ECSEL Joint Undertaking

Electronic Components and Systems for European Leadership

AQUAS SAFETY SECURITY PERFORMANCE

Deliverable 4.1

Report on Co-Engineering Process Support

ECSEL Joint Undertaking

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737475. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Spain, France, United Kingdom, Austria, Italy, Czech Republic, Germany.

DISSEMINATION LEVEL		
X	**PU**	Public
	CO	Confidential, only for members of the consortium (including the Commission Services)

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	AQUAS
Project Full Name:	Aggregated Quality Assurance in Systems
Grant Agreement No.:	737475
Programme	ICT-1: Cyber-Physical-Systems
Instrument:	Research & innovation action
Start date of project:	01.05.2017
Duration:	36 months
Deliverable No.:	D4.1
Document name:	Report on co-engineering process support
Work Package	WP4
Associated Task	Task 4.1
Nature [1]	R
Dissemination Level [2]	PU
Version:	0.1
Actual Submission Date:	24-05-2018
Contractual Submission Date	30-04-2018
Editor: Institution: E-mail:	Christian Fuss / Mario Winkler ANSYS medini Technologies AG christian.fuss@ansys.com mario.winkler@ansys.com

[1] **R**=Report, **DEC**= Websites, patents filling, etc., **O**=Other

[2] **PU**=Public, **CO**=Confidential, only for members of the consortium (including the Commission Services)

Change Control

Document History

Version	Date	Change History	Author(s)	Organisation(s)
0.1	06.09.2017	Initial draft of document structure	C. Fuss	AMT
0.2	18.01.2018	Revised document structure and resorted contributions	C. Fuss	AMT
1.0	24.5.2018	finalized	M. Winkler	AMT

Distribution List

Date	Issue	Group
11.09.2017	0.1 Call for inputs	design.tooling@aquas-project.eu all@aquas-project.eu
26.04.2018	0.2 Call for additional inputs and modifications	design.tooling@aquas-project.eu all@aquas-project.eu
24.05.2018	1.0 Final version	ECSEL JU all@aquas-project.eu

Table of Contents

Executive Summary

This deliverable is the result of AQUAS task 4.1. Its roles towards supporting work in AQUAS are:

- specifying required functionalities enabling tool support of co-engineering processes according to the AQUAS methodology as it is currently understood in respect to the requirements, it will be adapted as the project progresses;
- starting the formalization of the methodology in a tool-based process model by using the Software & Systems Process Engineering Meta-Model (SPEM);
- explaining the adaptations of this methodology for the different use cases;
- outlining the partner's tools to support this methodology and especially explaining the planned extension to support the analysis of safety, security and performance properties and the foreseen interactions amongst the tools.

1 Introduction [AMT]

The work documented in this deliverable is based on the methodology on safety, security and performance processes that is elaborated in WP3. The AQUAS methodology will emerge from the use cases actual processes, as derived from the application of the different standards, from the emerging definitions of co-engineering and interaction points and from the application and integration of tools to support the different processes.

This report specifies required functionalities enabling tool support of co-engineering processes according to this approach. Functionalities will be described for common and global factors as well as their tool-specific implementation. Functionalities comprise tracking process progress, data and artefacts, and interactions points where cross-domain and cross disciplinary analyses must be executed to realize safety-security-performance analysis, supporting system design space exploration and trade-offs. Activities will also address ways for requirements to be linked/transferred to different tools allowing the requirement continuities in the safety, security and performance domains.

In this process, the information that should be collected in the design models for enabling this co-engineering support will be considered. Beyond models and formats, data repositories and management will also be addressed.

Structure of this Document

This document starts with a description of the tool-based process and methodology description in **Chapter 2**, contributing to Objective O6 to provide a well-defined process with clear descriptions of interactions among continuous engineering activities.

Chapter 3 is structured along the lines of the use cases and describes the actual process planned for each use case, with a focus on interaction and tools. Using a formalized model with tools attached to tasks and tasks composed to more complex activities, allows to identify reliably all interaction between the different technologies and tools, fulfilling Objective O1.

The tools used in the use case are described with their data models and interfaces available for interaction throughout the process are presented in **Chapter 4**. This clear definition of data models and interactions can reduce the unwanted repletion, supporting Objective O7.

The document closes with a **Conclusion** and a list of **References**.

2 Process Model

One of the core goals of the AQUAS project is to develop a methodology for co-engineering in the product life cycle that supports qualitative and especially quantitative techniques for the analysis and assessment of safety, security and performance properties, both in separate and integrated ways. The main tasks contributing to this goal are carried out within WP3 (see deliverables D3.1).

To make this methodology accessible and allow reuse of process building blocks, it is planned to provide a tool-based process description for co-engineering processes, that enables the definition of the individual process including the work products, roles and activities but also allows to define the required synchronization among these different processes. This task is carried out within WP4 but is based on the outcomes of tasks of WP3 so far. The model shall be maintained and enhanced, as WP3 progresses.

The following sections will give an overview on the approach and outline the content.

2.1 Meta-Model

The process description is formalized in SPEM (Software & Systems Process Engineering Meta-Model) using the tool EPF Composer. The outcome is an EPF model and a generated website describing all parts of the process and allowing navigation through the knowledge base.

This EPF model of the AQUAS methodology is specified as an initial version and is considered as a living document. The final definition of the process phases and especially of the interaction points will be adopted according to the results that will be developed throughout the AQUAS project within the use cases.

"The Software and Systems Process Engineering Meta-model (SPEM) is a process engineering meta-model as well as conceptual framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes. An implementation of this meta-model would be targeted at process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for their development organizations or individual projects." [OMG-SPEM]

The idea of SPEM is to separate reusable method content from the described software processes which can be seen in below figure. Because of this the processes that are contained are reusable i.e. role, work products, tasks etc. This approach shall be employed for the AQUAS methodology, by describing generic content and process patterns, that can be tailored to the specific project. The viability of this approach shall be demonstrated by showcasing tailored processes for the AQUAS use cases.

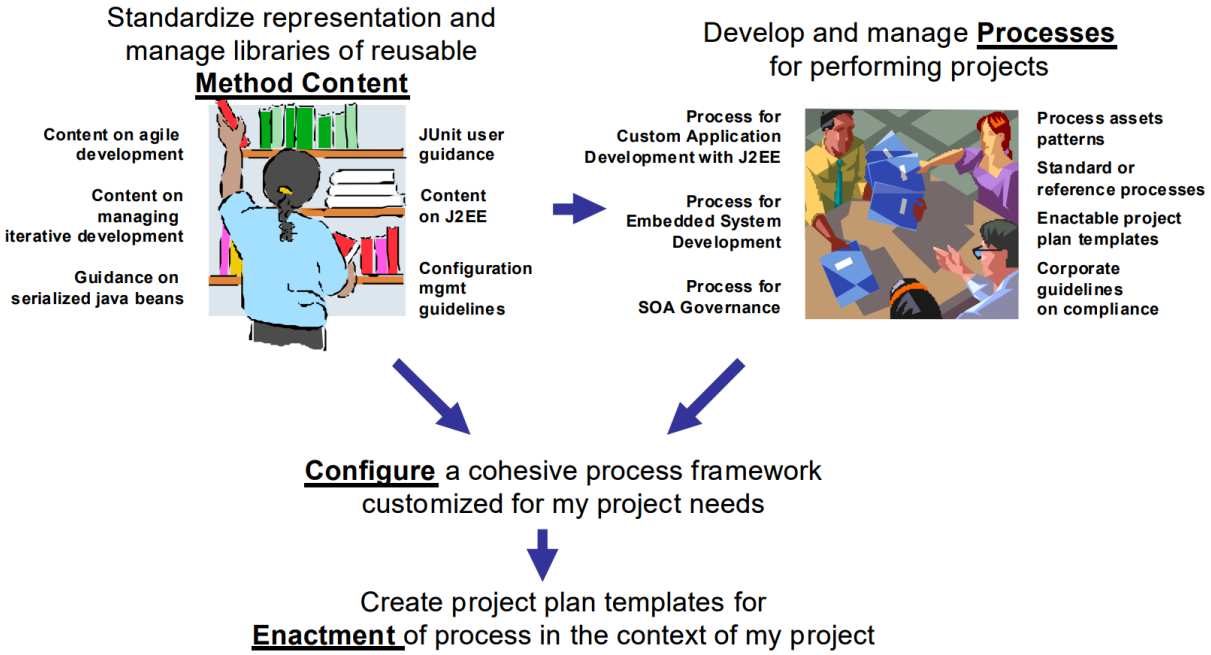

Figure: Separation of reusable method content, assembly in process patterns and configuration [OMG-SPEM]

As reusable method content, the AQUAS methodology specifies tasks, that are performed by roles. A task can have input and output work products. A task can be supported by a guidance, e.g. a tool, user manual or a concept.

- **Task**. A unit of work a role may be asked to perform.
- **Role**. A definition of the behavior and responsibilities of an individual, or a set of individuals working together as a team.
- **Work Product**. A work product is a content element that represents anything used, produced, or modified by a task.
- **Guidance**. Guidance describes proven advice for accomplishing a goal. It generalizes all forms of content whose primary purpose is to provide explanations about other elements. Guidance being itself a content element, it is possible to associate guidance to other guidance.

Furthermore, the AQUAS methodology provides building blocks of the bigger process in the form of capability patterns made up of activities, which group task instances.

- **Capability Pattern**. A special process that describes a reusable cluster of activity. Capability patterns express and communicate process knowledge for a key area of interest such as a discipline and can be directly used by practitioners to guide their work.
- **Activity**. An activity is something that one or more roles do. It is a breakdown element which supports the nesting and logical grouping of related process elements such as descriptors (task instances) and sub-activities, thus forming breakdown structures.

The above building blocks are finally used to provide delivery processes for the use cases as examples for tailoring the AQUAS methodology and as proof of concept.

- **Delivery Process**. A delivery process is a special process describing a complete and integrated approach for performing a specific project type. It provides a complete lifecycle model that has been detailed by sequencing method content in breakdown structures.

For publication, all elements in an EPF model must be structured in categories, that build up a tree of the knowledge graph of the entities. The following kinds of categories are distinguished in EPF Composer:

- **Discipline**. A collection of related tasks that define a major 'area of concern'.
- **Domain**. An area of knowledge or activity characterized by a family of related values. A specific problem category that is characterized by a body of knowledge, activities, and behaviours. A hierarchy that groups related work products.
- **Work Product Kinds**. Standard category that represents a grouping of related work products which, in contrast to domain, is more presentation oriented (like models, specifications, plans, and so on).
- **Role Sets**. Used to group roles with certain commonalities together.
- **Tools**. A standard category used as a container for tool mentors. It can also provide general descriptions of the tool and its general capabilities.
- **Custom**. Used to categorize content based on user criteria. One important use is for constructing views for publication.

2.2 Overview

The safety life cycle model of the IEC 61508 standard provides an initial contribution to the definition of the AQUAS methodology. The SESAMO project proved that this life cycle model can well be extended to further quality criteria, like security or performance. The IEC 61508 life cycle is stage-based, below figure shows the stages.

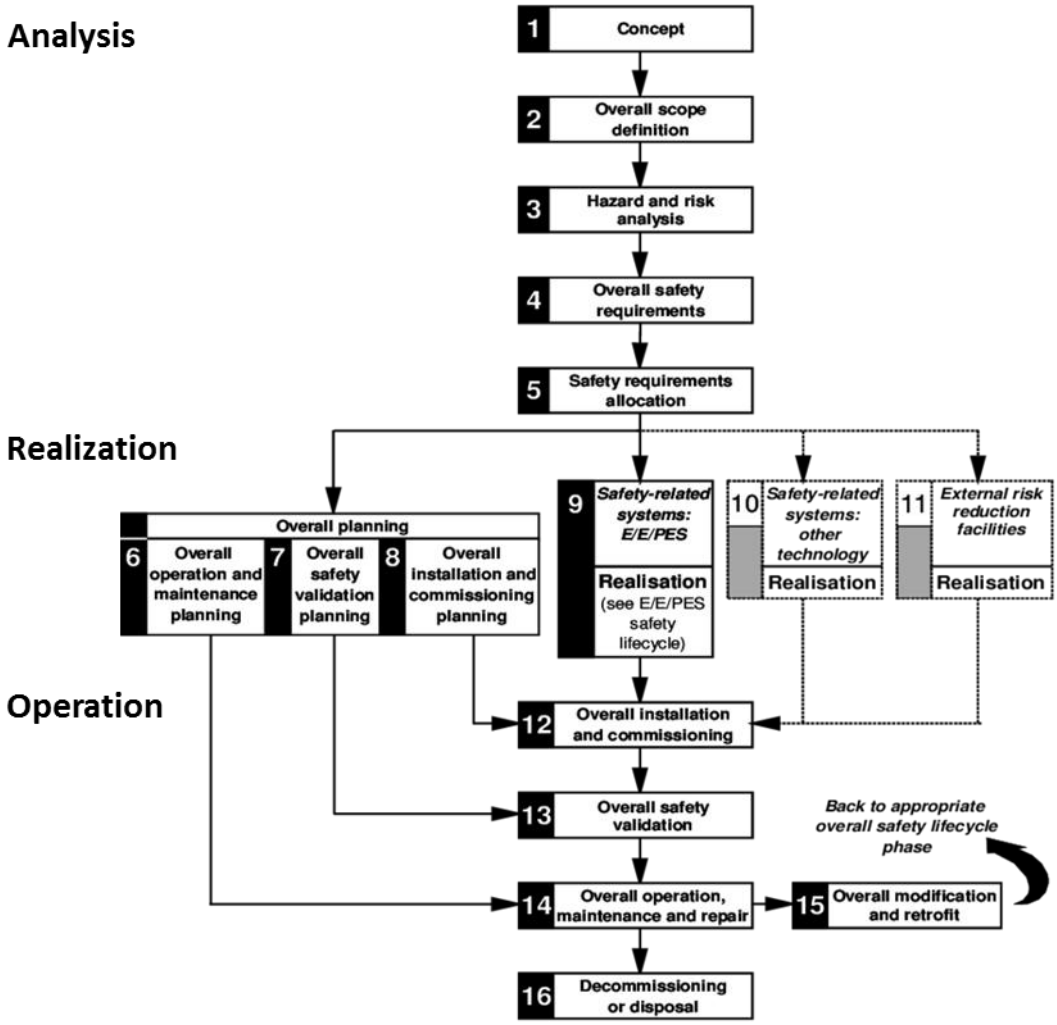

Figure: Phases of the IEC 61508 life cycle model. [IEC 61508]

Concept. The goal here is to gain sufficient understanding of the equipment under control (EUC) and its environment.

Overall Scope Definition. In this phase of the safety life-cycle, the boundaries and the relation between EUC and EUC control system (ECS) are defined, preliminary hazards are identified and the scope of the hazard and risk analysis are defined.

Hazard and Risk Analysis. Hazards at all reasonably foreseeable circumstances for EUC and ECS are determined.

Overall Safety Requirements. Overall safety function requirements and safety integrity requirements are defined.

Overall Safety Requirements Allocation. Safety functions are allocated to safety-related systems (SRS) and other risk reduction measures (ORRM).

System Safety Requirements Specification. System safety requirements definition (safety function requirements and system safety integrity requirements) to reach functional safety.

Safety-Related Systems Realization. Implementation of the SRS according to system safety-, system safety function-, and system safety integrity requirements.

As initial approach to the AQUAS Methodology, the analysis, realization and operation stages defined in the IEC 61508 standard have mapped to a number of product life cycle areas as described in the following figure:

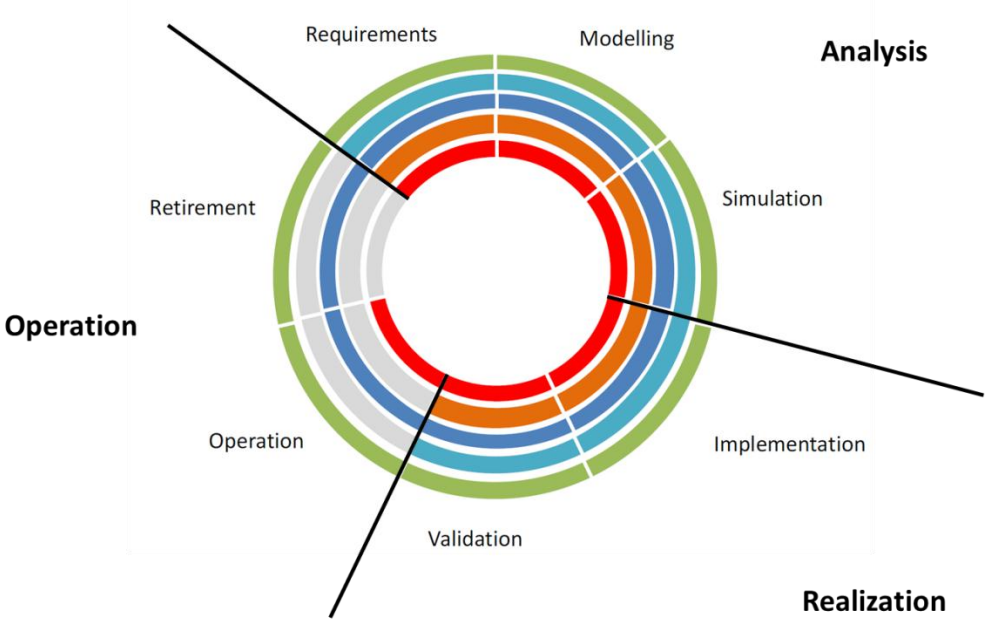

Figure: Stages of the AQUAS methodology

Requirements, modelling and simulation (or formal analyses) are the pre-development activity areas that constitutes the main focus of AQUAS and are expected to support also further activities during realization and operation.

This structure is modelled as Capability Patterns in the EPF model as depicted in the following figure.

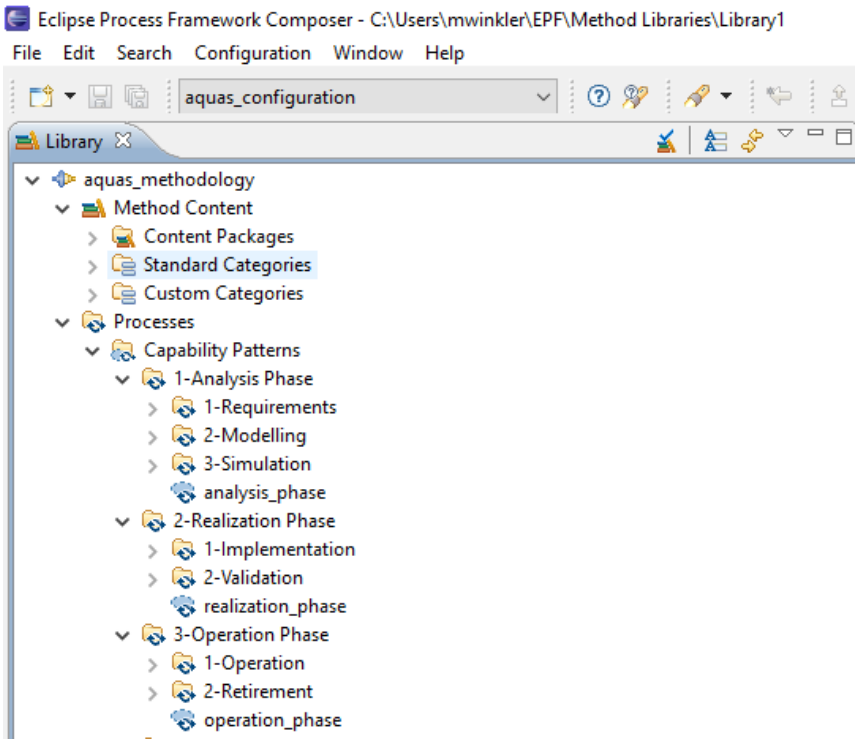

Figure: Stages modelled in the EPF Composer

Every phase in itself is structured into sub-processes that contain a set of tasks. The following figure shows this approach by using the Analysis Phase as an example.

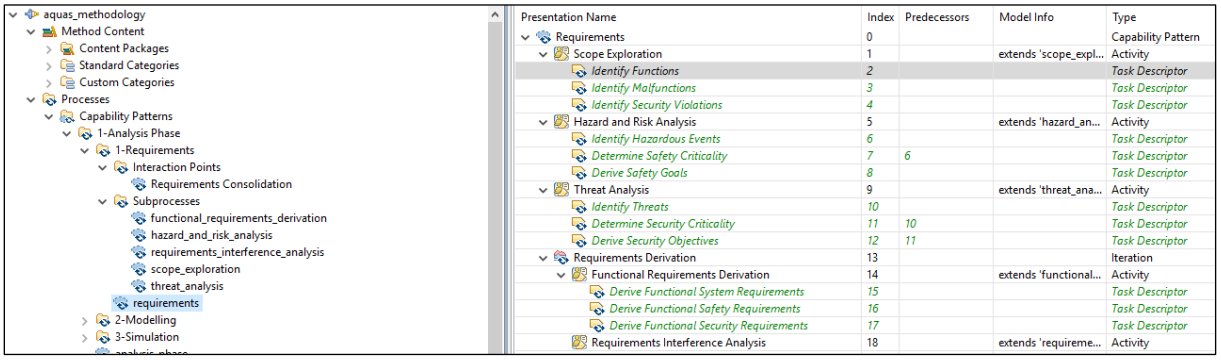

Figure: Sub-processes modelled in the EPF Composer

The sub-processes include:

Scope Exploration. This is to identify the functions of your system, to identify potential malfunctioning behaviour and to explore possible violation scenarios. All these activities are modelled as Tasks in the EPF model.

Hazard and Risk Analysis. Here the malfunctioning behaviour is related to a relevant set of operational situations including environmental conditions and operation modes in order to determine the criticality of safety violations. Subsequently safety goals have to be derived as top-level requirements to mitigate the risk. Some aspects of performance must also be taken into account, e.g. the violation of real-time conditions.

Threat Analysis. Similar activities as in the Hazard and Risk Analysis are performed in respect to security. This includes the identification of potential threats, the determination of their criticality and the definition of security objectives that initiates the design to a secure system. The identified objectives have to be analysed in respect to their performance implications.

Requirements Derivation. Starting with the safety goals and the security objectives functional requirements for both perspectives are derived. This also includes an interference analysis defined as an Interaction Point.

Interaction Points

With its focus on co-engineering, the AQUAS methodology pays particular attention to the interaction points between the engineering of the different quality aspects, throughout the whole process. Deliverable D3.1 motivates the introduction of interaction points and describes them as both activity as well as a point in the product life cycle at which it occurs. It includes the exchange of information between experts in the different domains (e.g. safety or security). The analysis methods applied in that interaction have to be combined in order to assess various measures of interest for alternative design options. As a result decisions and recommendations have to be produced by the various experts that provide solutions for identified trade-offs between desirable properties.

Thus in the EPF Model, interaction points are modelled as capability patterns and instantiated in delivery processes or other capability patterns as activities. An own discipline called “Interaction Points” has been created for collecting all interaction points as reference workflows. The globally identified interaction points are outlined within the outline of each phase below.

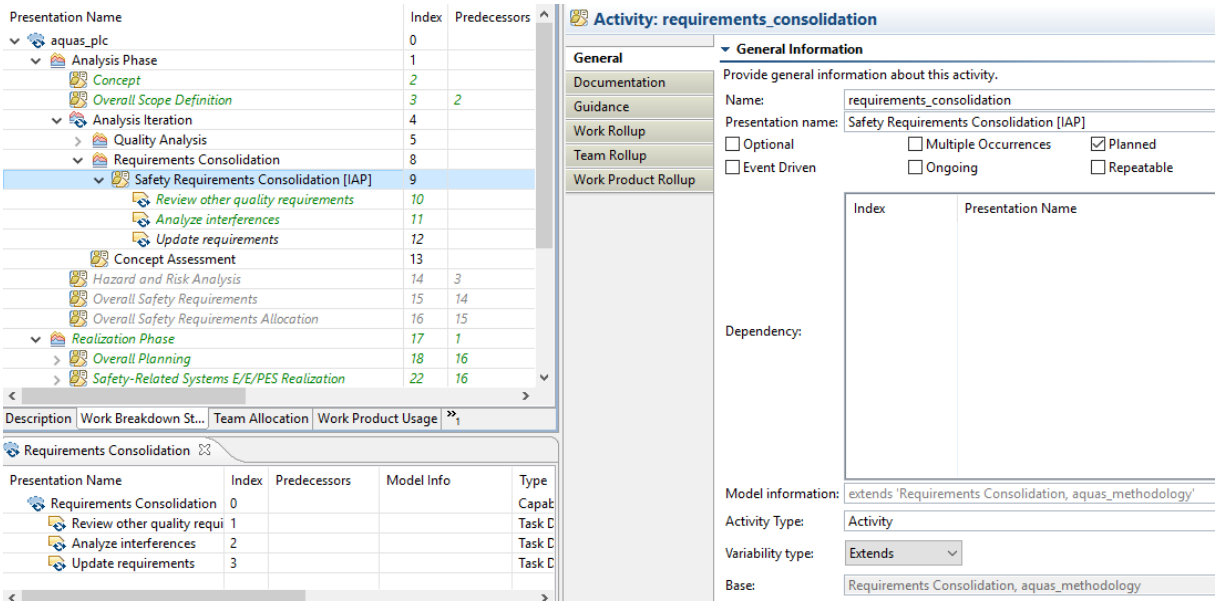

Figure: Modelling of interaction points as capability patterns and process usage as activities

The picture above reflects the considerations that were undertaken in the deliverable D3.1 chapter 4. Here it is formulated that co-engineering adds a new process step as interactions whereby they can be implemented either as reviews and discussions between experts or more formalised by tool supported analysis. Therefore, for instance, the interaction point “Safety Requirements Consolidation” shown in the figure is defined as a step in the Analysis Phase that requires the involved parties to review other quality requirements, to analyse their interference in respect to SSP and to update the requirements according to the findings. This approach of defining interaction points can and will be applied to other stages in the PLC.

3 Use Cases

3.1 UC1 – Air Traffic Management [ISYS]

The AQUAS ATM use case focuses on offering innovative situational awareness services to UAVs operating in Very Low Level scenarios (city environments, mountainous terrain or areas covered by vegetation). In these cases existing surveillance technologies are often limited due to radio signal propagation issues or lack of infrastructure, resulting in air traffic that could remain momentarily or permanently hidden for the UAV pilot depending on mission conditions.

The ATM use case intends to apply alternative communication technologies, such as LTE, to facilitate the exchange of surveillance related data among small UAVs in VLL environments, and leverage flight information gathered from governmental and trans-national agencies (e.g. Eurocontrol) to enable detecting hidden air traffic surrounding UAVs.

Similar performance targets as those applying to techniques currently in use for air traffic monitoring in terms of latency and accuracy are aimed. Fulfilling such requirements while coping at the same time with security and safety constraints in embedded computing platforms that are designed for small UAVs (thus subject to additional limitations, for example, in terms of battery power consumption) is a challenge that would benefit from adopting a co-engineering approach from the first development stages. Thus AQUAS methods and tools will be applied to the development of the UTM service elements across the major demonstrator components.

Next figure shows the overall UTM demonstrator architecture with major sub-systems involved and communication links among them.

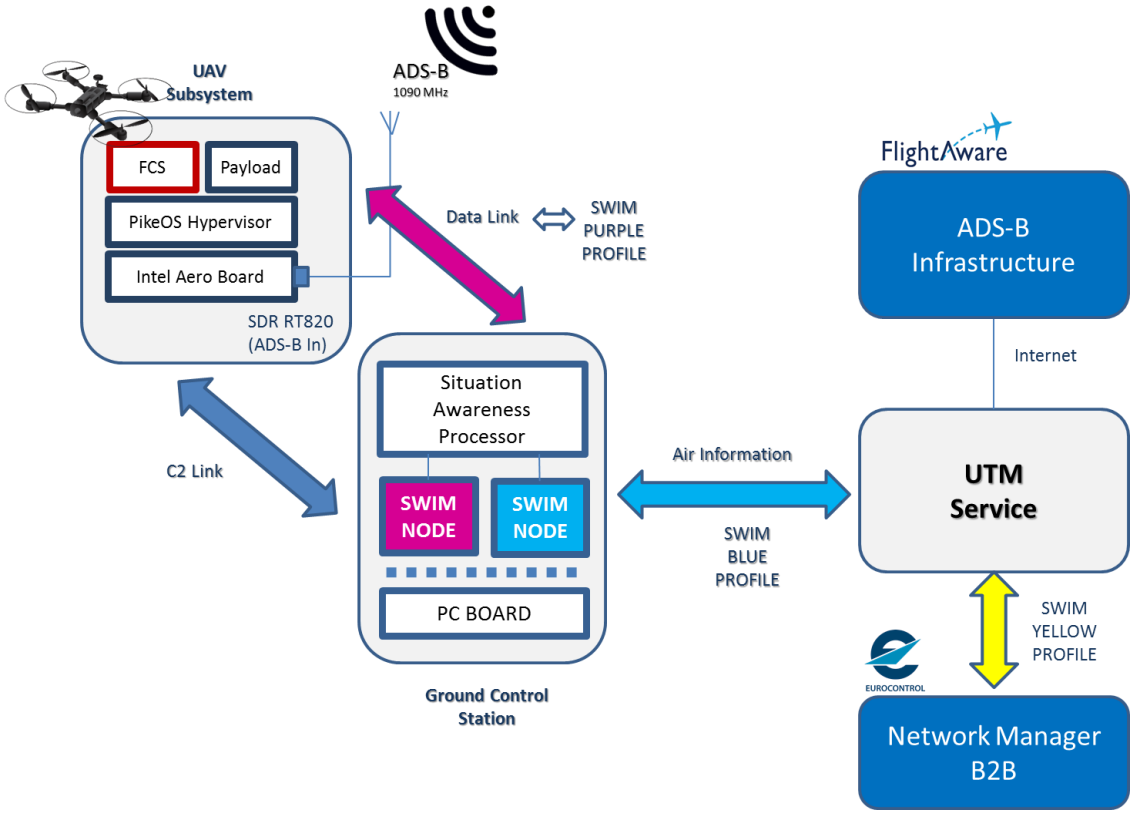

Figure: UTM Use Case Demonstrator Architecture.

Overall, the ATM use case demonstrator consists of an airborne UAV platform, a remote control station and a server integrated in the ADS-B (Automatic Dependant Surveillance-Broadcast) ground infrastructure. Interaction with two external systems is also planned: the FlightAware ADS-B ground infrastructure and Eurocontrol Network Manager Business to Business (NM B2B) services, which will be providing information on current aircraft positions and flight plans respectively.

Tools supporting SSP co-engineering within the AQUAS Product Life Cycle, from requirement specification and analysis to system modelling and simulation, implementation and validation will be showcased. The following figure summarises the tools applicable at each phase and the expected use of outputs among tools participating in the tool chain.

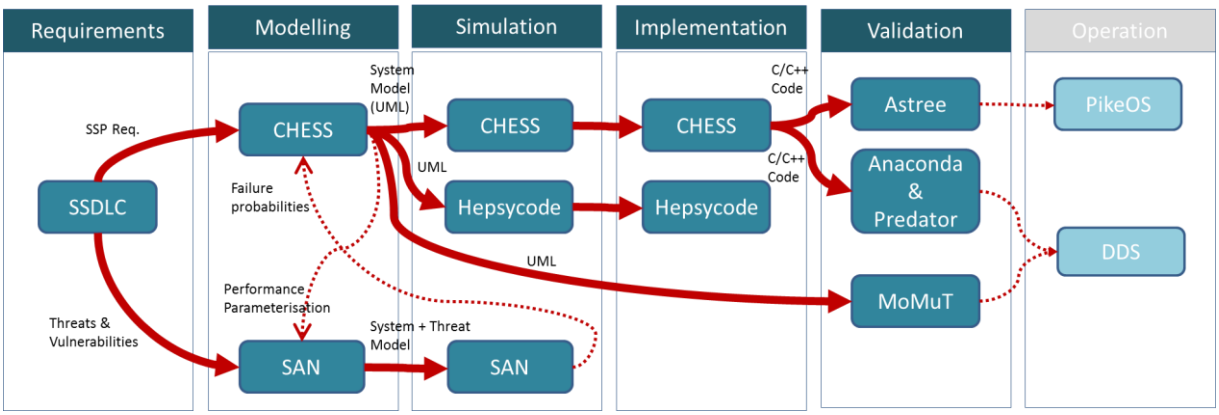

Figure: Preliminary tool chain envisaged to support co-engineering within the ATM use case.

In particular, **TrustPort** plans to define a set of security requirements using the SSLDC tool. For further evaluation of Security-Performance trade-offs the TTool is being considered (in cooperation with Telecom ParisTech). Based on inputs from SSDLC, TTool permits the evaluation of the performance in two different scenarios: one with no security and one with security mechanisms. Also, the evaluation of different security mechanisms/algorithms/methods could be evaluated. For example, without taking into account penalties of the hardware platform. Outputs from the SSDLC tool will be integrated in the form of SSP requirements within CHESS and SAN models.

Then, **INTECS** proposes the CHESS tool to support the modelling of the ATM UC1 system architecture, down to the software level. The CHESS Dependability profile, embedded in the CHESS modelling language, can be used to enrich the system and software architecture model with information about error model of the system and software components, to finally enable dependability analysis, like the failure propagation analysis.

INTECS plans to investigate the usage of the aforementioned model to automatically feed the quantitative analysis proposed by City to support the co-analysis phases. Also we are investigating an extension of the CHESS dependability profile for the modelling of security threats; in this regard we are collaborating with TrustPort to understand how the information related to security requirements derived with the Secure Software Development Life-cycle (SDLC) tool can be used within CHESS.

Finally INTECS will investigate the decoration of the software architecture components with real time constraints, by using the MARTE language features embedded in the CHESS modelling language; in this case, the application of the CHESS tool support for schedulability and worst-case end-to-end response time analysis enabled by the aforementioned MARTE properties will be also evaluated.

Still in the design phase, City will use the Mobius software tool (developed by the University of Illinois at Urbana-Champaign, USA) to build stochastic activity networks (SAN) model of the ATM use case. This model will be used for combined analysis of performance and security.

An initial description of a SAN model of the ATM UC is included in Deliverable D2.2.1. The model includes a performance sub-model, which captures the effects of a variable load (e.g. due to varied number of drones communicating with ground services) on the end-to-end delays in communication between drones and the ground services and a sub-model of the impact (i.e. deterioration) of different cyber-attacks on communication delays. Upon accurate parameterisation (for which direct measurements will be required on a dedicated testbed) the SAN model will allow one to explore a wide range of situations with variable load and different attacks.

The SAN models to be used in the ATM use case will be developed using the Mobius tool on its own. No integration is envisaged with other software tools. However the City team, together with the INTECS team, intend to study whether the functionality of the dependability plug-in of the CHESS tool can be extended so as to make it possible for one to derive directly from the SysML model of the ATM UC, defined in CHESS, a stochastic model, which is functionally equivalent to the SAN model(s) developed by the City team for the ATM UC. City will assist INTECS with the functional requirements for the extension of the dependability plug-in and, in case the plug-in extension is implemented, with its validation.

Additional support for simulation and implementation closer to the target platform will be considered in the use case. In this sense, the methodology developed by **UNIVAQ** involves different aspects in AQUAS project and ATM use case. The task-level application model, to be used as input, will be provided by UCs, using Model-2Model transformations in order to adapt and extract performance metrics into safety scenarios. Connected to the demonstrator platform selection (considering Unmanned Aerial Vehicle architecture), UNIVAQ will evaluate and compare performance with respect to different processors and HW architecture technologies.

The HW/SW Co-Design methodology will be supported by external performance analysis tools provided by different partners, in order to collect system data used during the Design Space Exploration (DSE). Other Non-Functional requirements (e.g. security, power consumptions, fault tolerance issues etc.) will be considered in future works. UNIVAQ will also extend its benchmarking activities related to PikeOS, to identify possible behavioral anomalies and system vulnerabilities in terms of safety and performance constraints (e.g. spatial/timing isolation, scheduling overheads, communication bottlenecks etc.).

For validation purposes, **AbsInt** will help in assessing the safety and security of the system. AbsInt will use its tools Astrée and RuleChecker to analyse the operating system (SYSGO's PikeOS kernel) statically for finding violations of certain safety and security principles, or proving the absence of such violations. Results of these static analyses can be used as arguments for safety and security in the certification process.

In order to assess the safety of PikeOS, RuleChecker is used to detect or exclude violations of MISRA-C:2004 and MISRA-C:2012. Security properties are assessed by using RuleChecker to detect or exclude violations of CERT, CWE, and “Secure C” (ISO/IEC TS 17961) rules. RuleChecker will for all checks be run in combination with Astrée in order to exploit Astrée’s sound semantic analysis of C code. Astrée will also be used to statically detect invariant assertions and invariant branching conditions. Statically decidable branching conditions can be exploited to increase system performance; while detecting invariant assertions yields further insights regarding the system’s safety and security.

BUT will use the ANaConDA tool to strengthen safety properties related with concurrency, in particular for the OpenSplice DDS middleware, which is written in C/C++ and uses multithreaded environment. Another candidate is sDDS implemented and provided by HSRM partner.

Finally, **AIT** will apply its tool model-mutation based test case generation called MoMuT, to the ATM use case. The MoMuT family of tools derives test cases from models of the system under test for fault-based testing. The tools generate test cases that are guaranteed to detect models that contain

certain user-selectable, seeded faults. Those tests can be already used during behavior modelling for validation and improvement of the model. During test case generation, also the fulfillment of component contracts, if available, will be checked. Additionally, robustness and performance tests will be provided, to verify that certain components react safely to unexpected inputs and that certain performance goals are met with a high probability.

3.2 UC2 – Medical Devices [ITI]

RGB (UC2 Leader) has developed a blood pressure (BP) and neuromuscular transmission (NMT) monitoring device for hospital operating room critical care performance. The system is using very innovative technology to support the anaesthesiologist in simultaneously monitoring BP and NMT during an operating room procedure.

In the case of BP control, the system operates by delivering vasoactive drugs with the goal of reducing patient´s hypertension, and precisely controlling blood pressure measurements in a patient undergoing surgical intervention in the operating room or in post-cardiac surgery in the Intensive Care Unit.

In the case of NMT control, the system will use relaxation drugs at the beginning of the operation, in the so-called induction phase, and will make sure that the patient is at the required relaxation level all along the operation, either in moderate, deep or intense blockade level; Then a second drug will be used in the recovery stage at the end of the operation.

From a technical architecture view point, the Medical demonstrator is composed of the Hardware in the Loop benchmarking system, on one side, and the test environment, based on PC software, on the other, as depicted in the figure below. The Medical demonstrator is a combination of the Medical use case with established and novel methodology and tooling provided by the use case partners.

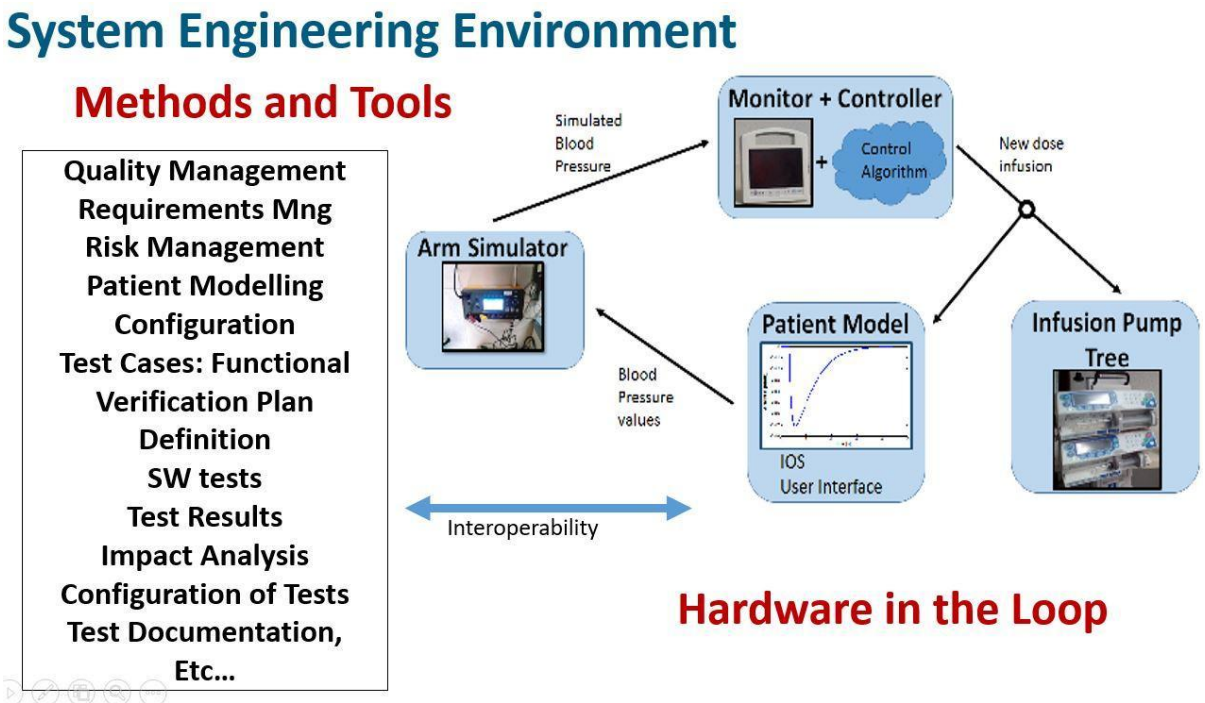

Figure: System Engineering Environment

The Hardware in the Loop prototype (HiL) will model selected parts of the real-world demonstrator with the goal to evaluate tools and methodology that enable the evaluation of safety, security and performance implementation attributes throughout the product life-cycle.

The basic idea for the technical part of the AQUAS Medical Devices Use Case demonstrator is to virtualize a real-world (physical) demonstrator where the patient is replaced by a mathematical model that behaves as a human. In practice it is a transfer function that relates as output the changes of the physiological parameter under control, having as input the specific drug(s) infusion(s) value(s). The blood pressure controller in the picture also includes the NIBP and NMT monitors, so that in practice, together with the Infusion Pump Tree, these are the two Medical Devices that will eventually be linked and used in the Medical practice

Physical Demonstrator: The objective will be to provide an environment to be used for the Functional Verification of the system allowing with some limitations:

- Automatic execution of the Test Plans.
- Increase Requirement Traceability incorporating tools into the development process.
- Accelerate design by using tools to analyze the performance of different potential architectures.
- Increase the safety of the patient by using Hardware in the Loop System to verify the functionality of the Controller before Clinical Validation.

To implement these objectives, ITI is developing a Quality Management (QM) module for the Art2kitekt (A2K) software suite. This will act as a tool to support co-engineering tasks. The architecture of the system is shown below.

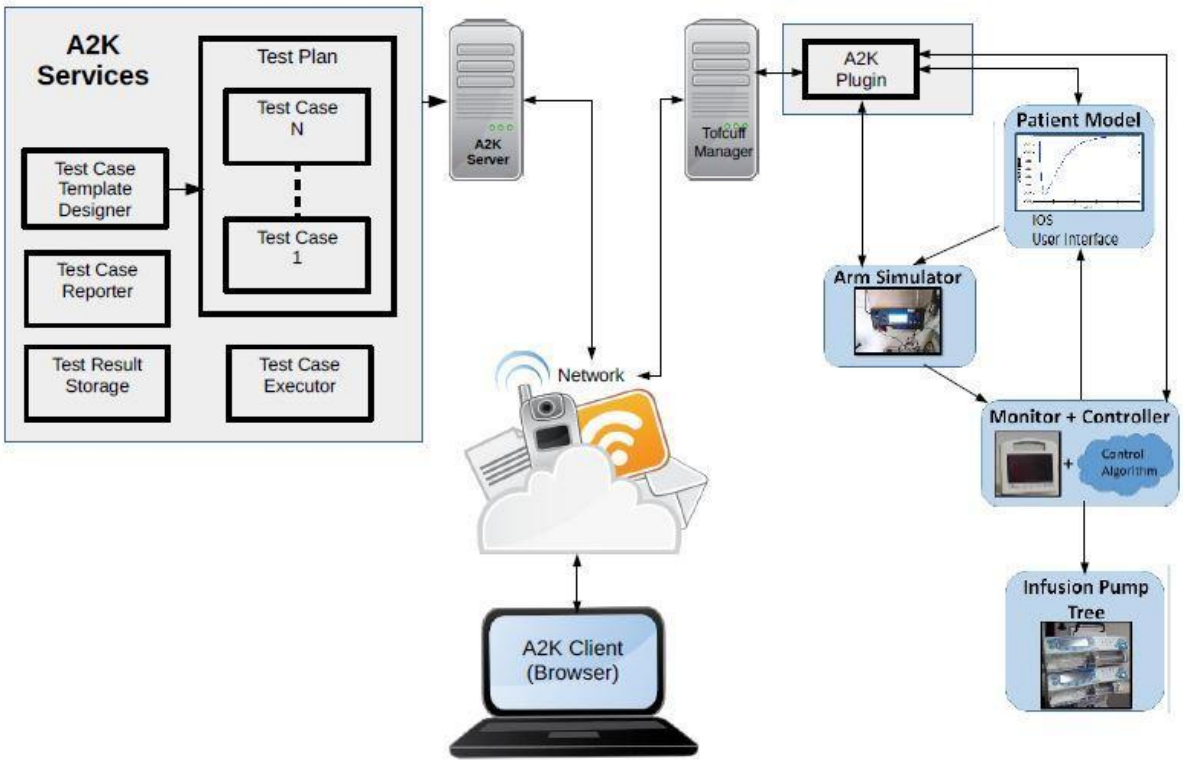

Figure: Architecture of the System

This architecture will act as a harness which will manage inputs to and outputs from the various hardware components of the demonstrator system. It will compare the measured test results to those expected and provide reports. It will also manage the relation between real and expected results in terms of the constraints provided by the various tools used during the design, implementation and test procedures.

The tools, procedures and workflows that we intend to integrate within the A2K QM are as follows:

ALL4TEC proposes a Safety and Security co-engineering method based on Safety Architect and Cyber Architect tools, as presented in Section 4.9. The proposed method supports risk management inside product life cycles (requirement or modelling phases) and across these two phases. If system requirements or architecture models are created in partner tools based on SysML/UML model, the resulting SysML/UML models could be imported in Safety Architect for classical safety analysis, RRA002 (FMEA/FMECA tables that enables tabular linking of safety requirements to safety barriers that symbolizes modelling elements to reduce the failure propagation at system component outputs) or in Cyber Architect for Security risk analysis, RRA003 (vulnerabilities and threats modelling in tabular form with the possibility of defining dependencies). Based on the analysis results, the system architecture solution could be confirmed or rejected with the need for re-design or enhancements across product life cycles.

AMT will create a SysML model using medini analyze, that will allow safety and security analysis. Applicable risk analysis methods will be applied. The model might be imported into the other SysML tools.

City - main planned contribution concerns analysis of system requirements/specs TrustPort's contribution in AQUAS Medical use case can be security requirement definitions like sensitive data misuse, data confidentiality, integrity, availability, non-repudiation and quality of service. From the viewpoint of human factors, addressing requirements of e.g. IEC 60601-1-10, specifically about safety/security implications of: mental model the operators have of the device; cases in which the operator must intervene (clauses 5.1, 6.1. Also relevant: EN62366-1, "use error"-related clauses). The main novel aspect of this is analysis of safety concerns regarding operator intervention scenarios.

In addition, we expect to possibly contribute to:

- threat analysis from the combined safety/security viewpoints, especially regarding human contributions, as part of early interaction points;
- specification of the test plan such that an argument, possibly statistical (*cf.* D2.1.2) can be derived as to how the test demonstrates that the software is properly verified (matches specification) and/or validated (fit for purpose and safe).

Last, the City team will contribute to the planning of interaction points and study of their effectiveness, according to its role in WP3.

CEA – wish to use a combination of Papyrus modeler and Frama-C static code analyser to apply a model-based software/security co-engineering method for security properties verification. The method is described in Section **Error! Reference source not found.**. In particular CEA will use Papyrus or the software architecture modelling, and the security requirements and refined properties modelling. Modelled elements will be traced and related with Papyrus traceability tools. Papyrus will also be used to generate code for the software architecture, with ACSL annotations representing modelled properties. Such ACSL annotations can be either directly generated, if properties are directly modelled, or inferred from higher-level requirements that are modelled and refined.

BUT - will contribute in modelling and simulation of the whole patient-in-the loop system. BUT will design and implement a complex simulation framework covering the control mechanism, drug injection pump, model of the patient and measurement device (blood pressure, neuromuscular relaxation etc). These parts of the framework will be modular and allowing to plug in real technical components. The goal is to experiment with various types of new drugs, control mechanisms and safety issues (robustness of the control mechanism in cases of malfunction of the pump or measurement device). BUT will also consider possibilities of analysing resilience of the control mechanism against errors, including errors in the input measurements. Apart from that BUT will offer its experience and tools analysis of various aspects of the use case code, including, e.g., concurrency aspects, memory safety aspects, and/or performance-related aspects. Finally, BUT will leverage its experience in computer security and, in collaboration with TrustPort, help evaluating security features of the proposed solution.

Tecnalia - The main contribution is to analyse the use case and to develop an assurance case by using OpenCert tool. We will include co-engineering factors to the safety cases and it will be used through the lifecycle. The assurance case provided by RGB will be modelled, considering, the safety, security and performance requirements requested by the standards use case. The assurance case model must ensure the co-engineering factors are shared among the different phases of the lifecycle.

TrustPort - In the Medical Use Case TrustPort is planning to contribute to analysis of security requirements and specifications according to norm ISO11073, OWASP and best practices and offering inputs into modelling of security requirements in SysML language in conjunction with other tools like CHESS, with help of SSLDC Tool.

In addition, TrustPort plan to contribute to threat analysis from the combined security/performance viewpoints, including human contributions in design, modelling and verification of product life cycle.

TrustPort's contribution in AQUAS Medical use case can be security requirement definitions like sensitive data misuse, data confidentiality, integrity, availability, non-repudiation and quality of service.

3.3 UC3 – Rail Carriage System [ClearSy]

As described in WP2 D2.2, Clearsy expects advances in security, safety and performance analysis through new processes and methods applies to the COPPILOT metro doors control system. This system must achieve high safety and performance constraints. Basically, the screening doors should open when a train is stopped at the right position on the platform and its doors are opening, and the screening doors should close when a train is stopped at the right position on the platform and its doors are closing.

The COPILOT system design approach mainly rely on formal methods cascading to code generation (development), testing, commissioning and exploitation (return or experience as inputs to system improvement).

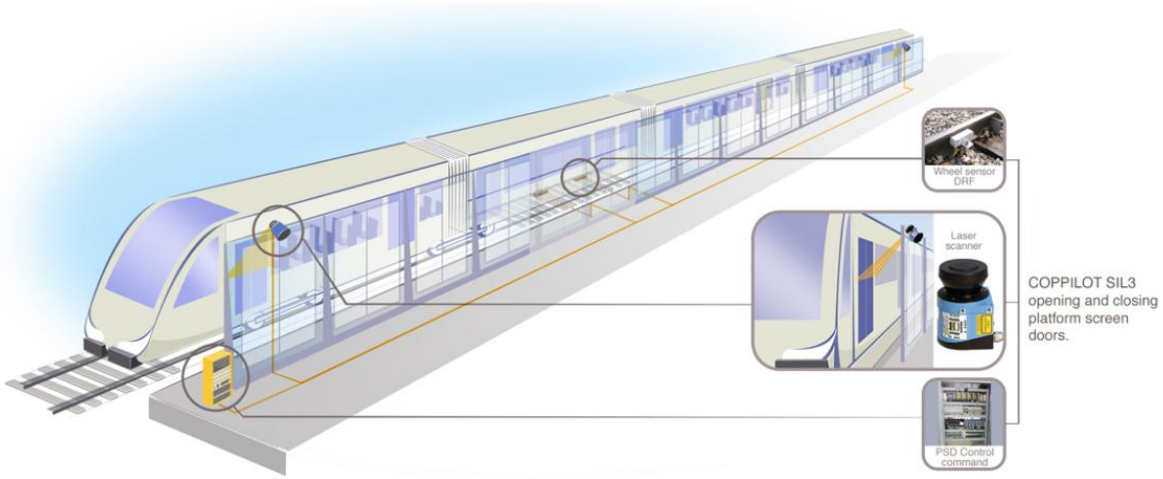

Figure: Overview of the COPILOT platform

Design and analysis tools chalenges are first to ensure always faster and safer controllers but also to bridge domains with system level and consider systems interactions. As illustrated below (only for the ground sensing part), the COPILOT system dynamically interacts with many other systems (rolling stock, other ground systems, sensing…) of sometime variable configurations, leading to adress relatively high operation complexity, which must be captured by tools and not only considered as boundary conditions.

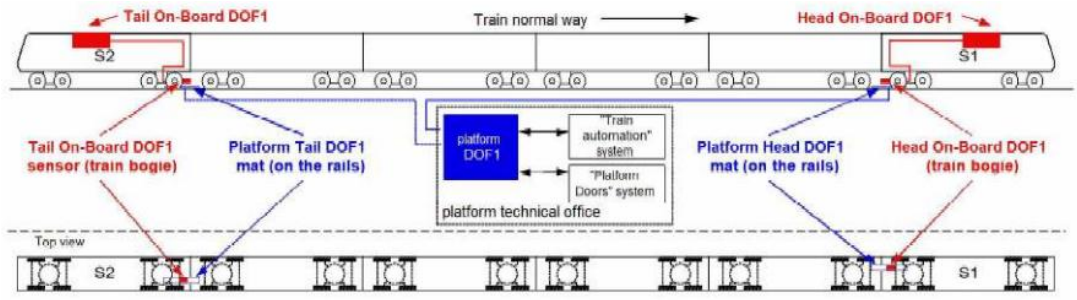

Figure: COPILOT interaction with train

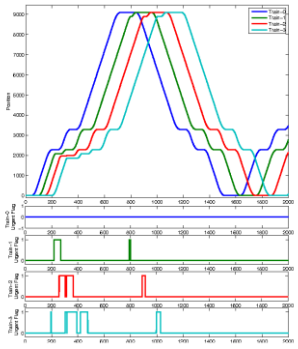

Figure: Train dynamics and bus signals

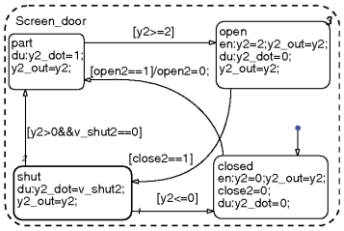

Figure: Doors openning State machine

So, the tooling challenge will be to improve state of the art approaches used by Clearsy, ranging from the embedded software development to etended virtual system SSP validation.

BUT will investigate new tools interactions to automatically analyse selected critical safety and performance code features and checking the absence of critical code errors whenever new code is introduced or changes are required after an interaction point.

It will be achieved through

- static analyses with tools like Predator, 2LS, or Ranger/Loopus using the approaches of abstract interpretation, (bounded) model checking, and SAT/SMT solving.
- dynamic analyses with tools like ANaConDA and Perun use extrapolating dynamic analysis, noise injection, and statistical analysis.

The BUT tools (Predator/Forester/2LS, ANaConDA, Ranger/Loopus and Perun) and their planned extensions will stay generic but with a stress on modularity and efficiency needed at interaction points, and they will be particularly optimized and validated for the use case.

CEA activities will investigate specifications translation from B0 into ACSL specifications analyses of security properties through static code analysis (WP plug-in of Frama-C tool) and relate static code analysis to system/software models. CEA aims too achieving tool cooperation by import/export of models in Papyrus UML/SysML and new properties modelled as annotations in Papyrus UML/SysML model and code generated (with annotations) for static code analysis. CEA aims also at developing collaboration with MTTP on SysMLSec integration and partner tools involved in CHESS to promote interoperability through Papyrus.

MTTP will explore the use of the SysML-Sec approach (supported by the free and open source toolkit "TTool") in order to verify the security (and safety, and performance) requirements on the system. The analysis will be performed on the HW/SW partitioning stages of SysML-Sec. More specifically, MPPT will identify security and safety requirements, identify and model attack trees, model the functional and architectural view of the system to finally propose and evaluate system safety and security requirements.

SISW will contribute to the behavioural/time-domain system analysis investigations to extend SSP analyses practices. These analyses will involve various tools interactions (controls runtimes, physical models, requirements…) at different stages of the product life cycle (development an operations) and will involve techniques like time domain system simulation combined with SSP requirements. Along with the use case leader, these investigations should achieve better knowledge on how complex CPS design practices could be improved to achieve higher SSP. Tools interactions will be investigated using as possible standard interfaces like FMI, Sfunction or any other relevant interfaces.

Trustport aims at investigating the transformation and interactions of security and safety requirements created from security standards, norms and best-practices to use-case PLC tools and provide final set of security requirements to compliance process. Trusport aims at developing collaboration with BUT regarding test scenario definitions.

3.4 UC4 – Industrial Drive [AMT]

In UC4 a virtual HW prototype of an industrial drives system will be created that shall be used to verify performance constraints together with safety and security requirements for a representative set of scenarios. The scenario considers the development phases concept, design, development and validation. Operation and retirement phases are not in the scope of this use case. See deliverable D2.1.4 for a detailed description of this use case. D2.2.4 provides details on the architectural and processual requirements.

The following sections describe the process followed by the use case with a focus on the tools and their interaction from the perspective of co-engineering the three quality aspects safety, security and performance.

3.4.1 Process Definition

The SAG system development generally adheres to standards IEC 61508 for functional safety and derived standard IEC 61800 (Adjustable speed electrical power drive systems). The process defined within AQUAS must be compatible to the latter standard. Additionally, it should follow the security guidelines for the industrial domain set out by the international standard IEC 6244300 to provide a solution that is secure-by-design.

To ease the evaluation of co-engineering, the approach in UC4 is to limit the scope of a full-blown product life cycle to a subset and later integrate successful results into the full product life cycle.

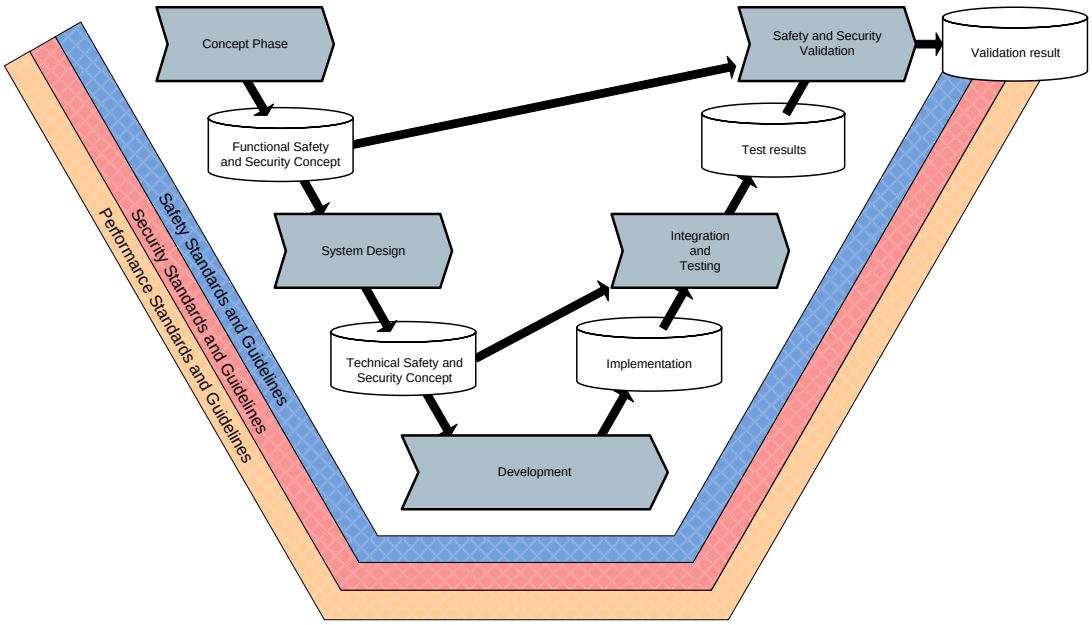

Figure: Overall life-cycle of UC4

The above figure depicts the overall life-cycle method with additional tracks indicating safety, security and performance standards and guidelines. It serves as base life-cycle for the industrial drives use case demonstrator.

Below diagram breaks the product life cycle down into activities from the different development disciplines and maps them to the tools from the AQUAS partners that are applied in this use case. The central data for interaction throughout the use case is a system model in SysML that is annotated with quality attributes and a requirements model for safety, security and performance requirements. These models are initially created and maintained in medini analyze and can be exported to other tools, e.g. INTEC's CHESS tool, which can provide additional supportive quality criteria for balancing the architecture. The results of functional safety analysis lead to the addition of safety functions in the design, and additional design requirements, which can be traced with medini analyze in each product life-cycle phase.

It is desirable to enhance the current flow with features that enable a balancing of safety, security and performance. The more detail is provided to the balancing mechanisms (e.g. implementation-, timing-, area-costs of safety functions) the higher is the accuracy for hitting the optimally balanced architecture for all quality aspects.

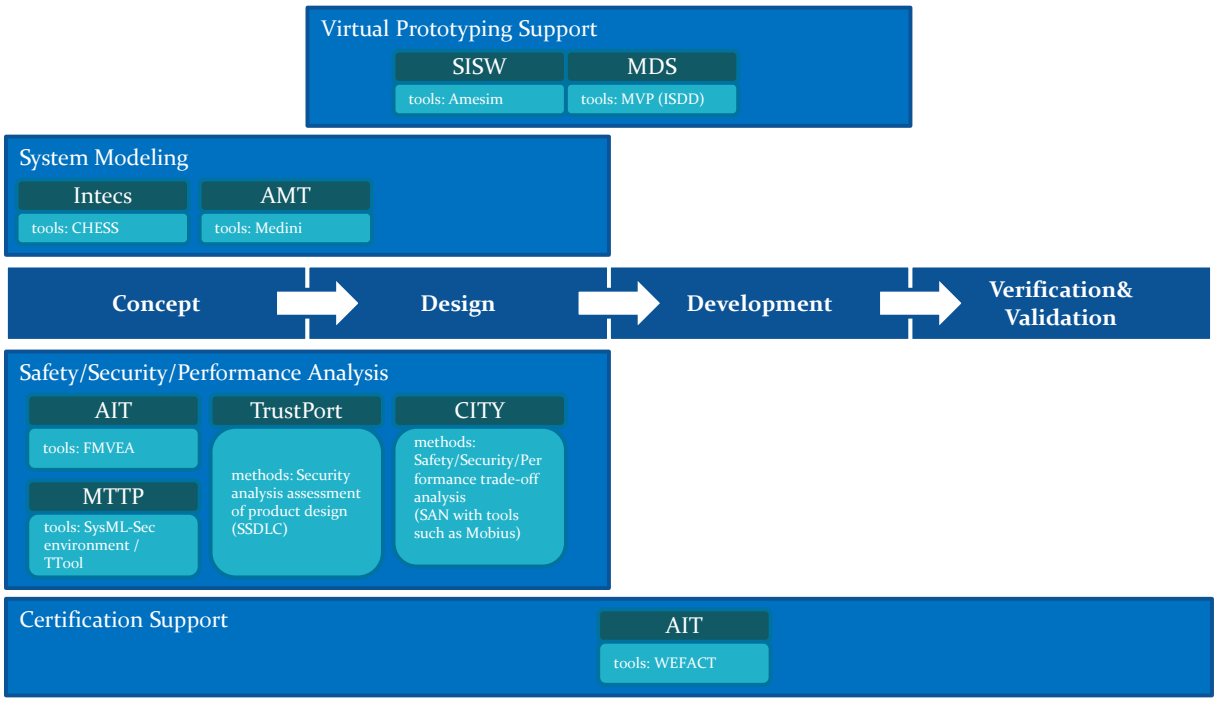

Figure: Activities of the UC4 product life cycle mapped to AQUAS partners' tools

As mentioned above, UC4 limits its scope to the following activities and their interaction in the product life cycle.

System Modelling: The methodology elaborated in SESAMO shall be enhanced with performance considerations in early stages. medini analyze is used to relate selected security and safety mechanisms to requirements. CHESS is used for timing analysis. Particularly, the CHESS tool can be extended with code generation features and WCRT (worst-case response time) analysis capabilities.

Safety-Security Analysis: Evaluate applicability of the FMVEA (Failure Mode Vulnerability and Effects Analysis) for combined safety and security analysis and compare it to other suitable approaches (e.g. SESAMO's Security FMEA).

Safety-Performance Analysis: Evaluate feasibility for time-domain behavioral analysis of the combined virtual controller and the virtual physical system.

Virtual Prototyping: Support the verification of safety features and other quality aspects. Implement a seamless flow from System Level Model to the Virtual HW Prototype. Provide a platform that allows direct comparative analysis of Virtual Prototype vs. FPGA approach and combined controls models and virtual physical environment as well as combined virtual platform and virtual physical environment.

Certification Support: WEFACT as framework for supporting a general assurance case covering all relevant dependability attributes (safety, security and performance).

The product life cycle of UC4 maps to the analysis and realization phase of the AQUAS methodology, the operational phase is not in the scope of this use case. The UC4 product life cycle has been modelled using the process building blocks from the AQUAS methodology (see section 2) as delivery process in the EPF model using EPF Composer. The following sections describe important aspects of the delivery process and the deviations from the AQUAS methodology. For a more detailed description and navigability between the process artefacts, please refer the additionally delivered EPF model and generated process website.

Analysis Phase

The analysis phase of UC4 is separated into a concept and a design phase.

The goal of the Concept Phase is to collect basic system information, derive functional safety and security requirements, as well as creating preliminary architecture. Tools and methods applied for this phase usually capture system information in an organized way (spreadsheets, databases) that allows the definition of dependencies and enables tracing of requirements and artefacts throughout the process. medini analyze is used for requirements engineering, safety analysis (FMEA, FTA), security analysis (Attack Tree, Security-FMEA) and for system modelling (SysML).

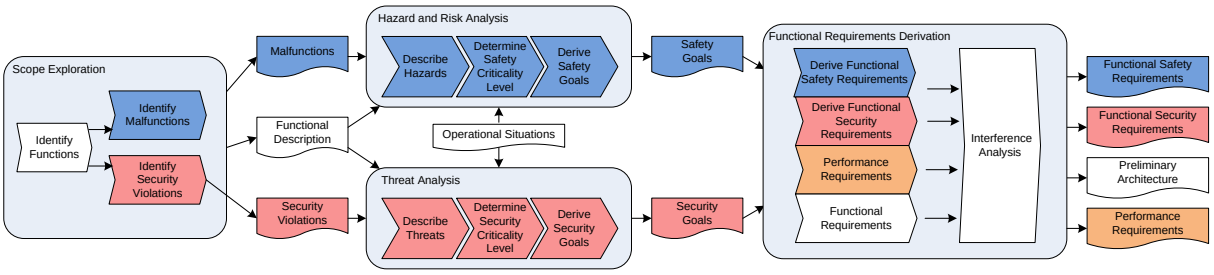

Figure: Concept phase of UC4

The System Design phase consists of two activities, Define System Design and Hardware and Software design.

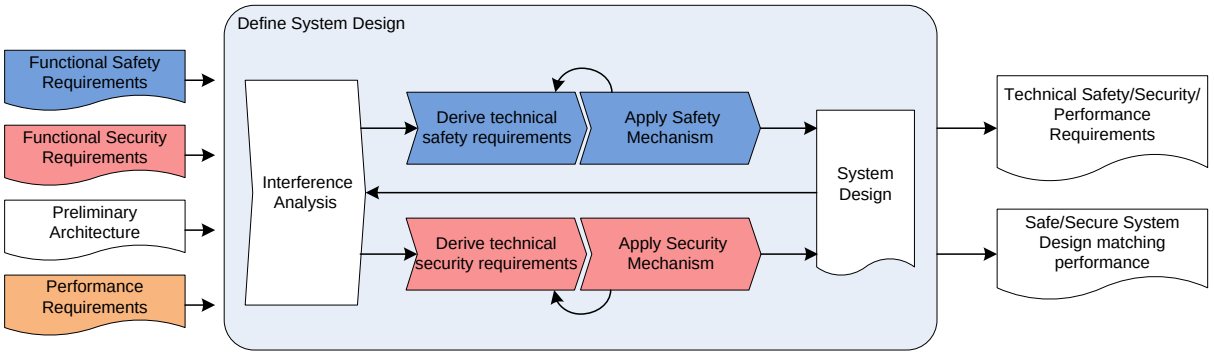

Figure: Product Life-Cycle: System Design

The purpose of the System Design Phase is to create technical safety, security and performance requirements as well as a safe and secure system design that is aligned to performance requirements. Safety and security mechanisms are applied concurrently at first, but also balanced with respect to performance requirements. Each applied mechanism (e.g. safety building block or security building block) shall have its own performance attributes and thus enabling a way of performance-aware interference analysis.

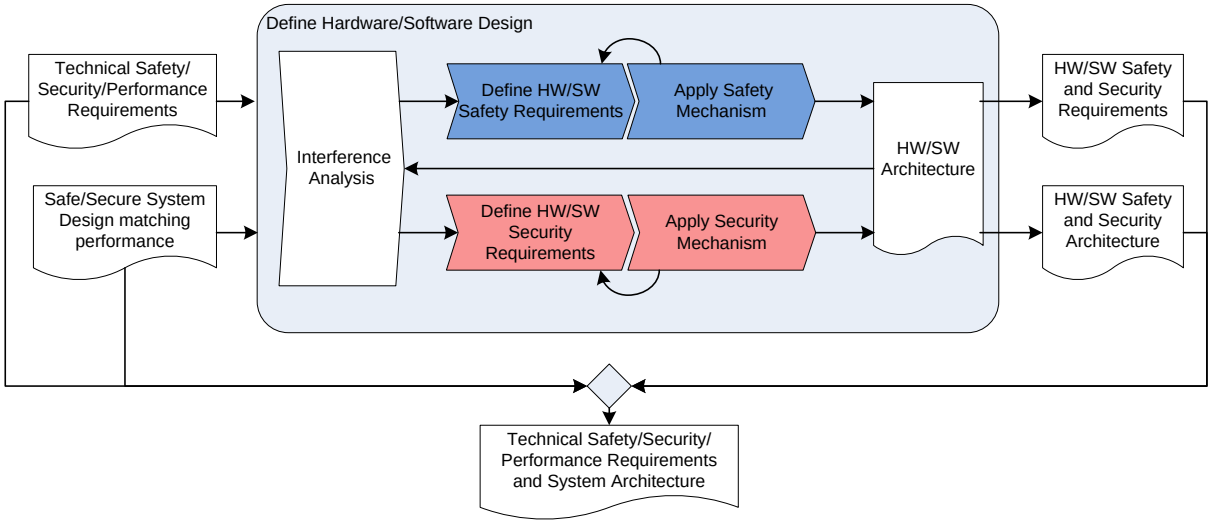

Figure: Product Life-Cycle: Hardware and Software Design

With the technical safety/security/performance requirements and a system design in place, the next step is to decide which parts of the system are to be implemented in SW or HW. The flow applied is basically the same as the one for system design, but ending with technical safety/security/performance requirements and a system architecture including the system distribution into SW/HW.

Realization Phase

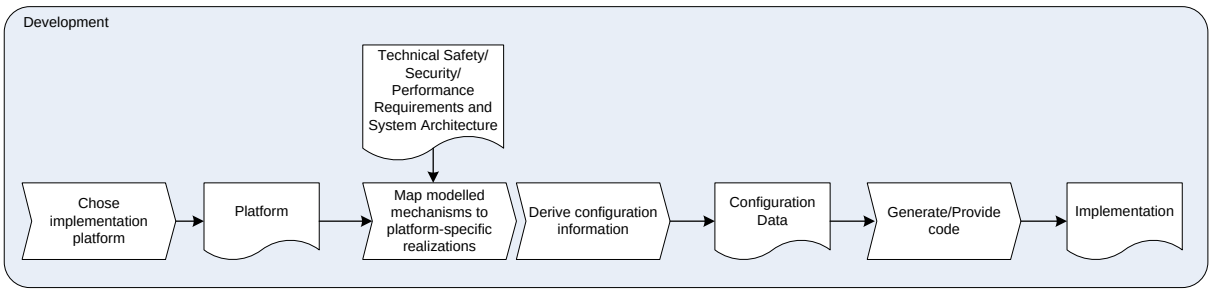

Figure 11: Product Life-Cycle: System Development

In the development phase, the modelled system architecture with its safet/security mechanisms is mapped onto a chosen implementation platform. Ideally, that platform has code generation capabilities inhibiting the need of manual coding, thus saving costs and efforts.

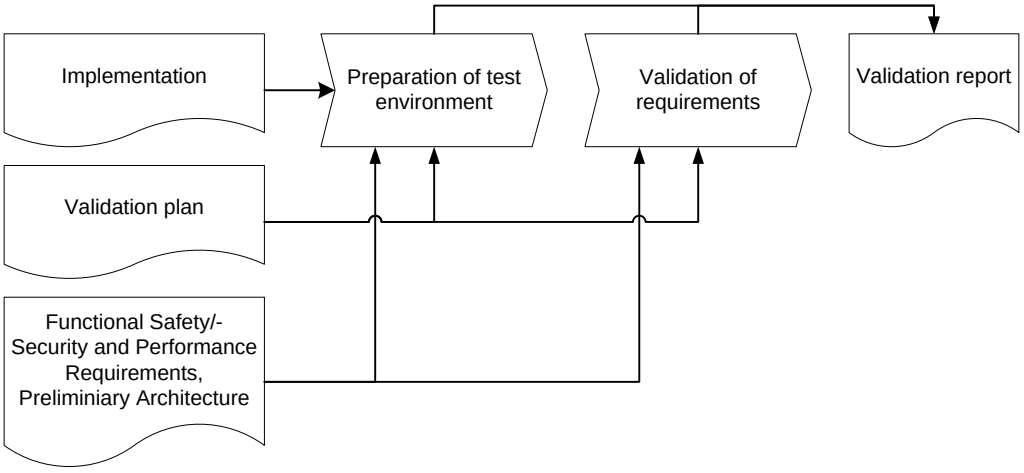

Figure: Safety and Security Validation

In phase Safety and Security Validation requirements are validated against the implementation. Note that for the industrial drives demonstrator a specific testing phase is omitted due to the experimental nature of virtual prototyping.

3.4.2 Tasks and Activities

In order to avoid inconsistencies please have a look at the list of partner contributions, which can be found in D2.2.4 – Demonstrator Architecture, chapter 2.

Nevertheless, a short overview is provided hereby. Following tasks related to tooling are expected for UC4 (please note that this list may evolve during the implementation phase and should be considered as living content):

- Requirements modelling
- SysML concept architecture model
- FMEA/FTA
- Security-FMEA/Attack Tree Analysis
- FMVEA
- Timing analysis of the system (with model imports from Medini Analyze) in CHESS
- Interference analysis for safety/security/performance (at various PLC points – see D2.2.4 for an interaction points example, as well as D3.1 for a list of expected interaction points in UC4)
- WEFACT modelling for certification support
- SAN modelling (e.g. with Mobius) for safety/security/performance system validation in early phases
- Virtual prototyping support (SystemC) by MDS with MVP tool
- Virtual prototyping workflow with ISDD (MDS) (safety, security, performance)
- Virtual prototyping workflow based on SysML-sec (TTool) (safety, security, performance)
- Electronic motor modelling with Amesim coupled to the SystemC-based virtual prototype

- Secure Software Development Lifecycle (SSDLC) for security requirements engineering and security testing/assessment, as well as supporting security activities during the development process

3.4.3 Methods and Tools

The current tooling of UC4 provides some integration that has been implemented for the SESAMO project between medini analyze and CHESS. This included the

- Hazard and Threat Analysis in medini analyze,
- derivation of functional and technical safety/security concept by formulating requirements based on qualitative fault tree analysis that was done on a SysML model,
- export of the SysML model to CHESS in order to do some complex Worst-Case-Execution-Time analysis that resulted in more requirements that were later be exported back to medini analyze in order to show traceability to the system design model in SysML.

In AQUAS there are some potential enhancements planned, that shall be addressed to better support interaction between disciplines and integrate quality analysis results in one common database for review and analysis in the interaction points.

The exchange of SysML models between CHESS and medini analyze shall be enhanced, on the one hand the exchange of diagrams and their layouts shall be supported. Additionally, re-importing models from CHESS into medini analyze would be comfortable, making a roundtrip integration possible: Changes on the system model in CHESS could be fed back to the SysML models in medini analyze. In CHESS, the support for incremental updates could be improved, so that, when the system architecture design in medini analyze requires changes, not most of the separation of the system model into HW/SW in CHESS needs to be redone.

With medini analyze and CHESS being used across life-cycle phases, the other tools need to integrate with the data taken from these two tools. medini analyze will provide requirements engineering and system modelling, CHESS will provide modelling and simulation with special focus on timing analysis.

3.4.4 Interaction Points

A detailed example for an interaction point is described in D2.2.4 – Demonstrator Architecture, chapter 1.3 – Product Life-Cycle. Furthermore, the current list of expected interaction points for other PLC phases can be found in D3.1.

3.5 UC5 – Space Multicore Architecture [TASE]

For UC5 Thales Alenia Space in Spain is developing an application that will run on a multi core platform which is certified by the ESA for flight purposes.

The selected platform is a Gaisler GR712RC board which incorporates a Leon 3 dual core processor. In the scope of AQUAS the software aspect of this platform will be studied in order to study the interdependencies between Safety, Security and Performance.

The software will perform tasks like memory scrubbing, watchdog functions and telemetry and telecommands receival/sending, which will run in separate cores in the defined platform. The

possibilities of locks, interdependencies and inconsistencies due to concurrency problems will be the main focus.

Requirements for each aspect have been defined and subjects like achieving the target performance while maintaining the safety and security aspects of the application intact will have to be studied.

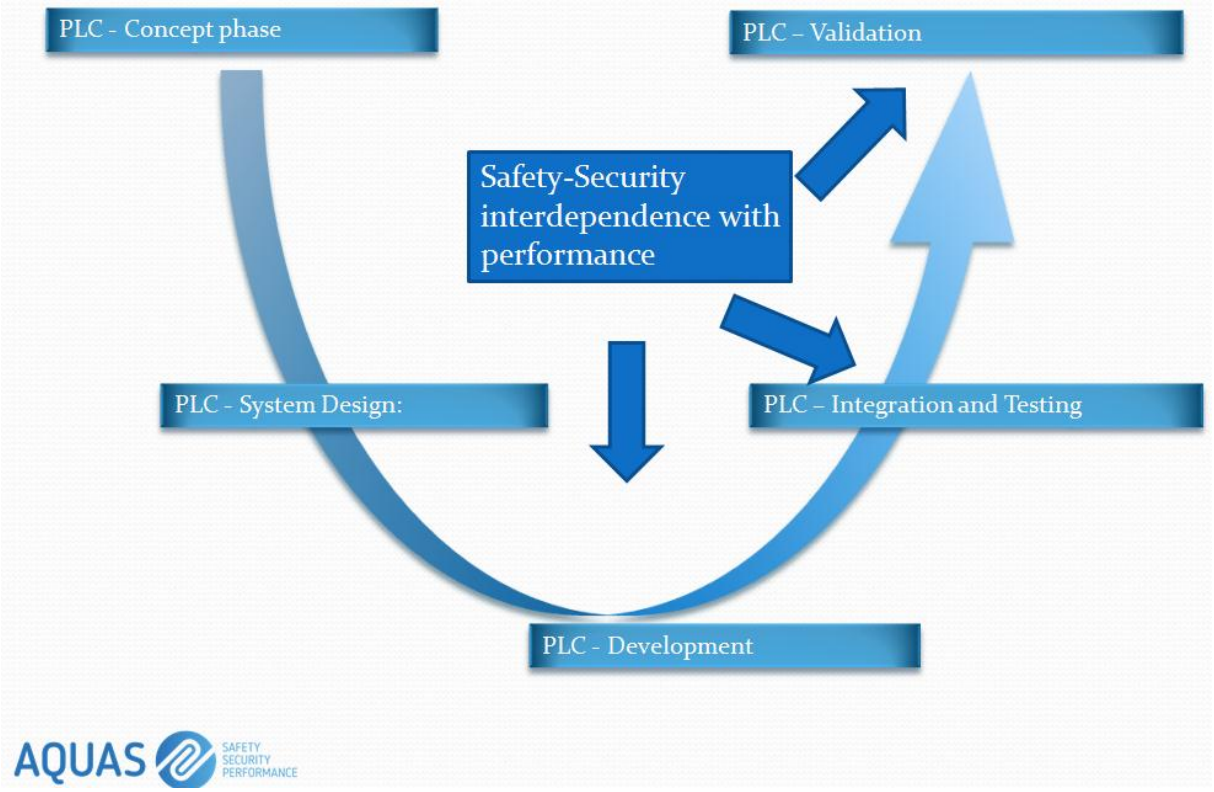

Figure: Stages in UC5

AQUAS partners will contribute to this study with the proposed methodologies and tooling.

INTECS: Provide support for the modelling of the SW architecture and associated timing properties.

To be used to enable schedulability, worst-case response time analysis.

CHESS information could be used to instrument the TimingProfiler tool.

Collaboration with All4Tec. Understand safety/security related information that can be exchanged between the two tools.

TECNALIA: Analyze the use case and to develop an assurance case by using the OpenCert tool.

OpenCert is a product and process assurance/certification management tool to support the compliance assessment and certification of safety-critical systems.

MDS: Participate in the formalization of a workflow for the balancing of Safety, Security and Performances.

Show the capabilities of the ISDD© which is a platform integrating each process of a project: Specification, Design and Documentation.

ITI: A2K tool upgraded to simulate, generate code, and monitor dual-core, shared memory, Leon processors running RTEMS.
A2K tool upgraded to provide sensitivity analysis of task schedulability to task timing parameters.

BUT: BUT will leverage its long-term experience with hardware-accelerated solutions of various computationally expensive tasks to evaluate the efficiency of the considered system architecture of the use case and to investigate possible optimizations of the proposed solution using, e.g., specialized hardware.

BUT will contribute tools and methods for detecting concurrency defects mainly through dynamic analysis using its ANaConDA framework.

TRT: State of the art done on performance verifications techniques for multicore GR712RC

Coordination meeting with ITI.

Ongoing: Work on detail software architecture of the case-study, definition of methodology for FPGA.

UNIVAQ: Introduced Safety (Sa)and Performance (Pe) constraints into the Hepsycode Methodology

Extended the Design Space Exploration (DSE) to consider processes/tasks/threads partitioning (with HPV SW-partitions)

Analyzed different HW/SW multi-core LEON3 scenarios.

SYSGO: SYSGO will contribute to the AQUAS project its safe and secure real-time hypervisor PikeOS, which will be used to examine novel techniques for improving safety, security, and performance in combination, by trying out new static analysis techniques. In the context of this use case, SYSGO will explore to create its own special analysis tools especially suited for specific problem classes with the PikeOS hypervisor, where we do not have any tool available.

HSRM: Integrating verification and testing. Methods and languages for Worst Case Response Time (WCRT). Layered overall architectural design of a real-time microkernel.

ABSINT: Adapted TimingProfiler(TP) tool to UC5. Supplied all partners with TP and training. Support partners in usage of TP.

ALL4TEC: Integration of Safety Architect tool with Papyrus/CHESS tool to import system architecture model. Safety analysis (FMEA/FTA) on the UC software architecture model. Evaluate Safety Architect and Cyber Architect tool integration for combined safety and security analysis.

Aside from the SW development, as a sub-use case, **TRT** will develop an activity involving an FPGA which they will coordinate and will have contribution from several partners under the scope of UC5.

4 Tools

This chapter describes the tool-specific functionality required to support the co-engineering process according to the AQUAS methodology. The descriptions focus on tool features supporting the interaction of domains and disciplines, e.g. in the area of tracking process progress, data and artefacts, and particularly interaction points, where cross-domain and cross disciplinary analyses must be executed to realize safety-security-performance analysis, supporting system design space exploration and trade-offs.

Providing this functionality requires tool integration in many cases. The tool descriptions reference back to the delivery process of the use cases, where applicable.

4.1 System Modelling and Analysis of Quality Criteria [AMT]

AMT is supporting the safety and security co-engineering by bringing in its tool medini analyze, a toolset supporting the safety analysis and design for software controlled safety related functions. As depicted in the figure below system models that represent the design of these functions are enhanced by properties that support the different analysis methods that are applied in the context of safety considerations, like Hazard and Risk Assessment (HARA), Fault Tree Analysis (FTA) or Failure Mode and Effect Analysis (FMEA).

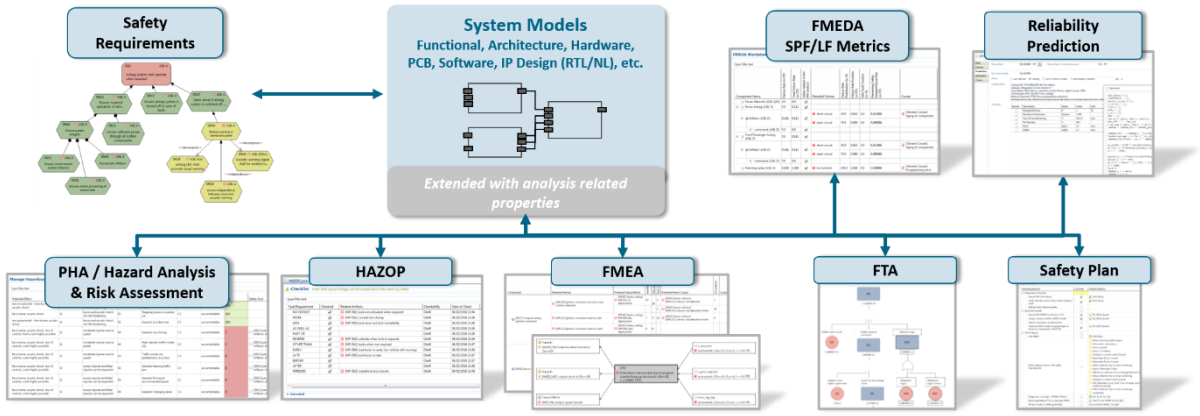

Figure: Safety related analysis in medini analyze

Beside the safety aspects medini analyze supports the analysis of security quality criteria based on the same mentioned system models (see figure below). It allows for the modelling of assets that might be potential goals for a security breach. These breaches can be examined by conducting an Attack Tree Analysis that result in Attack Paths describing potential scenarios how the asset can be attacked. These scenarios are assessed in respect to the criticality by applying a Threat Analysis and Risk Assessment.

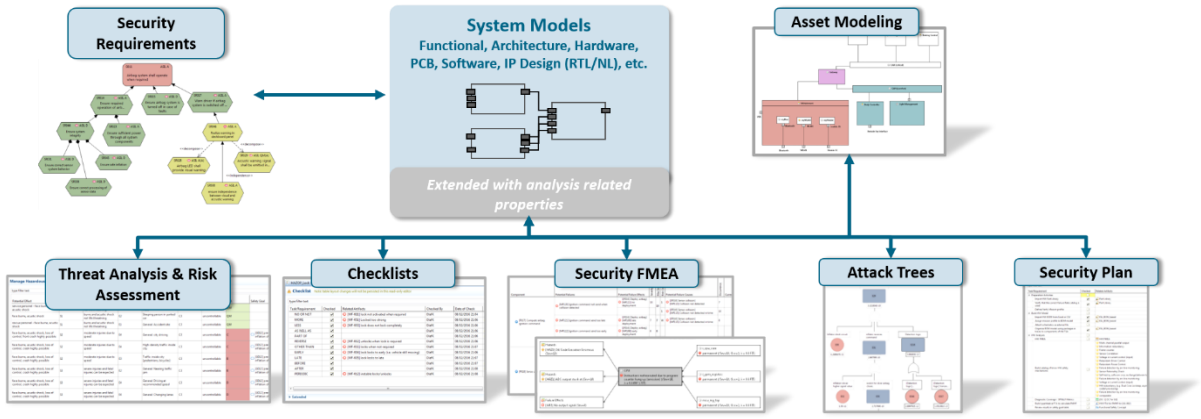

Figure: Security related analysis in medini analyze

In order to support the different requirements coming from the safety standards in the different domains (e.g. Medical or Industry) and additionally to support the various methods that are applied in the area of security medini analyze will be extended by the concept of so-called domain profiles. These profiles introduce the domain specific terminology, the different risk graphs and different analysis methods in a highly adaptable manner. This is especially important in the security domain because here the standards are currently under development and not yet finalized.

Examples for the differences in standards and methods supported by domain profiles are:

- Criticality Levels (SIL, DAL, ASIL)
- Analysis Methods like
    - Hazard and Risk Assessment
    - Threat Analysis
    - Support for threat model classification according to STRIDE [STRIDE]
    - Fault Tree Analysis
    - Attack Trees
    - Failure Mode and Effect Analysis
    - Support for Hardware Metrics (Safe Failure Fraction, Single Point Fault Metric, Latent Fault Metric, etc.)

The potential interaction points that medini analyze supports consist in the exchange of system design models and requirements.

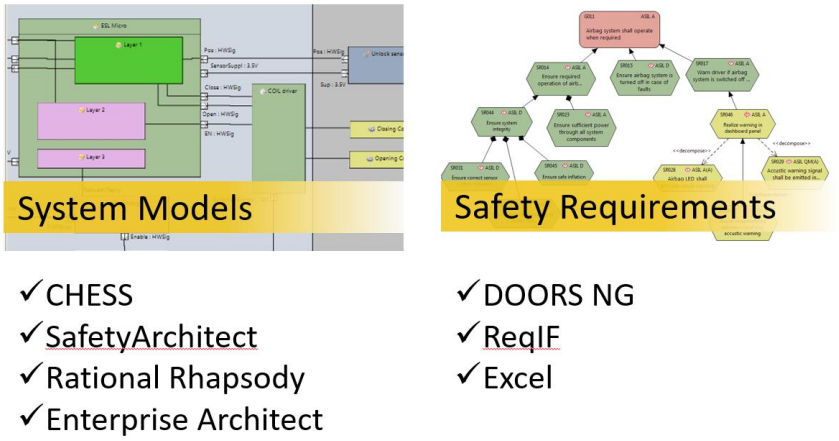

Figure: Interfaces for interaction points

Thus various interfaces to other tools to support this exchange of models exist and will be extended in the AQUAS project. To be emphasized, in particular, are on the one hand side the interfaces to the CHESS platform that were initially developed in the context of the SESAMO project and that are planned to be extended and on the other hand the interface to DOORS NG that was developed within the AQUAS project for UC2.

The following table shows the proposed specific functionalities of medini analyze to be addressed in AQUAS.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Concept and system design	medini analyze, Papyrus/Chess, Safety Architect, SSLDC	Safety and Security analysis with medini analyze and exchange of requirements and system models	Tools support to improve Safety, Security properties	UC2 - Medical
Concept and system design	medini analyze and Papyrus/Chess	Safety and Security analysis with medini analyze and interactions with CHESS	Tools support to improve Safety, Security and performance properties	UC4 – Industrial

4.2 Co-Design and Implementation of Safety and Performance [ITI]

ITI is contributing to the AQUAS project objectives by providing the foundations in the development processes to execute co-engineering activities efficiently, especially in the domains of safety and performance. These processes will in particular define and implement the mechanisms of interaction between the different tools and technologies used in the AQUAS co-engineering tasks. In other

words, ITI is providing the technical support, expertise, software and work-flows to facilitate the variety of interaction points that will emerge during the project.

At this stage in the project most of the interaction points are still under discussion and not yet well defined, so we are adopting an agile, flexible approach to their design and implementation processes. These activities will focus on researching and identifying the ways and mechanisms of allowing partners' tools to perform co-engineering analysis of their models and design resources. This will be driven by the outputs of Work Package 3 (methodology) and of course the ongoing work in each use case. An important main goal is to facilitate tool interactions that will enable analysis and modeling to check validation of the associated requirements.

ITI is also exploring the possibilities of Design Space Exploration where the engineer can try out different system configurations to see how trade-offs between safety, security and performance (SSP) artifacts and metrics affect the final system design. This task of course depends strongly on the definitions of the appropriate SSP metrics and how these relate to system design parameters. These quantities are now beginning to emerge from the other work packages in the project.

ITI is using its Art2kitekt (A2K) tool in two use cases: UC2 (medical devices) and UC5 (space multi-core). A2K is a tool that enables modeling, analysis (timing, power, etc.), simulation, code generation, monitoring, and quality management of heterogeneous cyber-physical systems. In UC2, A2K is being used as a test manager for the medical hardware-in-the-loop. In this application A2K will provide stimuli to the system under test, collect system outputs, and compare these to desired safety and performance results. A2K will also interface with other tools as a manager and implementer of the (to be defined) interaction points.

In UC5 A2K will be used for modeling and timing analysis of the multi-core architecture. We are interested in exploring how aspects of safety and performance are related to different deployment conditions of, for example, the distribution of tasks to processor cores, the data communication between the cores, and use of shared resources. This is a good example of the close relationship between design and implementation in a safety and performance context.

At the present time ITI is currently collaborating with the partners Absint and BUT in building some co-engineering demonstrator applications. With Absint ITI is planning to use their TimingProfiler tool in conjunction with A2K to provide accurate estimates of the worst-case execution times of software tasks. These timings will be imported into A2K to enable timing, schedulability and sensitivity analyses to be performed. ITI is also exploring how to integrate BUT's static code analysis tools (Anaconda & Perun) into A2K to provide safety and performance information during the design and implementation processes.

Regarding other tool interactions, as the project develops ITI will take into account the various information that needs to be collected in the design models and analysis tools that should be collected for enabling co-engineering support. Beyond model types and formats, semantic gaps, data repositories, management, communications, and dependencies are all being addressed, in an agile fashion.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Concept and system design	A2K – simulation, timing analysis & code generation. BUT Code Analysis Tools Absint Timing Profiler	Safety and Performance. Sensitivity analysis. Design Space Exploration	Rapid exploration of system timing performance under different conditions and configurations.	UC5 - Space
Integration & Testing	A2K Test Manager BUT Patient model Tools providing requirements for development of test plans.	Interaction of security and safety.	Hardware-in-the-loop simulation and system testing. Simulation and evaluation of security aspects.	UC2 - Medical

4.3 Behavioural system analysis for improved SSP [SISW]

The Safety-Security-Performance attributes of a Cyber Physical System are the indicators defining on how the Cyber Physical System realizes its functions in the physical world. That means that the embedded system software and hardware computational performances, safety and security rules are not always sufficient qualifying the SSP of a Cyber Physical System.

As straightforward illustration is that even a simple PID (Proportional Integral Derivative) controller can't drive efficiently an elastic actuator without considering its dynamic behaviour. By dynamic behaviour, we consider the time and frequency domain of the physical component operations. We also consider that the static attributes of the component are not sufficient to express the dynamic limitations of the system as well as its efficiency (especially energetic) in transient operations. These stability issues grow with the raising orders and complexity of the controllers combined with the mechatronics complexity.

So, at first order designing a performant Cyber Physical System implies to consider at design stage control systems and physical systems in a combined way (mechatronic co-engineering) in order to orient and calibrate the controller design as well as modifying the physical design to achieve safe and optimal performance of the Cyber Physical System as a whole.

In order to achieve optimal performances, it is necessary first to define the key SSP attributes of the Cyber Physical System. While for embedded system the performance is mostly straightforwardly measurable (computation time, delays, power consumption) and the safety and security covered by design rules and processes, the full Cyber Physical System SSP is specific to the product itself. Most of

the time, theses SSP attributes are the key end user/customers observable criteria of purchasing choice (customer or business). As example, productivity and energy consumption will be considered for a production machine, energy consumption, emissions and longitudinal accelerations for a car, uncommanded train door openings occurrence for a train subsystem, operation robustness to SW attacks for a drive…

Once these key SSP performance attributes defined, they must be evaluated during the design process and the right design tradeoff found an actualized. A typical tradeoff is the one between the energy consumption and the function speed.

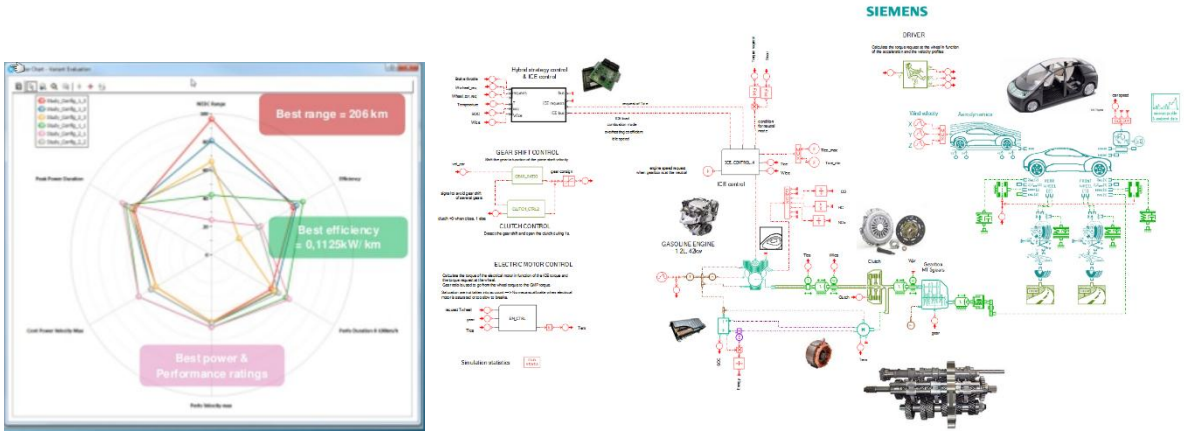

Figure: system performances attributes balancing applied to an hybrid vehicle (Siemens Amesim with Matlab Simulink controllers)

This co-engineering approach between embedded systems hardware, software and mechatronics involve several design interaction points at different livecycle stages of the product, function of the product development techniques.

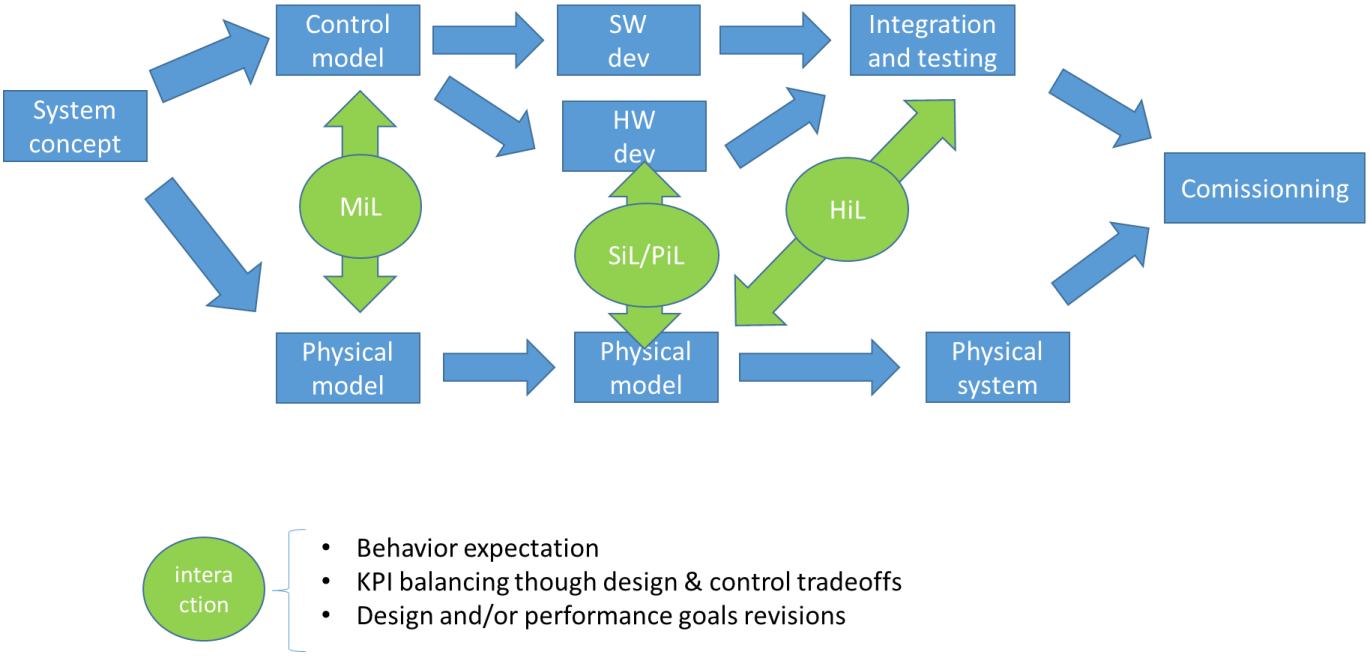

Figure: typical performance interaction points in a CPS development process

First interactions should happen at early stage in a model-based system engineering context (not always applicable), where designers and developers could implement “model in the loop” (MiL) interaction points, coupling control models (like matlab simulink) and physical models (1D, 2D, 3D) to investigate the system behaviour, balance design attributes and design choices. Anyway, despite its apparent simplicity, these approaches must carefully address the traps of heterogeneous models assemblies (effects of discretization’s at the interfaces). Some progresses have to be achieved in this field.

Later in the development process, a Software-in-the-Loop (SiL) and/or Processor-in-the-Loop (PiL) interaction could be implemented. Such interactions are today limited due to the necessity to couple a full virtual execution platform (HW&SW) with a realistic enough plant model (enough to trick the control safety and security itself). Most of the state of the art implementation of such interactions deal only with adapted software coupling (for SiL) and HW emulators couplings (SystemC, QEMU…). In the future, such interactions will became more and more complex as well as critical for the system optimal performances. Such interaction will require significant technological progresses on the coupling technologies to be fully relevant. This is also the stage where the most detailed plant models will be used.

The last CPS development interaction point is the coupling of the physical control system with the virtual plan (Hardware in the Loop) to evaluate, calibrate and final tune the controllers prior to their integration and commissioning. At this stage this interaction could also drive some plant modification too. Here, the main drawback is the realtime execution constraint which limits the plant models representativity (fixed step, RT computation).

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Integration and Testing	System-level safety requirements management with coupled controller (HiL or FMI or other) and virtual environment in Amesim	System-level safety requirements with embedded software and dynamic mechatronic simulation	Improved testing (coverage, unforeseen SoS behaviors…), faster commissioning and improved operational diagnostics	Rail
Integration and Testing	qBox (controller) couplings with Amesim (plant)	IAP_10 Safety requirements with virtual control platform and dynamic mechatronic simulation	Improved testing (coverage), faster commissioning and improved operational diagnostics	Industrial drive

4.4 Asset and Artefact Management for Co-Engineering [MDS]

In Aquas project, Magillem brings four tools[3].

Integrating Specification, Design and Documentation

In UC4 and UC5, Magillem brings **ISDD©,** which stands for Integrating Specification, Design and Documentation. The main goal is to provide a unique, integrated software environment that streamlines specification, design and documentation processes during the product design cycle. All data in the project (requirements, specifications, metrics, software, test cases, etc.) are linked, all stakeholders are aligned. Indeed, any change in the project may generate impacts: they are detected and identified; experts are notified. Key elements are then brought to the forefront to facilitate analysis of these potential impacts. Magillem proposes to organize interaction points when a change reveals the need to balance Security, Safety and Performance (SSP).

Magillem Content Publisher

A good complement to ISDD© is **MCP** (Magillem Content Publisher): it is a comprehensive Digital Publishing suite that facilitates the review, reuse and repurposing of content created in many heterogeneous formats. For instance, it features a set of powerful metadata and a semantic linking framework that enable to synchronize content between electronic system design and documentation workflows.

Magillem Virtual Platform

In UC4, thanks to Magillem's technologies, SystemC models are wrapped in IP-XACT, a format that defines and describes electronic circuit designs allowing to use them in EDA tools like **MVP** (Magillem Virtual Platform). Thus, it is easy to connect each IP in order to realize the design. Moreover, using IP-XACT standard enables to use all its advantages: automated configuration through vendor-neutral scripts, exchange of complex components libraries between EDA tools, etc. It also enables the possibility to defines properties for each element which can have an impact on performances (number of cores, core frequencies, memory available, etc.). Finally, MVP generates all mandatory files to run the simulation (netlists, CMake files…). Furthermore, the full integration of MVP with ISDD© can give an edge during the debugging phase.

Magillem Platform Assembly

In UC5, Magillem proposes **MPA** (Magillem Platform Assembly): also using the IP-XACT standard, it is possible to use this EDA tool to design hardware platforms (components with their connections, FPGAs, memory maps…) and link them to the rest of the project: specifications, test plans…

In order to strengthen the capability of ISDD, Magillem uses standards (ReqIF for requirements, DITA for documentation, IP-XACT for electronic circuit designs…). Therefore, the favored method to integrate with the other tools of the project is to leverage their standards: SysML, Marte, UML, OpenPSA, SACM, GSN, CVSS… Thus, the process that will be built during AQUAS will be agnostic of the tools used.

4.5 Modelling of security requirements and properties, and verification through static code analysis

In T4.1, CEA proposes tooling for process support of the model-based static code analysis method for software/security co-engineering. The tooling will be based on existing Papyrus and Frama-C tools. Frama-C is a static code analyser that takes as input C code with annotations written as comments in

[3] In fact, MVP and MPA are packed together in our EDA suite.

ACSL, a properties specification language. Papyrus is a UML modeller that can be customized for DSMLs and supports MDE tooling such as code generation.

The goal is to bridge the gap between low-level static code analysis with high level requirement and architecture models made with partner tools or Papyrus itself.

In T4.1, CEA will study interoperability (import/export) features necessary to import/export requirement, system and software models of the use-cases, related to other tools working on the use-case, into/from Papyrus. CEA will also implement the necessary DSMLs in Papyrus for specification and specification interoperability. For example the SysML-Sec profile may need to be implemented in order to import existing security requirements. Finally, CEA will extend to its code generators to support the generation of ACSL annotated C/C++ code for static code analysis, and therefore relate such analysable code and analysis results to the requirements, system, and software architecture models.

In tasks T3.1 and T3.2, for the railways application, the WP plug-in of Frama-C will be used for verifying the conformity of the C-functions (issued from the code generator of the Atelier B) with their specification expressed in B0 refinement language. To reach that objective, the translation of B0 specification to ACSL specification has to be studied, and will be achieved as part T4.2. In T4.1, CEA will specify the infrastructure to support the interaction of Atelier B and Frama-C, as well as the infrastructure to measure the performance of such an approach.

The implementation of tools specified in T4.1 will enable us to apply the software/security co-engineering method described in D3.1. The following figure is an excerpt from the deliverable that sums up the method:

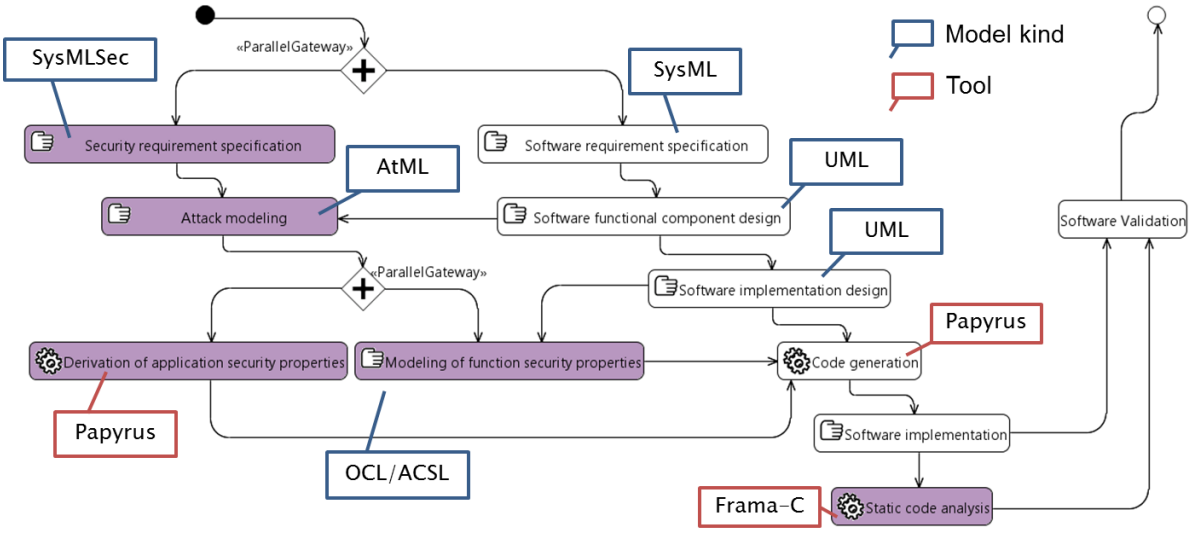

Figure: Co-engineering method

As mentioned previously, CEA also wish to bridge the gap between such a method and models produced by partner tools. In terms of cross-domain interactions, the following table shows those that are planned.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Requirement	Security requirements modeler	Interoperability between system-level functional and non-functional requirements	Improved interoperability and traceability between system-level requirements	Medical
Implementation	Code/ACSL properties inference and generation too	Conformance between system-level requirements and software-level properties	Improved traceability and conformance between software-level properties and system-level requirements	Medical
Implementation	B0 specification to ACSL generator	Conformance between B0-level requirements and software-level properties	Improved trustworthiness of conformance of C code generated from B0 specification	Rail

4.6 Modelling and Analysis of Co-Engineering Requirements [Intecs]

One important aspect required to properly support interaction points regards the modelling and analysis of co-engineering requirements across different tools. One example comes from the SESAMO project, where Medini Analyze and CHESS tools have been integrated to support safety requirements and components traceability from system to software design phases and then to analyze the possible impact of performance requirements (via worst case response time analysis) upon the safety requirements.

In the AQUAS project, and more properly in T4.1, the goal related to the CHESS tool is to enrich the aforementioned support in different dimensions, like extending the CHESS-Medini tool integration (e.g. by supporting in CHESS the automatic creation of diagrams starting from the architectural model imported from Medini) and by adding support for security requirements definition and analysis.

Concerning security requirements, other tools proposed in AQUAS are focusing on security requirements definition, like the Security Development Life cycle management tool (SSLDC) proposed by TrustPort (see Chapter 4.13): the foreseen goal in AQUAS is to be able to connect security requirements derived by SSLDC with the CHESS design and analysis tool. For this, two possibilities have been identified: (i) import of SSLDC requirements in CHESS and (ii) usage of traceability support to trace requirements available in SSLDC to the CHESS model entities (e.g. components, components contracts [Benevniste, 2012]). The first solution, import of requirements in CHESS, allows the representation of the requirements derived with SSLDC in the CHESS environment, basically by using the SysML support for requirement modelling possibly enriched with dedicated stereotypes to represent requirements properties. The requirements, once available in CHESS, could then be traced to the CHESS modeling entities, still by using the standard SysML support regarding traceability links. The other identified possibility (ii) foresees the usage of a dedicated traceability tool which allows the

modelling of traceability links between the CHESS modeling entities and the requirements available in the SSLDC and their storage in a dedicate traceability model. Solution (ii) avoids duplication of information (i.e. the requirements) between CHESS and SSLDC, while leaving the focus on the traceability relationships definition and usage.

The goal of CHESS is to support analysis of requirements, by using the analysis provided by CHESS itself or provided by external tools integrated with CHESS. First, requirements (defined/imported in the CHESS model or available from an external tool) have to be formalized in CHESS by using dedicated (modelling) languages. In particular, performance requirements, like for instance the ones related to worst case response time and deadlines, are formalized by using the OMG standard MARTE modelling language. Safety requirements formalization is supported by using the contract-based approach, in particular by using the OCRA temporal language [OCRA]; moreover formalization of safety requirements concerning failure modes of components is supported by the dependability profile coming with the CHESS modelling language [CHESSML].

MARTE properties can then be used to feed schedulability and worst-case response time, by using the integration with the MAST tool [MAST].

In the context of the ATM use case, the CHESS toolset can support the modelling of the IMA (and ARIJNC-653) concepts of partitions, processes and operations and analysis of the two-level scheduling regime assumed in the IMA architecture. CHESS provides means to help the designer configure the system and compute a partition schedule for each available processing unit. The model and the generated schedule can then be used as inputs for a response-time analysis engine that calculates the worst-case response time of each task and therefore, assesses the overall system schedulability.

Information provided about failure (propagation) models can be used to executed failure propagation analysis [FLA], directly supported by the CHESS tool, or quantitative state-based analysis available via integration with the DEEM tool [DEEM].

OCRA contracts can also be used to express security requirements; this is currently under study in the context of the AMASS Ecsel project [AMASS]. Furthermore in AMASS there is an ongoing study focusing on CHESS FLA failure propagation analysis adaptation for security: this requires an extension of the CHESS dependability profile to enable the representation of security aspects, like security threats, and this extension will be evaluated by considering the needs arising in the AQUAS projects too. For instance, the concept of security threats is also considered by the SSLDC tool, so regarding CHESS-SSLDC interaction, in addition to the aforementioned traceability between security requirements, traceability between security threats definition between the two tools will also be investigated.

To enrich the set of possible analysis to be used, an integration between CHESS and the SafetyArchitect tool from All4Tec is foresees (see Chapter 4.9), so by sharing the information about the system architecture, safety and security properties modelled between the two tools; this would allow to enable the different analysis provided by CHESS and SafetyArchitect while relying on a common and consistent set of information.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Requirements/ Modeling/Analysis/ Implementation	Import of Safety/Security Requirements (SSDLC) Safety/Security State Based Analysis (SAN)	Analysis of the interaction of SSP properties in the architecture/ design	Impact and trade off analysis	ATM
Requirements/ Modeling/Analysis/ Implementation	Exchange of requirements a system architecture models (medini)	Conformance between system-level requirements and software-level properties	Improved traceability and conformance between software-level properties and system-level requirements	Industrial Drive
Requirements/ Modeling/Analysis/ Implementation	Exchange of system architecture with safety and security models (All4Tec tools)	Interaction of SSP properties in the SW architecture	Impact and trade off analysis	Space

4.7 OPENCERT [tecnalia]

Tecnalia brings Eclipse/Polarsys OpenCert to AQUAS. OpenCert helps integrating the engineering activities with the certification activities from early stages. During AQUAS, Tecnalia will contribute on the specifications of tool extensions based on WP3 inputs to represent safety, security and performance co-engineering requirements.

At the core of OpenCert functionalities we can find information management capabilities regarding standards and regulation. That means that the applicable reference frameworks for the use cases can be stored, retrieved, categorized, associated, searched and browsed in a modelling framework. For example, the following figure shows an excerpt of the European Cooperation for Space Standarization standard ECSS-E-ST-40C which deals with general requirements for software. In this specific excerpt, the figure shows reference activities, subactivities as well as reference artefacts that are defined as inputs and outputs of the activities.

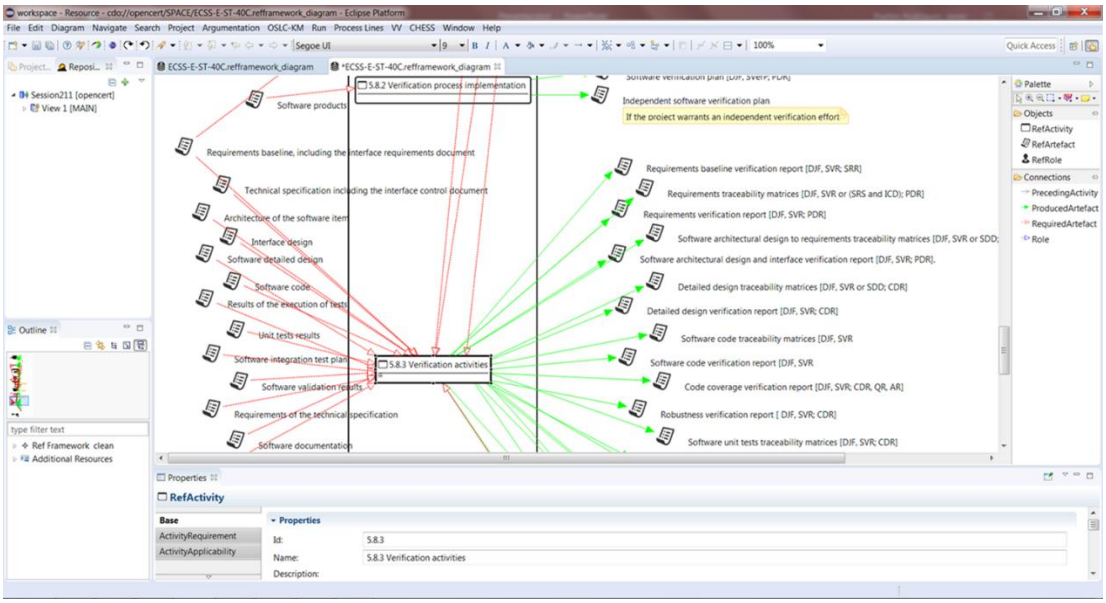

Figure: Excerpt from ECSS-E-ST-40C

In AQUAS, it is of special interest to extend the tool to investigate the interplay of quality criteria at the level of standards. Our intuition is that interaction points can be expected and planned according to the processes, activities and artefacts defined in the reference frameworks that applies to a system (e.g., an activity related to performance has a sub activity related to security).

Apart from the modelling of standards, OpenCert supports the management of the assurance project. This is possible through the modelling of assurance cases (GSN notation is used), evidence management, assurance process management, and global monitoring of the compliance with standards and regulations. OpenCert also tries to automate traceability, compliance checking, assurance process planning or metrics management. It also has functionalities to see the effective progress of the work and level of compliance as well as functionalities for effective reuse of artefacts from one project to another.

It can be considered that the co-engineering process is supported by the OpenCert tool to some extent. Also, the co-engineering activities need to consider the regulation space affecting the decisions regarding the alternatives in the trade-off analysis. Tecnalia will strength and enhance OpenCert with support to data and artefacts management which are the evidence used in these contexts. Interoperability approaches with other tools will also be investigated.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Concept and system design	OpenCert: Location of safety, security and performance interplays at the normative space and within assurance projects. Argumentations in the assurance projects will be related to results of partners' tools (e.g., ALL4Tec)	Safety and security requirements to comply with standards.	Improved traceability of co-engineering results to certification assets.	UC2 – Medical and UC5 – Space Multicore

4.8 Static and Dynamic Code Analysis in the Co-Engineering Process [BUT]

BUT will contribute to finding suitable ways of supporting safety-security-performance co-engineering in static as well as dynamic analysis tools, reflecting its long experience with both kinds of the analyses. A stress will be on combining various kinds of analysis across the safety-security-performance boundaries (e.g., by exploiting results of memory safety analysis from tools such as Forester or Predator, combined with dedicated analysers for performance of integer programs such as Loopus, to support performance analysis of more complex programs; or using means for dynamic safety analysis of concurrent programs from the ANaConDA tool to facilitate security analysis), supporting modular or focused analyses to be as efficient as possible at interaction points, and design of techniques automatically detecting performance degradations between different versions of software in the Perun tool whose development recently started at BUT. Moreover, BUT will also contribute to finding suitable ways of measuring the obtained assurance guarantees with a particular stress on the area of concurrent systems handled by the ANaConDA tool. The tools above mentioned are briefly characterized below.

ANaConDA

ANaConDA is a framework provided by BUT. This framework tackle adaptable native-code concurrency-focused dynamic analysis built on top of Intel PIN instrumentation tool. The main goal of the framework is dynamic analysis of multi-threaded C/C++ programs on the binary level. The framework provides a monitoring layer offering notification about important events, such as thread synchronisation or memory accesses, so that developers of dynamic analysers can focus solely on writing the analysis code. In addition, the framework also supports noise injection techniques to increase the number of interleaving witnessed in testing runs and hence to increase chances to find concurrency-related errors.

In order to support the co-engineering process, we plan to contribute by enhancing ANaConDA framework by focussed strategies. The goal of these strategies is to provide a fast feedback to the engineers targeting the recently modified parts in the project under development. Contrary to whole project testing, the focused analysis can be much more efficient in running time and subsequently

allows immediate safety bug-discovery incidentally introduced together with changes targeting security (resp. performance) improvement.

Static tools for memory Safety

Dynamically allocated memory space and complex pointer manipulations are a frequent cause of nasty errors that are easy to cause but often difficult to discover. The BUT provides two tools targeting the memory safety-Predator and Forester.

Our goal is to enhance techniques behind our memory safety-static analyser to support composite verification by means of abduction methods. These methods compute contracts for particular functions and allows one to perform a targeted static analysis of a small part of the code without evaluating the whole project and providing a test harness. Hence it can target only the modified functions in the code and reuse as much as the so-far computed contracts for the unmodified parts of the code. Subsequently, one can quickly discover memory safety errors introduced within the last project updates.

Performance

Static program analyses targeted at automated derivation of various program complexity measures (resource bounds) is nowadays very active and demanding research area. One of the most light-weight, scalable, and practically applicable approaches is the approach implemented in the Loopus analyser. Loopus was originally designed primarily for programs with integer variables at the Vienna University of Technology and recently extended in collaboration with BUT to handle also pointer manipulating programs in the Ranger tool based on the collaboration of Forester (memory safety) and Loopus tools.

The Loopus already provides composite resource analysis of particular functions, which can be employed in the co-engineering process. Moreover, by allowing composite static analysis in our memory safety tools, we can easily extend Ranger to composite complexity analysis of pointer programs as well.

The Perun framework represents a dynamic analysis targeting performance. It provides a generic framework for collecting various performance measures (running time, memory consumption, etc.) for various input data. The obtained measures can be subsequently analysed by means of statistical methods, allowing one to, e.g., derive performance models of the analysed programs.

Our recent goal is to enhance Perun framework by automated detection of performance degradation. The principle is based on comparison of performance of the current version of the project with an older referential version (versions). Moreover, we like to compare also performance of particular modules of the project to their referential versions. By this, we can detect a potential degradation immediately.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Integration and Testing	Art2kitekt or code repository with Forester/2LS/Predator x Loopus/Ranger	General code-level safety requirements	Improved safety (Forester/2LS/Predator) and performance (Loopus/Ranger) properties	Space Multicore
Integration and Testing	Art2kitekt or code repository with ANaConDA/Perun	Safety requirements with created test harness	Improved safety and performance properties	Space Multicore
Integration and Testing	Art2kitekt or code repository with ANaConDA with TrustPort penetration tests	Safety and security requirements	Improved safety and security properties	ATM

4.9 Safety and Security Co-engineering Including Performance [All4Tec]

ALL4TEC has proposed in deliverable D3.1-Section 2.10 a tool-based methodology for safety and security co-engineering. As presented in figure below, the process consists in exploiting partner's tools based on UML/SysML language (e.g., Papyrus/Chess or Capella) for system modelling and All4TEC tools (Safety Architect and Cyber Architect) for Safety and Security co-analysis.

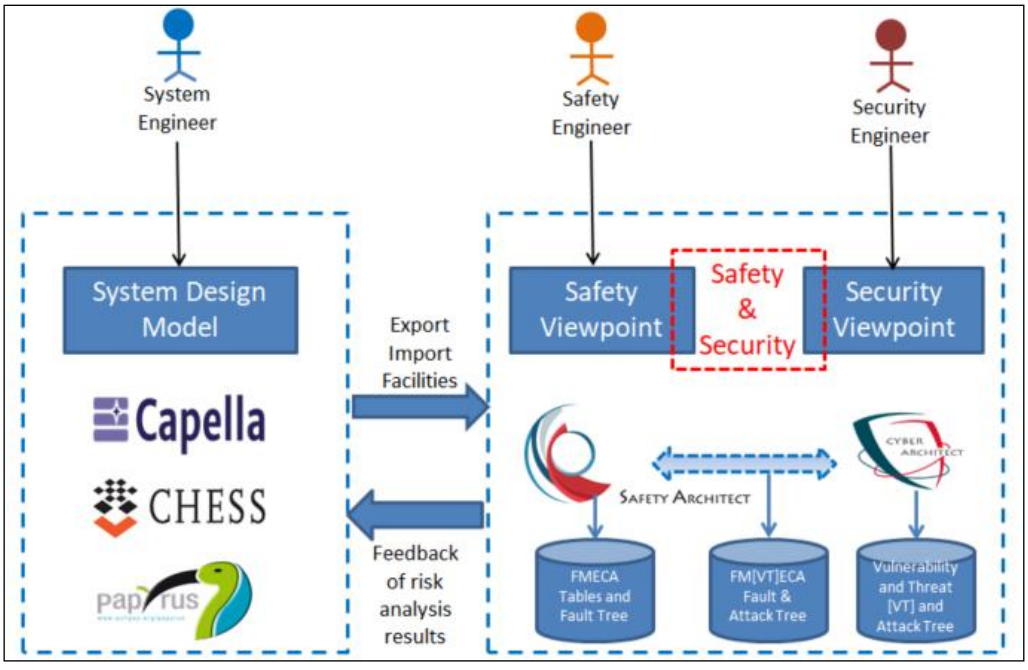

Figure: ALL4TEC Safety and Security co-engineering methodology

The seamless interoperability in this tool-based process allows decoupling the system architecture model from safety or security analysis Thereby, each engineer (System Engineer, Safety Engineer or Security Engineer) can solely focus on his concerns, with dedicated tools and terminology. For

example, Safety Engineer could use an interface between Capella and Safety Architect to generate system or component level fault trees or FMECA tables. Secondly, the development of a specific-concern viewpoint in a tool dedicated to another concerns allows co-engineering between these concerns. For instance, with the Security viewpoint in Safety Architect, Safety engineer can use the results of security analysis realised in Cyber Architect (e.g., Attack Tree) to generate a merged Fault tree and Attack Tree for assessing the influence of security-relevant events on safety top event.

ALL4TEC also proposes to expand this model-based safety and security analysis by including performance concerns. In addition of ALL4TEC's safety and security analysis tools, the proposition is to use performance-based tools to analysis the impact of safety and security requirements on performance issues. For example, system design models enriched in Safety Architect or Cyber Architect with Safety or Security requirements can be imported in these dedicated tools (for example, CHESS supports End-to-End Response Time Analysis, which can be used for the performance of components interactions). Performance analysis with respect of Safety or Security requirements could lead to the definition or detailed technical Safety or Security requirements, which can in turn be exported back to Safety Architect or Cyber Architect.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Concept and system design	Safety Architect with Papyrus/CHESS and OpenCert	Safety and Security analysis with A4T tools and interactions with partner tools	Tools support to improve Safety, Security and performance properties	UC2 - Medical
System design	Safety Architect with Papyrus/CHESS and Art2kitekt	Safety and Security analysis with A4T tools and interactions with partner tools	Tools support to improve Safety, Security and performance properties	UC5 – Space Multicore

4.10 Performance evaluation before implementation [TRT]

In general, many legal mappings potentially exist for an application, but the goal is to find one or a few mappings (and associated scheduling policy) that fit at best a set of Quality of Service (QoS) criteria, in particular on computing throughput and latency. Those real-time requirements are related to the behaviour in time of all elements of the machine that work simultaneously to run the mapped application. In some cases, when the machine is available and has the necessary monitoring hardware, some figures of throughput and latencies can be measured on target.

However, in particular with today's complex heterogeneous SoCs, finding the right mappings needs to envisage several forms of implementation of some nodes in the application graph (e.g. targeting general purpose or FPGA) before achieving a well-balanced usage of resources. In such cases, developing the full executable code for each possible implementation is rapidly tedious. Using a platform simulator that quickly approximates the behaviour in time of the machine for each mapping variant simplifies largely the process, which speeds up the iterative process of finding efficient

mapping. In addition, such a simulator gives potential access to timing records of any resource of interest in the machine (bus traffic, memory accesses...), giving an enhanced visibility of bottlenecks.

SPEAR DE

Spear DE is a parallel framework which can be used to represent a computing intensive application as a non-cyclic graph composed of nodes called “tasks” (application model), producing and consuming multi-dimensional arrays of data. A task represents a function call executed into a nest of loops with linear accesses to and from output and input arrays. Two ways exist to create this application model: the user can create from scratch this application by hand (graphically) or the user can import the application model from a C99 code respecting some coding rules.

Spear DE also considers another model about the execution platform. This model provides a representation of the target system, which may not be the exact representation of the physical architecture. Few objects are enough to model such an execution platform: processor, memory, busses and interconnection link objects. Modelling memory hierarchies is of paramount importance when considering multi/many-core architectures.

Graph nodes from the application model are then graphically mapped onto the execution platform model of the execution platform that provides a topological view of CPUs, memories and communications. Some nodes can be mapped on several processing elements having one or more loops of the nest partitioned. If the memory is distributed or data reorganisation is needed, communication nodes have to be inserted in the application graph. Spear DE helps the programmer to identify where the communication nodes have to be inserted and computes them automatically.

Spear DE provides a set of graph transformations and allocations that can be used by the designer to quickly create legal mappings of an application to a target computing platform. Among other verifications, Spear DE guarantees that data will be correctly deployed in time into the right memories of the machine, not exceeding the memories capacity, and made available to the processors that use them. The mapped application summarises all the design choices of the programmer. As output, Spear DE produces an XML file which can be further used to generate the intermediate representation (IR) to be used for the back-end tools (e.g. performance simulator), or to generate the final code to be compiled for the actual target architecture.

Architecture simulator

A simulator platform has been developed in PtolemyII environment. The goal is to add to the topological view of the platform model used for mapping the necessary building blocks to represent the behaviour of time. These include additional timing information for parameterisable CPUs, memories, and communication elements, and sequencers that hold and schedule abstracted views of programs within CPUs. The simulator has also the possibility to represent resources interferences: this happens in complex SoCs, where an activity on one resource often creates an access conflict to another shared resource or requires some pre-empting service from a control resource that was busy doing another activity.

Once the platform has been captured within the simulator, candidate mappings (from e.g. Spear DE) can be deployed on the platform model. The simulation itself is run under the control of PtolemyII’s Discrete Event director, where time-stamped events are sent between actors of the simulation, emulating the behaviour in time of the target platform.

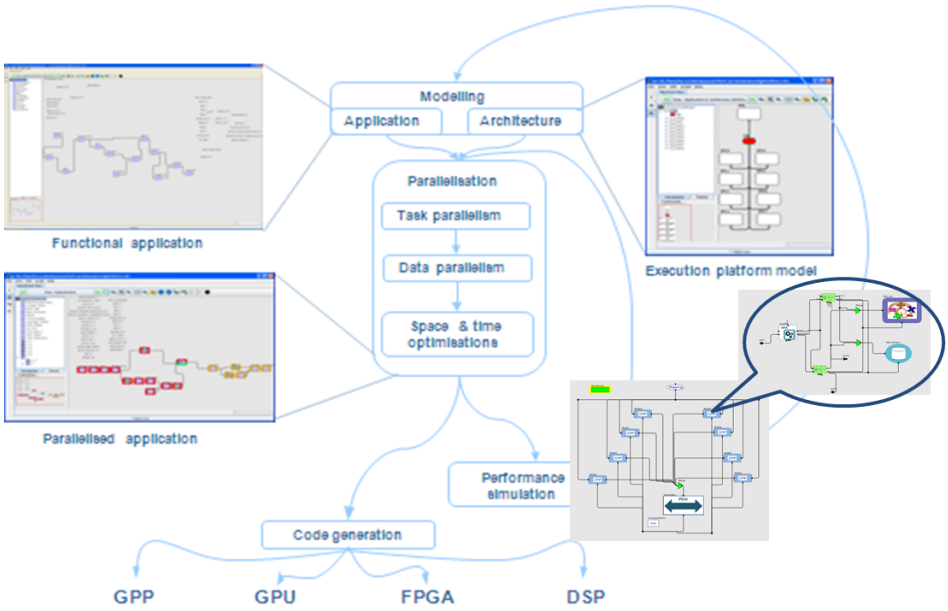

Figure: Plattform

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Implementation	Architecture simulation with SPEAR DE and HLS generation	Conformance between system-level requirements and performance	Generate and place code matching performance requirements	Space Multicore

4.11 Timing behaviour verification for Performance and Safety at early design phases safety [TRT]

It has always been a challenge to introduce timing verification early into the industrial development process as the inputs required for the verification, in particular the worst-case execution time and the system behaviour description, are moving targets all across the different development process phases. Thanks to the introduction of model-based methods (and the ability to express non-functional properties with dedicated, concern-specific viewpoints) in the industrial development process, this goal seems to be in the reach. Starting from very high-level system architecture and rough timing allocations, the timing verification has to be refined at each step of the project (architectural design, detailed design, coding, unit test and software validation phases) down to concrete timing measurements on the final delivered system.

4.11.1 Time4Sys

To apply the current state of the art of timing behaviour verification techniques, a major problem still persists: model-based timing verification techniques, such as scheduling analysis and simulation, are often not directly applicable to conceptual design due to the semantic gaps between their respective models. Solving this issue is essential to break the remaining walls separating model-based timing

verification from the development process of real-time embedded systems, and to enable its use in the industry.

Time4Sys is a timing performance framework that fills the semantic gaps between the design models of real-time systems and the models of timing verification tools. Time4Sys is composed of two building blocks (the Design and the Verification pivot models) as well as a set of transformation rules between them. Both pivot models are based on the Time4Sys meta-model.

Time4Sys uses a subset of the MARTE OMG standard as a basis to represent a synthetic view of the system design model that captures all elements, data and properties impacting the system timing behaviour and required to perform timing verification (e.g. tasks mapping on processors, communication links, execution times, scheduling parameters, etc.). Time4Sys is not limited to the use of a particular design modelling tool and environment. It can be connected to various environments and languages such as UML, SysML, AADL, or any other proprietary environment (e.g. Capella).

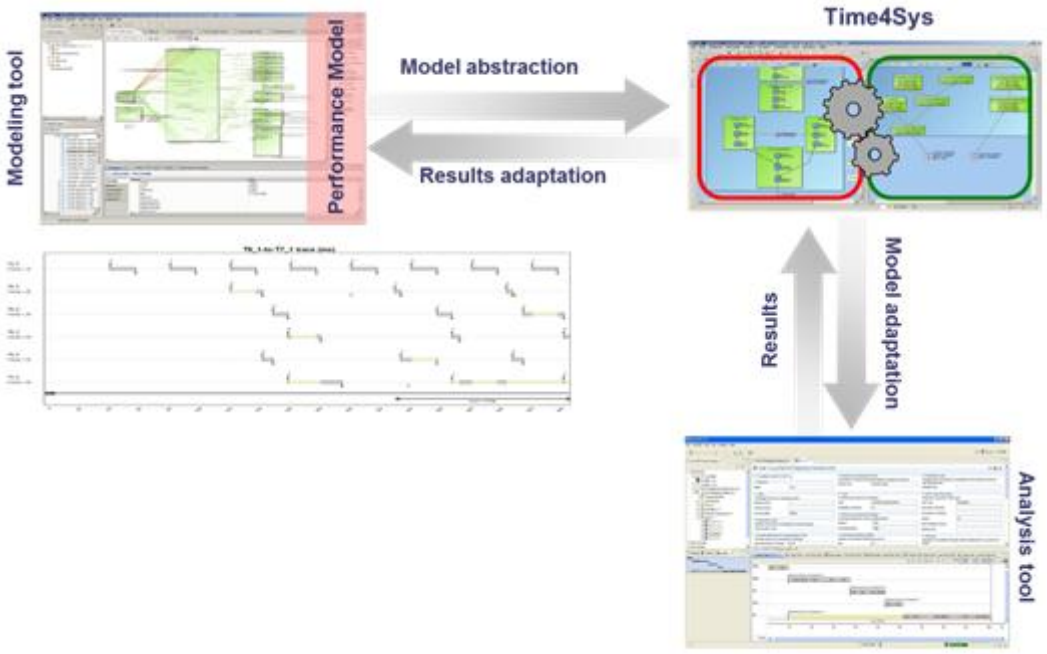

Figure: Time4Sys: Bridging the gap between design tools and verification tools

Scheduling analysis and simulation are seldom directly applicable to the conceptual design models in general and to Time4Sys Design models in particular due to the semantic mismatch between the latter and the variety of analysis and simulation models known from the classical real time systems research and represented by academic and commercial tools. Transformation rules are therefore required to generate a Time4Sys Verification model preserving the timing behaviour modelled in the corresponding Time4Sys Design model, while ensuring the compatibility with the variety of existing timing verification tools. After timing verification in the selected tool, results are injected in Time4Sys Verification model. Then, they are translated to be compliant with the original design model and injected back in Time4Sys Design.

4.11.2 Tempo Verifier

Many years of research on timing behavior verification at early design stages have conducted TRT to capitalize our experience and developments into a single tool called Tempo Verifier. This tool has gained in maturity over the years, taking benefit from our different collaborations

within Thales GBUs (through Use Cases) but also through projects with partners across Europe and direct engagement with academia.

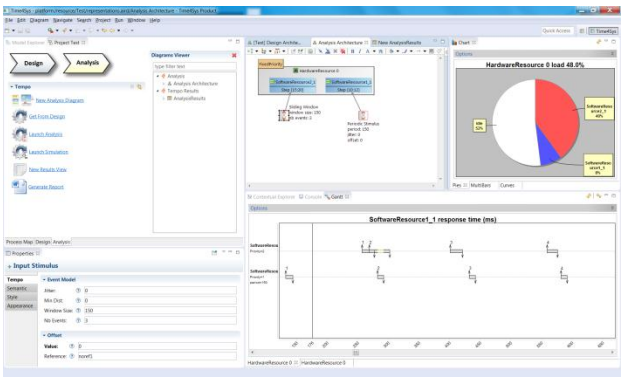

Figure: Tempo Verifier

Tempo Verifier is a timing verification tool which aims to assist designers and safety engineers of Real Time Systems by

- Early dimensioning of your system

  Quickly try different architectures, mappings and priority assignments, and see the impact on the latencies.

- System development life-cycle follow-up

  As your system is being developed, models can be updated with new measured values to check that timing requirements are still respected.

- Bottleneck identification

  Thanks to worst case latencies computation one is able to find critical situations where deadlines are missed. The ability is provided to point out timing problems origin with graphical illustrations. The graphical illustrations will assist the designers to correct its architecture and will assist the safety engineer to proof the correctness of the architecture according to the time requirements.

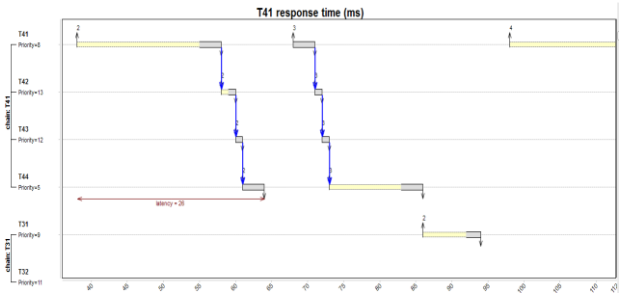

Figure: Worst case latency scenario proved

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
System and software Design	Time4Sys with Art2kitekt	Early performance and safety verification	Performance verification according to performance and safety requirements	UC5 Space Multicore
System and software Design	Time4sys with Chess/Papyrus	Safety and Performance analysis and exchange of performance requirements and system models	Tools support for verify and to exchange safety, security and performance models[2]	UC5 Space Multicore
Test and integration	Tempo Verifier with Code repository and/or execution logs generate by Art2kitekt	Integration of performance multi-core verification	Tool support to improve Safety, Security and performance properties	UC5 Space Multicore

4.12 Workflow Automation for Multi-Concern Assurance [AIT]

Automating workflows across multiple iterations of system development helps to accelerate the development flow while avoiding wrong or incomplete process chains caused by human error in the case of manually managing the activities. Workflow automation is therefore essential from the efficiency perspective as well as with view to a high and controlled quality level.

AIT has developed the tool WEFACT (Workflow Engine For Analysis, Certification and Test) from a web-based test bench in the FP6 IP DECOS towards a multi-purpose workflow engine in Artemis project SafeCer. Since then, in Artemis project EMC^2, workflows have been extended towards security analysis, and an Eclipse RCP-based new version 2 of WEFACT has been developed in ECSEL project AMASS. Now in AQUAS, the scope of the tool is extended towards a workflow engine for multi-concern assurance, and it is integrated with the AQUAS platform. The following figure shows a screenshot of the Eclipse-based WEFACT tool.

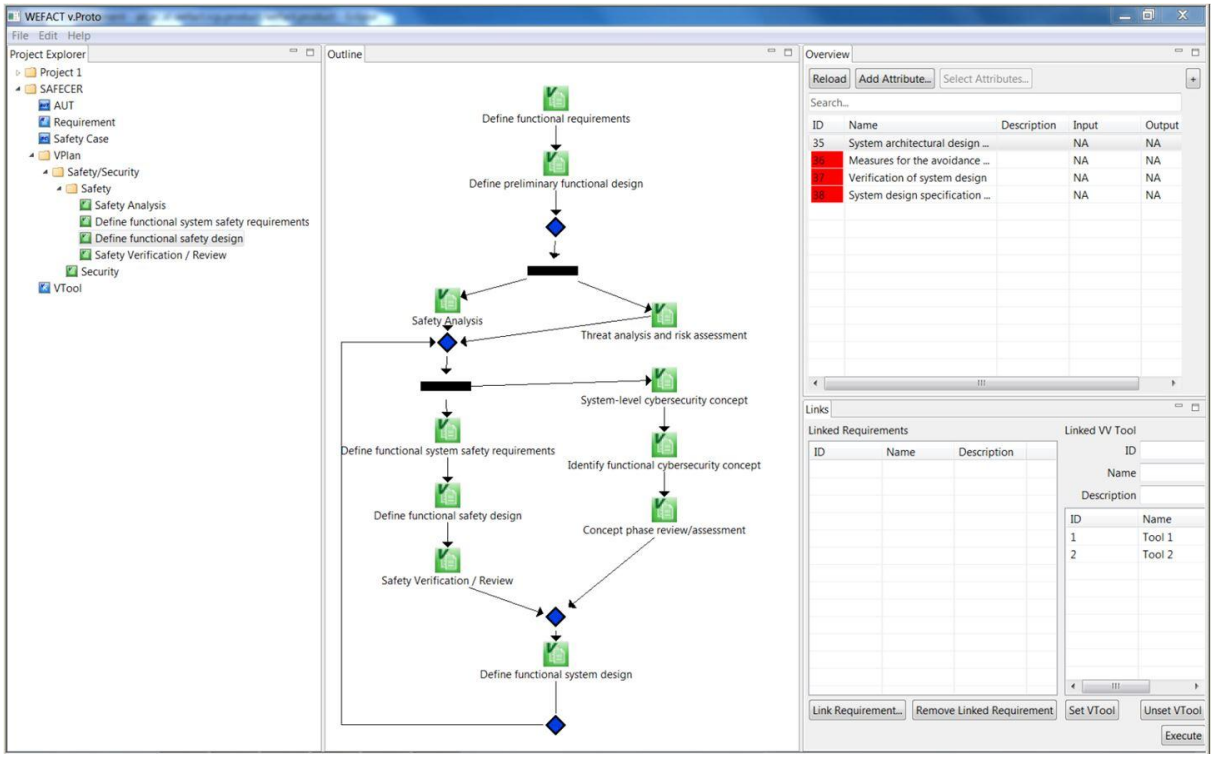

Figure: Screenshot of the Eclipse-based WEFACT Version 2

The essential goal of the multi-concern assurance process is to demonstrate in a credible manner that the requirements associated with the different quality attributes are fulfilled. This includes the argumentation with all assumptions and contexts, and finally the evidences for the arguments. From this perspective, multi-concern assurance is the linear superposition of the argumentations for all quality attributes.

One might argue that there are interdependencies between these multiple quality attributes. This is, of course, correct for the development processes, but the final design contains all decisions related to these attributes, and the final assurance provides separate, distinct evidences for the individual quality attributes. It shall be noted, however, that sometimes one evidence can support more than one quality attribute. For instance, following the MISRA standard guarantees a good level of safety as well as security from the code quality perspective.

The AQUAS approach allows separate as well combined processes for the individual quality attributes under consideration (e.g. safety, security, performance); at least in the case of separate processes, after performing them, an interaction point is needed to analyse the trade-offs between the potentially contradictory quality attributes treated in the separate parallel processes.

WEFACT allows to model process flows with defined predecessor-successor dependencies and also with forking for enabling parallel processes. WEFACT is based on a database (currently PostgreSQL), but for efficiency purposes work products used as inputs or outputs of the process steps can be stored in repositories, e.g. svn. The process model can be defined within WEFACT, but also models created with EPF (Eclipse process framework) can be imported (and exported from WEFACT again if needed). Defining the process steps, which represent the verification of (process or product) requirements, means in most cases assigning a tool to perform the process step automatically. WEFACT contains an execution process and can automatically start activities, which have not yet been performed successfully. After successful completion, WEFACT marks the process step and the associated requirement as fulfilled, which represents a particular evidence for the assurance case.

WEFACT supports different types of process execution, manual and automatic. While manual tools require the user to save the results to a specific location, automatic tools return the results which are consequently evaluated and stored. The outputs of the executed processes are typically stored in a centralized SVN.

After the evaluation of the Process Result, the status of the executed Process and associated Requirements is modified.

The figure below shows the WEFACT Activity Diagram.

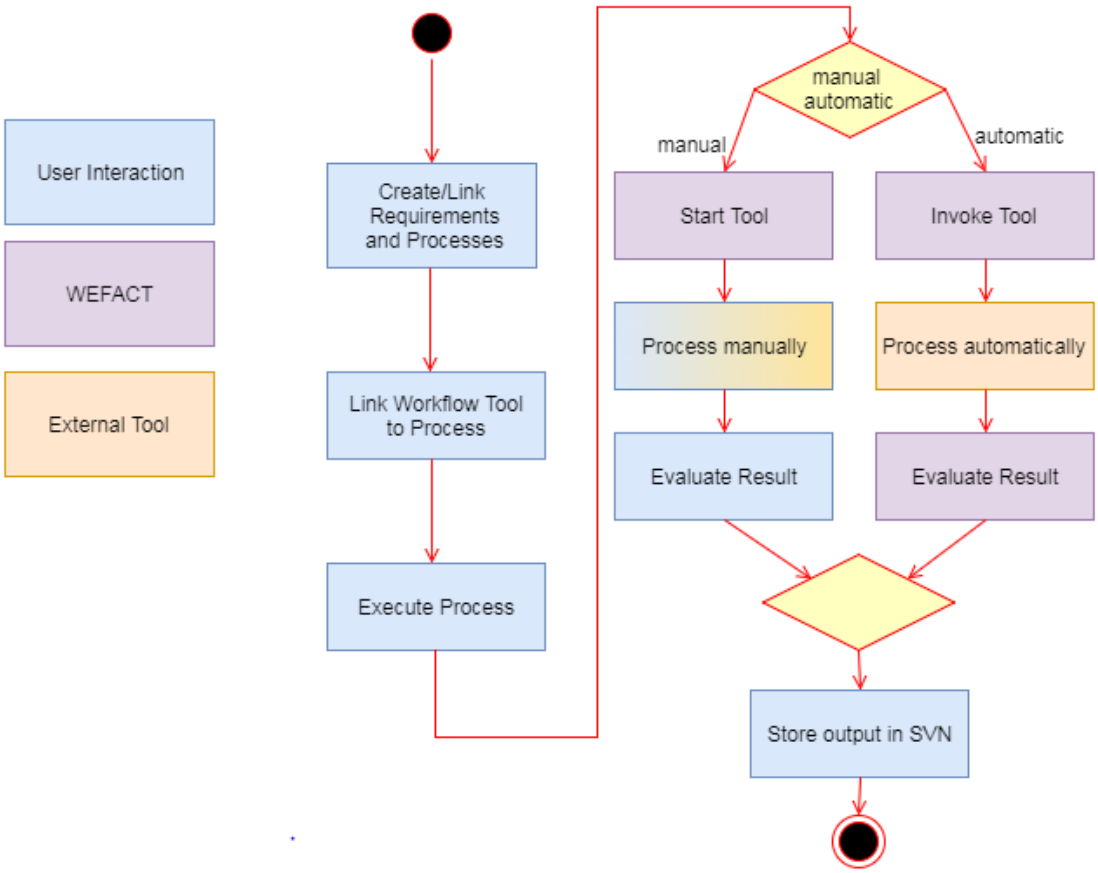

Figure: WEFACT Activity Diagram

The above explained capabilities of WEFACT allow to instantiate exactly the process flow structure which is needed according to the AQUAS methodology. **Error! Reference source not found.** shows an xample for a part of a multi-concern assurance process. Basically, it includes product as well as process assurance.

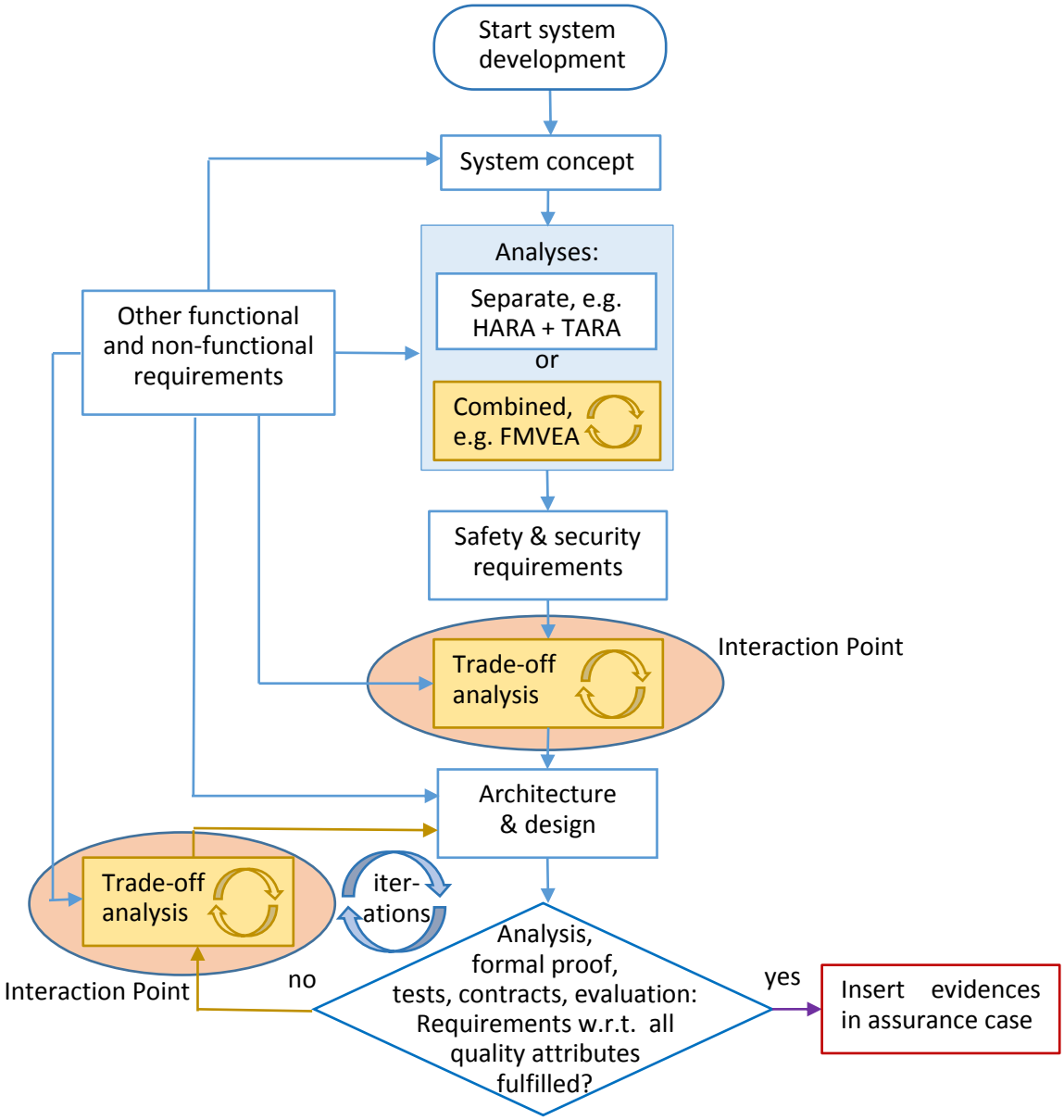

Figure: The multi-concern assurance process

The process is usually an iterative one, therefore the exemplary interaction points in the figure are traversed more than once.

In the Industrial Drive use case, WEFACT shall be applied for controlling the workflow of parallel assurance processes with tools of different partners. It is intended to follow the recommendations of parts of the standard IEC 62443 and combine the processes with the ones needed in conformance with IEC 61508.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Concept and system design	WEFACT with FMVEA, SSDLC, Medini Analyzer, SAG Performance Analysis	Safety, Security and Performance Analysis with SAG, TP, AMT and AIT tools and interactions of WEFACT with partner tools	WEFACT supporting the automated workflow of first performing Safety, Security and Performance Analyses, and finally starting interactions between the quality attribute-specific analysis processes	UC4 – Industrial Drive

4.13 Sub-System Hardening of Communication Protocols [TrustPort]

In Task 4.1 TrustPort will contribute to the specification of co-engineering criteria, indicators and requirements for security of communication protocols with aim at sub-system hardening and increasing of overall level of security. For ensuring the required availability, reliability and recoverability of the systems we plan to use Security Development Life cycle management tool (SSLDC).

In SSDLC tool development we are going to implement requirements of norms and best standards (ISO/IEC 62443, NIST FIPS, STIG, OWASP, CIS Benchmarks etc.) and generate security requirements in a form of a standardized outputs tailored for individual UCs as inputs into other tools used in co-engineering processes.

Trustport's aim is to define general set of security requirements with respect of safety and performance, define threat model for the UCs and create a simple tool for implementing the security requirements in a life-cycle of selected UC's – Secure Software Development Life Cycle tool (SSDLC)- and create a CVSS system, scoring severities of potential threats with the aim to suggest it's mitigation. SSDLC approach brings several security aspects into development life cycle and provides:

- General security controls for information systems for effective risk management;
- Flexible catalog of security controls to meet the security threats, requirements and technologies;
- Involving security activities into the whole PLC (creating security requirements, verification of analysis and design from security point of view, developer guidelines, code reviews, verification of security requirements, security assessment);
- Based on generally accepted standards (NIST, OWASP, Microsoft SDL) to treat the various attack vectors;
- Measurement metrics for security control effectiveness.

Therefore, SSDLC tool will help to create security, hardening, testing, and validation reporting guidelines for selected UCs. The SSDLC tool is an environment for defining the current and future security requirements based on variation of standards, recommendations, best practice, and many

others (i.e., ISO/IEC 27034, ISO/IEC 11073-00103 or the frameworks from HITRUST, NIST or FDA/FTC). Connecting the SSDLC with other partners' tools will improve the automation process of PLC (product life cycle). The SSDLC gives a connections and context between security, safety and performance parameters. Compared with static security requirements definition, the SSDLC provides simple future extension and straight integration to the PLC. The interaction of SSDLC with other partners' tools and integration in the PLC process is displayed in figure below.

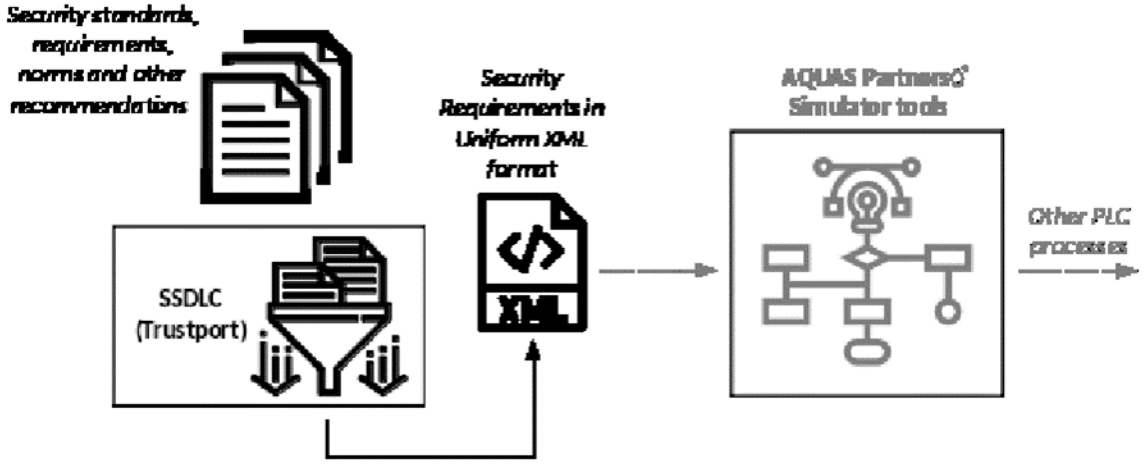

Figure: Integration of Trustport's SSDLC software in the PLC process and interaction with other partners' tools.

PLC phases addressed	Tools Interactions to be implemented	Domains interactions Identified interaction point	Expected SSP benefits	Use case
Security Validation	Cryptographic algorithm validation program, documentation review, hardening, security audit, vulnerability testing	System-level security requirements with methodology according standards and best-practice. Evaluation via simulation/measurements influence of security requirements on performance/safety	Improved testing (security methodology), System-level security agreement, and improved operational faults. Immunity against attract and threads	Industrial drive
Technical Safety/Performance and Security Concept	Standards, best-practice, SSDLC	Security requirements for particular domain and severity of these requirements on performance	Improved future design with focusing on synergy security and performance, testing (coverage), faster commissioning and improved operational diagnostics	Industrial drive

5 Conclusion [AMT]

At this point in the AQUAS project, we started to specify a methodology for co-engineering in the product life cycle that supports qualitative and especially quantitative techniques for the analysis and assessment of safety, security and performance properties, both in separate and integrated ways. For this purpose, it was elaborated to use the Software & Systems Process Engineering Meta-Model (SPEM) to define the process model. It was emphasized that this process model should be considered as an initial version whose finalization in respect to the definition of the process phases and especially of the interaction points will be adopted to the results within the use cases throughout the AQUAS project.

This report has shown furthermore how this general methodology is applied and adopted in the different use cases and what interaction points will be addressed. The application, planned extensions and integrations of the tools provided by the project partners play an important role and have been described in detail, especially the proposed specific functionalities to address foreseen challenges of the demonstrator – including interoperability in a phase and across phases with other tools and use cases.

6 References [all]

[IEC 62443] IEC 62443, *Industrial communication networks – Network and system security,* 2010.

[IEC 61508] IEC 61508:2010 (ed. 2), *Functional safety of electrical/electronic/programmable electronic safety-related systems*, 2010.

[IEC 61800] IEC 61800, *Adjustable speed electrical power drive systems*, 2015.

[Benevniste, 2012] Benveniste A., Caillaud B., Nickovic D., Passerone R., Raclet J. B., Reinkmeier P., Sangiovanni-Vincentelli A., Damm W., Henzinger T., Larsen K.G. Contracts for System Design. Research report RR-8147. Inria. November 2012.

[OCRA] OCRA: "a command-line tool for the verification of logic-based contract refinement for embedded systems", [Online], Available: https://es-static.fbk.eu/tools/ocra/ [Accessed: July 15, 2016].

[CHESSML] CHESS Modelling Language: https://www.polarsys.org/chess/publis/CHESSMLprofile.pdf

[MAST] MAST: "Modeling and Analysis Suite for Real-Time Applications", [Online], Available: http://mast.unican.es/ [Accessed: July 15, 2016].

[FLA] B. Gallina and E. Sefer, "Towards Safety Risk Assessment of Socio-technical Systems via Failure Logic Analysis" submitted to RISK 2014.

[DEEM] DEEM: "DEpendability Modeling and Evaluation of Multiple Phased Systems", [Online], Available: http://rcl.dsi.unifi.it/projects/tools [Accessed: July 15, 2016].

[AMASS] AMASS project: "Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems", [Online], Available: http://www.amass-ecsel.eu/ [Accessed: July 15, 2016].

[STRIDE] The STRIDE Threat Model, [Online], Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)