Guaranteed Component Assembly with Round Trip Analysis
for Energy Efficient High-integrity Multi-core Systems

Project Number 333053

D4.2 – Modelling, Analysis and Transformations for Logical, Physical and Virtualized Partitioning on Multi-core Targets - Spec Version

Version 2.0

29 April 2016

Final

Public Distribution

UPD, TCS, ISEP, CSW, AICAS

Project Partners: **AENSys, Aicas, Atego, Budapest University of Technology and Economics, Critical Software, EADS, Intecs, ISEP, Maelardalens University, OTG Solutions, SINTEF, Thales Communications & Security, The Open Group, University of Florence, University of Padova**

Every effort has been made to ensure that all statements and information contained herein are accurate, however the Partners accept no liability for any error or omission in the same.

© 2016 Copyright in this document remains vested in the CONCERTO Project Partners.

DOCUMENT CONTROL

Version	Status	Date
0.0	Initial coarse-grained document structure	14 Nov 2013
0.1	First draft implementation of the agreed report plan	22 Sep 2014
0.2	Consolidated version after discussion at plenary	24 Oct 2014
1.0	Version approved for public release	2 Nov 2014
2.0	New version as required at Y2R	29 Apr 2016

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This report emanates from Task T4.2 and discusses the understanding at M18, of the industrial use cases with bearing on its subject matter, that is to say, modelling, analysis and transformation of partitioned systems. Notably, of the large spectrum of industrial use cases active in CONCERTO, only two of them, the avionics use case by Airbus, and the automotive use case by CSW exhibit needs in the area of concern of this document. Interestingly, however, while this document was being edited, attention was mounting in European research around the theme of Mixed-Criticality Systems (MCS), which has evident bearing on the scope of WP4 and Task 4.2. In view of this fact, this document includes a critical review, seen from the perspective of CONCERTO, of the research landscape in the MCS area as reflected in the research portfolio analysis produced by the Mixed-Criticality Cluster initiative promoted by the European Commission.

1. INTRODUCTION

This document, tagged D4.2, entitled “Modelling, analysis and transformation solutions for logical, physical and virtualised partitioning on multicore targets” and originally due at m12, emanates from task T4.2. According to the CONCERO work plan, D4.2 is the first of two deliverable reports intended to should incrementally address the vast subject matter. Its successor, D4.6, was to follow at m18.

The production of this document has been more laborious than anticipated. The maturation of the industrial use-case requirements with bearing on the ambit of WP4 and of task T4.2 in particular, proved slower for progress and narrower for scope than expected. In view of the slowness, the release of D4.2 was first postponed by 6 months, from m12 to m18, to allow more time for the maturation of the use-case requirements. To mitigate the reduction in scope, instead, more attention was given to positioning CONCERTO in the landscape of the European research on mixed-criticality systems.

The result obtained at m18 however was judged not adequate by the expert reviewers, too distant from the scope declared in the Description of Work. D4.2 thus had to be revised and resubmitted.

This version of the document addresses the critique received at the 2nd Year Review, with the benefit of hindsight and the consequent deeper understanding of the industrial use cases. The document provides an outline of the two industrial use cases with bearing on the domain of responsibility of D4.2 and summarises the essential challenges that emanate from them, which have in fact consolidated in ARINC-653 type solutions for time and space partitioning (from the avionics use case) and mixed-criticality design and safety analysis (from the CSW automotive use case). In view of the latter element, this document retains the essential elements of the survey of the mixed-criticality landscape produced in the year 2014 by the Mixed-Criticality Cluster of FP7 projects established by the European Commission, and a scrutiny of the research and collaboration opportunities that opened for CONCERTO in that area.

2. USER REQUIREMENTS WITH BEARING ON PARTITIONING AND MIXED-CRITICALITY

2.1 AVIONICS USE CASE (AIRBUS)

2.1.1 Outline of the Use-Case Needs of Relevance

The Airbus Avionics Use Case explores the support that CONCERTO can offer for modelling systems that conform to the IMA (Integrated Modular Avionics) reference architecture [1]. IMA seeks isolation by means of partitioning CPU time and processor memory. For single-core processors, time partitioning is achieved by adopting (at design time) and enforcing (at run time) an execution plan that ensures that no two partitions can ever contend for the CPU. A partition is the unit of assignment of CPU time and of processor memory, and corresponds to a distinct aggregate of application tasks, with a collective execution-time budget and a collective period, often seen as a rate.

Airbus develops IMA-compliant systems using the concepts of Major and Minor Frames, called MAF and MIF respectively. The MAF corresponds to the hyper-period of all partitions assigned to the processor, further requiring that all partitions have a fixed periodic rate and that periods are harmonic. The MIF is usually the highest common integer divisor of the MAF, and it corresponds to the unitary duration of continuous execution assigned to a single partition. The full execution of a whole partition normally spans multiple MIFs that need not be contiguous. The schedule of MIFs within the MAF normally follows a table-driven regime, so that switching between partitions at the boundary of two adjacent MIFs does not require pre-emption unless in case of a partition overrunning its MIF. To counter this risk, the MIF assignment to partitions is conservative and always includes a non-zero slack. As scheduling within the MIF uses normal fixed-priority pre-emptive scheduling, the residual slack corresponds to a background task, with the lowest priority, which is always deferred to the end of the MIF and in effect relinquishes control to the higher-level scheduler.

The execution model of such an architecture entails a hierarchical, two-level, scheduler: the upper one assigns CPU time to individual partitions; the lower one assigns CPU time to tasks within individual partitions. The upper-level scheduler divides time into MIFs within a MAF, and assigns whole MIFs to individual partitions following a static scheduling table. The lower-level scheduler assigns MIF time to the tasks of the partition active in that MIF, using fixed-priority preemptive scheduling.

Whereas partitions are isolated for time and space, communications are possible among them, using buffer-like structures for data transfers from one memory space to another, which do not need mutexes, as – in single-processor systems – partitions can never contend for execution and therefore can never pre-empt one another. Mutexes may instead come of use within partitions, which use pre-emptive scheduling for their tasks and do not exclude that tasks may share logical resources.

2.1.2 Summary of Challenges

To support the avionics use case from the perspective of D4.2, CONCERTO shall:

1. Allow the user to model the IMA (and ARINC-653) concepts of partitions, processes and operations.
2. Support the two-level scheduling regime assumed in the IMA architecture, the automated generation of a partition schedule that meets the design requirements, and the analysis of feasibility of the resulting two-level model of execution.

2.2 AUTOMOTIVE USE CASE (CSW)

2.2.1 Outline of the Use-Case Needs of Relevance

Safety is one of the key issues in automobile industry. New functionalities of competitive interest, such as driver assistance and vehicle infotainment systems, increasingly interact with the domain of system safety. The development and integration of multiple functionalities with different safety guarantees in the same execution platform (which is associated to the notion of mixed-criticality system and obey cost-reduction objectives) increases the complexity of safety-aware development processes and raises the obligation for the developer to provide evidence that all system safety objectives are satisfied.

The safety requirements associated to software components can be met partitioning the assignment of execution resources, and further providing safety barriers between components assigned to distinct levels of safety, so that those with lower guarantees (hence possibly lower development quality) cannot undermine those with higher guarantees.

To achieve this kind of partitioning, the system development practice requires every individual software component, as well as every system resource that those components may use, to be assigned to a criticality level determined by the safety requirements that apply to it.

This assignment allows the deployment of measures to detect resource-sharing risks across safety partitions, which may endanger safety. Those measures may employ design-time precautions (hence physical isolation, which is not of interest to mixed-criticality systems), or run-time means implemented in hardware or software.

Interestingly, the above assignment is not always statically fixed for all system resources. At run time in fact, certain resources, originally available to lower-criticality software components, may have to be assigned to higher-criticality software components, therefore becoming simultaneously unavailable to the former. This contingency may be regarded as a change of mode of operation. In that respect, a mode of operation specifies rules for the assignment of system resources to software components. System resources that are accessible to lower-criticality software components in normal conditions may be reserved to higher-criticality software components in executional situations, for example, in the event of a collision between the car and an obstacle. Support for the specification of multiple modes of operations should be provided at design level, so that the user may specify the desired assignments, as well as at run time, where barriers should be provided in the execution environment that ensure that all resource-to-component assignments conform to the rules specified in the current mode of operation. Such barriers can be implemented in hardware or software.

2.2.2 Summary of Challenges

For the purpose of the automotive use case, CONCERTO should:

1. Support the specification of mixed-criticality systems, with at least two levels of criticality.
2. Support the specification of modes of operation that specify the allowable assignments of system resources to software components.
3. Support the specification of contingency plans that switch between modes of operation on the occurrence of specific (kinds of) events.
4. Support the specification of core affinities that allow component-to-core assignments that match the level of criticality assigned to components and system resources.

2.3 SUMMARY OF REQUIREMENTS

Table 1 below singles out the user requirements captured in WP1 that relate to the two use cases discussed above. It is against this set of requirements that coverage analysis shall be conducted at M36, to determine the success of WP4 in the support of the described industrial use cases.

Req. No.	Piority	Derived Requirement
R1.1	SHALL	CONCERTO shall support the definition of modes of operation for SW components. The set of components that operates under a specific mode of operation constitutes a scenario.
R1.2	SHALL	The analysis tools shall be able to compute the response time relevant to user-provided scenarios.
R1.3	SHALL	The model transformations shall support modes of operation.
R1.4	SHALL	The code generation shall generate containers that are able to switch mode of operation.
R6.1	SHALL	The modeling language shall support the definition of different types of schedulers and their parameters. There shall be support for multicore schedulers for SMP and heterogeneous systems.
R26.1	SHOULD	The modeling language should define resources like buffers, semaphores and their non-functional properties (like, size, queuing policy) and services to manipulate these resources (intra and inter partition). The communication mechanisms shall be instantiated to the specific domain of interest.
R26.2	SHOULD	The modeling language should be able to represent resources used in Aerospace like ARINC 653 buffers, semaphores and their non-functional properties (like size, queuing policy) and the services to manipulate these resources.
R32.1	SHALL	The modeling language shall support the definition of different levels of criticality.
R32.2	SHALL	The analysis tools shall be able to analyse mixed-criticality systems.

R32.3	SHALL	The runtime environment shall support non-critical/non-trusted, as well safety-critical SW components.
R32.4	SHALL	The runtime environment shall support mixed-criticality systems under the assumption that components execute in a pre-determined HW resource partition within the same level of criticality.
R33	SHALL	All system components (HW and SW) shall have their criticality specified as an attribute.
R42.1	SHOULD	The modeling language should define communication attributes for SW and HW components.
R42.2	SHOULD	The CONCERTO design space should enforce model constraints (i.e. constraint the user actions) to ensure the correctness of the model regarding communication concerns.
R69	SHALL	CONCERTO shall support time partitioning between critical and non-critical tasks through CPU budgets
R71.1	SHOULD	The modeling language should allow specifying core reservation for SW components.
R71.2	SHOULD	Core reservation should be taken into account in model transformations.
R71.3	SHOULD	Core reservation should be taken into account in the assignment process.
R72.1	SHOULD	The modeling language should allow specifying processor affinity for (critical) SW components.
R72.2	SHOULD	Processor affinity should be taken into account in model transformations.
R72.3	SHOULD	Processor affinity should be taken into account in the assignment process.
R73.1	SHALL	The CONCERTO deployment view shall allow the user to define several execution nodes, containing single core or multicore processors, and to define the allocation of the SW components to the execution resources on these nodes. There shall be no specific limitation when the user deploys the components, either on a single node or on several nodes.
R73.2	SHALL	The CONCERTO design space shall assist the user to handle multicore deployment by enabling specific model constraints for HW components.

Table 1: Summary of use-case requirements with bearing on partitioning and mixed-criticality.

3. THE MIXED-CRITICALITY SYSTEMS LANDSCAPE

3.1 SAFETY CONSIDERATIONS

Before being allowed to deploy applications with bearing on safety (often referred to as safety-critical), to a physical system, a certification authority must validate whether all the safety-related norms – with bearing on development practices and implementation features – are fulfilled in the system under consideration. All the components constituting that system (the software, the hardware and the interfaces) are scrutinised to ensure conformance to domain-specific safety standards. These standards impose a host of constraints, on the design process as a whole, on the hardware and software implementation, on the expected operation behaviour and responsiveness of the system.

Most of these constraints address safety-related issues. In the ideal case, a safety-critical application should be executed on dedicated (non-shared) computing elements and its code and data should be stored in a highly protected (part of the) memory. From a timing perspective, guarantees are obtained through timing and schedulability analysis techniques, which are typically simpler and more accurate when the applications running in the systems can be considered as being sufficiently isolated from one another.

The issues related to spatial and temporal isolation have been brought further upfront by the advent of mixed-criticality components integrated in a same system. Indeed, as a first step in the certification process, each component (software and hardware) is categorised by its level of criticality. A component can be either critical or non-critical. Critical components can be further subdivided using *safety integrity levels* (SIL). Systems that comprise components of different criticalities are usually referred to as mixed-criticality systems. When integrated in the same system, components of different criticalities co-exist and form a fragile ecosystem: they can be “connected"” to each other, directly or indirectly, by communicating results to (or by reading inputs from) a same component. In addition, they can share low-level hardware resources such as cache memories, communication buses, main memory, etc. In this mixed-criticality context, the IEC 61508 standard [IEC:2010] has put very tight constraints on the certification process: *“An E/E/PE safety-related system will usually implement more than one safety function. If the safety integrity requirements for these safety functions differ, unless there is* ***sufficient independence of implementation*** *between them, the requirements applicable to the highest relevant safety integrity level shall apply to the entire E/E/PE safety-related system”.* The EN 61508 represents a generic safety standard that is widely used throughout the industry and serves as a common base for domain specific standards such as for example the ISO 26262 (automotive domain, see 26262-4, req. 7.4.2.3) [ISO:2011]. The higher the integrity level of a component, the more complex and costly the certification process will be.

3.2 MIXED-CRITICALITY SYSTEMS CONCERNS

The last round of the FP7 work program in the area of embedded systems, which occurred in 2013, launched three projects that, though exploring different research avenues with different methods and techniques, had the common trait of addressing mixed-criticality-systems (MCS) challenges. The European Commission therefore formed those three projects into a "Mixed-Criticality Cluster" (MCC) and assigned them the following two tasks:

- to produce a comprehensive report including a portfolio analysis of the position of various relevant, both large and medium-size, initiatives (from FP7, ARTEMIS and their respective programme successors) into the MCS research landscape[1], and the possible avenues of collaboration among them[2];
- to set up a Web-hosted forum[3] where their MCS-related findings could be shared more promptly with the outside and feedback could be sought from a wider community.

At the time this document was first written, considering that the use-case requirements discussed in section 2 brought forward partitioning and mixed-criticality needs, it was thought as opportune to highlight areas of contact between this trait of CONCERTO and the MCC, so that common research challenges and opportunities of collaborations could be identified.

To this end, this document first provides a critical analysis of the MCC portfolio, then positions CONCERTO in the corresponding context, and finally singles out research opportunities that CONCERTO might explore.

3.2.1 MCC Portfolio Analysis

The opening line of the MCC portfolio quotes the report from the Workshop on Mixed Criticality Systems, held on 3 February 2012, in Brussels [MCR: 12]: "Multi-/many core computing is seen as both a key enabler for future systems and also a driver offering opportunities to produce new functionality. In addition to more computing performance being available, high value-added can be envisaged, e.g. safety, energy efficiency, maintenance, augmented reality interfaces and graphics. The increasingly connected world is also leading to integration of currently separate mobile/public/private functionalities. Here new services are being introduced via Internet-services such as navigation systems for cars and remote support/configuration capabilities providing better and faster customer service. User's increasing expectations for multimedia services driven by the consumer market are placing strong demands for the same services to be available in our aircraft, rail and road transport".

All indicators suggest that the use of multicore devices will progressively replace applications that traditionally used single-core processors, and multi-core architecture will supplant single-core processors, leaving designers no choice but to migrate to the processors that are available commercially.

With the advent of multicore processors, industrial developers get the opportunity to move from a federated system, with its numerous design constraints and vast unused capacity, to a more integrated one. A typifying example of this case arises from the automotive domain, where each application is assigned its own Electronic Control Unit.

[1] Some preliminary elements of that landscape are outlined at http://ec.europa.eu/digital-agenda/en/successful-research-projects under the heading "Mixed Criticality Computing".
[2] This report is not yet public at the time of this writing, but it has been circulated – for editing and approval – among all partners in the Mixed-Criticality Cluster, which includes members of the CONCERTO team.
[3] Live as of 15 September 2014 at http://www.mixedcriticalityforum.org/home/.

Integrating multiple applications on the same multicore processor is attractive as it leads to a reduction in manufacturing cost and weight, as well as to better energy efficiency. This integration however may cause critical and non-critical functionality to coexist on the same platform, which has caused the concept of mixed criticality to emerge.

Two distinct flavours of mixed criticality exist. One where the central concern is *timeliness*; the other where it is *safety*. Time-critical systems warrant the availability of processing outputs at given points in time, whether absolute or relative to some set reference. Safety-critical systems warrant the integrity of functions, which must either provide the correct output (correct against a given specification) or indicate its failure, with no allowable deviation in either the value or the time domain. In the context of multicore processors, the provision of timeliness must place strict (in quantification) and trustworthy (in assurance) upper bounds on the interference that the execution of the function of interest can suffer from contention on access to shared physical (and possibly logical) resources. Multicore processor architectures cause a frightening increase in the number and intrinsic use of shared hardware resources, which poses a serious threat to the achievement of *sufficient degree of isolation* in the time domain. The provision of safety instead must assure that the execution of the function of interest suffers no interfering disturbance from the outside.

The difference between the two concerns was more marked in traditional single processors: timelines would be sought by placing – by design, analysis and execution means – upper bounds on the execution-time interference possibly caused by competing foreign applications; safety would be sought instead by opting for strict isolation, in space (for the value domain) as well as in time. At the cost of some, possibly large, extent of over-provisioning, isolation would serve both purposes well. The considerable simplifications accrued in design, integration and verification was regarded to amply counter-balance the increased costs of hardware and harness.

This tenet is now shattered by the nature and intent of multicore processors. The level of underutilisation (in effect, waste of capacity) incurred in single processor systems as caused by the overprovisioning inherent to the quest for a *sufficient degree of isolation* was naturally limited by their limited capacity in size and throughput. In a multicore environment instead, the extent of wasted capacity is amplified by the number of cores in the processor, which makes for a much bigger loss.

Two extremes can be identified in the possible response to this evident contradiction.

The architectural approach taken in the DREAMS project[4] places at one end of the spectrum. DREAMS seeks guarantees of isolation from unbounded interference by statically apportioning all accesses to shared resources throughout the entire execution stack of the system, networks, memories, busses, and cores. This is expected to cater for statically defined upper bounds, which can be used in validation and certification arguments, and prevail over any concerns with underutilization. The central challenge for DREAMS is thus to enact a design process (and support techniques, both offline and

[4] http://www.uni-siegen.de/dreams/home/

online) that hide from human developers the complexity of resolving all conflicts by way of static allocation.

The other end of the spectrum, directly evoked by H2020 call ICT 4 [ICT4:14], is a system in which the extent of contention for shared hardware resources actually occurring at run time – and the consequent interference effects – is holistically controlled by appropriate agents in a manner that opportunistically (preventively better than reactively) self-adapts to the context of execution. The intent behind this approach is that the waste of computational capacity is kept at bay while sufficient guarantees of isolation can be asserted for the design as well as for the implementation.

Owing to its heritage from CHESS, recalled in the introduction, CONCERTO has features that differentiate it from both extremes. CONCERTO strictly separates PIM (user-level modelling) concerns from PSM (run-time environment) concerns. It captures platform features, virtues and constraints of consequence to non-functional properties of interest to the application (including isolation), and amalgamates them in transformation rules that encapsulate application components in platform-specific containers and connectors which controllably bind those components to all the services that they may need to access platform resources at run time. This approach postulates a model of computation that can be treated compositionally, so that the interference effects arising from parallel and concurrent execution can be upper bounded analytically with sufficient tightness. CONCERTO applies that benefit to coarse-grained application components, product of decoupled top-down design. This contrasts with DREAMS, where application parts (not necessarily seen as components) result from the bottom-up pressure to scant contention effects, which inevitably carries the burden of considerable architectural coupling. As CONCERTO does not develop its requisite compositional model of computation, but rather uses them as they emerge from the state of the art, it has the interesting opportunity to move along the solution spectrum, away from DREAMS towards the ICT 4 end. In that respect, there is solid potential that the CONCERTO solutions, in concept and to some extent in technology, can fit a wider range of innovative MCS platforms.

3.2.2 Positioning of CONCERTO in the MCC Portfolio Analysis

While the reasoning above is expressed in general terms, it is useful to narrow down specifics of the MCC Portfolio Analysis to the work plan of CONCERTO for its final 18 months.

- *Reference architectures* are needed to build critical mass, enable cross-domain uptake and avoid fragmentation.

  CONCERTO aims to demonstrate that its component-based development method and its reliance on automated transformations of development artefacts are in good fit with industrially-qualified reference architectures such as IMA (avionics) and AUTOSAR (automotive), and can be a useful platform where the pros and cons of their transposition to multicore processors can be studied. These challenges are specifically addressed in tasks T4.1, T4.2 and T4.3, in support of task T5.3.

- *Methodologies and integrated toolsets* are required that reduce certification effort and allow incremental certification and re-certification.

  CONCERTO seeks to support *incremental development and qualification*. Thanks to the principles explained in D2.2 (released at m12), using CONCERTO, the distinct application parts (i.e., components) can be developed independently and then integrated in the final system, with guarantees on their composition. Such guarantees are established by (platform-aware) model-based analysis for non-functional properties such as timeliness and dependability, and subsequently preserved at run time by the combined action of containers and connectors that police the take of their designated component on shared system resources. The ability to express, analyse, and preserve these guarantees stems from two distinguishing features of the CONCERTO approach: (1) the existence of a clear separation between the development and verification of the functional behaviour of the components that will coexist in the system; and (2) their *containers* and *connectors*, which integrate them in the run-time environment and ensure the properties and guarantees sought by the components. As containers and connectors directly rest on specialized services of the hardware, middleware and operating system for time and space budgets and consumption (not available to individual components, hence not subvertible from them), they determine what the components can do to the outside environments in the use of space and in the use of time. These important guarantees can be attained even in a multicore environment; what is more difficult instead is to determine application budgets in the execution time dimension that can set trustworthy yet tight enough upper bounds. There is room in CONCERTO for drawing direct benefits from recent research results produced in this regard by the FP7 projects operating in the cited Mixed-Criticality Cluster. Opportunities are being sought to use in CONCERTO some of the technology solutions developed in the PROXIMA project[5]. PROXIMA develops execution platforms (processor hardware and operating system), both ad-hoc and commercial-off-the-shelf, that remove the causal nature of the dependence on execution history that traditional processor architectures exhibit in their timing behaviour, which is one of the root causes of concern in mixed-criticality systems. Interestingly, PROXIMA solves platform-specific concerns with solutions that can be almost seamlessly integrated in CONCERTO owing to the cited separation of concerns. WP4 will use some of the execution platforms developed in PROXIMA for the avionics and space domain, for the processors and operating systems of relevance to those domains.

  In addition to assisted design facilities, static analysis and correct-by-construction code generation, CONCERTO also integrates run-time monitoring capabilities, attached to containers and connectors (hence outside of the application part) to verify the preservation of the non-functional properties ascertained at design time. Those facilities help detect misconceptions, inaccuracies and/or errors in the modelling of the system. The data extracted during system execution are treated and compared with the static analysis results, to back-propagate useful and understandable run-time information to the system designer. This allows for

[5] http://www.proxima-project.eu/

incremental development and verification of the system and improves the accuracy of the system parameters on which static analyses are performed. Furthermore, as an exploratory solution for increasing the reliability of the deployed application, an inline monitoring and verification framework is under development in CONCERTO. Functional and non-functional properties are verified at run time to detect misbehaviour of the system, in which case monitors can notify the system and trigger safeguarding measures, thereby acting as a safety net for safety critical applications. In this regard, solutions and technologies will be drawn from the EMC^2 project[6], an ARTEMIS JU project, identified as a primary interlocutor by the MC Forum. EMC^2 addresses mixed-criticality concerns at all the levels of the system design; from hardware to modelling, and including run-time monitoring and verification.

- *Parallelisation techniques* that allow application parallelism to be exploited and mapping of tasks to cores are required to help with programmability of homogeneous and heterogeneous multi-core processors.

  Although this is not an explicit objective of CONCERTO, some elements of that have been captured by research opportunity O1.1 discussed in D4.1, and further pursued in task T4.1. The solutions developed for the modelling and timing analysis of parallel tasks might be demonstrated in the automotive use-case conducted by Intecs: a decision on the placement of this feature in that use case will be made before m24.

- *Virtualisation techniques* will become key technologies to preserve the integrity of highly integrated architectures where real-time and safety-related tasks co-exist with consumer applications. Virtualisation techniques are further required to decouple the underlying hardware from the application and allow migration of legacy code onto new platforms.

  In the quoted text, still assuming a single-processor mode of operation, the use of virtualization provides for two distinct benefits. Firstly, it enables isolation so that – normally via arbitration from a trusted hypervisor – applications run in dedicated virtualized time-and-space assignments, which are designed and implemented so as to never cause unwanted interaction. Secondly, it allows legacy applications traded in binary to continue to execute in spite of changes in the host platform, for processor hardware or operating system. The transition to a multicore setting leaves much of this concept to be reconsidered. A model where a hypervisor running on core 0 takes control over the entire processor and strictly arbitrates access to all shared hardware resources across all cores is in fact plausible for warranting isolation, but it is hardly an option where real-time behaviour is of consequence. Hence, there is no consolidated industrial reference model for this feature, on which CONCERTO can confidently invest its limited resources. The FP7 MultiPARTES project[7] recently addressed that challenge. Its technical approach uses model-based development principles for (a) designing the application, (b) guiding its deployment

[6] http://www.artemis-emc2.eu/
[7] http://www.multipartes.eu/

to virtualized guest environments, and (c) determining TDMA-based scheduling of guest applications. Thanks to its heritage from CHESS, CONCERTO sports solutions that are very similar to MultiPARTES for (a) and (b), in fact more flexible than them owing to its approach based on sharp separation between components (in the user space), and highly configurable containers and connectors (in highly assured PSM space). As (c) is very platform specific, it has modest bearing on CONCERTO, whose development principles rest on the automated generation of platform-specific artefacts. It can be therefore maintained that if there was an industrial use case in CONCERTO where a platform like that assumed in MultiPARTES was required, model-to-model transformations could be developed in CONCERTO that would allow the automated calculation of the required TDMA schedule and the generation of the required executables. In fact, this schedule generation will be demonstrated in the avionics use case, described in D4.5 and D4.6.

3.3 Research Opportunities in the MCS Landscape

As a direct reflection of the MCS challenge that permeates the industrial context of interest to CONCERTO, the opportunity has been sought to extend the requirements base discussed above with research themes that fit well in the CONCERTO use cases addressed by the WP4 partners and that those partners are ready to address.

- *Mitigation of state-sensitive timing perturbation (link with PROXIMA)*. One of the way to mitigate the effects of state perturbation caused by pre-empting execution is to use recent research results obtained in the PROARTIS project[8], precursor to PROXIMA, cited member of the MCS Forum. The results of interest specifically concern the development of a time-composable implementation of an ARINC-653 compliant real-time operating system [BMV:12]. That solution has three distinguishing features of interest to MCS. (1) It provides constant-time primitive services, so that their response time is not jittery and sensitive to history of execution. (2) The execution of those services does not perturb the state of the hardware resources, so that the effect of their execution is strictly additive to the application timing. (3) It offers non-preemptive scheduling within partitions so that partition processes do not interfere with one another. For more details, refer to D4.5 and its Annex. That same solution is being ported in PROXIMA to an avionics multicore processor, which will therefore serve the needs of the avionics use case in CONCERTO perfectly.

- *Extension of support for modelling, deployment and execution of IMA systems on multicore processors (link with WP2 and with MultiPARTES).* Viewed holistically, the support provided in CONCERTO to respond to Reqs. 32, 33 and 69 recalled above will bear a very direct resemblance with the solutions adopted in the MultiPARTES project, from modelling down to deployment to execution. The CONCERTO solutions will have competitive differences with those adopted in MultiPARTES (from augmenting the CHESS component model so that it can capture the modelling of IMA partitions, to finding a partition

[8] http://www.proartis-project.eu/

schedule with low specification cost for the user, to providing a time-composable multicore execution environment without using a hypervisor).

- *Use of modelling attributes to specify component criticality (link with CONTREX and WP2).* For supporting the specification of domain-specific criticality levels and their attachment to system components (or aggregates thereof), CONCERTO will use solutions proposed by the CONTREX project[9], member of the cited mixed-criticality forum. The solutions of interest appear in the CONTREX deliverable report D2.1.1, entitled "CONTREX System meta-model", kindly shared with CONCERTO by the authors, with permission by the consortium lead, and will apply, in WP4, to the modelling of partitions for the avionics and the automotive use cases. The intent here is to instigate a technical and methodological dialogue with CONTREX to determine solutions that draw benefit from a common effort, larger than a single project.

- *Development of a run-time monitoring and verification framework for safety critical systems (link with the EMC project[2], especially with its WP2).* In mixed-criticality systems, high criticality tasks with hard timing constraints may share the same execution platform with non-critical or non-trusted applications. Furthermore, the high number of hardware resources shared between the different cores of modern COTS multicore processors (e.g., shared buses, caches, memories, etc.), avoid a complete isolation in space and time between tasks. The calculated interference caused by those shared resources is often extremely pessimistic. This gave birth to probabilistic analysis methods such as those developed in the PROARTIS project and currently extended in PROXIMA. Yet, those methods are based on probabilities and hence associate a probability of failure to the bounds calculated for the timing properties of the tasks. However low, this probability of failure exists and cannot be neglected. Consequently, an application running in a multicore environment could never be totally trusted, and unexpected behaviours of the applications may potentially be observed. As an answer to this major issue brought in by multicore architectures, CONCERTO is developing in the context of T4.4, inline monitoring techniques with the intention of creating a *safety net* around safety critical applications. This framework allows functional and extra-functional properties to be verified at run-time. Abnormal behaviours can be detected and safeguarding measures may be triggered. The reliability of the system is therefore improved as unexpected execution scenarios or system behaviours should never remain undetected and unaddressed. Furthermore, the solution foreseen in CONCERTO should exhibit multiple desirable properties for mixed-criticality systems:

  (i) although running concurrently with the monitors, the monitored tasks are unaware of the existence of the monitors as well as their numbers,

  (ii) the monitors are unaware of the specific implementation of the application,

  (iii)the monitors are unaware of the other monitors coexisting in the system,

[9] https://contrex.offis.de/home/

(iv) monitors and monitored applications can be completely isolated from each other from a memory address-space viewpoint,

(v) monitors cannot communicate or exchange data with tasks, neither monitors can communicate between each other.

4. CONCLUSION

This report has discussed the understanding at M18, of the industrial use cases with bearing on the subject matter of Task T4.2, that is, modelling, analysis and transformation of partitioned systems. Of the comparatively large spectrum of industrial use cases active in CONCERTO, only two of them, the avionics use case conducted by Airbus, and the CSW variant of the automotive use cases exhibited needs in the area of concern of this document. Interestingly, both use cases had prevalent interest in modelling and analysis, prime specialities of CONCERTO, and did not extend to code generation and execution on target. In the case of the avionics use case, the latter limitation was due to the difficulty in getting access to realistic functional code and yet not too dependent on the specifics of a proprietary processor and run-time environment. In the case of the CSW automotive use case, instead, the choice appeared to reflect an intentional priority placed on exploring modelling-based solutions.

5. REFERENCES

[MCR:12] http://cordis.europa.eu/fp7/ict/embedded-systems-engineering/documents/sra-mixed-criticality-systems.pdf

[ICT4:14] https://ec.europa.eu/digital-agenda/events/cf/ictpd14/document.cfm?doc_id=28847

[BMV:12] A. Baldovin, E. Mezzetti, T. Vardanega, “Kernel-level time composability for avionics applications”, in Proceedings of the 12th International Workshop on Worst-Case Execution Time Analysis (WCET), pp. 69–80, 2012.