induce resource contention during the traversal of the NoC, which lead to longer traversal times. Notice however that in a concrete case these delay increases may be “hidden” by other delays that vary when running the system with different configurations. In general, the probability that these alternative configurations lead to “deadline overruns” during demonstration is very low, since worst-case or unfavorable scenarios have by nature a very low probability to occur.	Yes (implicitly)	Yes	Yes (implicitly)
	Tool for offline cluster-level scheduling including tool adaptation for the	In principle configuration is also possible w/o a tool but has proved to be so complex and time consuming that it is not practicable to work w/o	Since the tool is used “off-line” shutting down is not applicable in this context	Yes	No	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
	(TTEthernet) configuration of DREAMS clusters enabling DREAMS-specific configurations.	tool. (If you want to drive a nail into a wall, you also use a hammer and not your bare fist!)				
	Architectural exploration as a means for optimizing configurations	Architecture exploration techniques search the spaces of possible products for one that exhibits some good properties of interest. The number of such products grows exponentially with the number of features and quickly exceeds designers' ability to review all possible product, by hand.	Architectural exploration at the product-line level, remains an optimisation. It helps designers spot relevant features combinations is a space otherwise too large for manual exploration. Without it, designers rely on their expertise, at the risk of overlooking better suited products, which would have a stronger impact on the market.	No	Yes	No
	Tools for model transformation to generate the configuration file for the execution model based on XtratuM	The configuration tools permits to capture the system specification and build a system model that allows to analyse the coherence, validate the system, generate schedules and produce the final configuration file for the execution platform based on XtratuM. The generation of the execution model for XtratuM is the final process in the modeling process that allows to achieve guarantees or evidences of the process. This is required in any certification process	Manual edition of the configuration file can introduce errors, incoherence and, what could be more important, lack of traceability of the design process.	Yes	Yes	Yes
	Algorithms for generation of static real-time scheduling plans for partition management based on the temporal model	Mixed criticality systems integrate critical and non-critical applications. The sum of several applications with its internal periodic and non-periodic activities can deal to very complex systems. The generation of a static plan considering the temporal constraints is strongly required. The Xoncrete tool used for this	Static plan generation without a scheduling tool from the model is a very hard process that has to be performed every time the model parameters or temporal constraints are modified. As an example, for the avionics model, with 6 partitions and more than 20 periodic and	Yes	Yes	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		purpose will permit to generate, from the system model, a multi-plan and multi-core schedule for the hypervisor.	aperiodic activities in a hyperperiod of 12 seconds, the tool takes more than 10 minutes to generate a schedule that involves more than 10000 temporal windows. This activity without the tool is a intractable problem.			
Variability	Method for variability description in applications and platforms	Variability is central to the concept of "product-line", where designers can create variants of a product by simply enabling or disabling features. Understanding variability, that is the parts that vary or remains is necessary to further automate the derivation of related artefacts such design models, test plans, requirements or safety cases, to name a few.	The idea of variability analysis is to speed up the development process by maximising reuse among variants. Without variability analysis, each new variant has to be de developed and verified from scratch, which necessarily increases the cost and time to market.	No	Yes	No
	Variability analysis and testing techniques for mixed critical systems	Variability analysis and testing techniques aim at reducing the cost of verification and validation of variants. Instead of verifying each variants separately, it helps verify the product line as a whole, regardless of the specific variants of interest.	Overlooking variability testing techniques imply the separate verification of each variant. As variants share significant part of their architecture, this would repeat many times the verification of similar fragments of systems.	No	Yes	No
Safety Concept	Functional Safety Management (FSM)	Without following a functional safety management process the system will not be assessed by a certification authority.	It is impossible to shut it down; the creation of the system can be done with or without the tool. If it is done without the tool the system will not follow the required tools and design methodologies in order to be certified.	No	Yes	No
	Modular safety-case for hypervisor, multicore and mixed-criticality network solutions	Each modular safety case has been first assessed by the certification authority TÜV Rheinland and then, within the wind-power use case. It is expected that the use of modular safety-cases	It is impossible to shut it down; the creation of the system can be done by using the traditional safety process by defining the system as a whole or by using	No	Yes	No

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		increases the efficiency in terms of certification effort compare with the effort of assessing a series of full certifications. There will not be direct feedback from the demonstrator development. It will be done through the assessment of WP4 toolchain where they have been integrated as support documentation.	the modular concept improving the amount of work that has to be done in order to recertification the system if one of the sub-modules Is changed.			
	Tool supported definition of IE61508 based Safety Models for mixed-criticality product-line applications	The tools used in order to define the IEC61508 based safety models helps to improve the process of certification. Without this tool the certification process will be more complex and difficult, increasing costs and development time.	It is impossible to shut it down; the creation of the system can be done with or without the tool. If it is done without the tool the system will not follow the required tools and design methodologies in order to be certified.	No	Yes	No
	Tool supported IEC61508 based Safety Constraints Checker and Safety Rules Checker	The safety constraints checker and safety rules checker tools check in an automatic manner the compliment of safety rules and constraints in the system. Without these two tools the checking process should be done in a “traditional“ way increasing the effort and the possibilities of doing something wrong within the design process.	It is impossible to shut it down; the creation of the system can be done with or without the tool. If it is done without the tool the system will not follow the required tools and design methodologies in order to be certified.	No	Yes	No
	Tool supported Safety Consistency Report generator	The safety consistency report generator creates the documentation needed in a certification process asked by the certification authority. If we do not have this tool all the documentation required within the certification process should be created by hand.	It is impossible to shut it down; the creation of the system can be done with or without the tool. If it is done without the tool the evidences that must be presented to the certification authority should be created by hand increasing the difficulty and time needed to finish the project.	No	Yes	No

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
Virtual Platform	Gem5 model of STNoC with configurable network interface	The STNoC gem5 model can be used to study the characteristics of STNoC and doesn't serve any purpose, except for validation of the TT schedule and NoC properties.	The cycle-approximate gem5 model has been used in WP2 (e.g. deliverable D2.4.1) to demonstrate characteristics of STNoC and thus cannot be shut down during the final demonstration.	Yes	No	Yes
	Wireless TTE communication services	Since we are also targeting moving systems using cables looks rather strange (consider an electric vehicle with cable connection! Absolutely strange idea!)	Certainly you can shut WL connection down, than you don't have a connection at all. If this is no problem you can do it, but we doubt it severely that no control of the system will lead to any performance at all.	No	No	Yes
Chip-Level Virtualization Technologies	Bandwidth regulation policies (MemGuard) in Linux kernel and as kernel module and related network regulation policies (NetGuardXt) as kernel module. Notice that CPU bandwidth can be considered as part of the Linux kernel scheduling policy.	Memory and Network Bandwidth Regulation policies (MemGuard and NetguardXT) provide hard guarantees by reserving the aggregated guaranteed memory/ network bandwidth requested by available sources and provide efficient on-the-fly partitioning which is crucial for real-time multicore SoC. Without them, there will be no on-the-fly partitioning and there will be longer jitter for safety-relevant memory and network accesses.	As the implementation of those building blocks is performed at the kernel level, one can shut them down.	No	No	Yes
	Time-triggered and event-triggered on-chip extensions (LRS) [14]	The time-triggered on-chip extension (LRS) is essential for the time/space partitioning on the NoC. It ensures the independence between application subsystems, enables enabling modular certification, improves temporal predictability and fault-tolerance. Without the LRS, these properties would be lost.	Traffic shaping can be disabled (e.g., configuration of rate-constrained communication with MINT=0), thus demonstrating the missing temporal/spatial partitioning in case of core failures.	Yes	Yes	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
	Macro-architecture of on-/off-chip bridge (bus-to-NoC) via PCIexpress	You can always find a weaker solution that will not perform sufficiently. We targeted a solution that is viable.	This would lead to data loss, does not look to be recommended.	Yes	No	No
	Memory interleaving extensions at the network interface layer of the Spidergon STNoC	Memory interleaving extensions at the network interface layer allows parallel access to the DRAM device, increasing the theoretical amount of memory bandwidth by balancing the network load among different channels through memory interleaving techniques.	As this technology is performed at the initiator network interface (RTL implementation), there is no possibility of shutting down.	No	No	No
	Real-time scheduling heuristics and coordination for KVM supporting low execution time and low frequency	Linux KVM has been extended to include coordination mechanism between the virtual machines and the host system. Three schedules are affected by these changes, the task scheduling, the I/O scheduling and the memory bandwidth management (based on Memguard). It improves soft real-time behavior of applications inside virtual machines. For the task and I/O scheduling, it reduces the latency and the execution time, while the memory bandwidth management can guarantee minimal bandwidth. Without these technologies, the real-time applications running inside virtual machines are not guaranteed to have the same real-time behavior as if they were running in the host system.	The memory bandwidth management and the I/O scheduling coordination can be disabled “on the fly” by the virtual machines themselves, while the task coordination scheduling is an offline configuration that need to be enabled in the kernel of the host and the virtual machines. For instance, disabling those mechanisms will show a bigger variance of the latency of real time applications running inside virtual machines.	No	No	Yes
	Security services for on-chip communication, in particular hardware security mechanisms at the	Security-services for on-chip communication provide confidentiality and integrity during ECG transmission, and support for patient anonymization via a hardware NoC Firewall on Zedboard (ARMv7).	As security services for on-chip network interface is implemented by in the hardware, there will be no possibility of shutting it down.	No	No	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
	NoC network interface layer					
	XtratuM for ARM	The hypervisor provides time and space partitioning at processor execution level. User applications with different criticalities can be executed on the same platform in an isolated way in virtual machines or partitions. Without the hypervisor, faults or wrong execution in some partition (e.g. non-critical applications) could have a severe impact in the whole system. Additionally, the hypervisor allows to use different O.S., with different programming language with different resources allocation based on the type of application.	End-user applications running inside partitions delegate the multicore scheduling to the scheduler offered by the hypervisor. Additionally, the communication among partitions is also based on communication services provided by XtratuM. Therefore, the possibility of shutting down the hypervisor have a strong impact in the functional behavior of the applications and it could not be considered.	Yes	Yes	Yes
Chip-Level Virtualization Technologies	XtratuM for x86	The hypervisor provides time and space partitioning at processor execution level. User applications with different criticalities can be executed on the same platform in an isolated way in virtual machines or partitions. Without the hypervisor, faults or wrong execution in some partition (e.g. non-critical applications) could have a severe impact in the whole system. Additionally, the hypervisor allows to use different O.S., with different programming language with different resources allocation based on type of application.	Multicore resources management is delegate to the hypervisor. Which execute two different O.S. with different criticality. Therefore, the possibility of shutting down the hypervisor would not allow a parallel execution of the O.S. and it could not be considered.	No	Yes	No
	XtratuM for Power PC multicore architectures	The hypervisor provides time and space partitioning at processor execution level. User applications with different criticalities can be executed on the same platform in an isolated	Multicore resources management is delegate to the hypervisor. End-user applications depend on the multicore scheduler offered by the hypervisor.	Yes	No	No

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		way in virtual machines or partitions. Without the hypervisor, faults or wrong execution in some partition (e.g. non-critical applications) could have a severe impact in the whole system. Additionally, the hypervisor allows to use different O.S., with different programming language with different resources allocation based on the type of application.	Additionally, the communication among partitions is also based on communication services provided by XtratuM. Therefore, the possibility of shutting down the hypervisor have a strong impact in the functional behavior of the applications and it could not be considered.			
	Integration of STNoC in the hypervisor XtratuM	This integration offers a transparent way to perform the communications among the end-user applications. It is due to the hypervisor provides a common and well defined interface to inter-partition, inter-tile and inter-node communications. Hypervisor provides a separation between the implementation of the application and the configuration of the communications. The application uses a common interface to send and receive messages, but the application is not aware about type of communication used. Which allows to perform reconfiguration in the communications and reallocation of partitions in the system without needs to modify the application software.	Multicore resources management and software STNoC driver implementation are delegate to the hypervisor. End-user applications use the interfaces provided by the hypervisor. Therefore, the deactivation of this integration would require modifying the application implementation, embedding the STNoC driver in each partition and considering multicore issues to use the device.	Yes	Yes	Yes
	Firmware monitor layer	The firmware monitor layer enables, on an ARMv8-A platform, the native concurrent execution of two operating systems, such as for example a safety critical RTOS and a GPOS with the option to use virtualization extensions (e.g., Linux-KVM) in order to instantiate a variety of	For demonstration purpose, it can be possible to implement a test case where the OS scheduling and isolation ensured by the secure monitor firmware (VOSYSmonitor) will be disabled, thus	No	No	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		different VMs. It ensures the isolation of each world by using the hardware security extension called TrustZone and provides, at the same time, functions to enable a safe and secure communication between them. Therefore, the critical tasks, running in the Secure world, are totally isolated from applications executing in the Normal world. Without the firmware monitor layer, the consolidation of systems with different levels of criticality on a common ARMv8-A hardware platform, while ensuring the isolation of the safety critical tasks, would not be possible.	demonstrating that the native co-execution of two Operating Systems is lost.			
	Integration of TTEthernet in the hypervisor XtratuM	This integration offers a transparent way to perform the communications among the end-user applications. It is due to the hypervisor provides a common and well defined interface to inter-partition, inter-tile and inter-node communications. Hypervisor provides a separation between the implementation of the application and the configuration of the communications. The application uses a common interface to send and receive messages, but the application is not aware about type of communication used. Which allows to perform reconfiguration in the communications and reallocation of partitions in the system without needs to modify the application software.	Multicore resources management and software TTEthernet driver implementation are delegate to the hypervisor. End-user applications use the interfaces provided by the hypervisor. Therefore, the deactivation of this integration would require modifying the application implementation, embedding the TTEthernet driver in each partition and considering multicore issues to use the device.	Yes	No	Yes

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
Cluster Level Virtualization	On-chip/off-chip gateways (different protocols), e.g. on-chip/off-chip gateway between Spidergon STNoC and TTEthernet	You can always find a weaker solution that will not perform sufficiently. We targeted a solution that is viable.	One will not be able to use different protocols. Reduced functionality not fulfilling requirements will be the result.	Yes	No	Yes
	Wireless TTE gateway	No TTE traffic via wireless possible and thus significantly reduced function violating the requirements.	This would lead to data loss, does not look to be recommended.	No	No	No
	Secure communication in TTEthernet	No secure communication means that everybody can interfere and hack into the data stream and disturb or miss-use the system as he likes w/o any means to protect the system. As an alternative, one could try to solve that on application level but this would take significantly higher efforts to achieve the same. In addition every new system would have to “develop the same thing over and over again” rather than making use of a proven design.	If you shut down the communication data exchange and command execution is interrupted and the system would fail completely.	Yes	No	Yes
	Safety Communication Layer (SCL)	The SCL will be used in the wind power demonstrator to transport safety related input/output data between EtherCAT slaves and safety protection system deployed in the harmonized platform. It implements a set of measures and diagnostic techniques for communication errors and implements system reactions to errors. For example, in the case that an error is detected, the SCL will activate a safe-state until a manual reset of the system is performed. Without the SCL, communication errors could lead the wind power system to an	SCL implementation resides in application software running in DHP cores and EtherCAT node. It cannot be shut down and activated again by means of a switch. However, if application software files are modified, SCL feature could be deactivated. This process requires to download new binaries to DHP cores and EtherCAT node.	No	Yes	No

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
		unsafe-state. Data would arrive in raw format to DHP cores and no communication errors or data integrity would be checked.				
	Data-acquisition system for capturing and storing EtherCAT process	The Datalogger is part of the workbench to test the SCL. It is a data-acquisition system that analyses the safety frames interchanged among the EtherCAT slaves and the master.	As it is not part of the final Wind Power system, it can be switched off and put apart.	No	Yes	No
Validation and Verification Techniques	Verification framework of the components on the network level (i.e., establishment of a framework for the transfer of test case results from the simulation to tests performed on the target hardware)	You can always do things w/o tools too, but one would need an army of employees to do it!	Since the framework is used “off-line” shutting down is not applicable in this context	Yes	No	No
	Fault-injection framework for EtherCAT industrial communication protocol	The Real-Time Fault-Injector is an FPGA based system that is part of the workbench to test the SCL. It injects faults in a communication between two devices by placing it between them.	As it is not part of the final Wind Power system, it can be switched off and put apart. Besides, in idle operation mode when not fault is required to be injected, the Faults Injector behavior is transparent for the communication, that is, the frames received are automatically forwarded.	No	Yes	No
	Secure resource management to verify authenticity, integrity and confidentiality of the resource management components, generic	The security services on application layer ensure that an attacker cannot perform attacks such as sniffing, spoofing, man-in-the-middle, packet injection or replay attacks. Performing those attacks, an attacker can manipulate the system, e.g. sending false configuration messages. This manipulation can	The communication channels between the resource management components (and between the applications in WP6) are preconfigured during development to use the security service. During runtime, it is not possible to change the security level.	Yes	No	No

Cate-gory	DREAMS Result	Effect of not having …	Possibility of Shutting down	WP6	WP7	WP8
	security services for execution environment	lead to a system failure, e.g., a node is shut down because of the invalid configuration message.	Shut down the service or fallback on insecure communication is not a reasonable option as an attacker might use this situation to enforce a system failure.			
	LRM: Deadline overrun management service	Critical applications could miss their deadline when executing on a multi-core with non-critical applications. So typically we would increase the critical application WCET (e.g. 2x in a dual-core), which could hinder the schedulability of the application and if schedulable would make an inefficient use of the core.	It can be deactivated at configuration (YAML/DLRM configuration file), but not much sense of deactivating it. Actually what should be done is a different configuration with critical applications with an augmented WCET.	Yes	No	No
	Local resource monitoring services including fault detection (MON) and fault isolation services at chip level	Local resource monitoring (LRM) services including fault detection (MON) is essential for the local and global reconfigurations. Without MON, the failures cannot be detected and without the LRM, recovery procedures and GRM requests cannot be applied.	LRM and MON can be disabled. In this case no local and global configuration will be provided in case of core failures or deadline overrun.	Yes	No	Yes
	LRM: QoS service (extension on deadline overrun management)	Inefficient use of computing resources (cores) by non-critical applications on deadline overrun management actions.	It can be deactivated at compilation. Could be deactivated at configuration (YAML/DLRM configuration file), but support is not there currently.	Yes	No	No
	Recovery Strategies	In this case no local and global configuration will be provided in case of core failures or deadline overrun.	It can be disabled.	Yes	No	Yes
	Reconfiguration services: implementation of local reconfigurations and global decisions	In this case no local and global configuration will be provided in case of core failures or deadline overrun.	It can be disabled.	Yes	No	Yes

Table 7 Expected effect and possibility of shutting down the DREAMS technologies

6 Demonstrator Support

This section summarizes the demonstrator support activities in WP1-5.

6.1 WP1

The activities in WP1 separate into support for the DREAMS Metamodels (see Section 6.1.1) and the DREAMS Harmonized Platform (see Section 6.1.2).

6.1.1 DREAMS Metamodels

6.1.1.1 Introduction

The DREAMS metamodels provide a number of viewpoints that allow creating formal descriptions of mixed-criticality systems. They have been derived from the DREAMS architectural style, and serve as pivot to integrate the different tools of the DREAMS toolchain. The project developed models for each of the application demonstrators investigated in WP6-WP8, along with several smaller models used to verify tool integration and for demonstration. The MCS viewpoints defined in DREAMS provide metamodels to describe the logical architecture of applications, the virtual and physical platform, deployments [5], as well as resource allocations, reconfiguration and the platform-specific configurations [6]. WP1 developed in T1.4 and T1.6 an implementation of the DREAMS metamodels that is based on the Eclipse Modeling Framework (EMF)[5], and the model-based AutoFOCUS3[6] CASE tool for embedded systems. The metamodels have been integrated into a dedicated release of the tool, *AutoFOCUS3/DREAMS edition,* which also serves as an integration platform for the tools and configuration generators developed in T4.1-T4.3.

6.1.1.2 Support Channels

Support requests (questions to clarify the metamodel, error reports, change requests) from consortium partners have primarily been received via e-mail. When required, dedicated telephone conferences were arranged to discuss specific issues. Furthermore, in the context of physical DREAMS meetings (e.g., meetings in Barcelona in 2016 and the Munich meeting in 2017), WP1 discussed modeling issues, assisted partners in creating models, and supported tool-providers to integrate their results with the DREAMS metamodel.

On the one hand, feedback was directly from the demonstrator partners, and referred to questions or issues that came up during the preparation of the demonstrator models. On the other hand, demonstrator feedback was received indirectly from tool providers who approached WP1 in order to request changes in the metamodel or the *AutoFOCUS3/DREAMS edition* Eclipse RCP application in order to address issues in the integration of their tools discovered when applying the DREAMS toolchain to one of the demonstrator models.

6.1.1.3 Metamodel Updates

The following modifications to the platform metamodel have been requested:

- A `MediaType` annotation for off-chip networks has been added.
- Minor parameter adjustments of the virtual platform (part of the DREAMS platform metamodel) to enable a correct and consistent configuration generation for *XtratuM* hypervisors.
- A bug in the linkage of `MemoryArea`s and their respective `MemoryUnit`s was fixed.

[5] http://www.eclipse.org/modeling/emf/

[6] http://af3.fortiss.org/

- In addition, the composition of some platform model elements has been fixed that enforced that the resulting models obeyed the DREAMS architectural style. This, affected models of the off-chip networks, watchdogs, clocks, memories, and health monitor requirements.

Furthermore, WP1 applied the following changes to the platform-specific metamodel:

- The model elements associated with the `OffChipNetworkInterface`s were adjusted such that the reconfiguration at this level is correctly aligned with the representation of the reconfiguration at the `OffChipNetwork` device level.
- The platform-specific model was enriched with some additional attributes required to correctly generate configuration for the *XtratuM* hypervisor.
- Blackout zones for task schedules and bypass windows for the on-chip network schedule were added to the schedule model.

6.1.1.4 Modelling Infrastructure Improvements and Support Services

The work on the metamodel and the creation of the final demonstrator models resulted into the following improvements of the modelling infrastructure:

- FORTISS and RTAW extended the Java utility library with several additional methods that ease the handling of models (e.g., timing model, configuration models, schedule model, reconfiguration model).
- The work on the final demonstrator models resulted also into usability improvements of the model editors, such as the integration of an automatic layouter for diagrams, an improved editor for deployment-specific parameters, a generator for logical components representing resource management (RM) entities, as well as timing model generators).
- WP1 integrated the feedback from the demonstrator partners and tool providers into *AutoFOCUS3/DREAMS edition* that was distributed among the consortium partners as binary releases via a project-internal download site, including an Eclipse update site for convenient updates of existing installations. During the runtime of the project, WP1 distributed 53 builds to partners, from which roughly 20 builds fall upon the reporting period of this deliverable. In addition to the metamodel updates discussed in the next section, these builds included an update of BVR, support to install the *TTEthernet* Eclipse tooling into AutoFOCUS3/DREAMS edition, as well as updates of tools, model-transformations, and configuration generators provided by WP4.

6.1.1.5 Demonstrator Modelling Support

WP1 supported WP6-WP8 partners in preparing the final models of the application demonstrators onto which they could then apply the DREAMS toolchain.

For the avionics demonstrator (WP6), FORTISS has provided an initial version already for the intermediate integration milestone [10], which consisted of a logical application architecture and a platform architecture. WP1 supported TRT to prepare the final version of the demonstrator model based on this template that accounts for changes in the application specification, and that was extended with additional parameters and aspects. In particular, this included models to capture the RM infrastructure (using the RM component generator mentioned above) and its configuration in terms of reconfiguration graphs and schedules (relying on the aforementioned PSM changes and utility library enhancements). TRT updated the model with realistic WCETs of logical components, which resulted into the creation of the improved deployment-specific parameters editor (required due to the size of the model). In the platform model, WP1 supported modifications to match it with the final demonstrator setup. The most noticeable change is that all twelve cores of a *T4240* processor are now contained in a single Tile (i.e., the T4240's three core "clusters" consisting of four cores each are not represented in this model), since it is under the regime of a single *XtratuM* instance.

Furthermore, WP1 provided assistance in modeling the ROSACE application [15], e.g., it supported modelling RM components. The model was created to obtain a model from the avionics domain that could be published in the public second WP4 *Integration and Support Report* [16], and that has also been used in the context of the DREAMS summer school. WP1 also handled change requests on the metamodel as a result from the application of the reconfiguration tools onto the ROSACE model.

Finally, RTAW created a model of the WP8 healthcare demonstrator that enables to apply the DREAMS toolchain to generate *TTEthernet* configuration.

Two conference publications on the DREAMS development methodology, metamodel and toolchain document the demonstrator models that have been created with the help of this support task:

- [17] presents the model of the WP7 wind turbine to illustrate product-line and design space exploration.
- [18] provides a high-level and partly anonymized view on the WP6 avionics model to discuss how the DREAMS toolchain supports resource allocation and reconfiguration.

6.1.2 DREAMS Harmonized Platform

WP1 has been supporting the demonstrators since the early phase of the project by providing the DREAMS Harmonized Platform (DHP) and integrating HW developments from ST (STNoC for mixed criticality context) USIEGEN (LRS at on-chip NI to control the traffic injection according to the traffic classes) and TTT (TTE IP for the on-chip/off-chip gateway) was tailored to be used in each of the 3 demonstrators.

During the last year of the project, WP1 supported the demonstrators by providing the new version of the DHP, in which the new version of the TTE IP was used.

6.2 WP2

In the latest period, WP2 mainly supported the demonstrators in guaranteeing that the use cases selected will be compliant with the HW/SW developed in DREAMS WP2.

ALSTOM, IKL and FENTISS had close looping interactions to validate the usage of resource monitoring and allocation algorithms within XtratuM hypervisor and porting of OSs on top of hypervisor for X86 and ARM platforms.

Moving to even more cores in PowerPC architecture for THE Avionics demonstrator, FENTISS, ONERA and Thales worked closely together to integrate XtratuM with the resource monitoring and allocation services developed by ONERA in the framework of the Thales avionics demonstrator. These partners ruled their own collaboration plans, sharing code development and extensively sharing specifications and feedbacks.

Moving to even more cores in PowerPC architecture for Alstom windpower demonstrator, ONERA and FentISS had close looping interactions to validate the usage of resource monitoring and allocation algorithms within XtratuM.

ST, TEI and VOSYS have been jointly collaborating on the definition of the use case for the Healthcare demonstrator, trying to reach the best tradeoff in functionality demonstrated vs potentialities offered by the ARM v7 and ARM v8 platforms. Several use case scenarios where elaborated by WP2/WP8 partners to make sure that the demonstrator will reflect the development in both WP.

In the end, all technologies developed in WP2 are showcased in one of the demonstrator, which is in itself a great achievement for the DREAMS project.

6.3 WP3

This section summarizes the demonstrator support for the cluster-level communication services (6.3.1), the resource management services (6.3.2), the security services (6.3.3) and the safety communication layer (6.3.4).

6.3.1 Cluster-level communication services

Technologies developed in the WP3 T3.1 mainly consist of the extension of the building blocks of the virtualization technologies, network components and middleware necessary for the implementation of distributed networking services that support the mixed-criticality requirements from the DREAMS architectural concept. These building blocks range from the actual communication subsystem (i.e. switches, end-systems, firmware, drivers and configuration) as well as the mechanisms for global reconfiguration and end-to-end security.

In the final integration, the demonstrators were supported to integrate the communication building blocks correctly into the respective layers. WP6 and WP8 were supported to integrate TTE into the hypervisors XtratuM for the avionics demonstrator and KVM for healthcare demonstrator. Practical support on the usage of the configuration tool chain to generate working and correct configuration for the gateways for the respective scenarios was also given. Additionally, training courses were provided to help them using the building blocks in the correct manner.

6.3.2 Resource Management Services

The Global Resource Manager (GRM) is defined and developed in T3.2, and integrated with Avionics demonstrator of Work package WP6.

The GRM is implemented as part of the DREAMS Resource Management (DRM) library. Functionality for secure communication between GRM and LRM is also incorporated in this library. The DRM library has been integrated with the Avionics demonstrator. The GRM runs as a critical application in its own XtratuM partition on the DHP. With respect to Avionics demonstrator, GRM provides the following main services:

- Gather status from LRM: GRM receives regular updates from the LRM via update messages informing the GRM about the current configuration of the LRM, and failed cores on the corresponding nodes.
- Obtain configuration: The GRM is in charge of the database of all off-line pre-computed configurations.
- Global reconfiguration decision: The GRM will analyze the updates sent by the LRMs, and take reconfiguration decisions that allow the system to adapt to different faults. The GRM takes into account updates from all LRMs to make a decision.
- Send orders to LRM: Once a reconfiguration decision has been taken, the GRM will communicate it to the LRMs involved in the reconfiguration, via order message using network and middleware. For example, if a change in the scheduling plans of an application tile is required, the GRM will provide a new mode for the corresponding LRMs, as well as for the network interfaces of the application tiles involved, because reconfiguration of the network is expected.
- Synchronize start of applications on different nodes via LRM and LRS

Some additional results of GRM are as follows:

1. Bounded Reconfiguration time

For the reconfiguration time to be bounded, the MaCs on different nodes should be synchronized and be of same length. The reconfiguration time after a reconfiguration request is received by the GRM is

a. One MaC, if the GRM is scheduled atleast
    i. after a period of one TT VL for update channel after the LRM in previous MaC and
    ii. before a period of one TT VL for orders channel before the LRM in the current MaC.

Note: In case of Avionics demonstrator, the constraint 1a (i) is disregarded as the LRM is extended by a much larger time to print debug data from apps and other RM components at the end of its execution.

b. Two MaC, if the constraints mentioned in (1a) for scheduling the GRM are not met.

2. Predictable Reconfiguration results

Global reconfigurations may occur only in case of permanent core failure on a node. GRM is informed about the current configuration of a node and whether a global reconfiguration is required by the node each MaC by the corresponding LRM. From this information, it applies global reconfiguration changes when required and if possible. This entails for local reconfiguration graphs to be complete, i.e. from any configuration, the GRM must be able to deduce all failed cores (this constraint must be taken into account by the off-line scheduling tools). The new configurations are selected from a Global reconfiguration graph which consists offline pre-computed configurations (XtratuM scheduling plans). Hence, the result of reconfiguration is predictable

In addition, the reconfiguration strategy follows two rules in case not all applications could be locally hosted after some failure(s) on a node: critical applications are locally reconfigured in priority and complete applications must be moved, i.e. an application cannot run on two nodes at the same time.

3. Behavior in presence of GRM faults

It is assumed that the core on which the GRM executes is fail-safe. GRM only makes global reconfiguration decisions when necessary, but it is not required for the continuous operation of the system. Thus, in case of GRM failure, the overall system dependability is not compromised as the system will still keep on executing; just no new global reconfigurations will be possible. Thus, this failure is not considered)

4. Timely communication with the LRM

The GRM must be informed via an update message of the node's current configuration, as this information is used by GRM to select new global configuration if required. The GRM sends new global configuration to the LRM via an order message.

All master LRMs must send one update message to the GRM every MaC. The GRM will consider the node of the corresponding LRM dead (i.e., all cores have failed) in case of failure to receive an update message in a MaC.

To ensure that the update and order messages are delivered in a timely manner, the communication between LRM and GRM takes place using Time-Triggered messages. Orders messages are sent using a Multicast Time-Triggered Virtual link (TT VL) from the GRM to LRMs on each node. There are as many Multicast TT VL as there are nodes. In case, of avionics demonstrator there will be 2 multicast TT VL (We don’t need TT VLs for LRMs on the same node (DHP) as the GRM. Only XtratuM communication channels are used).

Updates messages are sent via a point-to-point TT VL. There is an update channel from LRM on every core of a node to the GRM. In case of Avionics demonstrator, there are 24 TT VL if all cores of T4240 are used (We do not need TT VLs for LRM on the same node (DHP) as the GRM Only XtratuM communication channels are used).Table 8 summarizes the RM communication channels.

Communication channel	Source	Destination	XM Channel Type	TTE Type
Orders	GRM	LRM	Sampling	Multicast TT VL
Updates	LRM	GRM	Queuing	TT VL

Table 8 Resource management communication

6.3.3 Security Services

The security services in DREAMS are classified into cluster-level and application-level security services. The cluster-level security services provide the secure off-chip communication and the secure time synchronization.

During the integration, WP6 and WP8 received support for the cluster-level security services and WP6 received support for the application-level security services.

For the cluster level, MACsec has been integrated to provide the secure communication between the nodes and to achieve the secure time synchronization. These services are used by WP6 and WP8. The support for the demonstrator was provided in conjunction with the cluster-level communication services as MACsec extends TTE on layer 2.

For the application level, the security library has been developed in T3.3. It provides security services on cluster level to provide confidentiality, integrity and authenticity. A detailed description of the security library is given in [15], [16] and [17]. The security library is located on top of the hypervisor and is used by the applications for a secure communication. In addition, the core service resource management uses the security library. The avionics demonstrator in WP8 uses the security library for all applications. Appropriate security levels had to be selected for the different channels between the applications.

6.3.4 Safety Communication Layer (SCL)

The Safety Communication Layer has been developed within T3.3. This layer isolates the end application from the communication channel faults and implements the safety measures (techniques) defined by IEC 61784-3-3 to avoid the communication errors such as transmission errors, repetitions, deletion, delay, etc. (further detailed in deliverable D3.3.1 section 7.1.1.1).

The SCL has been implemented over the EtherCAT-DHP channel within the wind power demonstrator in order to guarantee data integrity in the data exchange between the EtherCAT node and the DHP. SCL logic makes sure the received frame is a valid frame. If these values exceed the threshold or the frame is not valid a safety line is opened.

WP7 has received support during the integration of the SCL API (application Programming Interface) in the EtherCAT node and in the DHP and during the assessment phase. As the mechanism for transmission of safety telegrams is Host (Master) - Device (Slave) based, the “F-Host” has been implemented in the node and the “F-Devices”, in the DHP. In both cases support has been given in the integration of the API High (App to SCL) and the API Low (SCL to Medium) being EtherCAT the medium for the node and the STNoC for the DHP.

The SCL has been stressed in the evaluation plan by means of the real time fault injection framework, an FPGA-based system developed within WP5 that is able to inject faults in a communication between two devices by placing it between them. In order to monitor the exchanged safety frames and be able to track when and where an unexpected behavior takes place a Data logger tool has been developed within WP3. It is part of the support tools and allows users to store network frames and extract the variables that travel within them.

6.4 WP4

6.4.1 Introduction

WP4 has defined a set of algorithms for the transformation steps of the model driven development process defined by the DREAMS project. These algorithms are implemented into different kinds of tools: Model Editors, Design Tools, Verification Tools, and Configuration File Generators. The DREAMS meta-model defined in WP1 is the backbone of the toolchain in the sense that every tool either directly reads or modifies the common DREAMS system model or indirectly, with the help of tool connectors (importers and exporters).

6.4.2 Demonstrator Support

In order to help the demonstrator work package in applying WP4 algorithms and tools, three tool chain use cases have been identified and described in details in D4.4.1[4], together with an inventory of the main functionalities of each tool. Furthermore, help has been provided to the demonstrator work package in identifying the appropriate tool chain use cases and in the concrete usage of the different tools in the numerous steps of the use case.

The application of the tool to the wind power demonstrator is documented in D4.3.2 [12], D4.3.3 [13], and D4.4.2 [15]. The focus was on variability and safety.

The application to an avionics example is described in the public deliverable D4.4.2 [15]. The application to the confidential avionics demonstrator is described in D6.3.2. The focus of the use case for the application of tools was offline adaptation strategies.

The tool chain has also been applied to the configuration of the mixed-critical off-chip communication of the health-care demonstrator as described in D4.4.2 [15].

6.4.3 Adaptations to account for feedback

The concrete application of the tools to the demonstrators has led to the identification of shortcomings or inabilities of (configuration algorithms) to cope with the actual systems, as well as other feedbacks on the usability of the tools. In order to take into account the feedback and the learnt lessons, not only tool functionalities have been adapted or extended but also some algorithm or configuration strategies had to be reworked, impacting also the associated tool connectors or configuration file generators.

6.5 WP5

The goal of this work package is to pave the way towards a competitive development and certification of mixed-criticality solutions. Competitive development and certification emphasizes the need for solutions to manage complexity, increase re-usability, reduce product cost and reduce product overall certification cost and time. For this purpose, modular safety-cases, patterns, tool integration, test beds and guidelines are provided based on the architectural style, virtualization, multicore and mixed-criticality network contributions already provided in other WPs. Taking into consideration these results and contributions, demonstrator support includes not only help in using the provided tools but also support on the identification of opportunities to apply them. The following sections summarize the integration of the solutions and techniques implemented during the development of WP5 into the Avionics, Wind Power and Healthcare demonstrators of the European project DREAMS. IEC-61508 is considered to be the reference safety standard for WP5 and, WP5 results are mainly considered in WP7 wind power demonstrator where all the results have been applied. An extended description of the support of WP5 to the development of demonstrators can be found on deliverable D5.6.1. The evaluated and supported results regarding WP5 are the modular safety cases, cross-domain patterns,the safety communication layer SCL, the EtherCAT data logger and the virtual platform.

6.5.1 Modular Safety Cases

A modular safety case describes arguments to demonstrate that safety properties are satisfied and risk has been mitigated. Deliverables D5.1.1, D5.1.2 and D5.1.3 define the Modular Safety Cases (MSCs) for an IEC 61508 compliant generic hypervisor, safety partition, COTS multi-core device and mixed-criticality network. Those MSCs define the safety requirements that a safety hypervisor, partition, COTS multi-core device and mixed-criticality network shall fulfill to be compliant with the IEC 61508 safety standard. In addition, those deliverables provide linking analyses that define the way in which a commercial hypervisor (XtratuM), a COTS multi-core device (Zynq-7000 ZC706) and a mixed-criticality network (TTE and EtherCAT) fulfill the requirements stated in the MSCs. These components are integrated into the wind turbine case study for composing the protection system. The integration of the defined MSCs, using the safety lifecycle, into the wind turbine case study has been done by using the Goal Structuring Notation (GSN) language and defining abstraction layers. The details of integration can be found on deliverable D5.6.1.

The MSCs may apply to the avionics demonstrator, always extending the measures and diagnostic techniques defined to accomplish the requirements of the DO-178C avionic standard. The avionics use case is not analysed from a product line perspective due to time limitations. However, the product line approach presented in deliverable D5.6.1 (Subsection 2.6) may be extended to the DO-178C standard, thus covering the avionics use case.

In the case of the healthcare demonstrator, the MSCs defined in DREAMS cannot be applied, in a first instance, due to they are defined for an IEC 61508 compliant system and they are not compliant with healthcare-related standards. However, the presented product line approach may be extended to include healthcare-related standards.

6.5.2 Cross Domain Patterns

The cross-domain patterns provide reusable generic solutions and diagnostic techniques for mixed-criticality systems, and they are applicable in the ascending branch of the development process defined in the safety lifecycle.

Regarding the wind power demonstrator, Figure 2 shows the integration of the cross-domain patterns in the safety argumentation hierarchy at component a graphical representation that illustrates the integration of the cross-domain pattern as extra measures and diagnostic techniques which may be implemented for developing a component of a product sample.

The cross-domain patterns defined this research project can be applied to the avionics use case, such as those patterns extend the measures and diagnostic techniques recommended by IEC 61508, and they can be applied to any mixed-criticality system. However, the applicability of those patterns is theoretical, no pattern has been applied to the avionics demonstrator. For that purpose, the cross-domain pattern should be extended to cover avionics standards-related requirements.

6.5.3 Safety Lifecycle and Tools

This subsection deals with the integration of the tools defined in deliverable D5.6.1 (Section 2.4), the life-cycle development process defined in deliverable D5.6.1 (Section 2.3) and the demonstrators. The implemented tools include model edition, design, verification, platform configuration file generator and fault-injection tools, which are mapped to the phases of the development process. The integration between tools and each demonstrator lifecycle is based on the mapping of the tools defined in the deliverable D5.4.1.

The model editor is used to describe the applications, the technical architecture and the constraints. Then design tools are used, where possible, to automatically create the parts of the system configuration. These parts may be manually completed, where necessary, with the help of the model editor. Verification tools allow checking the correctness of the automatically or manually created

configurations. If a design tool produces a configuration that is correct by design, then verification tools check the configuration. Furthermore, the configuration file generators enable translating the verified system configuration automatically into platform configuration files, without errors that would be introduced by manual translation. These tools have been integrated in the development branch of each demonstrator defined lifecycle.

The tools applied in the windpower, avionics and healthcare demonstrator cover the development branch of the safety (when applied) lifecycle.

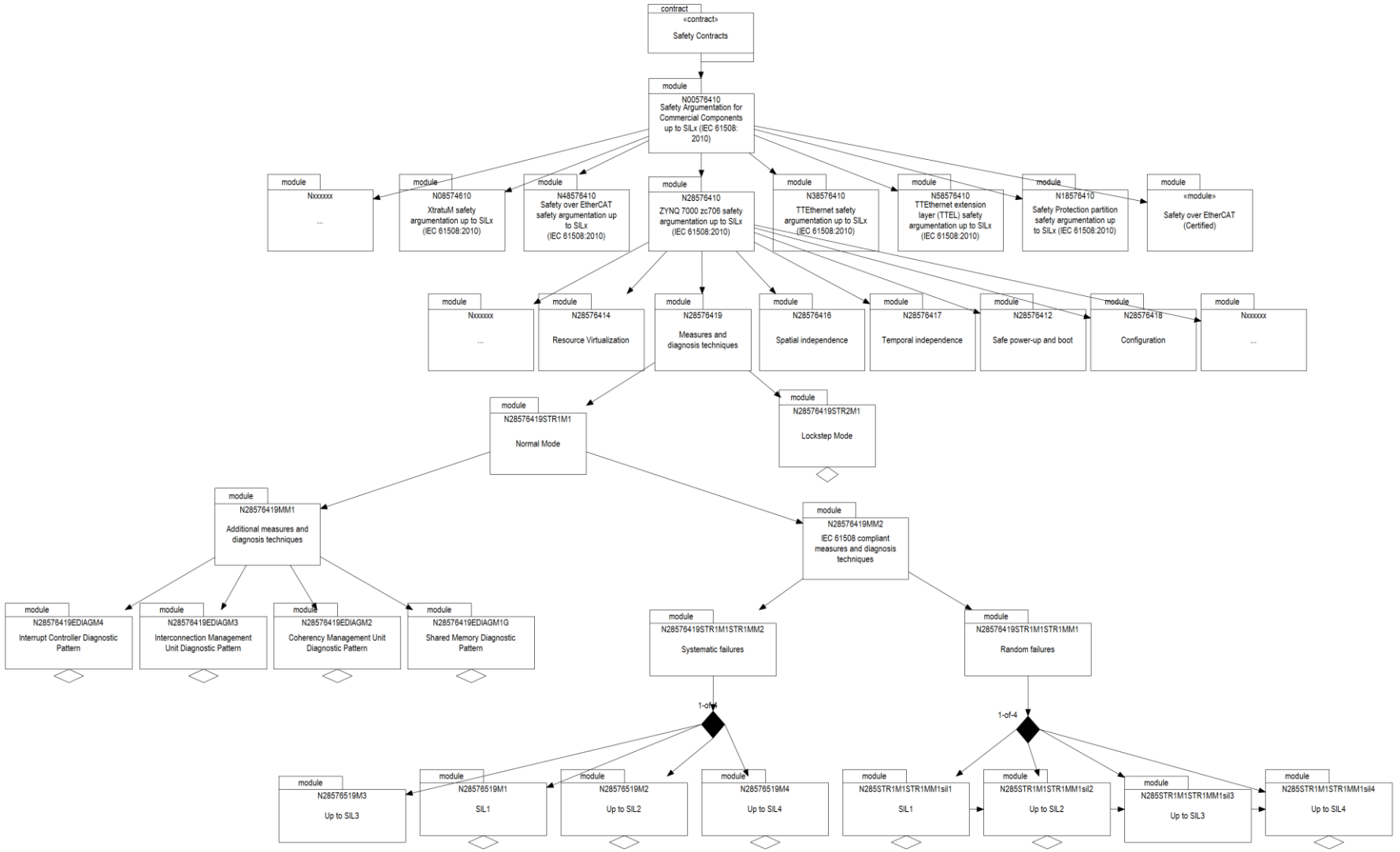

Figure 2: Wind Power Product Line and Cross-Domain Patterns.

6.5.4 Simulation, verification and fault-injection framework

The simulation, verification and fault injection framework is supporting supported on all the demonstrators. These tools have been used to simulate the building blocks of the DREAMS architecture for the cluster and chip levels. Simulation tools have supported chip level protocol verification and the investigation of safety properties, assuring that the timing performance of chip level building blocks supports the required behavior of the demonstrators. The fault injection components, for the injection of operational faults and design faults in the simulation components, allow checking if the safety function of the corresponding demonstrator is working properly.

7 Bibliography

[1] DREAMS Consortium, "D1.2.1: Architectural style of DREAMS".

[2] D. Consortium, *Description of Work,* 2013.

[3] DREAMS Consortium, "D1.3.1: Description of development process with model transformations".

[4] DREAMS Consortium, "D4.4.1: Tools feature map and interoperability capabilities".

[5] DREAMS Consortium, "D1.4.1: Meta-models for application and platform".

[6] DREAMS Consortium, "D1.6.1: Meta-models for platform-specific modelling".

[7] D. Consortium, "D6.3.2: Avionics assessment report," 2017.

[8] D. Consortium, "D7.3.2: Wind power assessment report," 2017.

[9] D. Consortium, "D8.3.2: Health-careassessment report," 2017.

[10] DREAMS Consortium, "D1.5.1: Intermediate integration of DREAMS platform with virtual platform prototype".

[11] S. Barner, A. Diewald, J. Migge, A. Syed, G. Fohler, M. Faugère and D. Gracia Pérez, "DREAMS Toolchain: Model-Driven Engineering of Mixed-Criticality Systems," in *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS '17)*, 2017.

[12] DREAMS Consortium, "D4.3.1: Variability analysis and testing techniques for mixed critical systems".

[13] DREAMS Consortium, "D4.3.2: First implementation and improvement of variability analysis and testing techniques for mixed critical systems".

[14] DREAMS Consortium, "D4.3.3: Final implementation and improvement of variability analysis and testing techniques for mixed critical systems".

[15] S. Barner, A. Diewald, F. Eizaguirre, A. Vasilevskiy and F. Chauvel, "Building Product-lines of Mixed-Criticality Systems," in *Proceedings of the Forum on Specification and Design Languages (FDL 2016)*, 2016.

[16] DREAMS Consortium, "D4.4.2: Integration and Support Report".

[17] DREAMS Consortium, "D2.2.2: Report on monitoring, local resource scheduling and reconfiguration services for mixed-criticality and security with implementation (source code) of low- and high-level monitors, scheduling, security and reconfiguration services supporting MC".

[18] DREAMS Consortium, "D2.3.4: Hypervisor adaptation and drivers for local resource manager".

[19] H. Ahmadian, R. Obermaisser and M. Abuteir, "Time-Triggered and Rate-Constrained On-Chip Communication in Mixed-Criticality Systems," in *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, Lyon, France, 2016.

[20] C. Pagetti, D. Saussié, R. Gratia, E. Noulard and P. Siron, "The ROSACE case study: From Simulink specification to multi/many-core execution," in *Proceedings of the 19th Real-Time and Embedded Technology and Applications Symposium (RTAS '14)*, 2014.

[21] DREAMS Consortium, "D3.3.1: High-level design of cluster-level safety and security services".

[22] DREAMS Consortium, "D3.3.2: First implementation of cluster-level safety and security services".

[23] DREAMS Consortium, "D3.3.3: Final implementation of cluster-level safety and security services".

[24] DREAMS Consortium, "D4.4.2: Integration and Support Report," DREAMS Consortium, 2017.

[25] "ZC706 Evaluation Kit webpage," Xilinx, [Online]. Available: http://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html. [Accessed 1 April 2015].

[26] Xilinx, *ZC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable SoC Datasheet,* Xilinx, 2013.

[27] M. D. G. R. L. G. M. L. P. Marcello Coppola, Design of Cost-efficient Interconnect Processing Units, CRC Press, 2009.

[28] Xilinx, "MicroBlaze Processor Reference Guide," Xilinx, 12.

[29] Xilinx, "Zynq-7000 All Programmable SoC Technical Reference Manual," Xilinx, 2015.

[30] DREAMS Consortium, "D9.1.2: Intermediate Community Building Report".

[31] DREAMS Consortium, "D8.3.1: Preliminary assessment report related to improving or calibrating the technological results".

[32] DREAMS Consortium, "D8.1.1: Specification of the use cases along with technologies and assessment metrics".

[33] DREAMS Consortium, "D7.3.1: Preliminary assessment report related to improving or calibrating the technological results".

[34] DREAMS Consortium, "D7.1.1: Wind power use cases specifications".

[35] DREAMS Consortium, "D61.1.1: Avionics use cases specifications and technology assessment method".

[36] DREAMS Consortium, "D6.3.1: Preliminary avionics assessment report".

[37] DREAMS Consortium, "D5.6.1: Functional safety management for DREAMS architecture".

[38] DREAMS Consortium, "D5.5.3: Method for certifying mixed-criticality product lines".

[39] DREAMS Consortium, "D5.4.1: Guidelines for process and tool integration".

[40] DREAMS Consortium, "D5.3.1: Cross domain mixed-criticality patterns".

[41] DREAMS Consortium, "D5.2.3: Fault injection framework".

[42] DREAMS Consortium, "D5.2.2: Prototype implementation of simulation framework for DREAMS architecture".

[43] DREAMS Consortium, "D5.2.1: Specification of simulation framework".

[44] DREAMS Consortium, "D5.1.3: Modular safety-case for selected mixed-criticality networks".

[45] DREAMS Consortium, "D5.1.2: Modular safety-case for selected COTS multicore processors".

[46] DREAMS Consortium, "D4.2.2: Final implementation of a platform configuration files generator".

[47] DREAMS Consortium, "D4.2.1: Specification and first implementation of a platform configuration files generator".

[48] DREAMS Consortium, "D4.1.3: Final implementation and improvement of the offline adaptationstrategies for mixed criticality".

[49] DREAMS Consortium, "D3.2.3: Final implementation of global resource management services".

[50] DREAMS Consortium, "D3.2.2: First implementation of global resource management services".

[51] DREAMS Consortium, "D3.2.1: High-level design of global resource management services".

[52] DREAMS Consortium, "D3.1.3: Final Implementation of Mixed-Criticality Cluster Communication Services".

[53] DREAMS Consortium, "D3.1.2: First implementation of mixed-criticality cluster communication Services".

[54] DREAMS Consortium, "D3.1.1: High-level design of mixed-criticality cluster communication services".

[55] DREAMS Consortium, "D2.4.2: Extensions and modifications related to supporting integration with industrial demonstrators".

[56] DREAMS Consortium, "D2.4.1: System platform integration with virtualization-specific HW/SW extensions, system software, reconfiguration and preliminary test benches".

[57] DREAMS Consortium, "D2.3.3: System software extensions for the Spidergon STNoC".

[58] DREAMS Consortium, "D2.3.2: Extended real-time KVM hypervisor with regression tests and validation on ARM's Fast Models".

[59] DREAMS Consortium, "D2.1.3: RT-level design specifications of a) virtualization and memory interleaving support of the Spidergon STNoC backbone at the network interface layer and b) a bus-to-noc bridge for seamlessly interconnecting Spidergon STNoC and network gateways".

[60] DREAMS Consortium, "D10.2.3: Annual report 3 on dissemination, exploitation and training including updated dissemination, exploitation and training plans".

[61] DREAMS Consortium, "D1.8.1: Report on intermediate assessment and support for demonstrators".