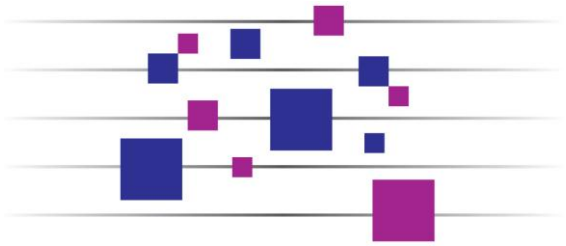

CONCERTO

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

Project Number 333053

D4.6 – Modelling, analysis and transformation solutions for logical, physical and virtualised partitioning on multicore targets - Toolset Initial Version

Version 1.0

18 November 2014

Public Distribution

UPD, CSW, ISEP

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**

DOCUMENT CONTROL

Version	Status	Date
0.0	Initial structure	01 Set 2014
0.1	UPD and CSW contributions	22 Oct 2014
0.2	Ready for internal review	23 Oct 2014
0.3	CSW additional contribution	30 Oct 2014
0.4	UPD additional contribution	5 Nov 2014
0.5	Added Requirements table	10 Nov 2014
0.6	ISEP contribution	11 Nov 2014
1.0	Final version	18 Nov 2014

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CHESS ML	CHESS Modelling Language
PIM	Platform Independent Model
PSM	Platform Specific Model
WCET	Worst Case Execution Time

EXECUTIVE SUMMARY

This document describes the proposed technological solutions for the support of partitioning on multicore targets for a subset of the CONCERTO use cases. Each of them poses a number of challenges with respect to the partitioning and multicore support, notably, how that support requires the extension of the methodology, the metamodel and the implementation of transformations needed to generate the inputs of the analysis and for the code generation. This document focuses mostly on the methodology extensions that provide a broad view of the activities to investigate during the next months and the metamodel extensions that are a prerequisite for the next activities.

1. INTRODUCTION

This document describes the proposed toolset modifications required to support logical, physical and virtualized partitioning on multicore targets. Partitioning support is required for a small subset of the industrial use cases: the Airbus Group use case for the Avionic domain and the CRITICAL SOFTWARE use case for the Automotive domain.
The CHESS Methodology and Toolset baseline do not support partitions of any kind, therefore the development of a solution must consider the extension of the methodology, the metamodel, the implementation of specific analysis and the related transformations and back-propagation of the analysis results.
The rationale of this deliverable is to first present a brief description of the use case in section 2 – more details are in D1.2 – and then to propose the solutions and their challenges in section 3. Due to the crosscutting nature of the deliverables, it is possible that some of the solutions are already described in other deliverables, in which case, only the corresponding reference is presented here.

2. USE CASE DEMONSTRATORS RECAP

2.1 AIRBUS GROUP

2.1.1 Overview

This use case requires the conformance to the IMA (Integrated Modular Avionics) reference architecture [2]. The IMA goal is to achieve isolation by means of the partitioning of time and space. The time partitioning is achieved by statically designing a pattern for the activation of different partitions using the concept of Major and Minor Frames: MAjor Frame (MAF) is usually the least common multiple of the periods of partitions. MInor Frame (MIF) is usually the highest common factor of the periods of partitions. Within partitions, tasks are usually interleaved using a fix-priority scheduling protocol. A two-level scheduler is therefore implied. Communications, either inter or intra partitions, are performed using common real-time resources like buffer and mutexes. The space partitioning is achieved by assigning one core (in the case of multicore) and one memory partition to one IMA partition.

2.1.2 Challenges

The main goals of CONCERTO are:

1. how to model the concept of IMA partitions;
2. how to provide a timing analysis for the two-level scheduler and the related transformations;
3. how to provide transformations for code generation.

Section 3.1 describes the proposed solutions and their challenges.

2.2 CSW

2.2.1 Overview

Mixed criticality systems (MCS) require a strong partition on hardware (HW) resources in order to maintain a known HW state avoiding unexpected errors or attack vectors to the system critical software components (SWC).

To ensure this partitioning the system must be modelled and implemented with each component assigned its criticality depending on the system requirements. This allows an early detection of resource sharing conflicts and a correct planning on the access each critical/non-critical component should support, as well as the type of constrains each of them requires.

Furthermore, at run time, the isolation between software and hardware components of different criticalities is enforced through a hardware mechanism already described in annex of D4.3.

Additionally, some events happening during the system execution may require hardware components (HWC) to change their criticality level due to being required by higher criticality SWC. This can be interpreted as a change in the execution mode of the overall system. In the use case context, it consists in the detection of a collision between the car and an obstacle. Whilst the execution was initially considered as being in a normal mode of operation, the event triggers changes throughout the system and the system must be reconfigured. In CONCERTO, this will be supported by the introduction of modes of operation in the model, thereby allowing the designer to model distinct criticality classifications and connections in each mode (see D2.4). In each operation mode, the task properties and constraints, resource allocations and SWC partitioning may be different.

2.2.1.1 Partitioning support

For HW to SW component assignments, there are four types of connections:

1. Fixed criticality level (FCL): These HW devices do not change their criticality throughout the execution. They can only be assigned to SWC of the same criticality level.

2. Upgradable criticality level (UCL): The upgradable devices can have their criticality increase due to an event (e.g., a shock between the car and an obstacle) where a higher criticality SWC requires that resource. They are initially assigned to a lower criticality level and can only be connected to SWC of that same level. When the overall system mode changes that resource criticality increases and the assigned SWC changes.

3. Virtual criticality level (VCL): Resources that are required by both critical and non-critical SWC fall into this category. They are managed by the critical

runtime environment (RTE) but available as a virtual device to the non-critical RTE through a secure communication layer.

4. Split Criticality level (SCL): These are HWC that have the capability of isolating their resources and are aware of the SWC criticality level. Split criticality level components must be composed by two or more sub-components (one for each level available). For example the system memory can be securely shared between two distinct criticality RTE with concrete space partitioning.

2.2.2 Challenges

For the purpose of the automotive use case proposed by CSW, CONCERTO should support the modelling of a mixed-criticality infotainment system with the following properties:

1. Multi-core with core affinities respecting the restrictions presented in the previous section;
2. Mixed-Criticality with two (or more) levels of criticality;
3. Distinct system modes of operation for differentiating component attributes and connections constraints for each mode;
4. Events to switch between modes of operation.

3. DEVELOPMENT

3.1 AIRBUS GROUP

The extensions to CONCERTO to support partitions, as stated in section 2.1, address three main goals.

Modeling the concept of IMA partitions

Since IMA partitions are functional concerns we envisage that modeling a *functional partition* means creating a parent component that embeds other child components. This idea requires the support for hierarchical components. The concept of functional partition is also resumed in D2.3.

Following what was stated in D2.2 we propose to develop hierarchical components that support both top-down and bottom-up development processes. In this context, supporting functional partitions, along with hierarchical components, requires both the extension of the methodology and the modelling language. Section 3.1.1 and 3.1.2 describes, respectively, the proposed solutions.

Moreover, as stated in D4.2, supporting IMA partitions can be seen as supporting programs of different criticality where, however, currently we do not envision an explicit criticality level associated to a partition.

Providing a timing analysis for the two-level scheduler and the related transformations

MAST [4], the model-based timing analysis tool used in CONCERTO has to be extended to support the two-level scheduler required for IMA. The extension of MAST is described in D4.5, section 5, here we focus on what are the impacts for the modeling language and the transformations. Section 3.1.3 describes the proposed solution.

Providing transformations for code generation

Code generation implies transformations that instantiates predefined containers and binds them together with connectors according to the system design [1]. The process is similar to what was achieved in CHESS. Containers are already described in D4.5, section 6. In this deliverable we focus on the transformations, which are described in section 3.1.4.

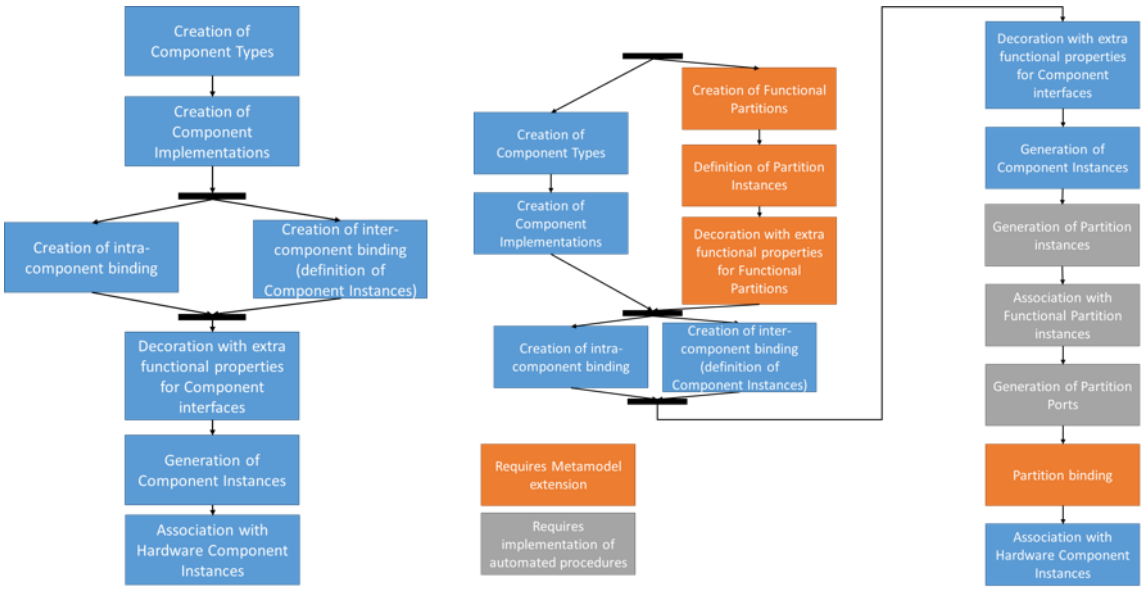

Figure 1 Design steps focused on functional and extra-functional concerns. The definition of the deployment concerns are not considered in its entirety, i.e. creation of hardware architecture is necessary for the last step. Left hand side: design steps inherited from CHESS. Right hand side: design steps extension proposal to support functional partitions.

3.1.1 Methodology extensions

From the methodology standpoint (Figure 1), the user should be able to define partitions either prior to the components they are going to include (in a top-down fashion) or after the constituent components are defined (in a bottom-up fashion). The definition of the components must include the definition of the component types, the component implementations and the component bindings, so that the component instances can be generated by the automated procedure of the tool. Once the partitions and the component instances are generated, the user is able to associate each partition with the desired group of instances in the functional view. This association is constrained by the

rule that an instance must pertain to one and only one partition. The tool should be able to ease this process by letting the user define the association in a declarative way and then providing for the actual creation of the entailed model entities. After the association is successful, the tool is able to generate the behaviour of the partition by promoting operations of the component implementations that are not yet bound to other component implementations.

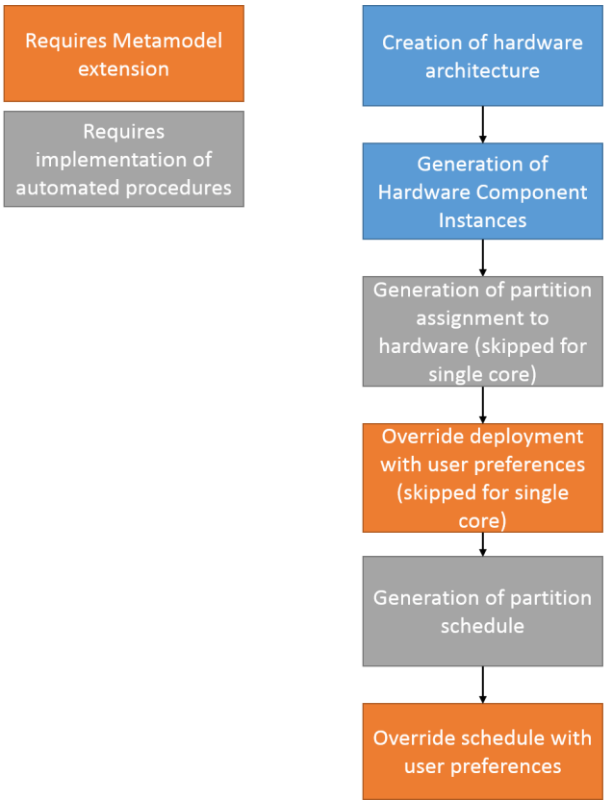

Figure 2 Proposed design steps for functional partition focused on deployment concern.

The deployment of partitions requires a major extension with respect to CHESS (Figure 2). In fact, while in CHESS the deployment was limited to the definition of the hardware architecture and to allocation of software instances to hardware components of the architecture, in CONCERTO the partitions need also to be assigned to a schedule defining when and for how long they can execute. An automated procedure to generate partition assignment will be implemented to both ease the burden of manual allocation of partition to cores and, most importantly, to suggest an allocation that maximizes the system utilization. With similar goal, an automated procedure to generate the schedule inside a core will be implemented. In both cases the user can override the tool suggestions as desired.

To summarize, the extensions to the methodology place the following challenges:

- Generation of partitions instances: this automated procedure will be similar to the one already implemented which generates component instances.

- Association with functional partitions instances: the way partition instances and component instances are associated together is similar to the association of software

components to hardware components in the deployment view. The challenge in CONCERTO is the automated generation of such modeling elements based on user choices by means of a "partition assignment dialog" (Figure 3).

- Generation of partitions ports: this automated procedure should collect all the component ports that are not bound with other ports and create the corresponding delegation ports in the partition the components are assigned to. The actual challenge is the integration of the concept of delegation in the CHESS component model, that is extending the component model to be hierarchical, rather than the implementation of the procedure itself.

- Generation of the assignment between partitions and hardware for multicore: in order to suggest an assignment that maximizes the utilization of the computation resources a sort of optimization must be implemented. Moreover it is possible that this procedure depends directly on the scheduling tool described in the follows. A prerequisite is that the modeling language should support the allocation of software entities to cores of a multicore processor.

- Generation of partition schedule per core: a scheduling tool must be implemented. This tool should take as input the user preference for the scheduling position of a partition and the dependencies among the partitions allocated on the same core.

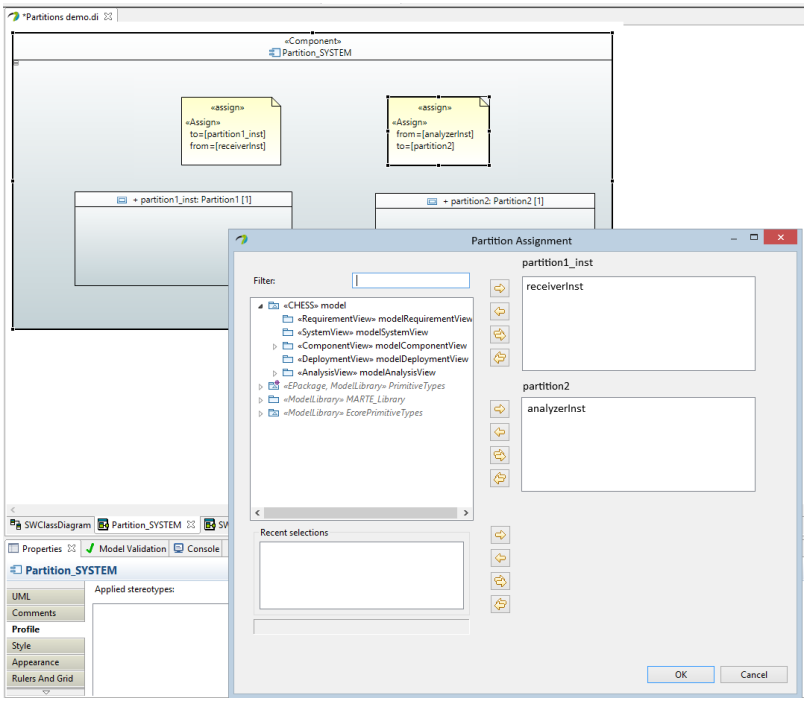

Figure 3 Partition assignment dialog proposal.

3.1.2 Metamodel extensions

From the Metamodel standpoint, the language should be able to represent a functional partition and to associate component instances and memory partitions to functional partitions. Here we present only PIM-level extensions since PSM-level ones needs further investigation, in particular they depends on what the analysis actually expects and how multicore deployment information is actually modelled. As stated in D2.4 section 5.3.4, the extensions will be provided first for a single-core processor then on a multicore processor, so it is acceptable that there is an undergoing investigation on how to represent multicore information at PIM-level and, most importantly, at PSM-level. A solution for representing PSM-level entities may be come from the CONTREX project [3] (https://contrex.offis.de/home/index.php/dissemination/deliverables, deliverable D2.1.1).

Currently there is not a proposal for modelling memory partitions, therefore the association to memory partitions will not be mentioned in the following.

A partition is represented by a UML Component and a <<FunctionalPartition>> stereotype. The stereotype should contain extra-functional properties of the partitions that are explained in section 3.1.3. The partition has to be created in the functional view while its properties has to be modified in the extra-functional view. The allocation of component instances to partitions is represented by the MARTE stereotype <<Assign>> already used in CHESS for assigning software components to hardware components.

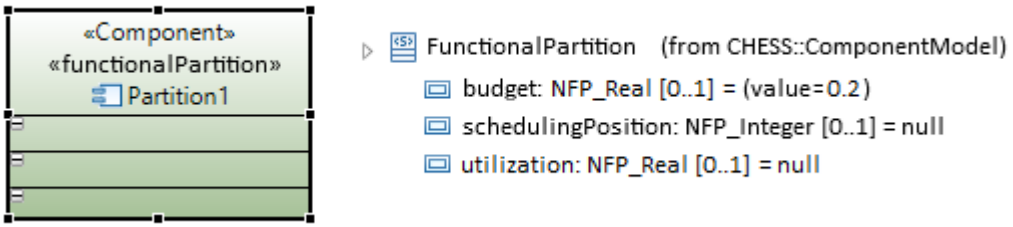

Figure 4 The proposed stereotype that represents the functional partitions and its properties.

The challenge to extend the CHESS ML entails the creation of the <<FunctionalPartition>> stereotype and the inclusion of the corresponding permissions in the extra-functional view to allow the editing of its properties. The extension needed for the <<Assign>> stereotype is the inclusion of the corresponding permission in the functional view, since currently this stereotype is used only in the deployment view.

3.1.3 Analysis

Supporting partitions in MAST entails the modification of the input and the output of MAST and thus it requires that the CONCERTO modeling language represents the information needed by the transformations to produce the correct input (*analysis phase*) and to store the back-propagated results of the output (*back-propagation phase*).

For the *analysis phase* the modeling language shall include for each partition the budget and the position in the scheduling.

The "budget" is mandatory and of user responsibility. It is a measure of the percentage of the processing resource utilization. This information, along with the WCET of the tasks of a partition – produced by the assignment of the component instances to the partitions – and the shared resources – produced from the connectors of the components -- are fed into the transformation. The latter calculates the values of the MAF and the actual values for the schedule on each core.

The "scheduling_Table" is the absolute ordering of the partition in the core assigned to it. It is calculated by the scheduling tool and can be overridden by the user.

For the *back-propagation phase* the modeling language shall include for each partition the utilization calculated by the analysis tool as the sum of the utilization of the tasks inside it. The utilization is therefore a read-only property and has a value only after the first analysis will be performed.

The challenge to support partition analysis is how to calculate the input of the analysis starting from the PIM-level information. This is a prerequisite for the actual implementation of the transformations. Once this is understood, it is possible to define the PSM-level information and to implement the transformations. Moreover, the notion of MAF, along with the current industrial practice, is based on single core. The definition of MAF for multicore is under investigation.

3.1.4 Code generation transformations

The challenge to support code generation for the avionics target is bounded to how to define PSM-level information needed to the transformation to produce code which conforms to TiCOS (see D4.5 for details). In particular it is necessary to understand how to represent multicore deployment information in the PSM prior to implementing the code generation transformation.

3.2 CSW

The main point of interest in CONCERTO is in the validation of a multi-core mixed-criticality platform at design time, and for that is required to ensure that all components are correctly defined and assigned on every operational mode.

Resource sharing is a critical step in this design as components criticality classification and connections interleave between system modes requiring a special attention in their validation.

As an example, the next figure depicts a system with two HWCs, a GPS and a 3G connection, and two ASIL classifications, namely ASIL-D for critical components and QM for non-critical components. In this example, the GPS can be accessed by both critical and non-critical SWCs through a safe communication channel managed by the real-time execution environment (see Annex 1 of D4.3 for more details). The 3G connection on the other hand is exclusively used by non-critical SWCs when the system is in the normal operational mode. Yet, as soon as a predefined event is detected (a shock between the car and an obstacle for instance), the system changes its operational mode from normal to critical, in which case non-critical tasks are killed and both the GPS and the 3G connection become accessible by ASIL-D SWCs only. That is, the 3G connection which was initially classified with the criticality QM in the normal operational mode is reclassified as ASIL-D in the critical operational mode. Hence, non-critical SWCs, if some should remain active, cannot access the 3G connection anymore. The main challenge of CONCERTO is to enable the modelling of such mixed criticality multi-mode systems and enforce the respect of the isolation/partitioning constraints on shared resources at design time for each and every operational mode.

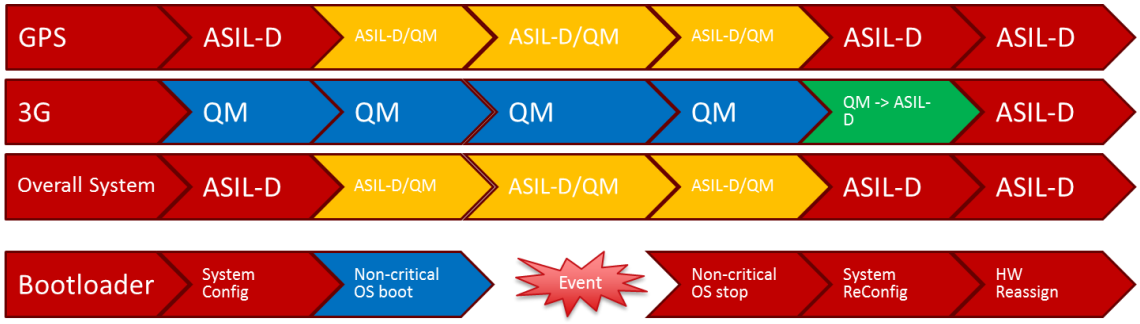

Figure 5 - HWC reclassification

Follows a more detailed explanation of each steps of the example depicted in Figure 5. Note that this example depicts the expected runtime behaviour of the eCall system that should be demonstrated in CSW's use case:

1. The system initially boots with all HWC managed by the critical RTE;

2. At step 2, the critical RTE configures all devices so that they can be used by non-critical, critical or both critical and non-critical SWCs as specified by the model;

3. Then, the non-critical execution environment (Android for instance) boots and starts managing its assigned HWC;

4. At step 4, an event happens that requires all HWC to be transferred under the control of the critical RTE;

5. All CPUs are reclassified as critical only and the non-critical OS together with the non-critical tasks are stopped;

6. The critical RTE reconfigures the necessary HWC;

7. Finally, the overall system mode is now in full ASIL-D with no non-critical SWC executing.

3.2.1 Methodology extensions

From a methodology viewpoint, the major difference with CHESS stands in the fact that the user should be able to model multiple operational modes as well as the transitions between those modes. In each mode the components may have different attributes, connections and assignments.

Furthermore, as a second improvement of the CHESS methodology, due to the mixed criticality aspect of the targeted application, the tool should enforce isolation between critical and non-critical components. Contrarily to the avionics use case, which uses the new functional partition stereotype at the PIM level to enforce both time and space partitioning at the PSM level, the properties of the execution platform developed by CSW (see D4.3) allows the application designer to focus exclusively on the space partitioning aspect at design time. This shall be achieved by defining the criticality attributes of each component in each operational mode. Then, based on their criticalities, the tool should automatically generate constraints on the allowed connections between components (see Section **Error! Reference source not found.**). Hence, the tool must only allow connections between components of the same criticality level. As an example, a critical SWC should never be allowed to access a non-critical HWC such as the 3G connection in the example of Figure 5. Such a connection would only be possible if the HWC was reclassified as being critical (as it is the case when the system switches to the critical operational mode in the previous example). If the user tried to connect a critical SWC to the 3G connection in the normal operational mode without changing the criticality level of the 3G, the tool should generate a design error. As a special case, a component without classification cannot be connected to any other component and should be deemed as inactive.

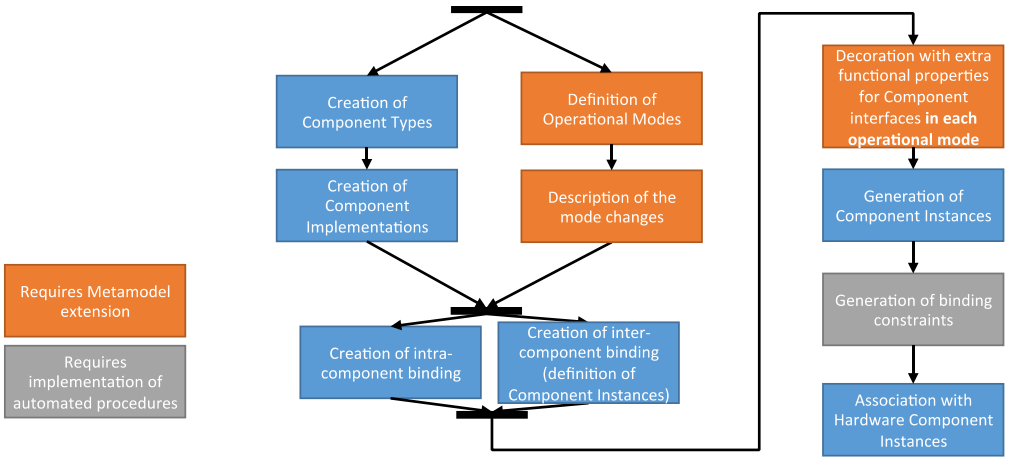

Figure 6: Design steps extension proposal to support operational modes and space partitioning. The design steps are focused on functional and extra-functional concerns. The definition of the deployment concerns are not considered in its entirety, i.e. creation of hardware architecture is necessary for the last step.

To summarize, the CHESS methodology should be extended with the following steps (see Figure 6):

- In parallel to the definition of the component types and implementations, one should define the operational modes of the system together with the possible transitions between those modes.
- The extra-functional properties of the components can then be defined for each and every mode.
- Based on the criticality attributes of the components, for each operational mode, constraints on the possible bindings between SWC and HWC should be generated by the tool.
- The user can finally bind the SWC with the HWC whilst respecting the generated constraints in each operational mode.

3.2.2 Metamodel extensions

In order to model CSW's use case hardware and software components, each HWC must at first have their type defined, this determines the possible changes to their attributes between execution modes. Both HW and SW components shall then be included in a normal mode where each component criticality classification and connection is set according to the system requirements. A second model shall then represent the system in a critical state where specific HWC can have their criticality reclassified and their connections changed.

3.2.2.1 System Operational Mode

The system operational mode can change between at least two modes: normal and critical. This change will affect UCL and VCL resources and their connections. This operational mode change will be modelled by a "modeBehaviour" state machine, each state describing one mode of operation. The mode transitions will be triggered by events

happening during the system execution. The support for "operational modes" is further described in D2.3.

3.2.2.2 Hardware Components Attributes

Each hardware component must have a new set of attributes:

1. ASIL classification;
2. Connection constrains to different ASIL;
3. Connection type as defined in section **Error! Reference source not found.**.

3.2.2.3 Software Components Attributes

Each service provided by a software component must have the following attributes:

1. ASIL classification;
2. One execution time budget, period, deadline and priority per operational mode supported at system level;
3. Core assignment that may change from one operational mode to the other.

As pictured in Figure 5, an event happening during the system execution can raise the ASIL classification of some HWCs but also stop the executions of some SWCs. This corresponds to a mode change (i.e., modification of the state in the modeBehaviour state machine used to model the transitions between operation mode). The variation of the attributes content when changing mode should be modelled following the MARTE specification as further discussed in Section 3.3 of D2.3.

This description of the attributes based on operational modes allows to model any variation in the system behaviour due to external or internal events changing the criticality of the system. For instance, following that approach, the modelling of tasks being killed when passing from the Normal to the Critical operational mode, as described in Figure 5 for example, could simply be modelled as setting the execution time budget of the associated SWC to 0.

3.2.2.4 CPU Core Attributes

Each core component must have the following attributes:

1. The ASIL classification for the SWC assigned to it;
2. Upgradable ASIL classification (i.e., one classification per operational mode).

The following restrictions must be enforced:

1. Software components can be assigned only to cores with the same level of criticality;
2. Cores with no classification do not accept any software component.

3.3 COVERAGE OF CONCERTO DERIVED REQUIREMENTS

This section summarizes the current coverage of the CONCERTO derived requirements concerning the support for the aforementioned use cases.
The column "Comments" is used to trace what is discussed in this document.

Req. No.	Overall Priority	Derived Requirement	Partners involved	Comments
R1.1	SHALL	CONCERTO language shall support the definition of modes of operation for software components. The set of components that operates under a specific mode of operation constitutes a scenario.	UPD,INT, ISEP	See D2.3.
R1.2	SHALL	The analysis tools shall be able to compute the response time based on the user provided scenarios.	ISEP, UPD	See D4.5.
R1.3	SHALL	The model transformations shall support modes of operation.	UPD, INT	See D4.5.
R1.4	SHALL	The code generation shall generate containers that are able to switch mode of operation.	UPD	See D4.5.
R3.1	SHALL	CONCERTO model validation shall raise precise messages when the model contains syntactic errors.	UPD	Not yet covered.
R3.2	SHALL	Model transformations shall issue precise error messages when a required property is missing or incorrect	UPD	Not yet covered.
R6.1	SHALL	The modeling language shall support the definition of different types of schedulers and their parameters. In particular there shall be support for multicore schedulers for SMP and heterogeneous systems.	INT, ISEP	Among the different types of schedulers there is also the scheduler for IMA, i.e a two-level scheduler. A solution however is not yet envisaged (it may come from the CONTREX project).
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 then be instantiated to the specific domain of interest.	INT, EADS	Not yet covered. It should be addressed when the PSM-level entities will be defined.
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.	UPD, EADS	See the comment for R26.1

R32.1	SHALL	The modeling language shall support the definition of different levels of criticality	INT, UPD, CSW	Covered in section 3.1.2 (as stated in the introduction of 3.1) and 3.2.2
R32.2	SHALL	The analysis tools shall be able to analyse mixed-criticality systems	ISEP, CSW	See Sections 5.2 and 5.5 in D4.5
R32.3	SHALL	The runtime environment shall support non-critical/non-trusted, as well safety-critical software components	CSW	See Annex of D4.3
R32.4	SHALL	The runtime environment shall support mixed-criticality systems under the assumption that components execute in a pre-determined hardware resource partition within the same level of criticality	CSW	See Annex of D4.3
R33	SHALL	All system components (hardware and software) shall have their criticality specified as an attribute.		Not currently envisioned for the Airbus Group use case. Covered in 3.2.1 for CSW use case.
R42.1	SHOULD	The modeling language should define communication attributes for software and hardware components.	INT	Not yet covered.
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	INT	Not yet covered.
R69	SHALL	CONCERTO shall support means for time partitioning between critical and non-critical tasks through CPU budgets		Covered in section 3.1.2 (extra-functional properties of the partition).
R71.1	SHOULD	The modeling language should allow to specify core reservation for software components.	INT	For Airbus Group use case, section 3.1.1 discusses the partition assignement automated procedure that is indirectly related to the "assignement process" emanated from R73 in D4.1 which is also related to R71 and R72 and R55 requirements. For CSW use case, section 3.2.2 covers R71 and related requirements.
R71.2	SHOULD	Core reservation should be taken into account in model transformations	UPD	See the comment for R71.1.
R71.3	SHOULD	Core reservation should be taken into account in the assignment process (R73.3)	MDH,ISEP	See the comment for R71.1.
R72.1	SHOULD	The modeling language should allow to specify processor affinity for (critical) software components.	INT	See the comment for R71.1.

R72.2	SHOULD	Processor affinity should be taken into account in model transformations	INT	See the comment for R71.1
R72.3	SHOULD	Processor affinity should be taken into account in the assignment process (R73.3)	UPD, ISEP	See the comment for R71.1
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 software 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.	INT, MDH, CSW, UPD	See the comment for R71.1
R73.2 (R55)	SHALL	The CONCERTO design space shall assist the user to handle multicore deployment by enabling specific model constraints for hardware components	INT, CSW	See the comment for R71.1.

4. CONCLUSION

This document presents the proposed extensions to the CONCERTO Methodology and Toolset to cover the CONCERTO requirements for Airbus Group use case for the Avionic domain and the CRITICAL SOFTWARE use case for the Automotive domain.

Regarding the Airbus Group use case, the solution proposed for extending the methodology and modelling IMA partitions are in a rather advanced status. Conversely, the solutions for the transformations from/to the analysis and the code generation need more investigation. There is currently an ongoing study on how to define PSM-level entities to represent multicore deployment and to represent the input and the output of the analysis. This study is a prerequisite to the implementation of the transformations.

Regarding the CRITICAL SOFTWARE use case the solution proposed for mixed-criticality infotainment systems relies essentially on two main aspects: (1) the enforcement of connection constraints between SWC and HWC based on their respective criticality levels and (2) the modelling of modes of operation. As discussed in D2.3, the modelling of operational modes and the transitions between those should be implemented following the MARTE specification. The development of the runtime environment and execution platform is well underway as already presented in D4.3.

5. REFERENCES

[1] M. Panunzio and T. Vardanega, “Charting the Evolution of the Ada Ravenscar Code Archetypes”, in ACM SIGAda Ada Letters, 2013.
[2] R.T.C. for Aeronautics, “Integrated Modular Avionics (IMA) development guidance and certification considerations”, 2005.
[3] https://contrex.offis.de/home/, CONTREX project homepage
[4] http://mast.unican.es/, 2014. MAST homepage.