



## Deliverable 3.2

# Combined Safety, Security and Performance Analysis and Assessment Techniques - Preliminary



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.

The author is solely responsible for its content, it does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of data appearing therein.

DISSEMINATION LEVEL		
	<b>PU</b>	Public
	<b>CO</b>	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.:	D3.2	
Document name:	Combined Safety, Security and Performance Analysis and Assessment Techniques - Preliminary	
Work Package	WP3	
Associated Task	T3.2	
Nature <sup>1</sup>	R	
Dissemination Level <sup>2</sup>	PU	
Version:	2.0	
Actual Submission Date:	10-05-2019	
Contractual Submission Date	10-05-2019	
Editor: Institution: E-mail:	Marwa Gadala, Lorenzo Strigini City, University of London {Marwa.Gadala,L.Strigini}@city.ac.uk	

<sup>1</sup> R=Report, DEC= Websites, patents filling, etc., O=Other

<sup>2</sup> 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)
1.0		First version for internal review	Lead editors: Marwa Gadala, Lorenzo Strigini  Authors: Abel Balbis, Alejandra Ruiz, Antonio Gonzalez, Asma Smaoui, Bernhard Fischer, Chokri Mraidha, Emmanuel Vaumorin, Isaac Moreno Asenjo, Jabier Martinez Perdiguero, Javier Puellas, Jean Godot John Favaro, Jose Manuel Sanchez, Jose Cordero, Ken Sharman, Korbinian Christl, Lorenzo Strigini, Ludovic Apvrille, Marc Born, Marc Sango, Mario Winkler, Martin Matschnig, Marwa Gadala, Maysam Zoor, Olga Dedi, Pavel Mrnustik, Peter Popov, Petr Mlynek, Radek Fujdiak, Ricardo Ruiz, Robert Kaiser, Rupert Schlick, Sebastian Chlup, Sebastian Hunt, Shuai Li, Stefano Puri, Thomas Gruber, Vittoriano Muttillio, Vladimir Stankovic	City  City ITI AIT Trustport MTTP RGB Intecs HSRM All4Tec Tecnalia Magillem Integrasys AMT CEA Siemens Thales Univaq
1.1	2 May 2019	Second version for internal review	ditto	ditto
1.2	2 May 2019	For internal review, updates to sections 2-3	ditto	ditto
2.0.	10 May 2019	Final, integrated chapter 5	ditto	ditto

Distribution List

Date	Issue	Group
11/04/2019	Version for review	John.Favaro@intecs.it
10/05/2019	Final version	EC all@aquas-project.eu

**TABLE OF CONTENTS**

<b>LIST OF FIGURES.....</b>	<b>8</b>
<b>LIST OF TABLES.....</b>	<b>10</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>11</b>
<b>1 INTRODUCTION.....</b>	<b>13</b>
<b>2 GENERIC REQUIREMENTS AND TERMINOLOGY FOR "AQUAS METHODOLOGY" 15</b>	
2.1 Generic requirements for an AQUAS Methodology.....	15
2.1.1 Requirements on interaction points and combined analyses .....	16
2.1.2 Interaction points and combined analyses through stages of the lifecycle .....	17
2.1.3 Differentiating between SSP requirements.....	21
2.1.4 Tooling for combined analyses.....	22
2.1.5 Where in the product lifecycle interaction points should occur .....	22
2.2 Common terminology: The Product Lifecycle Conceptual Model.....	23
2.2.1 The 4-model structure of the AQUAS conceptual model.....	23
2.2.2 Product Life-Cycle Model.....	24
2.2.3 The Product and Organization model.....	27
2.2.4 Quality Analysis model.....	30
2.2.5 Co-Engineering model.....	32
2.2.6 Traceability across the PLC.....	33
<b>3 TOOLING REQUIREMENTS FOR SUPPORTING INTERACTION POINTS AND TRACEABILITY .....</b>	<b>34</b>
3.1 General scenario for supporting IPs.....	34
3.2 Typical expected needs .....	35
3.3 Specific usage requirements for tools supporting Interaction Points.....	36
3.4 Requirements on the use flow .....	38
3.5 Requirements on the use model .....	39
3.5.1 Use model-Step 1: Files collection.....	39
3.5.2 Use model-Step 2: Fragments collection .....	39
3.5.3 Use model-Step 3: Trade-off description .....	40
3.5.4 Use model-Step 4: Action plan description.....	40
3.6 Conclusion.....	40
<b>4 METHODS FOR COMBINED ANALYSES .....</b>	<b>41</b>
4.1 Hazard and Operability Analysis for identifying safety/security interactions at requirement/conceptual design stage (Medical Use Case Example) .....	41
4.1.1 Aim of the example.....	41
4.1.2 Method .....	42
4.1.3 Results.....	42
4.1.4 Lessons Learned .....	44
4.1.5 Further Developments.....	45
4.2 Combined Hazard Analysis and Threat Assessment Including a Threat Identification Based on Assets (Medical Use Case Example) .....	45
4.2.1 Aim.....	45
4.2.2 Method .....	45
4.2.3 Results.....	49

4.2.4	Lessons Learned .....	51
<b>4.3</b>	<b>Combined Analysis of Trade-Offs Regarding User Authentication (Medical Use Case Example) .....</b>	<b>51</b>
4.3.1	Aim.....	52
4.3.2	Method .....	52
4.3.3	Results.....	52
4.3.4	Lessons Learned .....	56
4.3.5	Further Developments .....	56
4.3.6	Further Details.....	57
<b>4.4</b>	<b>Probabilistic Analysis of Performance-Security Trade-Offs via SANs (ATM Use Case Example).....</b>	<b>57</b>
4.4.1	Aim.....	58
4.4.2	Method .....	58
4.4.3	Results.....	60
4.4.4	Lessons Learned .....	60
4.4.5	Further Developments .....	61
4.4.6	Further Details.....	61
<b>4.5</b>	<b>Analysis of Safety-Security-Performance Trade-Offs via SANs (Industrial Drive Use Case Example) ...</b>	<b>61</b>
4.5.1	Aim.....	63
4.5.2	Method .....	64
4.5.3	Results.....	65
4.5.4	Lessons Learned .....	66
4.5.5	Further Developments .....	66
4.5.6	Further Details on the Model .....	67
<b>4.6</b>	<b>Combined Analysis of Safety and Performance to Support Design Space Exploration and Technical Solutions Comparison (Space Use Case Example) .....</b>	<b>67</b>
4.6.1	Aim.....	67
4.6.2	Method .....	69
4.6.3	Results.....	70
4.6.4	Lessons Learned .....	72
4.6.5	Further Developments .....	72
4.6.6	Further Details.....	72
<b>4.7</b>	<b>Combined Analysis of Security and Performance to Support the Product Lifecycle using SSDLC and TTool (Industrial Drive Use Case Example).....</b>	<b>73</b>
4.7.1	Aim.....	73
4.7.2	Method .....	73
4.7.3	Results.....	74
4.7.4	Lessons Learned .....	77
4.7.5	Further Developments .....	77
<b>4.8</b>	<b>Failure Modes, Vulnerabilities and Effect Analysis (FMVEA) (Industrial Drive Use Case Example).....</b>	<b>78</b>
4.8.1	Aim.....	78
4.8.2	Method .....	78
4.8.3	Results.....	80
4.8.4	Lessons Learned .....	81
4.8.5	Further Developments .....	81
<b>4.9</b>	<b>Combined Analysis of Safety, Security and Performance in the Design Stage (Space Use Case Example).....</b>	<b>81</b>
4.9.1	Aim of Use Case Example.....	82
4.9.2	Method .....	82
4.9.3	Results.....	83
4.9.4	Lessons Learned .....	85
4.9.5	Further Developments .....	85
<b>4.10</b>	<b>Translation Validation: Checking C Code Conformity (Rail Use Case Example) .....</b>	<b>86</b>

4.10.1	Aim.....	86
4.10.2	Method: Verification of conformity of Generated C code .....	87
4.10.3	Results.....	88
4.10.4	Lessons Learned .....	90
<b>4.11</b>	<b>Safety and Performance Analysis in Multiprocessor Task Scheduling (Space Use Case Example) .....</b>	<b>91</b>
4.11.1	Aim.....	91
4.11.2	Method .....	91
4.11.3	Results.....	92
4.11.4	Lessons Learned .....	93
<b>4.12</b>	<b>Efficient Formal Verification of System Software Using Ada 2012 and SPARK 2014 (Space Use Case Example).....</b>	<b>93</b>
4.12.1	Aim in the Use Case .....	94
4.12.2	Method .....	94
4.12.3	Results.....	95
4.12.4	Lessons Learned .....	97
4.12.5	Further Developments .....	97
<b>4.13</b>	<b>Combined Model-Based Testing for Multiple Concerns (ATM Use Case Example).....</b>	<b>98</b>
4.13.1	Aim.....	98
4.13.2	Method .....	99
4.13.3	Results.....	101
4.13.4	Lessons Learned .....	102
4.13.5	Further Developments .....	103
<b>5</b>	<b>INTERACTION POINT PLANNING IN THE USE CASES.....</b>	<b>104</b>
5.1	IP Plan for the The ATM Use Case (UC1) .....	104
5.2	IP Plan for the Medical Devices Use Case (UC2).....	108
5.3	IP Plan for the Industrial Drive Use Case (UC4).....	113
5.4	IP Plan for the Space Multicore Architectures Use Case (UC5).....	117
<b>6</b>	<b>CONCLUSIONS .....</b>	<b>120</b>
<b>7</b>	<b>GLOSSARY AND ABBREVIATIONS.....</b>	<b>122</b>
	<b>REFERENCES .....</b>	<b>124</b>
	<b>LIST OF ANNEXES.....</b>	<b>125</b>

## List of Figures

Figure 2-1: Stages of refinement. Analyses and discussions that involve two or more aspects among Safety, Security and Performance are the AQUAS Interaction Points .....	18
Figure 2-2: Fragment of activity diagram representing detailed stages of refinement for a possible concrete PLC .....	19
Figure 2-3: A possible concrete requirement production stage, including the subdivision of activities between different specialisms.....	20
Figure 2-4: A possible interaction point following the requirement elicitation in Figure 2-3. ....	21
Figure 2-5: The 4-model structure of the AQUAS conceptual model .....	23
Figure 2-6: Layered representation of the analyses .....	24
Figure 2-7: SEBoK core concepts and their relevance to the PLC.....	25
Figure 2-8: The Product Life Cycle (PLC) model.....	26
Figure 2-9: Relation between the PLC model and the product model .....	27
Figure 2-10: ISO/IEC/IEEE 42010 Architecture Conceptual Model .....	28
Figure 2-11: SEBoK core concepts related to work products and organization .....	29
Figure 2-12: The product and organization model .....	30
Figure 2-13: Relation between the PLC model and the product and organization model.....	30
Figure 2-14: SEBoK core concepts related to product quality.....	31
Figure 2-15: Relation between the PLC model, the product and organization model, and the quality analysis model .....	32
Figure 2-16: Relation between the PLC model, the product and organization model, the quality analysis model, and the co-engineering model.....	33
Figure 3-1: Use flow of an interaction point.....	38
Figure 3-2: File collection in the IP. <i>Note: "IP project" is, following ECLIPSE terminology, a tool-defined entity in which the references and other information created for the IP are stored.</i> .....	39
Figure 3-3: Fragments collections in the IP .....	40
Figure 3-4: Trade off description .....	40
Figure 3-5: Action plan description .....	40
Figure 4-1: Link Between HAZOP Analysis and other Analyses by UC2 Partners in the Requirements Phase.....	44
Figure 4-2: Initiating a new function in medini analyze .....	46
Figure 4-3: Applying HAZOP analysis in medini analyze.....	46
Figure 4-4: Identifying malfunctions in medini analyze .....	47
Figure 4-5: Asset Identification in medini analyze.....	47
Figure 4-6: Threat derivation from assets .....	47
Figure 4-7: Threat assessment table in medini analyze .....	48
Figure 4-8: Imported process HAZOP and derived attacks .....	48
Figure 4-9: Linking attack scenarios to threats .....	49
Figure 4-10: Identified hazards, threats, attacks and the relations between them .....	50
Figure 4-11: Relations view in medini analyze .....	50
Figure 4-12: Relationship Between the Various Factors in the Decision to Implement Authentication .....	54
Figure 4-13: The SAN "composed" model of the ATM demonstrator. ....	59
Figure 4-14: The SAN "composed" model of the "industrial drive" demonstrator. ....	64
Figure 4-15: Hepsycode Methodology .....	69
Figure 4-16: HEPHYCODE Process Network Model .....	70



Figure 4-17: Hepsycode AQUAS Version 1 (No Security) - LEFT, Hepsycode AQUAS Version 1 (with Security) - RIGHT .....	71
Figure 4-18: Hepsycode Performance/Safety Analysis - LEFT, Hepsycode Performance/Safety/Security Analysis - RIGHT .....	72
Figure 4-19: Combined Analysis of Security and Performance to Support the Product Lifecycle using SSDLC and TTool .....	74
Figure 4-20: TTool – results security algorithm vs. performance .....	74
Figure 4-21: Inference analysis using SSDLC and TTool .....	77
Figure 4-22: System-Model .....	79
Figure 4-23: Defined Rules .....	79
Figure 4-24: Analyzer Results .....	80
Figure 4-25: Results shown in the System Model.....	80
Figure 4-26: Description of the interactions.....	83
Figure 4-27: Software architecture .....	83
Figure 4-28: Schedulability analysis results.....	84
Figure 4-29: Example and part of a safety-security tree .....	84
Figure 4-30: Example of a tree exported in OpenPSA format with tags on the nodes.....	85
Figure 4-31: Output of Concept-aware analysis tool .....	85
Figure 4-32: Hierarchical approach.....	95
Figure 4-33: Distribution of efforts within implementation and project coordination .....	96
Figure 4-34: Method overview – combined model-based testing for multiple concerns.....	99
Figure 4-35: Example state machine from UC1.....	102
Figure 5-1: PLC for Use Case 1, with interaction points .....	104
Figure 5-2: PLC for the Industrial Drive use case, with interaction points .....	113

## List of Tables

Table 4-1: Example row from the HAZOP analysis.....	43
Table 4-2: Example row from the qualitative trade-off analysis .....	53
Table 4-3: Outline analysis of security viewpoint – likelihood of various threats .....	56
Table 4-4: Probability distribution of the message delivery times (PDMRT) .....	60
Table 4-5: Sensitivity analysis results of the effect of cleansing interval on model behaviour with client only attacks.....	65
Table 4-6: Results - Computation time according to different security levels and algorithms .....	74
Table 4-7: Performance in terms of cycles depending on the clock divider .....	76
Table 4-8: formal B source algorithm.....	89
Table 4-9: C translation .....	89
Table 4-10: ACSL translation.....	89
Table 4-11: Validation report .....	90
Table 4-12: Bin packaging thread allocation results .....	93
Table 5-1: Interaction Points of ATM use case. ....	106
Table 5-2 : Interaction Points of Medical use case .....	109
Table 5-3: Interaction Points of Industrial Drive use case.....	114
Table 5-4 Interaction Points of Space use case. ....	118
Table 7-1: AQUAS-specific terms and AQUAS-specific word uses.....	122
Table 7-2: Abbreviations used in the text .....	122

## Executive Summary

This deliverable, D3.2, reports progress in AQUAS Workpackage 3, Methodology.

AQUAS aims at improving how the non-functional requirements of safety, security, performance (SSP) are dealt with during the product lifecycle for embedded computer systems, both in the sense of reducing the risk of surprises – unsatisfied requirements or unsatisfactory trade-offs between conflicting requirements – late in the lifecycle, when they would be more expensive; and of achieving this risk reduction in a cost-effective way.

The AQUAS approach to methodology improvement is based on (a) applying methods (where possible supported by software tools) for *combined analyses* of project artefacts from the viewpoints of safety, security, performance (and possibly other non-functional requirements); (b) limiting the overhead cost of these combined analyses by only applying them at a limited number of points in the product lifecycle, called *interaction points*, where decisions can thus be taken about any need for rework, for agreeing trade-offs between conflicting requirements, or prescriptions for later steps in the PLC, e.g., more detailed requirements or further analyses that will be needed.

According to the AQUAS workplan, the purpose of D3.2 is to "demonstrate, through example applications, the application of combined analysis and assessment of safety, security and performance, that is, how the 'interaction point' concept can be implemented in practice with the techniques considered" on small scale examples suitable for demonstrating the methods, so as to support their application in the use cases.

In addition to this specification, the present document takes into account the recommendations from the first review of AQUAS, which asked for an "initial high-level SSP co-engineering methodology description" to help harmonize work in the project. Therefore, we have added in this document more detail to the introduction to the AQUAS approach previously provided in D3.1, including a "PLC conceptual model", which establishes a terminology for the essential components of the AQUAS approach, and preliminary requirements for tools supporting documentation of interaction points and traceability of information and decisions between them.

The bulk of D3.2 (Chapter 4) is the set of descriptions of methods with which AQUAS is experimenting, including: methods for hazards and risk analysis addressing the effects of both accidental events and malicious attacks, and the effects of defense mechanisms introduced against either; extensions of standard design analysis techniques, like fault tree analysis, to combined analysis; probabilistic modelling to allow quantitative predictions of the implications of design decisions on risk; tool-supported techniques for checking that software-hardware architecture designs satisfy execution performance requirements; and verification techniques (formal verification as well as testing) to detect problems in the implemented systems. These descriptions of analysis techniques are written as stand-alone items that can be used for reference within AQUAS; they could evolve into supportive annexes in a standard for co-engineering of quality attributes.

All in all, at this stage the trials of these techniques have been successful: no major obstacles have been encountered in applying them, and these techniques have helped to detect interdependencies between the SSP requirements and design decisions driven by them, to trace possible hazards and their causes, etc. Although the examples documented here are limited in scope, for the sake of readability, most of them are already being extended to address broader subsets of the demonstrator systems. The activities under way now in each demonstrator will lead to further interaction points, so that it will be possible also to assess the effectiveness of the approach and of the specific techniques

used in propagating necessary information about SSP interdependence along the PLC, and to learn some lessons about the appropriate placement of interaction points along the PLC.

# 1 Introduction

The AQUAS project aims to promote effective and practical methodological advances for the co-engineering of security, safety, performance of embedded computer systems.

Workpackage 3 in the project pursues these goals through the definition of methods and approaches for the combined analysis and assessment of safety, security and performance, at "interaction points" in the co-engineering process.

We recall the two-fold methodological problem that motivates the AQUAS project, and the approach that AQUAS is trialling:

1. on the one hand, system and subsystem attributes like safety, security, performance are interdependent: a developer or assessor who only focused on one of these attributes at a time would risk leaving unresolved conflicts, e.g., it could happen that once the design has been engineered to achieve satisfactory security, the mechanisms introduced for this purpose violate some requirement concerning safety. So, separate processes for enforcing these different requirements risk costly reworks during development, or worse, accidents in operation, or expensive recalls.  
On the other hand, it may happen that trying to separately provide safety and security features produces unnecessarily costly solutions – e.g., through not noticing that memory protection enforced for safety reasons are adequate to satisfy a security requirement as well.  
So, the AQUAS approach includes methods for *combined analyses* of these various interdependent attributes, to assess whether a system satisfies its requirements from these diverse viewpoints and/or to assist design decisions. Many partners in the AQUAS consortium are tool developers whose products are being improved to form toolsets that support these analyses;
2. on the other hand, analyses that encompass these various viewpoints can prove hard and time-consuming: because these analyses may be inherently complex, and because they require interaction between specialists who typically belong to different cultures. E.g., the specialist knowledge and skills required to analyze or enhance a system or system design from the viewpoint of security typically belong to different specialists from those dealing with safety.  
The AQUAS approach is to limit this expenditure of effort by restricting these combined analyses to specific points in the lifecycle (called "interaction points"); this is in line with proposals that have emerged in industrial environments (cf e.g. SAE J3061 Cybersecurity Guidebook).

Deliverable D3.2 is specified in the project workplan as follows:

*This deliverable will demonstrate, through example applications, the application of combined analysis and assessment of safety, security and performance, that is, how the "interaction point" concept can be implemented in practice with the techniques considered. The examples will be on small systems (e.g. parts of the demonstrators), suitable for demonstrating the methods so as to support application in the use cases. Each planned interaction point (one or more per use case) will be specified by collaboration of the one or more WP3 partners that provide the techniques, in general formats harmonised, as far as feasible, by consensus with the use case owners under coordination by City.*

In addition to this specification, the present document takes into account the recommendations from the first review of AQUAS in June 2018. The reviewers asked for an "initial high-level SSP co-engineering methodology description" to help harmonize work in the project. This has been done, within the constraints of the planned bottom-up approach of AQUAS: to this end we have included

(i) a "PLC conceptual model", developed by TecNALIA (the PLC goal leader) by interaction with WP3 partners and the various demonstrators; (ii) preliminary results from the focus group of tool developers who have studied the tooling requirements for documentation of interaction points and traceability of information between them.

The AQUAS workplan calls for experimentation-driven innovation: the five demonstrators experiment with applying methodological advances in a set of concrete, diverse development projects. The goal for the end of the project is to gain and report experience in the application of the AQUAS approach. The demonstrators involve companies of various sizes and backgrounds and different industrial sectors (and thus, not least, different applicable standards and different regulatory agencies) so that from this learning exercise AQUAS plans to extract not only lessons useful to the individual companies and sectors, but also to be able to extract what is common and can be called an AQUAS "methodology".

So, rather than a single universal, detailed and prescriptive process, the methodology emerging from the project is expected to consist of useful recommendations for applying the AQUAS approach, and especially lessons learned in more than one demonstrator, and thus validated, to the extent that is feasible. This does not preclude (a) some demonstrators producing more formal results, like prescriptive process manuals to be proposed for more extensive in-house piloting and validation; (b) AQUAS producing advice for standardization committees based on the lessons learned (and/or indeed on the problems that motivate AQUAS and their refined understanding derived from AQUAS experimentation. Standard-making inevitably lags behind awareness of problems).

## 2 Generic Requirements and Terminology for "AQUAS Methodology"

Contributors: Tecnia and City

The methodological advances expected from AQUAS are a set of lessons learned that will help both the project partners and the industry in general to improve their processes for managing safety, security, performance of embedded systems, making them less surprise-prone and, if possible, more cost-effective by better exploiting synergies between solutions for safety, security and performance. These lessons concern both the use of tools and techniques, and the timing and organization of the "interaction points" when combined analyses are performed that take into account the various "non-functional" requirements.

The project takes a bottom-up, empirically driven approach to methodological development. However, to facilitate this development during the AQUAS project, exchange of experience between the demonstrators, and harmonization of the final results, this chapter includes additional information and terminology and an initial high-level description of a SSP co-engineering methodology, in the form of a "PLC conceptual model".

The concept of *interaction point* was discussed extensively in D3.1 (Sections 4.1-4.6) and a preliminary planning of interaction points in the demonstrators was provided, described according to a common template. Without repeating that discussion, we recall here that we call "interaction point" "both an activity and the point in a product life cycle (PLC) at which it occurs. The activity is "interaction" in that:

- a) experts in the various aspects of the system and its properties interact, e.g. security and safety experts;
- b) their analyses are combined in some way, that may be anywhere in the range from informal discussion and mutual critique to using mathematical models, typically supported by tools, to assess various measures of interest for alternative design options, or even a single, summary measure to be optimized (e.g., probability of an undesired event);
- c) the need for changes or decisions may be recognized that require an integrated view, e.g. because of inevitable trade-offs between desirable properties, and these trade-offs are discussed between the various experts to produce recommendations/decisions.

About the *need* for interaction between specialist activities concerning e.g. security and safety analyses, and of these with the main development, validation or operation effort, this document, like most AQUAS activity, focuses on the technical concerns about interactions between the requirements produced by different specialisms and between the measures adopted to satisfy them. We do note that interaction between these activities would be required for various reasons even without these technical challenges. For instance, project scheduling has to co-ordinate the various analyses and development steps for efficiency, avoiding e.g. unnecessary waits and risk of needing rework due to late analyses. However, focus on the technical challenges is part of the AQUAS workplan and is justified by the difficulties they have been experienced to present to industry.

### 2.1 Generic requirements for an AQUAS Methodology

As development proceeds through stages of refinement from initial requirements and conceptual design through increasingly detailed design and implementation, and later, through the right branch of a V model, with verification and validation that the implementation matches progressively higher-level requirements, the AQUAS approach assumes that at certain points in the process – interaction points - "combined analyses" are applied, which consider all the non-functional requirements. This is

especially important in the refinement process – the left-hand branch of the V – to ensure that the product at its current state of development satisfies the set of requirements developed, for the whole system and for its component parts, in the previous steps, and to take early corrective action as needed.

### 2.1.1 Requirements on interaction points and combined analyses

In any development process, some verification is needed after each step of refinement. E.g., a system description version  $S_i$  includes, for a certain software component, just a specification; after the specification for that software component is turned into software code (a step of refinement), we have a system description  $S_{i+1}$ . It is required that the system produced by refinement,  $S_{i+1}$  (the one that contains the code instead of its specification) still satisfies the properties specified for the previous version,  $S_i$ . This verification implies:

1. verifying that the specific software code implements its specification
2. verifying that the refined system matches all the requirements defined for the system before refinement. The concern is that this is not necessarily guaranteed by the verification step in point 1 above.

To clarify point 2 above, suppose that  $S_i$  was accompanied by a set of "non-functional" requirements for the whole system or its subsystems/components. It may happen that the system at step of refinement  $S_i$  appeared to satisfy all these requirements.<sup>3</sup> This may be because at that level of detail, some implementation matters were still undecided, which could determine whether that requirement *would be* really satisfied; and/or because some potential way that requirements could be violated had not been foreseen; and/or the required analyses could not be performed before this refinement (e.g., whether certain requirements are satisfied depends on implementation details previously unavailable; the execution time of certain software components, the security characteristics of certain libraries). In addition, analyses at the level of refinement of  $S_{i+1}$  may serendipitously discover potential problems (e.g., security threats, hazards, performance problems) and thus requirements that ought to have been, but were not, identified earlier.

Thus, the ideal requirement for the combined analyses at an interaction point is that they verify that all real requirements (not just those formally stated) for the system and for its parts are satisfied. This is "ideal" in that complete knowledge is unattainable in principle, but in practical terms the requirements on these analyses can be stated as:

- the analyses must be such as to ensure, collectively, that a set of previously specified requirements are still satisfied; one may aim to verify this for the *whole* set of requirements, or just for a subset whose satisfaction – it is expected – may have been affected by the previous step[s] of refinement

---

<sup>3</sup> We take a broad view of what constitutes "requirements". Requirements can take many forms, and the boundary between "functional" and "non-functional" may be fuzzy; we take the broadest view: e.g., a requirement can be that a subsystem's architecture includes the feature that a standard recommends for Safety integrity level 4; or that that a subsystem exhibits a probability of failure on demand no greater than  $10^{-4}$ ; or that a communication link be encrypted with 256-bit key AES.



- and/or the analyses must be suitable for revealing either unexpected problems (violations of written or unwritten requirements) (or unexpected opportunities to change the architecture in such a way that a single architectural mechanism can satisfy multiple SSP requirements) that can be expected to become visible at this stage of refinement.

### 2.1.2 Interaction points and combined analyses through stages of the lifecycle

Figure 2-1 gives a coarse-level description of how this refinement process proceeds between stages of the lifecycle, irrespective of what kinds of non-functional requirements are involved. At stage  $i+1$ , a certain amount of "refinement" takes place, as stated above, e.g., moving from coarse-grain design to detailed design of some system part, or from software specifications to code for some software component. Analyses follow, to check that these refinements are "OK": that all requirements inherited are satisfied. There are three possible outcomes:

- all requirements are still satisfied, and the artefacts produced (the "system description", version  $S_{i+1}$ ) can be passed on the next stage of refinement;
- some requirements are not satisfied, but it is judged that some change to the last refinements made can satisfy them, e.g., selecting faster chip or optimizing some code, will satisfy them. Then an iteration occurs within the same stage: these changes are applied and the analyses are repeated. We note that sometimes an iteration may be requested because the analyses reveal that the requirements *are* satisfied, but a better solution is nonetheless possible;
- some requirements are not satisfied, and it is decided that the requirements inherited from stage  $S_i$  *cannot* be satisfied. It is necessary to go back to the previous stage where those requirements were generated: e.g., some required functionality must be abandoned, or a longer response time must be allowed.

In the AQUAS scenario, the analyses may deal with the various aspects of security, safety and performance. If requirements inherited at a stage cannot be satisfied, the decisions on new trade-offs may be for instance to relax a requirement on strength of encryption, or abandoning some functionality, or allowing higher cost or power consumption.

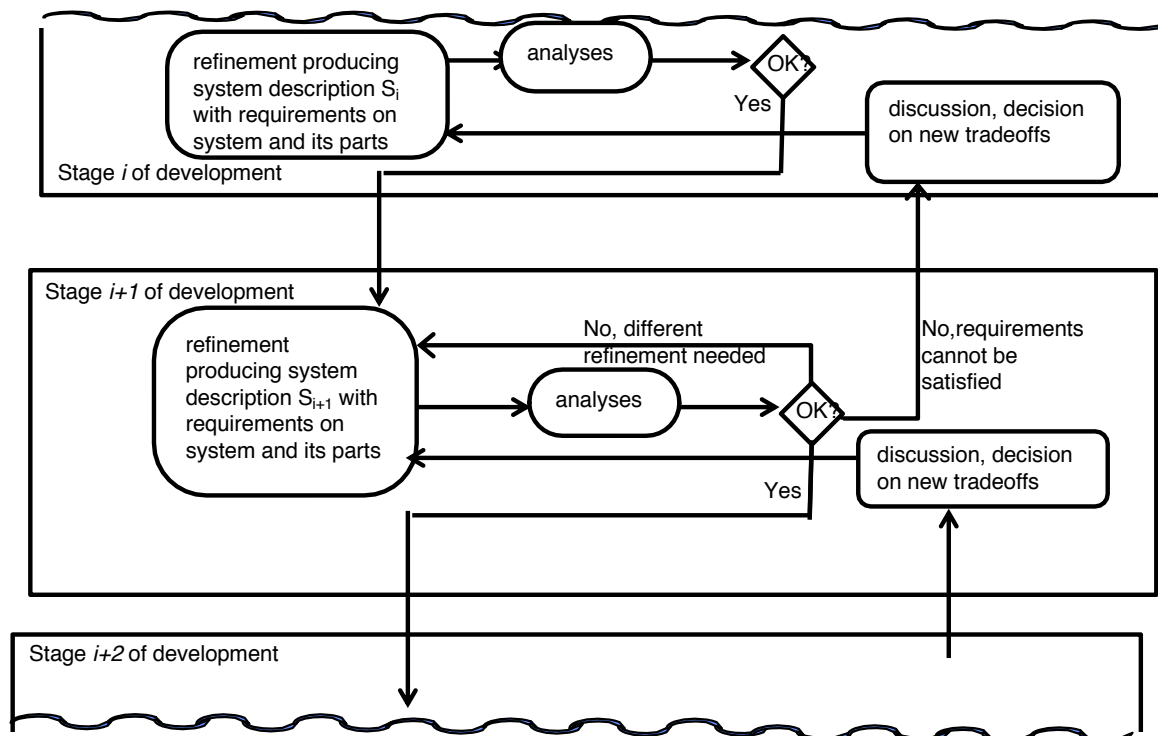


Figure 2-1: Stages of refinement. Analyses and discussions that involve two or more aspects among Safety, Security and Performance are the AQUAS Interaction Points

Figure 2-2 shows an example of concrete activities and artefacts involved in a possible example of PLC, but still without discriminating between activities and artefacts belonging to different specialisms, like safety and security.

Figure 2-3 and Figure 2-4, by contrast, show a possible organization of requirement elicitation and validation phases managed as separate specialist activities, and the interaction point appears in the Co-engineering part of Figure 2-4.

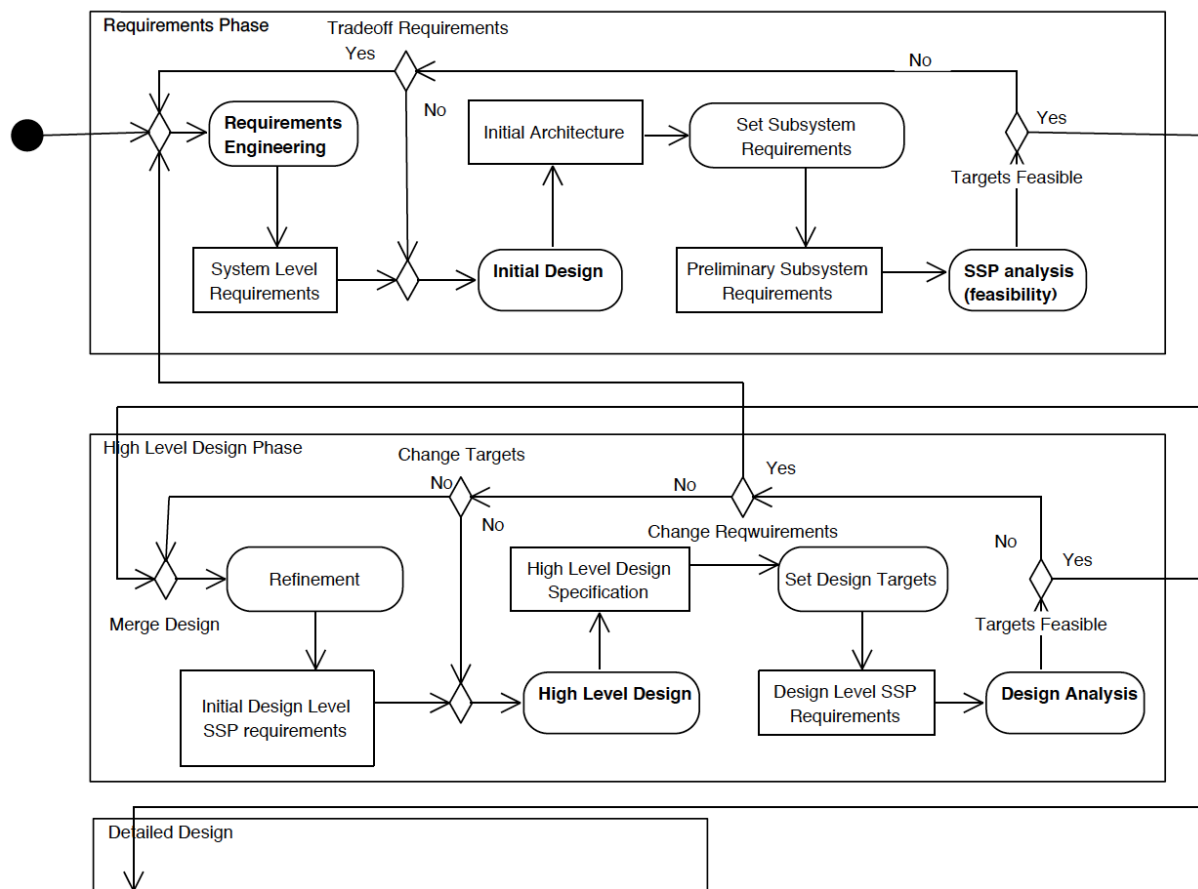


Figure 2-2: Fragment of activity diagram representing detailed stages of refinement for a possible concrete PLC

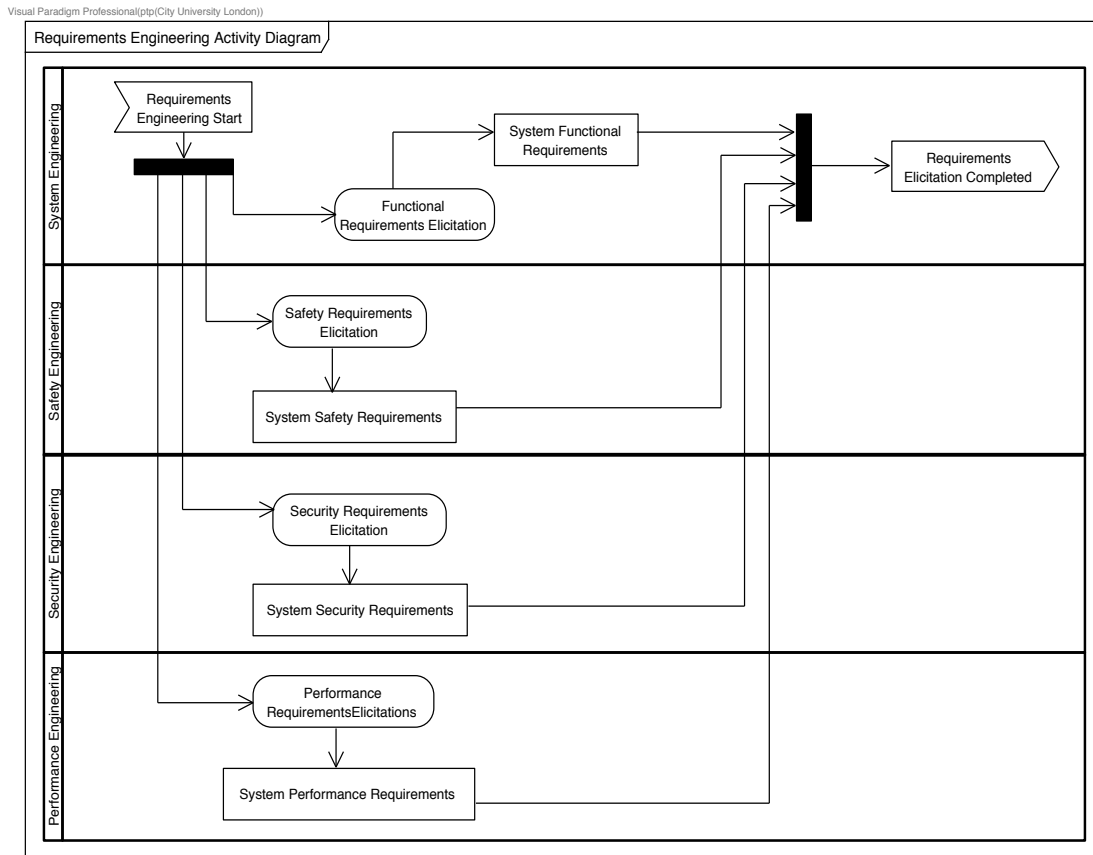


Figure 2-3: A possible concrete requirement production stage, including the subdivision of activities between different specialisms.

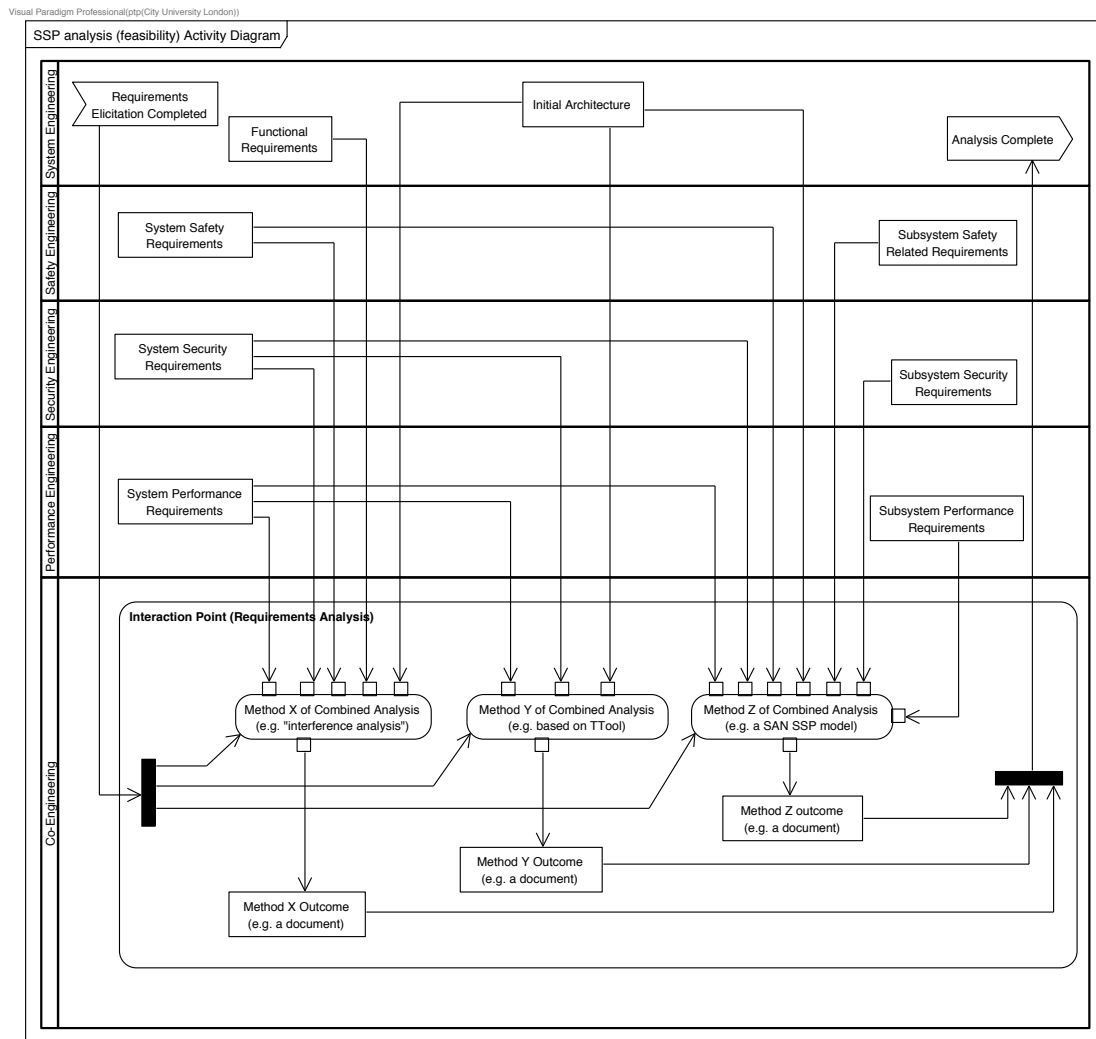


Figure 2-4: A possible interaction point following the requirement elicitation in Figure 2-3.

### 2.1.3 Differentiating between SSP requirements

The requirements whose satisfaction must be verified are often labelled as being "security requirements", "safety requirements", "performance requirements", etc. We must acknowledge that these labels are not mutually exclusive and thus this terminology can be somewhat misleading. For instance, *security* properties have to do with what a malevolent adversary might achieve through the system; *safety* properties address ways the system could cause harm; and a reasonable goal for an adversary is actually to use the system to cause harm: so a requirement for system safety dictates requirements for system security. More complex dependencies are in fact common. It is common, for instance, that a *safety* requirement at system level demands that certain computations be performed within acceptable response times or throughput requirements (*performance* requirements); therefore, a *security* requirement is that adversaries must not be able to cause violations of these performance requirements; and once security mechanisms are added for this purpose, there will be further non-functional requirements of reliability and performance of these mechanisms, to give sufficient assurance that they will both perform their security functions and not themselves become a cause of

under-performance that affects safety. This interconnection between the various "non-functional" properties of a system and of its parts is indeed a basic motivation for the AQUAS project.

Nonetheless, requirements, and the activities that they trigger, can be labelled as, e.g., 'security' requirements on the basis of their being so labelled in standards and guidelines; or because the work they require is work by security specialists. In these cases, the labelling applied is not meant to suggest that these qualities are separate and independent from other qualities like safety or performance, but to document organizational aspects of the PLC.

#### 2.1.4 Tooling for combined analyses

AQUAS is experimenting with various tools and techniques for combined analyses. However, it is expected that AQUAS methodological conclusions will set no constraints on which tools and techniques may be used for any stage of combined analysis: this freedom is required because:

- there are many suitable choices. In AQUAS itself, the set of tools and techniques being trialled includes alternative means of pursuing the same goals; alternative tooling options exist for the whole PLC;
- each company may have an established set of tools and techniques, dictated by its regulatory/standardization/certification regime, or by previous investment in software and training, and would want to extend this set to support co-engineering in a way that takes advantage of the previous investment, so will take into account factors like similarities of technical languages used, and of tool interfaces, tool interoperability, etc.

On the other hand, tool vendors that aim to support the AQUAS approach will aim to create a toolset of interoperable tools. Such efforts are indeed under way in WP4.

#### 2.1.5 Where in the product lifecycle interaction points should occur

In D3.1 (Section 4.2) we identified some requirements on when interaction points should occur, and identified two ways of triggering an interaction point (a set of combined analyses):

- "statically scheduled" interaction points, decided in advance to take place at pre-specified points in the PLC. These should be scheduled to be frequent enough that any necessary rework identified will not be too expensive, but also rare enough not to be an excessive overhead cost. We expect that:
  - a first interaction point must necessarily take place at the system requirements and concept phase of a project, to ensure that the requirements set are at least in principle consistent, that essential potential conflicts are identified, and that the conceptual system design envisaged is at least potentially adequate to satisfy them;
  - later interaction points would be scheduled for points in the PLC that precede important investments of effort, time and money: e.g., after specification of any bespoke software required in a system, before the implementation of this software proceeds, so as to avoid having to rework these specifications after problems are revealed in the implemented software.
- interaction points that are triggered by detection of some problem, which makes it desirable to perform combined analyses without waiting. These will require management mechanisms for reporting the problems to allow the decision to trigger these interaction points. We can expect that if a development organization is very effective at detecting problems, to trigger these IPs as

needed, it will need fewer "statically scheduled" IPs, and thus possibly achieve lower overall costs. This 'effectiveness' would depend on many factors: the type of systems they develop, their relationship with subcontractors and suppliers of off-the-shelf parts, and their management culture. We do not expect AQUAS to develop enough experience to recommend criteria for organizing these "dynamic" IPs. It can be expected that most organizations adopting an AQUAS-like approach would initially rely on scheduling IPs statically, and develop experience over time that might enable them to reduce their dependence on "statically scheduled" IPs.

## 2.2 Common terminology: The Product Lifecycle Conceptual Model

The AQUAS project aims to produce improvements, in the processes for co-engineering of quality attributes, that are flexible enough to satisfy the needs of diverse application environments – industrial sectors as well as individual companies and projects – as represented by the AQUAS use cases.

To allow interchange of experience within the project, and for AQUAS outcomes to be usable outside AQUAS, a common understanding of the essential common elements of the process is required. The AQUAS Conceptual Model presented here supports this goal by defining the relationships between the essential concepts introduced in AQUAS with those that belong to the common understanding of product lifecycles. This model has been developed by observing, and generalizing from, the processes in the AQUAS use cases and relating them to the overall concepts introduced in D3.1.

This document gathers general concepts of software engineering, relating them to existing standards and elaborates on new concepts which are specific to the co-engineering challenges. The conceptual model is the ground on which to build a comprehensible AQUAS methodology with a unified vision regarding concepts and terminology.

### 2.2.1 The 4-model structure of the AQUAS conceptual model

The AQUAS conceptual model is divided in 4 interconnected components.

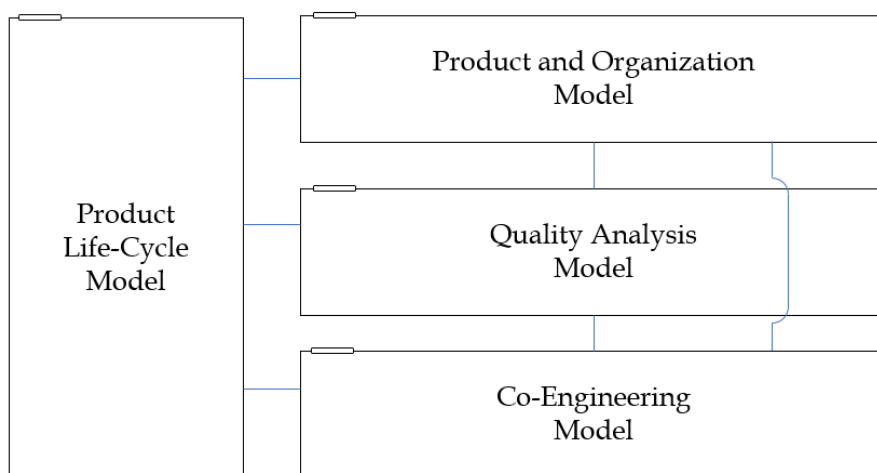


Figure 2-5: The 4-model structure of the AQUAS conceptual model

The PLC model, product and organization model, and quality analysis model gather basic concepts which are needed to understand the latest model in this document, the CE model, which serves to describe the ground for the AQUAS methodology.

A layered figure can be used as a metaphor. On the “surface”, the Product and Organization layer is followed by the Quality analysis layer. These two layers are much more mature in systems and software engineering than the Co-Engineering layer which is the focus of AQUAS (the Co-Engineering for Quality layer, at the bottom of the figure). The vertical layer of the Product Life-Cycle provides the time dimension where the product and organization evolve, and where the quality analysis and co-engineering takes place across the different stages.

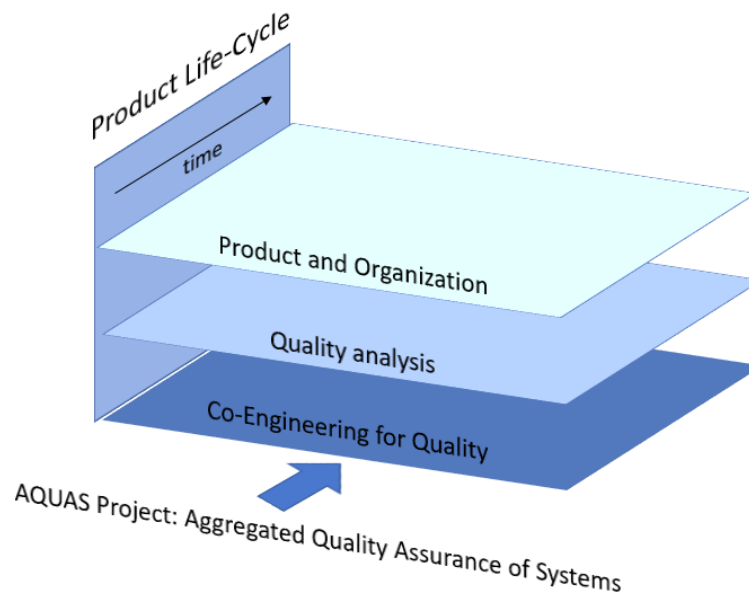


Figure 2-6: Layered representation of the analyses

As mentioned before, this conceptual model sets the ground where the AQUAS methodology is built; however, the conceptual model *per-se* provides limited support for guidance. The conceptual model provides a common understanding of the concepts that can then be instantiated in each specific project using their own implementation of the concepts and their relations. While we recommend reading all the sections, an expert might want to go directly to the Co-Engineering layer section.

## 2.2.2 Product Life-Cycle Model

### 2.2.2.1 Introduction to the PLC Model

The Product Life-Cycle (PLC) is of paramount importance in the AQUAS project, as co-engineering must span different life-cycle stages. Focusing only on one stage (e.g., requirements or software implementation) would limit the project to a very narrow scope as well as defeat some of the goals of AQUAS to discover when and how often IPs should occur. Thus, different AQUAS goals were defined to effectively manage co-engineering inside and across the PLC.

PLC is defined as the evolution of a system, product, service, project or other human-made entity from conception through retirement.

**ISO/IEC/IEEE 15288**



The following figure shows an excerpt of the SEBoK Core Concepts [SEBoK] that acknowledge the Life-Cycle Model as a prime entity supporting systems engineering. We added the circle to identify, inside it, concepts relevant to the PLC. They propose a PLC model containing several stages and referencing several PLC processes. These processes are then linked to the evolution and production of the system.

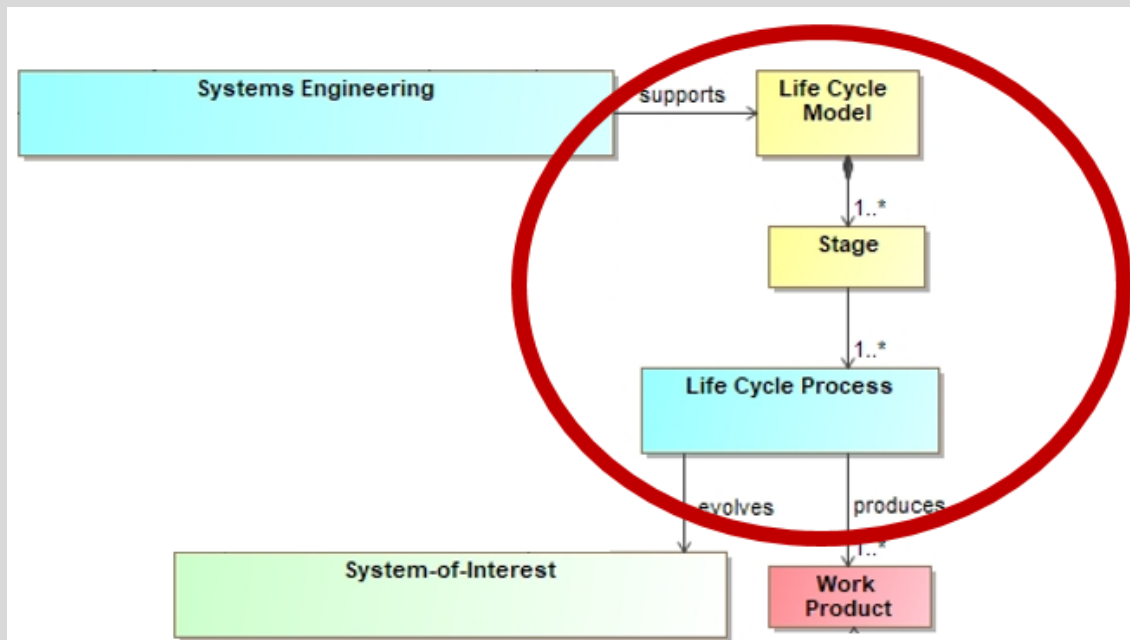


Figure 2-7: SEBoK core concepts and their relevance to the PLC

These concepts helped to define the AQUAS PLC model. Notably, we extended previous PLC model concepts, and, introduced links between the PLC model and the product model.

### 2.2.2.2 AQUAS PLC Conceptual Model

The following figure shows the main concepts of the PLC model:

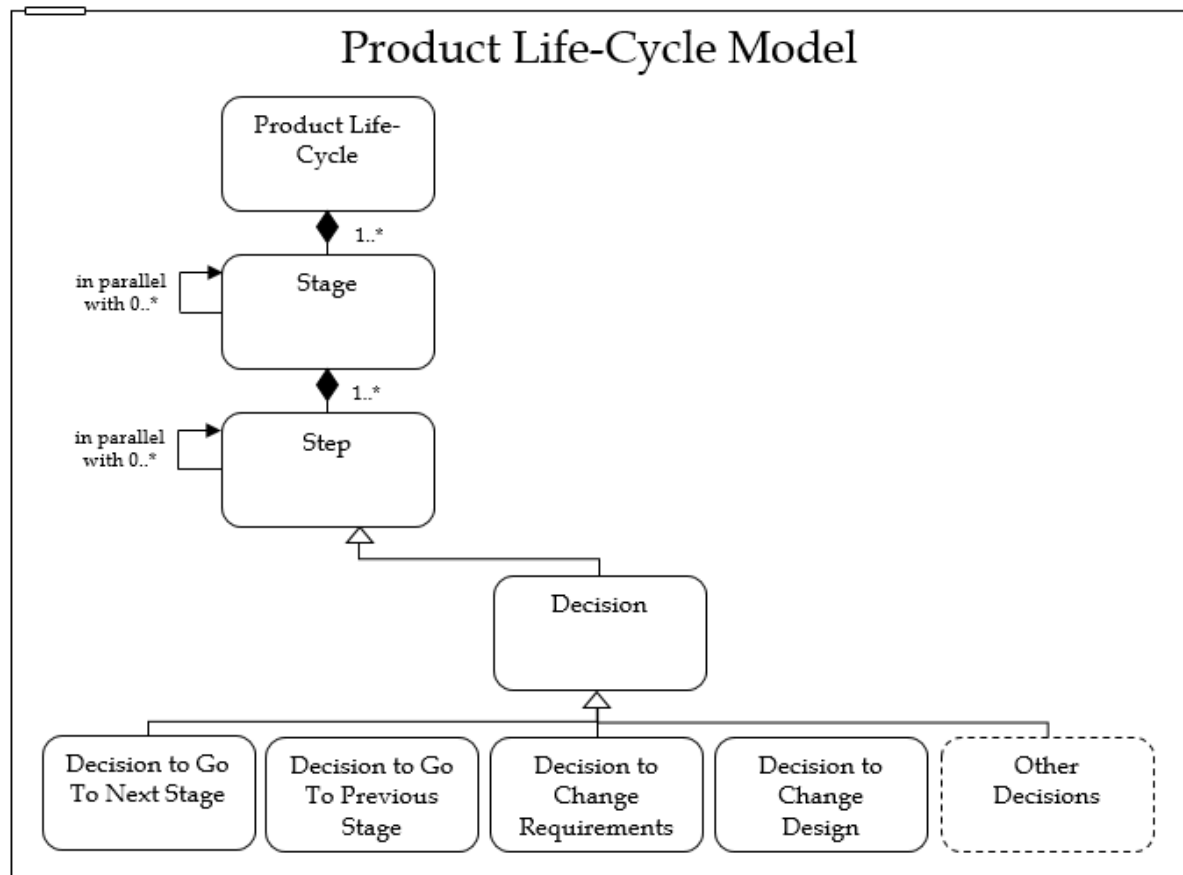


Figure 2-8: The Product Life Cycle (PLC) model

A PLC contains a set of *Stages* regarding “the evolution from conception through retirement” (ISO/IEC/IEEE 15288). A possible instantiation of the stages can refer to the phases of the V-Model, but it is open to instantiate other PLCs, given that each application domain might be required to comply with a PLC standard, or to follow an in-house methodology. Stages in turn contain *Steps*, describing the process at a more detailed level.

Stages and steps are linked via the containment relation. Iterative processes where one or more iterations are used (e.g., to refine a work product) can be expressed. It was also stated by industrial partners of some UCs that during the PLC, more than one stage can happen in parallel. Likewise, steps can also happen in parallel to other steps.

Then we have a part of the PLC model which is related to the *Decisions* and the rationale management (managing the rationale of each decision) needed to advance in the PLC, to move backwards, or to request changes. A Decision is a Step of the PLC. In the figure we added representative examples, as it is not possible to enumerate all the possible types of decisions that can happen in specific projects. In the next figure we focus on two examples, to show how each decision can be related to various concepts. The Decision to Go To Next Stage relates to the involved Stages, while the Changing Requirements Decision is directly related to a specific concept in the Product Model.

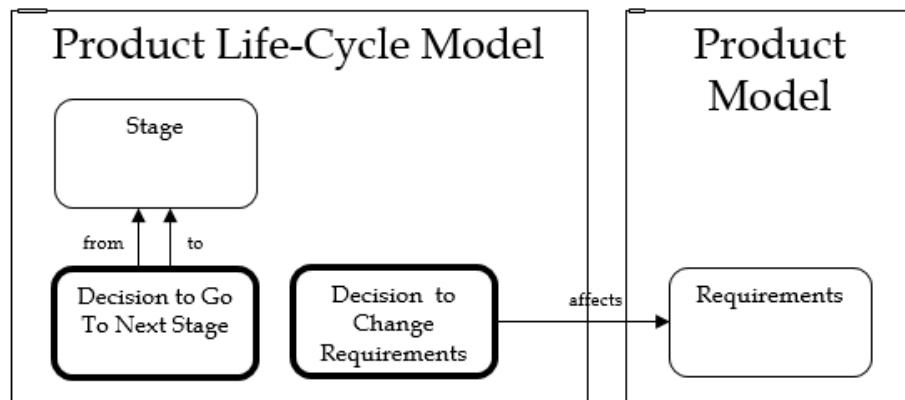


Figure 2-9: Relation between the PLC model and the product model

## 2.2.3 The Product and Organization model

### 2.2.3.1 Introduction to the Product and Organization model

The Product and Organization model focuses on the diverse types and versions of the work products, as well as on the teams, skills, and tools at hand.

The ISO/IEC/IEEE 42010 focusing on the Architecture as a key work product. We can also observe how the stakeholders and their concerns are represented.

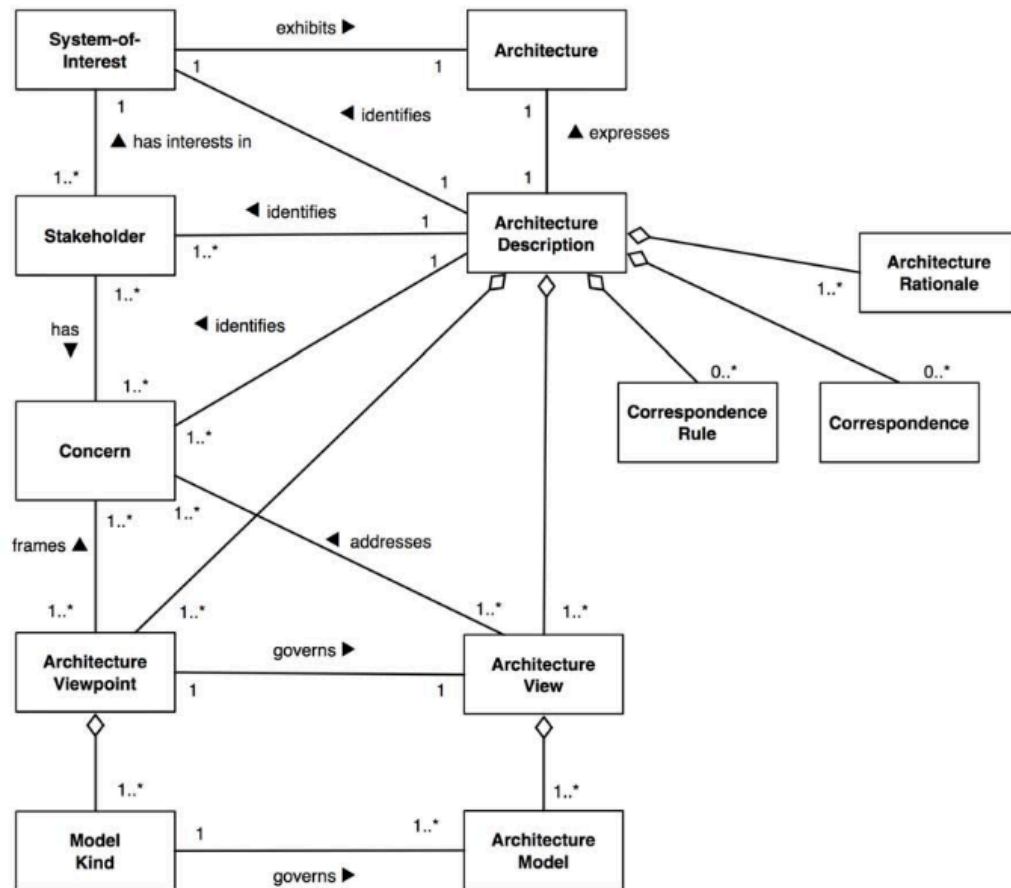


Figure 2-10: ISO/IEC/IEEE 42010 Architecture Conceptual Model

SEBoK conceptual model related to Work Product and Organization. Given the diversity of work products, SEBoK authors do not give an exhaustive list. Regarding organization, they describe the organization as a set of roles and systems engineers which are qualified in different competencies/skills.

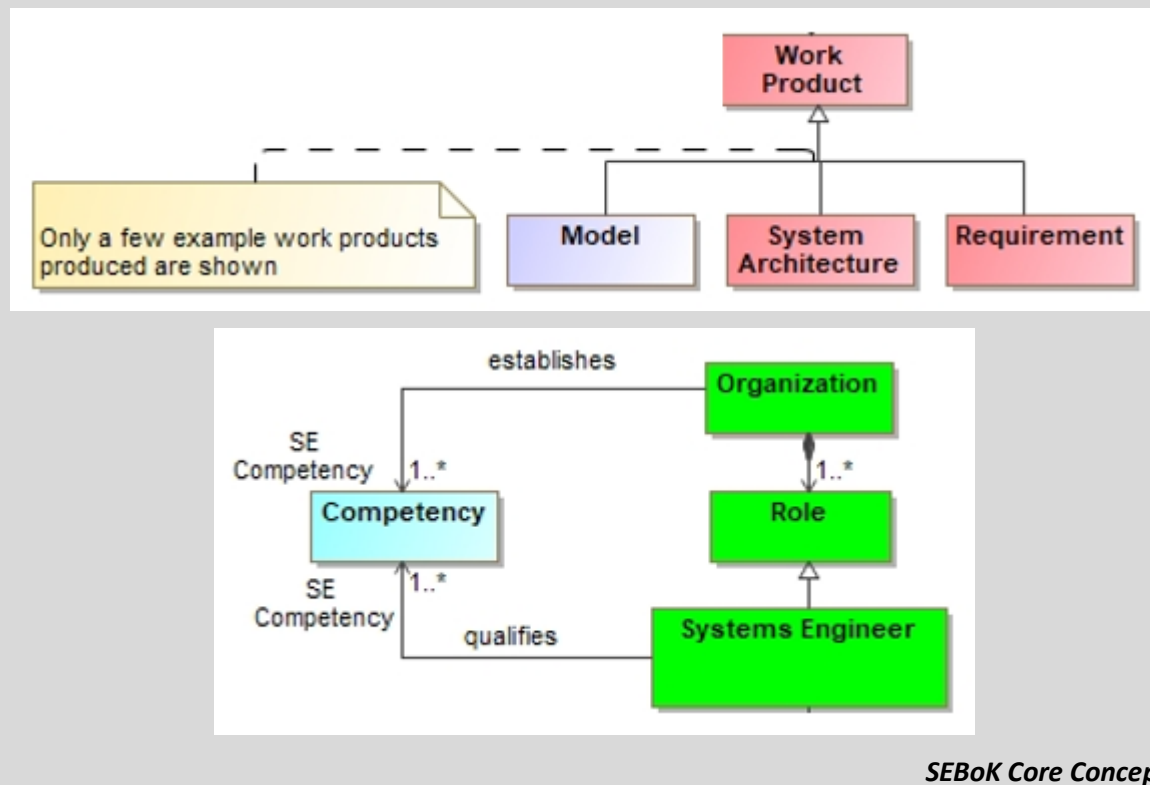


Figure 2-11: SEBoK core concepts related to work products and organization

#### 2.2.3.2 AQUAS Product and Organization conceptual model

A *work product* (and its various *versions*) is created by a *team* or a set of teams. A team is expert in a set of *techniques*, usually supported by *tools* available to the team, but in some cases not tool-supported, but applied manually. The work products can be very diverse. Examples can be Requirements, Architecture, System (and the hierarchy of sub-systems), Documentation (such as risk management documents or Assurance cases). The application domain will define the peculiarities regarding the required work products.

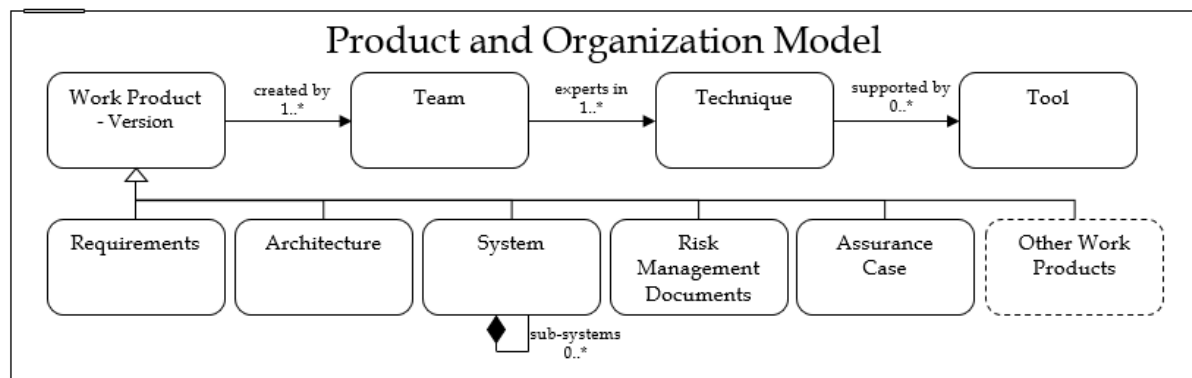


Figure 2-12: The product and organization model

The work product concept is also related to the PLC model. Notably, a PLC stage uses work products as inputs and produces outputs. Also, a PLC stage can have a set of associated work products which are expected to be created or refined during the stage. Also, a team, or a set of teams, are involved in the PLC decisions that need to be made.

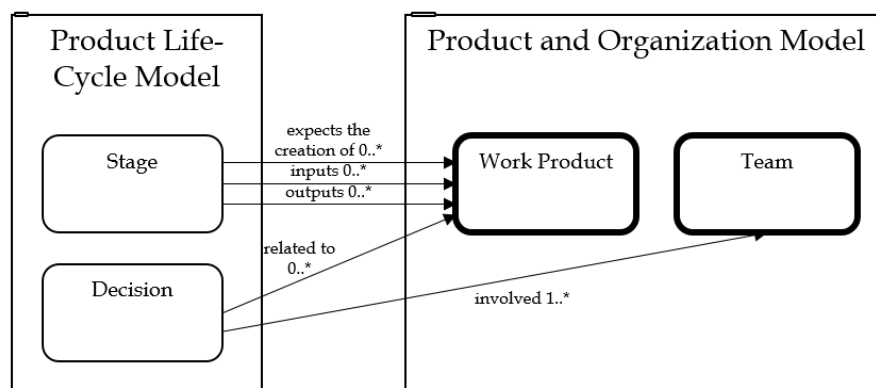


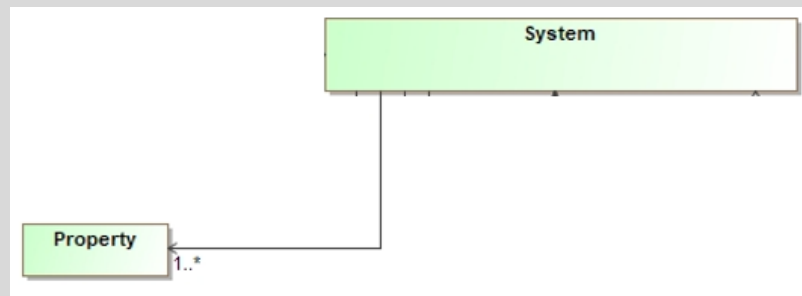
Figure 2-13: Relation between the PLC model and the product and organization model

## 2.2.4 Quality Analysis model

### 2.2.4.1 Introduction to the quality analysis model

Quality analysis in the PLC of a system is essential for checking that all the system requirements regarding functional and non-functional properties are satisfied.

The SEBoK conceptual model takes into account System properties, which refers to the relevant quality attributes of the system.



**SEBoK Core Concepts**

Figure 2-14: SEBoK core concepts related to product quality

As mentioned before, the matching notion in ISO/IEC/IEEE 42010 is "Concerns" of the various stakeholders.

**ISO/IEC/IEEE 42010**

#### 2.2.4.2 AQUAS Quality Analysis model

A *quality analysis* is a *step* in the PLC where, taking as input existing *work products*, a *team* analyzes one or several *quality attributes* (e.g., *safety*, *security*, and *performance*) using appropriate techniques and tools. A quality attribute can be an *aggregated quality attribute* like *dependability* which aggregates other, atomic quality attributes. The *quality analysis results* (e.g., *metrics* quantifying the quality attribute, *reports*) support the *decision-making* process in the PLC that is to be triggered by the quality analysis. A quality analysis result can be, in some cases, a *work product*, although this is not represented in the model. The *quality analysis consolidation* is a special type of quality analysis, i.e., the consolidation of results of different quality analyses for the same quality attribute.

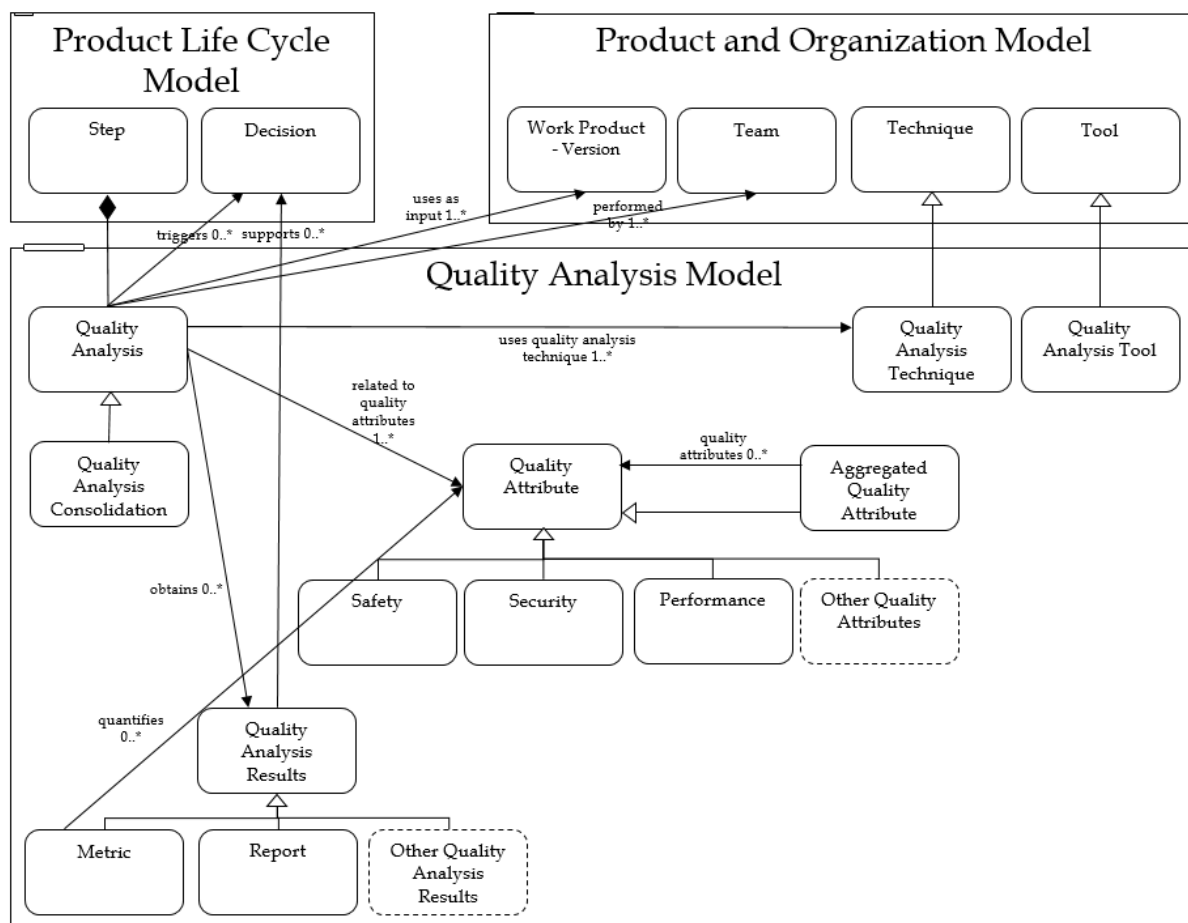


Figure 2-15: Relation between the PLC model, the product and organization model, and the quality analysis model

### 2.2.5 Co-Engineering model

A *focus area* within an organization involves personnel with a set of specialized competences (e.g., security), and deep knowledge of the system from their own perspective. Mainly because of this specialization, the team (or set of teams) within a focus area does not continuously interact with other teams of the organization, and the work products that they manage may have separate workflows. Because of this separateness, focus areas are sometimes described as "silos" or "islands".

An *interaction point* is a *step* in the PLC and a special type of *quality analysis* where several *quality attributes* belonging to different *focus areas* within the organization need to be analyzed together. It also contains a set of specific quality analysis techniques called *interference analysis techniques*.

Interaction points can be scheduled to occur at predefined moments of the PLC, or the need of an interaction point can be identified by a focus area during their activities. Regarding the latter, an additional type of PLC decision consists of triggering an interaction point.

In the model, *interference analysis techniques* may mean concrete techniques, such as FMVEA or HAZOP, or families of techniques, such as simulation, dependability analysis techniques, parametrization of quality attributes tradeoffs, sensitivity analysis etc. A more formal hierarchy of the families of techniques and concrete techniques is on-going work.



The Co-Engineering model also defines another type of *decision* called a *trade-off decision* where the involved teams need to explore and select the most appropriate solution (work product delta) to the engineering conflict.

In the diagram, the Interaction Point concept is not directly linked to work products; however, given that an Interaction Point is a type of Quality Analysis, it inherits this relation as well as all the other relations of the Quality Analysis concept.

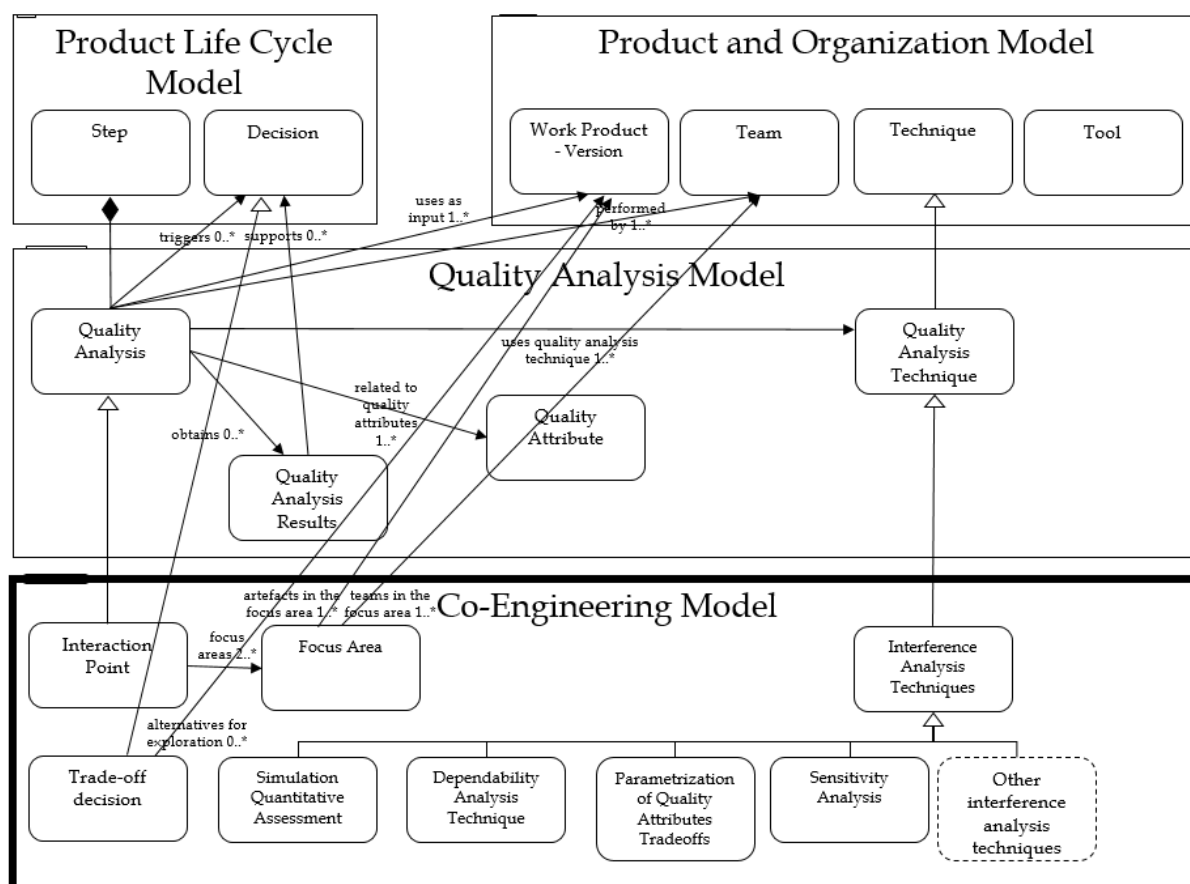


Figure 2-16: Relation between the PLC model, the product and organization model, the quality analysis model, and the co-engineering model

## 2.2.6 Traceability across the PLC

The presented model is expressive enough to capture relevant traceability information. This subsection discusses how traceability across the PLC can be achieved when these models are instantiated in terms of documents and other artefacts produced, actions decided, etc., in a specific project. First, the PLC differentiates the stages, and the stages and steps are activities in the PLC where we can add traceability to their corresponding work products' versions, the interaction points, and the decisions that took place in each PLC entity. In this way, for example, we can enable complete traceability if an interaction point triggered the decision to go back to a previous stage in the PLC to reconsider some requirement, or to make changes in the work products.

The next chapter goes into more detail about requirements on tooling to support traceability.

### 3 Tooling Requirements for Supporting Interaction Points and Traceability

Contributor: Magillem

The AQUAS approach implies that:

- Each IP will involve combined analyses encompassing various attributes; various forms of combined analyses are supported by the various tools available and/or enhanced in AQUAS, and part of WP4 work concerns the interaction between such tools; this chapter does not address these aspects;
- There will be an information flow from an IP to the later steps and stages of the lifecycle and in particular to the later IPs.

This chapter focuses on this second aspect, which lies at the interface between WP3 and WP4. A focus group was formed to analyse the requirements on tooling that arise specifically from the need to support IPs, that is, not just supporting combined analyses for safety, security and performance, but structuring the application of these combined analyses in the PLC in terms of IPs. The members were: Magillem (Emmanuel Vaumorin, Matthieu Pfeiffer, Vincent Thibaut), Intecs (Silvia Mazzini, Stefano Puri) and Ansys (Marc Born). The method adopted was to:

- Consider the inputs from WP3
- Use the description of the IPs from all UCs and analyze and generalize to feed and verify the proposed concept
- Define and propose a specific use model for IP
- Take into account the functionality available in existing tools.

This work, reported in this chapter, had these objectives: 1) to enable the implementation of support for IPs in several tools, provided by partners in the AQUAS project or to be developed further; 2) to ensure that those tools apply the concepts developed in the AQUAS methodology work on co-engineering; and 3) to illustrate the applicability of the tools with generic examples. We have considered the inputs from other deliverables of WP3, and also the descriptions of tooling support for individual use cases in WP4 deliverables.

#### 3.1 General scenario for supporting IPs

The following characteristics of an envisaged scenario for applying an AQUAS PLC have been extracted from D3.1 Chapter 4, on Interaction Points. They have been taken into account in the requirements for the tools supporting interaction points:

- IPs aim at ensuring satisfaction of various non-functional requirements, where "ensuring" means "achieving and demonstrating"
- It may not be feasible to expect a single team of experts to have the skills to perform specialist tasks related to different requirements, e.g. both cybersecurity and safety tasks to be performed by the same expert team
- So, specialists of different cultures are separated in "*focus areas*", or "*silos*"; information easily flows vertically within a focus area (a specialism) but not between them

- Non-integrated processes are thus present to deal with safety, security, performance, and communicate in more or less well-defined ways. In AQUAS these established means for interaction are known as interaction points

### 3.2 Typical expected needs

Some expected needs for recording and tracing IPs in an AQUAS-style PLC are :

R-IP01: For each IP, it is necessary to record that it took place, which analyses took place, what they concluded

For instance, the context of the interaction point creation, the team[s] involved, the list of modelling artefacts to be taken into account, the questions raised, the analyses which have been conducted, the conclusions of the analysis, etc.

R-IP02: For each IP, it is necessary to record what decisions were taken as a result of the analyses

For instance, a record that all was judged fine, so that development work would proceed further; or that more in-depth analyses were required; or that the requirements for some subsystem had to change (and what changes were decided); and for each decision taken, its genesis (who was involved and signed them off) and rationale (recorded so that at later stages in the PLC one can understand why the product is the way it is).

R-IP03: For each IP, it is necessary to record the changes made to artefacts as a result of decisions made

For instance, documenting that a specific parameter of the hardware platform was investigated and set to a value because it gave the best performance of software execution, while satisfying a safety requirement.

R-IP04: For each IP, it is necessary to record which iterations of that IP took place

The analyses performed at an IP may prompt changes to artefacts (possibly backtracking to an earlier stage in the PLC: e.g., an IP may trigger a decision to change requirements decided at that earlier stage), so that the IP (the set of analyses and the decisions) then must be repeated. The information to be maintained thus includes iterations of the IP, and, for each iteration, the various information about analyses, decisions, and artefacts affected or produced.

R-IP05: For each IP, it is necessary to record the artefacts produced for and by the analyses

Each analysis might involve models developed for that analysis - e.g. a SAN or FT model or flow chart, etc. and the outputs produced using that model.

R-IP06: For each IP, it is necessary to record any warnings or request for action to be propagated to later stages of work

This may involve later IPs, e.g. such a warning/request could be: "there is a potential problem with performance-security interaction, and whether this problem in fact arises needs to be checked once the code for subsystem Z has been written".

Remark: The focus group observed that these are special cases of more general capabilities, and chose to define these more general capabilities that would allow IP-supporting tooling to be configured for the needs of a specific AQUAS-enhanced PLC in a specific organization and industrial sector. These are listed in the next section.

### 3.3 Specific usage requirements for tools supporting Interaction Points

The following specific requirements for tools supporting interaction points are defined. These are requirements for the tool set chosen to support an AQUAS-style PLC: no specific set and allocation of responsibilities between them is assumed.

R-IP07: The tool set shall provide functionality to aggregate references of design artefacts or documents

The role of the tooling is to enable the IP stakeholders to select some artefacts in the information silos and to use references to those artefacts in order to create a new level of information at the IP level – a useful description of the IP and its results – aggregating the referenced data. For instance, a report on the combined analyses could include some references to several parameters concerning several components that are part of the subsystems involved.

R-IP08: The tooling shall not be only file-centric but should also offer some support for modelling semantics

Document formats may be very diverse, and could include special formats used by modelling tools. For instance, a reference to an artefact could be performed directly in a modelling tool in order to facilitate the selection of artefacts by the IP manager. For instance, a modelling tool could provide to an IP stakeholder the ability to select model elements as artefacts to be used in the IP, so that the references can be stored to achieve traceability between the IP report and the model elements referenced. The specific modelling semantics to be supported are not defined here, and will be the decision of each tool vendor depending on which other tools they wish to integrate; a minimal requirement, though, is that the mechanisms for referencing artefacts shall be compatible with any “text” based document.

R-IP09: The tool set shall provide a capability to collect definitions of exploration fields

Certain analyses may involve exploration by an analysis tool of a range of parameters or design characteristics, e.g. to perform sensitivity analysis or to seek feasible, or optimal, trade-offs. The tool set shall support entering the required selections of exploration field parameters for several types of artefacts.

R-IP10: The tool set shall bring traceability capabilities

The tool set shall have traceability capabilities to enable browsing from an artefact reference to all other connected artefacts. This kind of traceability capability, common to all tools, could be provided for instance by a traceability tool, but this requirement does not mandate that solution.

R-IP11: The tool set shall be compatible with any flow choices

To do so, the tooling shall be flow-agnostic, that is to say it will not introduce any specific semantics in the type of artefact to be managed. For instance, if the artefact is the representation of a fault tree, the tools shall provide a way to reference this fault tree or any element in this fault tree, at any granularity. So, any artefact is considered just as an artefact and in this example there is no need to specify that this artefact is a “fault tree”; we leave this work to the IP description.

R-IP12: The tool set shall provide a capability to register and aggregate the states of the artefacts relevant for a decision

The IP stakeholders shall be able to describe how the referenced artefacts are impacting their own artefacts, or additional artefacts created at the IP level. For instance, the tool set must allow a software stakeholder (that is, a person in charge of SW-related decisions in the IP) to record explanations about

how executing some software components on certain hardware processors could have an impact on execution times and on the safety requirements.

R-IP13: The tool set shall support the decision process but not take into account the flow of decisions downstream of the IP

The tooling shall provide the capability to manage the evolution of the IP iterations, but it might not provide capabilities to manage the decision flow itself. That is, when the interaction point has been fully described in the tool set, with decisions requiring actions within the silos, the IP record is then frozen and each silo manager goes back to their activity to decide how this impacts their next tasks to be performed. The results of these tasks performed within the silos may be fed back, as required, into a further iteration of the IP.

R-IP14: The tool set shall not integrate the loops of the decision process

The tooling functions for IPs shall provide support to describe the IP itself, but not the loops of silo activities in the flow that leads to the iterations of IPs. That is to say that the tooling shall support the management of several versions of the IP itself (see following requirement), but not include the silo-specific processes that use the information from the IP.

R-IP15: The tool set shall provide history information for information stored in IP description

The tooling shall provide versioning capabilities to support iterations of the IP: in Figure 3-1, an IP can have two or more iterations: records of all iterations, with modifications or additions to artefacts, shall be available for later reference, for instance to analyze the evolution of the design.

R-IP16: The tool set shall provide a capability to timestamp the referencing of information fragments<sup>4</sup>

The referencing of the fragments and of several modifications in the artefacts must be timestamped. A fragment (of information) is an atomic reference to a design artefact. When a fragment is used in an IP report, this fragment is timestamped and referenced. That means that in a further version of the IP, the fragment in reference could be changed (for instance I say that I'm not anymore taking into account this parameter A of the hardware platform but parameter B instead), and also the artefact could change (for instance the fragment on parameter A was done when A=1 and in another iteration, A=2). All those modifications must be recorded and traced to help IP stakeholder to be informed of modifications that could impact the analysis or further decision flow.

Note: Focus on Safety/Security/Performance trade-offs

In order to provide capability of inconsistency detection, risk analysis, and/or conflict detection, the tooling is dedicated to Safety/Security/Performance trade-offs, but other use models not defined here could be supported, because the implementation choices shall be generic.

In AQUAS, some tools like Magillem's are evolving from being general-purpose tools to include features to support IPs specifically, while others like CHESS or Medini, etc., which are natively able to

---

<sup>4</sup> In this context:

- An artefact is a piece of the design description; for instance: a processor instance in a hardware platform description in XML documentation standard
- A fragment (of information) is, in this example, the reference to that processor, so the piece of XML that is used in the standard to identify this processor; so the tool should store a reference of this fragment of XML code.

support a selected set of analyses, will take these requirements into consideration as inputs for evolution in the longer term.

### 3.4 Requirements on the use flow

The following figure illustrates how the tools dedicated to interaction points must manage the data used in the interaction points in a co-engineering context.

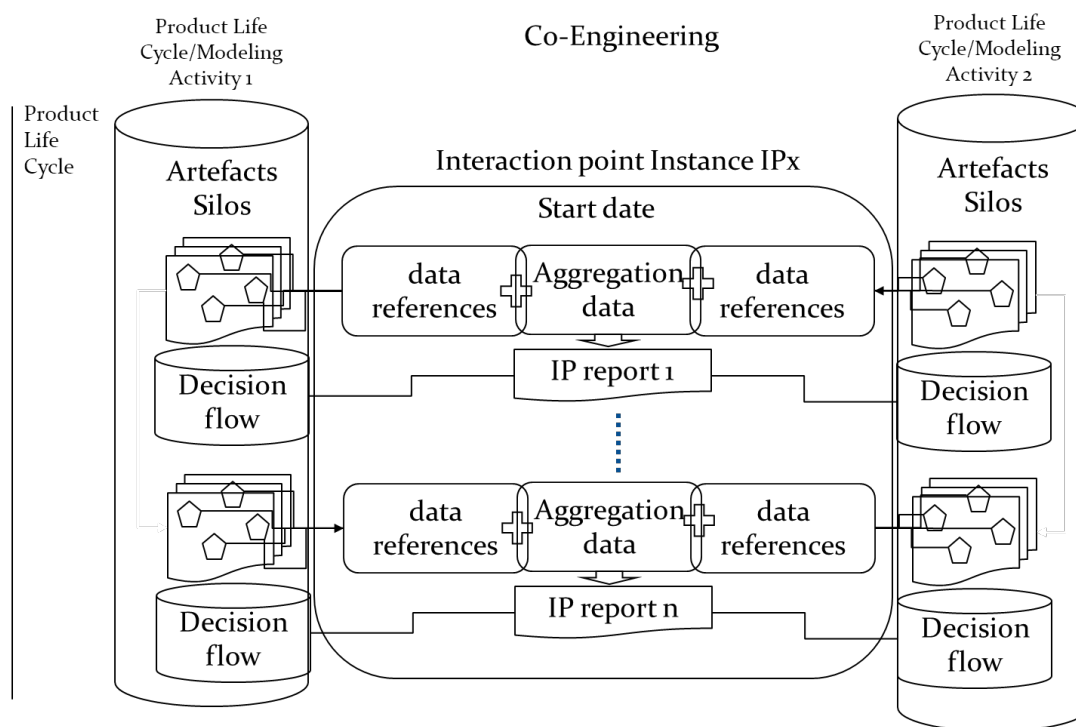


Figure 3-1: Use flow of an interaction point.

First one considers “*Artefact Silos*” which are the repositories of information dedicated to each specific modeling activity. These silos are used to support the product life cycle of the artefacts used in the activity. Those artefacts are considered as documents, in which a list of fragments of information are contained (represented as the pentagons). This first phase is detailed in Section 3.5.1.

Those fragments are defined at any granularity (fine or coarse) requested by the interaction point. The tool set supporting IPs must provide the capability to select the documents of each silo that are of interest for the interaction point and then provide the capability to select the fragments of interest. This import activity will be detailed in Section 3.5.2.

Then, in an instance of interaction point, at the first iteration, the fragments of information will be made available as references to realize an aggregation of data. This aggregation of data will be explained in detail in Section 3.5.3. The tools can generate the IP report #1 which will be used in the specific decision flow of every modeling activity. The cycle can then iterate with the same steps: for the next iteration of the interaction point, the same references or some additional ones are available for an updated aggregation of data and then for the generation of the new IP report. The IP tooling will provide features to trace the evolution and modification in referenced fragments.

### 3.5 Requirements on the use model

In this part, we detail the steps of the use model. In this use model, we have two kinds of roles:

- the silo manager is responsible for delivering the artefacts of interest and the fragments of information from its silo to the IP manager.
- the IP manager is responsible for collecting artefacts and fragments from each concerned silo manager, to build the IP description, to aggregate data and to build the IP report.

#### 3.5.1 Use model-Step 1: Files collection

The tooling provides a capability for users to select artefacts, and creates references to them through fragments of information across all design files. This gives the ability to have references to all files that are to be used in the analyses and/or discussions that are part of an interaction point. These may include several files for each silo (examples: hardware architecture description, requirements, performances reports, configuration documents, etc.). Also, previous versions (if they exist) of the IP report are imported. The tools fragment the documents (coarse or fine grain). At this point, the IP is identified with its name and version, and in a first phase the IP stakeholders have access to this list of references of files located in the silos, and for each file, the contained fragments of information is available for selection in the next phase.

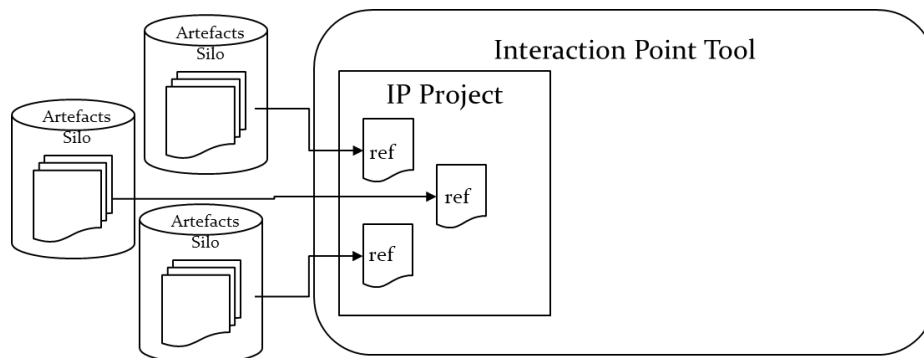


Figure 3-2: File collection in the IP. *Note:* "IP project" is, following ECLIPSE terminology, a tool-defined entity in which the references and other information created for the IP are stored.

#### 3.5.2 Use model-Step 2: Fragments collection

In each of the selected files, the fragments of information to be used for the IP activities are selected and referenced into an initial version of the IP report. An IP will always involve some kind of analysis, plus perhaps some discussions, explorations of possible trade-offs, decisions, changes to various artefacts. For instance, we consider the activity of seeking a trade-off.

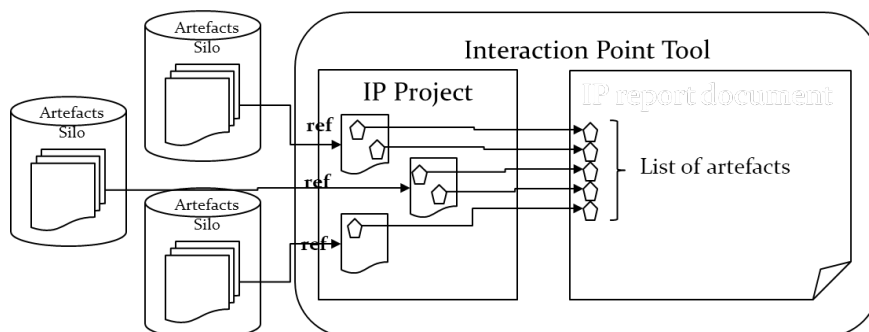




Figure 3-3: Fragments collections in the IP

### 3.5.3 Use model-Step 3: Trade-off description

Starting from the initial IP report, the IP manager writes, with the assistance of the silo managers, in natural language, the expression of each trade-off in the IP.

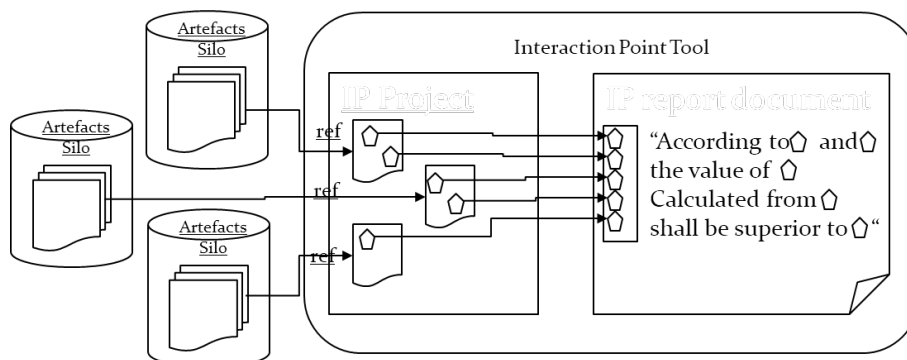


Figure 3-4: Trade off description

### 3.5.4 Use model-Step 4: Action plan description

All actions decided that must be performed as silo activities in relation to the selected artefacts are described in the IP report. Thus, this report will be referenced and used by the silos to perform redesign of system parts, changes to requirements, explorations of candidate designs, etc.

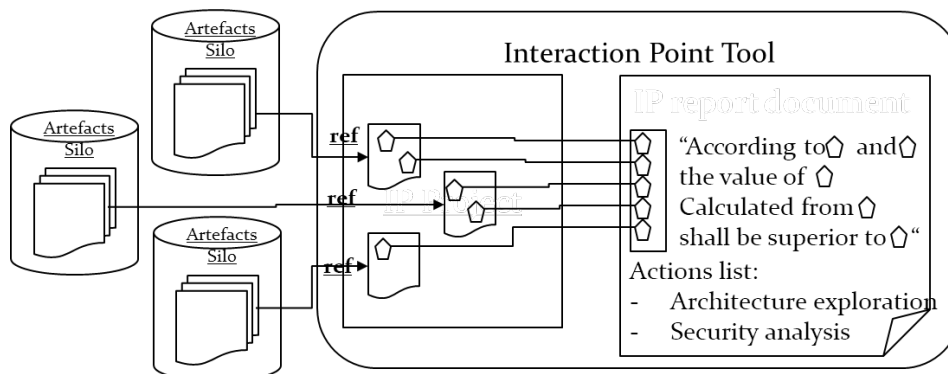


Figure 3-5: Action plan description

## 3.6 Conclusion

This chapter proposes a list of requirements for implementing tools that support the IP concept in AQUAS, in its entirety or limited to some of the functions required by IPs. These requirements are generic enough to support a variety of design and decision flows. They have been produced taking into account some existing functionalities of AQUAS partners' tools, so that the effort to prototype a demonstration solution in the frame of the AQUAS project is minimal; but other solutions may be envisaged following these requirements, ensuring potentially broad dissemination in the industry. An example of use of the Magillem Content Publisher (MCP) to support IPs in UC4 (industrial Drive) is in section 4.2.5 of deliverable D2.3.4.



## 4 Methods for Combined Analyses

This Chapter presents methods for combined analysis, which are being applied in the AQUAS project. For clarity of explanation (as per the AQUAS workplan), the methods are presented via small examples of application to problems in the AQUAS use cases; but they are grouped independently of the use cases, as the goal is to explain techniques that can be applied in a range of domains. In fact, several of the methods are presented with examples in more than one use case; for example, extended fault-tree analysis applied in both the space and medical use case (Section 4.9), and the Hepsycode methodology and framework applied in the ATM and space use cases (Section 4.6).

The methods are presented in three categories. The first category includes methods related to hazard identification that can be applied at the stage of requirement specification and conceptual design: HAZOP and HARA, and trade-off analyses triggered by the identified hazards (Sections 4.1-4.3). A second group of various methods (Sections 4.4-4.6) supported by various tools is applicable in lifecycle stages following the Requirements Phase for various forms of combined analysis of safety, security and performance (Sections 4.7-4.9). The final category of methods is concerned with verification and validation activities, for example in terms of conformity of source code with its requirements, or verification of timing requirements (Sections 4.10-4.13). Each method is defined by its aim, method, results, lessons learned, and when appropriate, further developments of the technique, or applications of it planned within AQUAS.

### 4.1 Hazard and Operability Analysis for identifying safety/security interactions at requirement/conceptual design stage (Medical Use Case Example)

Contributors: City and UC2 partners

Hazard and operability (HAZOP) analysis [IEC 61882:2001] is a structured and systematic method to identify potential hazards and determine appropriate mitigation strategies. Although originally developed for detecting safety hazards due only to accidental causes (that is, not security-related), and not generally applied to security matters, here we confirm its utility in a combined analysis involving malicious causes of hazards, i.e., the combined analysis of both safety and security.

A HAZOP exercise was applied in the Requirements Phase in the medical use case (Use Case 2).

We recall, as discussed in Deliverable 2.2 Chapter 1, that UC2 concerns the extension of an existing device for monitoring blood pressure and neuromuscular transmission to make it able to control an infusion pump and perform closed-loop control of these physiological parameters.

The subset chosen for analysis is a specific, informative use scenario: closed-loop control of blood pressure during a surgical intervention. This use scenario is expected to exhibit many of the new hazards that arise when extending RGB's previous monitoring device to the new, closed-loop control device. Specifically, it describes the steps and possible user interventions that may occur when maintaining a patient's mean blood pressure at 70 mm Hg by automatically calculating and infusing appropriate doses of Sodium Nitroprusside (SNP) and Glycerine Trinitrate (GTN).

#### 4.1.1 Aim of the example

The goal of the HAZOP analysis was to review the system requirements and preliminary design of RGB's closed-loop device, to identify potential hazards that could arise during the identified use scenario. Importantly, HAZOP relies on bringing together the various viewpoints/knowledge of the participants in a HAZOP session (in this case, for example, Trustport provided the security viewpoint, City

represented human factors concerns). Potential hazards, especially those with a high likelihood and severe consequence, highlight areas in the system that require appropriate remedies, or additional analyses about the likelihood of deviations, their consequences and the effectiveness of mitigation measures. Such remedies could result from work/suggestions by a single partner, or require the collaborative analysis of various partners of different specialties. Among further analyses that can be triggered, potential hazards, or feared events, identified in the HAZOP can be input to e.g. developing fault trees, and FMVEA tables, which further enhance the understanding of hazards and their propagation within the system.

#### 4.1.2 Method

For use as a HAZOP worksheet, an online, shared Excel sheet was set up with the headings shown in the example in Table 1. Four two-hour teleconferences were run between June and September 2018. A HAZOP analysis proceeds by systematically applying a series of guidewords to each step in the use scenario: for each guideword, session participants try to identify potential deviations of the system behaviour from the design intent. The guidewords<sup>5</sup> act as triggers to stimulate participants to envisage how the deviation might occur and its potential consequences. For each deviation, possible causes and likelihood, consequences and severity, existing safeguards, and action points (some requiring work outside the scope or timeframe of the AQUAS project) were documented. A recorder documented the discussions in the HAZOP worksheet. As the teleconferences were relatively short for the required task, participants were also encouraged to individually update/add to the shared spreadsheet using a “name tag” to identify their entries. After the fourth session, partners noticed the analysis had reached a phase of diminishing return and the HAZOP table for this scenario was deemed to have adequately demonstrated the usefulness of this style of HAZOP for this role in AQUAS-style co-engineering.

#### 4.1.3 Results

Below is an example of a single row from the HAZOP analysis worksheet. The complete worksheet can be found in the annex (status: project confidential).

---

<sup>5</sup> The guidewords applied were: no (not, none), other than (wrong/maliciously), early/late, before/after, faster/slower, where else, part of, less (lower), more (higher), as well as (more than), and reverse. For each guide word, participants were reminded to consider both accidental and malicious causes of deviations. At least one previous study of HAZOP applied to security (Winther R., Johnsen OA., Gran B.A. (2001) Security Assessments of Safety Critical Systems Using HAZOPs. In: Voges U. (eds) Computer Safety, Reliability and Security. SAFECOMP 2001. Lecture Notes in Computer Science, vol 2187. Springer, Berlin, Heidelberg ) proposes a far more complex set of guidewords but we opted for this simpler method considering that long lists may end up being counterproductive by taxing the ability of participants to stay focused.

Table 4-1: Example row from the HAZOP analysis

Step in the Use Scenario	Guide word	Deviation Identified	Possible Causes and Likelihood	Consequences and Severity	Existing Safeguards	Notes/ Recommendations	Action Points
Device is connected to tree of infusion pumps and user verifies that the connection is performed correctly	Other than (wrong)	Device connects to a different pump than the one intended.	Possible with point-to-point connection if the user connects the wrong cable, or with Ethernet connection.  Likelihood depends on hospital practice/user experience.	Could change dose rate for another patient, and/or fail to provide calculated drug dose to given patient.  Thus, major/maximum severity level consequences.	(1) Device or manual warns the user that they need to verify the pump's behaviour or the dose rate being delivered.  (2) User verifies that pump is performing correctly by observing its behaviour.	[City] Existing safeguards rely heavily on the user. Perhaps other safeguards could be built into the device to assist.  [RGB] It could be possible to include in the device a previous validation window to check the connection to the right infusion pump.	To study general security requirements, especially given Ethernet connection

The HAZOP analysis sessions revealed a number of interesting results that can be grouped into four main categories: (1) areas requiring further specialist analysis, (2) areas requiring further combined analysis, (3) changes to device design to help mitigate identified risks, and (4) useful inputs to other analyses in the Requirements Phase. Examples of these four categories of results are described below.

- *Areas requiring further specialist analysis:* In response to risks associated with the connection between the pump tree and the device (for example: another device influencing control of the pump, or the control device connecting to a different pump than the one intended, etc.), a need for further specialist security analysis looking at general security requirements of the Ethernet connection, as well as encryption options was identified.
- *Areas requiring further combined analysis:* Among inter-attribute concerns, a trade-off was noted involving user authentication. To enhance security of the device, some form of authentication (e.g. a pin/card) is likely required for important changes to the device's parameters (e.g. blood pressure target). However, this authentication must also allow the user to operate the device quickly for efficient care of the patient and must not introduce new safety hazards due to delays in decisions/actions. This point raises the need for further combined analysis of these trade-offs by security and human factors specialists, and this trade-off analysis is discussed further in Section 4.3.
- *Changes to design to help mitigate identified risks:* Within the steps involved in the set-up of the device, several hazards related to incorrect sequence of steps, or missing steps were noted (i.e., user forgets to load the syringe, or connection between the pump tree and device was not established, etc.). As a potential mitigation to these, RGB has included in their new design of the device interface a validation sequence (including a set of mandatory steps and input parameters) to aid the user through the various set-up steps and confirm that they have been correctly performed before initiating automatic control.
- *Input to other analyses in the Requirements Phase:* The HAZOP exercise links to the work of other partners in the medical use case. The feared events identified in the HAZOP become inputs to the Fault Tree Analysis (FTA), Hazard Analysis and Risk Assessment (HARA) / Threat

Analysis and Risk Assessment (TARA), and the cyber risk analysis. These possible combinations of methods are shown in Figure 4-1.

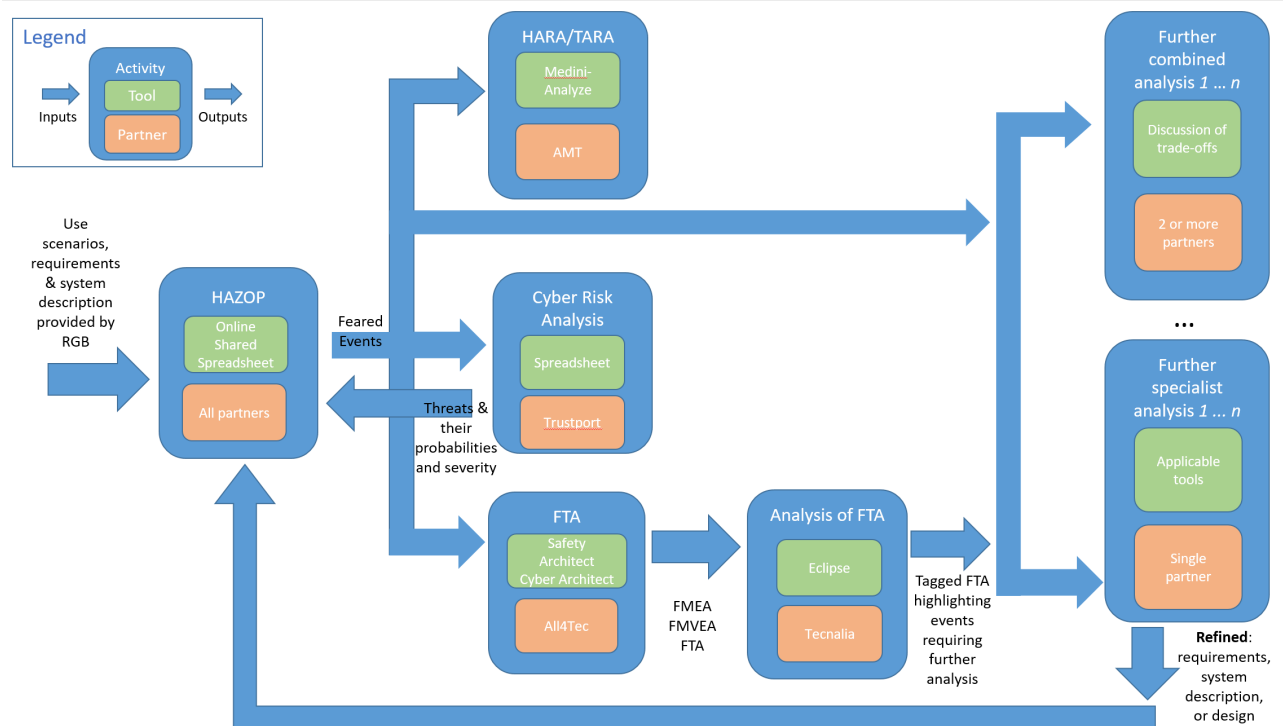


Figure 4-1: Link Between HAZOP Analysis and other Analyses by UC2 Partners in the Requirements Phase

#### 4.1.4 Lessons Learned

The HAZOP Analysis fulfilled its purpose as a useful method to share the knowledge and expertise of different partners, combining multiple perspectives, in the medical use case. Although most partners involved had no previous experience of HAZOP, it was productive; no major difficulty was experienced and the mixed guidewords seemed effective for the combined analysis of accidental and malicious deviations and consequences. In the early Requirements Phase of the medical use case, it successfully identified a range of hazards, which served as triggers for changes to the device design and also as useful inputs/triggers to additional combined/specialist analyses, run by various partners: (1) All4Tec and Tecnalia in developing their combined fault tree analysis, (2) AMT in developing their combined Safety HARA and Security TARA described further in Section 4.2, (3) City and Trustport in exploring authentication trade-offs described further in Section 4.3 and (4) Trustport in progressing their specialist cyber risk analysis.

There were limitations in this trial application. HAZOP is meant to be a heavily interactive process, so working remotely via teleconferences and using an online, shared spreadsheet was a challenge. Some stakeholders were also not represented as would be expected in a complete HAZOP analysis; for example, RGB was compelled to act as a representative of end-users. Both these observations suggest that the technique would be more effective, not less, in the usual industrial environment than it is in the environment of the research project. Apart from effectiveness (finding hazards that would not be found without this technique), the other objective of HAZOP (or any other structured hazard identification technique) is *completeness* (finding them all); how close a technique gets to completeness cannot be tested by a trial on a very limited effort and timescale (and, especially, without the ultimate test of system operation to reveal whether essential hazards were left unnoticed and

untreated), as imposed by the size of AQUAS. However, we will monitor whether later verification steps reveal hazards that were missed by the HAZOP parts performed.

#### 4.1.5 Further Developments

The HAZOP analysis of the closed-loop control of blood pressure during a surgical intervention scenario is now at a stage that helps demonstrate its usefulness and has provided useful outputs. Currently, RGB and UC2 partners are discussing how best to move forward within the AQUAS budget constraints: to extend the HAZOP analysis to other scenarios of use, especially in the intensive care unit (ICU), or to use AQUAS resources on some of the other analyses currently underway in this use case, especially those that have been triggered by the HAZOP analysis.

The complete HAZOP worksheet is provided in the Annexes (status: project confidential).

### 4.2 Combined Hazard Analysis and Threat Assessment Including a Threat Identification Based on Assets (Medical Use Case Example)

Contributor: AMT

A hazard and risk assessment is a structured and systematic method to identify potential safety hazards. Thereby, potential malfunctioning behaviours of a device are considered in a certain usage. Depending on the likelihood of that scenario and the severity of the potential hazard, a criticality level is determined. This allows later application of appropriate mitigation strategies. In the medical domain, standards like ISO14971 define the application of this method.

Threat Assessment is a method to estimate the credibility and seriousness of a potential threat, as well as the likelihood that the threat will happen. This implies the identification of the threats.

An exercise for a combined hazard and threat analysis including a threat identification based on assets modelled in SysML was applied to the Requirements Phase in the medical use case (Use Case 2). UC2 concerns the extension of an existing device for monitoring blood pressure and neuromuscular transmission to make it able to control an infusion pump and perform closed-loop control of these physiological parameters [Deliverable 2.2].

The analysis was based on the same example that was chosen for the HAZOP analysis in Section 4.1. Additionally, the results of that analysis were used as an input for the threat assessment.

#### 4.2.1 Aim

A key aim of the combined analysis and assessment of hazards and threats is to identify them and to estimate their criticality. At the same time, it helps to expose the relations between threats and hazards to emphasize those threats that potentially lead to safety hazards. Thereby tool-based automated threat derivation from a design model provides a systematic way to find new threats which can then be assessed in a threat assessment.

#### 4.2.2 Method

The tool medini analyze, which is dedicated to performing analyses in the area of functional safety and cybersecurity, was first applied to perform a hazard analysis based on the high-level function of UC2, namely, to inject a medication to control a patient's blood pressure. This function was modelled in medini analyze:

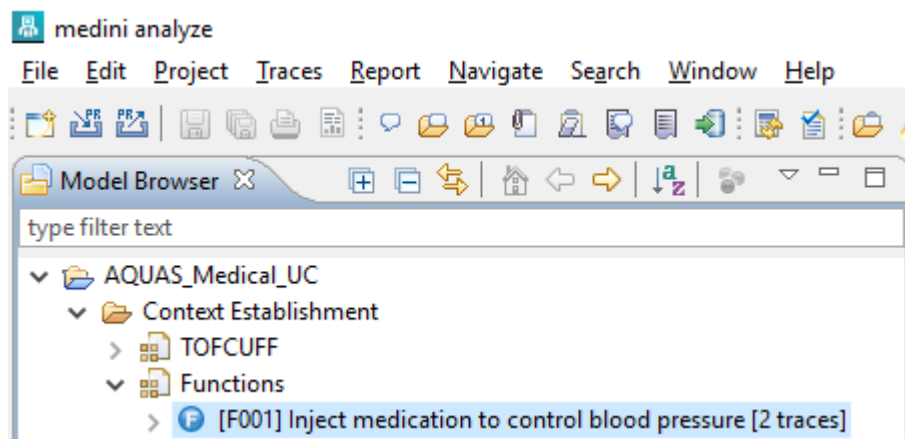


Figure 4-2: Initiating a new function in medini analyze

The required malfunctioning behavior of that function as an input for the hazard analysis was modelled as a set of malfunctions that are found by performing the medini analyze built-in HAZOP analysis:

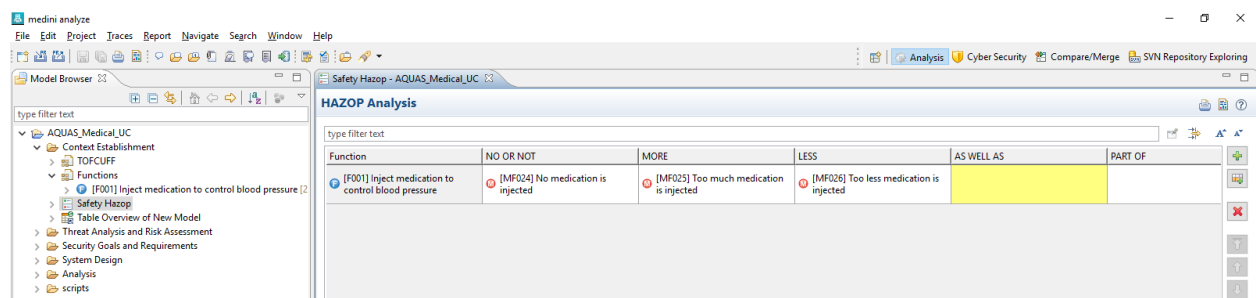


Figure 4-3: Applying HAZOP analysis in medini analyze

Here the table editor shows in the columns the guide words to be applied to the function in the rows. The intersecting cells can be used to define the malfunctions that match the guide word in the context of the function.

While in Section 4.1 HAZOP was applied on a process view of UC2, here the HAZOP is used on a functional view of the system. In the process view the guide words help to identify deviations from the intended sequence of steps in the process whereas in the functional view the guide words support the derivation of malfunctioning behavior of the analyzed function. As shown later in this Section, medini analyze as a model-based tool could have been used too to execute a HAZOP on the process view of a system.

The found malfunctions are then input for the hazard analysis whereby medini analyze allows filling the lines in the HARA table either manually or automatically by permutationally combining the set of malfunctions with a list of operation modes.

Scenario Analysis

type filter text

ID	Operation Mode	Malfunctioning Behaviour	Hazard	Potential Effect	Safety Goal	Safe State
HE004	during surgery	[MF024] No medication is injected	Hemorrhage	fatal bleedings		
HE005	during surgery	[MF025] Too much medication is injected	Ischemia	fatal multiple organ failure		
HE006	during surgery	[MF030] Wrong medication is injected	Ischemia	fatal multiple organ failure		

Figure 4-4: Identifying malfunctions in medini analyze

For the threat assessment, again medini analyze is used. In order to come up with an initial set of threats, the tool allows, in the step of threat identification, the automatic derivation of threats out of a SysML model that either describes an architecture of the target of evaluation (TOE) or defines the item under consideration on a functional level. The SysML model is used to annotate the artefacts, whether they are assets or not, and what security attributes are of interest.

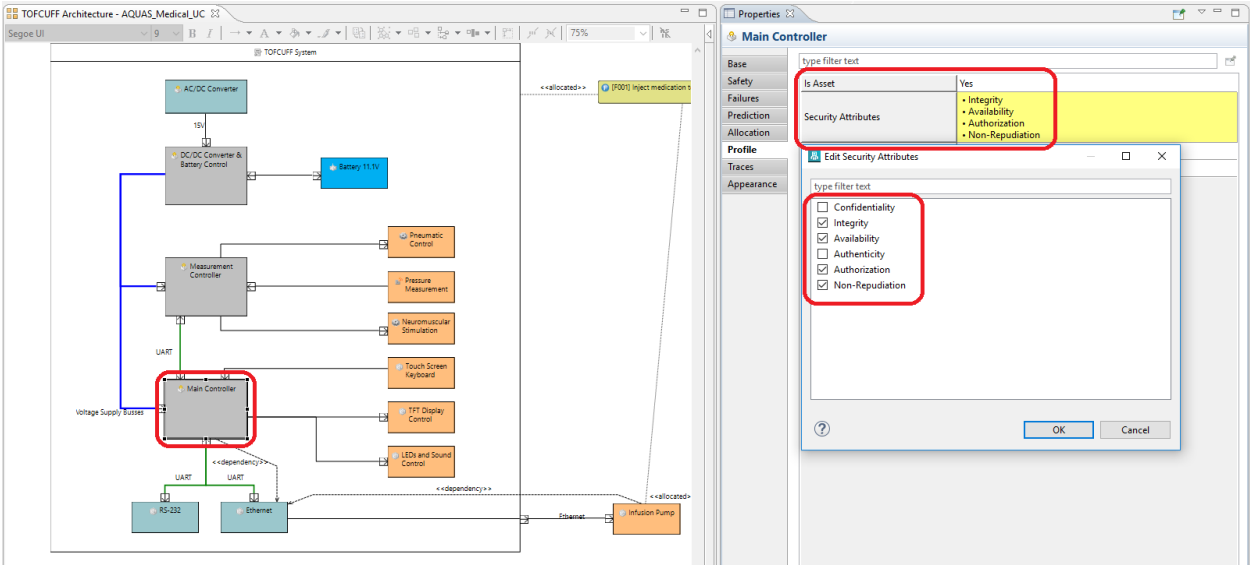


Figure 4-5: Asset Identification in medini analyze

From the annotated model, threats are derived automatically by creating a threat for every security attribute of an asset whereby the security attributes are mapped to a corresponding STRIDE category [STRIDE].

Context Establishment

- TOECUFF
  - New
  - Derive
  - Open Editor
  - Open With
  - Related Elements
  - Open Trace Matrix
  - Allocate Elements
- Threat A
  - Safety
  - Prediction
  - Security
    - Fill Threat Identification Analysis...
    - Fill Risk Assessment Analysis...
    - Pre-fill Risk Treatment

type filter text

ID	Name	Affected Asset	Category
TH_01	Tampering of Main Controller	Main Controller	Tampering
TH_02	Denial of Service of Main Controller	Main Controller	Denial of Service
TH_03	Elevation of Privilege of Main Controller	Main Controller	Elevation of Privilege
TH_04	Repudiation of Main Controller	Main Controller	Repudiation
TH_05	Tampering of Pressure Measurement	Pressure Measurement	Tampering
TH_06	Spoofing of Infusion Pump	Infusion Pump	Spoofing
TH_07	Elevation of Privilege of Infusion Pump	Infusion Pump	Elevation of Privilege
TH_08	Tampering of Ethernet	Ethernet	Tampering

Figure 4-6: Threat derivation from assets

Triggered by a tool interaction, the derived threats collected in the list are then filled into a threat assessment table that allows estimation of their severity according to different fields (e.g. safety,



financial, etc.) and their likelihood. The estimation of the corresponding parameters for impact and likelihood follows the definition published in the HEAVENS security models [HEAVENS].

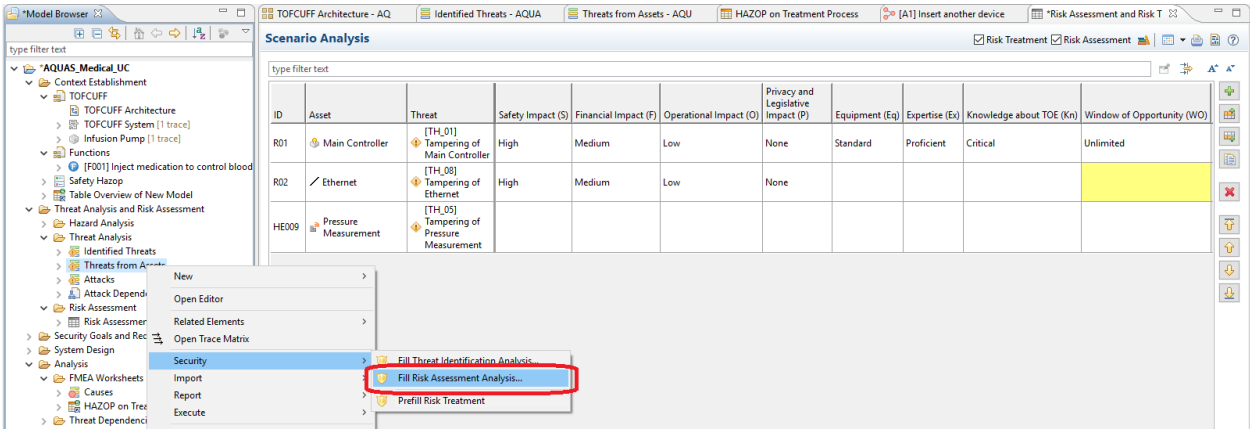


Figure 4-7: Threat assessment table in medini analyze

The Excel-based HAZOP analysis introduced in Section 4.1 and its corresponding annex was imported manually into medini analyze and helped identify potential attacks that in turn can cause one of the identified threats. The resulting table in medini analyze (see Figure 4-8) also allows tracing of malfunctioning behaviour resulting from the hazard analysis as effects of an attack. Here the relation between security and safety aspects becomes visible.

Component/Function	Guideword	Deviation	Derived Attacks	Potential Effekt
[F001] Inject medication to control blood pressure				
⊖ Patient is placed on the operating table				
⊖ Patient is prepared for surgery				
⊖ Device is switched on				
⊖ Cuff is placed on a free arm of the patient				
	Other than (wrong)	[MF001] Device connects to a different pump than the one intended.		[F001] Inject medication to control blood pressure [MF030] Wrong medication is injected
⊖ Device is connected to tree of infusion pumps and user verifies that the connection is performed correctly	Other than (maliciously)	[MF002] Another device between the device and pump influences the control of the pump.	[A1] Insert another device between original device and infusion pump	[F001] Inject medication to control blood pressure [MF025] Too much medication is injected [F001] Inject medication to control blood pressure [MF024] No medication is injected [F001] Inject medication to control blood pressure [MF026] Too less medication is injected

Figure 4-8: Imported process HAZOP and derived attacks














type filter text				
ID	Name	Affected Asset	Category	Attack Scenarios
TH_01	Tampering of Main Controller	 Main Controller	Tampering	 [A2] Unauthorized person configures the device wrongly by intension
TH_02	Denial of Service of Main Controller	 Main Controller	Denial of Service	
TH_03	Elevation of Privilege of Main Controller	 Main Controller	Elevation of Privilege	
TH_04	Repudiation of Main Controller	 Main Controller	Repudiation	
TH_05	Tampering of Pressure Measurement	 Pressure Measurement	Tampering	 [A4] Excessive cuff pressure on the patient
TH_06	Spoofing of Infusion Pump	 Infusion Pump	Spoofing	
TH_07	Elevation of Privilege of Infusion Pump	 Infusion Pump	Elevation of Privilege	
TH_08	Tampering of Ethernet	 Ethernet	Tampering	 [A1] Insert another device between original device and infusion pump

Figure 4-9: Linking attack scenarios to threats

### 4.2.3 Results

The results of the applied method are a list of safety hazards, threats and attacks and their relation between each other in respect to causes and effects.

**Hazard Collection**

ID	Name	Kind	Severity	Occurrence	Description	Affected Asset	Attack Scenarios	Category
Hz01	Ischemia		5	2				
	Hemorrhage		0	0				

**Threat Collection**

ID	Name	Affected Asset	Category	Attack Scenarios	Description	To Be Assessed
TH_01	Tampering of Main Controller	Main Controller	Tampering	[A2] Unauthorized person configures the device wrongly by intension		Yes
TH_02	Denial of Service of Main Controller	Main Controller	Denial of Service			No
TH_03	Elevation of Privilege of Main Controller	Main Controller	Elevation of Privilege			No
TH_04	Repudiation of Main Controller	Main Controller	Repudiation			No
TH_05	Tampering of Pressure Measurement	Pressure Measurement	Tampering	[A4] Excessive cuff pressure on the patient		Yes
TH_06	Spoofing of Infusion Pump	Infusion Pump	Spoofing			No
TH_07	Elevation of Privilege of Infusion Pump	Infusion Pump	Elevation of Privilege			No
TH_08	Tampering of Ethernet	Ethernet	Tampering	[A1] Insert another device between original device and infusion pump		Yes

**Attack Collection**

ID	Name	Attack Category	Dependencies	Description	Context Information	Expertise (Ex)
A1	Insert another device between original device and infusion pump	Man in the Middle	[E11] Insert another device between original device and infusion pump			
A2	Unauthorized person configures the device wrongly by intension		[E09] Unauthorized person configures the device wrongly by intension			
A3	Burgle the surgery					
A4	Excessive cuff pressure on the patient		[E12] Excessive cuff pressure on the patient			

Figure 4-10: Identified hazards, threats, attacks and the relations between them

The impact that an attack has can be made visible in the relations view of medini analyze. Here it can be seen that an attack might lead to a threat and the threat itself could cause some malfunctioning behaviour of the device that controls the infusion pump, resulting in one of the serious safety hazards ischemia or hemorrhage, depending on the specific malfunction.

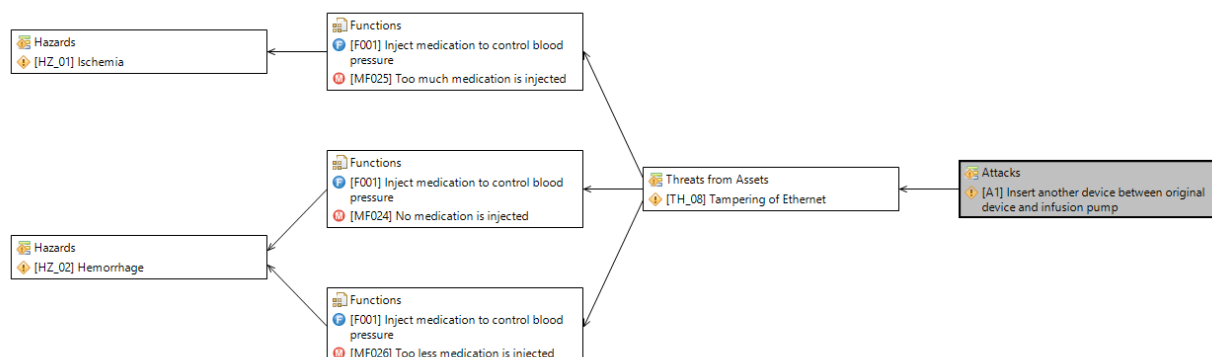


Figure 4-11: Relations view in medini analyze

#### 4.2.4 Lessons Learned

The combined Hazard Analysis and Threat Assessment as presented here is a suitable method to identify hazards and threats. In particular, the automated derivation of threats based on an architecture model, enriched by the possibility to mark certain artefacts in the model as assets with corresponding security attributes, enables improved argumentation with respect to the completeness of the potential threats that must be assessed.

Furthermore, it has been proven that the results of the HAZOP analysis described in Section 4.1 were a good starting point for further analysis, especially with respect to security. Thus, attacks could easily be derived from the deviations that were described there.

### 4.3 Combined Analysis of Trade-Offs Regarding User Authentication (Medical Use Case Example)

Contributors: City and Trustport

This analysis aims at clarifying trade-offs that arise from a novel security requirement. Its intended output is a description of the risk associated to each alternative design solution, so that designers can choose on a rational basis.

This is an example of a challenging combined analysis in which (a) security controls meant to preserve safety of operation conflict with safety and operation performance goals and (b) users' attempts to preserve safety and performance may impair security. That trade-offs exist is clear, but the analysis needs to start with identifying how they arise – the causal chains, complex and possibly including loops to be analysed – and choosing what aspects of alternative design solutions need to be presented to the system designers for them to be able to make informed decisions. Ideally, one can then translate this complex description into quantitative estimates of overall risk associated with each design solution. Whether this quantitative description is feasible in practice is to be seen, and if not, the goal will be again to assist the system designer with the part of analysis that *is* feasible.

This kind of complex trade-off is likely to arise in any safety-critical system with human operators; we are applying this analysis in the AQUAS medical use case (UC2).

UC2 concerns the extension of an existing device for monitoring blood pressure and neuromuscular transmission to make it able to control an infusion pump and perform closed-loop control of these physiological parameters [Deliverable 2.2].

Risks associated with malicious use or unintentional misuse of the closed-loop control infusion pump, identified in the HAZOP analysis presented in Section 4.1, raised a need to consider a form of user authentication in the UC2. Although authentication aims to reduce security risks related to accidental or malicious use of the device, it has negative effects as well: it may be a nuisance to some users, especially if repetitively required; may reduce their efficiency by causing delays; and most importantly, it may inhibit a clinician's timely response to patient emergencies, thus posing a safety hazard. These considerations imply a need to perform a trade-off analysis.

The analysis begins by describing various authentication methods (knowledge, token, and biometric-based) from the different viewpoints (security, performance, usability, cost, and safety). This allows a pre-selection of methods; this descriptive work is now being extended to a deeper, and potentially quantitative approach. In this use case, we limit this deeper analysis to token-based authentication methods, as the preliminary descriptive analysis indicated them as the most promising authentication approaches.

### 4.3.1 Aim

The goal of the analysis is to describe to a system designer the effects on risk of various options for authentication (and to explore whether this description is complete enough to turn the problem into one of numerical constrained optimisation of a single variable describing risk). The options are not limited to matters of technology ("shall I use passwords or RFID badges? Shall I add a second factor for authentication, say fingerprints?") but involve *system design* options about *how the system will use* authentication, as in the following list:

1. Is authentication required in this medical device, and against what threats/risks?
2. Is authentication required for all modes of use of the device (operating room (OR) and intensive care unit (ICU)), or could certain modes require different levels of authentication strength, due to differences in physical security?
3. Which tasks require authentication? For example, monitoring versus infusion.
4. How often should a user need to authenticate themselves? For each command they wish to input? For whole surgery interventions, or watch shifts? Or periodically? What is an appropriate "grace period"?
5. What is more appropriate, single or multi-factor authentication, and what type of authentication is most appropriate?
6. For the chosen method of authentication, are there policies or design variations that can help mitigate the downsides of that choice? For example password policies, or "break-glass" options.

These questions also require understanding of the system's design constraints (e.g., does the device need to have both an "authenticated" and a "non-authenticated" operation mode, so that the setting of this mode becomes a critical operation to be protected by authentication?) and intended secondary purposes (e.g., ensuring that logs of operation of the device are correctly attributed to the clinicians in control, e.g. to support incident analysis in case of adverse events). A first round of conversation about these aspects with the use-case owner RGB took place to help weed out some options; it is expected that at least one additional iteration will be needed.

### 4.3.2 Method

The analysis begins by brainstorming to identify which specific aspects (of authentication solutions) should be analysed. For example, performance of the authentication mechanism is an important aspect, but which specific aspects of performance need to be considered (average time to authenticate, probability of failure on first attempt, probability of delay long enough to cause patient harm, etc.)?

Following this initial brainstorming of important aspects, a descriptive analysis began. This analysis is documented in a worksheet describing the different authentication options, and the associated Cost (Development and Maintenance), Usability, Performance, Security, Safety (First-Order Risks and Indirect Effects) of each. These descriptions are based on several teleconferences between human factors specialists at City, security specialists at Trustport, and the device manufacturers at RGB.

Equally important was to identify minimum requirements for each of these aspects, to eliminate unacceptable options. For example, the time required to input a password is considered too long a delay and too disruptive for passwords to be considered a practical authentication option.

### 4.3.3 Results

Below is an example of a single row from the analysis table. The complete worksheet can be found in the Annexes (status: project confidential).

Table 4-2: Example row from the qualitative trade-off analysis

Authentication	Cost		Usability	Security	Safety	Performance	SSP	Potential Tweaks
Level/Type of Authentication	Vendor's cost	Adopter's Cost	Usability	Change to Security Risk (Intended)	First-Order Safety Hazards	Delay to Intervention (if not infinite)	Indirect Negative Effects to Consider	
Description of the Factor as well as Potential Measure(s) that May Be Used to Quantify It	Cost may be measured in hours/pounds and should include cost of any required hardware as well as development cost	Cost may be measured in hours/pounds. Adopter costs include: Set-up costs (ex: training); Enrollment time (ex: time needed to set up a new user account); Maintenance costs (ex: protecting the database, printing cards); and Recovery costs (ex: replacing stolen/damaged card).	Usability is a wide term meant to capture factors such as: ease of use, learnability, frustration, acceptability, accessibility, convenience, and universality. It may be measured on a Likert scale based on a survey of potential users, or instead approximated according to other factors and how they affect usability (see dependency diagram).	Security risk measured as probability of the hazard * severity of the consequence. For all rows, the main risk considered is incorrect modification of target parameters. This can cause serious harm; severity of the risk is unchanged for all rows and only the probability changes. All authentication levels are also expected to improve accountability/traceability not just in case of faults, but also for learning purposes, etc.	Safety risk measured as probability of the hazard * severity of the consequence. For all rows, the main risk considered is that the patient requires emergency treatment, which is delayed due to authentication. This can cause serious harm; severity of the risk is unchanged for all rows and only the probability changes.	Different time measures may be considered: average delay imposed (in seconds), probability of failure on first attempt, probability of a large enough delay to cause task disruption, and/or probability of a large enough delay to harm the patient.	Indirect risks measured as probability of the hazard * severity of the consequence	Tweaks that may be implemented in combination with the authentication method and which are usually designed to mitigate/address some of the risks raised.
Card	Card reader	Printing card for each new user; Replacing damaged cards	Fairly easy to use; Reliability of the card system will likely affect usability	Probability of security hazards, especially unintentional ones, is reduced. Exception: malicious use can occur if card is stolen.	Exceptions affecting likelihood of the risk: User forgets to carry their card; User fumbles with card/pockets; User swipes the card incorrectly; Card is damaged; Card reader failure	Ideally takes less than 5 seconds (for a user carrying their card...)	Users may lend their cards to others for convenience/speed, especially in an emergency	Reliability and ease of use of the card reader (i.e., swipe versus contactless) will likely affect performance, usability, and first-order safety hazards; Certain high-priority alarms override the need for authentication; Use the card for other purposes around the hospital to improve usability

Some observations about Table 4-2 follow.

- Some of the authentication solutions (rows), such as password authentication, can be eliminated based on the minimum requirements set out under each aspect. This helps simplify the later, detailed analysis.
- One of the solutions compared must be "no authentication": there is no a priori certainty that authentication will reduce overall risk. In particular, an authentication mechanism is a desirable target for DoS attacks, meant to make the device unavailable.
- There is a need to consider indirect negative effects. For example, users that find an authentication method inconvenient or difficult to use may invent workarounds (such as sharing the token or posting it in a visible location) that introduce new hazards or reduce the protection offered by the authentication. Such indirect effects are often neglected in SSP analyses, but considering them may reveal that the solution considered least risky based on direct effects is actually riskier than some alternative<sup>6</sup>.
- Each column describes a possibly complex aspect of the design solution: not necessarily a single attribute with a single scalar measure. This stage of analysis is about identifying which aspects of each aspect need to be considered individually. There is a complex web of different factors all interacting with one another. These interactions were further clarified in a dependency diagram. This diagram has two main goals:
  1. Identifying a small, and clearly-defined subset of specific aspects to consider in the detailed, and potentially quantitative, analysis.
  2. Highlighting dependency chains, especially unexpected ones, that may lead to patient harm.

<sup>6</sup> To give a prominent example, the U.K. National Cyber Security Centre and the U.S. NIST recently reversed their long-standing advice on password policies, acknowledging that policies previously considered "most secure" (complex passwords, to be changed frequently) caused users to invent workarounds that undermined the password-based authentication.

Figure 4-12 shows the dependency diagram summarizing the various factors in the decision to implement authentication and the relationships between them. It represents an incomplete yet complex account of how patient harm can be caused.

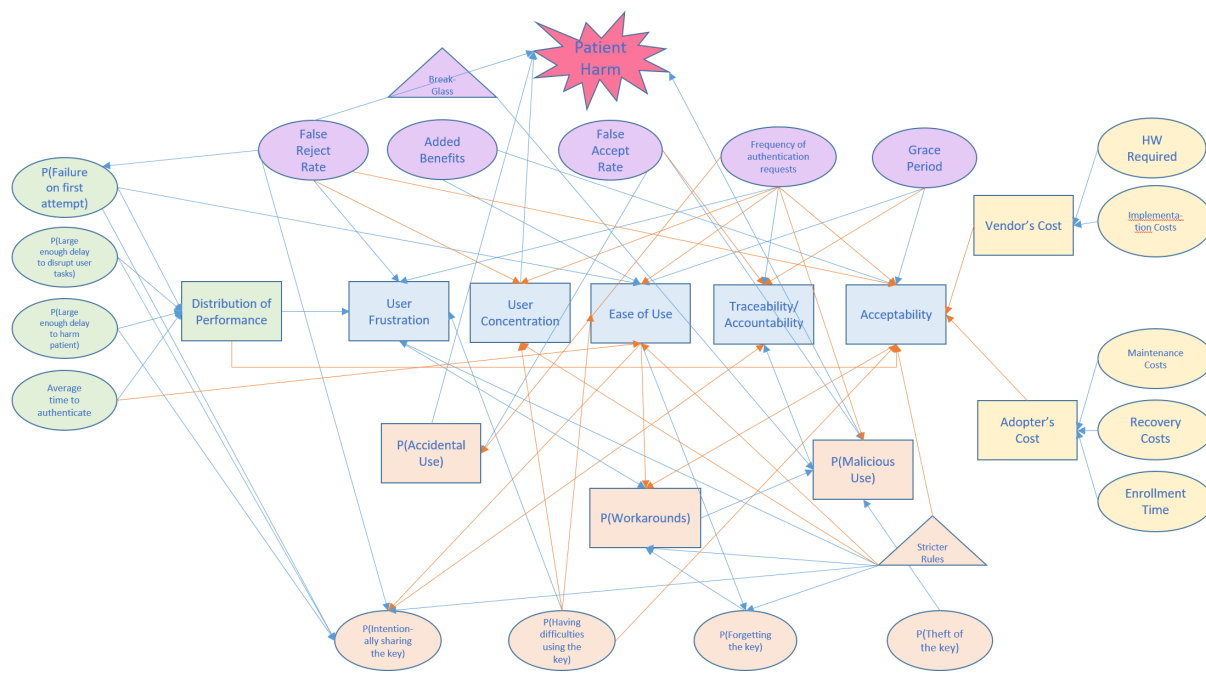


Figure 4-12: Relationship Between the Various Factors in the Decision to Implement Authentication

The colours of the shapes represent the category to which they refer: purple for characteristics of the authentication method (for example: false accept rate, grace period), peach for security-related factors (for example: probability of theft), yellow for costs (for the manufacturer and for the user organization), green for performance-related factors (for example: average time required to authenticate, probability that time to authenticate exceeds a certain amount), and blue for usability issues (for example: ease of use, user frustration). Ovals represent factors that are relatively easy to provide as inputs to the analysis, while rectangles represent factors that may not be as easy to estimate directly, and can instead be approximated based on the oval inputs. The arrows between the nodes indicate causal links. Blue arrows represent an “increase relationship” (an increase in the source node factor leads to an increase in the target node factor) while orange arrows represent a “decrease relationship” (an increase in the source node factor leads to a decrease in the target node factor). Triangles represent potential mitigation strategies: variations to the authentication method to reduce the strength of the specific link to which they are attached.

At the top of the diagram is “patient harm”, which describes all scenarios where there is an increased risk of harm to a patient. For example, in the case of the medical device considered in UC2, this node represents scenarios including but not limited to: the pump infuses the patient with an incorrect dosage, or the patient’s physiological parameters fall outside the target range but go unnoticed by the clinician.

We note in the diagram four input arrows to the “patient harm” node. These represent our four main conjectures about how patient harm may be triggered by authentication-related issues (four plausible examples out of other possible causes). More specifically:

- *Malicious Use*: an attacker intentionally gains access to the device and makes changes to patient care that put the patient at risk of harm. This may be by directly altering the delivery of drugs, or by tampering with the alarm functions of the device (e.g. producing excessive false alarms for patient conditions, or omitting alarms for exhaustion of drug supply in the pump).
- *Accidental Use*: a user unintentionally gains access to the device and unintentionally makes changes to patient care that put the patient at risk of harm.
- *False Reject*: the authentication method falsely rejects a legitimate user thus preventing them from assisting the patient when needed. For example, due to a card reader failure, or because the user's fingers are chapped and yield an error in a fingerprint biometric reading. We note that this can be the goal of an attacker.
- *User Concentration*: the authentication method negatively impacts the user's concentration to a level that distracts them or affects their ability to respond to an emergency, thus putting the patient at risk of harm.

Some of the important outcomes of this dependency diagram are:

- *Unexpected chains*: We note that although mitigation strategies are designed to reduce the risk of potential harm to a patient, they may also introduce new causal chains that increase this same risk. For example, to reduce theft of an authentication key, an organization may enforce stricter protocols. These protocols may reduce the risk of malicious use due to theft of the key (i.e., the link between nodes "Malicious use" and "P(Theft of the key)" is interrupted by the mitigation "Stricter rules"). However, at the same time we note a number of arrows leaving this mitigation strategy ("Stricter rules") and leading to increased "probability of forgetting the key", increased "probability of intentionally sharing the key", decreased "ease of use", etc. – all of which can then lead to patient harm through a different chain. As another example, to reduce the risk linked to illicit duplication of authentication cards, a hospital could require that cards be renewed monthly, but this would increase the risk of a clinician trying to use an expired card, hence risk to the patient.
- *Reduction of inputs required for a detailed/quantitative analysis*: Usability factors such as user frustration, user acceptability, ease of use etc. are relatively difficult measures to quantify (one can define measures for them, but quantifying the effects of their parent nodes on them and of them on their child nodes would be hard), but the diagram suggests that other factors, relatively easier to quantify (such as performance measures or estimated costs) might be suitable proxies – this simplifies and reduces the required inputs to a detailed/quantitative analysis.
- *Clarification of aspects*: The preliminary descriptive analysis presented in Table 4-2 contains broad, general categories, which require further clarification and definition in order to analyse at a deeper level. For example, what is called "usability" in Table 4-2 was divided in Figure 4-12 into: ease of use, user concentration, user frustration, acceptability, and traceability/accountability. These sub-categories are closely related (thus all arrows between them have been removed for simplicity), but still useful to clarify what aspects of usability are being considered.

We are also working on a deeper analysis from the security viewpoint. Here, rather than speaking only about security of the device in a general sense, security has been broken down into specific threats. For each threat, the likelihood is considered. A preliminary detailed analysis is sketched in Table 4-3, limited to the most promising methods identified via the preliminary descriptive analysis and minimum requirements described in Table 4-2. These are two token-based methods: RFID and magnetic stripe.

The estimated likelihoods in the table are a first-cut high-level evaluation. The likelihoods have three different levels: Possible (Red and Amber) and Not Possible (Brown). "Red" (or "Yes") means that there



is an obvious reason for thinking that the specific attack will be possible. On the other hand, "Brown (No) means that there are technical or other obvious reasons why this specific attack is not possible for the selected technology. Last, "Amber" ("May") stands for possibilities, where we closer investigation is required as simple high-level evaluation does not give a justified result "yes" or "no".

Table 4-3: Outline analysis of security viewpoint – likelihood of various threats

	RFID	Magnetic stripe card
<b>Social engineering methods</b>	<b>Possible</b> E.g. the attacker can borrow the card	<b>Possible</b> E.g. the attacker can borrow the card
	Types to be assessed: phishing, spear, pretexting, scareware, baiting	
<b>Malware</b>	<b>Possible</b> High skill attack, mostly necessary to combine with theft	<b>Not possible</b>
	Types to be assessed: adware, bot, bug, ransomware, rootkit, spyware, Trojan horse, virus, worm, keylogger	
<b>Guessing</b>	<b>Not possible</b>	<b>Possible</b> Necessary to combine with other theft
	Types to be assessed: brute force, dictionary, rainbow	
<b>MITM</b>	<b>Possible</b> Due to radio frequency, fully hidden, remote	<b>Possible</b> Necessary to install malicious HW in the reader
	Types to be assessed: replay attacks, eavesdropping attacks, reflection attack, OTP interception	
<b>Server-side attacks</b>	<b>Possible</b>	<b>Not possible</b>
<b>Shoulder surfing</b>	<b>Not possible</b>	<b>Not possible</b>
<b>Theft of Authenticator</b>	<b>Possible</b> due to use of physical token	<b>Possible</b> due to use of physical token

#### 4.3.4 Lessons Learned

The aim of this trade-off analysis is to help system designers (of a medical device, in this case) to decide on an appropriate authentication method for their device by considering the decision from all viewpoints. This goal has not yet been reached. However, the work done so far, besides demonstrating the complexity of the issue, highlights a number of lessons:

- The importance of clearly identifying the aspects that are part of the decision, at an early stage.
- The danger of introducing mitigation strategies "in isolation" without considering their effects from all viewpoints, as these effects may introduce new hazards.
- Hence, the importance of taking a holistic approach to the decision that considers all viewpoints.

#### 4.3.5 Further Developments

This trade-off analysis remains under development. We have documented the steps completed thus far. Ultimately, we would like to describe the authentication trade-off as an optimization problem. Certain design parameters can be tuned; for example, grace periods (how long a user can keep working despite failing authentication) or frequency of authentication requests are relevant inputs to this analysis and may reveal how certain values can sway the choice of authentication. To achieve this, more detailed analysis of the different viewpoints is underway, with considerations of how factors in the decision may be quantified.



It is certain that the overall risk will depend on the attack modes likely to be prevalent. Without any attackers, the only risk reduction gained from authentication is to prevent accidental inputs, while the risk is increased by the possibility of improperly denied access. Scenarios with different dominant attacks, e.g., attempts to input harmful command to specific patients vs attempts to harm patients at random via denial of service, will bring different optimal solutions. So, system designers will need assumptions (tentative predictions) on the threat environments; or, more likely, tunable authentication options, with their additional security and safety concerns.

#### 4.3.6 Further Details

The Annexes contain a copy of the complete worksheet documenting the descriptive analysis partially presented in Table 4-2.

### 4.4 Probabilistic Analysis of Performance-Security Trade-Offs via SANs (ATM Use Case Example)

Contributor: City

This section presents an example of a combined analysis of safety and security of a simplified version of the ATM demonstrator. The analysis relies on a probabilistic model, which is built using the formalism of “stochastic activity networks” (SAN). SAN is a generalisation of the Stochastic Petri Nets formalism. The model has been built and solved using the software tool Mobius, developed and maintained by the University of Illinois at Urbana-Champaign (<https://www.perform.illinois.edu/>).

The architecture of the demonstrator is described in detail in deliverable D2.3.1. We demonstrate the combined analysis on a part of the demonstrator, the middleware used for communication between the drones and the ground services. The middleware uses an implementation of the DDS specification (Data Distribution Service). The model includes: i) an implementation of the middleware, ii) models of the drones, i.e. Unmanned Aerial Vehicles (UAVs), which generate legitimate traffic around a number of topics, and iii) additional sources of traffic, e.g. different other applications that may share DDS with the ATM infrastructure.

The model also includes models of various malicious activities applied to the middleware and which may generate *malicious traffic*, e.g. by increasing the number of new topics, and/or associated data samples, with their publishers and subscribers, and alter the number of publishers/subscribers of legitimate topics. A key idea that we explore in the analysis is that the load on the middleware affects the response time in delivery of all messages, an effect which has been studied, e.g. [Bellavista] with different implementations of the DDS specification. In the model, the load generated by the publishers and subscribers is “recorded”, which in turn affects the response time to any message sent over the middleware.

The number of messages served by the middleware at any moment will vary. Some of the variations are intrinsic, e.g. due to overlaps in message delivery times, which is bounded for a fixed number of topics of well-known structure, defined in the middleware.

When the system is under attack, however, the number of topics can increase uncontrollably as malicious agents may create new topics. The size of the exchanged data (payload) is also unknown.

#### 4.4.1 Aim

The aim of the study is to explore the scale of the problem that can be caused by maliciously created traffic and also how these adverse effects can be limited by adding generic security controls such as periodically cleansing the middleware from malicious topics.

A measure of interest in the studies is the probability distribution of the message response times (PDMRT) for the legitimate messages under different regimes of operation of the middleware. This distribution will allow us to establish whether the message response time exceeds the hard real time constraint of 2 seconds defined as an essential requirement for the ATM systems and, if so, the likelihood of this for a particular regime of operation. Note that violation of the hard-real time constraint may have safety implications: the drone may become uncontrollable and eventually may be either lost or may collide with other drones or even planes. Thus, the analysis may also be seen as a special form of combined safety and security analysis.

The study also includes a comparison of the PDMRT of the “base line” case (when the system operates in a “trusted environment” without attacks) with the cases when the system is subjected to attacks of different intensities. Another aim of the study is to look at the effectiveness of additional “security controls”, e.g. applying an implementation of OMG DDS Security specification (<https://www.omg.org/spec/DDS-SECURITY/About-DDS-SECURITY/>), which apart from the obvious benefits in terms of data integrity, etc. introduces additional computational overheads and may lead to additional message delays.

#### 4.4.2 Method

A model-based method of analysis is used. We develop a SAN model of the system under study, which contains a number of parameters. Some of these parameters will define the size of the system, e.g. the number of legitimate topics, the (average) number of publishers and subscribers, the size of the payload, etc.

Other parameters model the conditions of operation, e.g. the intensity of the attacks, the effects that they can have (e.g. the number of additional topics that a successful attack can lead to and the parameters of each of these additional topics – publishers, subscribers and payload size). Finally, some of the parameters will characterise the “security controls”. For instance, if “cleansing” of middleware is used as a security control, additional model parameters are needed to capture: i) the frequency of cleansing, ii) the time needed to cleanse a replica of the middleware, etc.

The results for each set of model parameters are obtained by solving the respective model using the solvers offered by Mobius, e.g. the Monte-Carlo simulator. The results are presented in the form of probability distributions of the response time defined for several intervals of message delays: i) 0 – 0.5 sec, ii) 0.5 – 1 sec, iii) 1 – 1.5 sec, iv) 1.5 – 2 sec, and v) over 2 sec.

A simulation run will consist of running the system for 5000 sec (i.e. over an hour). During each run, a number of messages are sent/delivered. The duration of each message is established and counted towards one of the intervals defined above. At the end of the observations, each of the intervals will contain the number of messages which occurred during the run which happened to have response time within the respective interval. The probability of each interval for the particular run is then estimated as the ratio of the number of messages within the interval and the total number of messages that occurred during the run. Thus, PDMRT conditional on the particular run is established. The PDMRT established for different runs may differ. At the end of a study (with many runs of the model with the same values of the model parameters) we compute the *average* PDMRT, i.e. the probability of each

interval, with the respective confidence. It is this PDMRT “on average” that is used to compare the different operational regimes.

#### 4.4.2.1 The model

The model used in the studies is shown below in Figure 4-13.

This is a “composed” model, which includes a number of “atomic” models (shown in black).

In the Annex, we provide further details on the atomic models, the load model, which is the essence of the combined analysis, and a detailed description of the model parameters and the values assigned to them in different studies.

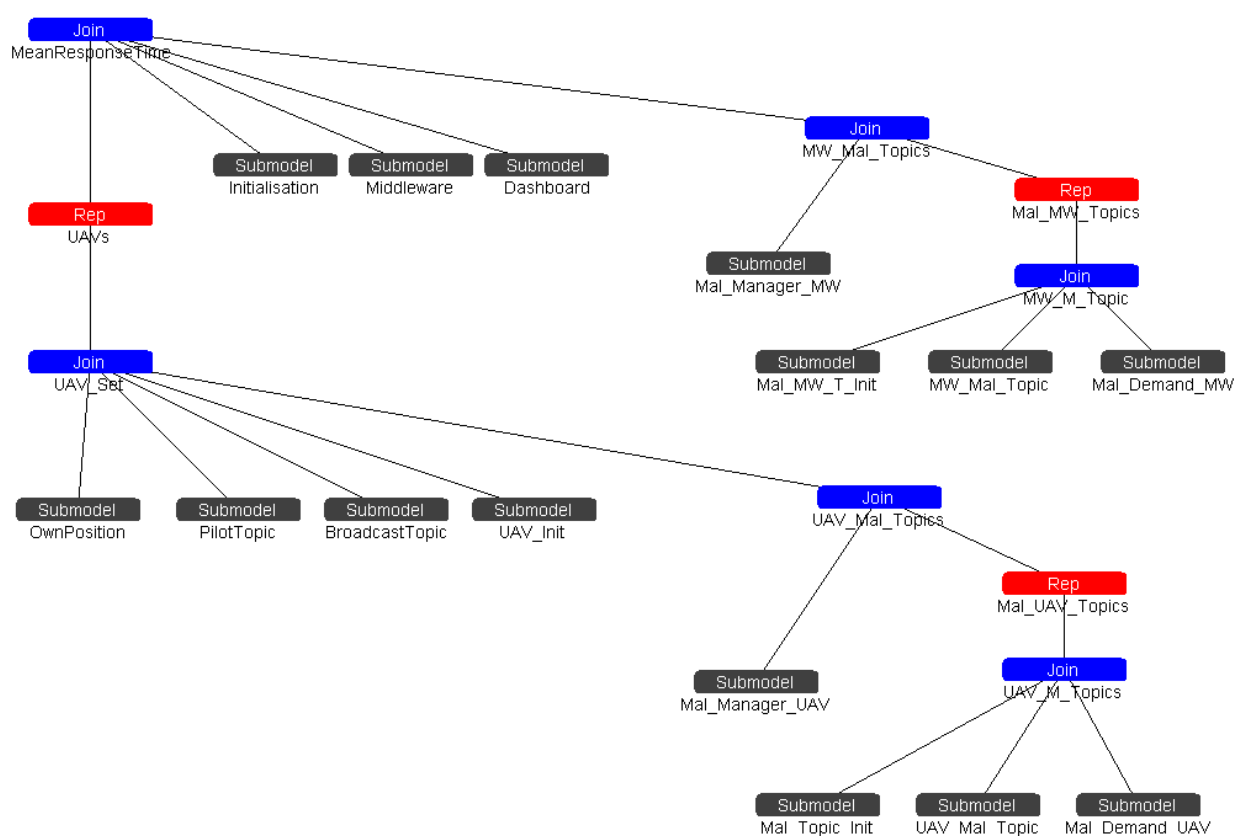


Figure 4-13: The SAN “composed” model of the ATM demonstrator.

#### 4.4.2.2 Model parameterisation

An essential part of the presented model is the “load model”, which captures the dependence of the response time on the load on the middleware. We demonstrate the method using a simple *regression load model*. A credible analysis will require a careful parameterisation. A number of *empirical studies* are currently under way at TrustPort and at City University to construct a *valid load model*. These initial studies are exploratory in nature, and will be made more specific and tailored to the ATM demonstrator so that the respective results are fully usable. The initial results from this work are given in the Annex.

### 4.4.3 Results

Some of the results obtained with the model are presented below in Table 4-4.

Table 4-4: Probability distribution of the message delivery times (PDMRT)

Experiment	Measure	Taken at:	[0-0.5] secs	[0.5-1.0] secs	[1.0-1.5] secs	[1.5-2.0] secs	[> 2] secs
Experiment 1	Mean	1000	0.0E+00	0.376	0.469	0.784	0.216
	Conf. interval		0.0E+00	0.0403	0.0398	0.0242	0.0242
	Mean	2000	0.0E+00	0.386	0.592	0.788	0.212
	Conf. interval		0.0E+00	0.0264	0.0259	0.0174	0.0174
	Mean	3000	0.0E+00	0.404	0.6396	0.790	0.210
	Conf. interval		0.0E+00	0.0229	0.0200	0.0125	0.0125
	Mean	4000	0.0E+00	0.413	0.6607	0.7901	0.209
	Conf. interval		0.0E+00	0.0208	0.0162	0.0109	0.0109
	Mean	5000	0.0E+00	0.405	0.613	0.792	0.208
	Conf. interval		0.0E+00	0.0173	0.0139	0.0094	0.0094

Each experiment for a given parameterisation was repeated 50 times ("Monte Carlo simulation runs"). Each run lasted 5000 seconds of system operation. For each of the experiments we estimated the "average" message response time and the PDMRT at predefined points of simulation time: 1000, 2000, 3000, 4000 and 5000 seconds.

Table 4-4 presents the PDMRT in the form of a *cumulative distribution* function for one of the experiments. The mean message response time and a fuller description of the results illustrated in the table can be found in the Annex.

The results are grouped by the averages of the cumulative probability for the selected time intervals [0...0.5], [0.5...1.0], [1.0...1.5], [1.5...2.0]. The tail of the distribution, i.e. the probability that the delay is longer than 2 seconds, is shown under the heading "[> 2] secs". For this experiment, this value indicates that the probability of a message delay exceeding the acceptable delay (of 2 seconds) is more than 20%. A figure of this magnitude is unlikely to be acceptable.

### 4.4.4 Lessons Learned

The lessons learned from this method of analysis can be divided into two groups:

- Building the model event for a relatively small group of attacks required non-trivial effort, which suggests that an effort to automate the analysis models will bring about significant benefits. This lesson reinforces the importance of the on-going collaboration between City and Intecs to derive a SAN model from a system model developed in CHESS. Even a partial success here will drastically reduce the effort required to construt a SAN model.
- The Monte-Carlo execution time is significant. The decision to show results from only 50 repetitions of each experiment was dictated in part by the long simulation time needed to complete a simulation run. In another similar study that we have conducted (related to the Industrial Drive use case), a simulation run is much faster (see Section 4.5 for further details). The

difference is due to the chosen level of abstraction. Performance (response/delay times related to individual messages) requires models in which individual messages sent via the middleware are explicitly represented. Modelling even a short system operation implies a very large number of messages being sent. Thus, it is the nature of the problem which leads to long simulation time.

#### 4.4.5 Further Developments

The model presented in this chapter focuses on attacks on middleware. Although the model includes placeholders for attacks on the individual UAV, these are yet to be developed and added to the model. We envisage attacks on the communication channel(s) between the UAVs and the ground stations, which may include common threats such as DoS, etc. These aspects of the models will be developed in our future work.

Of particular interest is to study the impact of adding “DDS security” to the middleware. DDS Security is a specification, complementing the DDS specification, and implementations of the DDS Security specification are provided by several vendors. The model will allow us to examine the trade-offs involved in adding DDS Security: on the one hand DDS security may make some attacks of middleware more difficult; on the other hand adding DDS security is likely to introduce further message delays, which may increase the likelihood of violating the threshold for message delays of 2 sec. The current model already contains the essential parts which will allow us to study the trade-off – the probability of successful attack (attackSuccessRate) and the additional delays of a message (Start\_Delay) – due to encryption. The missing parts which will make the comparison possible are the load models for the two cases: middleware without DDS Security and middleware with DDS Security. The empirical work summarised above on model parameterisation will allow us to construct credible load models and undertake the outlined trade-off analysis.

A related strand of work has been initiated at BUT after a technical meeting between City, TrustPort and BUT to look at vulnerabilities in several popular implementations of DDS. The focus of this work is to establish vulnerabilities in different implementations of the DDS specification, which allow a malicious agent to launch attacks on specific DDS implementations. We consider known vulnerabilities, e.g. published in the public vulnerability databases such as NVD - <https://nvd.nist.gov/>, CVE - <https://cve.mitre.org/>, etc. but which may not be patched in the available implementations of the DDS specification, or unknown vulnerabilities, e.g. established via fuzzing or using some other methods of searching for vulnerabilities in a chosen implementation.

This work on vulnerabilities of DDS implementations is ongoing and we expect some tangible results in the future, which will help build a useful model of the entire demonstrator.

#### 4.4.6 Further Details

A fuller version of the study is presented in the Annexes. In addition, we provide a complete documentation of the presented model which includes a full account of the completed studies with the probability distributions related to the different modes of operation.

### 4.5 Analysis of Safety-Security-Performance Trade-Offs via SANs (Industrial Drive Use Case Example)

Contributor: City

This section presents an example of a combined analysis of safety and security of a simplified version of the Industrial Drive demonstrator. The analysis, similarly to Section 4.4, relies on the “stochastic activity networks” (SAN) formalism.

The Industrial Drive is a demonstrator with distributed architecture – a remote “client” application is used to send commands to, and retrieve the status of an industrial motor from, a “server” application. The actual control is achieved by the server application and dedicated hardware. A detailed description of the architecture of the demonstrator is presented in AQUAS deliverable D2.3.4.

The part of the demonstrator used in the combined analysis includes a model of i) the client application, ii) the server application, and iii) a “safety function” to guarantee that should the control motor deviate from its safe operation, the safety function will bring the entire system to a safe state.

The model of the server application is a model of several sub-systems defined in the prototype – the server (which passes the data received from the client via shared memory for further processing by the dedicated parts of control), the control (computing the values to be passed to the dedicated hardware board), and the communication of the board with the motor.

In the model the client and the server applications are assumed implemented *without redundancy*, which is typically used to improve reliability and availability. For the safety function, however, we assume that two channels are used which implement a 1-out-of-2 architecture. That is, each of the channels on its own is sufficient to bring the system to a safe state should either the client or the server applications, or both, fail.

In the model we assume that each of the two applications (client or server) either work correctly or may fail. The model concentrates on *software failures*. The failures due to hardware faults are implicitly excluded from the analysis. We assume that the safety functions may fail in two different ways:

- Failing to detect a failure of the client/server application, provided these have failed (i.e. send incorrect control value to the motor) – false negative.
- Spurious false alarm – i.e. flagging the operation of the system as incorrect while in fact both the server and the client are working correctly.

Probabilistic parameters are used to characterise the behaviour of the applications such as failure rate and repair rate to characterise the transitions between failure and repair of the client and the server applications.

Similarly, we use probabilistic parameters to characterise the behaviour of the two channels of the safety function. These parameters are:

- Probability to detect a failure of the system when such failure occurs (“coverage”).
- Rate of spurious alarm when the system works correctly. The model of spurious (false) attacks is somewhat more complicated and in fact includes two parameters – rate of occurrence of a false alarm and a probability of detecting the false alarm when it occurs. In other words, the model assumes some form of *self-checking capability* for the false alarms.

Since two channels are used for the safety function, it is important to include in the model the possibility of *simultaneous*, i.e. *common cause/mode failures* of the two channels. We model explicitly the occurrence of simultaneous failures of the two safety channels to detect a failure elsewhere in the system and of a false alarm, which rely on a few probabilistic parameters.

The 2-channel safety function can be in one of the following 4 states:

- The system is *OK*. This is when the client, the server and both safety channels work correctly.

- The system is in a *safe failure* state when there is a failure of either the client or the server, but at least one of the safety channels detects the failure correctly. We assume that in this case the safety function will carry out the necessary steps so that the industrial drive is brought to a *safe system state* (e.g. stop the motor)
- A *false alarm* occurs when the client and the server application are OK, but at least one of the safety channels flags the state as a failure. In this case the system will be driven to a safe state, but this will be unnecessary. False alarms compromise availability of the industrial drive. A high rate of false alarms, although not dangerous, is clearly undesirable.
- *Unsafe failure* occurs when either the client or the server applications (or both) have failed and both safety channels fail to detect the failure. This is a dangerous situation and reducing its likelihood to an acceptably low level is the primary safety concern.

In the model we consider *two types of attacks*, which are recognised as important in D2.3.4, namely:

- an attack on the client application. Should an attack on the client application succeed, the client application is considered to be in a “compromised” state. The failure rate from the “compromised” state to the failed state of the client is assumed to be typically greater than the rate of failure of the client from the OK state. The rationale for such a model has been extensively discussed elsewhere, e.g. [Popov, 2017].
- an attack on the safety function(s). The successful attacks of this kind result in *changing the “coverage”* of a safety channel, i.e. the probability to detect a failure, provided there is a failure somewhere in the systems (client, server or both), or *changing the rate of occurrence of false alarms* by the respective channel. A similar model of consequences of an attack was developed elsewhere, e.g. [Popov, 2015].

#### 4.5.1 Aim

The aim of the study is to demonstrate a method of modelling, which allows one to analyse the safety of the industrial drive under attacks.

The model includes a model of “prevention”, i.e. models the measures taken by the developers of the industrial drive demonstrator to *reduce the likelihood of successful attacks* and the effects of generic security controls such as “proactive recovery” [Sousa], which minimise the harm caused to the safety of the system should some of the attacks succeed.

The model is based on the assumption that the modelled system works on a “mission” of a given duration, say 1000 hours of system operation. We simulate a large number of missions, say up to 100,000, and for each of the missions we record the mission outcome as follows:

- The entire mission is completed successfully (i.e. the system remains in an OK state throughout the mission) and no anomaly occurs;
- A detected failure occurs before the end of the mission. In this case the mission is aborted immediately, i.e. before the 1000 hours of the particular simulation have elapsed;
- A false alarm is raised before the end of the mission and the mission is aborted;
- An unsafe failure occurs before the end of the mission and the mission is aborted.

Our aim with the analysis is to establish the probabilities of the 4 outcomes listed above for a *randomly chosen mission*. The main concern in the analysis is, of course, the *probability of unsafe failure*, but the probabilities of safe failure and especially the probability of false alarms are also of interest as these lead to reduced availability of the industrial drive.



We look at a range of models to establish the impact of attacks on the probabilities of interest listed above. These models share the same structure (i.e. consist of the same modelling elements and relationships), but are *parameterised differently* to capture different circumstances, for which we would like to analyse system behaviour. Some of the parameters used in the model reflect the presence or absence of security controls. Some other parameters reflect the intensity of the attacks, the likelihood that they will succeed, etc. A full account of the parameters and their purpose are provided below.

The model includes security controls, such as “cleansing” [Arsenault] of software components which might have been compromised by an attack. The frequency of such cleansing is a design choice that the designers should make. Our model offers help with such decisions. It allows one “to see” how efficiency of cleansing as a countermeasure against successful attacks changes when the cleansing frequency is increased - this allows for decisions to be taken rationally.

## 4.5.2 Method

The method used in the study is based on comparing the results from models with different values of the modelling parameters. Each parametrisation corresponds to a credible system design. Solving each of the models provides insight about the likely behaviour of the system with different designs. The models are “solved” via Monte-Carlo simulation. The results are the estimates of the 4 probabilities, defined above, for a mission of fixed length. These probabilities are different for the different model parameterisations. The model results would inform design decisions about which design should be adopted.

Somewhat arbitrarily we chose a mission time of 1000 hours (~ 40 days), and provide estimates for the probabilities of interest at intervals of 100 hours: 100, 200, etc., which allow one to see how the probabilities of interest evolve with the mission length.

### 4.5.2.1 The model

First, we describe the model and provide a brief description of the different modelling decisions. The structure of the model is shown in Figure 4-14.

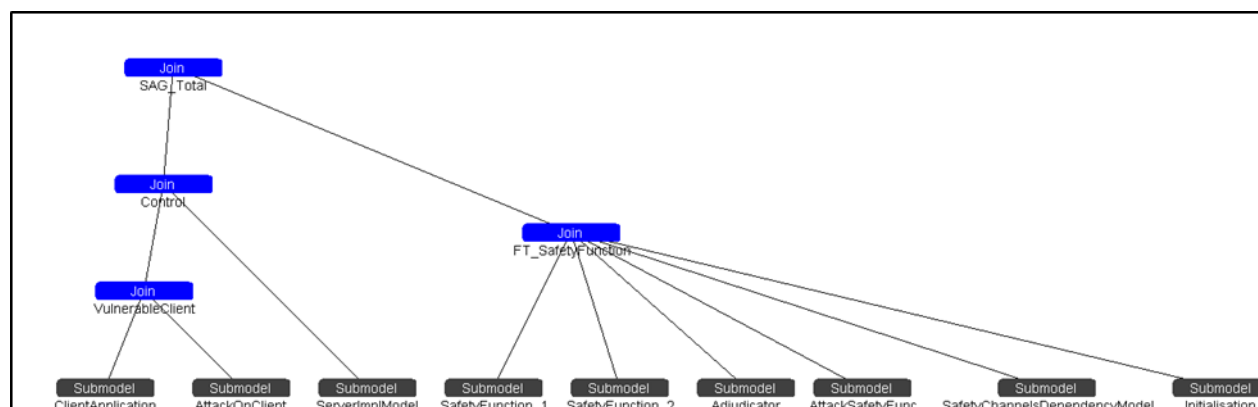


Figure 4-14: The SAN “composed” model of the “industrial drive” demonstrator.

In the Annexes we provide further details on the atomic models, how attacks affect system behaviour, and a detailed description of the model parameters and the values assigned to them in different studies.



### 4.5.3 Results

An illustration of the benefits from the method is presented below. Fuller description of the conducted studies and of the observations are presented in the Annexes.

#### 4.5.3.1 Sensitivity analysis

Let us look at how modelling parameters affect the model behaviour. For instance, let us consider the case of attacks on the client application only (assume that the attacks on the safety functions are “impossible”). We would like to find out how the *frequency of cleansing* the client application (e.g. rebooting it from a clean, uncompromised copy) will affect the model behaviour. Table 4-5 shows the results from several studies in which the period of cleansing is varied: 1 hour, 10 hours, 100 hours and 1000 hours.

Table 4-5: Sensitivity analysis results of the effect of cleansing interval on model behaviour with client only attacks.

Measure	Time [hours]	Estimate	Experiment 6 (Cleansing period = 1 hour)	Experiment 7 (Cleansing period = 10 hours)	Experiment 8 (Cleansing period = 100 hours)	Experiment 9 (Cleansing period = 1000 hours)
Probability of Successful Missid (OK)	100	Mean value	7.00E-01	4.87E-01	1.65E-01	1.65E-01
		Conf. interval	5.10E-03	9.34E-03	2.30E-03	2.30E-03
	500	Mean value	1.94E-01	1.27E-01	1.31E-03	1.00E-05
		Conf. interval	4.40E-03	6.23E-03	2.24E-04	1.96E-05
	900	Mean value	5.34E-02	3.60E-02	3.50E-04	0.00E+00
		Conf. interval	2.50E-03	3.48E-03	1.16E-04	0.00E+00
Probability of False Alarm (FA)	100	Mean value	1.65E-01	1.38E-01	1.06E-01	1.05E-01
		Conf. interval	4.13E-03	6.45E-03	1.91E-03	1.90E-03
	500	Mean value	4.85E-01	3.46E-01	1.22E-01	1.20E-01
		Conf. interval	5.56E-03	8.89E-03	2.03E-03	2.01E-03
	900	Mean value	5.74E-01	4.06E-01	1.22E-01	1.20E-01
		Conf. interval	5.50E-03	9.18E-03	2.03E-03	2.01E-03
Probability of System Safe Failure (SF)	100	Mean value	1.22E-01	3.40E-01	6.57E-01	6.57E-01
		Conf. interval	3.65E-03	8.85E-03	2.94E-03	2.94E-03
	500	Mean value	2.90E-01	4.80E-01	7.90E-01	7.93E-01
		Conf. interval	5.05E-03	9.34E-03	2.53E-03	2.51E-03
	900	Mean value	3.37E-01	5.08E-01	7.90E-01	7.93E-01
		Conf. interval	5.26E-03	9.34E-03	2.52E-03	2.51E-03
Probability of System Unsafe Failure (USF)	100	Mean value	1.23E-02	3.48E-02	7.23E-02	7.24E-02
		Conf. interval	1.23E-03	3.43E-03	1.61E-03	1.61E-03
	500	Mean value	3.04E-02	4.72E-02	8.72E-02	8.76E-02
		Conf. interval	1.91E-03	3.96E-03	1.75E-03	1.75E-03
	900	Mean value	3.55E-02	5.01E-02	8.72E-02	8.76E-02
		Conf. interval	2.06E-03	4.08E-03	1.75E-03	1.75E-03

Increasing the intervals of cleansing has detrimental effects on model behaviour: the probability of surviving a mission decreases while the probabilities of all anomalies go up. While this is not surprising, the insight that the model brings is the magnitude of the effect, which may be useful in making a decision about what interval of cleansing to deploy in the demonstrator.

#### 4.5.3.2 A discussion

The Results presented in the section above (and detailed further in the Annexes) indicate that the effects of the 2 types of attacks may have noticeable consequences. The effect of a successful attack on a client application is not surprising. The attacks on the safety function offer some interesting observations. If the integrity of the safety function is compromised, the negative effects may be very dramatic. The increased probability of unsafe failure may easily invalidate any safety case claimed for “trusted environment”. Even unlikely attacks may increase the probability of unsafe failure significantly.

In the Annexes, we present results related to false alarms, caused by attacks on the safety function channels. These are particularly unpleasant. Even if one can design *very good safety functions* and practically eliminate the false alarms in trusted environment, the effort may become futile due to carefully crafted attacks on the safety functions. It seems that in these circumstances “cleansing” is particularly desirable as a control to counter the effects of cyber-attacks.

#### 4.5.4 Lessons Learned

The lessons from applying this method of combined analysis are yet to be learned via detailed scrutiny of the findings included in this study. By design this method operates at a high level of abstraction and many implementation details have so far been ignored or resolved in a way that allows for solving the model *fast*. For instance, instead of modelling the individual commands exchanged between the client and the server applications, which would make a solution via Monte Carlo simulation time consuming, the model concentrates on the essential anomalous events that may occur in operation – failures, successful attacks and resuming normal operation. With the chosen measures of interest (the four probabilities for a randomly chosen mission), the model behaviour after an anomalous event occurs (system failure or false alarm) is discarded. Should a different measure of interest be chosen, however, e.g. the interval between unsafe failures (within a mission) or something else, abandoning the mission after the first anomalous event may need to be revisited, which in turn may lead to much longer simulation time.

#### 4.5.5 Further Developments

The model currently only covers a fraction of the use case – the client application, the server application and the safety functions. These are modelled at a relatively *high level of abstraction* ignoring i) the specifics of the design choices made by the developers of the demonstrator, and ii) some important implementation details of the security controls included in the model. For instance, “cleansing” is modelled simplistically assuming that it can be executed “instantaneously” in a single atomic operation which does not affect availability of functional blocks that are subjected to cleansing. This assumption seems quite plausible for cleansing the safety functions, especially if cleansing can be achieved by overwriting the values of a few variables held in an SD card. Cleansing the client application, however, is likely to require a significantly longer time, e.g. to reinstall/restore it on the remote computer. During such “cleansing” the client application will simply be non-operational and must be designed to allow for cleansing, e.g. during the periods of maintenance. If high availability is required (especially in the case of a manufacturing line expected to operate 24x7), the design of the client application must include redundancy, too: the cleansing will then follow the suggestions made by many to use “intrusion-tolerant” designs. In such cases the models of client application will have to be extended and will follow our previous work [Popov, 2017].

The current model includes some rudimentary elements of performance penalty due to successful attacks, but this is an area for significant improvement in the future so that the analysis includes

assessment of the probability of violating the duration of the control loop of 62.5  $\mu$ s, either due to accidental or malicious events.

Finally, the model does not include *all attacks* envisaged in the demonstrator (see AQUAS deliverable D2.3.4), including the work previously done in the SESAMO project by the City team on triplicated communications between the sensors of the motor and the controller.

In the next period of the project the model will be extended to include the essential functional blocks and all attack types envisaged in the demonstrator.

#### 4.5.6 Further Details on the Model

Full details of the SAN model, the conducted studies and of the results obtained with the model to date are included in the Annexesto this section.

### 4.6 Combined Analysis of Safety and Performance to Support Design Space Exploration and Technical Solutions Comparison (Space Use Case Example)

Contributor: Univaq

The proposed method combines performance, safety and (possible) security analysis at a system-level of abstraction, considering schedulability and possible isolation alternatives. The utility of the approach is explored by analysis of safety/performance trade-offs in the space multi-core use case (UC5). It is also proposed to reuse the approach in the ATM use case (UC1).

The method involves the use of the Hepsycode methodology and framework [Hepsycode] with respect to performance analysis, while taking into account safety (fault identification and injection to evaluate possible critical paths) and security (impact of cryptography algorithms on performance). The whole framework drives the designer from an Electronic System-Level (ESL) behavioral model, with related non-functional requirements, including real-time and mixed-criticality ones, to the final HW/SW implementation, considering specific HW technologies, scheduling policies and Inter-Process Communication (IPC) mechanisms. Through the execution of different steps, including a system-level Design Space Exploration (DSE) approach that allows the related co-design methodology to suggest an HW/SW partitioning of the application specification and a mapping of the partitioned entities onto an automatically defined heterogeneous multi-processor architecture, it is possible to proceed with system implementation.

UC5 involves aerospace application code running on a LEON3 multi-core processor system. Timing and code safety analysis are essential to prove that the code can be safely scheduled and run under all operating circumstances.

#### 4.6.1 Aim

The main goal of this work is to present a combined analysis of performance/safety/security to support design space exploration and technology solutions comparison. The main idea is the modelling of a space multi-core application as a processes network where processes can communicate with each other by means of unidirectional and blocking channels. Such a Model of Computation (MoC) guarantees determinism and synchronous communications, allowing a deterministic data flow from the input triggers to the final output feedbacks.

The main problem is related to the analysis of performance/safety/security trade-offs on a multi-processor system (i.e., an unpredictable disrupted system where interferences are due to HW/SW components) considering schedulability and isolation methods.

The proposed solution is based on a simulation approach to evaluate safety/performance impact considering various HW/SW architectures. A semi-automatic Design Space Exploration (DSE) step involves several stages, from the definition of the solution space, the encoding with respect to the decision variable space, and the definition of the objective functions to solve a Multi-Objective Optimization Problem.

Starting from several system-level models (i.e., Application Model, Partition Model and Platform Model), the DSE exploits a search method that performs the “HW/SW Partitioning, Architecture definition and Mapping” (PAM) step, by using a genetic algorithm that allows one to explore the design space looking for feasible mapping/architecture items suitable to satisfy imposed constraints [Muttillio]. Then, a “Timing HW/SW Co-Simulator” [Ciabrone] considers the suggested architecture/mapping items to actually check for timing constraints satisfaction. The whole methodology is shown in Figure 4-15.

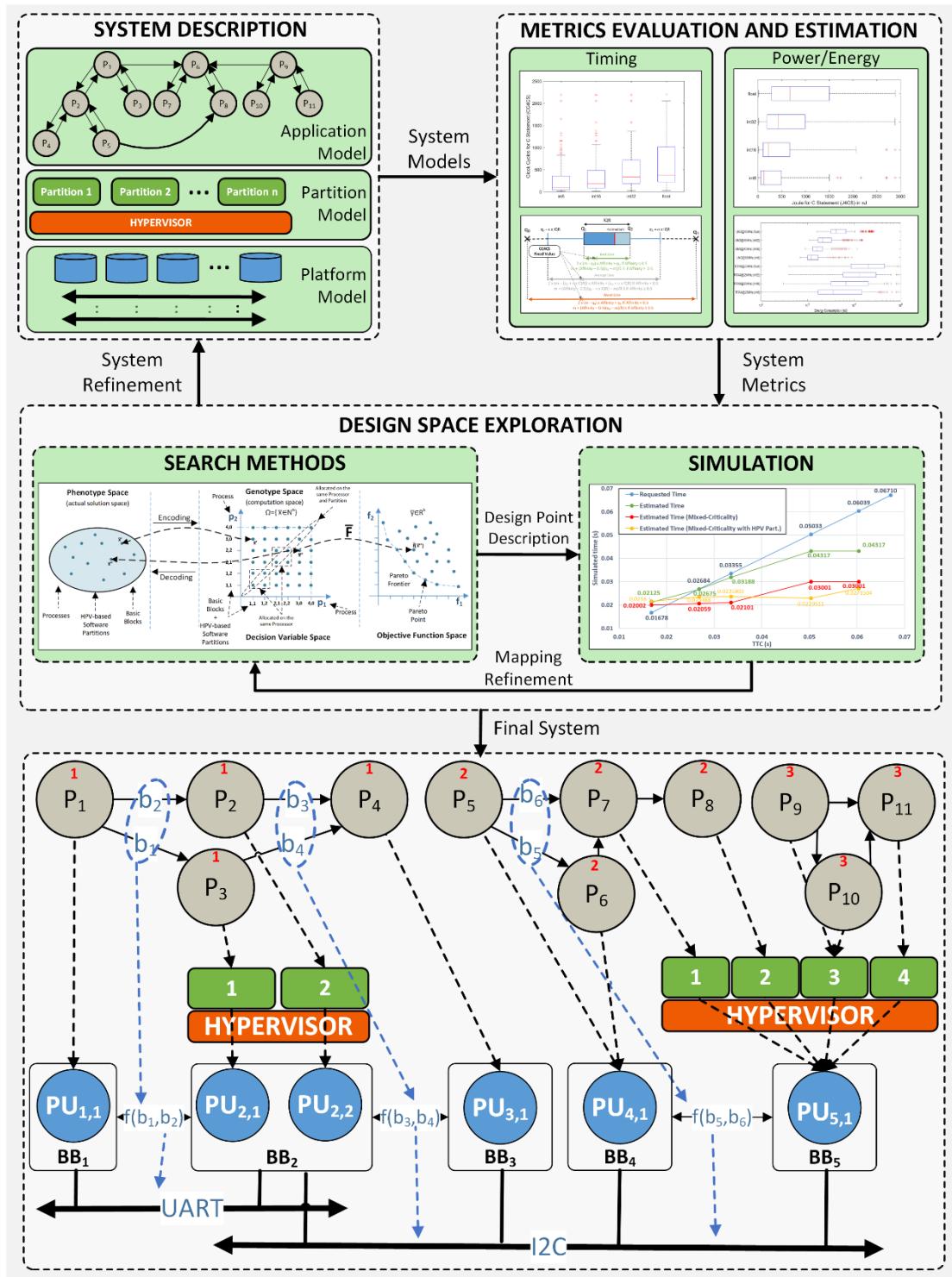


Figure 4-15: Hepsycode Methodology

#### 4.6.2 Method

The Hepsycode framework and methodology [Hepsycode] drive the designer from an Electronic System-Level (ESL) behavioral model, with related non-functional requirements, to the final HW/SW

implementation. In particular, the system behavior modeling language introduced in Hepsycode, named HML (HEPSY Modelling Language), is based on the well-known Communicating Sequential Processes (CSP) Model of Computation (MoC). Such a MoC allows modelling of the behavior of the system as a network of processes communicating through unidirectional synchronous channels. By means of HML, it is possible to specify the System Behavior Model (SBM), an executable model of the system behaviour, a set of Non Functional Constraints (NFC) and a set of Reference Inputs (RI) to be used for simulation-based activities. Through the execution of several steps, including metrics evaluation and estimation activities and a system-level Design Space Exploration (DSE) approach that allows the related co-design methodology to suggest a HW/SW partitioning of the application specification and a mapping of the partitioned entities onto an automatically defined heterogeneous multi-processor architecture, it is possible to proceed with system implementation. Hepsycode uses Eclipse MDE technologies, an extension of the standard SystemC simulator and an evolutionary genetic algorithm for partitioning/mapping activities, all integrated into a (semi)automatic framework that drives the designer from a system-level specification to a final solution, considering also Hypervisor-based SW Partitions in the evolutionary approach and timing HW/SW co-simulation runs.

So, the HEPHYCODE starting point considers different HW-based, OS-based, and Hypervisor-based solutions (both in the research and industrial domains), and uses specific modelling technologies, metrics evaluation and estimation activities, and a specific HW/SW co-simulator integrated into the Hepsycode Co-Design methodology and framework. Then, it is possible to find suitable sub-optimal solutions for the HW/SW partitioning problem by suggesting both the platform and the mapping for the specific mixed-criticality and real-time application, exploiting hypervisor-based SW partitions, also performing schedulability analysis and final validation activities to guarantee bounded errors.

#### 4.6.3 Results

Various activities have taken place during the past months:

- UC5 modelling activities: starting from a CHES model, UNIVAQ has adapted the application scenario to the HML. The application is composed of 7 processes (as shown in Figure 4-16), each of which matches with a CHES component.

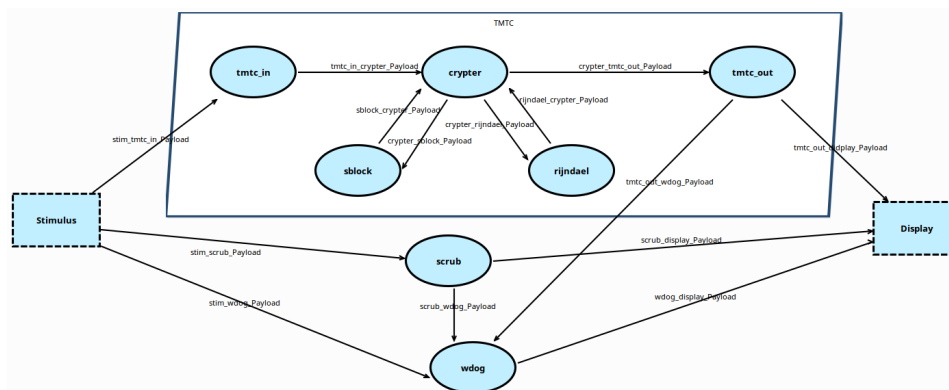


Figure 4-16: HEPHYCODE Process Network Model

- The second step has involved the extraction of several metrics by means of simulation activities. The considered metrics are: concurrency (i.e., an indication of the extent to which the set of process and channel pairs can potentially work concurrently), communication (i.e., the amount of data exchanged between process pairs), workload (i.e., processor utilization) and real-time behaviour (i.e., estimated WCET). In the context of UC5, some preliminary

results show that the channel pairs [stim\_scrub\_channel, wdog\_display\_channel] and [stim\_wdog\_channel, tmtc\_out\_display\_channel] are always concurrent, so the process pair [MEMORY SCRUB, WDOG] and [WDOG, tmtc\_out channels] shouldn't share the same link.

With respect to processes:

- SCRUB is 33% concurrent with WDOG, TMTC\_IN and TMTC\_OUT
- WDOG is 66% concurrent with TMTC\_IN and TMTC\_OUT
- TMTC\_IN and TMTC\_OUT are 100% concurrent (it is then possible to consider the introduction of a PIPELINE)
- The possible combined Safety/Security/Performance analyses are related to two different scenarios, as shown in Figure 4-17. The first one considers two safety-related subsystems, the non-critical subsystem, where the Telemetry/Telecommand application sends satellite information to the earth station, and the critical one, related to system monitoring and fault detection/avoidance.

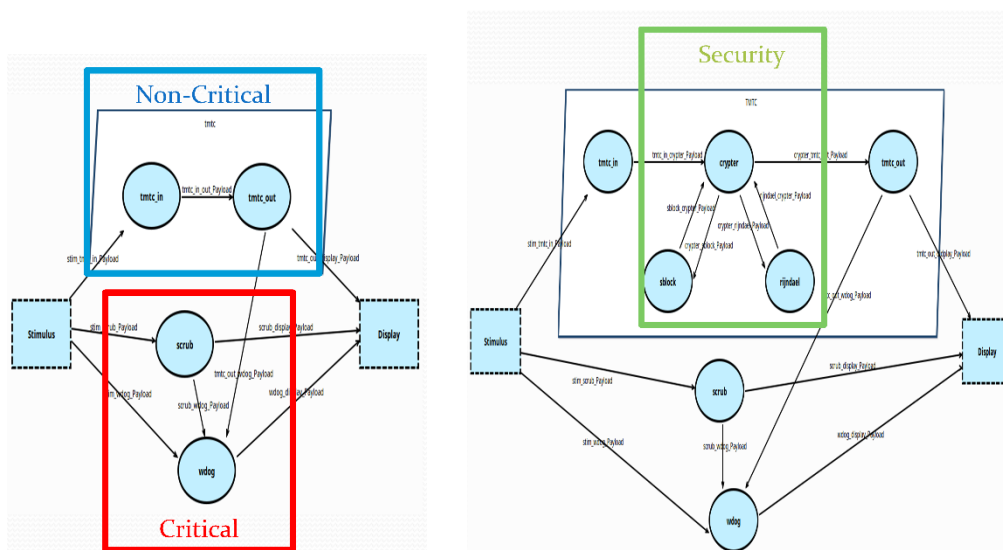


Figure 4-17: Hepsycode AQUAS Version 1 (No Security) - LEFT, Hepsycode AQUAS Version 1 (with Security) - RIGHT

- Starting from these activities, different failure paths can be investigated in order to check safety/performance anomalies, as shown in Figure 4-18. Finally, introducing security issues can degrade performance and, at the same time, negatively affect system execution, thus affecting the overall system safety.



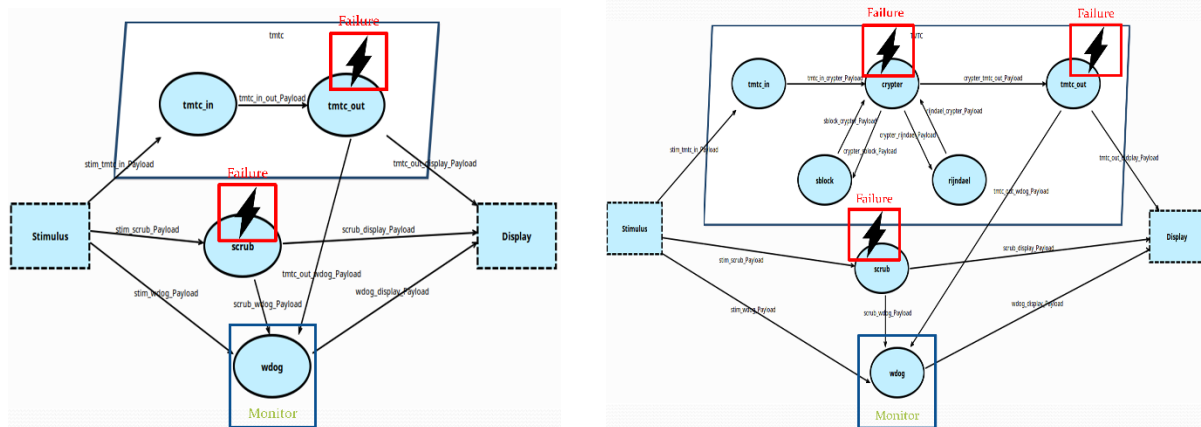


Figure 4-18: Hepsycode Performance/Safety Analysis - LEFT, Hepsycode Performance/Safety/Security Analysis - RIGHT

- The DSE and simulation activities are work-in-progress steps, while the different paths and failure scenarios will be evaluated with focus on possible architectural and behavioural improvements.

#### 4.6.4 Lessons Learned

We have released a framework and related tools that are able to model several applications injecting safety/security/performance requirements into the whole design step, with focus on simulation activities. The tools are able to extract information related to functional and timing issues, while the input model can be used to check possible erroneous/critical behaviours (e.g., deadlock, starvation, functional bottlenecks, etc.).

The Design Space Exploration tries to find possible allocation/binding alternatives, but this is a critical issue while the strict safety/security requirements can affect performance, decreasing response and execution times. The use of other external tools (i.e. A2K ITI for schedulability analysis, CODEO/PikeOS for SW partitioning improvements, CHESS for a Model-to-Model semi-automatic comparison, SystemC-TLM for timing simulation) can offer the possibility to decrease design time while improving system implementation reliability. Verification and validation activities on a real board environment are then needed for different aspect, from methodology refinement to system components improvement, while the Design Space Exploration can help designers to guarantee the fulfilment of input requirements. The use of Hypervisor technologies (e.g., PikeOS) will guarantee a behaviour compliant to certification according to relevant standards, but the qualification of such technologies is not easy to obtain.

#### 4.6.5 Further Developments

The Hepsycode approach will be extended to the ATM use case (UC1) and collaboration with other partners.

#### 4.6.6 Further Details

The complete Hepsycode methodology and approach is provided at <http://www.hepsycode.com/>.



## 4.7 Combined Analysis of Security and Performance to Support the Product Lifecycle using SSDLC and TTool (Industrial Drive Use Case Example)

Contributors: Trustport and MTTP

Safety, security and performance are mostly interdependent in the product life cycle management and therefore the task is to find a reasonable balance between them. Combined Analysis is based on deep analysis to define the security parameters (e.g. encryption, integrity, confidentiality) and security level (e.g. key lengths) based on the most current norms, standards, risk analysis, and best practice together with the modelling and verification of performance analysis in the TTool toolkit. We demonstrate the approach via application to the industrial drive demonstrator (UC4) to assist co-engineering within the demonstrator implementation.

UC4 concerns an Industrial Drive for electric motion control. Within AQUAS, a virtual HW prototype of the whole system will be created that shall be used to verify performance constraints together with safety and security requirements for a representative set of scenarios.

With the Software Development Life Cycle Management Tool (SSDLC) the product security requirements (based on standard IEC 62443 and best practices) were implemented. This approach discovered interconnections between the security requirements (security) and their impact on the final system response (performance) during the development stage.

Combined Analysis of Security and Performance to Support the Product Lifecycle using SSDLC and TTool was used to cover the whole product development cycle from security and performance requirements point of view. The implementation of requirements in SSDLC and TTool also follows the V-model starting from set of requirements continued through design of implementation and verification phase.

### 4.7.1 Aim

The goal of this combined analysis is to create a way to establish a compromise between security and performance requirements.

An interaction between security and performance was investigated for potential trade-off decisions (which security mechanisms such as encryption can be used under consideration of performance resources) by the Combined Analysis and using SSDLC and TTool.

### 4.7.2 Method

Figure 4-19 helps introduce a methodology of combined analysis and an approach to the implementation of interaction points in practice. The standard ISA/IEC-62443-3-3 and IEC62443-4-2 together with NIST 800-82, IEC 27001 and COBIT were implemented into SSDLC.

We are using an extensive Secure Software Development Life Cycle catalogue containing security requirements together with the advanced modelling framework TTool based on UML/SysML-Sec for performance analysis.

For performance analysis, the complexity curve of certain algorithms was calculated. The simple model (demonstrator) in TTool was used for verification of combined analysis. Thanks to the security and performance modelling and analysis techniques supported by TTool, it was possible to model the trade-off between security and performance to support the product life cycle and different security levels.

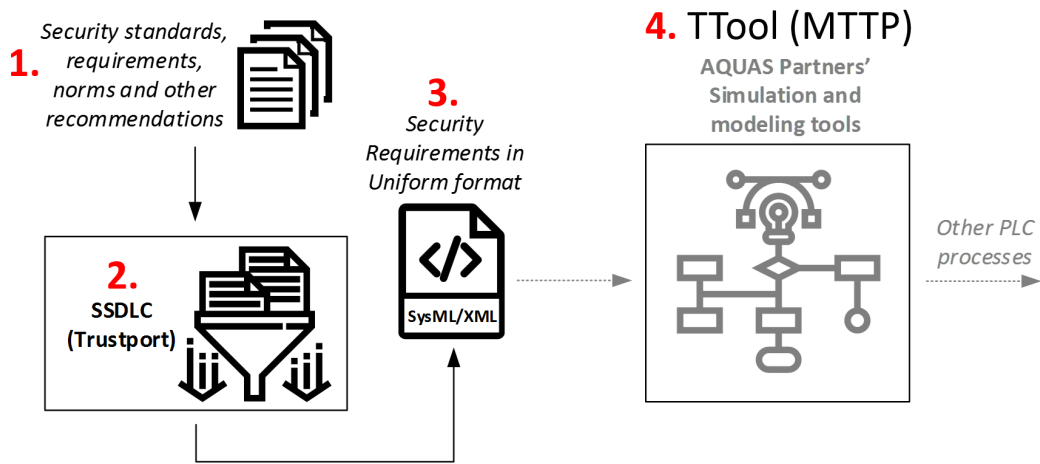


Figure 4-19: Combined Analysis of Security and Performance to Support the Product Lifecycle using SSDLC and TTool

4.7.3 Results

We provide deep analysis to define the security parameters and security level based on the most current norms, standards, risk analysis, and best practice together with the modelling and verification of performance analysis in the TTool toolkit.

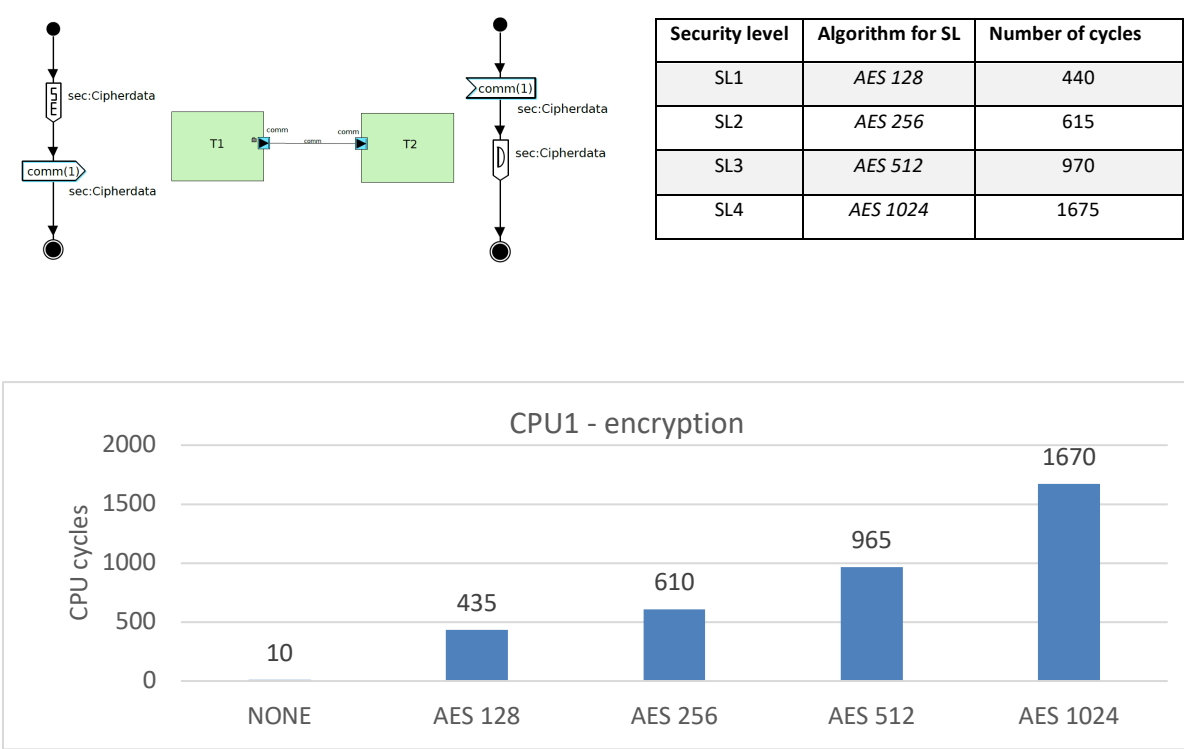


Figure 4-20: TTool – results security algorithm vs. performance

Table 4-6: Results - Computation time according to different security levels and algorithms

Title	Security level	Algorithm/Method for SL	Computation time [ $\mu$ s] for 400 MHz
Non-repudiation / authentication	SL1	<i>RSA 1024</i>	3.85
	SL2	<i>ECDSA P-256</i>	TBD
	SL3	<i>ECDSA P-384</i>	TBD
	SL4	<i>RSA 2048</i>	TBD
Communication integrity	SL1	<i>SHA-1</i>	1.95
	SL2	<i>SHA 224/256</i>	4.39
	SL3	<i>SHA 384</i>	5.54
	SL4	<i>SHA 512</i>	5.54
Using encryption	SL1	<i>AES 128</i>	1.09
	SL2	<i>AES 256</i>	1.54
	SL3	<i>AES 512</i>	2.43
	SL4	<i>AES 1024</i>	4.19

Table 4-7: Performance in terms of cycles depending on the clock divider

Master clock	T1/T2			CPU/Bus output			Output time
[MHz]	Clock divider	Width [Bytes]	Nb samples of	Name	Utilization	Cont. delay on MainBus_0	Cycles
200	1	40	1	CPU2	0.490234	15	1024
				CPU1	0.480469	32.5	
				MainBus	0.0195312	-	
200	2	40	1	CPU2	0.490272	15	2056
				CPU1	0.480545	34.5	
				MainBus	0.0233463	-	
200	3	40	1	CPU2	0.490291	12.8571	3090
				CPU1	0.480583	30.8571	
				MainBus	0.0252427	-	
200	4	40	1	CPU2	0.490348	15	4144
				CPU1	0.480695	37	
				MainBus	0.030888	-	
200	5	40	1	CPU2	0.489856	15	5126
				CPU1	0.480101	34.9	
				MainBus	0.0195084	-	

The combined analysis of Security and Performance using SSDLC and TTool was also used for Interference Analysis in UC4; see Figure 4-21.

				Module:	Client Platform					
		(individual R-Nr. (map))	Requirement / Standard Category	Requirement provider (UC partner)	Remote Control Platform					
					selectic	selection	selection			
						Expected Interference [yes/no]	Reason [optional]	When to deal with this interference? [concept phase, verification phase, ...]	Interference Analysis Output (actions to be taken, mark when action was completed)	How to analyze the expected interference?
					FR_GroupTo p001	*	*	*	*	*
Security	3.2	FR2 – UC	Use Control	Trustport	X	yes	Auditing activity can affect	concept phase	Report with QRA:	Quantitative risk
		FR3 – SI	System Integrity		X	yes	Potential ramifications on system performance and capability to recover from system failure (SR 3.1 – Communication integrity). Example of the simulation of SHA on CPU cycle/time in Ttool.	concept phase	Open: The proposal of algorithm/method for fulfilling SR 3.1 without affecting performance. E.g. Based on results of simulations in TTool.	simulation (TTool)
	3.3			Trustport						
		FR4 – DC	Data Confidentiality		X	yes	Potential ramifications on control system performance and the capability to recover from system failure or attack (SR 4.1 – Information confidentiality). Results of simulations in TTool.	design phase	The proposal of algorithm/method for fulfilling SR 4.1 – Information confidentiality, e.g. based on simulations in TTool.	simulation (TTool)
	3.4			Trustport						
	3.5	FR5 – RDF	Restricted Data Flow	Trustport	X	no	-	-	-	-

Figure 4-21: Inference analysis using SSDLC and TTool

#### 4.7.4 Lessons Learned

This approach helps when there is an investigation of whether or not the solution might move to the next security level (e.g., because of new regulations or others) without any necessary additive implementations or tests. We reveal that each security level (e.g. with higher key sizes) slows down the speed (e.g. encryption speed in cycles per byte) by particular exact value.

More generally, this approach helps to show the relation between each security level independently on hardware and implementation. This should serve in the decision processes in PLC phases before design or for example after regulation (law) change. It is obvious that it is necessary to think about the performance and security parameters already in the early stages of PLC as it might reduce significant number of issues caused by insufficient number proposal, which will lead to going back in PLC.

#### 4.7.5 Further Developments

The combined analysis will be extended to the other security requirements (e.g. non-repudiation/ authentication with The Elliptic Curve Digital Signature Algorithm (ECDSA) or different key sizes), which significantly influence the performance (e.g. delay for emergency stop).

## 4.8 Failure Modes, Vulnerabilities and Effect Analysis (FMVEA) (Industrial Drive Use Case Example)

Contributor: AIT

The proposed method uses Failure Modes and Effects Analysis (FMVEA) for combined analysis of safety and security at the concept phase. The approach is illustrated via application to the industrial drive use case (UC4).

FMVEA was devised as a security extension of the well-introduced safety-related analysis method FMEA (Failure Modes and Effects Analysis). The FMVEA tool has been developed as a research prototype in previous projects and now supports automated safety and security analysis by applying rules to a model of the system or specific component under consideration.

### 4.8.1 Aim

The goal of FMVEA is to provide an integrated analysis method comprising safety and security. The tool prototype, whose application is planned in UC4, enables a high degree of automation and reuse. A functional model of the item under consideration (system, subsystem, component) is imported or created, and enhanced with dependability attributes. Based on a previously created database of safety, security, and performance rules, an automated co-analysis of the model is started with the capability to immediately adapt the model to necessary safety, security or performance mitigation measures. The automated co-analysis can immediately be repeated so that the effect on the other quality attributes (violation of rules) can be seen immediately, and countermeasures can be taken.

### 4.8.2 Method

FMVEA (Failure Modes, Vulnerability and Effect analysis) is an extension of the well-established Failure Mode and Effect Analysis (FMEA) and was developed for the application on connected industrial systems. Additional evaluation was done on automotive systems and comparison with other safety & security co-analysis methods.

The basic concept behind FMEA is that, based on a system mode, failure modes for elements are identified and the consequences on the overall system are determined. FMVEA extends this with a parallel and even combined consideration of failure modes and threat modes, e.g. not only how a component could fail but also how a threat agent could misuse a component.

A tool has been developed to implement this methodology. The system is modelled in a Model-Editor and the threats and failure modes are described using simple grammar, allowing users to specify expected behaviour or potential risks.

Identified threat and failure modes can be taken from the tool and ranked based on their likelihood and combined with the impact of the system level effects to determine risks and decide on necessary risk treatment options. In the current implementation, risk assessment and risk treatment decision would be outside of the tool.

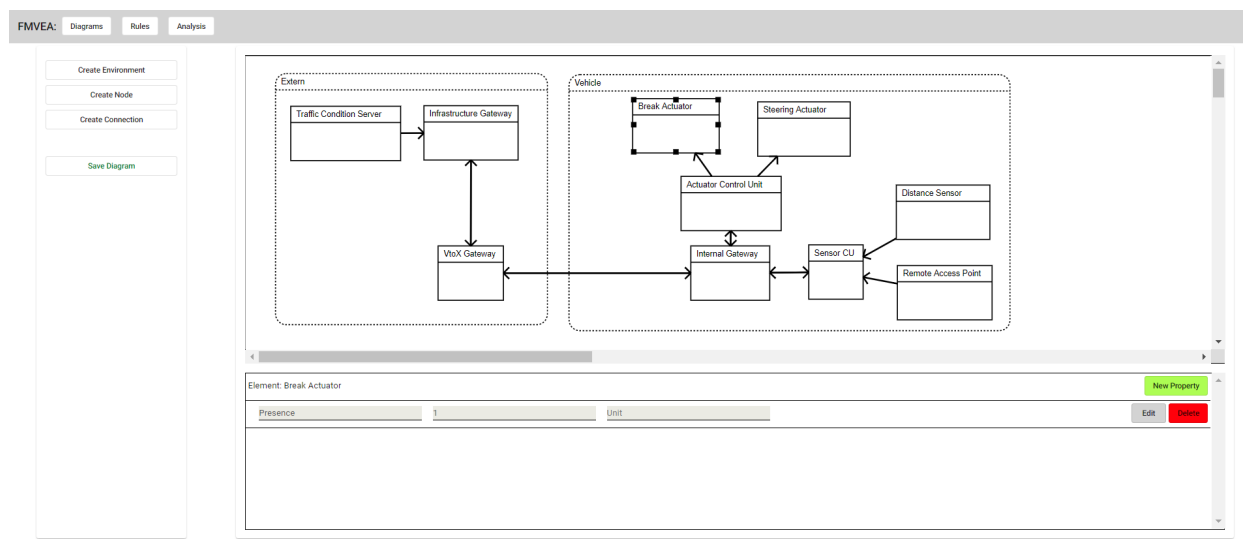


Figure 4-22: System-Model

Figure 4-22 shows the diagram from the use case modelled in FMVEA. On the left side of the Figure are related actions such as “Create Environment”, “Create Node” and “Create Connection”. An Environment can be considered as a container, which provides general attributes to its children. Attributes can be focused on security and safety. The attributes of an element are directly displayed below the diagram.

FMVEA: Diagrams Rules Analysis				
Rules:				
#	Name	Description	Rule	Actions
1	Secure Remote Acces Point	If the Remote Access Point of a Vehicle is not secured by a Authitication this Access Point could be used to harm the System	Remote Access Point attributes(Authentication=false) hasAncestor(Vehicle)	Edit Delete
2	Presence of Break Actuators	If a Break Actuator fails the system should have at least one additional one to activate the breaks in an emergency	Actuator Control Unit.hasConnection(Connection.to(Break Actuator attributes(Presence=2)))	Edit Delete
3	Update Frequency of the Traffic Condition Server	If the Traffic Condition Server does not send new data in a specific frequency the security of the system is no longer completely given.	Connection.from(Traffic Condition Server) to(Infrastructure Gateway) attributes(Update Frequency>10s, Encryption=false)	Edit Delete
4	External Gateway's Update Frequency and Security	External Gateway's must communicate their data in a sufficient manner and over a encrypted connection.	Connection.from(Infrastructure Gateway) to(VtoX Gateway) attributes(Update Frequency>10ms, Encryption=false)	Edit Delete
5	External Gateway to Internal Gateway Communication	The commutation between an external Gateway and an internal one must have sufficient encryption, authentication and throughput.	Connection.from(VtoX Gateway) to(Internal Gateway) crosses(ROOT) attributes(Encryption=false, Authentication=false, Throughput>64bytes/frame)	Edit Delete
6	Connection between Sensor and Sensor CU	The connection between a sensor and a sensor computation unit must provide a minimal throughput.	Connection.from(Distance Sensor) to(Sensor CU) attributes(Throughput>64bytes/frame)	Edit Delete

Figure 4-23: Defined Rules

Figure 4-23 displays the rules which should be used to analyse the use case diagram. From the left to the right you can see the name of the rule, then a short description and the “Rule”. The “Rule”-column is the most important one, because here, the actual rule for the analyzer is defined.

### 4.8.3 Results

FMVEA: Diagrams Rules Analysis				
Results:				
# 1	Diagram: Car UseCase	Date: 15/3/2019		
#	Rule	Affected Elements	Affected Connections	Show
1	Secure Remote Acces Point	Vehicle Remote Access Point		Show
2	Presence of Break Actuators	Break Actuator Actuator Control Unit	Actuator Control Unit — Break Actuator	Show
3	Update Frequency of the Traffic Condition Server	Traffic Condition Server Infrastructure Gateway	Traffic Condition Server — Infrastructure Gateway	Show
4	External Gateway's Update Frequency and Security	Infrastructure Gateway VtoX Gateway	Infrastructure Gateway — VtoX Gateway	Show
5	External Gateway to Internal Gateway Communication	VtoX Gateway Internal Gateway Extern Vehicle	VtoX Gateway — Internal Gateway	Show
6	Connection between Sensor and Sensor CU	Distance Sensor Sensor CU	Distance Sensor — Sensor CU	Show

Figure 4-24: Analyzer Results

Figure 4-24 displays the results of the use case analysis. The previously created rules were applied on the created diagram. From the left to the right, it's possible to see the applied rule and the results of the specific rule on the diagram. The affected elements and connections can be viewed in the diagram if the user clicks on the "Show"-button to the right. Inside the diagram, the affected elements and connections get highlighted by a red border as displayed in Figure 4-25.

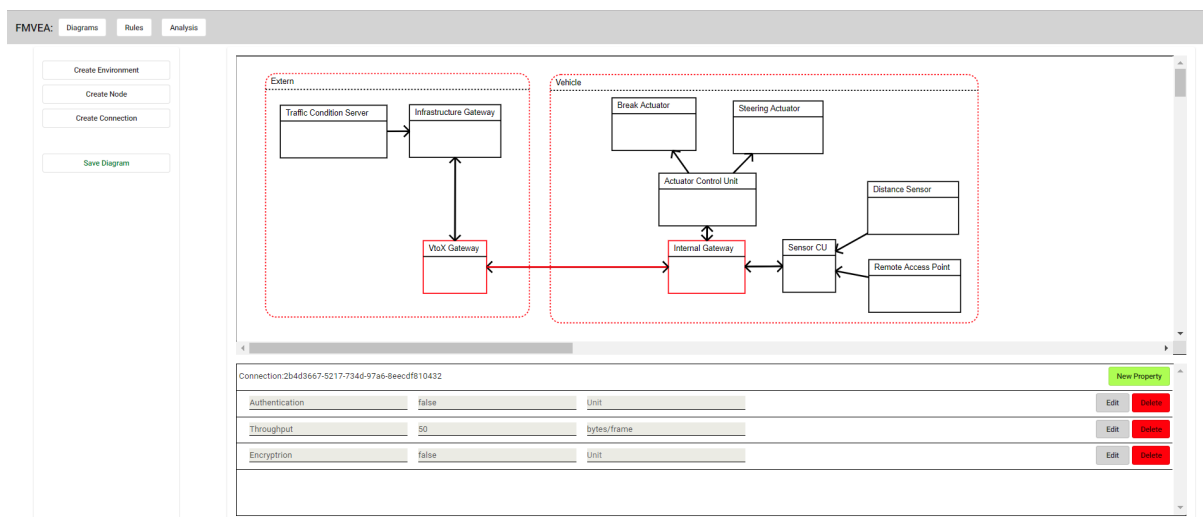


Figure 4-25: Results shown in the System Model

Figure 4-25 shows the affected elements of the fifth rule. The definition of the first rule says that if there is a connection between the "VtoX Gateway" and the "Internal Gateway" which has the attributes "Encryption=false", "Throughput < 64 bytes/frame" and which crosses the "Root Environment/Boundary" (leaves secured environment). As can be seen in the diagram, both attributes are fulfilled and the connection leaves the environment/boundary of the vehicle. The observed connection and the influencing elements like the gateways and the boundaries are marked red inside the diagram.



#### 4.8.4 Lessons Learned

##### *Revision of the rule-grammar*

When creating the rules, it has been noticed that many formulations are very cumbersome. This is because no negation of statements is possible. Furthermore, it is necessary that more generic rules can be defined. This would improve the reusability of rules. Often elements and connections are created, which fall under certain categories. These categories should also be able to be examined by rules. This results in generic analyses that can be made simpler and more comprehensive.

##### *Separate diagrams for logical and physical elements*

In this use case, an attempt was made to map both logical and physical aspects of a system to a single diagram. This approach has some disadvantages, as many logical aspects cannot be fully reflected in the physical image. This in turn means that essential information can be lost. Therefore, in a subsequent version of the tool, the logical and physical representation should be separated. The respective elements in both diagrams are then logically linked to each other by a shared Id. A basic rule here is that there must always be a physical one for each logical element, but a physical element can be modelled but not represented in the logical diagram.

#### 4.8.5 Further Developments

##### *SysML Import/Export*

As described in the previous section, some adjustments are made to the editor. The separation of the representation into two different diagrams also allows the editor to adapt to two standardized diagram types. A SysML diagram is implemented for the physical representation and a data flow diagram for the logical representation. For both diagram types, an import export function is also implemented after the implementation, so that existing models can be imported, or created diagrams can be exported.

##### *Predefined Categories and Diagram Elements*

When creating a diagram, the user has to spend a lot of time defining the individual elements and their connections as well as their properties. It has turned out that most elements and connections within a diagram occur multiple times. Therefore, a menu is to be implemented in which the user can predefine elements and connections and subdivide them into categories. These objects can then be easily used in the diagram, which saves a considerable amount of time.

##### *ReqIF Export Interface for GSFlow*

In addition to the development of the "FMVEA" tool, the tool "GSFlow" is also developed. GSFlow is a standard management tool designed to make it easier for the user to meet public safety / safety standards. Requirements are then to be generated from the identified risks in a system, which can then be subsequently exported in the "ReqIF" format. In addition to this general export function, an additional interface is created which allows these requirements to be automatically incorporated into GSFlow.

### 4.9 Combined Analysis of Safety, Security and Performance in the Design Stage (Space Use Case Example)

Contributors: All4Tec, Intecs and Tecnalia

The goal of this proposed method is to allow early validation of the security, safety and performance requirements coming from the Requirements stage, identify new safety, security performance

requirements (e.g. according to the introduction of mitigation solutions), check the feasibility of the updated set of requirements, to properly feed the implementation phase, and give a model-based support to be able to determine if the triggering of a trade-off meeting should be enabled.

The method performs safety, security and performance (SSP) analysis upon the defined model, by using seamless integrated tools. The method also enables concept-aware analysis by allowing tracing how the entities defined in the models, design and/or analysis-specific ones, are related to the SSP concern(s), to monitor their relationships (interference), evolutions, so to support the identification of the need of trade-off decisions and co-engineering meetings.

#### 4.9.1 Aim of Use Case Example

A model of the software architecture has been provided for the space use case (Use Case 5), by considering functional and safety, security and performance requirements, to support the design stage.

We recall that UC5 concerns the evaluation of software running in a multi-core processor supported by a well-known hardware architecture that is commonly used in space projects [Deliverable 2.2].

The proposed methodology supports the Design stage for the software system and the allocation of software components to the target platform. The intended usage scenario is expected to determine new requirements for the development stage, raise SSP conflicts warnings to be solved in specific trade-off meetings, to finally build a baseline of the requirements to feed the implementation stage.

#### 4.9.2 Method

First a model of the software design by using the CHES modelling language and toolset is provided. In particular software components satisfying functional and performance requirements are designed, together with the allocation of software components to processors cores. Then model-based schedulability analysis of the designed software architecture is enabled in CHES to check the feasibility of the solution.

The CHES model has been imported in the Safety Architect tool to conduct local safety analysis at the design level. The refined model (e.g., adding safety mechanisms) is complemented with the Cyber-Architect tool for the security analysis. As result of this safety-security analysis, fault tree analysis models are produced for the feared events.

Then, the Concept-aware analysis tool by Tecnalia was applied on the produced fault tree analysis models to create high-level reports on the interference of safety and security aspects, as well as in the interference of the logical, physical and functional layers of the design.

Figure 4-26 depicts the previously presented interactions.

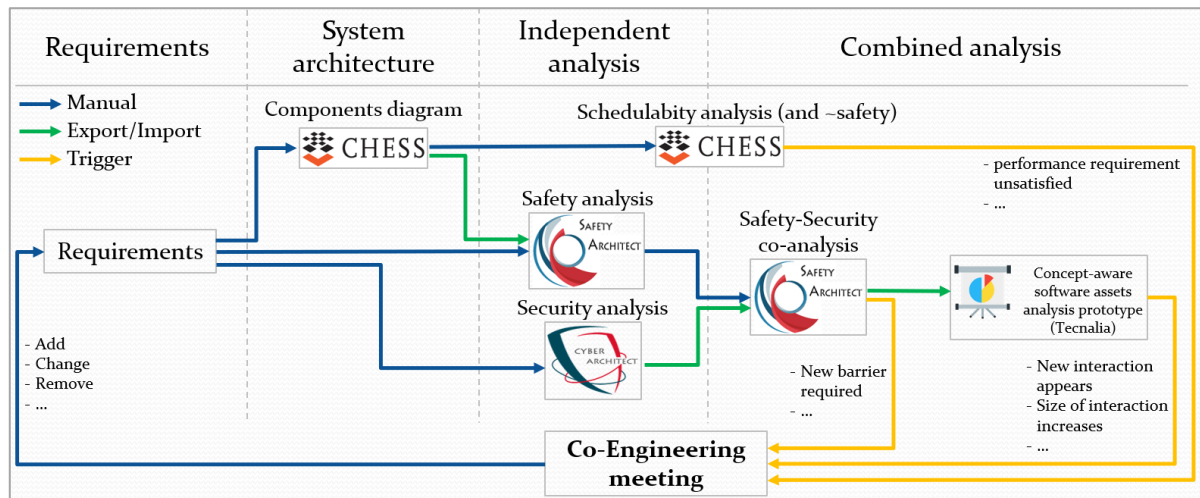


Figure 4-26: Description of the interactions

### 4.9.3 Results

Some of the results obtained are presented below. The following figure shows how the CHES models contain time-related annotations that are used to check if safety requirements related to performance are satisfied; further details about the CHES model of the Space Use Case Example are available in D2.3.5 Section 4.1.2.

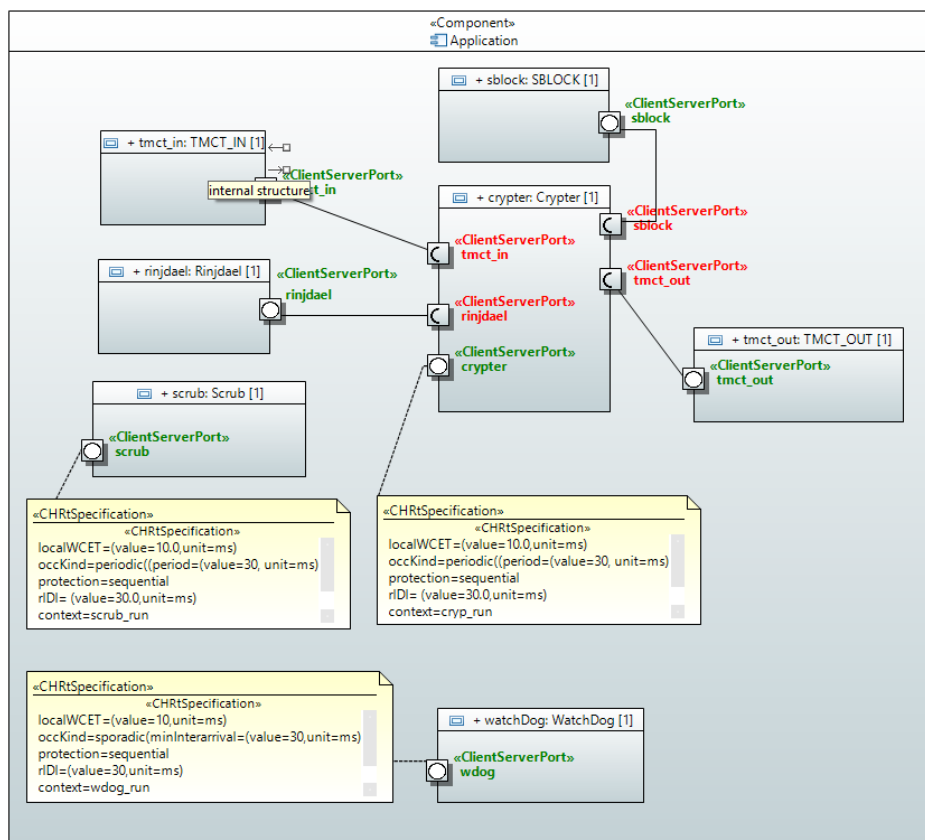


Figure 4-27: Software architecture

«SaAnalysisContext» AnalysisContext							
UML	SW instance	context	arrivalPattern	localWCET	rldl	ResponseTime	Sched
Comments	Application.scrub	scrub_run	periodic((period=(value=30, unit=ms))	(value=10.0,unit=ms)	30.0ms	10.0ms	YES
Marte	Application.watchDog	wdog_run	sporadic(minInterarrival=(value=30,unit=ms))	(value=10,unit=ms)	30ms	20.0ms	YES
Profile	Application.crypter	cryp_run	periodic((period=(value=30, unit=ms))	(value=10.0,unit=ms)	30.0ms	10.0ms	YES
Advanced							
ContractEditor+	HW instance	Utilisation	Res				
Ports	System.cPU	(value=,statQ=calc,mode='20190322113439')					
AnalysisContext	System.cPU_core0	(value=33.33,statQ=calc,mode='20190322113439')					
	System.cPU_core1	(value=66.67,statQ=calc,mode='20190322113439')					

Figure 4-28: Schedulability analysis results

Thanks to the prototype bridge between CNESS and Safety Architect, the software architecture was imported from the CNESS model to Safety Architect. A safety analysis was conducted in Safety Architect as explained in D2.3.5.

For the needs of the demonstration of the safety-security combined analysis and to feed the concept-aware tool of Tecnia, a scenario of software architecture evolution, from a very basic model to the current model containing safety and security barriers, was proposed. Safety and security analyses were performed at each evolution of the architecture and fault trees were generated. Finally, a safety-security co-analysis was realised in Safety Architect. These fault trees were exported to be exploited by Tecnia's tool. Figure 4-29 presents an example of a safety-security tree and Figure 4-30 shows a safety-security tree exported in OpenPSA format and integrated tags on the node of the tree to indicate their related-viewpoint (safety, security or safety-security) and their related-type (logical, physical, functional or generic).

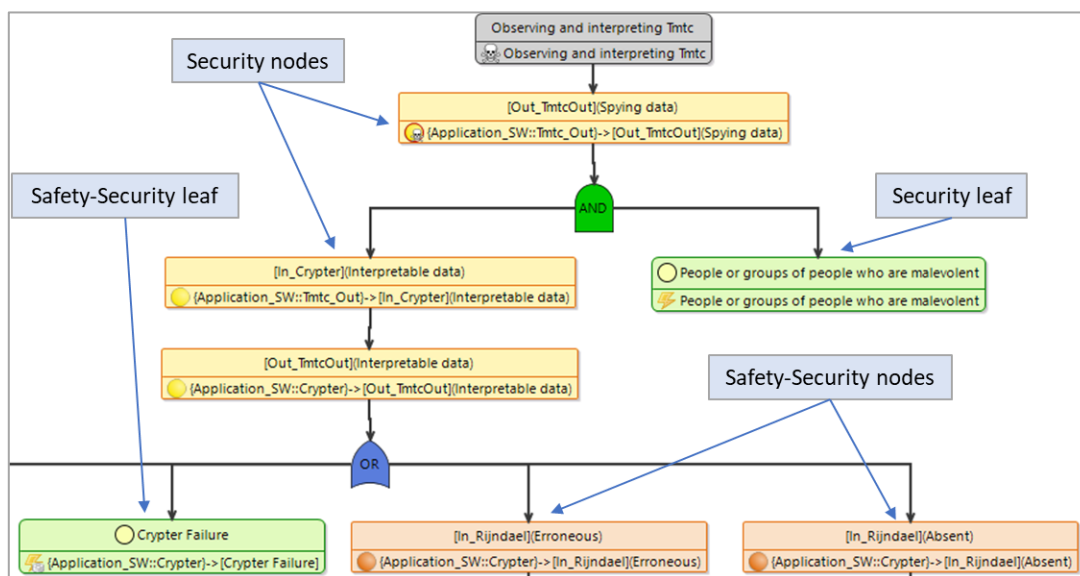


Figure 4-29: Example and part of a safety-security tree

```

<define-component name="In_Scrub" role="private">
  <label></label>
  <define-gate name="[In_Scrub] (Absent)" role="private">
    <label>{Application_SW}&#xD;::SBlock&#xD;[In_Scrub]&#xD;(Absent)</label>
    <attributes>
      <attribute name="viewpoint" value="Safety"/>
      <attribute name="type" value="Logical"/>
    </attributes>
  </or>

```

Tags

Figure 4-30: Example of a tree exported in OpenPSA format with tags on the nodes

Then, after applying Safety and Cyber-Architect, the concept-aware analysis is used as shown in D2.3.4.

#### 4.9.4 Lessons Learned

One of the lessons learned is related to the importance of the evolution of the assets in the design stage. Indeed, after several analysis iterations, the engineering teams (safety, security) produced analysis results such as fault trees, FMEA, attack trees or threat scenarios. The number of artefacts contained by these analysis results can indicate the advancement of the workload in each engineering field. Then, if it overcomes an arbitrary threshold or if the variation is important, it may trigger a co-engineering meeting, to encourage this practice and specially to perform co-engineering as early as possible. By analyzing how the assets evolved during the use of Safety- and Cyber-Architect tools, it was possible to prototype an enhancement of the Concept-aware analysis tool to provide reports on the evolution. A screenshot is presented below showing how the interference of safety and security appears at a given point in the evolution. It is being investigated how this evolution can happen within the design stage but also can continue in next stages after some iterations.

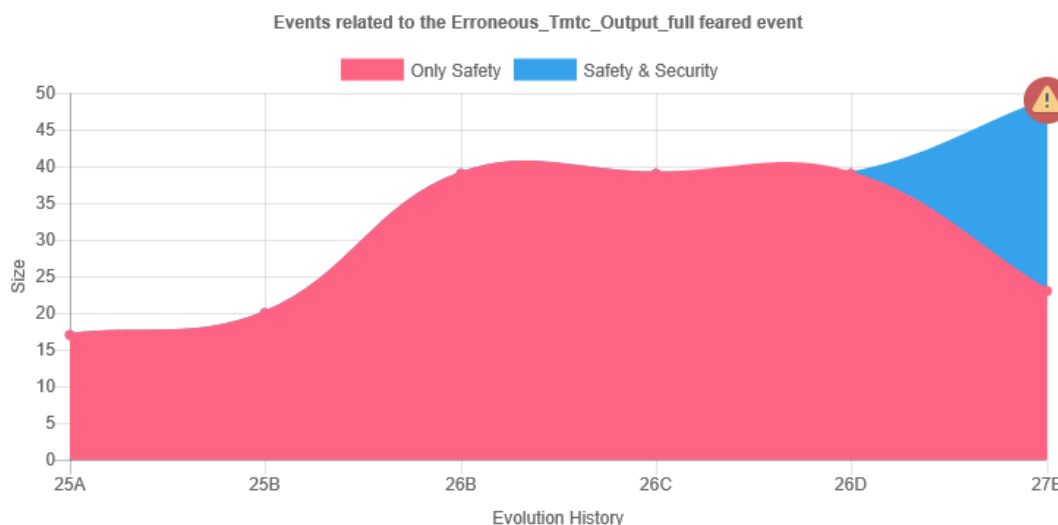


Figure 4-31: Output of Concept-aware analysis tool

#### 4.9.5 Further Developments

Further developments concern the tooling (WP4) to support this kind of combined analysis. One of the objectives is to enhance the safety-security analysis. Firstly, this requires improving the import of the security artefacts from Cyber Architect to Safety Architect. Next, the safety-security analysis is presented via the safety-security trees then it is extended to the FMEA with the creation of a FMVEA table.

The concept-aware analysis and tool will be extended to requirements and to the entities available in the CHES architecture model.

Support for traceability between architectural model-level entities and safety/security/performance requirements will be enabled in the CHES environment; a traceability view will also be provided to support the various experts involved in the interaction point triggered by schedulability analysis result, to understand the requirements actually involved in the context of the current architecture under review.

The complete models are provided in the Annexes.

## 4.10 Translation Validation: Checking C Code Conformity (Rail Use Case Example)

Contributor: CEA

The proposed method provides an essential verification step in the final stage of an iterative process of refinement. The refinement process transitions from an initial, high-level, abstract behavioural specification to an implementation, progressively incorporating combined safety, security and performance (SSP) requirements. This final stage completes the validation of the translation of a formal model to compilable source code and involves functional contract generation and verification with static code analysis.

To illustrate the process, we chose a simplified but representative version of the cyclic redundancy check (CRC) used in the railways use case (UC3) to test the integrity of received data. We recall that UC3 is COPPILOT, a safe controller for screen doors separating a platform from the tracks of metro rail systems. The development of the CRC follows the B method: a formal specification is refined into a formal algorithm, and we formally prove that the latter indeed satisfies the former. Finally, the formal algorithm is translated to C code, and the result is integrated into the rest of the software development.

A refinement verification method was developed to analyze the conformity of C code generated by the Atelier B (development framework of ClearSy supporting the B method) with the specifications of UC3 expressed in the B0 refinement language.

To reach that objective, the soundness of the approach and the translation of B0 specifications to ACSL specifications (<https://frama-c.com/acsl.html>) have been studied by CEA. The method was also applied by CEA to a representative example supplied by ClearSy using the Frama-C tool (developed by CEA).

### 4.10.1 Aim

The B method is a design process to develop software using B notations and formal proofs. The software development starts from abstract specifications of the software behaviours expressed in the B language. These specifications are iteratively and manually refined, progressively incorporating SSP requirements, until they only use B0 notations that become directly translatable into the programming C language. Atelier B performs the formal proof of each of these successive refinements and translates the last one into C source code to compile.

The goal of the proposed method is to provide the missing link (formal proof) between that C source code and the last refinement written in B0 language. This is to ensure that the obtained C source code correctly represents the formal algorithm. Applying such a method guarantees, through formal proofs, that the resulting C source code complies with the B0 specifications and, as a side effect, with the initial abstract specifications. Indeed, translation validation is not covered by the B method, but is crucial in

the safety process: an error in the translation would lead to a software not executing the intent of the developer, represented in the B formal model (traceability is broken).

The current conventional way of validating the translation is to have two separate software teams develop a translator satisfying some translation requirements designed by the validation and verification (V&V) team. From a given formal algorithm, the translation is accepted only if both translators yield the exact same result (up to spaces).

In order to ease both the validation of translation and the development of translators, we propose to translate the formal algorithm into a pure mathematical description of its effects in the ACSL language. The result is formally checked against its actual translation with the Frama-C platform, which allows one to prove that the obtained C source code satisfies the ACSL description.

While the conventional way relies fully on a syntactic correspondence, this new strategy enables semantic validation, which allows more flexibility in the translation to C code (optimizations, compositions, etc.). In the end, only one translator from B to C has to be developed, and translation validation could be automatically ensured by Frama-C, relieving the whole process from the V&V team.

#### 4.10.2 Method: Verification of conformity of Generated C code

##### *Verification approach*

The verification the conformity, with B0 specifications, of the C code issued by the code generator of the Atelier B (from the specifications of the software implementation expressed in the B0 refinement language) can be decomposed in two successive steps:

1. First, for each B operation, their B0 implementation must be translated into a function contract written in ACSL, similarly to the C code generator of the Atelier B that translates each B0 implementation of an operation into a C function.
2. Then, the WP plug-in of Frama-C tool (<https://frama-c.com/wp.html>) has to prove that all C functions satisfy their own ACSL function contract.

To completely prove that the generated C code refines the B0 implementation, an extra verification ensuring the absence of run-time errors (while executing the compiled C code) has to be done. This last verification was considered out of scope of this railway case study, even though it can be done with the EVA plug-in of Frama-C (<https://frama-c.com/value.html>).

##### *Verification method*

In order to reach that verification objective, Frama-C tool and the translation principle of B0 implementations to ACSL function contracts have to be considered.

The WP plug-in performs deductive verifications based on weakest precondition computation.

That technique provides proofs of the correctness of C-functions against their ACSL specifications. The proofs are modular: a function is verified independently from its calling context, and only from its C source code and the specification of the other functions. For each ACSL code annotation, the WP plug-in generates a bundle of proof obligations (i.e., first-order logic formulae) that entail their correctness. Then, these proof obligations are submitted to the automatic theorem prover Alt-Ergo or the Coq proof assistant. Notice that many other provers (such as Z3, CVC4...) can also be used via the Why3 platform.

The translation principle from B0 language to ACSL function contracts defines for each B0 operation two ACSL clauses:



1. an 'assigns' clause specifying that only one output (of the related B0 operation/C function) is allowed
2. an 'ensures' clause with a 'predicate' defining the constraint between these outputs and the inputs (of the related B0 operation/C function).

In order to verify that the C implementation conforms to the B0 implementation, contracts of the directly called operations have to be supplied and verified separately in the same way. This verification is modular and the complexity of the formal proofs do not increase by ascending in the call graph of the operations unlike most of the deductive approaches. That looks like a unit verification where the precise definition of the contracts related to the called operations is unnecessary. The called procedures can be seen as grey boxes where only the knowledge of their input/output operands has to be considered.

#### *First tooling results*

A prototype of a translator from B0 language into ACSL was issued by CEA during the AMASS project as proof of concept of the verification method. That prototype does not handle all B0 notations contained in the representative example supplied by ClearSy. It needs to be completed to handle loop statements and array data type.

A formal proof of the correctness of the method has been issued in Coq (<https://coq.inria.fr>) from the AQUAS project.

The translation rules have been defined. Particular cases leading to one or more alternatives in the translation have been identified. These alternatives have been evaluated in terms of proof automation result in order to implement the more efficient one into the next version of the prototype. To perform the conformity verification of loop statements in an automatic manner, dedicated lemmas have to be provided by the translator. These lemmas have been identified in accordance within the chosen external prover. The next version of the prototype will be improved in order to cover the translation of loop statements and the use of array data type in B0 notations.

CEA set up an evaluation environment including the first translator prototype, Frama-C with its WP plug-in and the external prover Alt-Ergo (URL: <https://alt-ergo.lri.fr/>). The environment has been delivered to ClearSy. The evaluation of Frama-C results, in terms of proof success, identified several issues. These issues come from the nature of the generated ACSL specifications to prove, which are quite different to the ones written by hand.

The next version of the prototype will be improved in order to cover the translation of loop statements and the use of array data type in B0 notations. Some issues in the WP plug-in of Frama-C have been fixed into the development version since the evaluation environment was delivered to ClearSy. The next delivery, planned for the end of June, will include new versions of both the B0 to ACSL translator and Frama-C with the WP plug-in.

### 4.10.3 Results

From an extract of the B formal initializer in Table 4-8, we get the corresponding C code in Table 4-9 with the B to C translator of the project.

```
INITIALISATION
vid := 0 ;
clmcr := 0 ;
clmrl := ( 0 .. c_sizeof_msg_minus1 ) * { 0 } ;
```



```
clmr2 := ( 0 .. c_sizeof_msg_minus1 ) * { 0 }
```

Table 4-8: formal B source algorithm

```
/* Array and record constants */
/* Clause CONCRETE_VARIABLES */
uint32_t donnees__vid;
uint32_t donnees__clmcr;
uint8_t donnees__clmr1[cst_projet__c_sizeof_msg_minus1+1];
uint8_t donnees__clmr2[cst_projet__c_sizeof_msg_minus1+1];

/* Clause INITIALISATION */
void donnees__INITIALISATION(void) {
    unsigned int i = 0;
    donnees__vid = 0;
    donnees__clmcr = 0;
    for(i = 0; i <= cst_projet__c_sizeof_msg_minus1;i++) {
        donnees__clmr1[i] = 0;
    }
    for(i = 0; i <= cst_projet__c_sizeof_msg_minus1;i++) {
        donnees__clmr2[i] = 0;
    }
}
```

Table 4-9: C translation

On the other hand, Table 4-10 shows the translation of the same B formal initializer to ACSL.

```
axiomatic donnees_i_predicates{
    predicate B0_donnees__INITIALISATION(uint32_t donnees__clmcr__1,
        uint8_t donnees__clmr1__1[B0sizeof1_donnees__clmr1],
        uint8_t donnees__clmr2__1[B0sizeof1_donnees__clmr2],
        uint32_t donnees__vid__1) =

    donnees__vid__1==0 &&
    donnees__clmcr__1==0 &&
    \forall integer B0i0;
    (0<=B0i0<=B0_cst_projet__c_sizeof_msg_minus1)==>donnees__clmr1__1[B0i0]==0) &&
    \forall integer B0i0;
    (0<=B0i0<=B0_cst_projet__c_sizeof_msg_minus1)==> donnees__clmr2__1[B0i0]==0);

module donnees_i:
function donnees__INITIALISATION:
contract:
    assigns donnees__clmcr,donnees__clmr1[..],donnees__clmr2[..],donnees__vid;
    ensures B0implm: B0_donnees__INITIALISATION(donnees__clmcr,
        (uint8_t [B0sizeof1_donnees__clmr1])donnees__clmr1,
        (uint8_t [B0sizeof1_donnees__clmr2])donnees__clmr2,
        donnees__vid);

at loop 1:
    loop assigns i, donnees__clmr1[0..B0_cst_projet__c_sizeof_msg_minus1];
    loop invariant idx1: 0 <= i <= B0_cst_projet__c_sizeof_msg_minus1+1;
    loop invariant init1: \forall integer k; 0 <= k < i ==> donnees__clmr1[k]==0;

at loop 2:
    loop assigns i, donnees__clmr2[0..B0_cst_projet__c_sizeof_msg_minus1];
    loop invariant idx2: 0 <= i <= B0_cst_projet__c_sizeof_msg_minus1+1;
    loop invariant init2: \forall integer k; 0 <= k < i ==> donnees__clmr2[k]==0;
```

Table 4-10: ACSL translation

Finally, the resulting C code is verified with regards to the ACSL specification with Frama-C, which leads to the translation validation result in Table 4-11.

Table 4-11: Validation report

Module	Function	Total	Valid	Failed	Success
crc_main	-	24	24	-	100
crc_i7	crc__Get_CRC	-	-	-	-
crc_i	INITIALISATION	-	-	-	-
util_i	util__get_add_uint32	2	2	-	100
util_i	util__INITIALISATION	2	2	-	100
verif_main	-	29	29	-	100
itf8	itf__get_ccrc1	2	-	2	-
itf	itf__get_ccrc2	2	-	2	-
verif_cpt_i	verif_cpt__dectc	2	2	-	100
verif_cpt_i	verif_cpt__dectr	2	2	-	100
verif_cpt_i	verif_cpt__initc	2	2	-	100
verif_cpt_i	verif_cpt__INITIALISATION	2	2	-	100
verif_cpt_i	verif_cpt__initr	2	2	-	100
verif_cpt_i	verif_cpt__testtc	2	2	-	100
verif_cpt_i	verif_cpt__testtr	2	2	-	100
verif_i	verif__INITIALISATION	2	2	-	100
verif_i	verif__process9	2	1	1	50
donnees_i	donnees__get_c1du	2	2	-	100
donnees_i	donnees__get_c1mr1	2	2	-	100
donnees_i	donnees__get_c1mr2	2	2	-	100
donnees_i	donnees__get_c1r2du	2	2	-	100
donnees_i	donnees__INITIALISATION	8	8	-	100

#### 4.10.4 Lessons Learned

The tool proved to be very efficient, though more work is needed in order to address more B constructs, and ultimately to cover the full language. Of particular interest for the following are loops:

<sup>7</sup> Translation of CRC implementation is not fully supported by the B0 to ACSL prototype (support for translation and proof of B0 loop statement is required).

<sup>8</sup> The ITF machine is not fully refined. Since there is no B0 specifications for it, the conformity of its operations cannot be validated with the proposed method.

<sup>9</sup> The PROCESS operation of the B0 implementation of the module VERIF is not fully automatic. The missing proof can be done using the interactive prover TIP of WP plug-in of Frama-C. In order to reduce necessary effort for performing that proof, improvements in the translator replacing some *existential* quantifiers by *let-in* constructs.

deductive verification requires the correct invariant to be computed, which might be tedious to automatize from the few examples we manually studied. Overall it was important to be able to verify, with the tool, that the implementation is conform to requirements derived from SSP tradeoff analysis, and that the tradeoffs were feasible from an implementation point of view. Feasibility feedback given by static code analysis can be used as an additional input in the iterative SSP requirements analysis.

## 4.11 Safety and Performance Analysis in Multiprocessor Task Scheduling (Space Use Case Example)

Contributor: ITI

The aim of the proposed method is to produce combined performance and safety metrics at the level of code generation and its assignment to processing cores in a multi-core system. The approach is illustrated by application to the Architecture Design Phase of the space multi-core use case (UC5). This example also illustrates the inter-operation of three different analysis tools and some newly-developed protocols which support the communications between the tools.

The method involves the inter-operation of three separate tools: A2K (ITI) is used to perform timing analysis computations and act as a manager for the other two tools; TimingProfiler (AbsInt) is used to calculate the Worst-Case-Execution-Times (WCET) of code modules; ANaConDA (BUT) is employed to measure dynamic safety metrics of the generated code.

The space use case (UC5) involves code running on a multi-core processor system. Timing and code safety analysis is essential to prove the code can be safely scheduled and run under all operating circumstances.

### 4.11.1 Aim

The main goal of this work is to produce performance and safety metrics for multi-threaded code running on multiple processing elements. These code segments are interruptible according to a specified scheduling protocol and they may intercommunicate with each other via a variety of mechanisms such as, for example, shared memory, communications ports, buses, etc. This is a general view of the specific scenario presented in the space use case.

The performance metrics which we desire are the response times and data throughput of the system's flows (defined below), while the safety metrics we require are (a) whether all the computational tasks meet their defined deadlines, and (2) measures of code-safety in terms of operations like access to shared memory areas and determination of possible data-races in multi-threaded code.

A secondary aim of this work is to investigate and develop communications protocols to enable inter-operation between the three analysis tools. Our goal is to provide the system architect with a tool suite which facilitates rapid interaction and analyses of different system designs, a process sometimes called "Design Space Exploration".

### 4.11.2 Method

The abstract computational model which we employ is based on a set of flows. A flow is defined as a collection of activities (or tasks) which are to be executed in some predefined sequence. Each flow has a specific activation pattern which defines the times that the flow's computations are started. The activities of each flow can be executed on different processing elements and have a predefined execution order. The activities might have access to shared resources such as memory, buses, or

peripherals. The various computational flows have defined priorities and their activities are scheduled according to the scheduling algorithm of the underlying operating systems.

The combined analysis employs 3 interconnected tools developed by different AQUAS partners.

#### A2K (ITI)

A2K provides editors to define the system architecture in terms of the hardware devices and their interconnections and also the software architecture in terms of code modules which are stored in a Git repository. A2K also defines a deployment model which is effectively a mapping of the code modules to processing elements along with definitions of the scheduling algorithm, the flow priorities, and execution orders of each flow's activities. A2K performs timing and scheduling analysis of the overall system. To do this, it requires the WCETs of the code segments. These are computed by the TimingProfiler tool described next. Finally, A2K manages communication with the TimingProfiler and ANaConDA analysis tools, and it also prepares analysis reports.

#### TimingProfiler (AbsInt)

This tool computes the worst case execution times of each code module stored in a GIT repository. We have developed a connection between A2K and TimingProfiler using the GraphQL protocol and a virtual machine (Docker) running the TimingProfiler. In this way, the A2K user can quickly obtain the execution times of code modules for analysis. If the code in the repository or the system's architecture is changed, then it is easy to re-run the timing analysis to obtain new reports.

#### ANaConDA (BUT)

ANaConDA is a framework that simplifies the creation of dynamic analysers for analysing 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 code. In addition, the framework also supports noise injection techniques to increase the number of inter-leavings witnessed in testing runs and hence to increase chances to find concurrency-related errors.

A connection between A2K and ANaConDA is currently under development. This is based on the Open Services for Lifecycle Collaboration (OSLC) protocol. Our goal here is to use the REST services provided by OSLC to send the code segments from a Git repository, perform the safety analysis, and return the results to A2K for assessment and reporting.

### 4.11.3 Results

At the present time, the algorithms for timing analysis in A2K and communication with TimingProfiler are complete and working. An example of the output of timing analysis is shown in Table 4-12. The important information to glean from this table is the computed response time of each task and whether this figure is smaller than the task's deadline. Results on the analysis of code safety using ANaConDA will be presented in due course when the OSLC interface is complete.

Table 4-12: Bin packaging thread allocation results

Bin Packing Thread Allocation Results

Allocation ViewSystem ViewAnalysis ViewUnallocated Threads

Change units for all fieldsUS

OPSW

Flow	Task	Priority	WCET	Offset	Computed Offset	Jitter	Blocking Time	Response Time	Deadline	Period	Feasible
F11 OPSW Memory Manager (17)		0						1,036,000	10,000,000	10,000,000	✓
	F11T1 OPSW Memory Manager (17)	0	1,036,000	0	0	0	0	1,036,000	10,000,000		✓
F6 OPSW Time Manager PPS Manager (25)		0						182	1,000,000	1,000,000	✓
	F6T1 OPSW Time Manager PPS Manager (25)	0	182	0	0	0	0	182	1,000,000		✓
F1 UART		0						10,000	300,000	300,000	✓
	F1T1 UART	0	10,000	0	0	0	0	10,000	300,000		✓
F7 FDIR EDR (26)		0						791	10,000,000	10,000,000	✓
	F7T1 OPSW FDIR EDR (26)	0	791	0	0	0	0	791	10,000,000		✓
F2 Coprocessor		0						250,000	1,000,000	1,000,000	✓
	F2T1 Coprocessor	0	250,000	0	0	0	0	250,000	1,000,000		✓
F3 HouseKeeping - TMTC		0						150,000	500,000	500,000	✓
	F3T1 HouseKeeping - TMTC	0	150,000	0	0	0	0	150,000	500,000		✓

#### 4.11.4 Lessons Learned

We are developing a flexible and highly interactive tool suite for performance and safety analysis of multicore code segments. The tool enables the design engineer to quickly and easily try out different system architecture designs and to evaluate their timing performance and code safety aspects. We are now applying this process to the code associated with the space use case. This, however, has required a certain amount of “reverse engineering” as, given the use case source code, we need to derive a suitable computational model in terms of flows and activities.

The verification of the timing analysis is somewhat problematic - the calculations are complex and rely on several assumptions. We need to determine if the computed response times are accurate. To do this we have built a monitoring system which downloads code to real hardware and physically measures the task response times. These are then compared to the results obtained from our analysis thus verifying the calculations or not. Preliminary experiments are encouraging and indicate that our timing analysis is correct.

#### 4.12 Efficient Formal Verification of System Software Using Ada 2012 and SPARK 2014 (Space Use Case Example)

Contributor: HSRM

HSRM's methodology aims to reduce the time and effort it takes to develop formally verified system software. In this example, the goal is to evaluate the said methodology and to cover the interaction between safety and security in an effort-efficient way (i.e. containing the development cost of applying formal methods). It is expected to achieve improvements in all three dimensions: safety and security will be improved due to the application of formal methods and the provision of formal correctness proofs, while development performance is expected to be improved (in comparison to other processes featuring formal verification) due to SPARK's ability to specify contracts as part of the source code. As for computational performance, a certain decrease is to be expected. The evaluation was done by

delegating a subset of the work to a student who was unfamiliar with the methodology and benchmarking his effort and his achievements.

#### 4.12.1 Aim in the Use Case

Ada 2012 is a programming language which was designed, and is commonly used, in the development of safety-critical systems. SPARK 2014 is a toolset and methodology based on a subset of Ada 2012 with extensions to enable the semi-automated definition and verification of formal correctness proofs.

A development process using Ada 2012 and SPARK 2014 was applied to the development of a critical component of a microkernel in the Space Multicore use case (Use Case 5). As the development process implies a defined methodology, the decision to apply it needs to be taken in the concept phase. The trade-off between increased safety, security and efficiency of development on one side and a potential loss of computational performance on the other needs to be taken into account in this early phase. This work is expected to yield criteria to facilitate such decisions.

We recall that UC5 concerns the development of software for space missions leveraging multicore processor architectures [Deliverable 2.2]. Space software applications rely on system services to be provided by an operating system kernel, which, for this use case, needs to support multicore scheduling. Such a kernel is the most critical component of any space-borne computer system; therefore, it should be proven to be correct by design.

The ultimate goal of our effort in UC5 is to apply our methodology to the design of a microkernel suitable for running space mission software. We aim to obtain formal proofs of correctness during the design phase. For this particular analysis, a central component which deals with the dynamic allocation of tasks to cores has been chosen: the scheduler. In order to demonstrate general effectiveness of the approach, this scheduler is implemented and formally proven.

#### 4.12.2 Method

HSRM's methodology uses a hierarchical approach to achieve formal verification more efficiently. A microkernel is being developed using the proposed methodology as an experimental platform to exercise the approach, the microkernel under development has a layered architecture where each layer is broken down into separate modules (see

Figure 4-32). Each module can be proven separately and can be used by higher layers. Additionally, if a module cannot be entirely or partially proven, it has to be tested intensively. A prototype of the microkernel was written in C, but the technology used for formal verification and implementation is SPARK 2014. SPARK is a subset of the programming language Ada which enforces necessary restrictions to make formal verification possible. Additionally, SPARK allows the definition of a formal specification in the form of contracts (verification conditions). Using the SPARK 2014 tools, a formal proof can be conducted to show whether the source code fulfils the contracts.

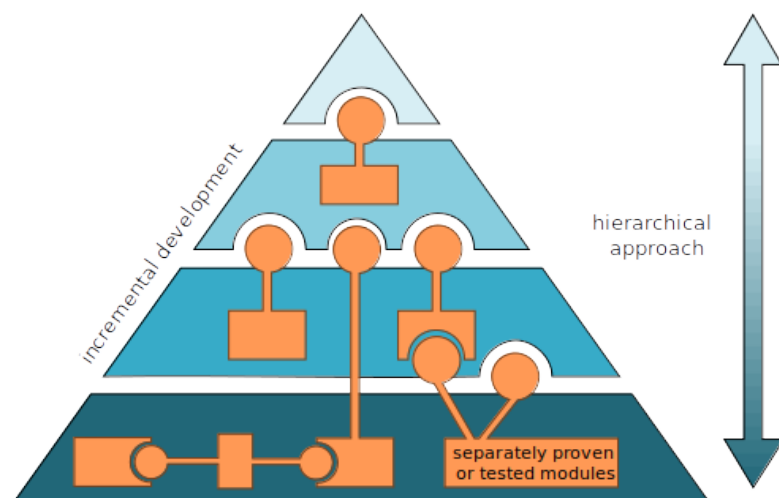
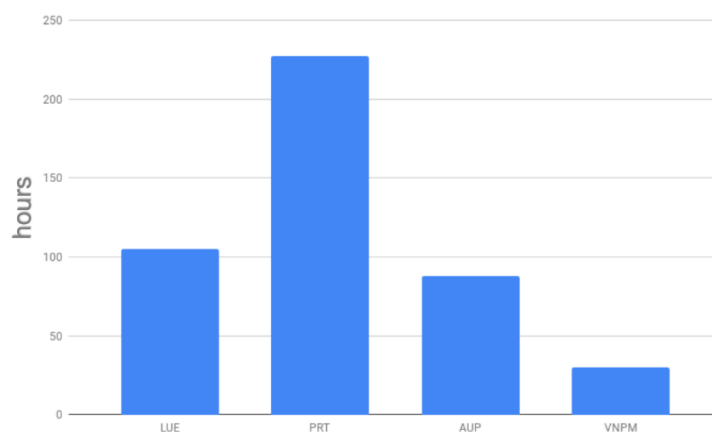


Figure 4-32: Hierarchical approach

### 4.12.3 Results

To benchmark this method, a student rewrote the scheduler module in SPARK and was able to verify 216 out of 224 verification conditions. The effort was limited to 450 hours. During his work, the student categorised and noted his effort as follows:



- **LUE:**
  - Literature study & Initial training
- **PRT:**
  - Implementation & Project Coordination
- **AUP:**
  - Documentation & Presentation
- **VNPM:**
  - Project Management

The category “Implementation & Project Coordination” is further split into the following subcategories:

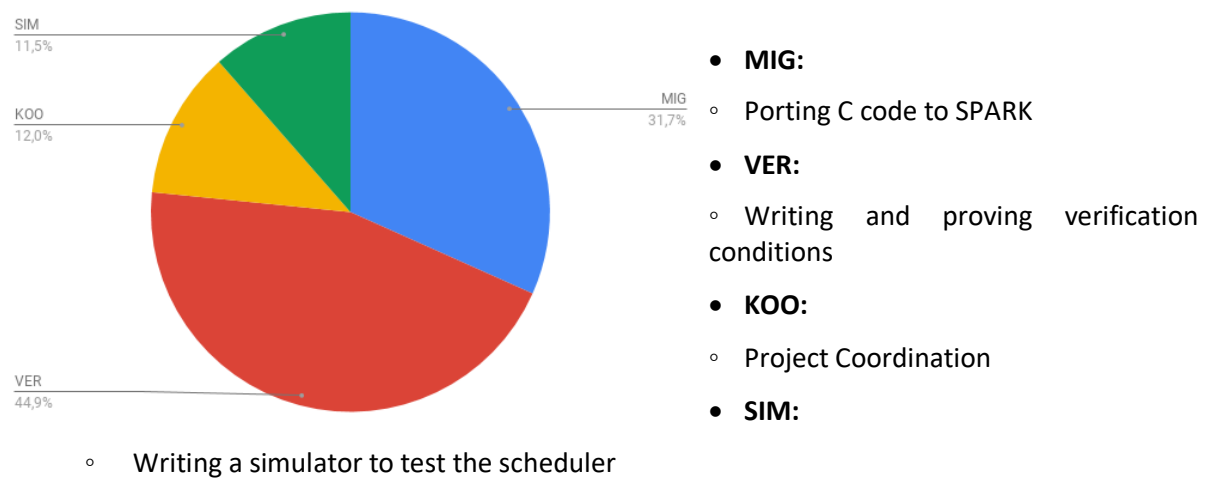


Figure 4-33: Distribution of efforts within implementation and project coordination

Further, the implementation and verification efforts were analysed based on the lines of code of the respective action and compared with the implementation and verification efforts of the seL4 [6] microkernel:

#### HSRM Efforts

C Code	cloc	Code Lines	Assertions	Verification per loc
sched.c	252	193	59	
sched.h	22	21	1	
test_sched.c	90	0	46	
<b>C Total</b>	<b>364</b>	<b>214</b>	<b>106</b>	<b>0.50</b>

SPARK Code	cloc	Code Lines	Verification Conditions	Verification per loc
sched.ads	317	273	44	
sched.adb	201	23	178	
sched_verification.ads	108	0	108	
<b>SPARK Total</b>	<b>626</b>	<b>296</b>	<b>330</b>	<b>1.11</b>

	Implementation	Verification
<b>Overall Effort</b>	62.25 h	95.25 h
<b>Effort per loc of scheduler</b>	~ 4.7 min	~ 19.31 min



The C code was a good starting point since it already had many assertions which were the basis for many verification conditions. The C code had 0.5 assertions per loc and the SPARK code 1.11 verification conditions per loc. Based on the documented efforts it took the student on average 4.7 minutes to implement a line of code and about 17.3 minutes to verify a line of code.

#### seL4 Efforts

	Haskell / C loc	Isabelle loc	Invariants	Proof loc
<b>Abstract spec.</b>	-	4,900	~ 75	110,000
<b>Executable spec.</b>	5,700	13,000	~ 80	55,000
<b>C implementation</b>	8,700	15,000	0	

For the seL4 microkernel, a different approach was used for the implementation and verification of the kernel. An Abstract spec was implemented and verified in Isabelle/HOL and is the basis of the verification. Further an executable spec was written in Haskell and verified in Isabelle/HOL and lastly, the Haskell code was transformed into C code. The following efforts can be found in [Klein] and were converted into min/loc.

- Abstract spec. ~ 4 person months (460 PH)
- Haskell prototype ~ 2 person years (3,520 PH)
- Executable spec. ~ 3 person months (480 PH)
- ⇒ **Verification ~ 29 person months (4,640 PH) ~ 32 min/loc**
- C implementation ~ 2 person months (320 PH) ~ **2.43 min/loc**

#### 4.12.4 Lessons Learned

The results with respect to the expected increase in development performance were satisfactory, especially since the student was unfamiliar with the methodology, SPARK 2014 and formal verification in general. This is also seen in the large amount of effort the initial training took. The student attempted to learn Ada first before looking at SPARK. According to his own assessment, this was unnecessary and only learning the SPARK 2014 subset would have been sufficient for this task. It has been shown that the critical microkernel component which was the subject of this study correctly implements its specification. Whether this increased assurance implies better safety or security of the code depends on the correctness of the specification. To this end, the SPARK based development process can be helpful by providing hints in cases of inconsistencies at the specification level, as these typically make verification harder, if not impossible. Therefore, in future work, the actual steps taken to transform a program to make it more suitable for verification will be examined more closely.

#### 4.12.5 Further Developments

The method will be used to implement the remaining modules of the C prototype and the implementation and verification efforts will be benchmarked. Furthermore, the microkernel's throughput as well as real-time capabilities will be evaluated to determine the effect of the development process on computational performance.

Furthermore, a comparison between verification and testing efforts will be conducted. This will require the development of test cases for verified components, while benchmarking the process.

### 4.13 Combined Model-Based Testing for Multiple Concerns (ATM Use Case Example)

Contributor: AIT

Combined Model-Based Testing for Multiple Concerns brings together functional, safety, security and performance aspects in verification.

The method is applied to UC1 Air Traffic Management, which provides an infrastructure for unmanned aerial vehicles (UAVs) to collect and share position data of UAVs and normal air traffic, with the purpose to avoid accidents and maintain no-fly zones. The use case comprises a client part in the UAVs and a ground based server part.

At this time, only the concept is presented, application to UC1 has just started. While the combined testing approaches share artefacts and tooling, they can in part be applied independently and it is not yet clear which parts will be fully applied within the project to the selected use case.

#### 4.13.1 Aim

The combined testing approach uses analysis outcomes for verification and validation and takes advantage of synergies between approaches to verify the different quality attributes. This in particular includes reuse of results of modelling efforts spent earlier in the project as much as possible.

The combined method itself is in principle applicable on system level down to software unit level. Where it is applied depends on:

- a. the amount of modelling effort that can be spent and the modelling detail achieved
- b. if the requirements and target properties out of the analysis steps are sufficiently detailed and applicable to that level (unit, component, sub-system or system)

The techniques combined in the method to address the different concerns are:

- Functional testing
- Robustness/Security testing
- Invariants (= "safety") checking of the behaviour model
- Invariants (= "safety") monitoring of the implementation
- Performance monitoring of the implementation
- Performance testing according to expectable loads
- Stress Testing
- Implementation performance predictions
- Implementation performance prediction validation

Within the ATM use case, a main goal will be evaluating the performance behaviour under expectable loads and if there are emerging effects on the system for a high number of participating UAVs.

### 4.13.2 Method

Figure 4-34 shows an overview of the different techniques comprising the method, their respective steps and how they play together. The descriptions below refer to the different artefacts and process steps of the diagram in *Cursive* letters.

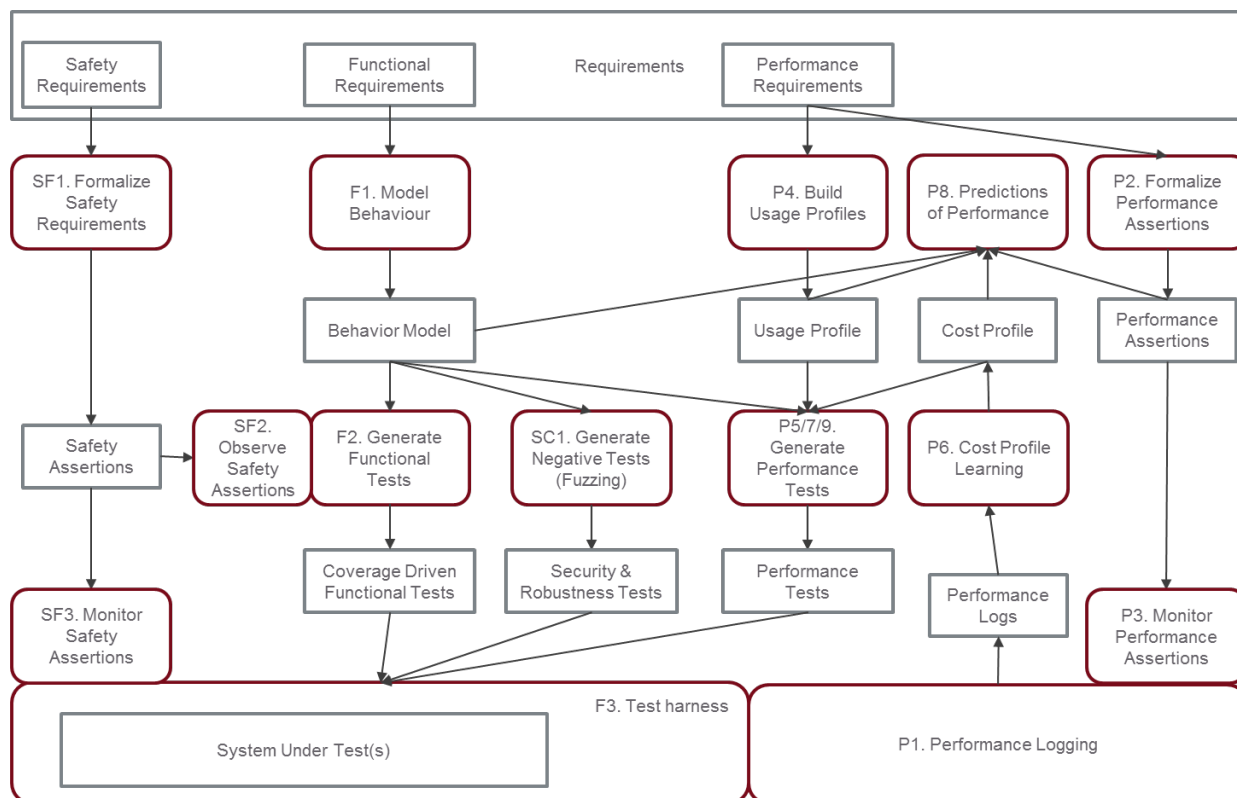


Figure 4-34: Method overview – combined model-based testing for multiple concerns

#### 4.13.2.1 Functional testing

In step *F1. Model Behaviour*, a *Behaviour Model* is built, formalizing the *Functional Requirements* - in our case into UML state machine diagrams. From this model, functional tests can be generated (step *F2.*). As generation strategy, either random walks or a coverage driven approach can be used. As coverage metric, the used tool offers model mutation coverage, which allows one to emulate several other coverage metrics, if needed. [Mutation-TCG] gives details on the mutation driven test case generation and how it is implemented.

The generated tests are sequences of stimuli (controllable events, things the tester can influence) and reactions (observable events, things the tester can observe and verify). Both types of events need to be part of the test interface expressed in the model.

The *Test Harness* (step *F3.*) takes those sequences, applies the stimuli to the system under test and compares the systems reactions to the expected/allowed events in the test. Usually, the Test Harness also needs to translate between the abstract events in the model and the concrete events the System Under Test.

#### 4.13.2.2 Robustness/Security testing

The *Behaviour Model* used for functional testing can also be used to *Generate Negative Tests* (step SC1.). This is done using Smart Fuzzing, where a protocol or behaviour model is explored, and unexpected inputs are randomly added to the allowed stimuli.

The tests are generated in a way, that these inputs are expected to be silently ignored and do not affect the observable behaviour of the system. In an alternative mode, after each unexpected input, an anonymous error handling response is inserted into the test. The *Test Harness* then needs to check if system responses after an unexpected input fall into a defined set of error handling responses and are not any unwanted behaviour.

This second mode can help to keep standardized handling of erroneous inputs out of the model, but more complex fault handling, that for example rolls back the inner system state to stay safe, needs to be modelled explicitly, if it shall be tested.

Since the tests continue after the unexpected input with positive tests, problems like dead locks should be implicitly recognized by the test harness, without implementing additional test oracles. In case there are unwanted conditions that could be provoked by fuzzing, but are neither affecting the expected responses nor the *Safety Assertions*, nor the *Performance Assertions* (both see further below), this would need an additional explicit test oracle to be implemented in the *Test Harness*.

#### 4.13.2.3 Invariants/Safety checking of the behaviour model

In case there are given *Safety Requirements* or other conditions that need to hold during the operation of the system, possibly also some contracts giving pre- and post-conditions for the behaviour, they are formalized into *Safety Assertions* in step SF1. *Formalize Safety Requirements*. The conditions can be expressions over the test interface events only or they can also contain internal variables.

During test case generation, these assertions can be checked by the test case generator in step SF2. *Observe Safety Assertions*. This is much weaker than full classical model checking. The test case generator tries to avoid exploring the full state space for performance reasons and hence can give no guarantees. In case a breadth-first-search strategy is used by the test case generator, the given guarantees are similar to bounded model checking. If a model checker for the formalism of the model is available and its use is feasible for the size of the model, this shall be preferably used.

#### 4.13.2.4 Invariants/Safety monitoring of the implementation

Given that either the *Safety Assertions* are only expressions over the events in the test interface, or the used internal state variables of the *System Under Test* can be observed in the *Test Harness*, step SF3 *Monitor Safety Assertions* can become part of the *Test Harness*. It can check that the modelled *Safety Requirements* hold during all tests – functional (random and coverage driven), robustness, and performance driven.

#### 4.13.2.5 Performance monitoring of the implementation

Based on a performance instrumentation of the test harness (P1. *Performance Logging*) and *Performance Requirements* that lead in step P2. *Formalize Performance Requirements to Performance Assertions*, these *Performance Requirements* can be monitored during testing as part of the *Test Harness* in P3. *Monitor Performance Assertions*.

The *Performance Requirements* can be either upper or lower thresholds for parameters that can be measured on the System Under Test (memory, processor load, bandwidths use, heat dissipation, ...) or response times for certain actions in the system.

This monitoring can be active during all types of tests – functional tests, robustness tests and dedicated performance tests (see below).

#### 4.13.2.6 Performance testing according to expectable loads

If in step *P4. Build Usage Profiles*, *Usage Profiles* can be derived from the *Performance Requirements*, they can be used in step *P5. Generate Performance Tests* to provide tests that mimic the expectable load of the system. This is done by steering random walks (see 4.13.2.1 Functional testing) with the probability distributions given for the input events and possibly their parameter values.

The actually achieved performance is checked in *P3. Monitor Performance Assertions*.

#### 4.13.2.7 Stress Testing

The *Performance Logs* recorded during testing or normal system operation can be condensed into *Cost Profiles* in *P6. Cost Profile Learning*. This uses forms of regression learning and relates the “performance cost” to the behaviour model.

In step *P7. Generate Performance Tests*, using the *Cost Profile*, *Performance Tests* can be generated that try to put stress on the system for each type of performance measurement, individually or combined.

Of course, while running such stress tests, not only the *Performance Assertions* can be monitored, but also behaviour that changes or *Safety Assertions* that do not hold under stress can be identified.

#### 4.13.2.8 Implementation performance predictions

With the *Cost Profile*, the *Usage Profile* and the *Behaviour Model* available, in step *P8., Predictions of Performance* can be made using methods like Statistical Model Checking. The result is a probability distribution of a certain performance hypothesis to hold. The chosen hypothesis is usually related to the *Performance Assertions*.

#### 4.13.2.9 Implementation performance prediction validation

Since the *Cost Profile* is only based on a sample of the behaviour and could be wrong, the performance predictions need to be validated on the *System Under Test*. Using hypothesis testing, the number of tests that need to be run on the *System Under Test* to demonstrate that the hypothesis holds, can be kept substantially smaller than the number of simulation runs done in the model. Step *P9. Generate Performance Tests* produces such a small test suite.

### 4.13.3 Results

As the approach builds on models, and behaviour modelling for UC1 is still ongoing, no results can be presented yet.

The current UC1 model contains several interacting state machines, modelling behaviour of several interacting actors in the system. As of now, as can be seen in Figure 4-35, the model expresses movement between various states but does not contain effects that would be observable on the outside, which is needed to decide if an implementation of this shows the same behaviour.

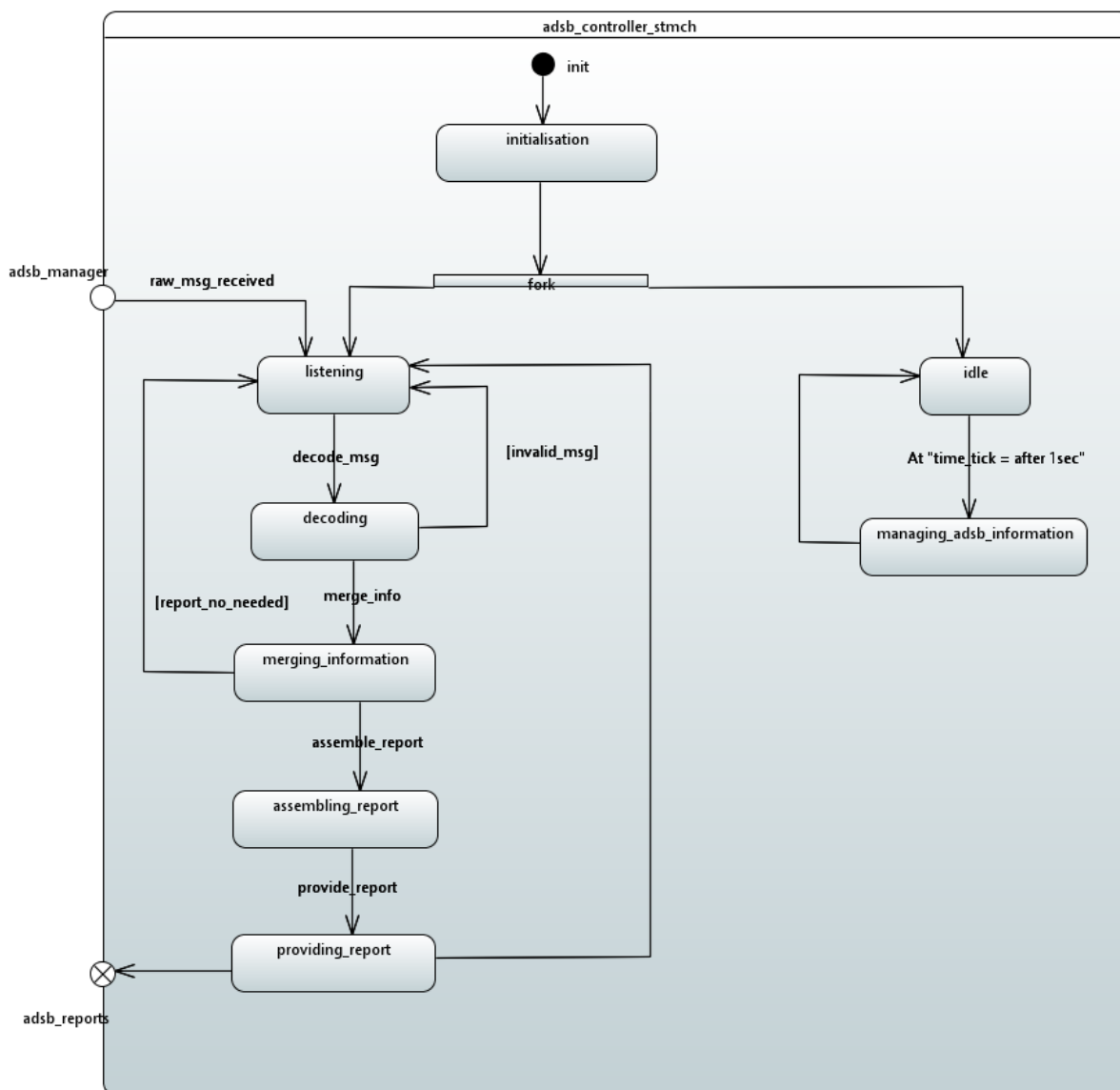


Figure 4-35: Example state machine from UC1

#### 4.13.4 Lessons Learned

While the approach aims at keeping additional efforts for modelling low, the model built without having testing in mind was - not a completely unexpected surprise – not directly usable. The focus on understanding a concept allows one to leave out details that would be needed for an “almost-executable” model, which again builds the basis for the described method. Experience from other projects shows that adding a notion of a test interface and a clear definition of borders of the system under test are not only indispensable for any form of test case generation, but implicitly contribute to better testability of the architecture and design. We expect the same to happen when we start formalizing performance criteria.

#### 4.13.5 Further Developments

Only part of the sketched method is already implemented and available in tools that play well together. The plan for AQUAS is to finish adding the described performance testing features (4.13.2.5-4.13.2.9 inspired by [SMC-response-times]) to the test case generator MoMuT, which already included random and coverage driven functional testing and fuzzing as features. To apply the method to the demonstrator, there is additional work to be done on the tooling, regarding a changed definition of the test interface in the model and completing the feature to check safety properties during test case generation. For application in the demonstrator, also several things need to be added to the *Test Harness*, in particular to measure and collect performance data.

## 5 Interaction Point Planning in the Use Cases

This chapter contains an update on the planning of the IPs that will occur during the part of each use case development that is included in the AQUAS project. The description format introduced in D3.1 has been retained as suitable for these descriptions; but the plans have been revised in view of experience with the combined analyses and updated estimates of effort required.

### 5.1 IP Plan for the The ATM Use Case (UC1)

As reported in deliverable D2.3.1, the PLC of ATM use case is based on the V-model (see Figure 5-1) that mainly represents the current baseline for Integrasys that is/has been used to develop applications internally (without AQUAS Co-Engineering methodology). Figure 5-1 shows the allocation of each partner involved in the use case within this PLC.

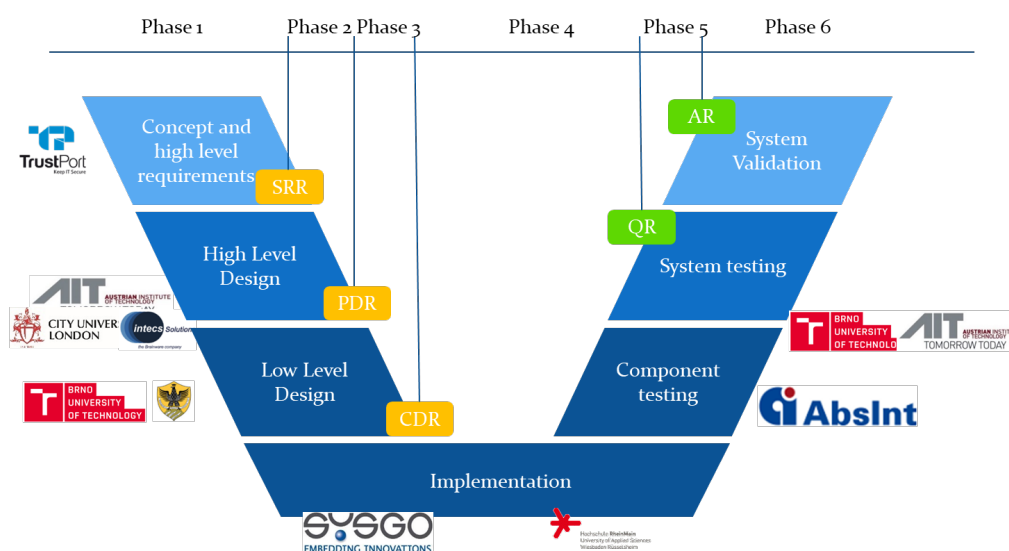


Figure 5-1: PLC for Use Case 1, with interaction points

This model allows us to have a rigorous development lifecycle where we can identify the next interaction points during the process that currently is carried out:

- SSR (Software Specification Review). The SSR is a technical assessment establishing the software requirements baseline of the system in order to ensure the preliminary design.
- PDR (Preliminary Design Review). The PDR is a technical assessment establishing the physically allocated baseline to ensure that the system has a reasonable expectation of being judged operationally effective and suitable.
- CDR (Critical Design Review). The CDR is a technical assessment establishing the build baseline to ensure that the system has a reasonable expectation of being judged operationally effective and suitable.
- QR (Qualification Review). The QR is a technical assessment in order to ensure that the integrated software is tested to provide evidence for compliance with the software requirements.



- AR (Acceptance Review). The AR is a technical assessment in order to verify the completeness of the specific end products in relation to their expected maturity level and assesses compliance to stakeholder expectations.

Furthermore, in the lifecycle we can differentiate the following time phases:

- **Phase 1:** definition of the idea and concept, project context, user needs and high level requirements (finishes with SRR assessment).
- **Phase 2:** preliminary (high level) design with main trade-offs identified and resolved, mapping of high level requirements in low-level requirements and selection of preliminary technologies and platforms (finishes with PDR assessment).
- **Phase 3:** final (low level) design, including possible changes in the design (e.g. using simulation tools or prototyped) and confirmation of technologies and platforms (finishes with CDR assessment).
- **Phase 4:** implementation and integration (finishes with QR assessment on pre-production). It includes implementation and testing according to the implementation in a pre-production environment. This stage can be divided into two sub-stages:
  - Implementation and unitary testing (4a).
  - Integration and testing at system level (4b).
- **Phase 5:** this stage covers the validation and deployment in production environment (finish with the AR assessment).
- **Phase 6:** maintenance.

The next table shows the interaction points defined for this use case taking into account that we are located on the phase 4 (implementation) and that another combined analysis could be included in this interaction points in order to solve potential security/safety/performance conflicts that could come out in next phases:

Table 5-1: Interaction Points of ATM use case.

Interaction Point Identifier	Interaction Point Informal Description	Combined Analysis	Who	On the basis of which information (artefacts)	Attributes studied	Producing what kind of results (outputs)	Gaps	Comments
<b>Interaction Point 1</b> (IAP_01)	<b>Type:</b> Discussions. <b>PLC Placement:</b> High Level Design. <b>Purpose:</b> high-level requirements consolidation first set of low-level requirements generated including identification of SSP controls and PDR approval. <b>Activities:</b> Modelling and analysing possible interferences and conflict points.	Cross-check of system-targets and low-level requirements using tools data and involving expert staff discussions.	Integrasys , Intecs, Trusport	System high level targets CON-OPS and preliminary architecture Environment characterization & restrictions Preferred technologies High level CHES model, SSDLC and Medini (TBC).	Sa/Se/P	A consensuated set of low-level requirements, a first functional implementation design and identification of specific safety/performance/security controls needed.  Re-write some of the targets if needed.		Safety objectives pursuing absence of i) software safety violations (reaching forbidden states), ii) deadlocks (where the program seems to stop running and stops responding to events) and iii) live-locks (where the program cycles endlessly but can't make progress).
<b>Interaction Point 2</b> (IAP_02)	<b>Type:</b> Simulations, Prototyping and Discussions. <b>PLC Placement:</b> Low Level Design <b>Purpose:</b> Final low level requirements established for the client-side including SSP controls and CDR aproval <b>Activities:</b> Modelling and analysing possible interferences and conflict points.	a) Client-side reliability, schedulability and deployment (Safety/Security/Performance) b) DDS client-side security configuration (Security/Performance). c) Mobile network selection strategy (Security/Performance).	Integrasys (a,b,c) Intecs, UNIVAQ, HSRM, BUT (a) Trustport , City (b)	Preliminary (Non-) Functional low level requirements including SSP controls. a) Low-level CHES model, HEPSYCODE/ANACONDA results. b) SAN, sDDS & SW prototyping. c) SW prototyping.	Sa/Se/P	A consensuated low level design of the UC1 client-side (including platform deployment) with agreed trade-offs of performance, security and safety. Re-write some of the requirements if needed.		CHES models to be reused by AIT in the verification phase.

<b>Interaction Point 3</b> (IAP_03)	<b>Type:</b> Simulations, Prototyping and Discussions. <b>PLC Placement:</b> Low Level Design. <b>Purpose:</b> Final low level requirements established for the server-side including SSP controls and CDR approval. <b>Activities:</b> Modelling and analysing possible interferences and conflict points.	a) DDS server-side analysis (Safety/Security) b) DDS server-side analysis (Performance/Security)	City, Trustport, BUT, Intecs (TBC)	Preliminary (Non-) Functional low level requirements including SSP controls.  SAN model and DDS implementations with traffic fuzzing.	Say/Se/P	A consensuated low level design of the UC1 server-side with agreed trade-offs of performance, security and safety.  Re-write some of the requirements if needed.	Intecs may provide support for easier and accelerated SAN simulation.
<b>Interaction Point 4</b> (IAP_04)	<b>Type:</b> Implementation-based assessment. <b>PLC Placement:</b> Verification. <b>Purpose:</b> Verify client-side implementation under realistic performance/safety/security threats. <b>Activities:</b> Testing, results assessment, requirements checking, pre-QR approval.	Component unit-level verification Overall software schedulability & dependability (simulated DDS server) LTE specific tests.	Integrasys (overall test responsibility) AIT, BUT, City (TBC) providing specific test support.	Implementation artefacts Testing artefacts (MoMuT, ANACONDA and SAN).	Sa/Se/P	Test results traceable to established low-level and high-level requirements.	During implementation, SYSGO/ABSINT tooling will be used but no formal IP is defined at that phase. City may provide support to test DDS service under attack. A full QR approval would need a full-system testing (client and server side) which is out of scope.

## 5.2 IP Plan for the Medical Devices Use Case (UC2)

The medical device use case is about developing a device for closed-loop control of patient blood pressure and neuromuscular transmission by extending a monitor product, already in widespread use, with control algorithms to directly control infusion pumps. The part of the PLC taking place during AQUAS is the development of core functions. How this process differs from a stylised V-model PLC was discussed in D3.1, section 4.6.2 (for details on the UC and demonstrator, see also D2.3.2).

By the end of AQUAS, the state of development is planned to be a prototype including full implementation of blood pressure and NMT control, with a “hardware in the loop” (HiL) test setup, suitable for testing (now under way for the blood pressure control) to demonstrate that the product is safe for lab testing with patients. In addition, it is expected that some of the improvements to human interface (for safety) and security aspects identified in IP1, intended for the final version for clinical use, will be implemented.

The planned IPs have thus evolved slightly, with no change to IP1, which concerns the stage of requirements and concept design and is mostly completed; while IP2 is envisaged to specify the verification activities for the complete product, meant to demonstrate achievement of all required SSP properties. Although the complete verification itself will be completed after AQUAS, on the full prototype for clinical use, parts of it are piloted in AQUAS, starting with the mentioned testing of the control algorithm for robustness.

Table 5-2 : Interaction Points of Medical use case

Interaction Point Identifier	Interaction Point Informal Description	Combined Analysis	Who	is expected to do What	When	On the basis of which information (artefacts)	Attributes studied	Producing what kind of results (outputs)	Comments and Known Gaps	Relates to Tools
Interaction Point 1 (IP1)	<p><b>Type:</b> Risk analysis from SSP viewpoints and refinement or requirements/high-level design</p> <p><b>PLC Placement:</b> Requirements/ Conceptual Design Phase</p> <p><b>Purpose:</b> early validation of SSP requirements coming from Requirements stage, identify new SSP requirements (e.g. according to the introduction of mitigation solutions), check feasibility of updated set of requirements, properly feed the implementation phase, give model-based support to be able to possibly trigger a trade-off meeting</p>	Hazard and operability (HAZOP) analysis. Identification of hazards and their likelihood/severity to drive any needed changes to requirements, inform decisions regarding design, and/or trigger further specialist/combined analyses	Led by City and Involving: RGB, Trustport, ITI, CEA, Tecnia, All4Tec, AMT	<p><b>Activity:</b> HAZOP analysis</p> <p><b>Method:</b> Systematically applying a series of guidewords to each step in a use scenario to identify potential deviations of the system behaviour from the design intent</p>	In the Requirements / Conceptual Design Phase after RGB has provided a preliminary description of the system and its requirements	Use scenarios, requirements, and system description provided by RGB	Sa, Se, Pe	List of hazards/feared events and potentially changes to system requirements/design, and/or triggers of further specialist/combined analyses	Trial of the HAZOP analysis was limited to the specific use scenario(s) chosen	
		Hazard analysis and risk assessment (HARA) and Threat analysis and risk assessment (TARA). To identify threats and attacks that potentially lead to hazards and perform a risk assessment based on the HAZOP analysis and design models, asset identification is performed and used to automatically derive threats and potential attack scenarios that are later assessed	Led by AMT but involving UC2 partners	<p><b>Activity:</b> HARA and TARA</p> <p><b>Method:</b> Partners' HAZOP analysis on Process view is imported into medini analyze; TOFCuff system design modelled in medini via SysML. This model is used to identify assets and their security attributes to automatically derive potential threats by applying STRIDE categories. Threats are assessed and treatments defined. From the HAZOP, attacks can be derived that are linked to corresponding threats.</p>	In the Requirements / Conceptual Design Phase when RGB has provided a preliminary design of the system and its requirements, and after the HAZOP analysis	System specification from RGB and output of the HAZOP analysis performed on a process level	Sa, Se	A list of hazards, a collection of threats and attacks that are related in a cause and effect relationship, a risk assessment and treatment on the threat level		Medini analyze
		Fault tree analysis	All4Tec, Intecs,	<p><b>Activity:</b> Fault-tree analysis</p>	In the Requirements	Use scenarios, requirements, and	Sa, Se, Pe	Critical paths that can lead to patient		Safety Architect,

			and Tecniaia	(FTA) <b>Method:</b> Creating fault tree using feared events identified in HAZOP. the FT is analysed for fault propagation, critical paths, etc. and further analysed using Tecniaia's concept-aware analysis tool to identify triggers of co-engineering meetings and reports on evolution	/ Conceptual Design Phase and after the HAZOP analysis	system description provided by RGB along with the feared events identified in the HAZOP analysis		harm. To inform mitigation solutions which can change system requirements and design. Also highlights important test cases (useful for IP2) and potential conflicts to be checked at later stages.		Cyber Architect, CHES, and Concept-aware analysis tool
		Authentication trade-off analysis: clarify trade-offs that arise from a novel security requirement by describing the risk associated with each alternative design solution, to support rational choice of an authentication method	Trustport, City and RGB	<b>Activity:</b> Trade-off analysis <b>Method:</b> authentication methods are described from different viewpoints for pre-selection; analysis then extended via dependency diagrams and comparative tables, and potentially to quantitative analysis.	In the Requirements / Conceptual Design Phase - triggered by the HAZOP analysis	Requirements and system description provided by RGB, as well as literature describing authentication methods and their effectiveness in various contexts	Sa, Se, Pe	Description of the risk associated with each alternative authentication method from a variety of viewpoints (safety, security, performance, usability, cost, etc.) so that designers can make informed decisions		
Interaction Point 2 (IP2)	<b>Type:</b> Detailed specification of testing and verification activities  <b>PLC Placement:</b> Testing/Verification Phase	Static code analysis, To verify the absence of undefined behaviours which lead to potential safety risks and security vulnerabilities.	CEA	<b>Activity:</b> Static code analysis <b>Method:</b> Static code analysis to guarantee, through formal proofs, that the C source code complies with the initial abstract specifications	In the Testing/ Verification Phase and after the code and traceability information is provided by RGB	C code, as well as traceability information linking code items and low-level code properties to the high-level system requirements (all provided by RGB)	Sa, Sec, Pe	Success/Failure documenting potential conflicts arising from implementation, and thus also changes decided to resolve any conflicts.	Scope is defined by RGB, identifying target parts of code.	Frama-C
	<b>Purpose:</b> To produce a verification plan	Preliminary verification to test robustness of the BP control algorithm,	RGB, BUT, ITI	<b>Activity:</b> System verification of BP control algorithm <b>Method:</b>	In the Testing/Verification Phase.	Medical consultations by BUT with partner medical centre in	Sa	Success/Failures of the control algorithm, which may trigger further	Initially limited to control of blood pressure, as in	A2K testing platform, patient

	that adequately covers the SSP requirements for the new device  Note: a subset of the verification activities will be within the AQUAS timeframe	especially to extremes in the patient sensitivity spectrum in preparation for clinical trial, and to demonstrate the functionality and use of the testing platform for future testing phases		Monte Carlo generation of different patient characteristics to test whether the control algorithm achieves given patient parameter targets. Noise/ perturbations applied to test robustness of the algorithm.	Currently underway	the Czech Republic to inform realistic and clinically meaningful parameters of patient sensitivity		developmental improvements. Also, expected outputs include: the range of patient parameters for which the algorithm is expected to successfully operate; information regarding the functionality of the testing platform for future testing.	current prototype and patient model. Testing concepts later applied to testing neuromuscular transmission control.	model defined in C-code, and test hardware: arm simulator, monitor, infusion pump tree, and BP control algorithm prototype
		Definition of test plans : define a set of test cases that address the various viewpoints (SSP and usability) and are traceable to the system requirements in order to verify the system.	RGB, BUT, City, Trustport, All4Tec, Tecnalia	<b>Activity:</b> System verification of the BP control algorithm <b>Method:</b> Each specialist defines test cases that address their specific viewpoint/specialty (e.g.: City re human factors exceptions, Trustport re security requirements, Tecnalia re requirements from medical standards, All4Tec re critical paths identified in the FTA in IP1, etc.). A sufficient, feasible test case plan is assembled that avoids duplications	In the Testing/ Verification Phase	System requirements provided by RGB, mitigation solutions introduced into the system design, and critical paths identified in the FTA in IP1	Sa, Sec, Pe	Sufficient, feasible and efficient set of test cases that is traceable to the system requirements.	AQUAS testing will only address capabilities of existing prototype. So, parts of the test cases defined will be for use in verification phases after AQUAS on complete system.	OpenCert, Concept-aware analysis tool, A2K testing platform, patient model defined in C, test hardware (arm simulator, monitor, infusion pump tree, and BP control algorithm prototype )

		Design of assurance case	Tecnalia co-ordinating all partners involved in IP2	<b>Activity:</b> Preparation of assurance case <b>Method:</b> Define how outputs of the various analyses specified in IP2 serve towards the assurance case.	In the Testing/Verification Phase. Preliminary to executing the various testing/verification activities on completed systems	Requirements as refined by IP1, preliminary specs of various verification/testing activities	Sa, Sec, Pe	Indications of elements required in the various verification/testing plans. Outline of assurance case and organization of the evidence used to support it.	Within the timeframe of AQUAS, the assurance case is planned but not completed; but contributes to establishing a sound V&V plan	OpenCert and Concept-aware analysis tool
Further IPs				most likely outside AQUAS timescale						



### 5.3 IP Plan for the Industrial Drive Use Case (UC4)

The lifecycle applied is a standard V-cycle as shown in Figure 5-2

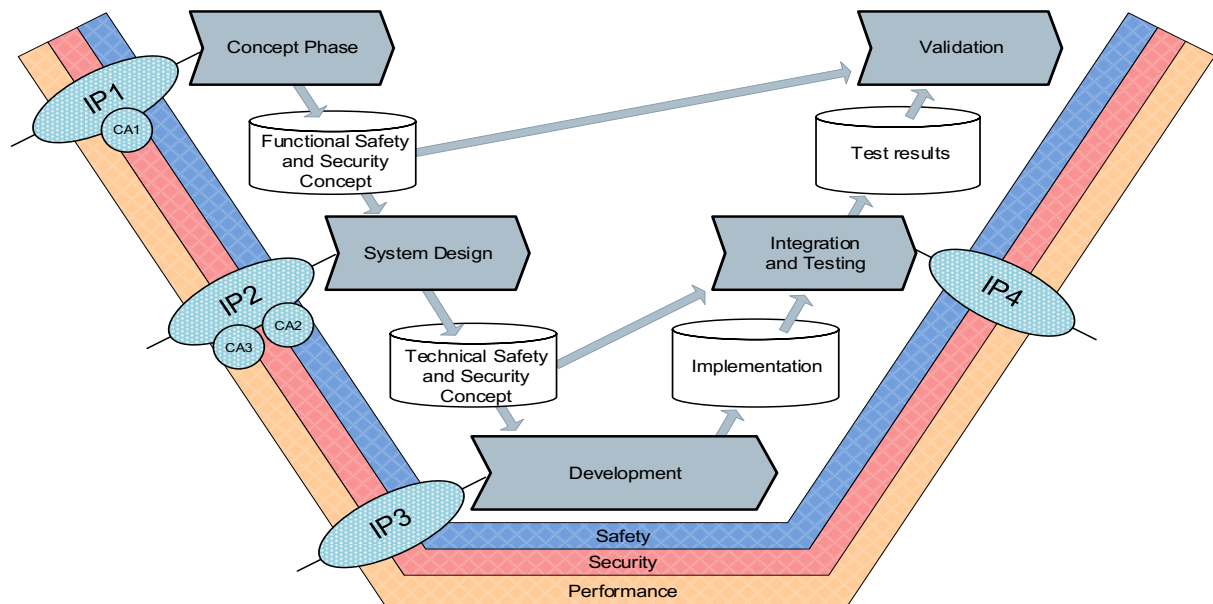


Figure 5-2: PLC for the Industrial Drive use case, with interaction points

The planned IPs are shown in Table 5-3.

Table 5-3: Interaction Points of Industrial Drive use case.

Interaction Point Identifier	Combined Analysis Identifier	Interaction Point Informal Description	Who	is expected to do What	When	On the basis of which information (artefacts)	Attributes studied	Background Info	Producing what kind of results (outputs)	Relates to Tools
Interaction Point 1 (IP1)	Combined Analysis 1 (CA1)	<b>Type:</b> Simulation-based system assessment <b>PLC Placement:</b> Concept Phase <b>Purpose:</b> Validation of current set of Sa/Se/Pe requirements against the actual system design. <b>Activities:</b> Modeling and simulation based on the current system description, followed by validation of the actual safety, security, performance requirements. Results are recommendations for requirements/system design for trade-off decisions.	CITY	<b>Activity:</b> Modeling and simulation of the current system design information for validating the current set of requirements <b>Method:</b> Moebius SAN modeling and simulation	In the Concept Phase when the interference analysis is finished	D2.2.4 Demonstrator Architecture, SaR, SeR, PeR from interference analysis	Sa, Se, Pe	D2.1.4 Domain Environment, "System Architecture - Additional Information.docx" (Additional Inputs for Partners in the Concept Phase).	Recommendations on requirements/system design (they are e.g. based on the system reliability results from simulation with Moebius) - this information is used for trade-off decisions. [table, prose]	Moebius (Stochastic Automata Networks)
Interaction Point 2 (IP2)	Combined Analysis 2 (CA2)	<b>Type:</b> Simulation-based system assessment <b>PLC Placement:</b> Design Phase <b>Purpose:</b> Safety/Security analysis <b>Activities:</b> Modeling	CITY/ SAG	<b>Activity:</b> The implications of security on safety are analyzed by simulating attacks on several attack interfaces in the design such as: Forged messages	Design Phase	D2.2.4 Demonstrator Architecture, D2.3.4, SaR, SeR, PeR from interference analysis (Concept Phase)	Sa, Se	D2.1.4 Domain Environment, "System Architecture - Additional Information.docx" (Additional Inputs for Partners in the	Recommendations on the system design on how safe and secure the system is based on the current design decisions taken [tables, prose]	Moebius (Stochastic Automata Networks)

		and simulation based on the current system description		on the Ethernet between Motor Control Platform and Remote Control Application workstation 2. Forged messages on the links between Motor Control Platform and Motor Power Board <b>Method:</b> Moebius SAN modeling and simulation				Concept Phase), D2.2.4, D2.3.4	[system reliability, availability, etc.]	
	Combined Analysis 3 (CA3)	<b>Type:</b> Simulation-based system assessment <b>PLC Placement:</b> Design Phase <b>Purpose:</b> Security/Performance analysis <b>Activities:</b> Modeling and simulation based on the current system description	TP/MTTP	<b>Activity:</b> Security recommendations from TP's SSDLC are taken into account for modeling the system in TTool (MTTP). Different security algorithms for confidentiality (AES) are tried out and the performance of the system is checked. <b>Method:</b> Modeling and simulation with TTool/SSDLC	Design Phase	D2.2.4 Demonstrator Architecture, D2.3.4, SaR, SeR, PeR from interference analysis, Interference analysis database from Concept Phase	Se, Pe	D2.1.4 Domain Environment, "System Architecture - Additional Information.docx" (Additional Inputs for Partners in the Concept Phase), D2.2.4, D2.3.4	Recommendations on the system design on performance for different security measures.	TTool, SSDLC
Interaction Point 3 (IP3)	most likely out of scope	<b>PLC Placement:</b>								

	for AQUAS	Development Phase								
Interaction Point 4 (IP4)	most likely out of scope for AQUAS	<b>PLC Placement:</b> Integration and Testing Phase								

## 5.4 IP Plan for the Space Multicore Architectures Use Case (UC5)

The planned IPs for this use case are shown in Table 5-4.

Table 5-4 Interaction Points of Space use case.

Interaction Point Identifier	Combined Analysis Identifier	Interaction Point Informal Description	Who	is expected to do What	When	On the basis of which information (artefacts)	Attributes studied	Background Info	Producing what kind of results (outputs)	Relates to Tools
IP_Cph_1 (IP1)	Combined Analysis 1 (CA1)	<p><b>PLC Placement:</b> Concept Phase (selection of the methodology has to be done in the concept phase)</p> <p><b>Purpose:</b> Methodology to develop safe and secure system software by applying formal methods.</p> <p><b>Activities:</b> Application of formal methods to development of System Software (e.g. OS kernel)</p>	HSRM	<p>Activity: Discuss about</p> <ul style="list-style-type: none"> <li>- safety/security integrity levels achievable with or without formal methods</li> <li>- impact of formal methods use on performance</li> <li>- impact of formal methods use on development effort</li> <li>- tradeoff between application of formal methods vs. rigorous testing/run-time checking</li> </ul> <p>Method: Formal verification</p>	After Requirements baseline and before architecture baseline.	<ul style="list-style-type: none"> <li>* Set of system software functional requirements (derived from use case application functional model)</li> <li>* Set of system software timing requirements (derived from use case application timing model)</li> <li>* Set of system software security requirements (based on ESA requirements and/or best practices in related domains, e.g. avionics)</li> </ul>	Sa, Se, Pe	D2.1.5	<p><b>ARTIFACT 1:</b> name: report of affected modules (input to system architecture model) input to IP: IP_Dsph_1</p> <p><b>ARTIFACT 2:</b> name: verified source code with verification conditions and proofs. input to IP: IP_Dvph_1 / IP_Dsph_1.</p> <p><b>ARTIFACT 3:</b> name: unverifiable source code with appropriate test cases input to IP: IP_Dvph_1 / IP_Dsph_1.</p>	(ESA) space worthiness certification requirements expert Security Engineer expert Safety Engineer expert (SPARK) Formal Methods expert System software expert
IP_Dsph_1 (IP2)	Combined Analysis 2 (CA2)	<p><b>PLC Placement:</b> Design Phase</p> <p><b>Purpose:</b> identify interferences due to safety-security barriers in the architecture.</p> <p><b>Activities:</b> Once CHES model has been enhanced with safety-security barriers, a co-engineering meeting for a combined safety-security analysis is held.</p>	INTECS ALL4TEC TECNALIA	<p>Activity: to take decision for the implementation of safety or security barrier in the architecture</p> <p>Method: Co-engineering meeting</p>	On every design iteration (after an architecture baseline)	<ul style="list-style-type: none"> <li>* System "enhanced" architecture (with safety-security barriers)</li> <li>* Set of security requirements</li> <li>* Set of safety requirements</li> <li>* Set of functional requirements</li> </ul>	Sa, Se	D2.1.5	<p><b>ARTIFACT 1:</b> name: Added or changed requirements affected IP:: IP_Cph_2</p> <p><b>ARTIFACT 2:</b> name: Safety/Security trees input to possible IP in validation phase</p> <p><b>ARTIFACT 3:</b> name: (HTML) formal concept analysis input to IP:: TBD</p>	System architecture: CHES Safety analysis: safety engineer with Safety Architect Security analysis: security engineer with Cyber Architect Safety-Security combined analysis: Tecnalia concept-

										aware tool (prototype)
IP_DsDvph_1 (IP3)	Combined Analysis 3 (CA3)	<b>PLC Placement:</b> Design and implementation phases. <b>Purpose:</b> find timing interference on multi-core systems <b>Activities:</b> Two-step timing interference characterization	TRT	Activity: Two-step timing interference characterization (architecture & application)  Method: characterization methodology	Iterate until convergence on safety / security / performance requirements are met.	* Functional architecture description * Hardware architecture description * set of safety requirements * set of security requirements * set of performance requirements	Sa, Se, Pe	D2.1.5	TBD	All4TEC - SafetyArchitecture MTTP - Ttool TRT - Time4Sys/Tempo
IP_DsDvph_2 (IP4)	Combined Analysis 4 (CA4)	<b>PLC Placement:</b> Between design and implementation. <b>Purpose:</b> Safe Scheduling, Safe Code generation and performance analysis <b>Activities:</b> Generation of threaded code modules and analysis of timing and scheduling performance/safety aspects of the overall system. Analysis of generated code for concurrency and memory safety.	ITI BUT ABSINT	Activity: Response time analysis & schedulability. Analysis of code safety and memory usage including shared resources.  Method: safe code generation		System hardware components description (platform model) High level description of application software (tasks, flows, precedence relations) Code modules for each task. Deployment model (task priorities, activation patterns, use of shared resources, software/hardware allocation constraints)	Sa, Pe	D2.1.5	<b>ARTIFACT 1:</b> name: threaded code modules with enhanced safety. input to IP: IP_Dvph_1 <b>ARTIFACT 2:</b> name: performance reports (latencies, throughputs) input to IP: IP_Dsph_1 (e.g if WCET exceed design constraints) <b>ARTIFACT 3:</b> name: Sensitivity results – How “close” system is to becoming un-schedulable. input to IP: IP_Dsph_1	A2K – Art2kitekt / TimingProfiler (provides execution times) BUT ANaConDA & Perun (OSLC Interface) for static code analysis.
IP_Dvph_1 (IP5)	could be in the scope for AQUAS	<b>PLC Placement:</b> Implementation phase. <b>Purpose:</b> New code implementation analysis								
IP_VVph_1 (IP6)	most likely in scope for AQUAS	<b>PLC Placement:</b> Verification and validation phases. <b>Purpose:</b> Verification of design expected results								
IP_Cph_2 (IP7)	could be in the scope for AQUAS	<b>PLC Placement:</b> Concept phase <b>Purpose:</b> Requirements Joint Review								

## 6 Conclusions

This document is delivered at month 24 of the project. Its contents are the basis for the final stage of the AQUAS methodology work. The AQUAS use cases have applied combined analyses on parts of their demonstrator development processes; these examples, documented here, will help all project partners in the final stage of work in the methodology work package. This final stage will involve:

- further application of combined analyses and new interaction points, so as to validate and refine the AQUAS approach; and
- reporting of the results so as to deliver methodology proposals, which will be
  - based on combined analyses and interaction points;
  - supported by the tool developments performed in AQUAS;
  - suitable for adoption by companies and for consideration by standard organisations.

Regarding the potential for broader adoption and standardisation, we note that the 2018 version of standard ISO 26262 (Road vehicles — Functional safety), in Part 2 (Management of functional safety) mandates that as part of "safety culture":

5.4.2.3 The organization shall institute and maintain effective communication channels between functional safety, cybersecurity, and other disciplines that are related to the achievement of functional safety.

EXAMPLE 1 Communication channels between functional safety and cybersecurity in order to exchange relevant information (e.g. in the case it is identified that a cybersecurity issue might violate a safety goal or a safety requirement, or in the case a cybersecurity requirement might compete with a safety requirement)."

AQUAS directly addresses the need identified in this standard, but it also offers concrete solutions. An organisation trying to satisfy the above "normative" statements in ISO 26262 will find in the "informative" annex E of the standard only a basic list of some of the goals for such communication. Instead, the present document describes a set of concrete approaches at the technical level, with preliminary validation of their practical applicability and effectiveness. A single project like AQUAS could never deliver complete assessment of a set of complex methods, which is only possible through larger-scale use. But the reports in this document, and the further experience that will be accumulated in the remaining part of AQUAS, will offer evidence for partners and other users to decide on adopting or piloting both the individual AQUAS techniques and the AQUAS approach.

Chapter 4, which describes the techniques for "combined analyses", documents that the trials of these techniques have been successful: no major obstacles have been encountered in applying them; some lessons have been learned about *how* to apply them; these techniques have helped to detect interdependencies between the SSP requirements and the design decisions driven by them, to trace possible hazards and their causes, etc. The examples documented here are limited in scope, for the sake of readability, but most of them are already being extended to address broader subsets of the demonstrator systems.

Apart from the combined analyses, the present document has added detail to the Interaction Point concept as developed so far. In particular:

- Chapter 2 gives a top-down view of the intended AQUAS improvements to the product life cycle (PLC) and the role played by interaction points. This includes the Conceptual Model of the AQUAS product life cycle, which will help interaction within the project, with a potential for possible adoption in standardization.



- The outline of tooling requirements for interaction points in Chapter 3 supports the work on WP4, Design Tooling.

In the remaining time of the project, the AQUAS Use Cases will move on to the next interaction points in their planning, and thus feed their experience back to improve these tooling requirements and guidance about managing co-engineering via interaction points.

Last, work on this deliverable has contributed to the planning of both the individual partners' activities and the joint use case activity that will support the methodology work in the remaining part of the AQUAS project. These plans are summarized in the "Further Developments" subsections in Chapter 4 "Methods for Combined Analyses" and in Chapter 5 "Interaction Point Planning in the Use Cases". As each use case proceeds through its PLC to subsequent interaction points, this will help AQUAS to further assess not just the effectiveness of analysis techniques but also the way that interaction points should be planned (their placements and the analyses they will include). The lessons learned will be fed into the final reporting from WP3.

## 7 Glossary and abbreviations

Table 7-1: AQUAS-specific terms and AQUAS-specific word uses

Combined analysis	An analysis that combines different viewpoints, e.g. safety and security. Equivalent to Interference analysis
Co-engineering	AQUAS usage: Managing the interactions between system qualities (key ones in AQUAS being safety, security, performance, but also usability). In particular, orchestrating the manual and automatic trade-offs within and across stages of the product lifecycle.
Focus area	see "silo"
Interaction Point	A step in a PLC consisting of running combined analyses from the viewpoints of two or more properties (e.g. security, safety etc), through some mix of automated and human analyses, holding discussions as needed to resolve problems arising, possibly iterating analyses and reaching trade-off decisions. Also, the point in the PLC at which one such activity happens
Interference analysis	Any analysis that addresses more than one non-functional requirements. Also used in UC4 for a preliminary analysis intended to
Silo (or "focus area")	<p>A set of specialist activities and specialists , e.g. a security team in a development project, and their activities.</p> <p><b>Note:</b> In AQUAS, "silo" is used without the negative connotations it may have in business literature (dysfunctional teams, unable to communicate effectively), and does not assume complete lack of communication, but rather communication organised through "interaction points. However, "focus area" has been introduced for contexts where "silo" may be misinterpreted</p>
Work product	AQUAS usage: any item produced during the lifecycle; artefact that is part of a system or used in its PLC, like design documentation or test plans.

Table 7-2: Abbreviations used in the text

ATM	Air Traffic Management
CE	Co-engineering
FMVEA	Failure Modes, Vulnerabilities and Effect Analysis
FT, FTA	Fault Tree, Fault Tree Analysis
HARA	Hazard Analysis and Risk Assessment
HAZOP	HAZard and OPerability analysis
HW	Hardware
IP	Interaction point
MITM	Man In The Middle (attack)
PLC	Product lifecycle
SAE	Society of Automotive Engineers
SAN	Stochastic Activity Network
SSP	Safety, security and performance
SW	Software

TARA	Threat Analysis and Risk Assessment
------	-------------------------------------

## References

- [Arsenault] Arsenault, D., A. Sood, and Y. Huang. Secure, resilient computing clusters: Self-cleansing intrusion tolerance with hardware enforced security (SCIT/HES). in 2nd International Conference on Availability, Reliability and Security. 2007. Los Alamitos, CA: IEEE Computer Society Press.
- [Bellavista] Bellavista, P., et al. Data Distribution Service (DDS): A performance comparison of OpenSplice and RTI implementations. 2013. Split, Croatia: IEEE.
- [Ciabrone] D. Ciabrone, V. Muttillio, L. Pomante and G. Valente, "HEPSIM: An ESL HW/SW co-simulator/analysis tool for heterogeneous parallel embedded systems," 2018 7th Mediterranean Conference on Embedded Computing (MECO), Budva, 2018, pp. 1-6.
- [HEAVENS] HEAVENS security models (March, 18 2016), [http://autosec.se/wp-content/uploads/2018/03/HEAVENS\\_D2\\_v2.0.pdf](http://autosec.se/wp-content/uploads/2018/03/HEAVENS_D2_v2.0.pdf)
- [Hepsycode] Hepsycode: A System-Level Methodology for HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems, <http://www.hepsycode.com>
- [Klein] seL4: Formal Verification of an OS Kernel, Gerwin Klein et al.
- [Mutation-TCG] Andreas Fellner, Willibald Krenn, Rupert Schlick, Thorsten Tarrach, and Georg Weissenbacher. 2019. Model-based, Mutation-driven Test-case Generation Via Heuristic-guided Branching Search. ACM Trans. Embed. Comput. Syst. 18, 1, Article 4 (January 2019), 28 pages. DOI: <https://doi.org/10.1145/3289256>
- [Muttillio] V. Muttillio, G. Valente and L. Pomante, "Design Space Exploration for Mixed-Criticality Embedded Systems Considering Hypervisor-Based SW Partitions," 2018 21st Euromicro Conference on Digital System Design (DSD), Prague, 2018, pp. 740-744.
- [Popov, 2015] Popov, P.T. Stochastic Modeling of Safety and Security of the e-Motor, an ASIL-D Device. in Computer Safety, Reliability, and Security. 2015. Cham: Springer International Publishing.
- [Popov, 2017] Popov, P., Models of reliability of fault-tolerant software under cyber-attacks in The 28th IEEE International Symposium on Software Reliability Engineering (ISSRE'2017). 2017, IEEE: Toulouse, France. p. 228-239. Available from: <https://doi.org/10.1109/ISSRE.2017.23>.
- [SEBoK] BKCASE Editorial Board. 2017. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.9.1 R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed DATE. [www.sebokwiki.org](http://www.sebokwiki.org). BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.
- [SMC-response-times] B. K. Aichernig et al., 'Learning and statistical model checking of system response times', Software Quality Journal, Jan. 2019, <https://doi.org/10.1007/s11219-018-9432->
- [Sousa] Sousa, P., et al., Highly Available Intrusion-Tolerant Services with Proactive-Reactive Recovery. IEEE Transactions on Parallel and Distributed Systems, 2010. 21(4): p. 452-465.
- [STRIDE] Kohnfelder, Loren; Garg, Praerit (April 1, 1999). The threats to our products. Microsoft Interface.

## List of Annexes

Contain extended details or confidential information related to the examples presented in Chapter 4.

Annex 4-1: HAZOP Table (Medical Use Case)

Annex 4-2: HARA/TARA Analysis (Medical Use Case) (medini analyze)

Annex 4-3: Authentication: Descriptive Trade-Off (Medical Use Case)

Annex 4-4: SANs Model (ATM Use Case)

Annex 4-5: SANs Model (Industrial Drive Use Case)

Annex 4-9: Design-Stage Model (Space Use Case)