

ECSEL Research and Innovation actions (RIA)



AMASS

**Architecture-driven, Multi-concern and Seamless Assurance and
Certification of Cyber-Physical Systems**

**Design of the AMASS tools and methods for
multiconcern assurance (b)
D4.3**

Work Package:	WP4 Multi-Concern Assurance
Dissemination level:	PU = Public
Status:	Final
Date:	30 April 2018
Responsible partner:	Thomas Gruber (AIT)
Contact information:	Thomas.gruber@ait.ac.at
Document reference:	AMASS_D4.3_WP4_AIT_V1.0

PROPRIETARY RIGHTS STATEMENT

This document contains information, which is proprietary to the AMASS Consortium. Permission to reproduce any content for non-commercial purposes is granted, provided that this document and the AMASS project are credited as source.

This deliverable is part of a project that has received funding from the ECSEL JU under grant agreement No 692474. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and from Spain, Czech Republic, Germany, Sweden, Italy, United Kingdom and France.

Contributors

Names	Organisation
Thomas Gruber, Siddhartha Verma, Christoph Schmittner, Sebastian Chlup	AIT Austrian Institute of Technology GmbH (AIT)
Stefano Tonetta, Alberto Debiasi	Fondazione Bruno Kessler (FBK)
Alejandra Ruiz, Estibaliz Amparan, Garazi Juez	Tecnalia Research & Innovation (TEC)
Helmut Martin, Robert Bramberger, Bernhard Winkler	Virtual Vehicle Research Center (VIF)
Irfan Sljivo, Barbara Gallina, Zulqarnain Haider	Maelardalens Hoegskola (MDH)
Stefano Puri	Intecs (INT)
Marc Sango	ALL4TEC (A4T)
Staffan Skogby, Detlef Scholle	ALTEN Sweden (ALT)
Jan Mauersberger	ANSYS medini Technologies AG (KMT)

Reviewers

Names	Organisation
Adedjourna Morayo (Peer-review)	Commisariat a l'énergie atomique et aux Energies Alternatives (CEA)
Siddhartha Verma (Peer-review)	AIT Austrian Institute of Technology GmbH (AIT)
Thomas Gruber (TC review)	AIT Austrian Institute of Technology GmbH (AIT)
Barbara Gallina (TC review)	Maelardalens Hoegskola (MDH)
Cristina Martinez (Quality Manager)	Tecnalia Research & Innovation (TEC)

TABLE OF CONTENTS

Executive Summary.....	8
1. Introduction (*).....	9
1.1 From Monoconcern to Multiconcern (*).....	9
1.2 Scope and Objectives of this Deliverable (*)	9
1.3 Relation to other AMASS Deliverables (*)	10
2. Conceptual Level	11
2.1 System Dependability Co-Analysis / Assessment.....	11
2.1.1 Co-Analysis and Risk Assessment	12
2.1.2 Trade-off Analysis	21
2.1.3 Further development of SiSoPLE for enabling process-related co-assessment (*)	25
2.1.4 Co-assessment for Safety and Security Assurance (*).....	32
2.2 Dependability Assurance Case Modelling	40
2.2.1 Introduction	40
2.2.2 Safety and Security Assurance Case (*)	40
2.2.3 Multiconcern Argumentation	44
2.2.4 Support for variability management at the argumentation level (*)	47
2.3 Multiconcern Contracts	47
2.3.1 Abstract functions in the contracts specification (*).....	48
2.3.2 Contract-based trade-off analysis in parameterized architectures (*)	49
2.3.3 General extensions to contract based multi-concern assurance (*).....	49
2.3.4 Contract-based trade-off analysis with the Analytical Network Process (*)	52
3. Design Level	53
3.1 Functional Architecture for AMASS Multiconcern Assurance	53
3.1.1 Overview.....	53
3.1.2 Dependability Assurance Modelling (*).....	56
3.1.3 Contract-Based Multiconcern Assurance (*)	57
3.1.4 System Dependability Co-Analysis/Assessment (*)	58
3.2 AMASS Multiconcern Assurance Metamodel	62
3.2.1 Elaborations	62
4. Way Forward to the Implementation.....	65
4.1 Potential Tool support.....	65
4.1.1 OpenCert – supports “Dependability Assurance Modelling”	65
4.1.2 CHeSS - supports “Contract-Based Multiconcern Assurance”	66
4.1.3 FMVEA - supports “System Dependability Co-Analysis/Assessment”	66
4.1.4 EPF-Composer - supports “System Dependability Co-analysis and assessment”	67
4.1.5 WEFACT - supports the assurance process workflow	68
4.1.6 Medini Analyze - supports the assurance process workflow (*).....	69
4.1.7 AMASS Farkle - supports product assurance	70
4.1.8 Safety Architect – supports “System Dependability Co-Analysis/Assessment”	72
4.1.9 AMT2.0 – supports “Contract-Based Multiconcern Assurance”	73
4.1.10 Extensions (*)	73
4.1.11 Implemented Multiconcern Assurance Related Requirements (*)	73
5. Conclusions (*).....	76
Abbreviations and Definitions (*)	77
References (*)	80

Appendix A: Changes since the Predecessor Version D4.2 (*)	84
---	-----------

List of Figures

Figure 1. Relationship between Dependability & Security and Attributes, Threats and Means (after [14])	11
Figure 2. Conceptual Overview of the SAHARA method	15
Figure 3. Main steps of FMEA.....	17
Figure 4. Depiction of the relation of cause and effect model for failures and threats	17
Figure 5. Example for hierarchical network structure	22
Figure 6. Unweighted supermatrix of mutual effects between safety and security	23
Figure 7. Limit supermatrix of mutual effects between safety and security	24
Figure 8. Interaction between safety and security engineering.....	29
Figure 9. Work Breakdown structure of process related to verification of system design	30
Figure 10. Process related to verification of system design in WEFACT.....	31
Figure 11. A testbed for fuzz testing of IEC 61850	34
Figure 12. Process of Security breach [77] [78].....	35
Figure 13. Error Model showing erroneous state transition due to security threat event and vulnerability	36
Figure 14. WEFACT user interface	37
Figure 15. Sub processes in WEFACT and the share of those fulfilled.....	37
Figure 16. Example for an automated safety-, security- and performance-verification process.....	38
Figure 17. Concept for safety and security co-analysis by combined process inn WEFACT	39
Figure 18. FMVEA tool architecture.	39
Figure 19. An Assurance Case Fragment.....	41
Figure 20. Assurance Case Structure, argument modules decomposition for Cooperative driving scenario	42
Figure 21. Multiconcern assurance case structure.....	43
Figure 22. GSN Argument Pattern for making multiconcern trade-offs.....	44
Figure 23. The dependency – impact relationship	45
Figure 24. The conflicting-impact relationship.....	46
Figure 25. The supporting-impact relationship.....	46
Figure 26. Assuring different concerns via multiconcern contracts, taken from [65]	48
Figure 27. Capturing interplay of concerns in argument contracts.....	50
Figure 28. Safety case contract argumentation pattern for capturing the conflicting relationships across concern-specific modules	51
Figure 29. Safety case contract argument pattern for dependency relationship	52
Figure 30. The multiconcern assurance process	54
Figure 31. The three WP4 functionalities with explanations	55
Figure 32. Assurance Case Specification.....	56
Figure 33. Dependability Assurance Modelling block.....	57
Figure 34. Process-related Co-assessment	59
Figure 35. System Dependability co-analysis.....	60
Figure 36. The FMVEA architecture	61
Figure 37. Mockup of the FMVEA model editor including properties definition	61
Figure 38. The FMVEA results	62
Figure 39. Relation with other metamodels	63
Figure 40. Contract concern	64
Figure 41. Functional decomposition of the OpenCert platform	65
Figure 42. Basic concept of FMVEA	66
Figure 43. The EPF approach, adapted from [64].....	67
Figure 44. Screenshot WEFACT Version2.....	68

Figure 45. medini analyze overview	69
Figure 46. Cyber security analysis with medini analyze.....	70
Figure 47. System Overview AMASS Farkle tool	72

List of Tables

Table 1.	TARA Method mentioned in SAE J3061	13
Table 2.	Evaluation of TARA method by [25]	14
Table 3.	Classification Examples of Knowledge 'K', Resources 'R', and Threat 'T' Value of Security Threats.....	16
Table 4.	CARE Attack Likelihood Parameter	19
Table 5.	Likelihood categories.....	19
Table 6.	WP4 requirements coverage	73

Executive Summary

This deliverable is the final result of Task 4.2 *Conceptual approach for Multiconcern Assurance*. As an update of the intermediate deliverable D4.2 *Design of the AMASS tools and methods for multiconcern assurance (a)* [2], it provides extensions to the multiconcern assurance features described there in order to cope with the full set of requirements identified in D2.1 *Business cases and high-level requirements* [4].

This document presents the various considerations and a consistent approach to multiconcern assurance at both the concept level and the design level. On the concept level, our multiconcern assurance approach focuses on analysis and risk assessment for assurance, assurance case modelling, and the extension of contract-based approaches for realising safety and security assurance at the same time. On the design level, we focus on how to implement the concept in toolchains and models for seamless and efficient assurance in cyber-physical systems, considering existing work. The implementation details are further elaborated, taking into account existing tools of the AMASS partners.

In this deliverable edition, enhanced methods for multiconcern assurance are presented and the scope is extended from the focus on safety and security in D4.2 towards a wider variety of dependability attributes, in particular in the sections on methods and tools for trade off analysis. The relations to the activities and results in other WPs are pointed out and the AMASS CACM metamodel parts relevant for multiconcern assurance are explained. Finally, a table depicts the coverage of the WP4 related requirements by the methods described here.

In the next step, the results presented in this deliverable will guide the implementation of the third iteration of the AMASS prototype (Task 4.3 *Implementation for Multi-Concern Assurance*), and the resulting implementation will be delivered as D4.6 *Prototype for multiconcern assurance (b)* [5] at the end of month 29.

Finally, Task 4.4 *Methodological Guidance for Multi-Concern Assurance* will build on the results identified here and on the experience in the case studies in order to provide methodological guidance to the AMASS end-users for the application of the multiconcern assurance approaches; this will be documented in D4.8 *Methodological guide for multiconcern assurance (b)* [7] in month 31.

This deliverable represents an update of the *AMASS D4.2 [2] deliverable released at M15*; the sections modified with respect to D4.2 have been marked with (*), then the details about the differences and modifications are provided in Appendix A.

1. Introduction (*)

The AMASS project builds on concepts and tools developed in former projects, in particular in OPENCOS [51] and SafeCer [85]. With respect to including security, ideas and approaches from EMC2 [87], SESAMO [86], MERgE [84] and CONCERTO [83] influence AMASS. More details on which concepts from previous projects were re-used or extended in AMASS can be found in D4.1 *Baseline and requirements for multiconcern assurance* [1].

1.1 From Monoconcern to Multiconcern (*)

In a broad sense, multiconcern assurance is taking a holistic approach to achieve and balance the assurance goals set by different quality attributes such as safety, security, performance, and reliability.

In AMASS, multiconcern assurance is focused on facing five challenges, which, if overcome, will enable multiconcern assurance:

- Dependability Assurance Modelling: Extending the OPENCOS CCL metamodel and vocabulary to include additional dependability related concerns besides safety, and also supporting mappings between concerns (presented in Section 3.1.2).
- Contract-Based Multi-Concern Assurance: Using contracts to support compositional assurance and trade-offs (presented in Section 3.1.3).
- System Dependability Co-Analysis / Co-Assessment: Addressing security issues, which may affect safety, and interrelations between safety and security, considering architecture related issues (presented in Section 3.1.4)
- Looking at the interplay between safety and security in terms of process requirements.
- Investigate security-informed safety-oriented process lines (SiSoPLEs).

For the dependability assurance modelling and, in a narrow sense, multiconcern assurance, the goal is to specify a unified assurance case in which all various quality attributes such as safety and security and their interactions and interplay are clearly specified, such that all presented claims, argumentation, and decisions are connected and traceable.

In a wider sense, it also relates to analysis/assessment and compositional approaches. The safety of a component may depend on a secure environment or a certain level of security. There are, thus, interdependencies between different quality attributes in a reusable component and its environment. Such concerns need to be addressed and solved. In order to identify the need for security and safety and to support trade-off analysis, co-analysis and co-assessments need to be used.

1.2 Scope and Objectives of this Deliverable (*)

This deliverable presents the final design of the multiconcern assurance features: Dependability Assurance Modelling, Contract-Based Multiconcern Assurance, and System Dependability Co-Analysis and Co-Assessment. It builds on the state of the art with respect to multiconcern assurance and the applicable standards presented in D4.1 [1], elaborating the way forward identified there and covering the respective requirements identified in D2.1 *Business cases and high-level requirements* [4]. It must be noted that the result of multiconcern assurance influences model instances which belong to other technical work packages.

Relations to other WPs are pointed out and the AMASS CACM metamodel parts relevant for multiconcern assurance are explained. This deliverable is the final edition of the *Design of the AMASS tools and methods for multiconcern assurance*; it builds on D4.2 [2] and presents extensions to the multiconcern features described there.

1.3 Relation to other AMASS Deliverables (*)

This deliverable is related to other deliverables: deliverables within WP4 as well as deliverables within other work packages.

Within WP4, this deliverable is related to the following deliverables:

It builds on the state of the art in the area of multiconcern assurance and the applicable standards presented in D4.1 [1] and on D4.2 [2], which contains the first iteration of the multiconcern assurance concepts and designs.

The output of the deliverable represents the basis for the iteration (c) of the Integrated AMASS Platform with respect to multiconcern assurance, which will be delivered as D4.6 *Prototype for multiconcern assurance (c)* [5] in August 2018.

Together with D4.1, and with the experience in the implementation gathered in Task 4.3, D4.3 also forms a basis for the guidelines to be developed in Task 4.4, which will be delivered as an updated version D4.8 [7] in October 2018.

(Remark: The deliverable D4.4 [8] for the iteration (a) of the Integrated AMASS Platform was submitted earlier than D4.2, in m10, and contained only the basic building block Assurance Case editor. It was influenced by early conceptual considerations on multiconcern assurance in Task T4.2, but neither D4.2 nor D4.3 was a basis for this early implementation step).

There are moreover relations to deliverables of other technical work packages:

D4.3 receives the WP4-relevant high-level requirements described in D2.1 [2]. It contains the concepts and designs for the implementation of the remaining requirements after some had been implemented in D4.4 and the major part in D4.5 (based on D4.2 concepts and designs).

The evidence as results of individual assurance processes represents the instantiation of the evidence metamodel, which is part of WP5. The result of a trade-off analysis can be used as annotations of the assurance case, which is within the scope of WP4, but they also represent the basis for multiconcern-aware design decisions, which influence the architectural metamodel instantiation in WP3.

2. Conceptual Level

In systems engineering, dependability is a measure of a system's availability, reliability, maintainability, and other attributes such as safety and security. Figure 1 gives an overview about attributes usually associated with dependability, typical threats to dependability, and means for increasing a system's dependability. It shall be noted that AMASS also deals with performance as an additional attribute, which is not included in dependability.

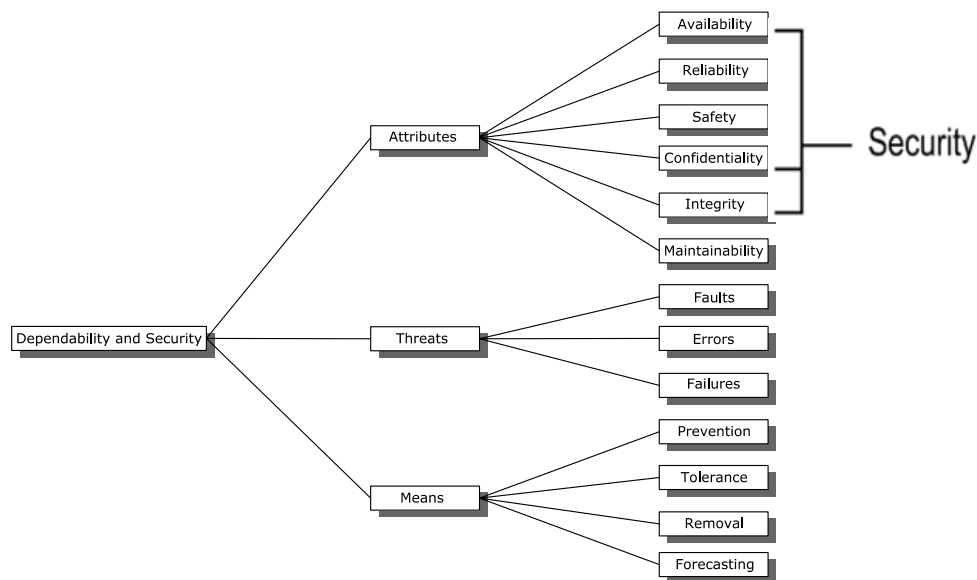


Figure 1. Relationship between Dependability & Security and Attributes, Threats and Means (after [14])

Multiconcern assurance is based on the consideration of dependability attribute during the whole system lifecycle. One of the challenges is that we cannot consider dependability attributes in isolation. Attributes interact and depend on each other. Therefore, co-engineering is necessary for reaching a sufficient level of dependability and balance between different dependability attributes. Co-engineering refers to the interactions between system engineering and the engineering of safety, security and other attributes.

In this chapter, co-engineering is explored and designed. More specifically, system dependability co-analysis and co-assessment are considered in Section 2.1, which provides subsections on co-analysis and risk assessment, on trade-off analysis, on further development of SiSoPLE for enabling process-related co-assessment, and, finally, on co-assessment for safety and security assurance. Then follows section 2.2 dependability assurance case modelling with, after an introduction, sections on the safety and security assurance case and on multiconcern argumentation. Section 2.3 provides information on multiconcern contracts.

2.1 System Dependability Co-Analysis / Assessment

Co-analysis and co-assessment are integral parts of multiconcern assurance. In this last iteration of the “Design of the AMASS tools and methods for multiconcern assurance” document [3], we extend the viewpoint of the predecessor version [2] to more quality attributes than merely safety and security.

In D4.1 [1] Section 4.2.2, we reviewed the state of the art concerning safety & security co-analysis, focusing on model-based approaches. In D4.1 Section 4.2.4, we briefly reviewed safety & security co-assessment in the context of safety & security co-engineering and assurance, focusing on the process of assessment framework.

Within the AMASS project, we distinguish between co-analysis and co-assessment:

- **Co-analysis and risk assessment** refers to the methods, techniques, and activities to identify safety hazards and security threats. For example, Hazard Analysis and Risk Assessment (HARA) and Threat analysis and Risk Assessment (TARA) are established methods enabling co-analysis.
- **Co-assessment** refers to processes, methods, and techniques to evaluate whether a component of a system fulfils its claims that safety and security risks are effectively addressed, such that one can obtain confidence that a system will achieve its dependability objectives (see also section 2.2). We distinguish two inter-related types of assessments:
 - Process-related co-assessment for standard compliance, e.g. the assessment of compliance to IEC 62443-4-1 [52] focuses on the secure development process (see also sections 2.1.3, 3.1.4, and 4.1.4, and D4.1 [1] for other domains).
 - Product-related co-assessment for product-specific safety and security measures, e.g. the assessment of compliance to IEC 62443-4-2 [52] focuses on the product-specific security requirements.

In the AMASS project, we adapt and extend existing co-analysis and co-assessment approaches which contribute to co- or multiconcern assurance. Note that safety & security co-engineering is currently under active development in research, industry, and the standards. Several AMASS partners play an active role on this topic. Some of the methods are mentioned in D4.1 [1]. In this deliverable, we focus on the methods that we deem to be the most promising within the AMASS project.

2.1.1 Co-Analysis and Risk Assessment

Co-analysis covers a wide range of methods and techniques to identify safety hazards and security threats, which are often the activities in the early stage of a product/system development lifecycle, e.g. in the requirements engineering as well as the design phase. These analyses are also regarded as approaches to risk assessment, because the goal of the analyses is often to identify safety and security risks. In the following, this document focuses on methods for those domains which are applied in the AMASS use cases.

In a recent work [23], the authors evaluate several best practice engineering approaches to safety and security, including the methods for systematic risk management and for system validation (risk management, Security-aware Hazard Analysis and Risk Assessment (SAHARA), FMVEA, and Attack Tree Analysis (ATA)) and for comprehensive dependability evidence provisioning (assurance case), especially in the context of ISO 26262 process landscape. While in the context of automotive functional safety the hazard analysis and risk assessment (HARA) method is standardised and mandated by the ISO 26262 standard, several candidates for a cybersecurity threat analysis and risk assessment (TARA) method exist. Some of these methods are mentioned in the SAE J3061 cybersecurity guidebook but there are more of such methods published.

SAE J3061 states on the collection of cybersecurity analysis techniques. “Appendix A - Description of cybersecurity analysis techniques” is provided as a reference to further research and to facilitate design and process improvements. Appendix A is not a comprehensive listing of “Cybersecurity analysis techniques” [24]. An overview and review of available threat analysis methods and their automotive applicability is given in [25]. In particular, this review also includes an analysis of the development phases in which these methods can be sensibly applied. While only a few are suited as TARA for early concept stages, some others have properties which are highly desirable at later development stages. Based on this analysis we selected the sequel of methods described in detail in the following. Notable methods are:

- TARA methods listed in SAE J3061:
 - E-Safety Vehicle Intrusion Protected Applications (EVITA) method [35]
 - Threat, Vulnerabilities, and implementation Risks Analysis (TVRA) [36]
 - Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [34]
 - HEALing Vulnerabilities to ENhance Software Security and Safety (HEAVENS) model [37]

- Attack Tree Analysis (ATA) [30]
- Software Vulnerability Analysis [38]
- TARA methods beyond SAE J3061:
 - Failure mode and Vulnerability Effect Analysis (FMVEA) - Failure mode and failure effect model for safety and security cause-effect analysis [39]
 - Security Aware Hazard Analysis and Risk Assessment (SAHARA) [40]
 - SHIELD method giving guidance for security, privacy and dependability assessment of embedded systems, developed in the European SHIELD project¹
 - Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS) [41]
 - Boolean Logic Driven Markov Processes (BDMP) [42]
 - Threat Matrix [43]
 - Binary Risk Analysis (BRA) [44]
 - STAMP (Systems- Theoretic Accident Model and Processes) Based Process Analysis (STPA-SEC) [45]

D4.1 [1] already outlined a set of safety and security analysis techniques. Some of those methods are further explained in [24] and Table 1 which provides an overview of the TARA methods mentioned and not mentioned in SAE J3061. Also in this case, the overview is taken from Macher et al. 2016 and gives an overview of the different TARA methods mentioned in J3061 (Appendices A-C) [25].

Table 1. TARA Method mentioned in SAE J3061

	Method Name	Applicable Phase	Key facts
SAE J3061 recommended	EVITA method	Concept phase	Outcome of a research project; classification separates different aspects of the consequences of security threats (operational, safety, privacy, and financial). Classification of severity is adopted and thus not conforming to the ISO 26262 standard; classification of safety-related and non-safety-related threats differs and could thus lead to in-balances; accuracy of attack potential measures and expression as probabilities is still an open issue.
	TVRA	---	Models the likelihood and impact of attacks; complex 10 steps approach; developed for data - and telecommunication networks; hardly applicable for cyber physical systems in vehicles.
	OCTAVE	---	This approach is best suited for enterprise information security risk assessments; hardly applicable for cyber physical systems in vehicles; brings together stakeholders thru series of workshops.
	HEAVENS model	System phase	Based on Microsoft's STRIDE approach; determination of threat level (TL), impact level (IL), and security level (SL) for classification of threats; requires a high amount of work to analyse and determine the SL of individual threats; implies lots of discussion potential for each individual factor of each single threat.

¹ <https://www.shield-h2020.eu>

	ATA	System phase	Analogous to fault tree analysis (FTA); identification of threats in a hierarchical manner; adequate for exploiting combinations of threats (attack patterns); requires more details of the system design to be more accurate, requires as prerequisite input identified attack goals.
	SW vulnerability analysis	SW phase	Examines software code to prevent occurrence of potential vulnerabilities; focuses on SW development level.

Table 2 gives an overview of the TARA methods not mentioned in SAE J3061. Also in this case, the overview is borrowed from Macher et al. 2016 [25].

Table 2. Evaluation of TARA method by [25]

	Method Name	Applicable Phase	Key facts
not in SAE J3061	FMVEA	System phase	Based on the FMEA; identify threat modes (via e.g. STRIDE model) for each component/function of the system, identify system level effects and risks, categorise risks via quantification of attacker effort, system properties for attack likelihood and threat effects.
	SAHARA	Concept phase	Threat analysis via STRIDE model; security and safety analysis possible in a combined and independent manner; easy quantification scheme; no adaptation of standardised quantification scheme for safety; requires less analysis efforts and details of the analysed system.
	SHIELD	System phase	Evaluates multiple system configurations; only evaluates system's security, privacy and dependability level; implies a high discussion potential for each classification, due to the lack of guidance on how to estimate the security, privacy, and dependability values.
	CHASSIS	Concept phase	Combined safety and security assessments; relies on modelling of misuse cases and misuse sequence diagrams; implies additional modelling expenses for the early development phase; structures the harm information in the form of HAZOP tables and in combination with the BDMP technique.
	BDMP	System phase	Based on ATA and FTA; fault tree and attack tree analysis are combined and extended with temporal connections.
	Threat Matrix	System phase	Proposed by US Department of Transportation; used to consolidate threat data; threat matrix is spreadsheet based; variation of the FMEA approach; geared towards the establishment of a threat database; not a preferable approach for concept analysis.
	BRA	Concept phase	Threat impact determination via 10 yes/no questions; quick risk conversations to enable discussion of a specific risk; not a full risk management methodology; quantitative analysis not based on statistics or monetary values; not a threat discovery or threat risk assessment technique on its own.
	STPA-SEC	---	Control model based analysis, originally developed for safety and later extended for security. A mixture of a system engineering approach and analysis technique, compatibility with ISO 26262 lifecycle still in discussion, modelling based on control loops which can mask security relevant issues.

2.1.1.1 SAHARA as co-analysis method

In the context of the AMASS project, the following methods represent a reference for co-analysis.

The SAHARA method [40] combines the automotive hazard analysis and risk assessment (HARA) with the security domain STRIDE approach to quantify impacts of security threats and safety hazards on system concepts at initial concept phase. STRIDE is a threat modelling approach and an acronym for Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privileges. The key concept of the STRIDE approach is the systematic analysis of system components for susceptibility to threats and mitigation of all threats to enable argumentation of a certain security of the system.

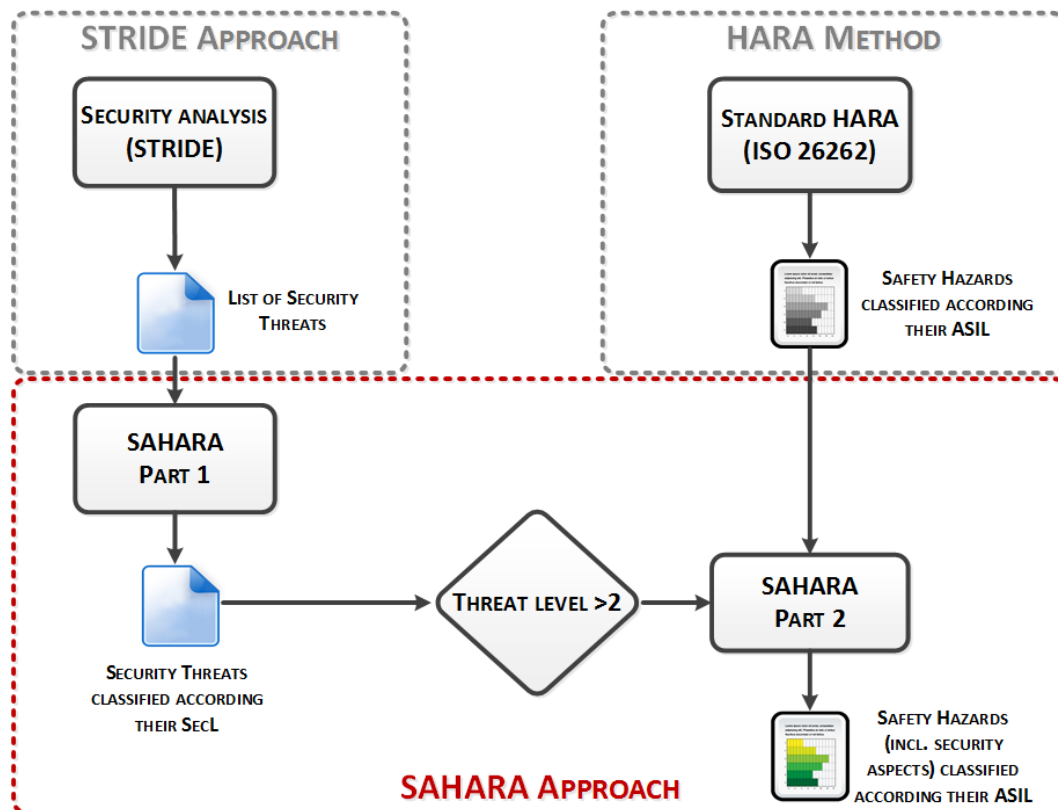


Figure 2. Conceptual Overview of the SAHARA method

Figure 2 shows the conceptual overview of the SAHARA method and coupling of the safety and security analysis methods involved. For the initial stage, ISO 26262 confirms HARA analysis (see the right side of Figure 2) can be performed in a conventional manner. This means that the functions provided by the system are analysed for their possible malfunction (hazards) and the worst possible situation in which this malfunction may happen. The hazard and situation combinations (hazardous event) are analysed and quantified according to the ISO 26262 standard regarding their severity (S) and controllability (C) by the driver in the event of an occurrence. Further, the frequency and duration of exposure (E) in which this hazardous situation may occur is quantified. These factors (S, C, and E) determine the automotive safety integrity level (ASIL), the central metric for determination of development efforts required for the rest of the development process.

The security-focused analysis of possible attack vectors of the system can be done independently by specialists of the security domain (see the left side of Figure 2). For this analysis, the STRIDE threat model approach is used to expose security design flaws of the system design by methodically reviewing the system design. This is done in five steps: 1) the identification of security objectives; 2) a survey of the application; 3) the decomposition of the application; 4) the identification of threats; and 5) the identification of vulnerabilities. This threat modelling approach does not prove a given design secure but helps to learn from mistakes and avoid repeating them. The two loosely coupled analysis steps (security

analysis and safety analysis) can either be performed by individual teams or in cooperation with safety and security experts.

Table 3. Classification Examples of Knowledge 'K', Resources 'R', and Threat 'T' Value of Security Threats

Level	Required Knowledge (K) Classification	Required Resources (R) Classification	Threat Criticality (T) Classification
0	Unknown internals (black-box approach)	No tools required	No impact
1	Basic understanding of internals (grey-box approach)	Standard tools	Annoying, partial reduced service
2	Internals disclosed (white-box approach)	Non-standard tools	Damage of goods, privacy intrusion
3		Advanced tools	Life-threatening possible, maximum security impact

After this identification of possible security threats and safety hazards, the SAHARA method combines the outcomes of the security analysis with the outcomes of the safety analysis. The ASIL concept of the safety analysis is thus adopted and applied to the security analysis outcomes. In order to quantify the security level (SecL) of a threat, the required knowledge (K) and resources (R) to pose the threat, as well as the impact of the successful attack (T), are estimated (cf. Table 3). The factor T also implies impacts on human life (quality of life) as well as possible impacts on safety features. This information on security threats that may lead to a violation of safety goals is passed on for further safety analysis (depicted as SAHARA part 2 in Figure 2).

The **required know-how - 'K'** - is classified as: Level 0 - no prior knowledge required (the equivalent of black-box approach). Level 1 - covers persons with technical skills and basic understanding of internals (representing the equivalent of grey-box approaches). Level 2 – represents white-box approaches, persons with focused interests and domain knowledge.

Required resources - 'R' - to threaten the system's security are classified as: Level 0 - threats not requiring any tools at all or an everyday commodity, available even in unprepared situations. Level 1 - tools that can be found in any average household. Level 2 - availability of these tools is more limited (such as special workshops). Level 3 - are advanced tools whose accessibility is very limited and are not widespread.

The **criticality of the successful attack - 'T'** - is classified as: Level 0 – indicates a security irrelevant impact. Level 1 - is limited to annoying, possibly reduced availability of services. Level 2 - implying damage of goods or manipulation of data or services. Level 3 – represents the highest criticality (affecting car fleets) and also implies impacts on human life (quality of life) as well as possible impacts on safety features.

In general, the SAHARA quantification scheme is less complex and requires fewer analysis efforts and details of the analysed system than other available approaches. The quantification of required know-how and tools can also be seen as equivalent to a likelihood estimation of an attack to be carried out. Nevertheless, this quantification provides the possibility to determine limits on the resources spent in preventing the system from being vulnerable to a specific threat (risk management for security threats) and the quantification of the threat impact on safety goals (threat level 3) or its non-impact on them (all others). Moreover, a combined review of the safety analysis by security and safety experts can also help to improve the completeness of security analysis. Bringing together and combining the different mind-sets and engineering approaches of safety engineers and security engineers, who are able to work independently from one another and also mutually benefit from each other's findings, is a fruitful approach that is likely to achieve higher analysis maturity standards.

2.1.1.2 FMVEA as co-analysis method

The FMVEA Method [27] was developed in the context of the ARROWHEAD project and extends the established Failure Mode and Effect Analysis with security related threat modes.

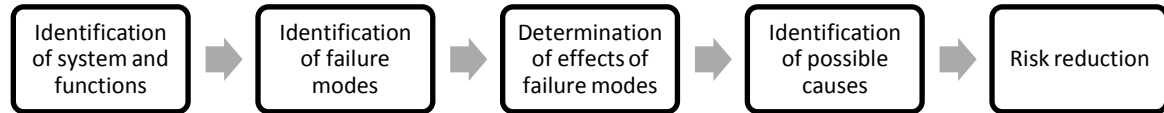


Figure 3. Main steps of FMEA

Figure 3 gives an overview of the main steps for the standard FMEA. A system is modelled and divided into parts and all the potential failure modes are identified for each part. Depending on the detail level, parts can be process steps, functions, system architecture elements or software/hardware parts. All system effects are identified for each potential failure mode and the severity is evaluated. For all failure modes with a critical severity, potential failure causes and their likelihood are evaluated and the criticality is calculated.

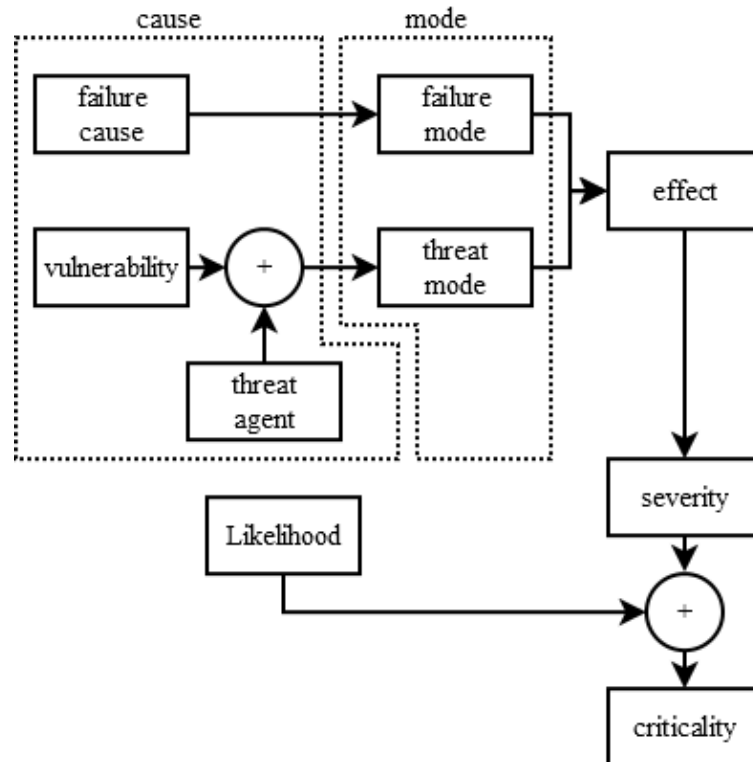


Figure 4. Depiction of the relation of cause and effect model for failures and threats

Figure 4 gives an overview of the cause and effect model for the Failure Modes, Vulnerabilities and Effects Analysis. The failure part consists, as before, of failure cause, failure mode, and effect. Security related parts are added here, including vulnerability, threat agent, threat mode and effect. Depending on the level of analysis a vulnerability can be an architectural weakness or a known software vulnerability. Compared to safety, security requires not only a weakness but also an element, which is exploiting this weakness. This can be a software or a human attacker. Different threat modelling concepts can be used for the identification of threat modes such as CIA (confidentiality, integrity, availability), summarizing security properties an attack could exploit, or also STRIDE. Based on the severity of the effect, measured in terms of financial damage, loss of confidentiality or privacy and operational or safety impact and the likelihood of the failure or threat the criticality is measured. In the likelihood context, the system properties and attacker properties should be investigated.

Existing databases and domain knowledge can be used for identifying potential failure modes. Since the challenge of security for the automotive domain has emerged relatively recently, there is less knowledge about the threat modes than is the case in some other fields and domains. The analysis is based on a system model, depicting network architecture and data flows. In the practice, we currently use threat modelling to identify and analyse threat modes for each element of the system model. The main steps involved in a threat modelling process include:

1. Model a system by drawing the system architecture in data-flow diagram (DFD), adding system details to the elements in the DFD, and draw the trust boundaries.
2. Identify threats stemming from data flows by using a threat identification methodology such as the STRIDE or CIA method [28]. An assessment of the severity of the threats can be added.
3. Address each threat by redesigning the system, adding mitigation, or ignoring it if the risk is acceptable.
4. Validate the threat modelling diagram against actual system and all identified threats are addressed.

A DFD diagram consists of five types of elements: process, data store, data flow, external interactor, and trust boundary. A process is a software component that takes input and performs actions and/or generates output. In a DFD, a process can be modelled in different levels of granularity. If necessary, a high-level process can be decomposed into more detailed low-level processes in a hierarchical manner. For example, if we start to model all software components of a “Head Unit” at Level 0, we can further decompose it into processes of “Communication Gateway”, “Linux OS”, “Applications”, and “HMI” at a lower level for Level 1. Depending on the available system details and threat identification needs, a process can be further decomposed into lower-level components such as specific Linux kernel modules.

Further to this, a data store in the DFD represents a firmware, file system, or memory. A data flow in the DFD is a directed arrow, representing the flow of data between two elements. For example, a data flow can be a protocol specific communication link such as CAN Bus, FlexRay, or HTTPs. An external interactor is either a human user or a user agent that interacts with a process from the outside. Trust boundaries divide the elements in the diagram into different trust zones, e.g. elements reside in the in-car systems and external hosts communicated from untrusted open networks. The assumptions on the trust boundary greatly influence the result of threat identification. A data flow originated outside the trust boundary is assumed to be untrustworthy by default such that additional verification or security controls should be applied.

When identifying threats, different methodologies can be applied. STRIDE is a popular methodology due to its easy-for-developer origin and extensive documentation of applications. However, depending on the granularity of the system information available and the timing of the threat modelling in the development lifecycle, alternative methodologies can also be used for optimal cost-benefit results. For example, the enumeration of potential attacks on each of the elements in a brainstorming session by domain experts will already improve the security posture of the design at the concept phase. Mitigations are technical or organizational countermeasures corresponding to the identified threats. The linking of mitigations to the threats ensures that all identified threats will be considered and addressed, and puts mitigations into perspective with the overall security architecture. Threat modelling is essentially a theoretical model of perceived threats to a system. Validating the theoretical model against the actual system will ensure the correctness of the results from the threat modelling. Validating that all identified threats are addressed provides additional layer of quality control on the security activities in the development process. Depending on the level of details for the failure modes either data from past events or generic failure modes can be used.

For the rating of severity, the FMVEA can either determine the severity directly or use information from previously conducted analysis such as e.g. SAHARA. Since FMVEA requires at least a basic system architecture more information for the rating of likelihood are available, like more detailed potential attack surfaces and weaknesses.

Table 4. CARE Attack Likelihood Parameter

Parameter	Values			
Capabilities	Amateur (3)	Mechanic, Repair shop (2)	Hacker, Automotive expert (1)	Expert team from multiple domains (0)
Availability of Information	Information publicly available (3)	Information available for maintenance of for customer / operator (2)	Information available for production, OEM, system integrator (1)	Information available in company of ECU supplier (0)
Reachability	Always accessible via untrusted networks (3)	Accessible via private networks or part time accessible via untrusted networks(2)	Part time accessible via private networks or easily accessible via physical (1)	Only accessible via physical (0)
Required Equipment	Publically available standard IT devices / SW ² (3)	Publically available specialised IT devices / SW ³ (2)	Tailor-made / proprietary IT devices / SW ⁴ (1)	Multiple Tailor-made / proprietary IT devices / SW (0)

Table 4 shows a likelihood rating system, which differs between the four factors:

- necessary capabilities of the attacker
- availability of information about the attacked systems
- reachability of the attacked systems
- required equipment for the easiest identified attack.

Ratings for all categories are added up and assigned to one of five Likelihood categories (Table 5).

Table 5. Likelihood categories

Range	0-2			3-5			6-8			9-11			>11
Category	Improbable			Remote			Occasional			Probable			Frequent
Values	0	1	2	3	4	5	6	7	8	9	10	11	>11

This was done to be consistent with the five likelihood categories presented in IEC 60812, Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA). The result is a Likelihood Rating from Improbable to Frequent.

2.1.1.3 ATA used in co-analysis

The Fault Tree Analysis (FTA) is widely known as a state of the art methodology to analyse systems and subsystems in the context of the functional safety of systems. It is a deductive failure analysis, meaning that a known failure mode or undesired state is decomposed into a quantity of lower level events. By doing so, a tree of events and their logical combinations is constructed, giving in-depth information about the occurrence of the investigated top-level failure mode.

² Readily available equipment, as example simple OBD diagnostics devices, common IT devices such as notebooks.

³ Equipment that is obtainable with little effort, as for example computing power from a cloud provider, in-vehicle communication devices (e.g., CAN cards), or costly workshop diagnosis devices.

⁴ Equipment that is not readily available, because it is either proprietary or custom made.

The fault tree analysis is a quantitative analysis, as each event or logical gate may be assigned a statistical probability. Subsystem failures occur at a failure rate λ and the logical combination with other subsystem failures leads to a quantified occurrence of the top level failure mode. This further leads to a better understanding of the system under investigation, especially when this system is integrated into a larger system-of-systems or is part of a distributed cyber-physical system. In the automotive domain, where complex multi-level integrator-supplier relationships exist, the FTA is requested by many standards (ISO 26262, IEC 61508) and is therefore state-of-the-art.

As tackled in D4.1 [1] Section 4.2.2.3, the concept of the FTA in the field of functional safety is also applicable to the field of security. This allows capturing malicious risks on an extended fault tree. In this case, the top level event expresses the occurrence of a security related incident of the system under investigation. At the lower levels potential attacks are logically combined aggregating information about the top level event.

Since the late 1990s a methodology evolved which uses structured data to identify threats to computer systems. While the so called attack tree analysis (ATA) was first applied within the domain of computer networks [30], it constantly evolved and was applied to other system categories, e.g. Supervisory Control and Data Acquisition (SCADA) systems. Conducting an ATA provides several advantages, compared to different methods. The ATA helps to understand what potential attack goals are, who the attackers are, what attacks are most likely to occur, which security assumptions apply to a given system, and finally, which investments regarding countermeasures are considered most effective.

Attackers may have different motivations, and opportunity crimes typically require less effort than well-planned operations. The kind of access to the system available to the attacker also plays a large role. Different unique skills may also be required by an attacker. The risk aversion of the attacker may heavily influence the attack execution. Acceptance of certain risks (e.g. publicity, jail time, death) leads to totally different attacks. Finally, a lack of all of these points may be compensated through the availability of appropriate funding. Attack trees help to describe the security of systems under investigation by building kind of knowledge databases. They are also a way to capture expertise, and make this knowledge available for future re-use, speeding up decisions and increasing their transparency.

Attack trees are basically data trees, where the root node represents an attack goal. An attack goal represents the violation of a security property, such as confidentiality or authenticity. The subordinate leaf-nodes represent attacks, targeting their linked attack goal. Multiple attack trees aiming at different attack goals may exist in parallel for complex systems. In this case, common attacks, which are relevant to multiple attack goals, are of special interest. When developing an attack tree, logical expressions are used to relate different applicable attacks to each other. A logical OR gate represents alternatives for attacking a system, whereas an AND gate determines attacks which are only successful in combination with each other.

Each leaf node of an attack tree may be assigned Boolean properties, e.g. to indicate the feasibility of an attack. The options in this case are “possible” and “impossible”. Depending on the tree data structure, known system properties, or implemented security measures, certain tree branches may become irrelevant during analysis, as these properties are propagated up the tree. In contrast to Boolean properties, continuous node values may be assigned to leaf nodes. Typical examples are cost, time, or resource estimations, as these help to quantify the probability of occurrence of attack scenarios.

Attack trees provide valuable information to safety- and security-engineers. The consideration of Boolean properties and continuous node values within a single analysis allows complex tree evaluations, e.g. to “determine the best possible attack which costs €1000 or less”. To determine if and which dedicated countermeasures against certain threats are taken, thresholds and guidelines are necessary to evaluate the selected metrics. From the automotive integrator’s perspective, assumptions are also subject to inclusion within an attack tree. A comprehensive list of assumptions, resulting from e.g. requirements, may influence security decisions based on attack trees.

An attack tree is built in three steps:

1. Identify the attack goals.
2. Identify attacks against each goal, repeat as necessary.
3. Re-use patterns of attacks for re-usable components.

If attack trees for a given system have reached a mature state, impact analyses give information on how a system modification affects the selected metrics. The value of an attack goal thus needs to be calculated as described. Following on from this first step, changes are applied to the system and new leaf nodes or even a new attack must be introduced. The tree is subject to upward modification as necessary. Finally, the new attack goal values are calculated and compared to the previous ones. This approach may also be used to compare and rank different attacks to the system under investigation.

2.1.2 Trade-off Analysis

Trade-off analysis deals with the attempt to satisfy requirements with respect to different competing quality attributes with the goal of finding a balanced set of mitigation measures for the design resulting in a “multiconcern-aware” architecture.

Several publications of Despotou et al. ([57], [58] and [59]) draft approaches called FANDA and TOM for assessing design alternatives and facilitating trade-offs in critical systems; they were discussed in D4.1 [1]. While FANDA and TOM aim at facilitating the dependability (or assurance) case, the approach presented here focuses on satisfying non-functional system requirements with respect to different quality attributes by modelling both security-related attacks and safety-related failures in a common scheme in order to find an optimised architecture and design.

The basic idea for Trade-off-Analysis presented in this subsection is to use *Analytical Network Process* (ANP) [66] to analyse the impact of failures and cyber-attacks on overall safety and security of the given system, and use this information as basis for system modification. ANP basically helps in integrating and analysing information obtained from several sources.

As for system architecture, we propose an application of the ANP which results in something similar to *FMEA*, where we divide the system into components and for each component we analyse failure and threat modes. Next, we need to analyse how these components interact at subsystem level i.e. derive the subsystem failure / attack rate and how these failures / attacks affect the system safety and security. A hierarchical structure of the system (with networks because of cross domain or intra domain dependability) can be obtained.

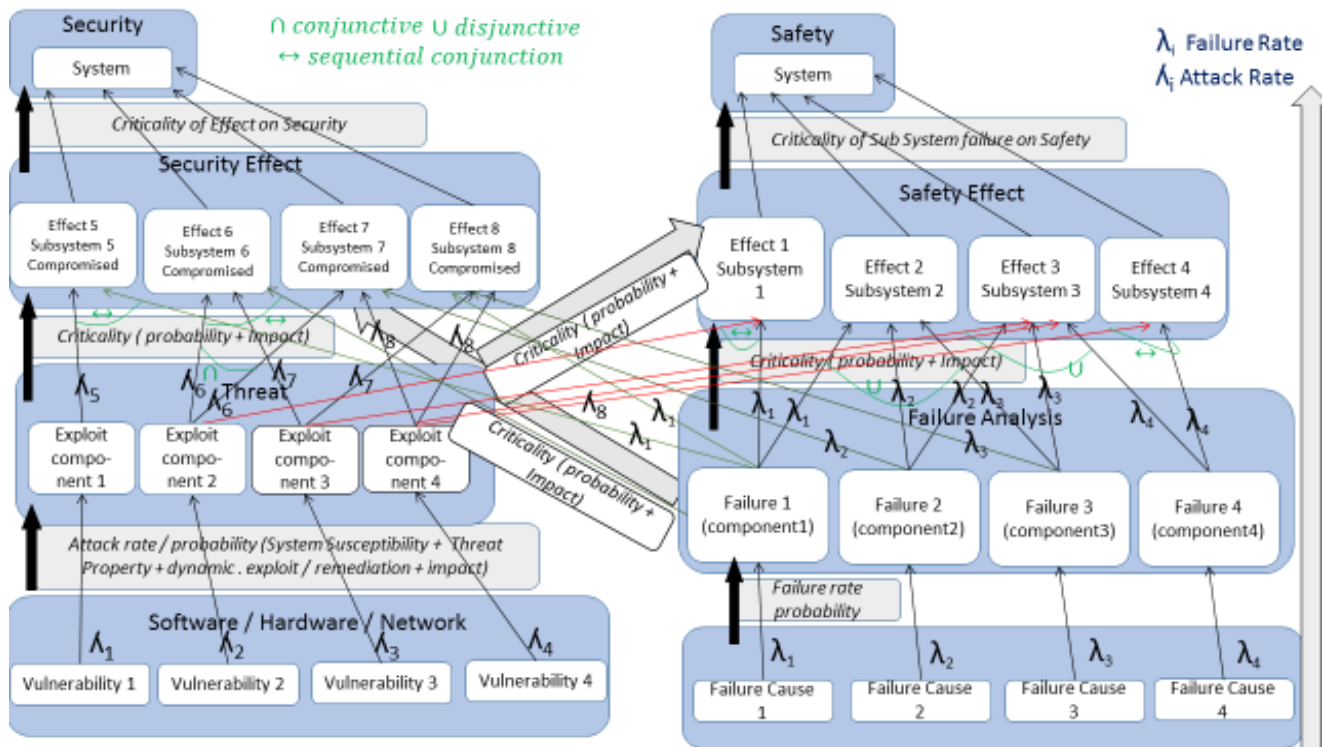


Figure 5. Example for hierarchical network structure

The above Figure 5 is a simple example (just for a rough idea) of a safety-security hierarchical network structure for analysis. For understanding, in safety domain (right side of Figure 5), “failure cause 1” causes failure of “component 1” with a rate λ_1 , failure of Subsystem 2 is a disjunction of failure1, failure2 and failure3 (if any of these failures occurs at component level, subsystem 2 will fail). Subsystem 1 is compromised by sequential conjunction, at first step, component 1 fails, which makes it possible for attacker who already exploited vulnerability 2 to attack subsystem 1 with rate λ_6 to get access and control.

Based on the failure rate/attack rate and effect of these subsystem failures/compromises on system safety, their criticality is evaluated. All this information is provided in a matrix form which is called *SUPERMATRIX*, see Figure 6. This Supermatrix includes all the information from several sources such as the impacts of component failure on subsystem, of subsystem failure on system safety, of any attack mode on system, or of any attack mode on subsystem failure and vice versa for security.

UNWEIGHTED SUPERMATRIX		Component Failures				Subsystem failure / compromise (safety eff.)				Component Compromised				Subsystem failure / compromise (security eff.)				Effect on Safety	Effect on Security
		1	2	3	4	1	2	3	4	Exploit 1	Exploit 2	Exploit 3	Exploit 4	1	2	3	4		
Component Failures	1	1	0	0	0	1	0.13	0	0	0	0	0	0	1	0	0	0.35	0	0
	2	0	1	0	0	0	0.52	0.18	0	0	0	0	0	0	0	1	0	0	0
	3	0	0	1	0	0	0.35	0.46	0	0	0	0	0	0	0	0	0.65	0	0
	4	0	0	0	1	0	0	0.36	1	0	0	0	0	0	0	0	0	0	0
Subsystem failure / compromise (safety eff.)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.21	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.34	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.29	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.16	0
Component Compromised	Exploit 1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
	Exploit 2	0	0	0	0	1	0	0.62	0	0	1	0	0	0	0.4	0.25	0	0	0
	Exploit 3	0	0	0	0	0	0	0.38	0	0	0	1	0	0	0.6	0	0.45	0	0
	Exploit 4	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0.75	0.55	0	0
Subsystem failure / compromise (security eff.)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.15
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.35
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3
Safety		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Security		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6. Unweighted supermatrix of mutual effects between safety and security

- I. Red circle 2 entries show what is the impact of subsystem compromise / failure on overall safety. To calculate this part we suggest using Logical Markov Continuous Time Models as explained later. Impact values are based on severity and rate of failure/ compromise of subsystem. The overall rate is obtained by combining failure attack rates appropriately according to logical operator present.
- II. Green circle 1 entries show the attack rate with which the component vulnerabilities are exploited; this finally leads to compromising the subsystems, which has failure effect. Red circle 1 entries show how component failures interact to cause subsystem failure. These entries are based on relative comparison of failure/attack rates of components w.r.t. to subsystem.

The ANP approach is based on steady state concept, which means after some powers, raised to the matrix it will become constant, and the matrix obtained is called *LIMIT SUPERMATRIX* as shown below in Figure 7. From this matrix, we know the impact of failure causes and attacks on the overall safety and security.

However, to take into account multistage, multiple failure causes / attacks, and their interaction at different levels, we can use logical operators *conjunctive* (\cap), *disjunctive* (\cup), *sequential conjunction* (\leftrightarrow). Combining these operators analytically is however a great challenge mathematically. *Continuous Time Markov Models (CTMM)* however, provide a great advantage analytically in combining these operators for the analysis, and are being used in both safety and security domain. To analyse *RAMS and Security*, one approach can be to combine failure/maintenance tree with attack-defence tree using Logic driven CTMM. A state based transition model can be used to combine RAMS and security aspects. Such as if we consider ETRMS level 3 railway system, the GSM-R communication system for communication between RBC's (trackside system) and trains, consists of many units such as BTS (Base Transceiver Station), RIU (Radio Interface Units), Euro Radio, GPRS infrastructure, Base Controller Station etc., failure of one or more of these units can cause failure of GSM-R communication system failure, which has effect on safety and availability of the railway system.

Similarly, cyber-attacks such as malware, access to communication network, unauthorised interception, cryptanalysis, and man-in-the-middle attack can cause a compromise on integrity of communicated messages or service denial which has effect on safety and availability. Combining these failures and attacks can be done using logical operators and continuous time Markov models which will help us analyse RAMS impact on the railway system due to compromise on GSM-R communication system caused due to propagation of failures/attacks/ their combination. Similarly, the impact on security of the system can be

analysed. This information obtained from CTMM analysis can be adequately provided in the ANP's Supermatrix for extending ANP analysis to other attributes than only safety and security. In context with the Swedish Railway Signalling System, Morant et al. [60] apply Continuous Time Markov Models for failure and availability analysis using combined failure/maintenance trees. Similarly, Jhawar et al. [61] also apply Continuous Time Markov Models using logical operators for security analysis.

LIMIT SUPERMATRIX		Component Failures				Subsystem failure / compromise (safety eff.)				Component Compromised				Subsystem failure / compromise (security eff.)				Safety	Security
		1	2	3	4	1	2	3	4	Exploit 1	Exploit 2	Exploit 3	Exploit 4	1	2	3	4		
Component Failures	1	1	0	0	0	0.45	0.13	0	0	0	0	0	0	0.2	0	0	0.088	0.1387	0.056
	2	0	1	0	0	0	0.52	0.063	0	0	0	0	0	0	0	0.6	0	0.1951	0.21
	3	0	0	1	0	0	0.35	0.161	0	0	0	0	0	0	0	0	0.163	0.1657	0.049
	4	0	0	0	1	0	0	0.126	0.25	0	0	0	0	0	0	0	0	0.0765	0
Subsystem failure / compromise (safety eff.)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Component Compromised	Exploit 1	0	0	0	0	0	0	0	0	1	0	0	0	0.8	0	0	0	0	0.12
	Exploit 2	0	0	0	0	0.55	0	0.403	0	0	1	0	0	0	0.4	0.1	0	0.2324	0.115
	Exploit 3	0	0	0	0	0	0	0.247	0	0	0	1	0	0	0.6	0	0.338	0.0716	0.221
	Exploit 4	0	0	0	0	0	0	0	0.75	0	0	0	1	0	0	0.3	0.413	0.12	0.229
Subsystem failure / compromise (security eff.)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Safety		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Security		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total contribution		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 7. Limit supermatrix of mutual effects between safety and security

Red circle entries show the impact of corresponding component failures (failure cause) on safety and security. Blue circle entries show the impact of corresponding component (vulnerability) exploit on safety and security.

One of the key aspect on using Logical Markov Models for integrating security concerns as above is the use attack rates. Attack rates, similar to failure rates provide a basis for combining cross domain multistage attacks/failures rate and impact (severity + rate of occurrence). However, in the current state of art for considering security concerns for safety as used in FMVEA, the likelihood of successful attack is based on semi-quantitative explicit analysis of susceptibility and threat property of system. The attack rate cannot be determined using empirical data and statistics as such due to constantly changing threat and defence scenario. Therefore, we need a comprehensive approach for calibrating empirical data with semi-qualitative analysis approach to arrive at an appropriate attack rate λ .

In addition to current parameters i.e. susceptibility and threat property, we may need to consider some other factors (with a scale as we have for susceptibility and threat property) which helps in taking into account the dynamics of threat and remediation technique, these factors may possibly include factors as mentioned in "Temporal Metrics" of Common Vulnerability Scoring System (CVSS). The three factors are Exploitability, Remediation Level and Report Confidence. Exploitability factor measures the current state of exploit techniques or code availability for exploiting a vulnerability. Current states of exploit in ascending order of their values possibly be Unproven (No exploit code, only theoretical), Proof-of-Concept, Functional (code available, works in most situations), and High. Similarly, Remediation Levels (RL) of vulnerability in

ascending order of their values can be Official Fix (when official patch is available), Temporary fix, Work around, and Unavailable. Report confidence is about the official status of acknowledgement of vulnerability. We should also include the impact of attack on several attributes, such as if the impact is safety critical and catastrophic then we should consider a higher attack rate and also patching rate should be less because a SIL 4 requirement needs more time for assurance of efficiency of patch, to be on safer side.

2.1.3 Further development of SiSoPLE for enabling process-related co-assessment (*)

This subsection first recalls basic information on SiSoPLE (Security-informed Safety-oriented Process Line Engineering), which is the extension of SoPLE (Safety-oriented Process Line Engineering), developed in the framework of the SafeCer project. The recalled information is mainly borrowed from [15]. Then, this subsection recalls a couple of normative spaces, where the need for SiSoPLE is evident and emerging.

Finally, this subsection sets the conceptual underpinning for a more in-depth development of SiSoPLE.

2.1.3.1 SiSoPLE and SiS-related terminological framework

SiSoPLE was initially introduced by Gallina et al. 2015 [15]. SiSoPLE builds on top of the dependability-related terminological framework and its expansion.

More specifically, as recalled by Gallina et al. 2015 [15], Aviezienis et al. 2004 [16] introduced a terminological framework aimed at characterizing dependability in terms of its attributes, threats (faults, errors, and failures) and means. Dependability is usually indicated as an umbrella term, which embraces various aspects (attributes) related to trustworthiness. Safety and security are two dependability attributes.

Safety is defined as absence of catastrophic consequences on the user(s) and the environment. Security is defined as composite attribute constituted of availability, integrity, and confidentiality. Availability is defined as readiness for correct service. Integrity is defined as absence of improper system alterations. Finally, confidentiality is defined as absence of unauthorised disclosure of information.

Security-informed safety is an expression that has been recently introduced [17] to indicate an old truth: “For a system to be safe, it also has to be secure”. To guarantee an agreed level of safety/security, besides knowing what can go wrong, a risk assessment is needed.

Despite the existence of the dependability terminological frameworks and despite the awareness related to the above-stated truth, the security and safety communities have progressed by following different development paths. For instance, they define risk in a slightly different way. The safety community defines risk as the evaluation of the effect of a failure condition. This assessment takes into consideration the probability and severity and thus enables the judgment with respect to acceptability.

The security community defines risk [18] as threat x vulnerability x consequence, where consequence takes into consideration the attacker capability, the asset (i.e., aircraft if the risk is assessed at aircraft level) exposure and thus enables the judgment with respect to acceptability.

Further to terminological differences, process differences exist between the security and the safety domains. However, there are strong reasons to align the safety and the security processes. Four main reasons were identified to motivate the introduction of SiSoPLE: (1) security assessment should be mostly focused on safety-critical and safety-related functions. If security assessment is performed without the knowledge of failure conditions, it may be performed inadequately and potentially not completely. Therefore, safety assessment should feed inputs to the security risk assessment process to highlight functions of importance to the security analysis; (2) safety decisions regarding requirements and architecture should ideally not interfere with similar security decisions. In the worst case, safety measures could conflict with security measures or one domain could limit technical solutions for the other domain. Architecture or equipment decision rather than being taken unilaterally should be taken in a collaborative

manner between safety and security; (3) once security threats are identified, they may need to be fed back into the safety process to show the relationship between threat conditions and failure conditions; and (4) finally, a common picture of risk assessment encompassing security and safety will likely be preferred by certification authorities. Certification authorities may accept separate system assessments for safety and security. However, the certification authorities will expect to see a global understanding of these risks and their influence on system design.

SiSoPLE is a process lines engineering method that, similarly to SoPLE, is constituted of a scoping phase, a domain engineering phase, and finally a process engineering phase.

During the domain engineering phase, commonalities and variabilities are identified. To do that, for each standard, the following actions are taken:

- identification of certification-relevant process elements (e.g., activities and tasks)
- identification of the order in which activities and tasks should be performed
- identification of the way in which tasks are grouped to form activities
- identification of the way in which activities are grouped to form phases

Then, activities are compared with activities, tasks with tasks, etc. We also compare the order of execution. To ease this comparison, several aspects such as: irrelevant terminological differences; irrelevant ordering differences; and irrelevant grouping differences have to be overcome. More specifically, to overcome irrelevant terminological differences, the dependability-related terminological framework constitutes the starting point.

Overcoming irrelevant terminological differences or identifying significant points of variations is crucial since it permits (process) engineers to reduce the complexity of the systems to be engineered as well as the complexity of the certification process.

Once the commonalities and variabilities are known, a SiSoPLE model should be provided. To engineer single processes, aimed at satisfying a single certification body, process elements are expected to be selected and composed: all the commonalities are expected to be selected, jointly with the required variants, selected at corresponding variation points.

A security informed safety process line is expected to enable the alignment of security and safety standards. As discussed in the background, there are strong reasons to enable such alignment since, if the alignment is not performed, the resulting safety assessment conclusions may be incomplete, the technical solution might be less than ideal and more engineering effort might be required to harmonise both security requirements and architecture with the safety requirements late in the design phase.

While there is also potential for re-use between the security and the safety processes, these aspects mostly highlight that without some level of synergy between the security and the safety processes, an organization may not produce a safe system or encounter resource and/or technical challenges.

Technical aspects related to in-depth SiSoPLE modelling and single SiSoProcess engineering are given in AMASS D6.2 [12]. Some initial results were published in [21]. Additional work regarding compliance checking in the context of co-assessment is under development. The direction is the one currently pioneered by Castellanos et al. 2017 [47] consisting of combining SiSoPLE with defeasible logics, and an approach for compliance by design specifically created for business processes.

Moreover, continuation of SafeCer work on generation of process-based argument fragments is also in focus. MDSafeCer [49][48] and THRUST-related [50] solutions are being adopted and extended to argue about compliance in the context of safety and security standards' interplay.

2.1.3.2 Normative spaces ready for SiSoPLE (*)

In this subsection, examples of normative spaces are given. In particular, the attention is focused on those domains (avionics and automotive) where multiconcern normative spaces seem to be defined and awareness regarding the need for co-assessment is spreading.

Avionics: RTCA DO-326A/ED-202A

RTCA DO-326A/ED-202A [19] is a joint product of two industry committees: the EUROCAE Working Group WG-72, titled “Aeronautical Systems Security” and the RTCA Special Committee SC216, also titled “Aeronautical Systems Security”. DO-326A is a document that provides guidance to handle the threat of intentional unauthorised electronic interaction to aircraft safety. More specifically, it defines a set of partially ordered activities that need to be performed in support of the airworthiness process to handle such threat. This set of partially ordered activities is known as Airworthiness Security Process. This process is constituted of a set of activities: Plan for Security Aspects of Certification (PSecAC), Security Scope Definition, Preliminary Aircraft Security Risk Assessment, Security Risk Assessment, Security Development related activities, Security effectiveness assurance, and Communication of evidence (via PSecAC Summary). These activities are in turn composed of various tasks.

In this section, we focus on a single activity, called Preliminary Aircraft Security Risk Assessment (PASRA), which belongs to the risk assessment set of activities. PASRA is aimed at identifying threat conditions and threat scenarios and assessing all security risks at aircraft level. PASRA takes as input the architecture under consideration, failure conditions and severity (which are established during the execution of the system development process described in ARP4761) and the information related to the security environment and perimeter, defined during the Security Scope Definition. Based on the input received, the following set of tasks is performed within the PASRA task: Threat Condition Identification and Evaluation, Threat Scenario Identification, Security Measure Characterization, and Level of Threat Evaluation. The final outcome of PASRA is the Preliminary Security Effectiveness Objectives, based on identified & evaluated **threat conditions**. DO-326A describes what security-related activities need to be performed but does not provide much guidance about how to perform these activities. DO-326A is expected to be used in conjunction with its companion document DO-356, which provides guidance and methods for accomplishing the activities identified in DO-326A in the areas of Security Risk Assessment and Effectiveness Assurance.

Avionics: ARP4761 Including its Expected Evolution

ARP4761 [20] Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment is an Aerospace Recommended Practice from SAE International. ARP4761 is a document that provides guidance to perform safety assessment. More specifically, defines a set of partially ordered activities that need to be performed in support of the airworthiness process to handle hazardous events (system and equipment failure or malfunction that may lead to hazard). This set of partially ordered tasks is known as Airworthiness Safety Assessment Process. This process, as newly stated in ARP4754A, is constituted of: Functional Hazard Assessment (FHA), performed at aircraft and system level, Preliminary Aircraft Safety Assessment (PASA), Preliminary System Safety Assessment (PSSA), System Safety Assessment (SSA) and, Aircraft Safety Assessment (ASA). Let us focus on Aircraft-level FHA. Aircraft-level FHA is aimed at identifying failure conditions and assessing all safety risks at aircraft level. Aircraft-level FHA takes in input the list of top-level functions plus the initial design decisions (architecture), the aircraft objectives and requirements. Based on the input received, the following set of steps is performed within the Aircraft-level FHA task: identification of all functions and corresponding **failure conditions**, determination of effects of the failure conditions, and classification of the determined effects. The final outcome of Aircraft-level FHA is the safety objectives and the derived safety requirements, based on identified & evaluated failure conditions.

Avionics: RTCA DO-326A/ED-202A and ARP4761 comparison

The Preliminary Aircraft Security Risk Assessment (PASRA) and the Aircraft-level Functional Hazard Assessment (AFHA), which are respectively defined in the above standards are further considered. By comparing PASRA and AFHA, commonalities and variabilities can be identified. PASRA and AFHA are both characterised by similar steps. PASRA and AFHA are both expected to produce in output a work product indicating the identified and evaluated **conditions**; such output can be seen as a partial commonality.

Commonality identification is not only useful for the purpose of reuse of cross-concern process information. It is also useful to enable in-depth co-assessment. As it was discussed by Gallina et al. [15], commonality identification and more in general SiSoPLE modelling would increase effectiveness since conflicts between safety and security will be dealt with early in the lifecycle and the risk of re-work later in the development cycle is reduced. SiSoPLE enables the alignment of multiple standards within a single model and thus it offers a means for the introduction of synergies between safety and security experts, avoiding potential conflicts.

Automotive: ISO 26262

ISO 26262 [62] regulates all phases of the entire lifecycle of the product (item), starting from the management and requirements specification phases up to the production release. The standard recommends the usage of a V-model at item level as well as at element (software and hardware) level. ISO 26262 consists of 9 normative parts, each of which is structured into clauses. All the clauses state the objectives, inputs for the clause, recommendations and requirements to be fulfilled and finally the work products that are to be generated. Notes are also included. Notes are not normative and are expected to help the applicant in understanding and interpreting the requirements. Additionally, obligations on the corresponding methods are also imposed based on the assigned ASIL (Automotive Safety Integrity Level).

Within this deliverable, the attention is limited to clause 8 of Part 6, which is related to the left-hand side of the software V-model, more specifically to Software Unit Design and Implementation. The first objective of this clause (Software Unit Design and Implementation) is to specify the software units in accordance with the software architectural design and the associated software safety requirements. A single activity (A1) can be identified for this purpose: A1-Specify the software units. The outcome for this activity is the work product Software unit design specification, which is the result of the application of the following requirements:

- The requirements of this clause shall be complied with if the software unit is safety-related ("Safety-related" means that the unit implements safety requirements).
- Software units are designed by using a notation that depends on the ASIL and the recommendation level.
- The specification of the software units shall describe the functional behaviour and the internal design to the level of detail necessary for their implementation.
- Design principles for software unit design shall be applied depending on the ASIL and the recommendation levels to reach properties like consistency of the interfaces, correct order of execution of subprograms and functions, etc.

Automotive: SAE J3061

SAE J3061 [24] is a recently published Cybersecurity Guidebook, that provides a process framework for a security lifecycle for cyber- physical vehicle systems. SAE J3061 methods and procedures are very similar to the ones described in ISO 26262. This similarity allows the process to be applied in three different ways: a) separately from a system safety engineering process with integrated communication points, b) the two processes can be tightly integrated, or c) develop shared process and steps that are shared with safety, and then add the unique Cybersecurity process and steps. Options b) and c) have in common that they allow for cross-concern reuse. This reuse is specifically mentioned in part 8 Process Implementation of SAE J3061: "if a Cybersecurity process is tailored from an organization existing safety process and the processes are analogous to each other (share a common framework), then the Cybersecurity process can be developed by leveraging work that has already been done in the safety process development". However, in system safety the focus is on safety-critical systems, whereas in system Cybersecurity, both safety and non-safety-critical systems are considered, since a Cybersecurity-critical system is a system which may lead to financial, operational, privacy or safety losses.

Within this deliverable, also in the case of SAE J3061, the attention is limited to a small portion related to clause 8, Part 6 of ISO 26262.

Section 8.6.5 of SAE J3061 describes the Software Unit Design and Implementation phase in which one of the activities is the Design of the software units. The result of this activity is the Software unit design and it is related to two guiding principles on Cybersecurity for Cyber-physical systems. These principles are:

- Design the feature with Cybersecurity in mind, starting in the concept phase of the development lifecycle. Engineers should consider Cybersecurity when defining the requirements that are to be met for the system and features.
- Have status reviews to assess whether design work is on track to meet the Cybersecurity requirements.

Automotive: *Interplay and comparison between ISO 26262 and SAE J3061*

By comparing ISO 26262, clause 8 of Part 6 and Section 8.6.5 of SAE J3061, commonalities and variabilities can be identified. Thus, similar observation as for the avionics domain can be formulated.

Moreover, it should be also observed that ISO 26262 is used to create a safety case where developers show that a system achieves a reasonable level of functional safety and is free of unreasonable risk. Functional safety concerns failures in electrical/electronic (E/E) components, which may lead to a hazard. Identification of hazards is performed with methods like hazard analysis and risk assessment and fault tree analysis. The ISO 26262-2011 concerning automotive functional safety does not mention any cybersecurity relation. This means that safety processes based on the first edition of the ISO standard did not cover any security aspects.



Figure 8. Interaction between safety and security engineering

However, the trend is towards implementing highly interconnected system functions in software, the systems are not isolated and they become cyber-physical. That implies security has to be part of the centre of interest. To overcome security issues, SAE J3061 is available to provide guidance for the development of cyber-physical vehicle systems. Its structure is analogous to the process framework from ISO 26262 but SAE J3061 introduces equivalent cybersecurity activities.

The existing safety-related processes have to be expanded with methods like threat analysis and risk assessment and attack tree analysis. The overall management of functional safety has to be extended with the management of cybersecurity.

An important aspect is the identification of the relationship between cybersecurity and safety. In some cases, cybersecurity influences only non-safety areas like privacy or financial impact. Our intention is to identify all possible ways how functional safety may be violated in the different development lifecycle phases. The concept phase intends to perform a risk analysis. In a combined process cybersecurity and safety risks will be identified jointly. In this context we have to consider that we have still risks which are only related to safety issues (e.g. hardware failure) and risks which are only related to cybersecurity (e.g.

attackers want to capture personal data). Cybersecurity risks without safety relation will be possibly identified but they are out of scope from our perspective.

Based on analogies between safety and cybersecurity it is useful to define processes, which are integrating both topics. An integrated point of view is necessary because safety and security analysis will lead to measures, which have the task to mitigate identified risks, which can be caused by both disciplines.

Co-engineering in our approach means to create integrated processes regarding safety and security. SiSoPLE is an appropriate method to bring activities from different domains together. It manages the handling of commonalities, variabilities and provides the opportunity to add optional activities. It improves the essential communication between the disciplines (see Figure 8). Furthermore, to tackle the co-engineering demands the approach has to cover hazards, which arise due to the combination of safety and security risks. We need to perform a safety and security co-analysis. This type of analysis should guarantee that we identify potential hazards, which would stay undiscovered if only one discipline is examined in an isolated way. The measures from competitive disciplines must not influence each other in a not admissible way ("freedom of interference"). To make sure that is true, we have to perform a trade-off consideration. Initially we had a trade-off between performance and safety, now we have to add cybersecurity as a further attribute. To find a tolerable balance between measures, we have to rate them with a "trade-off metric". In other words, developers have to decide how much impact is allowed for each of the safety and security measures. The metric is provided as an aid to find out the balance and as an argument why a specific safety security constellation has been chosen. Finally, all these arguments have to be collected in the assurance case, which covers the integrated safety and security case.

The following paragraph describes the process development in EPF-C and the process execution with WEFACT based on an example. To illustrate the approach an exemplary process concerning verification of system design has been created in EPF-Composer. The process is based on ISO 26262 and SAE J3061. It considers "Product development at the system level", "Supporting processes" and "Cybersecurity activities". Figure 9 shows the activities of the process in detail.

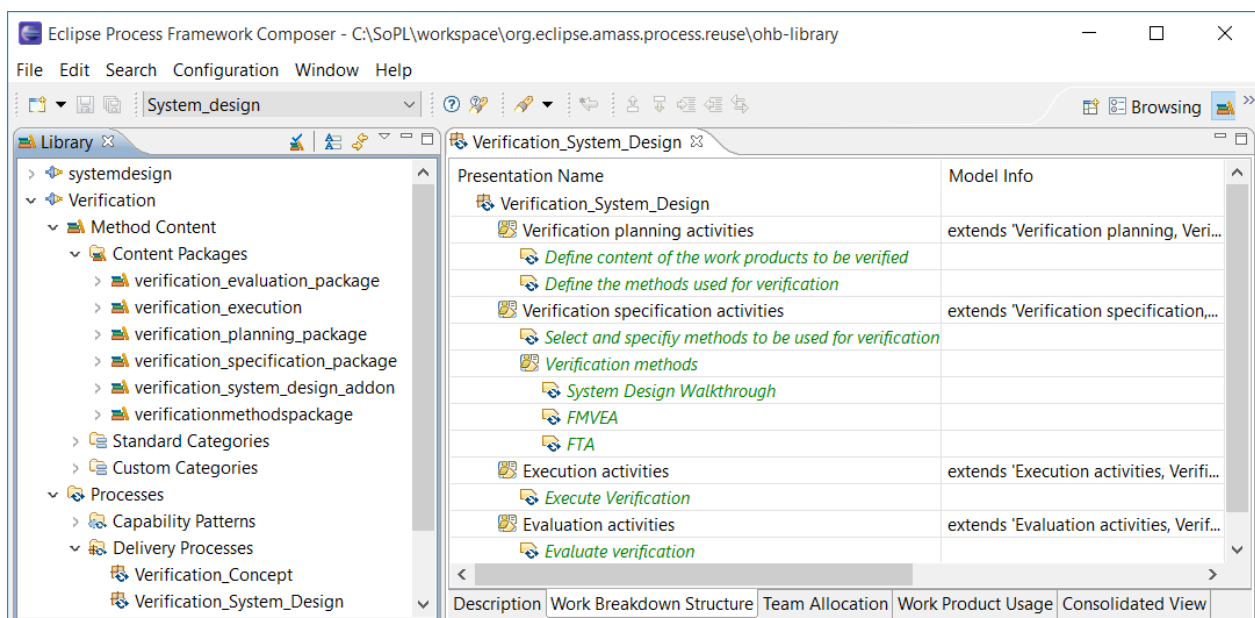


Figure 9. Work Breakdown structure of process related to verification of system design

The process is available in the work breakdown structure, which allows activity structuring. It is based on a verification pattern, which includes the general main activities related to verification. EPF-C stores patterns in the process folder "Capability Patterns". The verification pattern is extended by specific activities for system design, verification, and cybersecurity verification. The extension of activities is a feature of EPF-C.

To use this feature additional tasks are defined and added to the pattern using the "Content Variability" function of EPF-C. This function allows the extension of tasks with input from other tasks.

The example shows how cybersecurity is added to a process, which is mainly designed for functional safety consideration. Performing co-engineering is important because functional safety and cybersecurity issues are highly interactive. Cybersecurity can be taken into consideration by adding a safety and security co-engineering loop [71]. In this loop, the developers make sure that the added cybersecurity measures do not influence the safety measures in an unintentional manner. It is important that interactive activities are considered jointly and not separately. The cycle stops when the system fulfils the demanded requirements.

Once the process has been defined, it is ready for execution with WEFACT. Before an EPF-C model can be executed, it has to be exported from EPF-C to an XML file and subsequently imported to WEFACT.

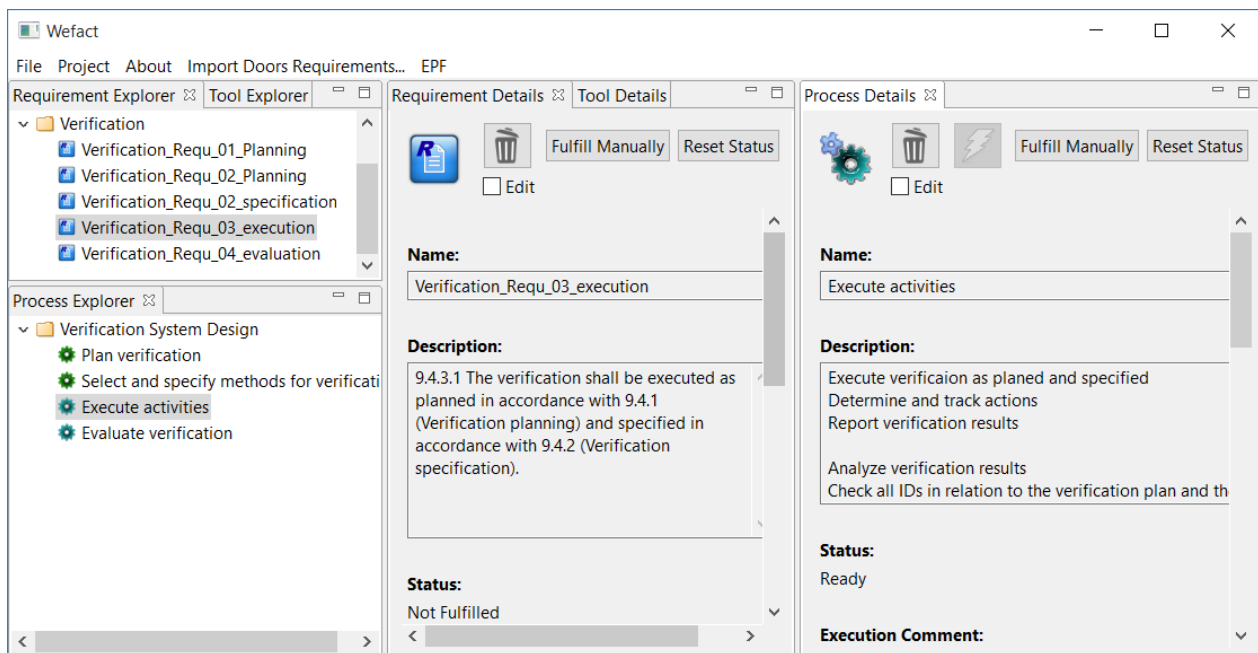


Figure 10. Process related to verification of system design in WEFACT

Figure 10 shows the process model imported to WEFACT. It appears in the "Process Explorer" in the lower left corner. All activities of the process have to be on one level because the current version allows no structured processes. The next step is to connect requirements, input- and output files to the process. Requirements can be defined in WEFACT or they can be imported from a DOORS database, or the process model is created in EPF-C and imported into WEFACT in UMA format.

Before the execution can be performed, workflow tools have to be defined and associated with the process. These tools use the available input files and produce output files according to the process specification. The lightning symbol in the upper right section of the process tab starts the tool. The button is enabled if the process is ready for execution. This is the case if the predecessor has been fulfilled and a tool has been linked. WEFACT provides the opportunity to fulfil processes manually by using the assigned button. The resulting output files are stored in a folder, which is under revision control by SVN. The appearance of a new file indicates that the process was executed successfully. The status changes to "successfully".

WEFACT supports process execution activities, makes sure that requirements are fulfilled, related processes are executed properly and all work products are available. The generated work product files are used as evidence in the assurance case. The deliverables D1.5 to D1.7 demonstrate the methodology, by applying it in automotive case studies.

The question, how can we define a metric to evaluate trade-offs, needs further investigation; in particular the following two aspects need to be taken into consideration:

- Risk reduction (e.g. overall risk decreases, even though safety or security risk may increase)
- Lifecycle costs (e.g. engineering and production costs may increase)

The idea of safety and security interaction is at the moment discussed in standard committees and should appear in the next release of ISO 26262.

2.1.4 Co-assessment for Safety and Security Assurance (*)

Parallel to process-related assurance assessment, **product-related co-assessment** is to determine the effectiveness of functional safety measures and functional security measures with respect to their safety and security objectives. The safety and security objectives can be specified by the requirements. Assessment methods include verification, validation, and testing. The results of the assessment can be used for safety and security argumentation in assurance cases [67].

Industries such as nuclear, aviation, railways, and their regulatory agencies have over the years developed standards, analytical techniques for safety assessment with interdisciplinary applications. Different lifecycle phases have to be covered for the safety assessment during the design and development of dependable systems. This starts with a description of functional hazard assessment (FHA), followed by the preliminary system safety assessment (PSSA) and system safety assessment (SSA). The aerospace industry has amongst the most rigorous standards. An important guiding document for safety in the development of new aircraft is ARP 4761 [20]. The methods employed are qualitative, quantitative, or both. The development process is iterative in nature with system safety being an inherent part of the process. The process begins with concept design and derives an initial set of safety requirements for it. During design development, changes are made to it and the modified design must be reassessed to meet safety objectives. This may create new design requirements. These, in turn, necessitate further design changes. The safety assessment process ends with verification that the design meets safety requirements and regulatory standards [20]. These techniques are applied iteratively. Once FHA is performed, PSSA is performed to evaluate the proposed design or system architecture. The SSA is performed to evaluate whether the final design meets requirements.

The **Functional Hazard Assessment (FHA)** is performed at the beginning of system development. Its main objective is to “identify and classify failure conditions associated with the system by their severity”. The identification of these failure conditions is vital to establish the safety objectives. This is usually performed at two levels, for the example of aircraft industry—at the completed aircraft level and at the individual system level. The aircraft level FHA identifies failure conditions of the aircraft. The system level FHA is an iterative qualitative assessment which identifies the effects of single and combined system failures on aircraft function. The results of the aircraft and system level FHA are the starting point for the generation of safety requirements. Based on this data, fault trees, FMEA can be performed for the identified failure conditions which are studied later. ARP 4761 provides guidelines on how an FHA should be conducted.

The **Preliminary System Safety Assessment (PSSA)** is a systematic examination of the proposed system architecture to examine how failures can lead to the functional hazards identified by the FHA and how safety requirements can be met. The PSSA addresses each failure condition identified by the FHA in qualitative or quantitative terms. It involves the use of tools such as FTA, Dependence Diagram (DD), and Markov Analysis (MA) to identify possible faults. The use of these is discussed later. The identification of hardware and software faults and their possible contributions to various failure conditions identified in the FHA provides the data for deriving the appropriate Development Assurance Levels (DAL) for individual systems. The process is iterative being performed at the aircraft level (for the case of airplanes) followed by individual system levels.

The **System Safety Assessment (SSA)** is a systematic, comprehensive evaluation of the implemented system to show that qualitative (system development assurance levels, item development assurance levels,

hardware design assurance levels and software levels) and quantitative (safety-related reliability targets) safety requirements, defined in the FHA and PSSA have been met. The SSA integrates the results of the various analyses to verify the overall safety of the system and to cover all the specific safety considerations identified in the PSSA. The SSA process documentation includes results of the relevant analyses and their substantiations as needed. The output of the SSA is used as an input for the Safety Case.

Co-verification and validation has been extensively discussed in D4.1 [1] Section 4.1.3. Regarding security testing, it is the process of exercising one or more assessment objectives under specified conditions to compare actual and expected behaviour.

Security assessment is domain-specific. In the following, we use an example in the automotive domain to explain the principles and common methods in security assessment.

Security assessment can generally be divided into two parts [31]: a theoretical security assessment and a practical security assessment. The theoretical security analysis identifies and understands the security weakness of an automotive CPS system based on a paper-based evaluation of the corresponding system specifications and documentations, for example, Threat Analysis and Risk Assessment (TARA) as described in SAE J3061. Methods such as architecture review, threat modelling, and attack tree can be used to identify attack surface, entry point, weakness in cryptographic algorithms, and potential attacks. However, the theoretical security analysis does not identify implementation flaws or the deviation of the implementation from the specification. Moreover, it cannot detect vulnerabilities that are part of insufficiently documented specification or flaws hidden in components from a third-party from the supply chain.

The practical security assessment can discover implementation errors that might be exploited by an attacker. It can also find unspecified functionality and deviation to the specification. Practical security assessment includes functional security testing for testing security-related functions for correct behaviour and robustness, vulnerability scanning to test the system for known-vulnerabilities, fuzzing to find new vulnerabilities of an implementation by sending malformed input to the target system to check for unknown, potential security-critical system behaviour, and penetration testing to mimic an intelligent human attacker to identify and exploit all vulnerabilities in a sophisticated way based on hacking experiences. However, practical security testing cannot give assentation on completeness of the test. Hence, it should always be complemented by a theoretical security testing to identify possible security flaws.

In the Industrial Automation and Control System (IACS) domain, security assessment often involved various security test methods, including stress test, port scan, vulnerability scan, protocol fuzzing. In stress test, a Denial of Service attack is launched on all TCP/IP protocols to ensure that the product can provide appropriate resistance against the attack. In port scan available ports (e.g. FTP TCP port 21) are targeted with malicious software which may lead to malfunction of system. In vulnerability scan, a software scanner is used to detect known vulnerabilities of the used and documented TCP/UDP ports and services, e.g. HTTPS port 443. In protocol fuzzing, a software fuzzer is used to cause a denial of service attack or a targeted system crash, by exploiting access violation or untreated program state. Variables in protocol fuzzing include features and constraints, forbidden or reserved values, linked parameters, and filed sizes.

To implement fuzz testing method, a test platform (Figure 11) can be tailored for security testing of IEC 61850, using a fuzzing simulator, IEDs, a remote-controlled power strip, and a switch [32].

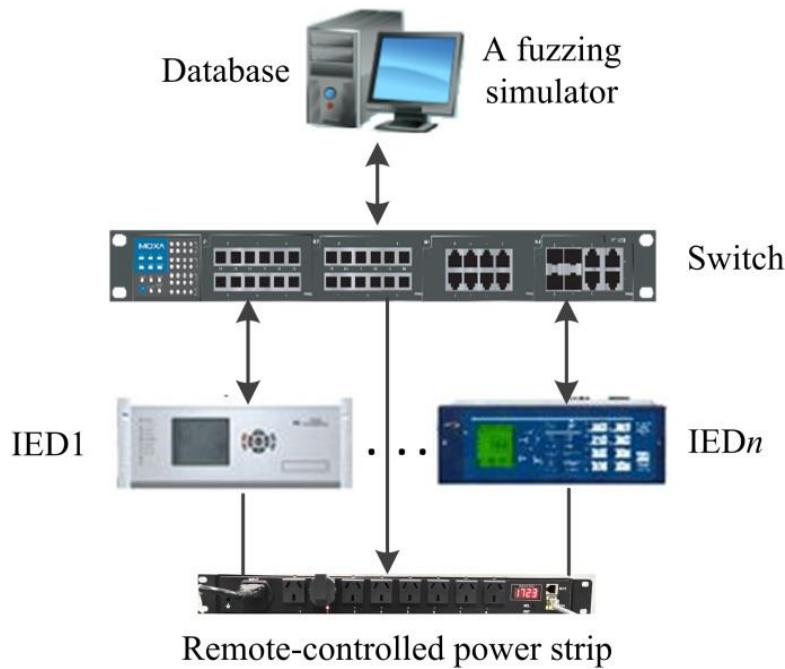


Figure 11. A testbed for fuzz testing of IEC 61850

2.1.4.1 System dependability co-analysis via ConcertoFLA (*)

ConcertoFLA [73] enables users (system architects and dependability engineers) to decorate component-based architectural models (specified using CHESMML) with dependability-related information, execute Failure Logic Analysis (FLA) techniques, and get the results back-propagated onto the original model. CHESMML is an extension of SysML [79] used in CHES toolset to enable component-based systems design. The dependability modelling is supported by SafeConcert [63], a subset of CHESMML, which allows the modelling of the failure behaviour for system components and so model-based dependability analysis, like failure propagation or state-based analysis.

Different FLA techniques are available in the literature [74], and can be used at the early stages of the design phase to achieve a robust architecture with respect to linear relationships. ConcertoFLA builds on top of Failure Propagation Transform Logic (FPTC) [75]. Similar to FPTC, ConcertoFLA is a compositional technique to qualitatively assess the dependability of component-based systems. In ConcertoFLA terms, a component can act in four different possible ways:

1. Source of the failure thus generating a failure due to an internal fault.
2. Sink of the failure thus avoiding the propagation of the external fault (failure in input) through fault tolerance.
3. Propagator of the failure.
4. Transformer of the failure into a different type.

ConcertoFLA rules are logical expressions, which specify the component's behaviour by describing the input/output relationship. ConcertoFLA rule is a combination of the port (input/output) and the guide word referring to the failure mode; supporting standard failure modes i.e., timing, value and provision. Each of these failure modes has two specializations, which are early & late, ValueSubtle & ValueCoarse, and Omission & Commission corresponding to timing, value and provision respectively.

ConcertoFLA allows users to calculate the behaviour at system-level, based on the specification of the behaviour of individual components. During the analysis, ConcertoFLA calculates the failure propagation paths and produces their representation according to the specifications of FlaMM meta model (see [76] for FlaMM structure and corresponding XML Schema). These failure propagation paths are utilized to generate a fault tree. In fault tree terminology, the failure at system level is referred to as *top event* and the

contributing failures are classified as intermediate and basic events. In safety context, the top event refers to a safety hazard which may cause accidents. In the security context, the top event is a breach of security properties i.e., confidentiality, integrity and availability. The intermediate and basic events contributing to the top event could also be due to the compromise of any of the concerns. For example, a cyber-security attack, which makes a component to halt its services, may cause a safety hazard, which, in turn, may cause an accident. Therefore, a fault tree integrating the security threat events contributing to the safety hazard could enable security-informed safety. To support the generation of such rich fault tree, a further elaboration is introduced to the input/output failure behaviour of the components. Before discussing modelling of the elaborated failure behaviour of components, first, security threat process and security related terms are introduced in what follows.

The causality-chain that leads to the violation of the security-related properties is illustrated in Figure 12. A threat event, initiated by a threat source agent, able to exploit a vulnerability of an asset (e.g. a component/system) may result in a loss to the confidentiality, integrity and/or availability (often, together referred to as CIA) of the asset [77] [78]. *Threat* refers to the event or capability to breach security and cause harm. Where the *vulnerability* is a weakness or internal flaw in the design, architecture or implementation of a service/application. A threat source could be a malicious cyber-security attack, non-malicious human errors, natural/human made disasters, etc. It is also worth noting that the accidents caused by safety hazards can also be a threat source enabling a situation, where a threat may exploit a vulnerability and cause a security breach. However, to this end, the focus is on cyber-security attacks as threat source; hence attacker as a threat source agent. The loss to CIA of a component or system, which is a consequence of an attack, could be as following:

- Unauthorized access of the system (loss of confidentiality)
- Halting services of the system (loss of availability)
- Corrupting the services of the system (loss of integrity)

For instance, a cyber-security attack on a component, where a *data corruption* threat exploits the *missing data integrity schemes* vulnerability, may result in corrupting the services of the component.

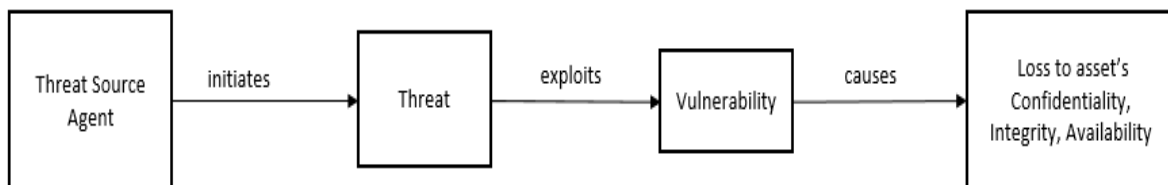


Figure 12. Process of Security breach [77] [78]

The further elaboration of the input/output failure behaviour of components could be modelled using state machine diagram. Safeconcert, which is the CHES dependability profile, allows to tag the state machine with <<ErrorModel>> stereotype. The error model provides support for modelling state transitions, erroneous state and the effect of this on a property of the component and its nominal behaviour. The state transition in the error model are specialized and could be tagged with <<Failure>> and <<InternalFault>> stereotype.

To model the security, using an <<ErrorModel>>-tagged state machine, the failure, internal fault and effect are extended to include security threats, vulnerability and consequences respectively. The security threats could be represented by a pre-loadable vocabulary (through exploitation of the connection with EPF Composer, where the requirements mandated by the standards are modeled), which refers to the specific threats used within a specific domain/standard. The inclusion of different types of threat could be collected from the catalogues of the domain and standard. For example, in the space domain, when engineering a space mission, the vocabulary provides a pre-defined enumeration of common/discovered security threats collected from CCSDS 350.1-G-2 [77] and NIST 800-30 [78] as well as by the personal competences (e.g., respective system engineer, security analyst, etc.). In a similar fashion, the vulnerabilities could be

represented as a pre-defined enumeration collected through different sources (for example personal competence, standards and results of previous threat analysis, etc.). Finally, the consequences could also be modelled using pre-defined effects, which refers to the loss of CIA. Figure 13 illustrates the error model, where a cyber-security attack initiates *data corruption threat* and exploiting the *value check function is false* vulnerability thus causing a transition to erroneous state.

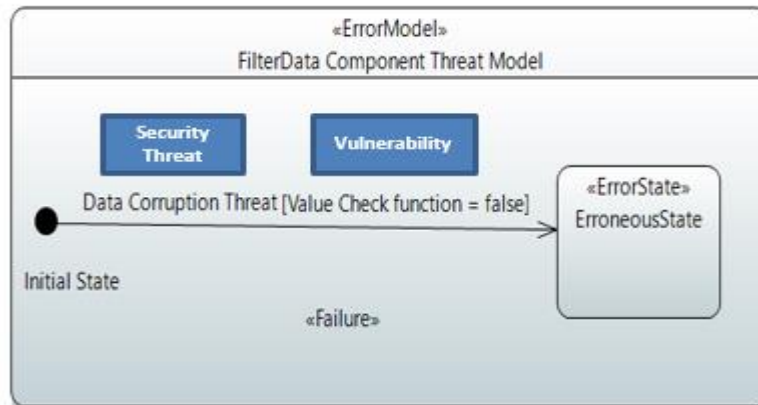


Figure 13. Error Model showing erroneous state transition due to security threat event and vulnerability

A component could have multiple instances of <<ErrorModel>>-tagged state machines, attached to it. Each instance would provide the elaboration of input/output failure behaviour addressing a specific concern.

2.1.4.2 WEFACT Tool Concept (*)

WEFACT has been delivered in D4.5 [6] and the executable as well as a user manual are available. The extensions described in the following refer to functionalities for which the necessary capabilities are already available in WEFACT. It is in this sense rather about using the tool appropriately or in an extended manner. Exploiting the existing WEFACT features in this extended manner fulfils a couple of additional requirements beyond those cited when releasing the first tool in D4.5. The following sections explain which requirement is fulfilled by the extension and how WEFACT must be used in order to exploit the capability.

Quantitative confidence metrics about an assurance case in a report.

This refers to the “Could” requirement WP4_ACS_013: Provide quantitative confidence metrics about an assurance case in a report. “The system could produce a status report indicating a quantitative confidence metric for assurance case.”

This requirement is partially fulfilled by WEFACT. Instead of a written, printable report, however, WEFACT displays the percentage of fulfilled evidences continuously on the screen.

The following Figure 14 shows the WEFACT user interface.

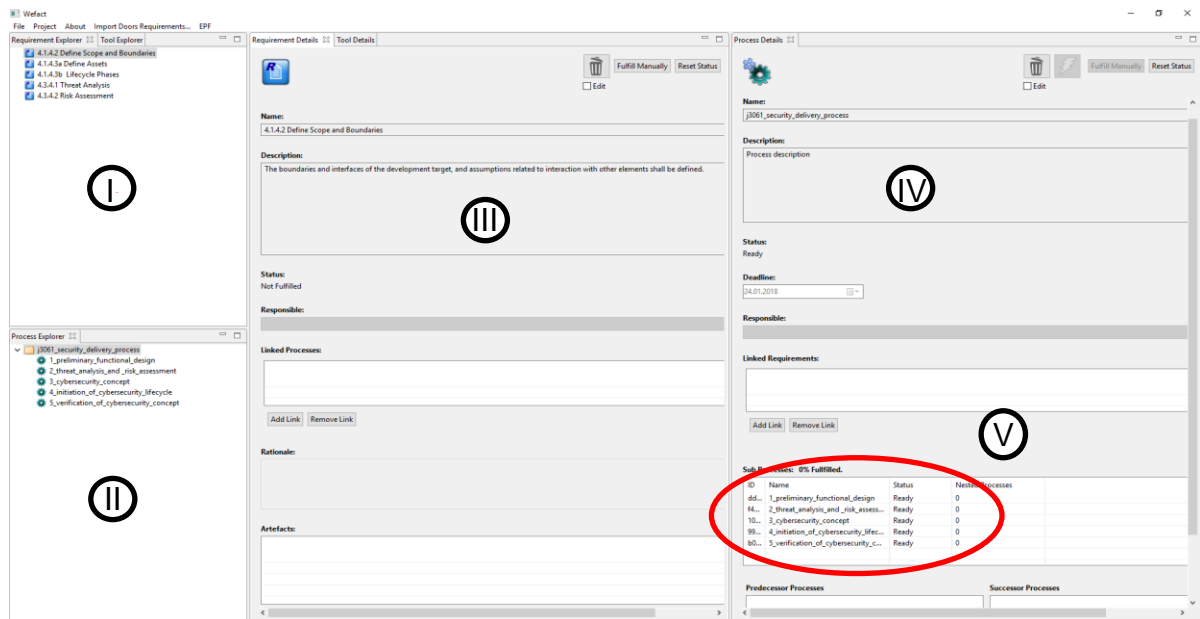


Figure 14. WEFACT user interface

The different parts of the screen marked with Roman numbers in black cycles contain information w.r.t requirements and processes as well as subprocesses:

- I. Project specific requirements
- II. Predefined engineering processes
- III. Requirement content
- IV. Process definition
- V. Child processes

On top of the list of child processes (in the red ellipse), the percentage of fulfilled subprocesses is displayed and continuously maintained (e.g. during process executions). Figure 15 shows this part of the screen with higher resolution.

Sub Processes: 0% Fulfilled.			
ID	Name	Status	Nested Processes
dd...	1_preliminary_functional_design	Ready	0
f4...	2_threat_analysis_and_risk_assess...	Ready	0
10...	3_cybersecurity_concept	Ready	0
99...	4_initiation_of_cybersecurity_lifec...	Ready	0
b0...	5_verification_of_cybersecurity_c...	Ready	0

Figure 15. Sub processes in WEFACT and the share of those fulfilled

Extension to Assurance case status report

This refers to the “Could” requirement WP4_ACS_011: Assurance case status report = “The system could provide the capability for querying the assurance case in order to detect: 1) undeveloped goals, 2) fallacies.

WEFACT is able to fulfill part of this requirement in the sense described in the previous section “Quantitative confidence metrics about an assurance case in a report”, namely by continuous information about fulfilled and not fulfilled requirements as it can be seen in the list box in Figure 15.

Undeveloped goals would probably already be identified when no (sub)process is defined for creating the evidence (GSN solution) for a certain requirement. This uncovers that the respective goal is not fully developed.

Also, certain fallacies can be revealed by WEFACT like for instance technically invalid argumentation strategies, namely when a solution is intended and used in the argumentation which – under the given conditions – cannot be proven because it exceeds the capabilities of the system.

Deficiencies in the GSN argumentation tree itself, however, would be an issue to be detected by the Assurance Case editor or an associated completeness checker tool.

System dependability co-verification and co-validation with WEFACT

This functionality fulfils the “Must” requirement WP4_SDCA_002 System dependability co-verification and co-validation - “The system shall support efficient system or component co-verification and co-validation with respect to multiple quality attributes”.

With the existing features of WEFACT it is possible to enable System dependability co-verification and co-validation. For processes in WEFACT, predecessor and successor processes can be defined. It is, thus, possible to combine verification (or validation) processes related to different quality attributes in a way that they form an efficient combined process which can be automated in the WEFACT workflow. Figure 16 shows an example.

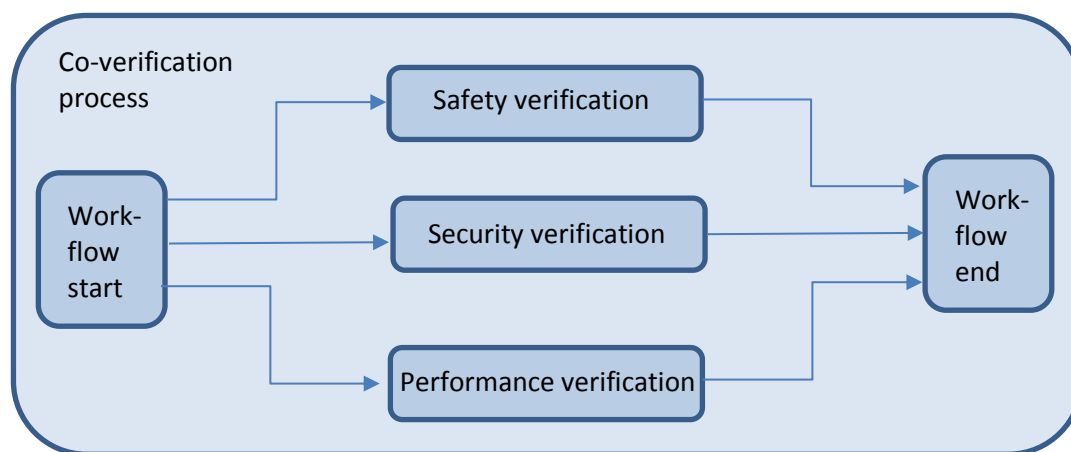


Figure 16. Example for an automated safety-, security- and performance-verification process.

WEFACT instantiated for safety and security analysis processes

This feature implements the “Must” requirement WP4_SDCA_003 - “The system shall allow combinations of safety and security analysis”.

There are two ways to achieve co-analysis:

- (1) Using a combined safety and security analysis tool like FMVEA, or
- (2) Combining separate tools by a WEFACT workflow.

The concept for case (1) is described in the following section 2.1.4.3.

Here, with WEFACT, case (2) is meant. The safety and security co-analysis is achieved by combining separate tools like e.g. Microsoft’s STRIDE-method-based Threat Analysis tool and a commercial Hazard analysis tool, for instance APIS FMEA or some HAZOP tool in a common WEFACT workflow. Figure 17 shows how this conceptually works.

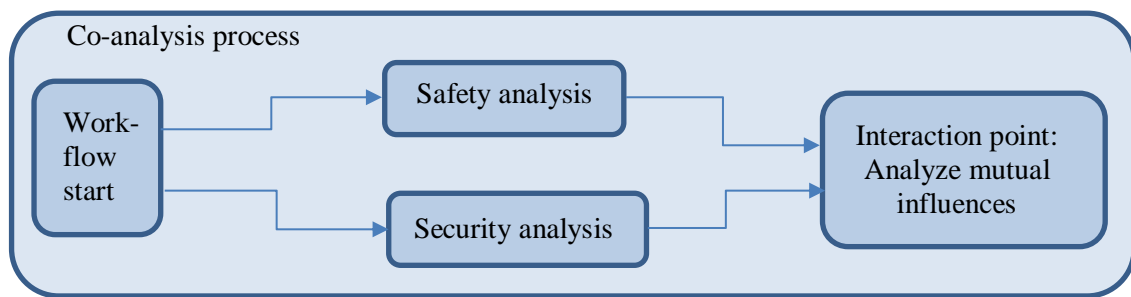


Figure 17. Concept for safety and security co-analysis by combined process inn WEFACT

The part that is not covered in the two separate tools is the interaction point, as it is known from several standards, e.g. SAE-J3061 [24]. This is an analysis by experts to find out whether the results of both analysis processes (safety and security requirements) are compatible, if they are then the result of the co-analysis process is PASS, otherwise the result of the process is FAIL and the analyses must be adapted.

2.1.4.3 FMVEA Tool Concept (*)

Chapter 2.1.1.2 explains the basic principles of the FMVEA method as a combination of the security analysis method of Threat Modelling with the safety-related model-based FMEA (Failure Modes and Effects Analysis). The outcome comprises Threats and Hazards with a risk evaluation. In the following, an outline of the tool currently under constructions is given.

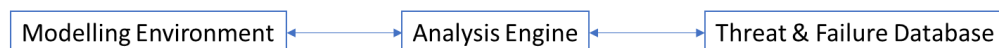


Figure 18. FMVEA tool architecture.

Figure 18 shows the basic architecture of the FMVEA Tool, which consists of three parts:

- the Modelling Environment
 - It is planned to support SysML and Dataflow Diagrams,
- the Analysis Engine
 - It parses the model and matches defined Threats and Failures to the system model,
- the Threat & Failure Database
 - It stores known threats and enables applying them to a system model

The Threat & Failure Database is customer-specific and embedded into the application, i.e. not accessible from outside of the tool. The tool will include an update functionality or the database which avoids overwriting the current database.

The tool applies the known threats and failures from the database to the model, which is for instance imported as SysML model elements. The result are the threats and hazards for the individual system and a risk evaluation.

In the assurance lifecycle, FMVEA can be used in the initial risk assessment for defining:

- the required SIL (safety integrity level) of the safety function needed for risk mitigation, and
- the target security level (SL-T in e.g. IEC 62443 terminology) to be implemented.

In later lifecycle phases, FMVEA is helpful for the repeated safety and security assessments necessary for safety and security validation, when the appropriateness of the implemented concepts has to be validated.

2.2 Dependability Assurance Case Modelling

The main focus of this section is to present an approach to create multiconcern assurance cases. Note that how to address various dependability attributes (i.e. multiconcern) in the system development lifecycle is outside the scope of this section. The focus is how to structure multiconcern assurance argumentation in a way which allows to easily understand interactions and support the maintainability of the assurance case and the system.

2.2.1 Introduction

Assurance cases use a structured set of arguments and a corresponding body of evidence to justify that a system satisfies specific claims with respect to its properties (i.e. safety, security, reliability, etc.).

Basically, an assurance case serves two groups of stakeholders in the assurance process: the one that creates the assurance case in order to claim that a product or system satisfies certain properties; and the one that reviews the claims and evaluates the completeness and soundness of the product or system against the claims. Hence an assurance case provides a structured and reviewable set of artefacts that make it possible to demonstrate to interested parties that the system's requirements have been met to a reasonable degree of confidence. However, a key difference between arguing security and arguing other dependability attributes of a system is the presence of an intelligent adversary.

In addition to that, there is an increasing need to consider the maintainability of a system in the assurance case. Security relies on frequent updates to close newly detected vulnerabilities or adapt security measures to the increased capabilities of an attacker. With a monolithic assurance case, such maintenance requires the complete repetition of the evaluation of the assurance case. In order to ease this burden, compositional assurance will play an important role.

In this section, we investigate how we can create a safety and security assurance case, as the first step towards the concept of multiconcern assurance.

2.2.2 Safety and Security Assurance Case (*)

2.2.2.1 Safety & Security Assurance Case Structure

In D4.1 [1] Section 4.1.4, we surveyed the existing work and proposals for assurance case structure. This is especially important in the cases where the assurance case of the different suppliers combine in order to create a greater assurance case of the system. In D1.1 *Case studies description and business impact* [9], the case study 3 presents a scenario, cooperative autonomous driving, where the assurance case of one vehicle is not enough and we need to assure the combination of the assurance cases of the vehicles involved in the driving action. This case study CS3 has been used as the scenario in which we have made our analysis, research and where we are applying our proposals.

Many existing works suggest to combine the dependability attributes in a unified GSN structure, including safety, security, availability, reliability, etc. However, there are so far no detailed specifications on how to go beyond the top-level split as illustrated in Figure 19. Furthermore, there is no agreed way to combine safety and security assurance cases that are currently accepted by safety and security standards.

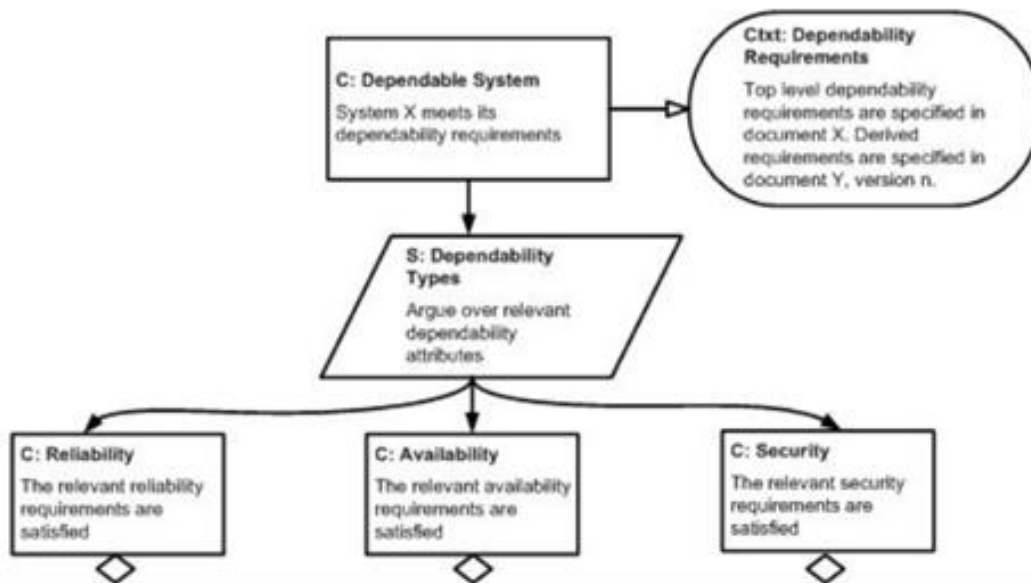


Figure 19. An Assurance Case Fragment

The vision of the AMASS project is to arrive at a unified safety & security GSN structure to specify a combined safety and security assurance case. We propose several views on safety and security assurance case, with different proposals on how to combine them. These proposals can be further refined and discussed to aim at a consensus.

One first attempt is to create a structure of argumentation modules in order to explicitly show the relations of the assurance cases with the components that form the system. We have taken the scenario of cooperative driving to work the possible assurance case architecture specification. In Figure 20 the first attempt of arguments allocation into different arguments modules is shown.

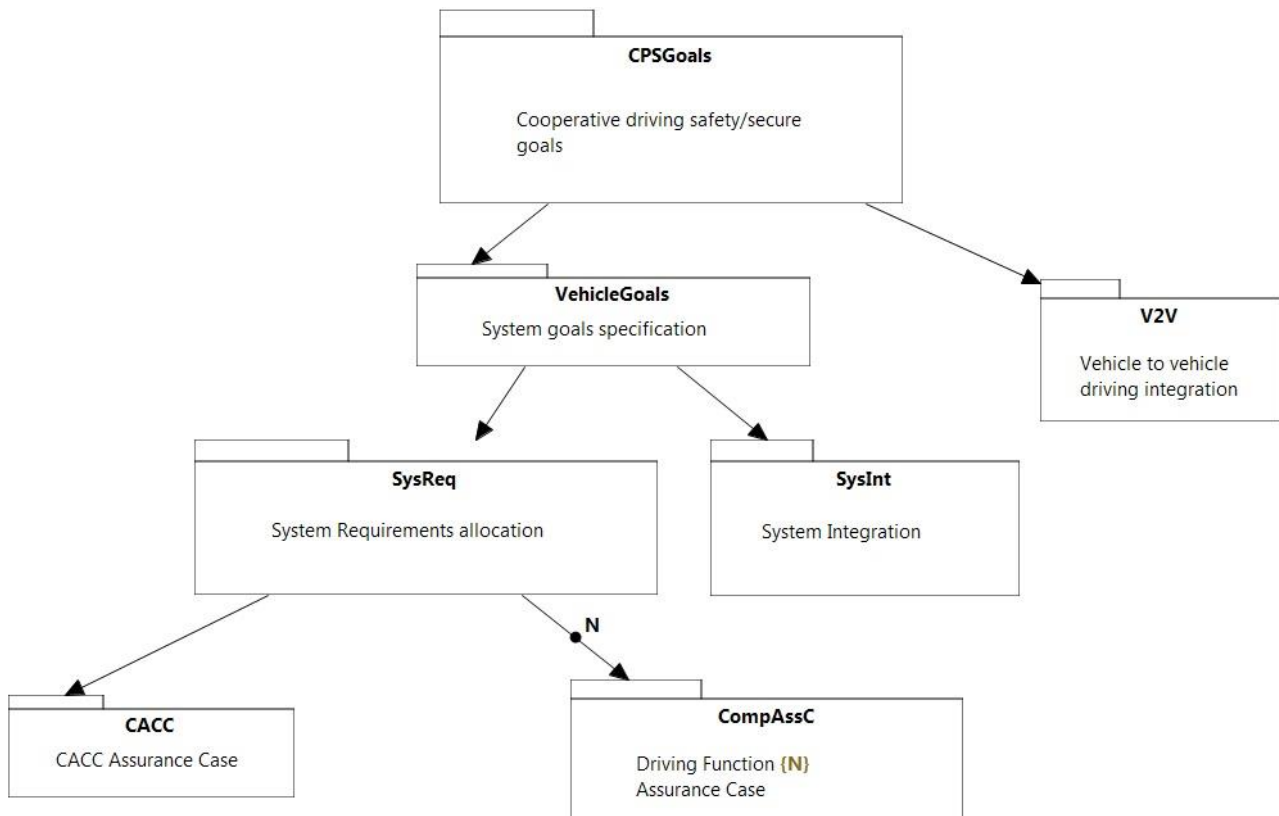


Figure 20. Assurance Case Structure, argument modules decomposition for Cooperative driving scenario

Figure 21 shows a proposal of the multiconcern assurance case structure. The system is assured for multiple concerns such that a set of system goals is developed for all the different concerns. The concern specific goals are the basis for the concern-specific assurance informed of other concerns, e.g., security-informed safety assurance. The system goals are supported by the system requirements developed for all the different concerns. The concern-specific system goals are supported by the safety requirements specific to different concerns. For example, a safety goal may be supported by both safety and security requirements. The system requirements are supported by the assurance case of different components, where each component assurance case includes assurance of that component for the different concerns. The different component concern specific modules support each other. For example, a safety module of one component may be supported by the security module of that or some other component. Interplay of the concerns on all the levels where cross concern trade-off occurs (goals, requirement and components) is handled in the trade-off module.

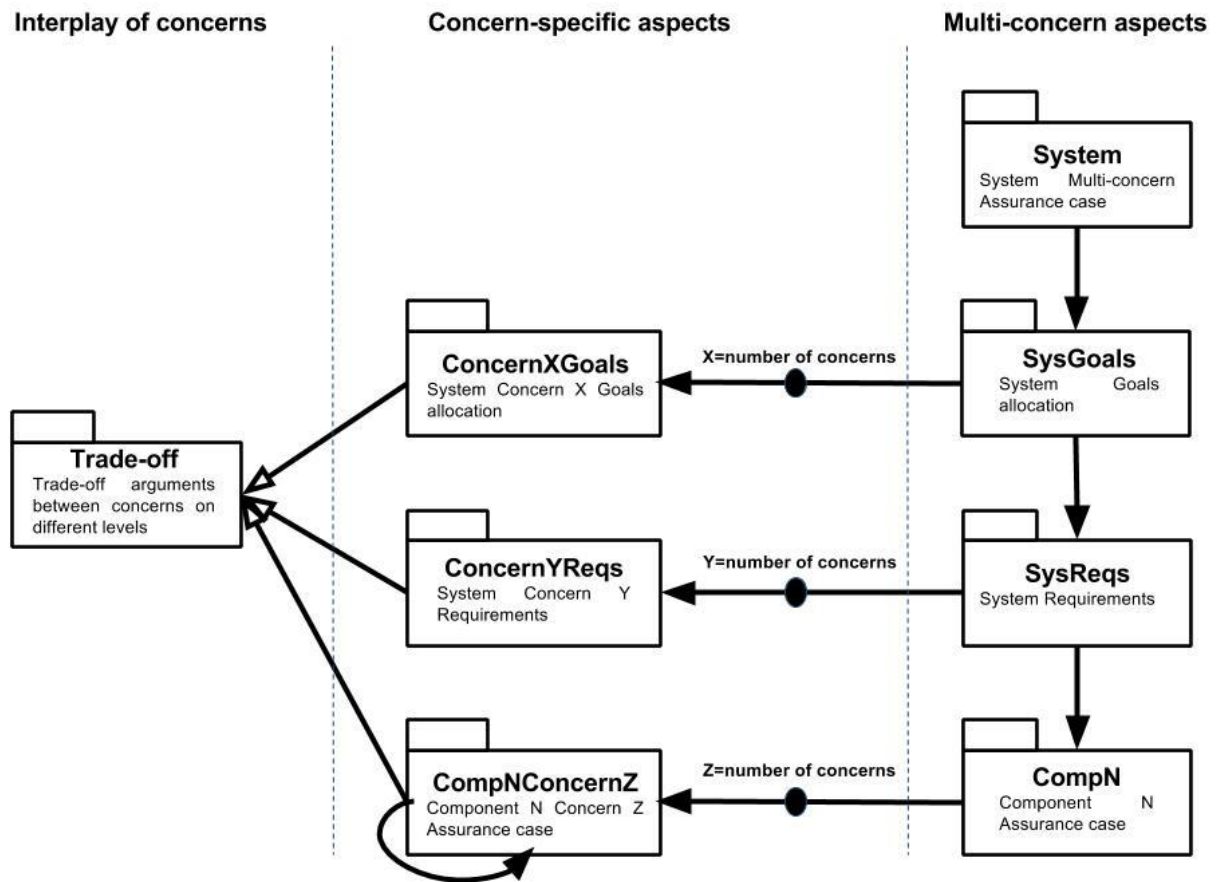


Figure 21. Multiconcern assurance case structure

If we look at the “Trade-Off” argument module, the content will be aligned to the argument pattern presented in D4.1 [1] for making multi-attribute trade-offs and published in [59] and shown in Figure 22.

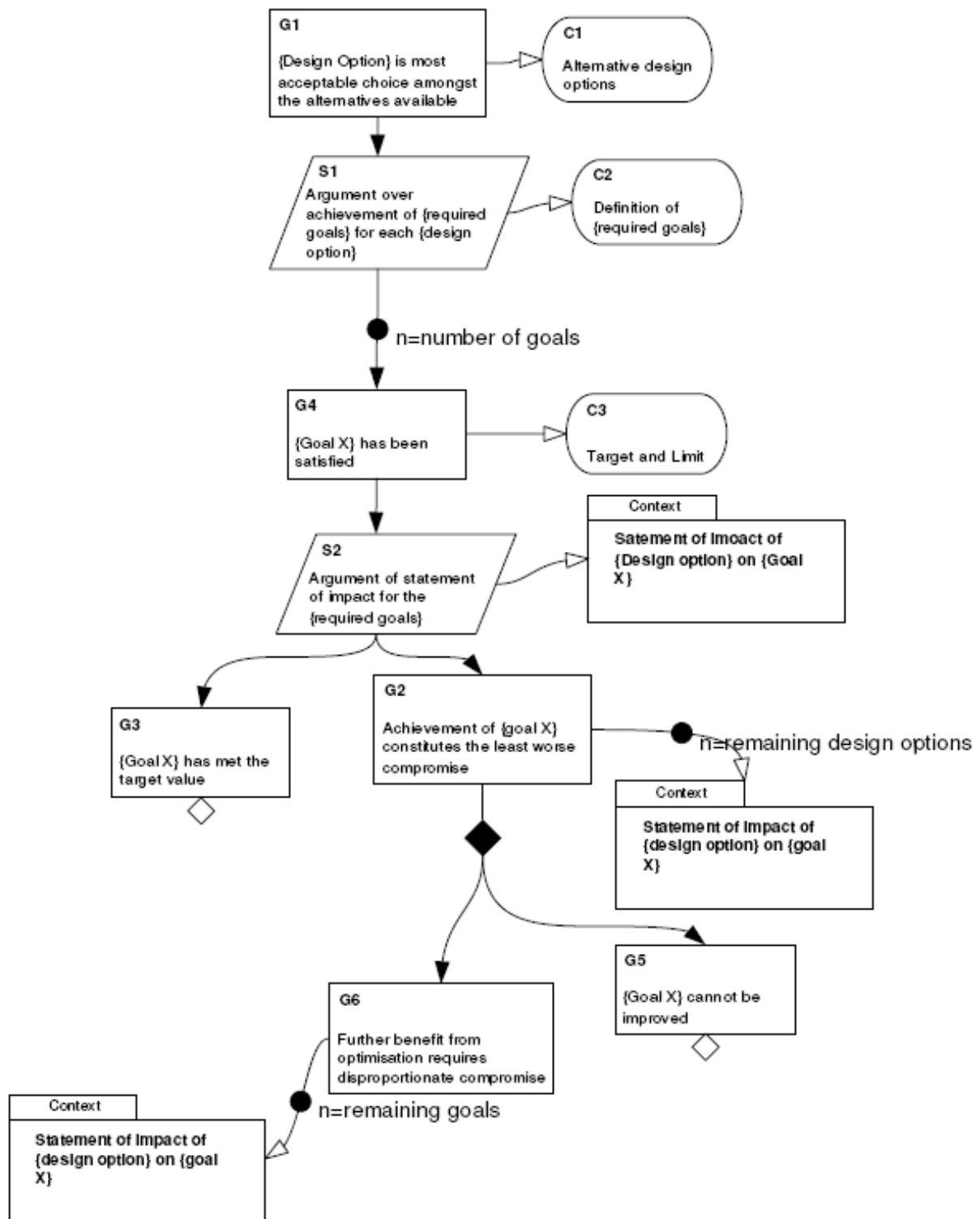


Figure 22. GSN Argument Pattern for making multiconcern trade-offs

2.2.3 Multiconcern Argumentation

When integrating different quality attributes in a unified assurance case, we identify the following generic relationships:

- **Dependency relationship:** The claim A of one attribute depends on the fulfilment of claim B of another attribute. For example, a fail-safe claim of attribute safety depends on the claim that the safety instrumentation system is not tampered of attribute security.
 - There is also a weaker connection, independency, meaning an element does not interfere / depend with other elements, e.g. is patchable /changeable without changing the rest of the assurance case. This relation is probably weaker than dependency, e.g. two goals do not depend on each other.
- **Conflicting relationship:** The assurance measure of attribute A is in conflict with the assurance measure of attribute B. For example, a strong password or blocking a terminal after several failed login attempts for security conflicts with the emergency shutdown for safety. Resolution of such a conflict need to be noted in the Assurance Case.
- **Supporting relationship.** The assurance measure of attribute A is also applicable to assurance of attribute B, such that one assurance measure can be used to replace two separate ones if the attributes are considered and addressed individually. For example, encryption can be used for both security for confidentiality and to check data integrity for safety instead of checksum. This means two goals can be addressed by one argumentation.

These relationships can appear at all levels of an assurance case structure, from safety and security goals to evidence and justifications. Therefore, a relatively simplified approach to multiconcern assurance is to adapt the existing GSN structure to cover the aforementioned three relationships.

We propose the use of an impact relationship concept to address the different relationships mentioned above head. The impact concept is an abstract relationship which takes advantages of proposals made by research groups working in this area.

The dependency relationship between a claim A and a claim B, so that claim A will only be true if claim B is also true can be explicitly specified using “in the context of” notation with a closed white arrow (see Figure 23). Usually this type of relationship in GSN is used connecting a claim within a context; however, here we use it connecting two claims which both need to be supported by evidences.

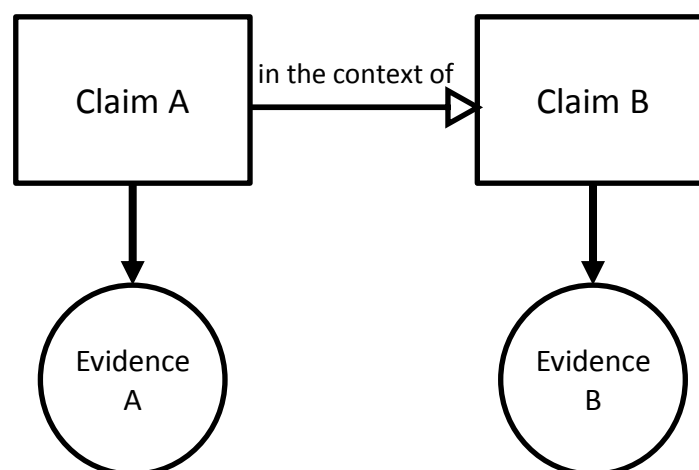


Figure 23. The dependency – impact relationship

Lately, the Object Management Group (OMG) has been working on a standard for Structured Assurance Case Metamodel (SACM) in order to provide a common and structured way for assurance case composition. One of the concepts included in the standard is the AssertedChallenge. “The AssertedChallenge association class records the challenge (i.e. counter-argument) that a user declares to exist between one or more Claims and another Claim”. In the OPENCOS project [51] Deliverable D5.3, it was proposed to have a graphical notation for this concept, a red arrow with a cross in the middle (see

Figure 24). The target of the arrow is a Claim D which is conflicting and will become false if the source of the arrow, Claim C, becomes true.

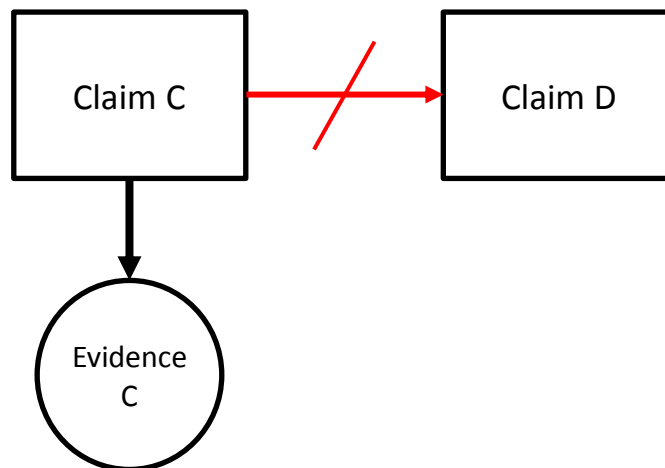


Figure 24. The conflicting-impact relationship

Finally, the supporting relationship mentioned before can be noted using already accepted notation to be used in argumentation patterns. There is a GSN option symbol, which is used to represent choices between lines of argumentation used to support a particular claim [83], for example, Claim E is supported either if Claim F is true or Claim G is true (see Figure 25). It is highly recommended to provide an annotation denoting the nature of the choice made.

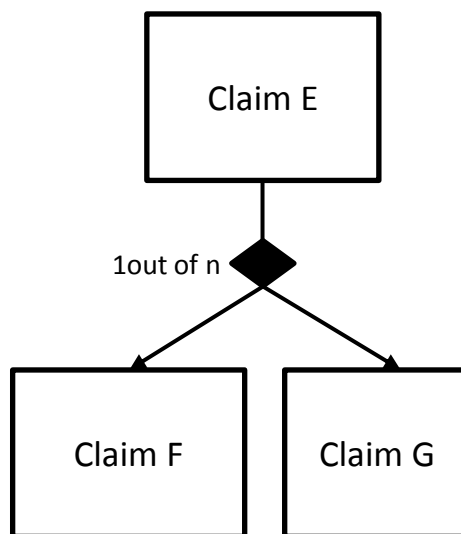


Figure 25. The supporting-impact relationship

In this context, since multiple quality attributes are considered in one picture, more specific questions arise such as:

- Safety is usually non-degradable. What if a non-safety assurance measure has the potential to lower the SIL level?
- Is it possible to reduce 100% availability in order to allow an emergency shutdown when safety issues occur?
- Security update is a solution to the changing threat landscape. But should a security update be delayed because it will compromise availability?
- Can we justify a decision for less expensive component as a trade-off for reliability?
- Security threats evolve in time, as attacks improve, will the security mechanism be effective after some time?

It is important to mention that when dealing with security, effectiveness is valid just for a period of time, for that reason, assurance cases should be checked periodically. We propose to re-create the evidences to ensure they are still supporting the claims and if not, provide an impact analysis and modify the system to ensure the vulnerabilities are well mitigated and/or avoided.

2.2.4 Support for variability management at the argumentation level (*)

As known [72], an assurance case is constituted of claims, contextual information, evidence, and reasoning structures aimed at explaining why the claims are sufficiently supported by the evidence.

As documented in D6.2 [12], these elements may vary (e.g., based on the criticality) and thus, it becomes clear that a single assurance case model does not fit all assurance needs. One size does not fit all. An entire family of assurance cases is embraced.

Thus, additional concepts are needed to enable the systematization of reusable assurance-case-related modelling elements between family members.

- Assurance case-related commonality: indicates the assurance case elements that do not vary and that characterize the family of assurance cases.
- Assurance case-related variability: indicates the assurance case elements that vary and that characterize the individuals within a family of assurance cases.
- Assurance case-related variation point: indicates points of variation where a product element may represent:
 - Assurance case-related options, when for instance an additional branch aimed at developing the argument is not always needed due to optional requirements.
 - Assurance case-related alternatives, when for instance alternative branches aimed at developing the argument can be chosen, due to requirements that can be met in different ways.
- Variability: Two kinds of variability might be identified within a set of assurance cases:
 - Intrinsic: whenever there is more than one argumentation style to support the claims of a particular product-line instance (see, for instance, alternative).
 - Extrinsic: whenever reusable assets (referenced in the assurance case and bound to concrete assets within product-line models such as the feature and reference architectural models) vary.

To enable the systematization of reuse when engineering families of arguments, within AMASS, an orthogonal solution based on the BVR Tool was proposed. The BVR-Tool-based solution permits users to reason about variability in a unified way regarding process, product, and assurance cases. The BVR-Tool-based solution is currently under development in the context of WP6, more specifically in D6.3, final and public version of D6.2.

2.3 Multiconcern Contracts

Contract-Based Design provides the mechanisms to formalise assumptions and guarantees of components and to formally verify that structural decomposition of a system into components is correct, i.e., that the guarantees of the system is assured by the subcomponents provided that the system assumptions hold and that the assumptions of subcomponents are assured by the sibling subcomponents, again provided that the system assumptions hold. If the contracts involved in the contract refinement are related to different concerns, we can talk of a multiconcern contract refinement. This information is typically however hidden and not exploited. It is instead important to highlight the concerns of the contracts to make explicit in the argumentation how they interact, how the proof of the contract refinement guarantees their compatibility, how a mechanism (e.g., a monitor or an encoding function) is used for both safety and security, or how a safety mechanism (e.g., a redundancy) has been introduced to make a security mechanism fault tolerant.

Employment in the AMASS Platform

The AMASS platform will allow to tag requirements, properties, and contracts with concerns (i.e., safety, security, performance, etc.). When the contract refinement involves multi concerns, the related argument fragment will be enriched with a rational explaining how they interact and/or interfere.

The tool functionality supporting this multiconcern contract-based assurance will extend the contract-based assurance described in D3.2 [10].

Usage of multiconcern contracts in assurance via argument-fragment generation

In D3.2 *Design of the AMASS tools and methods for architecture-driven assurance (a)* [10], we have presented how contracts traced with assurance information can be used to automate instantiation of argument patterns stating that the corresponding requirements are sufficiently satisfied. Extending CACM with concern tags for requirements allows us to distinguish to which concern belongs the particular requirement and also the associated contracts and the assurance assets. The concern tags then allow us to generate concern-specific argument-fragments that represent a skeleton of the concern assurance case. The safety engineers are envisaged to continue building upon the generated skeleton. This idea is depicted in Figure 26 for the case of safety and security concerns. Knowing to which concern is a particular argument-fragment related allows us to either build concern-specific viewpoints of the assurance case, or even to build a unified dependability case where the different fragments would support different concern-specific goals.

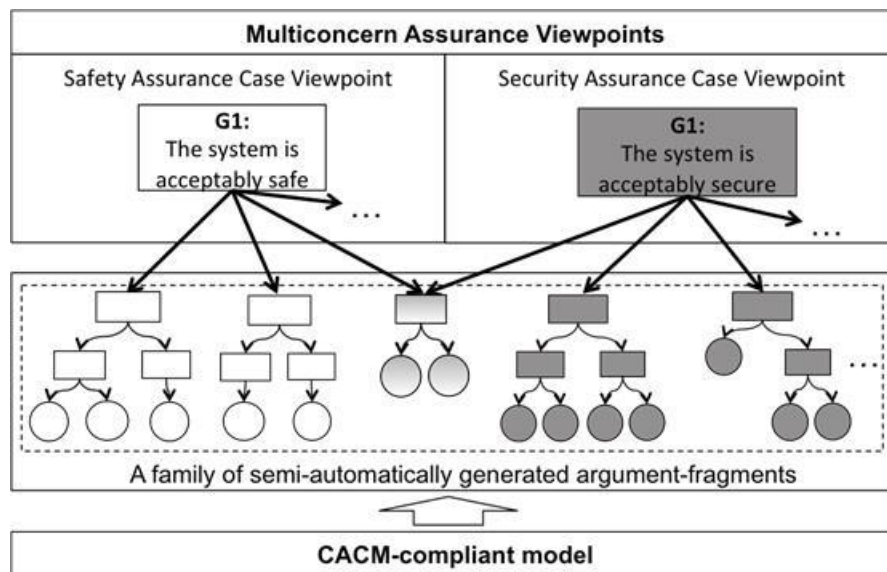


Figure 26. Assuring different concerns via multiconcern contracts, taken from [65]

2.3.1 Abstract functions in the contracts specification (*)

Supports WP4_CMA_002 and WP4_SDCA_002.

Security-related requirements sometimes refer to confidential data or high-level security data or similar attributes associated with data. The semantics of such attributes is often not clear and the actual implementation of the system must give them a meaning, for example saying that data coming from the hospital records are always confidential. Sometimes it is possible that specific component attributes always contain either high-level or low-level security data. More often, this depends on the actual data values so that we can represent it as a function of data (e.g. “is_confidential(record)”). When dealing with architectural design, it is sometimes impossible to give a specific semantics to these functions and it is useful to use “uninterpreted functions” to formalize security-related requirements and analyze their interaction with other concerns. For example, in the work proposed in [68], OCRA has been used to analyze

the contract refinement of a security-related requirement and how monitoring mechanisms can be used to ensure the security in case of component failure.

More specifically, the system-level requirement specified that “no high-level data shall be sent to the external world” formalized as “never is_high(output_data)”. This was structured into a contract assuming that “The user shall switch the dispatcher to high before entering high-level data” formalized as “always ((is_high(cmd_data)) implies ((not switch_to_low) since switch_to_high))”. The contract refinement was proved correct. Moreover, fault-tree analysis confirmed that no single failure of the dispatcher or the monitor could cause the system security failure (to be precise, single point of failures could be the user not respecting the assumption or the component inventing high level data).

2.3.2 Contract-based trade-off analysis in parameterized architectures (*)

Supports WP4_CAC_010.

Parametrized architectures, as defined and developed in WP3, provide the means to analyze the system architecture in different configurations. Each configuration may enable/disable some components, ports, connections, and contracts. Different configurations can be analyzed and compared with respect to different aspects: contract refinement, satisfaction of formal properties, fault tolerance, minimal cut sets, reliability measures. Such an approach was for example followed in the analysis of different configuration of the next generation of air traffic control design [68].

Comparing the different configurations allows the designer to perform *trade-off analysis* and *design space exploration*. Architectural choices are supported by the mentioned analysis results. In particular, the choice whether adding or removing a function (represented by a block or by a contract), enabling or disabling a redundancy, or other similar changes is supported by checking which functional and non-functional properties hold in the different configurations. This trade-off analysis is enhanced by the information about the concern addressed by the different properties and contracts: the analysis provides a direct way to evaluate the impact of the trading-off architectural elements on the multiconcern represented by properties and contracts.

2.3.3 General extensions to contract based multi-concern assurance (*)

OPENCROSS proposed a methodology for structuring argumentation in component-based systems and integrating it to form system-level assuring the interfaces with other components. From the safety perspective, safety is a whole system proper argumentation. In component based design, components are assumed to have a correct functionality just by aty and assuring the correct function of components does not mean that the (composed, integrated) system will remain safe. The context in which the component is going to be integrated is important, and as Ruiz [69] indicated for the SEooC (Safety Element out of Context) perspective, the assumptions of the item can be understood as the context characterization. In addition, to support safety assessment, failure behaviours of components, and their behaviour in the presence of failures, must be defined.

Similarly, to reconcile the bottom up component-based approach with top-down hazard and safety analyses, SafeCer proposed that generic evidence about components properties is linked with specific top-down safety requirements of one or more systems in which the component is used [81]. One method for presenting a safety case is via a safety argument (a logical decomposition arguing about the safety of the system) which is supported by evidence (e.g. software testing results or static analysis). The generic evidence about the component properties are captured in the component argument fragments.

In AMASS we propose to merge both approaches to take advantage of both OPENCROSS and SafeCer contributions to compositional assurance. In particular, we build upon the compositional assurance methodology proposed by OPENCROSS and use it as the basis for structuring the assurance case. Furthermore, for ensuring the validity of the component context, we take advantage of the formal

validation of multi-concern contracts envisaged in SafeCer and improved in AMASS, in the context of WP3. This supports conceptually WP4_CMA_003.

Building upon the OPENCROSS and SafeCer usage of the argument contracts for compositional assurance, we integrate the argument contracts within the AMASS multiconcern assurance case structure proposed in Figure 21. We use argument contracts to capture the interplay between the concerns by creating argument contracts between concern specific modules. In particular, we make the argument contracts both between the hierarchical levels (e.g., between goal and requirement modules) and between the different concern modules on the same level (e.g., between different concern specific modules about requirements), as depicted in Figure 27.

To capture the interplay of concerns in argument contracts, we should capture and describe the generic relationships for multiconcern assurance identified in Section 2.2.3. Capturing the supporting relationships across concerns can be done via the existing contract argumentation pattern proposed by IAWG and presented in D4.1 [1]. For capturing the dependency and conflicting relationships in argument contracts, we adapt the existing pattern to capture the information relevant for the new relationships.

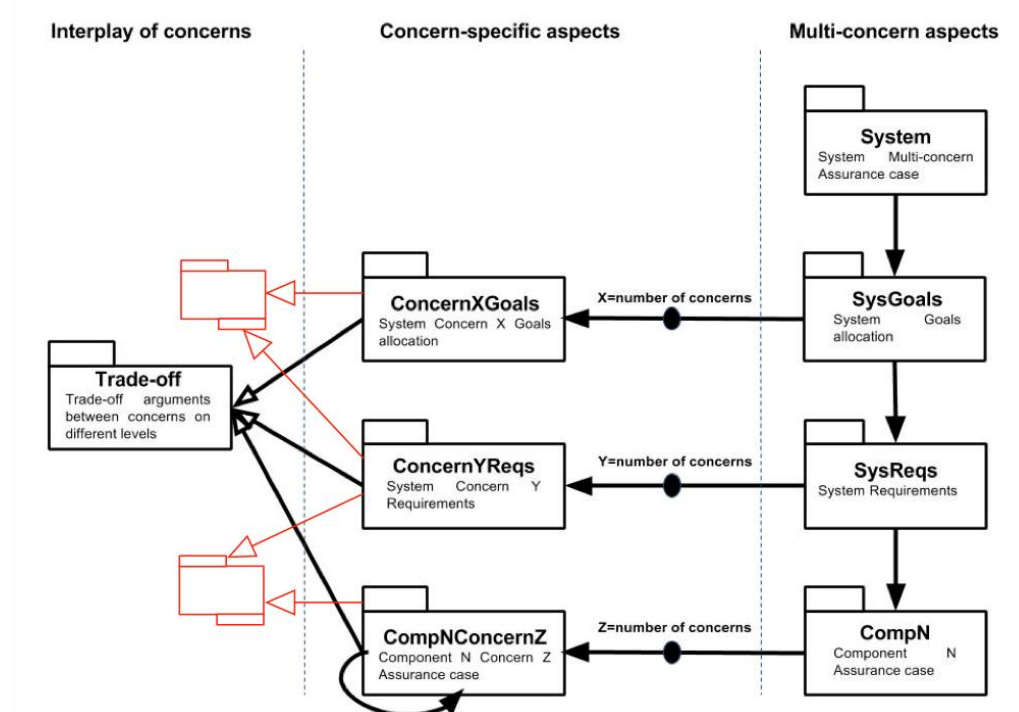


Figure 27. Capturing interplay of concerns in argument contracts

Figure 28 presents the safety case contract argument pattern for capturing the conflicting relationships across concern-specific modules. We relate the affected goal with all the conflicting goals, and for both include the inherited contexts, so that the conflicting relationship can be better understood. Furthermore, for each conflict, there is a choice to either resolve it and point to the trade-off argument discussing its resolution, or justify why the conflict does not require resolution. Since the resolution of the conflicts may imply the revision of the initial goals, the versioning of the goals can be used to indicate which versions were in conflict and needed resolution.

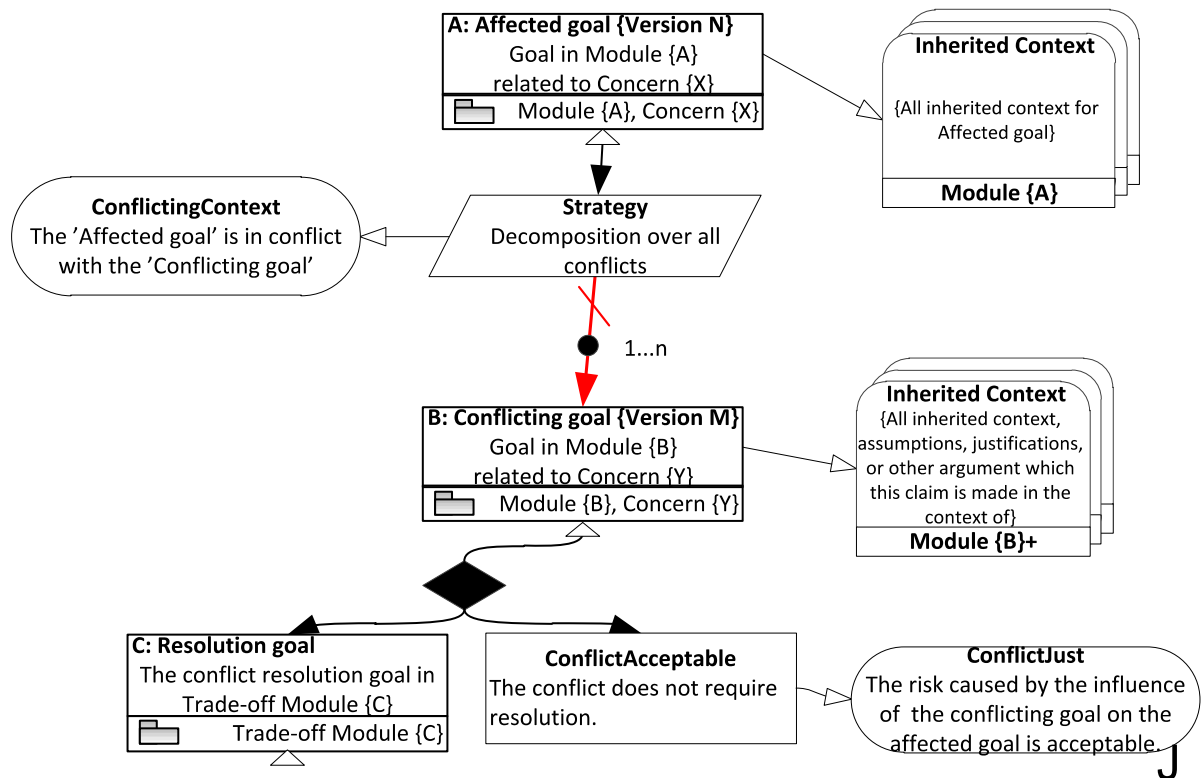


Figure 28. Safety case contract argumentation pattern for capturing the conflicting relationships across concern-specific modules

The dependency relationship follows a similar pattern as the supporting relationship where all dependencies are identified and explained via contexts of the related goals. The adapted safety case contract argument pattern for dependency relationship is shown in Figure 29.

The initial argument contracts for some generic relationships can be derived from the results of formal verification via component contracts. For example, the supporting relationships between different safety case goals can be identified when the same formal property or a component contract is used to formalize two different requirements of two different concerns. The conflicting relationships can be identified in case of identified inconsistency between different component contracts or formal properties related to different concerns. Finally, the dependencies can be identified between different component contracts when guarantees of a particular concern-specific component contract are needed to satisfy assumptions of another concern-specific component contract. By analysing the results of component contract refinement and property consistency checking we can automatically identify and generate skeleton argument contracts for some of the relationships relevant for multiconcern argumentation. Since not all relationships can be identified based on the results of component contract checking, the remaining multiconcern relationships need to be identified and captured in the argumentation contracts.

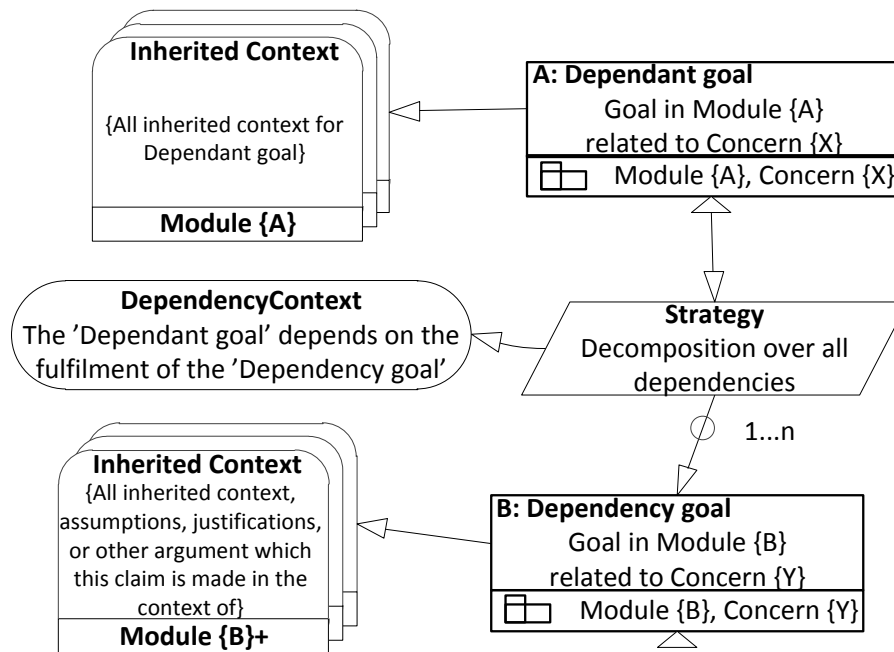


Figure 29. Safety case contract argument pattern for dependency relationship

2.3.4 Contract-based trade-off analysis with the Analytical Network Process (*)

Supports WP4_CAC_010.

As discussed in conceptual section the ANP allows to evaluate the impact of factors on the bottom of the hierarchy to the top dependability attributes which are set by our desired goal. This gives a design feedback that changes in design of which component or sub-system will enable us to achieve our desired goal. It also tells us the relative impact of all factors that are affecting that component or subsystem such as failure causes or vulnerabilities, so we know which security strategy or component with increased reliability (may be redundancy) can help us to achieve this goal, which will be verified by analysis for the impact on dependability attributes. All the appropriate design possibilities can be evaluated to find if the safety critical system requirements meet and for their impact on other non- safety attributes for tradeoff analysis.

Since ANP represents information in such a way that the impact of factors at bottom (say level 0) will be shown at all upper levels e.g. level 1, level 2, and dependability attributes at top, this makes contract based analysis easier.

3. Design Level

3.1 Functional Architecture for AMASS Multiconcern Assurance

3.1.1 Overview

Multiconcern assurance is the way assurance must be done if more than one quality attribute of a critical system has to be taken care of. As nowadays systems become more networked and are therefore more vulnerable to cybersecurity attacks, the discipline of safety-critical system engineering is learning that, in particular, security plays an essential role in order to assure safety. Other quality attributes (e.g., reliability, maintainability, etc.) play a significant role.

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

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

Figure 30 shows the multiconcern assurance process. As will be explained below, it includes product as well as process assurance.

In Figure 2, also a combined security and safety process is presented. While the classical HARA is used for safety-related hazard analysis, for security analysis the STRIDE approach is applied, and the SAHARA method is used to assess the safety hazards induced by security threats. As a whole, the approach of Figure 2 can be inserted in Figure 30 as a combined analysis method, i.e. instead of the FMVEA.

It shall be mentioned that Figure 30 contains also the multiconcern assurance activities for process assurance. The path splits between product and processes in the safety & security requirements: They are made up from system and process requirements. And there are also trade-offs between process requirements. E.g. the security-related process requirement “frequent updates” contradicts the safety requirement of “safety validated system in continuous operation”. To solve the entire problem, a solution in the product may be necessary (e.g. a cold-standby system to use during system software update and subsequent safety validation). Similarly, the mitigation measure for a system safety risk may be a process (e.g. a preventive maintenance process if useful lifetime of components is shorter than system lifetime).

Summarizing we may state that product and process assurance have to be done in parallel, there are equally trade-offs between quality attributes for both the product and the processes, and they are even interlinked when it comes to mitigation measures.

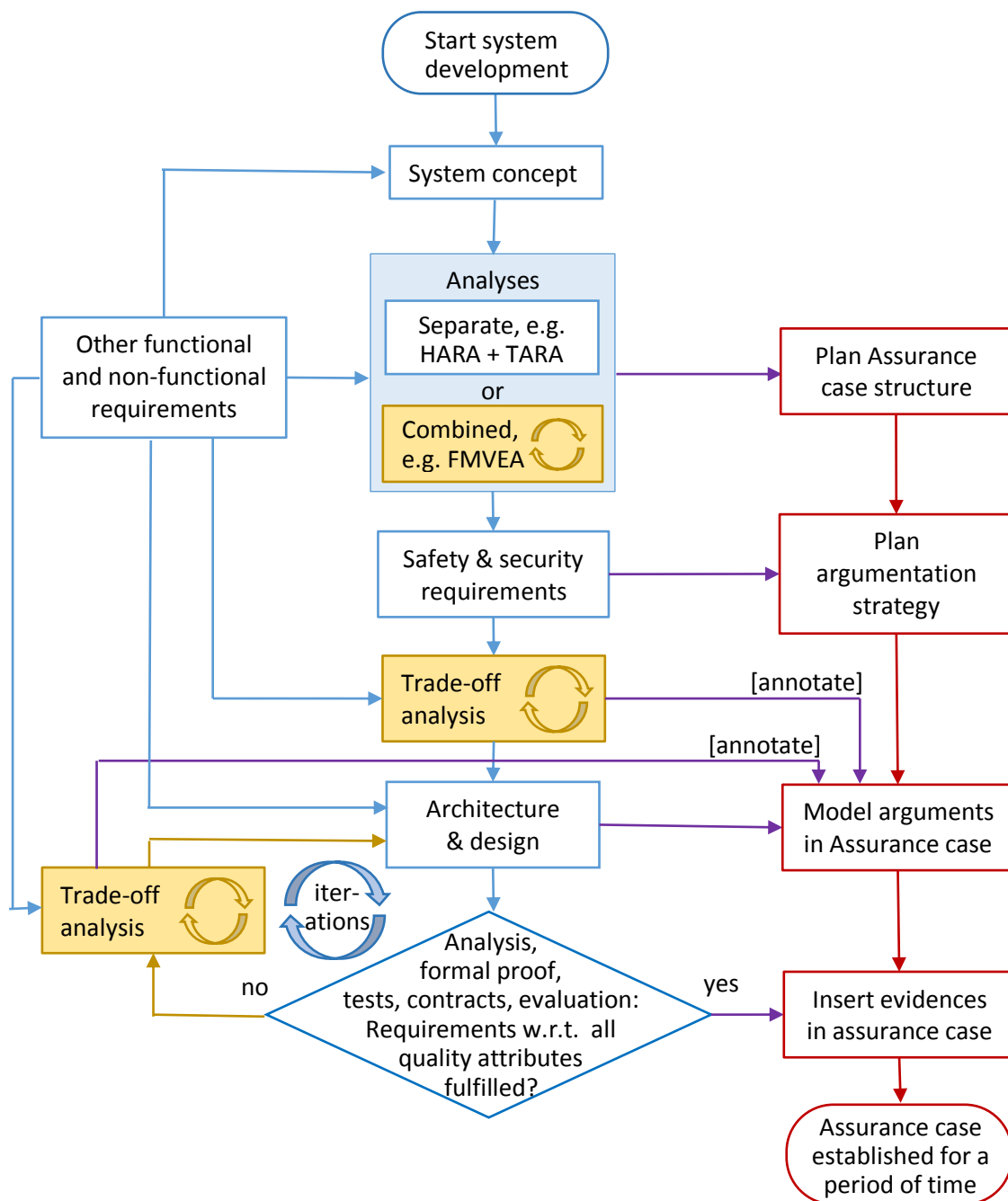


Figure 30. The multiconcern assurance process

There are mainly three points in the system development and assurance case development lifecycle, at which multiconcern-aware methods can provide an advantage compared to single concern assurance:

- the initial hazard and risk analyses,
- the architecture and design process where multiconcern-aware design decisions can help avoiding corrections afterwards, and
- the redesign process when multiconcern validation reveals insufficient achievement of the quality-attribute-specific targets.

Assurance is always based on a defined design and defined processes. This means that, when the evidences for the arguments are created, there is no negotiable trade-off anymore which could influence the argumentation. Arguments supporting the different quality attributes are treated separately on the basis of

the chosen architecture and design. As mentioned above, it is nevertheless possible that one proof delivers the evidence for more than one quality attribute.

Process Assurance

It shall be noted that also high-quality processes (during development as well as operation and maintenance) are indispensable in order to guarantee sufficient safety and security. Here we have trade-offs between attributes, too. As an example, safety requires restrictions w.r.t. updates of the system because, according to functional safety standards, any change in the system requires safety re-validation. On the other hand, security demands to install updates whenever one is available. At the end, we need a similar assurance process model as depicted above for the product. We just analyse processes instead of products and consider requirements to processes instead of product requirements. Process design takes the role of product architecture and design, and the decision whether the process implementation meets the various requirements is also analogous to the product assurance lifecycle. As already mentioned above, the multiconcern assurance process depicted in Figure 30 comprises, therefore, product as well as process assurance.

Design for Multiconcern Assurance

Multiconcern assurance is basically covered by methods which are used as well for single concern assurance. No additional design block needs to be introduced to cover the needs of multiconcern assurance. The following subsections discuss the three main functions assigned to WP4, as depicted in Figure 31 in more detail.

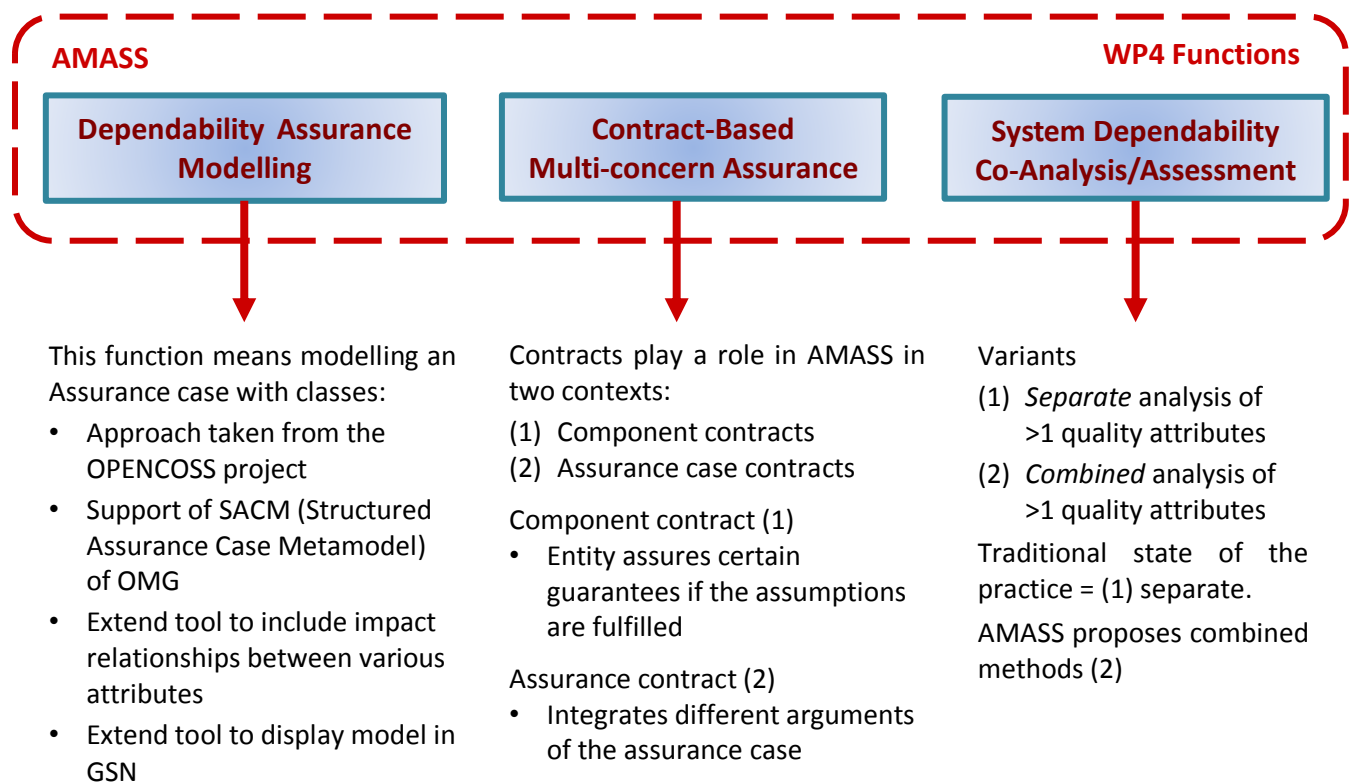


Figure 31. The three WP4 functionalities with explanations

Figure 32 shows the “Assurance Case Specification” basic building block, responsible for the assurance case creation and management which has evolved from the previous one defined in D2.2 [11] .

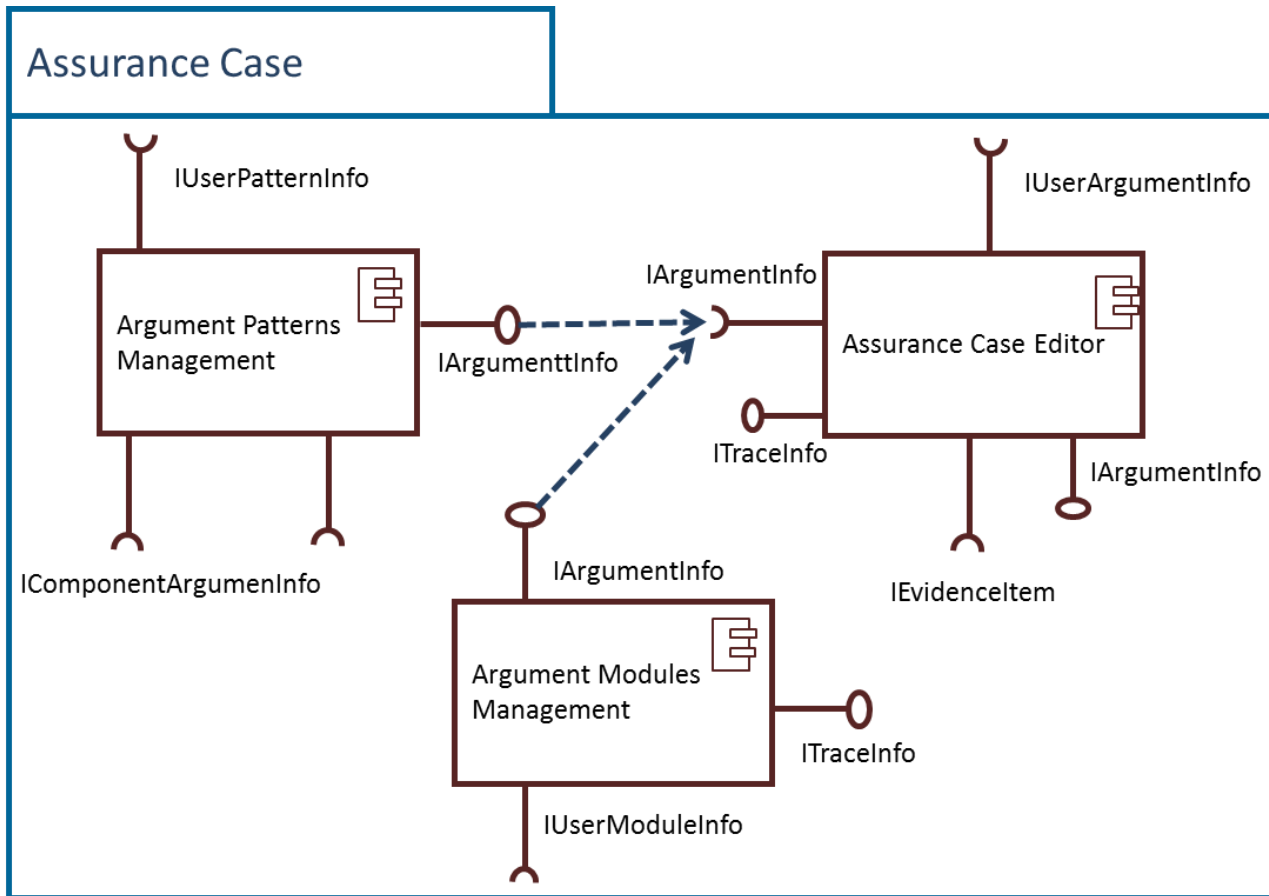


Figure 32. Assurance Case Specification

3.1.2 Dependability Assurance Modelling (*)

The Assurance case in AMASS is modelled in SACM (Structured Assurance Case Metamodel) (cf. Figure 33). The implementation of the respective tool, the OpenCert Assurance Case Editor has been done in the first iteration of the AMASS platform, i.e. it is a Basic Building Block. Details can be read in D4.4 [8] and the respective user documentation. The tool has been derived from the OpenCert safety case editor, originally called Prossurance, and enhanced with an extended vocabulary for multiple concerns and a graphical presentation of the assurance case in GSN notation. A specific extension to support impact was needed in order to cover multiconcern aspects. It can, however, make sense to include annotations related to multiple concerns treated in the arguments in order to make certain argumentations or design decisions better understandable.

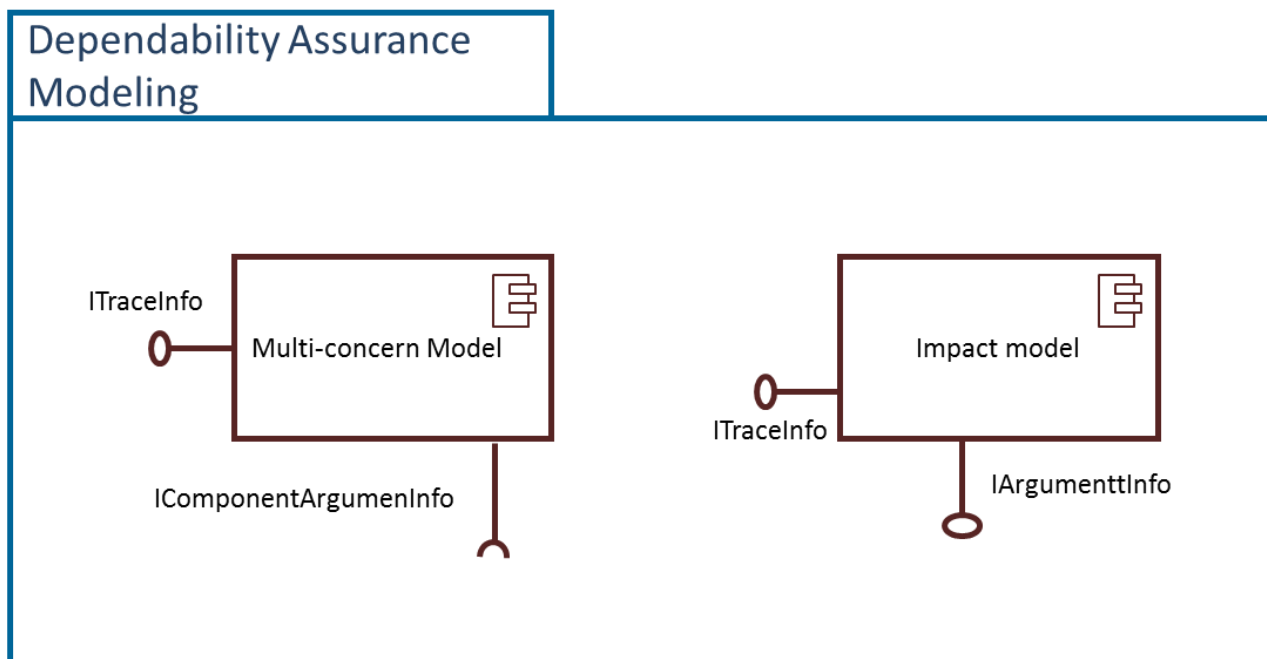


Figure 33. Dependability Assurance Modelling block

3.1.2.1 Support specification of variability at the argumentation level (*)

As mentioned in section 2.2.4, a solution based on the BVR Tool was proposed and is currently under development in the context of WP6, so details on the design can be found in D6.3.

3.1.3 Contract-Based Multiconcern Assurance (*)

As explained in Figure 31, we have to cover two aspects of multiconcern contracts.

Assurance contracts help bundling two or more argument modules of the Assurance case with inter-dependence between them. This is supported by OpenCert. When a claim about a specific concern in one argument module needs to reference an argument, which is being explained and supported in another argument module (which in turns deals with another concern), this dependency is collected in the assurance contract.

Component contracts enable assume-guarantee relations of components with their environment. Formal contract reasoning allows the re-use of the assurance case fragment of the component in its new application context and, thus, enables cost savings. Basically, if multiple concerns shall be treated, they mean just some more properties in the contract. OpenCert supports Assurance case fragments, and it is planned to use the CHES modelling environment for component contracts. It is, however, questionable whether one modelling language is equally adequate for presenting safety and security properties within one contract (for instance, temporal logic is good for safety contracts but less for security contracts). It may, therefore, be necessary to use safety and security contracts in different notations in parallel. The concrete application in use cases is expected to bring clarification.

3.1.3.1 Abstract functions in the contracts specification (*)

Supports WP4_CMA_002 and WP4_SDCA_002.

Abstract uninterpreted functions will be represented in CHES as UML FunctionBehaviors, which are functions that do not modify any objects or external data, and can be used also to represent primitive functions such as arithmetic operations. FunctionBehaviors declared in the CHES model will be used to enrich the constraints such as FormalProperties with uninterpreted functions. The CHES components for

editing FormalProperties and for importing/exporting OCRA models will be extended to support such functions.

3.1.3.2 Contract-based trade-off analysis in parameterized architectures (*)

Supports WP4_CAC_010.

The support for the Contract-based trade-off analysis requires the possibility to configure the system designing a parametrized architecture, i.e. a set of architectures specified using parameters so that static architectures can be instantiated by specifying the values of the parameters. With such parameters, it is possible to vary the number of components, the number of ports, the connections, and the static attributes of components to instantiate. In CHESS, this information can be mapped to the SysML multiplicity attribute, without the need to extend the CHESS model. The variables used to describe the multiplicity (e.g. the variable N of 1..N that may express the boundary of the elements to instantiate) can be represented by a static FlowPort and can be assigned using the Constraint element. To interpret the multiplicity and the involved parameters, CHESS will require some updates. The formal language used to edit the formal properties/contracts/constraints will be extended and the OSS importer/exporter will be enhanced to map the information about the multiplicity in the CHESS model with its corresponding imported/exported textual specification. Finally, the backend tool OCRA will be also updated to support the V&V of parametrized architectures.

3.1.3.3 Contract-based trade-off analysis with the Analytical Network Process (*)

Supports WP4_CAC_010.

Threat and failure propagation state machine models from CHESS will be used for generation of failure-maintenance and attack-recovery state transition model appropriate to the ANP tool. The tool will provide analysis of this model for evaluation of dependability attributes in concern such as safety, security, reliability and availability. For analysis we will use Markov and Simulation (Monte-Carlo). The results from this analysis will be input to final Analytical Network Process metric for evaluation.

3.1.4 System Dependability Co-Analysis/Assessment (*)

These are the promising novel combined methods which lead to potential improvements w.r.t. quality, cost and development time. As depicted in Figure 30, two combined methods are in scope - Co-Analysis and Trade-off Analysis. They are marked as light brown rectangles in Figure 34.

Co-Analysis is the combined analysis of more than one quality attribute, taking into account the inter-dependencies between them. Most prominent and already understood in the standardization scene is safety and security co-analysis. Here, the application of merely single-concern oriented analysis may pose the risk to overlook safety-related hazards which are caused by security breaches. An example for this method is FMVEA (Failure Modes, Vulnerabilities and Effect Analysis), which is shortly described further below in chapter 4.

Trade-off Analysis is a way to analyse the impact of failures and cyberattacks on overall safety and security of a given system, and to use this information as a decision support in the architecture and design phases. A respective novel approach using the Analytical Network Process (ANP) is described in detail in section 2.1.2. The expectation is to strongly reduce the number of iterations (see Figure 30) needed until an appropriate design that sufficiently satisfies safety and security requirements is found by applying matrices-based computations. The method is currently under development and more information is expected for the second iteration of this concept deliverable.

As it was explained in Section 2.1.3, process-related co-assessment is conceptually conducted via SiSoPLE. Figure 34 depicts the designed solution. SiSoPLE is supported by the integration of two tools: EPF-C and BVR tool [54]. The details regarding EPF-C & BVR Tool integration are expected to be given in D6.2 [12] and D6.5 [13]. Once a SiSoPL (regarding the planning processes) is modelled in EPF-C&BVR Tool, properly

tailored multi-concern single processes can be derived and used to feed the argument fragment generator (which implements MDSafeCer [49][48]). The generation might be needed in case an explicit argument is required to explain why the tailored process is compliant to the standards pertaining to the different concerns. The generated argument can then be visualized via the Assurance Case Editor.

MDSafeCer is also expected to be implemented in the context of WP6. The multiconcern knowledge base in terms of multiconcern method content as well as cross-concern commonality identification is being developed within WP4 and is being modelled within EPF-C & BVR Tool.

A properly tailored process plan obtained by configuring the SiSoPL can also be used to feed WEFACT in order to execute the plan.

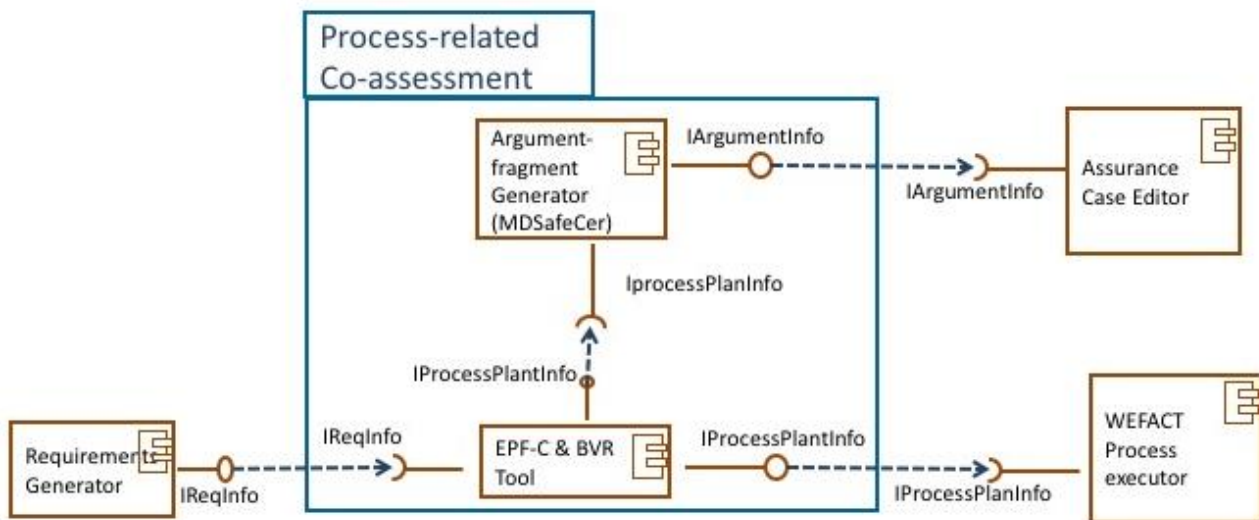


Figure 34. Process-related Co-assessment

3.1.4.1 System dependability co-analysis via ConcertoFLA (*)

Figure 35 shows the basic building block of the system dependability co-analysis, responsible for generation of security informed safety fault tree. To perform system dependability co-analysis, the component-based architecture of a system as well as the input/output failure behaviour for each component is modelled using CHESMML. The failure behaviour is further elaborated, to address different concerns, using the error model state machine, as explained in Section 2.1.4.1. The analysis engine (ConcertoFLA) generates the failure propagation paths, which are then utilized by the fault tree generator to generate fault trees. Finally, the fault tree editor is utilized for visualizing the fault trees.

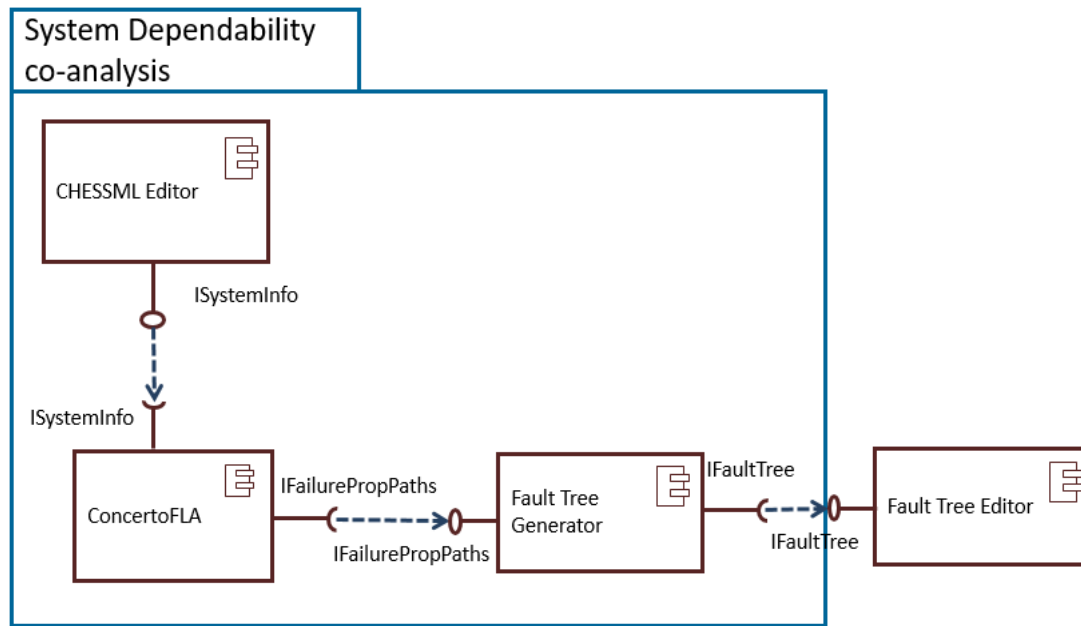


Figure 35. System Dependability co-analysis

3.1.4.2 WEFACT Tool Concept (*)

As mentioned in section 2.1.3.2, WEFACT V2, as delivered in D4.5 [6], contained all features which allow using the conceptual extensions described here in section 2.1.3.2. A description of the tool WEFACT can be found in D4.5 and – with details about the handling – in the user manual provided with the executable. For details like directory and file names see D4.5.

3.1.4.3 FMVEA Tool Concept (*)

This chapter gives first information on how the FMVEA tool will be implemented in the form of user interface mock-ups. A detailed description will be delivered with the executable itself with D4.6 [5].

As described in section 2.1.4.3, the FMVEA tool consists mainly of three components, the Modelling Environment, the Analysis Engine, and the Threat & Failure Database. Figure 36 shows the tool architecture in more detail.

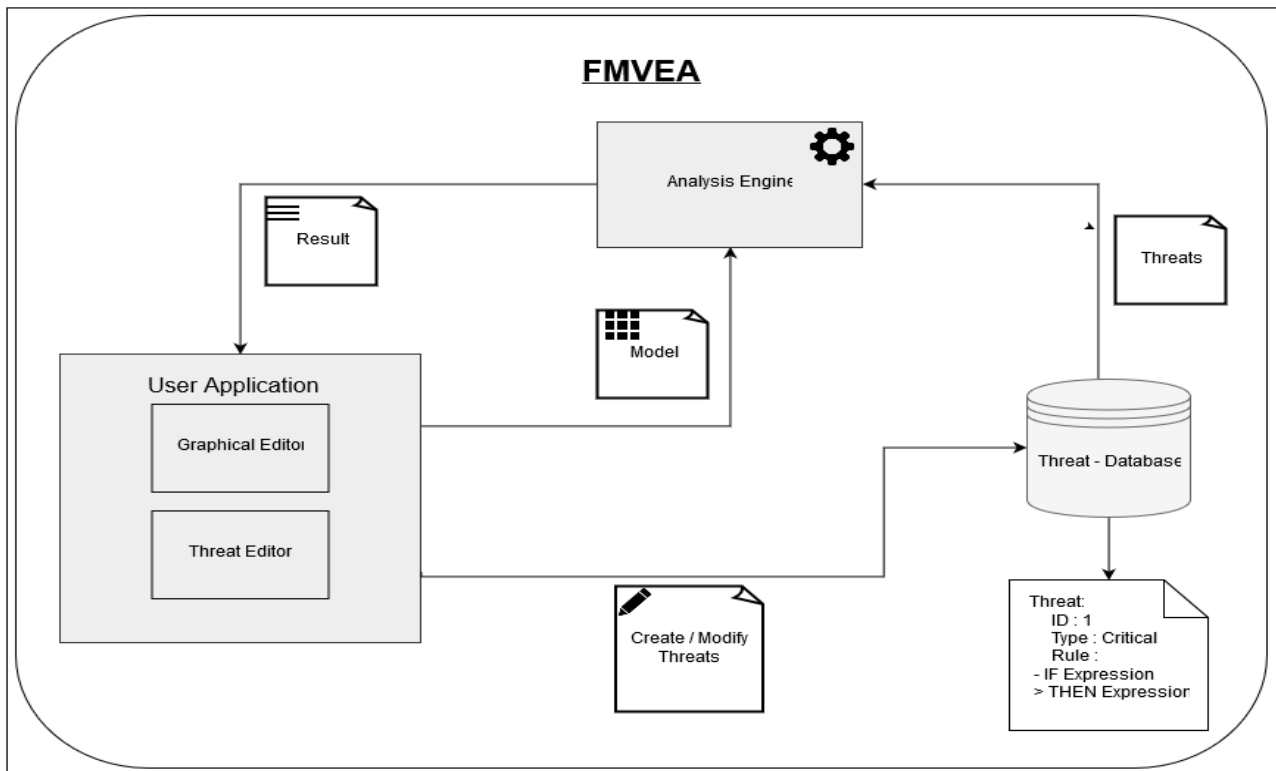


Figure 36. The FMVEA architecture

Figure 37 shows a mock-up of the designed user interface for the model editor.

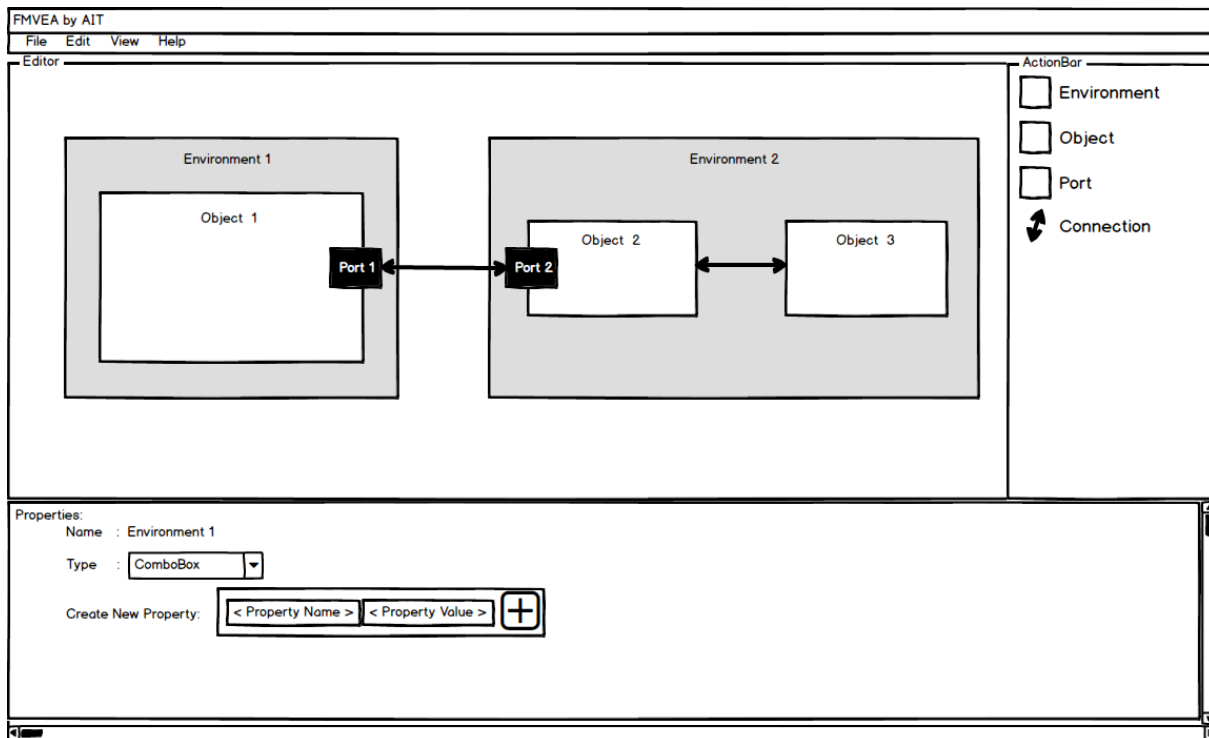


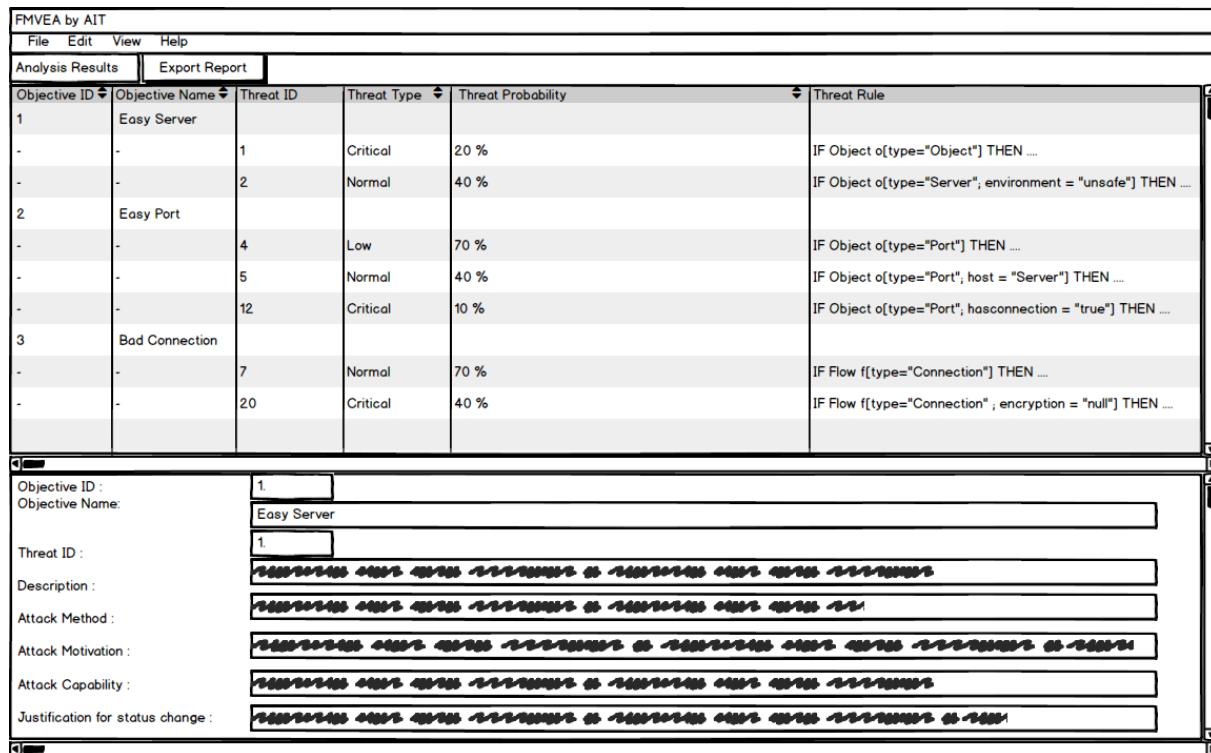
Figure 37. Mockup of the FMVEA model editor including properties definition

The model editor allows defining the nodes, here called objects, and connections. Additionally, properties can be specified.

The threat database is collected based on knowledge and experience over time using the threat editor, and the threats are stored in the form of rules.

The analysis engine is started from the respective menu item, applies these collected threats to the system model and yields the system-specific individual threats and failures including a risk assessment as outcome.

The result is displayed on the screen, as shown in Figure 38 below.



Objective ID	Objective Name	Threat ID	Threat Type	Threat Probability	Threat Rule
1	Easy Server				
-	-	1	Critical	20 %	IF Object o[type="Object"] THEN ...
-	-	2	Normal	40 %	IF Object o[type="Server"; environment = "unsafe"] THEN ...
2	Easy Port				
-	-	4	Low	70 %	IF Object o[type="Port"] THEN ...
-	-	5	Normal	40 %	IF Object o[type="Port"; host = "Server"] THEN ...
-	-	12	Critical	10 %	IF Object o[type="Port"; hasconnection = "true"] THEN ...
3	Bad Connection				
-	-	7	Normal	70 %	IF Flow f[type="Connection"] THEN ...
-	-	20	Critical	40 %	IF Flow f[type="Connection"; encryption = "null"] THEN ...

Objective ID :	1
Objective Name :	Easy Server
Threat ID :	1
Description :	...
Attack Method :	...
Attack Motivation :	...
Attack Capability :	...
Justification for status change :	...

Figure 38. The FMVEA results

Additionally, a function for exporting reports is planned.

The final implementation will be described in D4.6 (due in August 2018).

3.2 AMASS Multiconcern Assurance Metamodel

The metamodel is a review of the Argumentation metamodel from OPENCROSS project [51] and it is an extension of the SACM metamodel of OMG.

This metamodel is used to store argumentation patterns and assurance cases referring a specific system in a certain context. The concepts available in the Assurance Case Metamodel are compatible with the SACM metamodel version 2 which at the time of writing this deliverable is still a draft.

The Assurance Case metamodel provides the assurance case specification capabilities, connects to contracts elicitation, both argumentation contracts and connections to system architecture design contract. It identifies the links between argumentation-related entities and the other parts of the CACM, so to support the multi concern co-assurance approach.

3.2.1 Elaborations

This section addresses the modifications that have been addressed in the Assurance Case metamodel with respect to the version presented in AMASS D2.2 [11]; the modifications have been applied to cover the conceptual approaches discussed in chapter 2.

Some of the changes made in the Assurance Case metamodel have been due to feedback from the first prototype, and proposals made in chapter 2.

On one hand, some of the notations made about the impact relationship were available before. However, there were some constrains (OCL) such as *AssertedContext* which was only accepted for connecting claims objects with *informationElement* objects (contexts). Now the *AssertedContext* entity can link any *ArgumentationElement* class. This way it is allowed to connect claims together, so the dependency relationship mentioned in section 2.2.3 is possible, but also to connect *ArgumentPackage* together, which is especially useful when specifying the Argument Case structure in form of argument modules.

In this update, we have specially focused on changes due to connection with other metamodels and how we have extended the metamodel in relation with multi concern, compositional assurance and use of assurance patterns.

Relations with other metamodels from AMASS CACM

The argumentation metamodel is connected with the evidence model through the class “ArtefactElementCitation” whose cited artefact is the “Artefact” class from the evidence metamodel, see Figure 39.

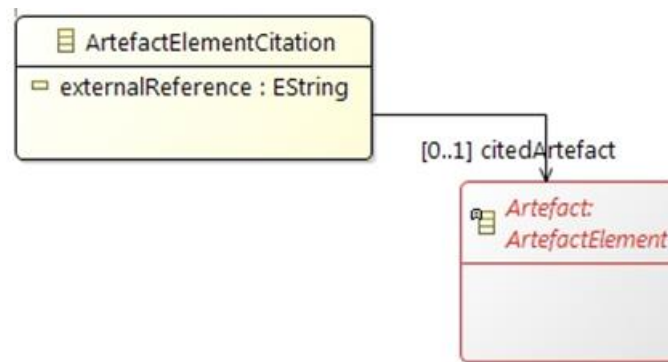


Figure 39. Relation with other metamodels

System Component Metamodel for Architecture-driven Assurance

The SystemComponentMetamodel developed in WP3 (see D3.2 [10]) is extended in order to support the notion of concern. In particular, according to what has been elaborated in chapter 3.1.3, there is the need to know the concern addressed by a given requirement and contract. The extensions, shown in Figure 40, are the following:

- **ConcerEnum:** new enumeration entity, representing the possible concerns.
- **Requirement:** the attribute *concern* typed with ConcerEnum has been added to represent the concern addressed by the requirement.
- **FormalExpression:** the attribute *concern* typed with ConcerEnum has been added to represent the concern addressed by the FormalExpression, which can play the role of weak/strong assumption or guarantee property of a Contract entity. The value for the concern attribute can be derived starting from the Requirement formalised by the FormalExpression.
- **Contract:** the attribute *concern* typed with ConcerEnum has been added to represent the concern addressed by the Contract. The value for the concern attribute is derived starting from the concern attributes specified for the guarantee FormalExpression. A “contract” from the system component metamodel refers to arguments which are encapsulated in the “ArgumentPackageInterface” in the SACM metamodel.

- **BlockInstance:** A component in the system model references to an argument. This connection is the one that links the “Blockinstance” class from the system component metamodel with the “ArgumentationElement”.

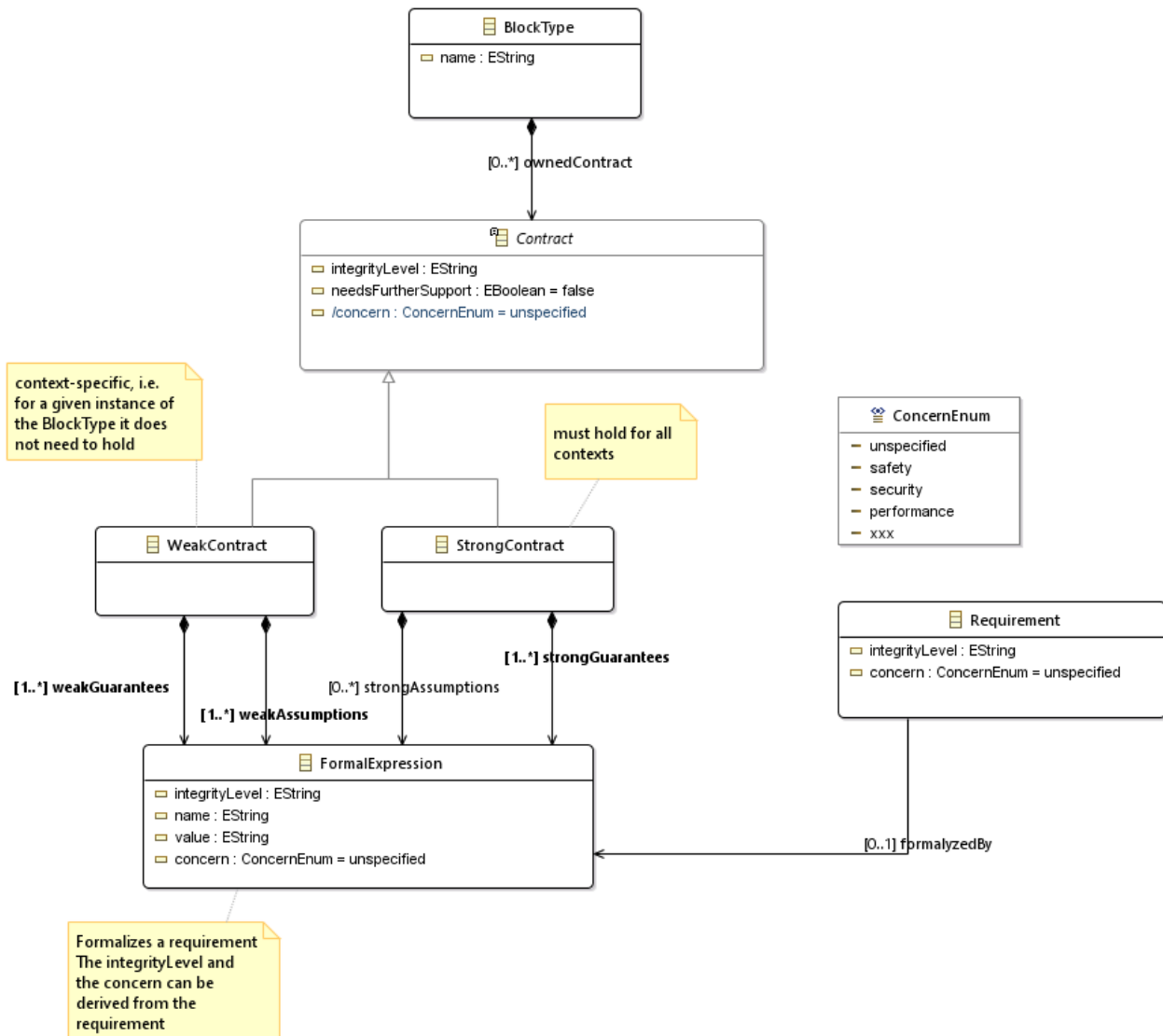


Figure 40. Contract concern

We have identified more areas to be improved with respect to the metamodels connection. One to mention is the one related to dependability modelling and possible needs to extend system component specification in order to support modelling certain concerns at the system component specification level.

4. Way Forward to the Implementation

4.1 Potential Tool support

This section provides an overview of the tools that are expected to play a key role in enabling the achievement of WP4 scientific and technological objectives.

4.1.1 OpenCert – supports “Dependability Assurance Modelling”

OpenCert is an open source tool which started as a result of OPENCROSS project and now is being updated and improved in the context of the AMASS project. OpenCert deals with product and process assurance/certification management to support the compliance assessment and certification of safety-critical systems in sectors such as aerospace, railway and automotive. Figure 41 shows a general view of the functional decomposition of OpenCert conceptual platform which are:

- **Prescriptive Knowledge Management:** Functionality related to the management of standards information as well as any other information derived from them, such as interpretations about intents, mapping between standards, etc. This functional group maintains a knowledge database about “standards & understandings”.
- **Assurance Project Lifecycle Management:** This functionality factorizes aspects such as the creation of safety assurance projects. This module manages a “project repository”, which can be accessed by the other modules.
- **Safety Argumentation Management:** This group manages argumentation information in a modular fashion. It also includes mechanisms to support compositional safety assurance, and assurance patterns management.
- **Evidence Management:** This module manages the full life-cycle of evidences and evidence chains. This includes evidence traceability management and impact analysis. In addition, this module is in charge of communicating with external engineering tools (requirements management, implementation, V&V, etc.).

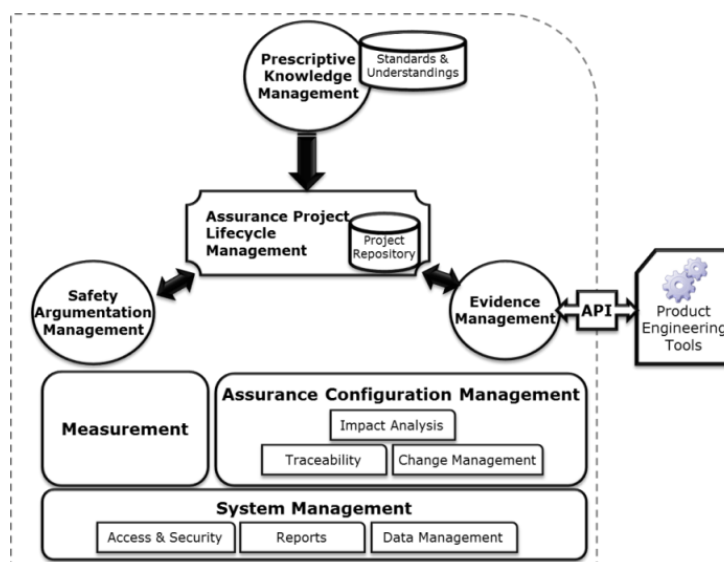


Figure 41. Functional decomposition of the OpenCert platform

- **Assurance Configuration Management:** This is an infrastructure functional module. This includes functionality for traceability management.
- **System Management:** It includes generic functionality for reports creation and data storage.

- **Measurement:** This module contains functionality related to indicators.

For multi concern assurance, we have used mainly the safety argumentation management functionality block. The first prototype implementations and changes made in this area are described in document D4.4 [8].

4.1.2 CNESS - supports “Contract-Based Multiconcern Assurance”

CNESS modelling language (CNESSML), based on UML/SysML/MARTE, and toolset, the latter available on Polarsys and based upon Papyrus, has been adopted in AMASS as basic building block for system component specification with contract-based design. Beside the aforementioned modelling support, CNESSML allows the modelling of timing concerns too, by reusing and extending what is available in the MARTE profile for real time system, to then enable model-based timing analysis. Moreover, CNESS comes with a dedicated profile for dependability, called SafeConcert, presented in AMASS D3.2 [10] and accepted for publication [63]. SafeConcert allows modelling of the failure behaviour for system components and so model-based dependability analysis, like failure propagation or state-based analysis. Regarding specific support for multiconcerns at modelling language level, the part of the CNESS profile related to contract-based design is enriched with the new features elaborated in section 3.2.1.

As it was documented in D4.5 [6] and as recalled within this document (see Section 2.1.4.1), the CNESS toolset also includes ConcertoFLA.

4.1.3 FMVEA - supports “System Dependability Co-Analysis/Assessment”

Failure Mode, Vulnerabilities and Effects Analysis (FMVEA) [27] is a holistic safety and security analysis methodology based on Failure Mode and Effect Analysis (FMEA) and STRIDE threat modelling. Figure 42 shows the basic concept of the method. Semi-quantitative assessment of threats and failures allows a combined impact evaluation and the combined definition of respective mitigation measures.

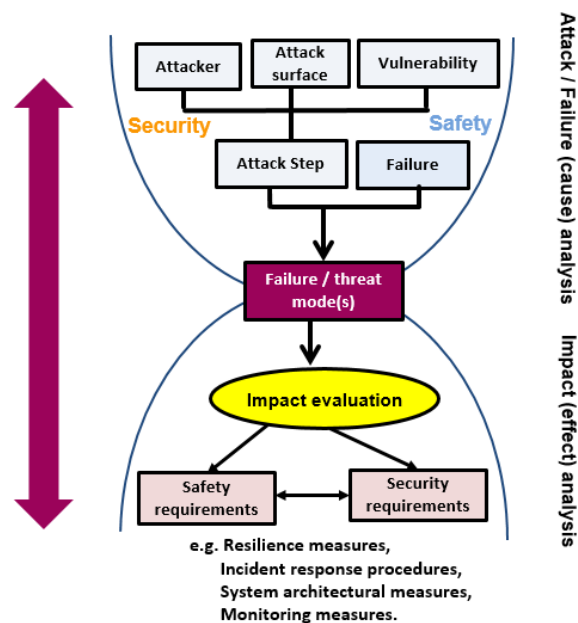


Figure 42. Basic concept of FMVEA

The methodology is supported by a tool developed by AIT (see section 4.1.3), which allows the user to specify the system model and perform a partially automated analysis, based on a dataflow model of a system architecture and a fault / threat model for the system components.

4.1.4 EPF-Composer - supports “System Dependability Co-analysis and assessment”

The Eclipse Process Framework (EPF) project⁵ has developed an approach for supporting customisable (software) process engineering frameworks. This approach, depicted in Figure 43 and called the EPF approach, consists in enabling the separation of method content definition (i.e., definition or reusable process element such as units of work, roles, guidelines, work products) from method content usage, i.e., creation of process models via reuse of pre-defined reusable process elements.

The EPF approach contributes to addressing the following needs:

- Development teams need easy and centralized access to the information
- Ensure compliance to standardised practices
- Teams need support for right-sizing their processes

The EPF approach is tool-supported via the EPF Composer, shortened EPF-C⁶, which is based on a metamodel, called UMA (Unified Method Architecture), which almost fully supports SPEM2.0 (the OMG standard for specification of systems and software processes⁷).

In AMASS, the EPF approach and its tool support have been integrated as core building block. Within WP6, D6.2 [12], EPF-C is currently being strengthened via integration with the BVR tool [54], outcome of the EU ARTEMIS VARIES (VARIability In safety-critical Embedded Systems) Project [55]. This integration will be beneficial not only for general reuse but more specifically for co-assessment and cross-concern reuse, focusing on the interplay of safety and security in line with WP4 objectives.

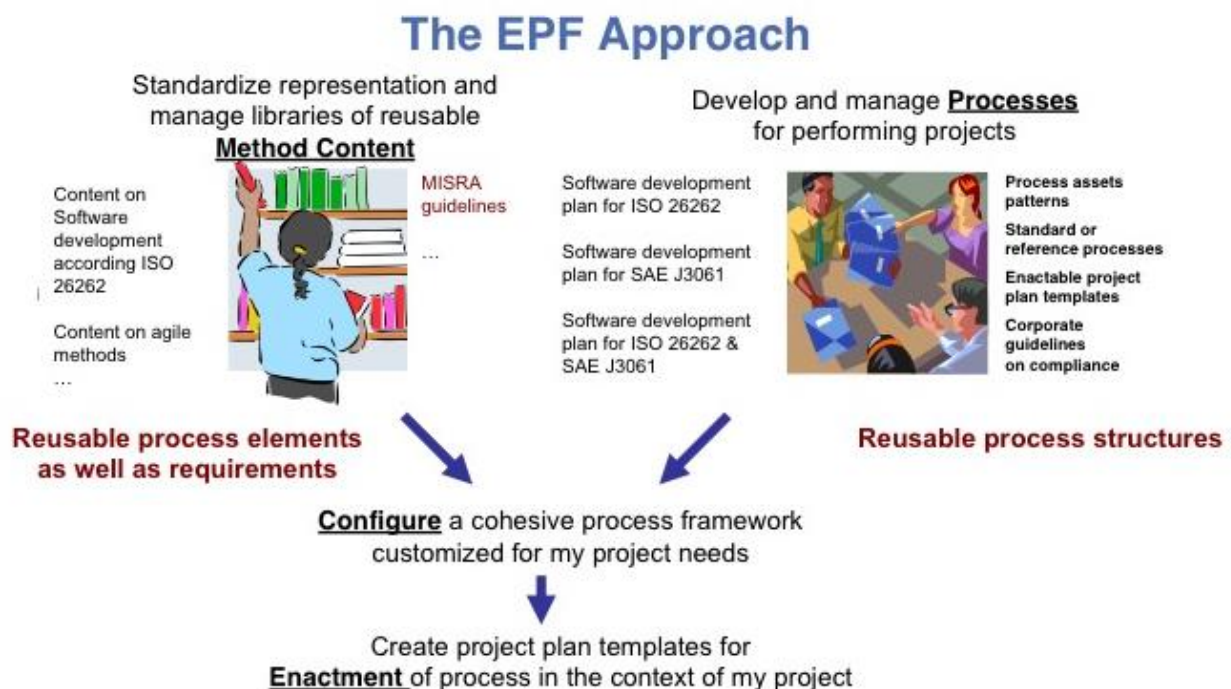


Figure 43. The EPF approach, adapted from [64].

⁵ <http://www.eclipse.org/epf>

⁶ https://eclipse.org/epf/downloads/tool/tool_downloads.php

⁷ <http://www.omg.org/spec/SPEM/2.0>

4.1.5 WEFACT - supports the assurance process workflow

The tool WEFACT (Workflow Engine For Analysis, Certification and Test) originated from the DECOS⁸ Test Bench, which was a Web-based distributed platform for requirements-based testing with continuous impact-assessment in order to support the safety case with evidences. In SafeCer⁹, the test workflow was extended to a workflow for safety certification, and in EMC2¹⁰ the quality attribute of security was integrated.

The WEFACT Version1 was based on the requirements management tool DOORS®.

Now, a new Eclipse-based WEFACT Version2 (see Figure 44) is available, see D4.5 [6]. This new tool version has been extended towards multiconcern assurance cases in the AMASS project.

AIT provides WEFACT as an external workflow tool and, provides interfaces to the AMASS platform.

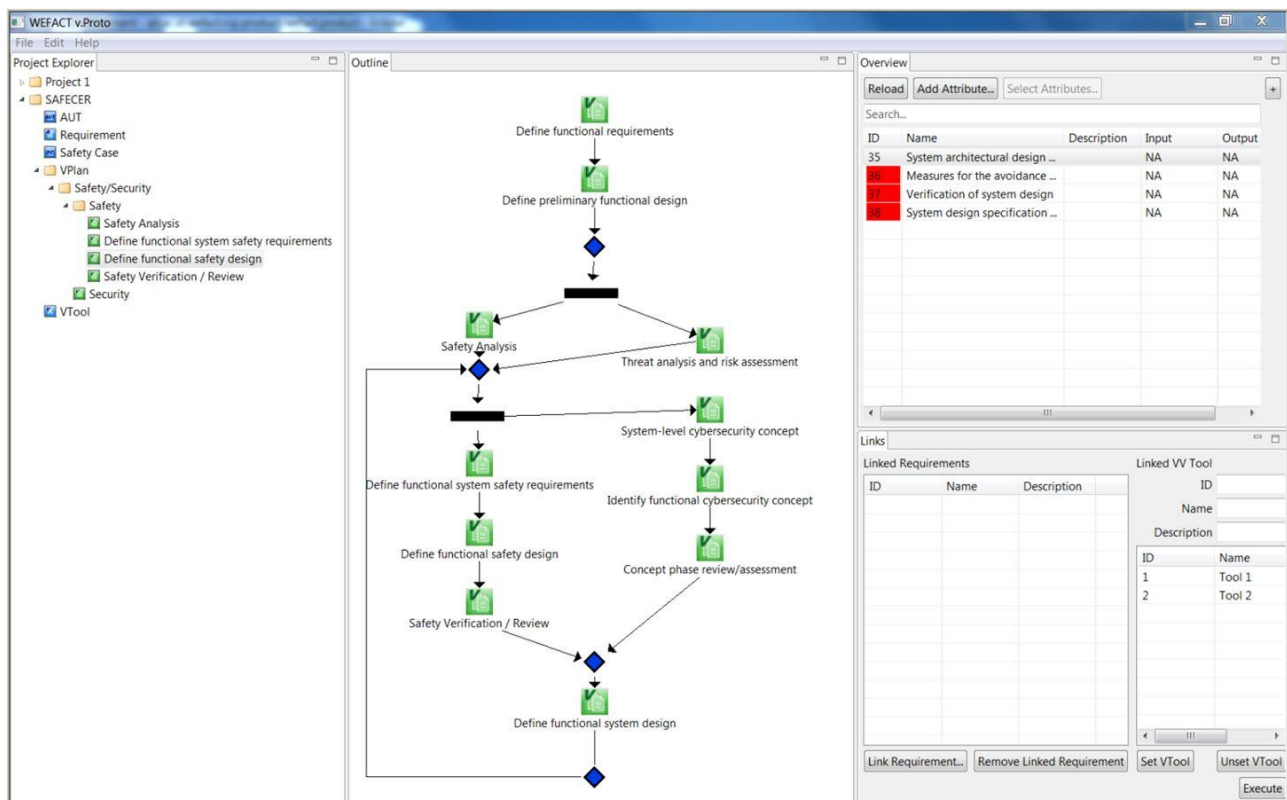


Figure 44. Screenshot WEFACT Version2

⁸ FP6 Integrated Project DECOS (Dependable Embedded COmponents and Systems)

⁹ Artemis project SafeCer (Safety Certification of Software-Intensive Systems with Reusable Components)

¹⁰ Artemis project EMC2 (Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments)

4.1.6 Medini Analyze - supports the assurance process workflow (*)

The medini analyze¹¹ tool-set supports the safety analysis and design for software-controlled safety related functions in various domains, following well known safety standards as ISO 26262, DO-178B and IEC 61508. The tool integrates system architecture design (based on SysML) and software functional design (for example MATLAB®/Simulink®/Stateflow®) with risk and hazard analysis methods - Hazard List, Risk Graph, Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA).

Being built on top of Eclipse technologies like EMF (Eclipse Modelling Framework), medini analyze can trace and track all safety - or more generally assurance case - relevant information and decisions throughout the whole development process. Moreover, it provides numerous tool interfaces for example to central ALM/PLM system and system architecture tools, automatic generation of work products and assurance of their consistency, reuse support by library concepts, as well as catalogues and templates mechanisms, and support of assessments and reviews.

medini analyze – a Model based and System oriented Solution

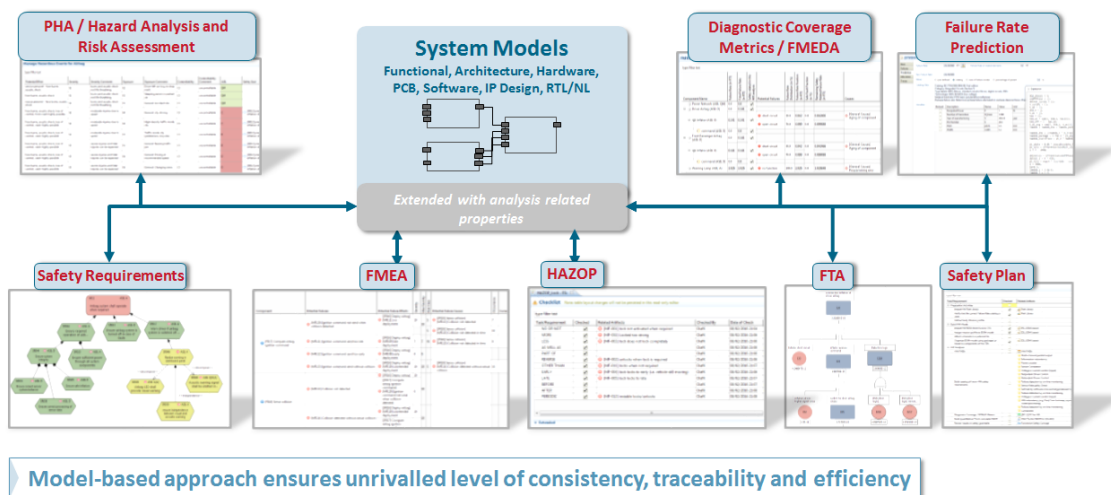


Figure 45. medini analyze overview

The tool has been extended in the SESAMO¹² project towards cybersecurity (see SESAMO deliverables D3.3 for details on the conceptual part). Methodology wise the tool supports Attack Trees, TARA (Threat analysis and Risk Assessment), as well as Security FMEA in analogy to system safety FTA, HARA and FMEA. Basically, all concept phase activities of the SAE J3061 Guidebook are supported as a prototype implementation.

Phase / Activity	Supported By
Project Setup / Preparation	Project templates for different guidelines <ul style="list-style-type: none"> • JASO TP-15002 • EVITA/SESAMO • SAE J3061 (in preparation)
Definition of Target of Evaluation	<ul style="list-style-type: none"> • Form-based target (item) definition • Graphical SysML editor

¹¹ medini™ analyze is a registered trademark of ANSYS medini Technologies AG

¹² Security and Safety Modelling, Artemis JU Grant Agreement no.: 295354

	<ul style="list-style-type: none"> Function list editor
Threat Analysis	<ul style="list-style-type: none"> Attack Tree Modelling Attack Path Computation Propagation to TARA
Risk Assessment	<ul style="list-style-type: none"> TARA with Customizable Risk Graph Out-of-the Box Support for CVSS/CRSS and Common Criteria/EVITA Traces to Security Objectives
Definition of Security Objectives and Requirements	<ul style="list-style-type: none"> GSN-based graphical editor Table-based DOORS-like editor Export to DOORS, PTC Integrity, JAMA Report Generation

As a result of having system safety as well as cyber security analysis methods in the same tool and being executed on the same architecture model enables cross-references from security analysis results to safety analysis results, which leads to potential synergies in implementing the mitigation/prevention functions and countermeasures.

Cyber security analysis based on system models

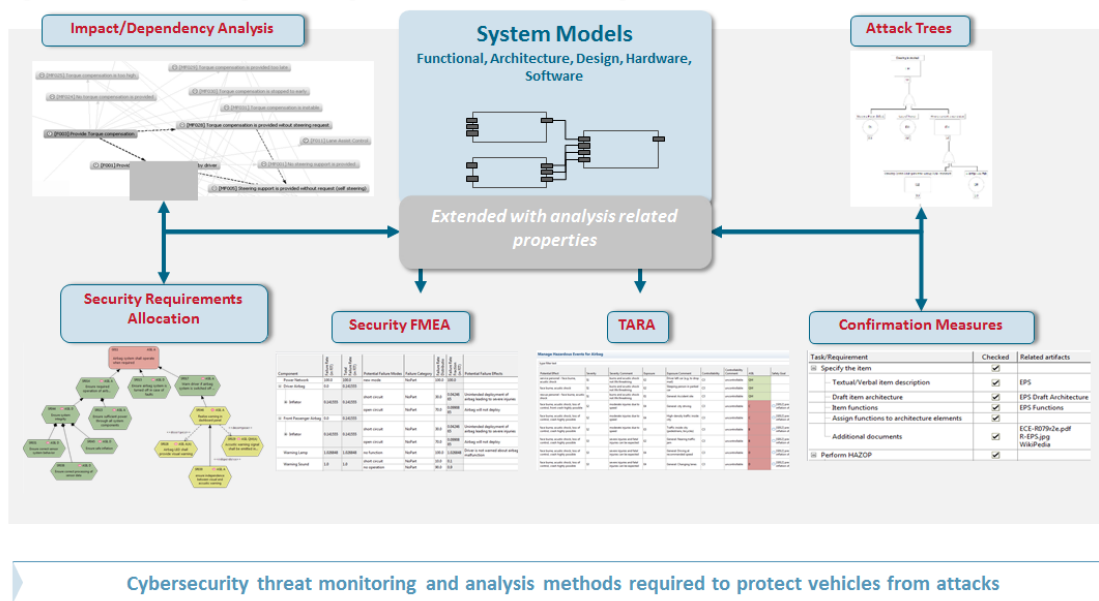


Figure 46. Cyber security analysis with medini analyze

4.1.7 AMASS Farkle - supports product assurance

The AMASS Farkle tool is a learning-based approach on model-based software testing. The AMASS Farkle tool is based on the Farkle tool for model-based certification that has evolved over the last years.

At this stage, we propose to include it in AMASS to prove the inter-operability of the tools when interworking with model-based tools of the AMASS platform. The tests will give proof point that the AMASS-tools support the flow towards the design for multiconcern systems.

The Farkle tool has previously been tested with an automotive use case EAST-ADL Brake-by-wire model [56], also known as model-based test case generation. The AMASS version includes machine-learning as concept.

The idea of LBT is to automatically generate many test cases by combining incremental automata learning algorithm or a model inference algorithm with a model checking algorithm. This results in incremental learning-based testing for reactive systems.

The three steps fundamental to LBT are:

1. Automated model checking algorithm
2. Execution of tests generated in step 1.
3. Assigning a verdict on a test outcome (the oracle step).

There is an iterative feed-back loop: This optimizes test case generation based on previously observed outcomes of test cases.

The interpreter subsystems of AMASS Farkle is interfacing the Linx library, which makes it possible to do tests on any platform on any location. The tool is a combination of Learning- and Model Based Testing.

With the growth of CPS there is a need for verification and certification in an automated way. The models are in some cases poor and not in sync with design which is a common problem in the industry. A tool chain that combines model learning and model checking offers a solution to this problem.

NuSMV model is sent to the model checker for convergence checking. This model is a full representation of the learned states and has more similarities to the Abstract Test Case, used as input for Extended Farkle. Assuming a generated model, the next step is running Extended Farkle. When Extended Farkle is done, LBTest needs to be provided with the output of the counter example to continue learning of the model. As Model Based Testing relies on comparison of In- & Output from the model and the SUT, Extended Farkle should output those sequences somewhere. Instead of just logging these, the tool chain can pass them back to the wrapper for processing. The wrapper could then select which results are relevant for LBTest and return those back to the learning algorithm.

In the overview Figure 47 the Learning Based Test (LBTest) and the Extended Farkle is interworking. The Extended Farkle connects the system under test (SUT) in the target and the complementing dummy under test (DUT).

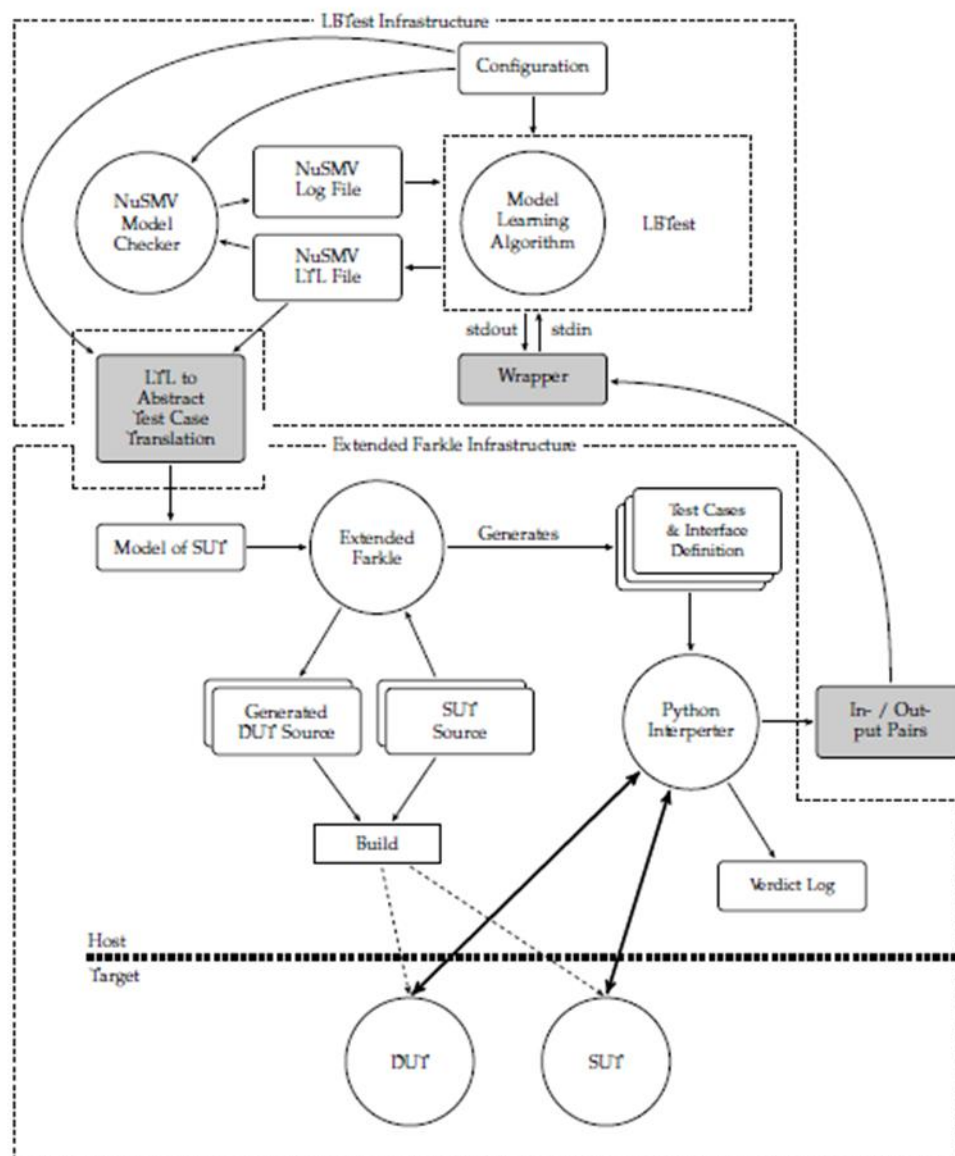


Figure 47. System Overview AMASS Farkle tool

4.1.8 Safety Architect – supports “System Dependability Co-Analysis/Assessment”

Safety Architect is ALL4TEC tool, initially dedicated to perform classical safety analyses: local Failures Modes, Effects and Analysis (FMEA), and automatic deduction of Fault Tree Analysis (FTA) of the identified Feared Events (FE). It is extended during the MERgE ITEA project and French Clarity Project to support Safety and Security Co-Analysis. As described in D4.1 [1] Section 4.2.2.3, Safety engineers and Security engineers can work within Safety Architect, either using separated views or a merged view, to describe the way failures and security threats propagate inside the system architecture. Then, dysfunctional analysis techniques already available in Safety Architect can be applied, such as the automatic generation of **fault trees** or **attack trees**. Safety Architect has different interfaces with many system engineering tools, such as Capella, System Architect, Papyrus, etc. One of ALL4TEC objectives in the AMASS project is to provide Safety Architect as an external tool-chain and to develop interfaces to the AMASS platform.

4.1.9 AMT2.0 – supports “Contract-Based Multiconcern Assurance”

AMT2.0 shall deploy methods for monitoring and diagnosing Cyber-Physical System (CPS) models in the Simulink. The monitoring activities would include translating informal system specifications into formal specification expressed in the extended Signal Temporal Logic (STL) declarative language. The tool would be integrated with the existing monitoring techniques at AIT to the Simulink environment. Novel methods shall be developed for system diagnosis and error localization in the Simulink models upon the detection of the specification violations.

4.1.10 Extensions (*)

The conceptual and design extensions described here in addition to what was already contained in D4.2 [2] are mostly implemented in existing tools (e.g. CHESS/OCRA, the OpenCert Assurance Case Editor or WEFACT, which were provided in D4.5 [6] or even in D4.4 [8]). Part of them are rather conceptual extensions, i.e. an extended usage of existing tool capabilities (in particular w.r.t. WEFACT and CHESS).

Only few new developments are ongoing and planned to be delivered as new tools in D4.6 [5]. This set of new tools comprises the FMVEA tool and the planned tool for trade-off analysis using an ANP (Analytical Network Process).

4.1.11 Implemented Multiconcern Assurance Related Requirements (*)

The following table gives an overview on the requirements covered by the concepts and designs described in this document, and partly in other documents: The assurance case editor was used from the Core prototype, which is described in D4.4 [8].

Table 6. WP4 requirements coverage

<i>ID</i>	<i>Short Description</i>	<i>Description</i>	<i>Proto- type N°</i>	<i>Priority</i>	<i>Elaborated in section</i>
WP4_ACS_001	Assurance case edition	The system shall be able to edit an assurance case in a scalable way.	P1	Must	see D4.4
WP4_ACS_002	Argumentation architecture	The system shall be able to edit a modular structure (argument architecture) associated with a system and/or component.	P2	Must	see D4.4
WP4_ACS_003	Drag and drop argumentation patterns	The system shall be able to instantiate in the actual assurance case an argument pattern (concerning safety and security) selected from the list of patterns stored.	P1	Must	see D4.4
WP4_ACS_004	Provide guidelines for argumentation patterns	The system should be able to provide guidelines to use and instantiate argument pattern (concerning safety and security) presented in the actual assurance case.		Should	
WP4_ACS_005	Provide a structured language to the text inside the claims	The system could be able to provide support for language formalization inside argument claims.	P1	Could	see D4.4
WP4_ACS_006	Provide guidelines for argumentation	The system could be able to provide guidelines about the assurance case edition based on the system/component development phase status.	P1	Could	see D4.4
WP4_ACS_007	Argumentation import/export	The system could be able to import/export argumentations to SACM.	P2	Could	see D4.4

<i>ID</i>	<i>Short Description</i>	<i>Description</i>	<i>Proto- type N°</i>	<i>Priority</i>	<i>Elaborated in section</i>
WP4_ACS_008	Traceability of the dependability case	The system should provide the dependability case reviewers the ability of tracing an overall dependability case (GSN) goal to the requirement within the dependability profile for a given system element and the attribute of interest with which goal is associated.	P1	Should	see D4.4
WP4_ACS_009	Find high level claims	The system shall be able to find high level claims, which are sufficiently cohesive to be supported by extremely diverse strands of argument, supported by diverse types of evidence.		Must	
WP4_ACS_010	Composition of the overall argument	The system should provide the capability of generating a compositional assurance case argument.	P2	Should	see D4.4
WP4_ACS_011	Assurance case status report	The system could provide the capability for querying the assurance case in order to detect: 1) undeveloped goals, 2) fallacies.	P2	Could	2.1.4
WP4_ACS_012	Formal validation of assumptions and context when arguments modules are connected	The system could be able to indicate the validation of assumptions contained in argument modules every time the modules are connected and/or modified		Could	
WP4_ACS_013	Provide quantitative confidence metrics about an assurance case in a report	The system could produce a status report indicating a quantitative confidence metric for assurance case.	P2	Could	2.1.4
WP4_CAC_010	Contract-based trade-off analysis	The system could provide the capability to evaluate safety and security requirements on different system architectures to perform trade-off analysis based on the contract specification.	P2	Could	2.3.2, 2.3.4, 3.1.3.2, 3.1.3.3
WP4_DAM_001	Capability to model relationships between concerns	The system shall be able to provide an assurance case which records the relationships between dependability attributes and how they are affected because of design decisions.	P2	Must	
WP4_DAM_002	Capability to capture conflicts occurring during system development and the trade-off process	The system shall provide the capability for modelling a dependability case which captures the conflicts that occur during system development and the trade-off process to justify why the taken design decisions are the most optimal ones.	P2	Must	
WP4_CMA_001	The AMASS tools must support specification of variability at the argumentation level	The system shall provide the capability for modelling arguments in the assurance case about multi-concern and multi-context. The multi-concern and multi-context argumentation could follow a variability modelling a solution. If GSN-like modelling	P2	Must	2.2.4

ID	Short Description	Description	Proto- type N°	Priority	Elaborated in section
		elements are considered, the diamond for representing alternatives as well as the octagon for extrinsic variability could be considered.			
WP4_CMA_002	Component contracts must support multiple concerns	The system shall provide a contract specification language that supports the formalization of both safety and security requirements.	P2	Must	2.3.1, 3.1.3.1
WP4_CMA_003	Contract based multi-concern assurance	The system must support features that support contract-based assurance with respect to multiple concerns; i.e. it must be possible to specify relations between safety contracts, security contracts and other-concerns-related contracts in order to take care of the influence of system modifications for mitigating the risks associated with one quality attribute on the contract belonging to another quality attribute.	conceptually developed	Must	2.3.3
WP4_SDCA_001	System dependability co-architecturing and co-design	The system shall provide features, which allow architecture modelling collaboration and co-designing a system or component with a balanced combination of different goals addressing various quality attributes.	P2	Must	WP3 ¹³ , 2.1.2, 2.3.2, 2.3.4, 4.1.9 ¹⁴ .
WP4_SDCA_002	System dependability co-verification and co-validation	The system shall support efficient system or component co-verification and co-validation with respect to multiple quality attributes.	P2	Must	2.1.4, 2.3.1, 3.1.3.1
WP4_SDCA_003	The system shall allow combinations of safety and security analysis	The system shall allow combinations of safety and security analysis.	P2	Must	2.1.4

¹³ This refers to the 1st sentence of the requirement; collaboration at architectural level is under development and expected in P2.

¹⁴ This refers to the 2nd sentence; first prototypes for trade-off analysis are expected in P2.

5. Conclusions (*)

In this deliverable, the conceptual approach for multiconcern assurance has been elaborated in two iterations, the confidential intermediate edition D4.2 [2] and the public final edition D4.3 – the document at hand. After an introduction which explained the scope, the relation to other tasks and deliverables within and beyond WP4, chapter 2 presents several approaches for the main features of multiconcern assurance including extensions added in the second iteration, namely “System Dependability Co-Analysis and Assessment”, “Dependability Assurance Case Modelling”, and “Contract-Based Multiconcern Assurance”, building on and explaining relations to the state of the art described in D4.1 [1].

In chapter 3, the design of the approaches described in the previous chapter is been presented, first on a conceptual design level and then on the level of Multiconcern Assurance Metamodel extensions. This design will guide the implementation of the third iteration of the AMASS platform implementation, which will be described in D4.6 [5].

Chapter 4 presents a set of identified tools for realizing the aforementioned features in the second iteration. The set comprises internal tools for the basic building block Assurance Case Modelling as well as external tools for the other features. In this edition D4.3, a table has been added describing the coverage of requirements by the realized multiconcern assurance features.

The relation between the implemented functionalities for Multiconcern Assurance and their realization and evaluation in the case studies is described in D1.5 [82], wherein Table 76 gives an overview, showing that, already in Prototype P1, 7 out of 11 case studies apply WP4 functionalities, in most cases more than one. Contract-based Multiconcern Assurance as described here in D4.3 is not yet covered there but prototypic implementations are planned for P2. Moreover, Contract-based Multiconcern Assurance at runtime according to section 4.1.9 is mainly described in D3.3 [88] and realized in CS3.

Abbreviations and Definitions (*)

Definitions are common to the whole AMASS project and are given in the AMASS glossary (deliverable D2.2 [11]). Following abbreviations are used in this document:

<i>Abbreviation</i>	<i>Explanation</i>
AFHA	Aircraft-level Functional Hazard Assessment
ALM	Application Lifecycle Management
ANP	Analytical Network Process
ARP	Aerospace Recommended Practice
ARTEMIS	ARTEMIS Industry Association is the association for actors in Embedded Intelligent Systems within Europe
ASA	Aircraft Safety Assessment
ASIC	Application Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level
ATA	Attack Tree Analysis
BDMP	Boolean Logic Driven Markov Processes
BRA	Binary Risk Analysis
BVR	Base Variability Resolution - a domain-specific language designed specifically to enable software product-line engineering (SPLE)
CACM	Common Assurance and Certification Metamodel
CAN	Controller Area Network
CCL	Common Certification Language
CENELEC	Comité Européen de Normalisation Électrotechnique (European Committee for Electrotechnical Standardization)
CHASSIS	Combined Harm Assessment of Safety and Security for Information Systems
CHESSML	CHESS Modelling Language
CIA	Confidentiality, Integrity, Availability
CPS	Cyber-Physical Systems
CS	Case Study
CTMM	Continuous Time Markov Models
CVSS	Common Vulnerability Scoring System
DD	Dependence Diagram
DAL	Development Assurance Levels
DFD	Data-Flow Diagram
DUT	Dummy Under Test
EAST-ADL	Electronics Architecture and Software Technology - Architecture Description Language (AUTOSAR-compliant modelling language for the Automotive industry)
ECSEL	Electronic Components and Systems for European Leadership
EMC ²	Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments
EN	European Norm
EPF/C	Eclipse Process Framework-Composer
EVITA	E-Safety Vehicle Intrusion Protected Applications

FANDA	Tool supplied by Fairmount Automation, used together with TOM for assessing design alternatives and facilitating trade-offs in critical systems
FHA	Functional Hazard Assessment
FMEA	Failure Modes and Effects Analysis
F(I)MEA	Failure (Intrusion) Modes and Effects Analysis
FMVEA	Failure Modes, Vulnerabilities and Effect Analysis
FTA	Fault Tree Analysis
FTP	File Transfer Protocol
GSM-R	Global System for Mobile Communications – Railway
GSN	Goal Structuring Notation
HARA	Hazard Analysis and Risk Assessment
HAZOP	HAZard and OPerability study
HEAVENS	HEALing Vulnerabilities to ENhance Software Security and Safety (Swedish Vinnova funded research project)
HMI	Human Machine Interface
HTTPS	Hypertext Transfer Protocol Secure
IACS	Industrial Automation and Control System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IL	Impact Level
ISO	International Organization for Standardization
IT	Information Technology
JU	Joint Undertaking
LBTest	Learning Based Test
MA	Markov Analysis
MILS	Multiple Independent Levels of Security
MISRA	Motor Industry Software Reliability Association
NuSMV	New Symbolic Model Verifier (a symbolic model checker tool for finite state systems)
OBD	On-Board Diagnosis
OCL	Object Constraint Language
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation
OSS	Ocra System Specification
PASA	Preliminary Aircraft Safety Assessment
PASRA	Preliminary Aircraft Security Risk Assessment
PLM	Product Lifecycle Management
PSecAC	Plan for Security Aspects of Certification
PSSA	Preliminary System Safety Assessment
RAMS	Reliability, Availability, Maintainability, Safety (and Security)
RL	Remediation Levels
RTCA	Radio Technical Commission for Aeronautics
SACM	Structured Assurance Case Metamodel
SAE	Society of Automotive Engineers
SAHARA	Security-aware Hazard Analysis and Risk Assessment
SCADA	Supervisory Control And Data Acquisition
SEooC	Safety Element out of Context

SIL	Safety Integrity Level
SL	Security Level
SiSoPLE	Security-informed Safety-oriented Process Line Engineering
SoPLE	Safety-oriented Process Line Engineering
SPEM	Software & Systems Process Engineering Metamodel
SSA	System Safety Assessment
STL	Signal Temporal Logic
STPA-SEC	STAMP (Systems- Theoretic Accident Model and Processes) Based Process Analysis
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
SUT	System Under Test
SW	Software
SysML	System Modelling Language
S&S	Safety and Security
TARA	Threat Analysis and Risk Assessment
TOM	Tool supplied by Fairmount Automation, used together with FANDA for assessing design alternatives and facilitating trade-offs in critical systems
TVRA	Threat, Vulnerability and Risk Analysis
TCP/IP	Transmission Control Protocol/Internet Protocol
TL	threat level
UDP	User Datagram Protocol
UMA	Unified Method Architecture
UML	Unified Modelling Language
V&V	Verification and Validation
WP	Work Package

References (*)

- [1] AMASS [D4.1 Baseline and requirements for multiconcern assurance](#), 30th September 2016.
- [2] AMASS D4.2 Design of the AMASS tools and methods for multiconcern assurance (a), 30th June 2017.
- [3] AMASS D4.3 Design of the AMASS tools and methods for multiconcern assurance (b), April 2018.
- [4] AMASS [D2.1 Business cases and high-level requirements](#), 28th February 2017.
- [5] AMASS D4.6 Prototype for multiconcern assurance (c), August 2018.
- [6] AMASS [D4.5 Prototype for multiconcern assurance \(b\)](#), October 2017.
- [7] AMASS D4.8 Methodological guide for multiconcern assurance (b), August 2018.
- [8] AMASS [D4.4 Prototype for multiconcern assurance \(a\)](#), 31st January 2017.
- [9] AMASS [D1.1 Case studies description and business impact](#), 30th November 2016.
- [10] AMASS D3.2 Design of the AMASS tools and methods for architecture-driven assurance (a), 31st March 2017.
- [11] AMASS D2.2 AMASS reference architecture (a), 30th November 2016.
- [12] AMASS D6.2 Design of the AMASS tools and methods for cross/intra-domain reuse (a), October 2017.
- [13] AMASS [D6.5 Prototype for cross/intra-domain reuse \(b\)](#), December 2017.
- [14] Jean-Claude Laprie - Dependability: Basic Concepts and Terminology [Book]. - Vienna: Springer, 1992. - Vol. 5.
- [15] B. Gallina, L. Fabre. Benefits of Security-informed Safety-oriented Process Line Engineering. IEEE 34th Digital Avionics Systems Conference (DASC-34), Prague, Czech Republic, September 13-17, ISBN 978-1-4799-8939-3, 2015.
- [16] Avizienis, A., J.-C., Laprie, B., Randell, C., Landwehr, 2004, Basic concepts and taxonomy of dependable and secure computing. In: IEEE Trans. Dependable Sec. Comput. 1(1): 11-33.
- [17] Bloomfield, R., R. Stroud, 2013, Security-Informed Safety "If it's not secure, it's not safe". Marc-Olivier Killijian. Proceedings of the International Conference on. Computer Safety, Reliability and Security (SafeComp) FastAbstract, Toulouse, France. pp.NC. <hal- 00926459>.
- [18] Gil-Casals, S., P., Owezarski, G., Descargues, 2012, Risk Assessment for Airworthiness Security. Proceedings of the International Conference on. Computer Safety, Reliability and Security (SafeComp), Magdeburg, Germany. pp.8, <hal-00698523>
- [19] RTCA DO-326A, 2014, Airworthiness Security Process Specification, RTCA.
- [20] ARP-4761, 1996, Guidelines and Methods for Conducting the Safety Assessment process on Civil Airborne Systems and Equipment.
- [21] I. Ayala, B. Gallina. Towards Tool-based Security-informed Safety Oriented Process Line Engineering. 1st ACM International workshop on Interplay of Security, Safety and System/Software Architecture (ISSA), Copenhagen, Denmark, November 28th, 2016.
- [22] M.Steger, M.Karner, J.Hillebrand, W.Rom and K.Romer. 2016. A security metric for structured security analysis of cyber-physical systems supporting SAE J3061. In 2016 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS, CPS Data 2016. IEEE, 6.
- [23] G.Macher, E.Armengaud, C.Kreiner, B.Brenner, C.Schmittner, Z.Ma, H.Martin, L. Krammer, "Integration of Security in the Development Lifecycle of Dependable Automotive CPS", Handbook of research for cyber-physical systems Ubiquity, IGI Global, 2017
- [24] SAE J3061- Cybersecurity Guidebook for Cyber-Physical Automotive Systems. SAE - Society of Automotive Engineers.
- [25] G. Macher, E. Armengaud, E. Brenner, C. Kreiner. A Review of Threat Analysis and Risk Assessment Methods in the Automotive Context. International Conference on Computer Safety, Reliability, and Security (SafeComp), LNCS, vol 9922, 2016

- [26] Macher, G., Hoeller, A., Sporer, H., Armengaud, E., & Kreiner, C. (2015). A Comprehensive Safety, Security, and Serviceability Assessment Method. Delft, The Netherlands: 34th International Conference on Computer Safety, Reliability, and Security - SAFECOMP 2015.
- [27] Schmittner, C., Gruber, T., Puschner, P., & Schoitsch, E. (2014). Security application of failure mode and effect analysis (FMEA). International Conference on Computer Safety, Reliability, and Security (SafeComp 2014).
- [28] Schmittner, Christoph, Zhendong Ma, Carolina Reyes, Oliver Dillinger, and Peter Puschner. "Using SAE J3061 for Automotive Security Requirement Engineering." In International Conference on Computer Safety, Reliability, and Security, pp. 157-170. Springer International Publishing, 2016.
- [29] Tidwell, T., Larson, R., Fitch, K., Hale, J.: Modelling internet attacks. In: Proceedings of the Second Annual IEEE SMC Information Assurance Workshop, pp. 54-59. IEEE Press, Los Alamitos 2001.
- [30] Schneier, B., Attack Trees, Dr. Dobbs's Journal of Software Tools 24, 12 (December 1999): 21-29.
- [31] S. Byer, T. Enderle, D. Oka, and M. Wolf, "Automotive security testing – the digital crash test", 3rd CESA automotive electronics congress, 2014
- [32] S. Basagiannis, R. Chabuckswar, Y. yang, K. McLaughlin, M. Boubekur, "Smart Grid Security: Chapter 10 Real world Experiences from Smart Grid Security Application", 2015
- [33] Charles B. Weinstock Howard F. Lipson John Goodenough, "Arguing Security – Creating Security Assurance Cases", 2007, <https://www.us-cert.gov/bsi/articles/knowledge/assurance-cases/arguing-security-creating-security-assurance-cases>
- [34] Alberts, C. et al. "Introduction to the OCTAVE Approach." Pittsburgh, PA, Carnegie Mellon University (2003).
- [35] Kelling, E., Friedewald, M., Leimbach, T., Menzel, M., Säger, P., Seudié, H., and Weyl, B. (2009). Specification and Evaluation of e-Security Relevant Use cases. Technical Report Deliverable D2.1, EVITA Project.
- [36] ITS, ETSI. Security: Threat, Vulnerability and Risk Analysis (TVRA). Technical report, ETSI, 2010.
- [37] M. Islam, C. Sandberg, A. Bokesand, T. Olovsson, H. Broberg, P. Kleberger, A. Lautenbach, A. Hansson, A. Söderberg-Rivkin, and S. P. Kadhivelan. Deliverable D2 - Security Models. HEAVENS Project, Version 1.0 (Release 1), September 2014.
- [38] Dowd, Mark, John McDonald, and Justin Schuh. The art of software security assessment: Identifying and preventing software vulnerabilities. Pearson Education, 2006.
- [39] Schmittner, Christoph, et al. "A case study of fmvea and chassis as safety and security co-analysis method for automotive cyber-physical systems." Proceedings of the 1st ACM Workshop on Cyber-Physical System Security. ACM, 2015.
- [40] Macher, G., Sporer, H., Berlach, R., Armengaud, E., & Kreiner, C. (2015). SAHARA: A security-aware hazard and risk analysis method. Design, Automation Test in Europe Conference Exhibition (DATE 2015), (pp. 621-624).
- [41] C. Raspotnig, P. Karpati, and V. Katta, "A combined process for elicitation and analysis of safety and security requirements," in Lecture Notes in Business Information Processing, 2012.
- [42] Bouissou, Marc, and Jean-Louis Bon. "A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes." Reliability Engineering & System Safety 82.2 (2003): 149-163.
- [43] National Highway Traffic Safety Administration. Characterization of Potential Security Threats in Modern Automobiles - A Composite Modelling Approach, October 2014
- [44] Sapiro, B.: Binary Risk Analysis. Creative Commons License. 1st edn.
- [45] Young, W., & Leveson, N. (2013). Systems thinking for safety and security. In Proceedings of the 29th Annual Computer Security Applications Conference (pp. 1-8). ACM.
- [46] SAE. 2016. Surface Vehicle Recommended Practice. Technical Report. 128 pages

- [47] J. P. Castellanos Ardila and B. Gallina. Towards Increased Efficiency and Confidence in Process Compliance. 24th European & Asian Systems, Software & Service Process Improvement & Innovation (EuroSPI&Asia2), Ostrava, Czech Republic, 5.-8. Sept. 2017.
- [48] B. Gallina, E. Gómez-Martínez, C. Benac Earle. Promoting MBA in the Rail Sector by Deriving Process-related Evidence via MDSafeCer. Computer Standards & Interfaces -SPICE-2016 Special Issue (CSI SPICE-2016), <http://dx.doi.org/10.1016/j.csi.2016.11.007>
- [49] B. Gallina. A Model-driven Safety Certification Method for Process Compliance. 2nd IEEE International Workshop on Assurance Cases for Software-intensive Systems (ASSURE), joint event of ISSRE, Naples, Italy, doi: 10.1109/ISSREW.2014.30, pp. 204-209, November 3-6, 2014.
- [50] B. Gallina, K. Lundqvist and K. Forsberg. THRUST: A Method for Speeding Up the Creation of Process-related Deliverables. IEEE 33rd Digital Avionics Systems Conference (DASC-33), doi:10.1109/DASC.2014.6979489, Colorado Springs, CO, USA, October 5-9, 2014.
- [51] OPENCROSS Project <http://www.opencross-project.eu/>
- [52] IEC 62443: Security for industrial automation and control systems
- [53] Ray, Arnab, and Rance Cleaveland. "Security Assurance Cases for Medical Cyber-Physical Systems." IEEE Design & Test 32.5 (2015): 56-65.
- [54] BVR tool," <http://modelbased.net/tools/bvr-tool/> , accessed: 2017-06-07.
- [55] VARIES," <http://www.varies.eu/> , accessed: 2017-06-07.
- [56] EAST-ADL Brake-by-wire model <http://www.east-adl.info/repository/examples/BrakeByWireSystem.pdf>
- [57] Georgios Despotou, Managing the Evolution of Dependability Cases for Systems of Systems, PhD Thesis, April 2007
- [58] Georgios Despotou, John McDermid, Tim Kelly , "Using Scenarios to Identify and Trade-off Dependability Objectives in Design", January 2005
- [59] G. Despotou and T. Kelly, "An Argument Based Approach for Assessing Design Alternatives and Facilitating Trade-offs in Critical Systems", Journal of System Safety, 2007
- [60] Morant A., Gustafson A., Söderholm P. (2016) Safety and Availability Evaluation of Railway Signalling Systems. In: Kumar U., Ahmadi A., Verma A., Varde P. (eds) Current Trends in Reliability, Availability, Maintainability and Safety. Lecture Notes in Mechanical Engineering. Springer, Cham
- [61] Jhawar, Ravi and Lounis, Karim and Mauw, Sjouke (2016) *A Stochastic Framework for Quantitative Analysis of Attack-Defense Trees*. In: 12th International Workshop on Security and Trust Management, STM 2016, 26-27 September 2016, Heraklion, Crete, Greece (pp. pp. 138-153)
- [62] ISO 26262. Road Vehicles-Functional Safety. International Standard, 2011.
- [63] L. Montecchi and B. Gallina. SafeConcert: a Metamodel for a Concerted Safety Modeling of Socio-Technical Systems. 5th International Symposium on Model-Based Safety and Assessment (IMBSA), Trento, Italy, September, 2017.
- [64] P. Haumer. Eclipse Process Framework Composer, Part 1: Key Concepts, 2007. <https://eclipse.org/epf/general/EPFComposerOverviewPart1.pdf>
- [65] I. Sljivo, B. Gallina. Building Multiple-Viewpoint Assurance Cases Using Assumption/Guarantee Contracts. 1st ACM International workshop on Interplay of Security, Safety and System/Software Architecture (ISSA), Copenhagen, Denmark, November 28th, 2016.
- [66] Saaty, Thomas L. The Analytic Hierarchy and Analytic Network Measurement Processes: Applications to Decisions under Risk, EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS, Vol.1, No. 1, 2008, (122-196)
- [67] Balakrishnan, Nikhil. Dependability in Medicine and Neurology: Using Engineering and Management Principles for Better Patient Care. Springer, 2015.
- [68] Marco Gario, Alessandro Cimatti, Cristian Mattarei, Stefano Tonetta, Kristin Yvonne Rozier: Model Checking at Scale: Automated Air Traffic Control Design Space Exploration. CAV (2) 2016: 3-22

- [69] Ruiz, Alejandra; Melzi, Alberto; Kelly, Tim / Systematic application of ISO 26262 on a SEooC: : Support by applying a systematic reuse approach .DATE '15 Proceedings of the 2015 Design, Automation and Test in Europe Conference & Exhibition. 2015. p. 393-396.
- [70] Alessandro Cimatti, Rance DeLong, Davide Marcantonio, Stefano Tonetta: Combining MILS with Contract-Based Design for Safety and Security Requirements. SAFECOMP Workshops 2015: 264-276
- [71] Amorim, T., Martin, H., Ma, Z., Schmittner, C., Schneider, D., Macher, G., ... & Kreiner, C. (2017, September). Systematic Pattern Approach for Safety and Security Co-engineering in the Automotive Domain. In International Conference on Computer Safety, Reliability, and Security (pp. 329-342). Springer, Cham.
- [72] ISO/IEC 15026-2:2011 Systems and software engineering - Systems and software assurance - Part 2: Assurance case
- [73] B. Gallina, E. Sefer and A. Refsdal, "Towards Safety Risk Assessment of Socio-Technical Systems via Failure Logic Analysis," *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, Naples, 2014, pp. 287-292.
- [74] L. Grunske, J. Han, "A Comparative Study into Architecture-Based Safety Evaluation Methodologies using AADL's Error Annex and Failure Propagation Models", 11th IEEE High Assurance Systems Engineering Symposium, pp. 283–292, Nanjing, China, 3-5 Dec., 2008.
- [75] M. Wallace. Modular architectural representation and analysis of fault propagation and transformation, vol. 141, no. 3, pp. 53–71, 2005.
- [76] CONCERTO Deliverable D3.3 November 2015 Design and implementation of analysis methods for non-functional properties – Final version.
- [77] CCSDS 350.1-G-1, Report Concerning Space Data Systems Standards, Security Threats Against Space Missions, December 2015.
- [78] NIST SP 800-30, Guide for Conducting Risk assessments, September 2012.
- [79] SysML v1.4 Specification Release, September, 2015. <http://www.omg.sysml.org/specifications.htm>
- [80] Origin Consulting (York) Limited, GSN Community Standard Version 1, Nov. 2011, www.goalstructuringnotation.info/documents/GSN_Standard.pdf
- [81] SafeCer Deliverable D2.3.1, July 2012, "Extension of techniques for modular safety arguments"
- [82] AMASS [D1.5 AMASS Demonstrators \(b\)](#), April 2018
- [83] CONCERTO ARTEMIS JU project - <http://www.concerto-project.org/>
- [84] MERgE Project – <http://www.merge-project.eu/>
- [85] SafeCer – consisting of Artemis JU projects pSafeCer <https://artemis-ia.eu/project/30-psafecer.html> and nSafeCer - <https://artemis-ia.eu/project/40-nsafecer.html>
- [86] SESAMO Artemis JU project - <http://sesamo-project.eu/>
- [87] EMC2 Artemis JU project - <https://www.artemis-emc2.eu/>
- [88] AMASS [D3.3 Design of the AMASS tools and methods for architecture-driven assurance \(b\)](#), 30th March 2018.

Appendix A: Changes since the Predecessor Version D4.2 (*)

New Sections:

Section	Title
2.1.4.1	System dependability co-analysis via ConcertoFLA
2.1.4.2	WEFACT Tool Concept
2.1.4.3	FMVEA Tool Concept
2.2.4	Support for variability management at the argumentation level
2.3.1	Abstract functions in the contracts specification
2.3.2	Contract-based trade-off analysis in parameterized architectures
2.3.3	General extensions to contract based multi-concern assurance
2.3.4	Contract-based trade-off analysis with the Analytical Network Process
3.1.2.1	Support specification of variability at the argumentation level
3.1.3.1	Abstract functions in the contracts specification
3.1.3.2	Contract-based trade-off analysis in parameterized architectures
3.1.3.3	Contract-based trade-off analysis with the Analytical Network Process
3.1.4.1	System dependability co-analysis via ConcertoFLA
3.1.4.2	WEFACT Tool Concept
3.1.4.3	FMVEA Tool Concept
4.1.10	Extensions
4.1.11	Implemented Multiconcern Assurance Related Requirements
Appendix A	Changes since the Predecessor Version D4.2

Modified Sections:

Section	Title	Changes
1.	Introduction	Updated
1.1	From Monoconcern to Multiconcern	Updated
1.2	Scope and Objectives of this Deliverable	Updated
1.2	Relation to other AMASS Deliverables	Updated
2.1.3.2	Normative spaces ready for SiSoPLE	Added description of process development in EPF-C and the process execution with WEFACT based on an example
2.2.2	Safety and Security Assurance Case	Updated
4.1.6	Medini Analyze - supports the assurance process workflow	Description extended with more details about the latest tool version
5	Conclusions	Updated
	References	Minor extensions
	Abbreviations and Definitions	Minor extensions