



Deliverable D6.1

Blockly4SoS Model and Simulator

Technical References

Document version	:	1
Submission Date	:	30/06/2021
Dissemination Level	:	Public
Contribution to	:	WP6 – Risk Analysis and Mitigation Strategies
Document Owner	:	RESILTECH
File Name	:	Blockly4SoS Model and Simulator
Revision	:	3.0
Project Acronym	:	BIECO
Project Title	:	Building Trust in Ecosystem and Ecosystem Components
Grant Agreement n.	:	952702
Call	:	H2020-SU-ICT-2018-2020
Project Duration	:	36 months, from 01/09/2020 to 31/08/2023
Website	:	https://www.bieco.org

Revision History

REVISION	DATE	INVOLVED PARTNERS	DESCRIPTION
0.0	30/03/2021	RES	Table of Contents
0.1	07/04/2021	ALL	Review of the Table of Contents
0.2	16/04/2021	RES	Initial contribution to Section 2 and Appendices A, B
0.3	30/04/2021	UMU, IESE	Review of the state-of-the-art reports (Sections 2.1-2.4)
0.4	30/04/2021	RES	Updates to Section 2 and 3, contributions to Sections 4 and 6
0.5	21/05/2021	RES	Finalization of contribution to Sections 4 and 6
0.6	21/05/2021	UMU	Contribution to Section 5 and acronyms table.
0.7	24/05/2021	RES	Finalisation and preparation for internal revision
0.8	27/05/2021	RES	Implementation of internal reviewer comments. Finalisation and preparation for external revision.
0.9	31/05/2021	RES	Updates in Executive Summary, Introduction and Conclusions sections
1.0	02/06/2021	UMU	External Review of the whole deliverable
1.1	11/06/2021	IESE	External Review of the whole deliverable
2.0	16/06/2021	RES	Implementation of Reviewers Suggestions
2.1	25/06/2021	UNI	Review by Coordinator
3.0	30/06/2021	UNI	Final Version

List of Contributors

Deliverable Editor and Contributors: Enrico Schiavone (RES), Francesco Brancati (RES), Diamantea Mongelli (RES), Gabriele Morgante (RES), Andrea Bondavalli (RES), Rosaria Esposito (RES), Francesco Rossi (RES), Andrea Ceccarelli (RES), Sara N. Matheu (UMU), Emilia Cioroica (IESE), Ioannis Sorokos (IESE), Adrián Sánchez (UMU).

Reviewers: Andrea Ceccarelli (RES, internal reviewer), Ioannis Sorokos (IESE, external reviewer), Sara N. Matheu (UMU, external reviewer), José Barata (UNI, coordinator).

Partners



Disclaimer

The publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Disclaimer: The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

All rights reserved.

The document is proprietary of the BIECO consortium members. No copying or distributing, in any form or by any means, is allowed without the prior written agreement of the owner of the property rights.

This document reflects only the authors' view. The European Community is not liable for any use that may be made of the information contained herein.



BIECO project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **952702**.

Acronyms

Acronym	Term
ACL	Access Control List
ADVISE	ADversary Vlew Security Evaluation
AEG	Attack Execution Graph
AG	Attack Graph
ANSSI	Agence Nationale de la Sécurité des Systèmes d'Information
APG	Attack Path Graph
APT	Attack Path Tree
BDD	Block Definition Diagram
CAPEC	Common Attack Pattern Enumeration and Classification
CIA	Confidentiality, Integrity, Availability
CPE	Common Platform Enumeration
CPSoS	Cyber-Physical System-of-System
CRS	Cybersecurity Requirements Specification
CRUD	Creating, Reading, Updating, Deleting
CS	Constituent System
CSMS	Cyber-Security Management System
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
DCS	Distributed Control Systems
DFD	Data Flow Diagram
DoS	Denial of Service
DREAD	Damage, Reproducibility, Exploitability, Affected users, Discoverability
EMF	Eclipse Modelling Framework
ENISA	European Union Agency for Cybersecurity
FMEA	Failure Mode and Effects Analysis
FMECA	Failure Mode, Effects, and Criticality Analysis
FTA	Fault Tree Analysis
GUI	Graphical User Interface
HARM	Hailstorm Application Risk Metric
HAZOP	Hazard and Operability study
hTMM	hybrid Threat Modelling Method
HWT	Hierarchical Weakness Tree
IACS	Industrial Automation and Control Systems

ICT	Information and Communication Technology
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IoI	Item of Interest
IR	(NIST) Internal or Interagency Report
ISA	International Society of Automation
ISA	International Society of Automation
ISO	International Organization for Standardization
IT	Information Technology
ITE	Intelligent Threat Engine
JSON	JavaScript Object Notation
LINDDUN	Link ability, Identifiability, Nonrepudiation, Detectability, Disclosure of information, Unawareness, Noncompliance
MDE	Model-Driven Engineering
MQTT	Message Queuing Telemetry Transport
MUD	Manufacturer Usage Description
NIST	National Institute of Standards and Technology
NVD	(US) National Vulnerability Database
OCTAVE	Operationally Critical Threat, Asset, and Vulnerability Evaluation
OEM	Original Equipment Manufacturer
OT	Operational Technology
OVVL	Open Weakness and Vulnerability Modeler
OWASP	Open Web Application Security Project
PASTA	Process for Attack Simulation and Threat Analysis
PnG	Persona non Grata
PyTM	Pythonic framework for Threat Modelling
QoS	Quality of Service
QTMM	Quantitative Threat Modelling Method
RUI	Relied Upon Interface
RUMI	Relied Upon Message Interface
RUPI	Relied Upon Physical Interface
SCADA	Supervisory Control and Data Acquisition
SDL	Security Development Lifecycle
SDLC	Security Development Lifecycle Chain
SEI	Software Engineering Institute
SoS	System-of-Systems

SP	Special Publication
SQL	Structured Query Language
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS), Elevation of Privilege
SUC	System Under Consideration
SysML	Systems Modelling Language
TTP	Tactics, Techniques, Procedures
UML	Unified Modelling Language
VAST	Visual, Agile, and Simple Threat
WCT	Weakness Chains Tree
XMI	XML Metadata Interchange
XML	Extensible Markup Language
YANG	Yet Another Next Generation
ZCR	Zone and Conduit risk assessment Requirement

Executive Summary

The main goal of this deliverable is to introduce the methodologies behind *ResilBlockly*, the Model-Driven Engineering tool that evolves Blockly4SoS, and which evolution has been devised in the context of BIECO in order to support not only the modelling of the main cyber-physical systems concepts, but also to perform hazard analysis, to enable threats modelling, and to address risk assessment from both the safety and security perspectives. Moreover, the evolved tool enables to graphically represent attack paths and, thanks to the integration with a simulation engine, to represent interactions between system components both under normal conditions and during attacks.

The document reports key concepts, and reviews and compares threat modelling solutions and reference security standards (Section 2, and Appendices A, and B).

It also describes: i) a novel hazard analysis methodology derived from systematic application of HAZOP to functions and interfaces (Section 3); ii) a threat modelling and risk assessment process which leverages CWE, CVE and CAPEC catalogues, provides attack trees/graphs visualization, and whose steps derive the integration of common steps of the analysed standards (Section 4); iii) introduces the Manufacturer Usage Description (MUD) standard, and presents how the latter is being extended with characteristics originated from the modelling and analysis (Sections 5 and 6).

Then, Section 6 also introduces ResilBlockly main features and details how the above-mentioned methodologies are provided within the tool.

Finally, Section 7 describes the ResilBlockly Simulation Engine, a completely new simulator which enables to simulate the interactions between components modelled in ResilBlockly.

Table of Contents

Technical References	1
Revision History	2
List of Contributors	2
Disclaimer	3
Acronyms	4
Executive Summary	7
Table of Contents	8
List of Figures	11
List of Tables	15
1. Introduction	17
2. State of the Art on Threat Modelling and Risk Analysis	19
2.1. Threat and Attack Paths Modelling Methods and Tools	19
2.1.1. Attack Tree	20
2.1.2. STRIDE	26
2.1.3. VAST – Visual, Agile, Simple Threat Modelling Method	32
2.1.4. LINDDUN	34
2.1.5. PASTA - Process for Attack Simulation and Threat Analysis	35
2.1.6. TRIKE	36
2.1.7. OCTAVE - Operationally Critical Threat, Asset, and Vulnerability Evaluation	37
2.1.8. ADVISE - ADversary Vlew Security Evaluation	38
2.1.9. Security Cards	41
2.1.10. PnG - Persona non Grata	42
2.1.11. hTMM - hybrid Threat Modelling Method	43
2.1.12. CORAS	44
2.1.13. HAZOP – HAZard and OPerability Study	45
2.1.14. Other Tools	47
2.2. Risk Rating and Security Scoring Systems	51
2.2.1. CWSS - Common Weakness Scoring System	51
2.2.2. CVSS - Common Vulnerability Scoring System	53
2.2.3. VERACODE	54
2.2.4. DREAD	55
2.2.5. OWASP Risk Rating	56

2.2.6. Cenzip HARM - Hailstorm Application Risk Metric	56
2.3. Comparison of the Threat Modelling Methodologies and Risk Rating Systems	57
2.4. Reference Security Standards for Threat Analysis and Risk Assessment.....	59
2.4.1. Introduction.....	59
2.4.2. Basic Concepts and Risk Model.....	60
2.4.3. Overview of the Selected Standards	67
2.4.4. From Standards to BIECO Risk Assessment Process	81
2.5. Modelling of CPSoS	83
2.5.1. SoS Basic Concepts	84
2.5.2. AMADEOS SoS Conceptual Model	86
2.5.3. AMADEOS SoS SysML Profile	86
2.5.4. Blockly4SoS	93
3. Definition of a HAZOP-based Risk Assessment Methodology.....	98
3.1. Functional Hazard Analysis	99
3.2. Interface Hazard Analysis.....	101
3.3. THROP: HAZOP for Security Assessment.....	102
4. Definition of a Threat Modelling and Security Risk Assessment Methodology	104
4.1. Preparation	105
4.2. Identification of the Assets	105
4.3. Identification and Modelling of Threats	105
4.3.1. CWE - Common Weakness Enumeration.....	106
4.3.2. CVE - Common Vulnerabilities and Exposures Catalogue	110
4.3.3. CAPEC - Common Attack Pattern Enumeration and Classification	112
4.3.4. The Weaknesses and Vulnerabilities Identification Process	115
4.4. Graphical Representation and Attack Paths Analysis	116
4.4.1. Hierarchical Weakness Tree.....	117
4.4.2. Weakness Chains Tree (WCT).....	117
4.4.3. Attack Path Tree and Attack Path Graph.....	118
4.4.4. Other Trees	120
4.5. Impact and Severity.....	121
4.5.1. Vulnerabilities Impact and CVSS Base Score.....	121
4.5.2. Weaknesses Impact and Severity	122
4.6. Likelihood determination	123
4.7. Risk.....	123

4.8. Assessment Report.....	124
5. Applicability of MUD Standard in Modelling Systems and Interfaces.....	125
5.1. The Manufacturer Usage Description Standard.....	125
5.1.1. The MUD Model.....	126
5.2. Limitations of the MUD standard	127
6. Implementation of the Methodologies in ResilBlockly.....	129
6.1. From Blockly4SoS to ResilBlockly.....	129
6.1.1. General Improvements.....	129
6.1.2. Introduction of Profiling and Modelling Features	130
6.1.3. Interoperability, Ecore and EMF.....	132
6.1.4. Conversion of SoS profile and Import of the ecore	134
6.2. Using the MUD standard for modelling.....	135
6.2.1. Importing the original MUD file in ResilBlockly	136
6.2.2. Exporting the extended MUD file from ResilBlockly	136
6.3. Hazard Analysis in ResilBlockly	136
6.3.1. Functional Hazard Analysis in ResilBlockly.....	137
6.3.1. Interface Hazard Analysis.....	138
6.4. Identification of Assets and Threat Modelling in ResilBlockly.....	141
6.5. Attack Paths and retrieval of additional Threats	143
6.6. Risk Assessment in ResilBlockly.....	144
7. The ResilBlockly Simulation Engine.....	146
8. Conclusions.....	151
9. References.....	152
Appendix A – Comparison Between Threat Modelling Tools	156
Appendix B – Risk Assessment Steps Descriptions and Reference Standards	162

List of Figures

Figure 1 – Goal-Oriented Attack Tree	21
Figure 2 – An ADTree modelled with ADTool representing an attack on a bank account	22
Figure 3 - An ADTree where quantitative values are expressed	23
Figure 4 – Attack Tree modelled with Isograph's tool, where the path of the minimum attack cost (on the left, in orange) and the mitigation applied (on the right, in green) are highlighted	24
Figure 5 Overview of RiskTree Designer software	25
Figure 6 – An example of prioritized risk table	26
Figure 7 Example of application of STRIDE to Commerce Server installation	27
Figure 8 STRIDE Per Element	27
Figure 9 A Possible Threat Analysis process based on STRIDE (image inspired by).....	28
Figure 10 - Model Diagram Example in Microsoft Threat Modeling Tool	28
Figure 11 An example of threat list provided by Microsoft Threat Modeling Tool	29
Figure 12 Threat Model Report produced with Microsoft Threat Modeling Tool	29
Figure 13 Example of DFD in OVVL	30
Figure 14 Threat (on the left) and Vulnerability (on the right) analysis within OVVL.....	30
Figure 15 – Data Flow Diagram with Threat Dragon Tool	31
Figure 16 – Example of Attack Tree for tampering category of STRIDE	32
Figure 17 - Example of Tampering Attack Tree with CVSS-based scores and no mitigations	32
Figure 18 - Example of a Web Application Diagram in ThreatModeler	33
Figure 19 - Example of Dashboard	34
Figure 20 - LINDDUN framework	35
Figure 21 – Steps of TRIKE Methodology	36
Figure 22 Trike Tool [78]: rules tree for intended actions (left); autogenerated attack tree (right).	37
Figure 23 Trike Tool [99]: Risk grid/threat visualization (left); actors view (right).....	37
Figure 24 – Phases of OCTAVE methodology	38
Figure 25 - The ADversary View Security Evaluation (ADVISE) method	39
Figure 26 - An Attack Execution Graph (AEG) represents possible attacks	39
Figure 27 An example of the ADVISE attack execution graph editor page	40
Figure 28 An example of the ADVISE adversary profile editor page in Mobius	41
Figure 29 - Security Card Dimensions	42
Figure 30 - Example of Persona non Grata	43

Figure 31 – Example of Risk modelled with CORAS, from	45
Figure 32 - HAZOP Analysis Process	46
Figure 33 Sample HAZOP Worksheet	46
Figure 34 - Example of Architecture	47
Figure 35 Threats view in Irius Risk.....	48
Figure 36 - Example of Critical Path in securiCAD	49
Figure 37 – Overview of Results given by securiCAD	49
Figure 38 - Example of Diagram realized with PyTM	50
Figure 39 - SD Expert Assessment	51
Figure 40 - CWSS Metric Groups	52
Figure 41 – The groups of CVSS metrics	54
Figure 42 - DREAD Mnemonic.....	55
Figure 43 Generic risk model and key risk factors from NIST SP 800-30	61
Figure 44 Risk Assessment Process – Step 2 Conduct Assessment Expanded View	70
Figure 45 Assessment for the interface between control systems and equipment, including level of impact and security requirements	74
Figure 46 Flow diagram of the ISA 62443-3-2 Risk Assessment Process	77
Figure 47 ETSI test-based risk security assessment	79
Figure 48 Test based risk identification.....	80
Figure 49 Test-based security risk estimation	80
Figure 50 Overview of SoS conceptualization in AMADEOS	84
Figure 51 Overview of AMADEOS SysML profile and viewpoint-related packages	88
Figure 52 SoS Architecture Package.....	89
Figure 53 SoS Communication package.....	90
Figure 54 SoS Dependability package.....	91
Figure 55 SoS Security package	92
Figure 56 Flow of MDE using the Blockly4SoS.....	94
Figure 57 – Architecture viewpoint related blocks in Blockly4SoS	95
Figure 58 Providing services through a RUMI	96
Figure 59 Example of behaviour of a service	97
Figure 60 Process view of the HAZOP-based methodology (in blue the steps assisted by ResilBlockly, in white the ones to be addressed offline)	99
Figure 61 Overview of the Methodology (in blue the steps object of this deliverable and assisted by ResilBlockly, as well as databases or external data integrated within it)....	104
Figure 62 Symbols for Weaknesses abstractions and types in CWE	107

Figure 63 Example of NVD Severity.....	112
Figure 64 Example of HWT with all the child weaknesses for the “CWE 287: Improper Authentication”	117
Figure 65 Example of WCT having as root the CWE 289: Authentication Bypass by Alternate Name.....	118
Figure 66 Example of Attack Path Tree for CWE-648	119
Figure 67 Attack Path Graph example related to CWE-648.....	120
Figure 68 Overview of the (vulnerability) risk determination methodology, integrating CVSS and the NIST SP 800-30 risk matrix	124
Figure 69 MUD standard model of the “mud” container.....	126
Figure 70 The GUI of ResilBlockly for the choice between Profiling and Modelling Features	130
Figure 71 The ResilBlockly flow and categories of users (to be compared with Blockly4SoS flow in Figure 56)	131
Figure 72 Example of Derivation of Profiles and Models	131
Figure 73 Hierarchy of Ecore components	133
Figure 74 Types of Relation in EMF.....	133
Figure 75 Key elements in ResilBlockly Profile Designer.....	133
Figure 76 Two of the classes composing the Architecture viewpoint reproduced in EMF	134
Figure 77 A portion of the SoS Profile imported as ecore into ResilBlockly	134
Figure 78 A portion of the exported ecore XML showing SoS and CS	135
Figure 79 ResilBlockly user interface to import MUD information.....	136
Figure 80 The ResilBlockly Profile Designer - Risk designer GUI with the Functions tab selected and an example of function identified	137
Figure 81 The ResilBlockly Model Designer - Risk Assessment GUI with the functional analysis tab selected and the interface for specifying the template.....	137
Figure 82 The ResilBlockly Model Designer - Risk Assessment GUI with the functional analysis tab selected and the result of a functional analysis	138
Figure 83 Logical representation of a Sensor Network example with two interfaces....	138
Figure 84 Example of profile with the meta-modelling of interfaces.....	139
Figure 85 The ResilBlockly Profile Designer - Risk Designer GUI with the Interfaces tab selected and the interfaces definition process ongoing	139
Figure 86 Example of model with modelling simple interfaces	140
Figure 87 The ResilBlockly Model Designer - Risk assessment GUI with the Interfaces tab selected and the result of the analysis	140
Figure 88 The ResilBlockly Profile Designer - Risk designer GUI with the Weaknesses tab selected and some random weaknesses associated to a sample class block.....	141

Figure 89 The CWE search interface with a sample example	141
Figure 90 The CAPEC search interface for the retrieval of attack patterns and association of related weaknesses	142
Figure 91 The ResilBlockly Profile Designer - Risk designer GUI with the Vulnerabilities tab selected and some random vulnerabilities associated to a sample class block	142
Figure 92 A generic Attack Path Graph.....	144
Figure 93 The ResilBlockly Profile Designer - Risk designer GUI with the Vulnerabilities tab with the CVSS base score(s) from the NVD integrated	145
Figure 94 Overview of the simulation process and integration of ResilBlockly model with external IDE and simulation engine.....	146
Figure 95 The BaseComponent . java class declaration.....	147
Figure 96 An example of auto-generated Java Class for a specific Model Component named "DHT11".....	147
Figure 97 An override example of the executeBehaviour() method	148
Figure 98 An example of ResilBlockly model	148
Figure 99 An example of auto-generated interface instance	149
Figure 100 Example of interfaces between simulated and real systems.....	149
Figure 101 An example of a real time chart, developed as behaviour of a "Dashboard" component, with third-party dependencies	150

List of Tables

Table 1 Attack paths for the tree in Figure 4, where the underlined path has the highest risk of threat occurrence	24
Table 2 STRIDE threat model.....	26
Table 3 - Steps and Activities of PASTA Methodology	35
Table 4 – Generic HAZOP Guide Words	45
Table 5 - Risk rating category-impact.....	55
Table 6 – Summary of the Threat Modelling Methods.....	57
Table 7 Summary of the Risk Rating and Scoring Systems	59
Table 8 - Taxonomy of Threat Sources	64
Table 9 Assessment scales for the level of risk.....	68
Table 10 Assessment scales for the level of risk–Combination of Likelihood and Impact	69
Table 11 – Steps of Risk Assessment Process from	69
Table 12 Categories of Adversaries to Information Systems in NIST.IR 7628	74
Table 13 ISA/IEC 62443	76
Table 14 Security Level and Threat Actors Definition in ISA/IEC 62443-3-3.....	78
Table 15 Full list of standards analysed in the context of this activity.....	81
Table 16 – BIECO Risk Assessment Process.....	82
Table 17 Possible HAZOP Keywords and their meaning for the Functional Analysis.....	99
Table 18 Columns in the HAZOP Functional Analysis Template	100
Table 19 Possible HAZOP Keywords and their meaning for the Interface Analysis	101
Table 20 Columns in the HAZOP Interface Analysis Template.....	101
Table 21 Additional Keywords identified for the THROP functional and interface security analysis.....	103
Table 22 Status of CVEs	110
Table 23 Chains leading to CWE 289: Authentication Bypass by Alternate Name	118
Table 24 Attack Paths represented with APT having CWE-648 as root weakness	119
Table 25 Qualitative and quantitative severity rating scale in CVSS	122
Table 26 Comparison of Profiling and Modelling in Blockly4SoS and ResilBlockly	130
Table 27 Tools based on Attack Tree Methodology	156
Table 28 Tools based on STRIDE methodology	157
Table 29 The Tool based on VAST methodology.....	158
Table 30 The Framework which includes an implementation of ADVISE	159
Table 31 The tool based on Trike methodology.....	159

Table 32 The CORAS Tool.....	159
Table 33 Other Tools	160

1. Introduction

The cybersecurity of an ICT system or component that is part of an ecosystem - or of a System-of-Systems (SoS)-, is limited by the weakest element of the chain. This is due to the fact that an attacker will likely try to target first the weakest component and then identify the attack path that will let access to the rest of the systems, eventually by exploiting additional weaknesses and vulnerabilities. This issue has strong implications, as organizations can be at risk just taking part in an ecosystem, independently on their security measures. Thus, it is necessary to have an effective management of risks that includes a detailed view of all the weaknesses and vulnerabilities that can affect the complete ICT supply chain, as well as each of its components.

Examples of supply chain risks may include insertion of counterfeits, tampering, theft, insertion of malicious software and hardware, as well as poor development practices that could impact the whole supply chain and have consequences also on safety. Therefore, managing the security and safety risks for the whole ecosystem is fundamental.

This deliverable describes a set of methodologies and processes for risk analysis that have been designed in the context of BIECO in order to address the above-mentioned challenges of ICT supply chains and ecosystems.

The methodologies are supported by the conception and development of *ResilBlockly*, a tool that assists designers in the early prototyping and design-time analysis phases, and which evolves an existing tool named *Blockly4SoS*. *Blockly4SoS* is the supporting facility proposed as result of AMADEOS project [104]. The design and modelling of complex ecosystems have to address several challenges, as the time required for early prototyping, the cost of modelling large and complex SoS due to their intrinsic complexity, as well as scalability, readability, manageability of the model. Since AMADEOS addressed and solved the above challenges, we choose *Blockly4SoS* as the starting point, evolving it especially for addressing security and risk related concepts.

Along with a wide set of general improvements, *ResilBlockly*, the new version of this model-driven engineering software, comes with a list of new features for threat modelling, hazard analysis, safety and security risk assessment. Moreover, the tool allows to identify those components, functions and interfaces that are most vulnerable and might cause the greatest impact if compromised. Furthermore, it allows to graphically represent the attack paths and patterns that an adversary may follow in order to penetrate the system or component. In addition, the tool complies with the Manufacturer Usage Description (MUD) [130] standard for specification of network policies, and extends the MUD with a set of characteristics originated from the modelling and analysis activities, allowing the user to generate the extended MUD file. Finally, it also permits to simulate the interactions between components (e.g., when attacks are exploited) thanks to the integration with a completely new simulation engine.

The document is structured as follows. Section 2 deals with the introduction of key concepts and traces the landscape of several existing solutions, tools and methodologies useful for modelling complex systems-of-systems, identifying and representing their potential threats, and analysing the intrinsic security risk according to the reference standards. Comparisons of the tools and further details about the standards are given in the appendices.

In Section 3, there is the description of a methodology derived from a state-of-the-art approach called HAZOP, for a systematic application of a hazard analysis and risk

assessment to specific parts of the system model: functions and interfaces; the methodology allows users, already in the early prototyping, to automatically identify - thanks to customizable guidewords and templates-, the potential hazards matched to these elements of the modelled system.

Then, Section 4 defines a threat identification and modelling methodology that supports in the phases of a security risk assessment process, from the identification of assets and threats (potential weaknesses and vulnerabilities affecting system components and interfaces), to the determination of impact, likelihood and risk, going through the analysis of attack paths.

Section 5 introduces the Manufacturer Usage Description (MUD) standard, and discusses its limitations, identifying a set of relevant characteristics that the original MUD file is not able to represent and that constitute a MUD model extension.

Furthermore, this deliverable provides in Section 6 a description of ResilBlockly, highlighting the main differences and improvements with regard to its previous version and how all the above-mentioned methodologies are specifically introduced within the tool, with the help of some examples.

Finally, Section 7 describes a completely new simulation engine that has been designed and implemented, and which enables to simulate the models realised with, and exported from, ResilBlockly.

2. State of the Art on Threat Modelling and Risk Analysis

This section introduces key concepts and traces the landscape of several existing solutions, tools and methodologies useful for modelling complex systems-of-systems, identifying and representing their potential threats and analysing the intrinsic security risks according to the reference standards. The main purpose of the research described in this section is to acquire knowledge about the best practices in the above-mentioned areas, to understand the pros and cons of the existing approaches, and to determine whether and how it is possible to design an integrated and all-encompassing solution for addressing this so wide and complex set of activities.

Section 2.1 deals with a brief introduction on threat modelling; describes the most popular threat modelling methodologies and the related tools typically employed to perform it, where existing. Section 2.2 presents some of the main risk rating and security scoring systems; Section 2.3 provides a summary and pros-cons comparison of all the methodologies presented, while the full comparison between the tools is given in Appendix A.

Then, Section 2.4 briefly introduces the basic risk related concepts, and provides an overview of the main reference standards and guidelines considered in the context of task T6.2, starting from standards related to the BIECO use cases domains (i.e., NIST SP 800-30 [43], NIST.IR 7628 [87], ISA/IEC 62443 [92]), and extending the analysis to several others. The result is a 9-step risk assessment process which has been outlined integrating common steps and similarities in the security life cycles of the standards. Further details on the association between steps and descriptions with the reference standards are provided in Appendix B.

Finally, Section 2.5 provides the required background about modelling of complex Cyber-Physical System-of-Systems (CPSoS), reviews some of the main contributions of AMADEOS project [104] and introduces Blockly4SoS main distinctive features.

2.1. Threat and Attack Paths Modelling Methods and Tools

Following the definition from NIST SP 800-154: threat modelling *is a form of risk assessment that models aspects of the attack and defense sides of a particular logical entity, such as a piece of data, an application, a host, a system, or an environment.*

Threat modelling methodologies can be used to create an abstraction of a system, in order to catalogue potential threats that may arise, but often include the outlining of profiles of potential attackers and their goals. Moreover, it is tightly linked to other security activities as risk assessment, security testing, and so on, and can provide useful inputs for them.

The main steps to build a scalable and repeatable threat modelling process are:

1. Characterize the system, identify assets and access points;
2. Identify, prioritize, and focus on high-risk threats;
3. Identify mitigation approaches;
4. Identify potential adversaries;
5. Reporting and operationalizing.

Threat modelling helps *threat intelligence*¹ analysts to identify, classify, and prioritize threats and to ensure effective documentation and reporting. In fact, an effective threat intelligence report helps the security defence and the security operations team protecting ICT assets from threats.

While the target of the modelling varies to different domains and levels of abstraction, e.g., systems, networks, software, hardware devices, business processes, and so on, typically, threat modelling is implemented following one of three approaches independently: asset-centric, attacker-centric, and software-centric. This means that the starting point and main focus of the modelling activity could be respectively: the set of assets of a system being modelled, the adversary that puts the system at risk, or the software which underlines the system.

There are several, best practice, threat modelling methodologies that can be applied, and the collection of tools supporting this activity is very large. In the following, a list of the most relevant and used ones is provided.

There are different methodologies applicable and multiple enabling software that assist the threat modelling activity. The purpose of modelling, the domain and characteristics of the object to be modelled, standards, and many other factors can drive the choice of a specific methodology or tool. To our knowledge, the following are the most popular ones.

2.1.1. Attack Tree

Attack tree provides a formal, methodical way of describing the security of systems, based on varying attacks. Basically, it represents attacks against a system in a tree structure, consisting of one root, leaves, and children [2]. This is similar to the *decision trees* [4] used to help with business decisions or the *fault trees* [5] used to understand the reliability of machines and machine-like processes. From the bottom up, child nodes are conditions which must be satisfied to make the direct parent node true; when the node root is satisfied, the attack is complete.

In any complex system, there are several root nodes, each representing a different goal. And there is a number of different strategies to be formulated that could let achieving the overall goal. These strategies can be expressed as a series of intermediate objectives that individually, or in combination, realize the root goal. This decomposition process continues, breaking the intermediate goals into ever finer grained activities. In details, *OR nodes* are used to represent alternatives and *AND nodes* are used to represent different steps toward achieving the same goal.

Figure 1 shows a generic, goal-oriented, attack tree [3]. The root models the overall goal and is represented with an OR gate; intermediate goals are modelled with OR and AND nodes depending on the need of achieving one or all the subgoals, respectively. The leaves, subgoals, are represented through simple rectangular nodes.

¹ Threat intelligence is the analysis of heterogeneous data sources (e.g., open source, social media, human, technical, etc.) to generate meaningful information about existing or emerging threats and threat actors targeted at risk management, security posture, decision making.

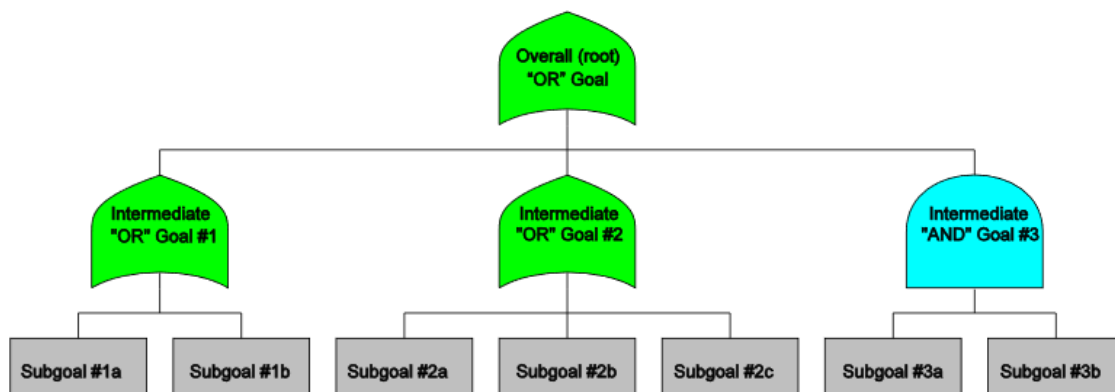


Figure 1 – Goal-Oriented Attack Tree [3]

The formalism of attack tree may slightly vary depending on the type and shape of nodes used, or on the additional information represented (*equipment required, cost, feasibility, likelihood, AND/OR gates, countermeasures, etc.* [2]). When the countermeasures are included in the model, the formalism is also referred as attack-defence tree.

By including a priori likelihood with each node, it is also possible to compute likelihood of higher nodes using Bayes Rule [42]. Since the Bayesian analytic techniques used in fault tree analysis cannot legitimately be applied to attack trees, analysts instead use other techniques to determine which attacks will be preferred by a particular attacker. These may involve comparing the attacker's *capabilities (time, money, skills, equipment)* with the *resource requirements* of the specified attack [2].

Attack trees are used in a variety of applications, e.g.: ICT, in the fields of defence and aerospace, industrial control systems (e.g., for electric power grid) and in general, also, to model threats to both cyber-only and cyber-physical systems.

Typically, three conditions must be present in order for a threat agent² to carry out an attack against a system [3]:

1. The system must have vulnerabilities or weaknesses;
2. The threat agent must have sufficient *capabilities* available to exploit the vulnerabilities.
3. The threat agent must believe they will benefit by performing the attack. The expectation of benefit (as known as *gain*) drives motivation.

There are many tools that represent attacks with this formalism and enable threat modelling according to this methodology.

2.1.1.1. ADTool – Attack-Defense Tree Tool

ADTool (Attack-Defense Tree Tool) [6] is a software that supports security analysis and risk assessment, allowing users to model and analyse attack-defence scenarios represented with attack-defence trees (*ADTree*) [6]. An *ADTree* [7] is a node-labelled rooted tree describing the measures an attacker might take in order to attack a system and the

² often referred as *attacker* or *threat source*. Various taxonomies of threat sources have been developed, as the Appendix D in NIST 800-30 **Error! Reference source not found.** that it provides an exemplary taxonomy of threat sources and associated threat characteristics: Adversarial, Accidental, Structural and Environmental. For other details can refer to document Technical Report 6.2: Analysis and comparison of reference security standards for threat analysis and risk assessment.

NB: The terminology used in this document mainly refers to NIST SP 800-30.

measures that a defender can employ to protect the system. It has nodes of two opposite types: *attack nodes* and *defence nodes*.

The two key features of an ADTree are the representation of *refinements* and *countermeasures*. Every node may also have one child of opposite type, representing a countermeasure. Thus, an attack node may have several children which “refine” the attack and one child which defends against the attack. The defending child in turn may have several children which refine the defence and one child that is an attack node and counters the defence [7]. As shown in Figure 2, refinements are represented below the nodes, and can be disjunctive (i.e., the corresponding of OR gates), or conjunctive (i.e., AND gates).

The purpose of ADTrees is to analyse an attack–defence scenario. The authors of [7] define an attack–defence scenario as a game between two players, the proponent and the opponent. As expected, the root of an ADTree represents the main goal of the proponent [7].

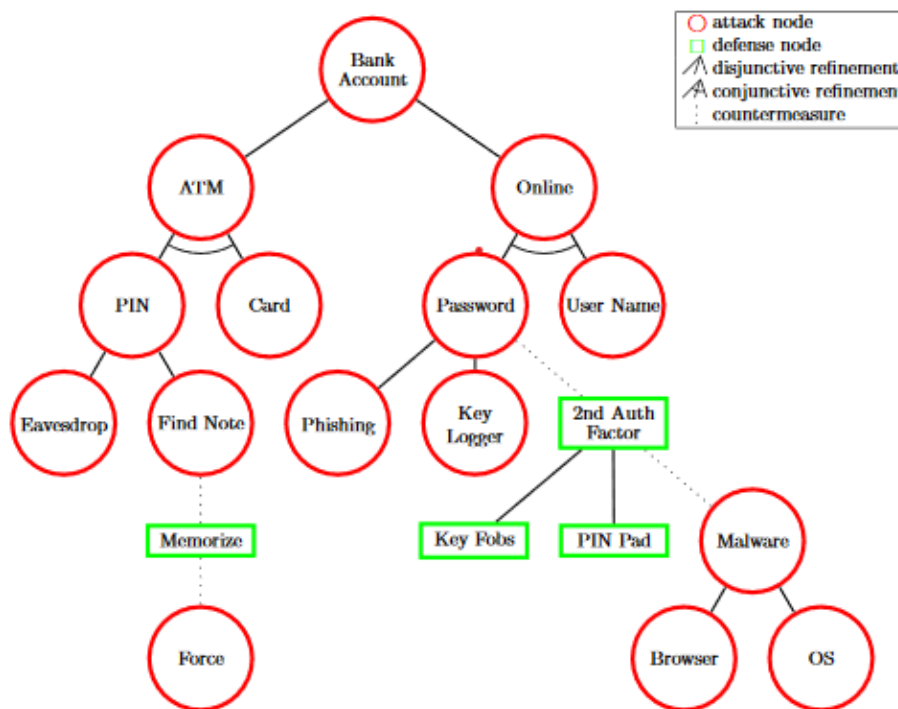


Figure 2 – An ADTree modelled with ADTool representing an attack on a bank account [7]

The tool includes some main features [6] [7]:

- Creation and editing of attack-defence trees and sequential attack trees;
- Modular display of attack-defence trees, which allows modelling of large real-life scenarios;
- Quantitative bottom-up analysis of attack-defence scenarios (see Figure 3 as example). The bottom-up algorithm for evaluation of attributes supports measures as:
 - real values (e.g., time, cost, probability),
 - levels (e.g., required skill level, reachability of the goal in less than k units of time),
 - Boolean properties (e.g., satisfiability of a scenario).
- The measures can be computed from the point of view of an attacker (e.g., the cost of an attack), of a defender (e.g., the cost of defending a system), or relate to both of them (e.g., overall maximum power consumption, as depicted in Figure 3).

- Ranking of possible attacks for certain attribute domains. Custom domains can also be specified by the user;
- Printing, exporting to various formats (i.e., pdf, png/jpg, tex) and saving of attack-defence tree models;
- Customizable layouts.

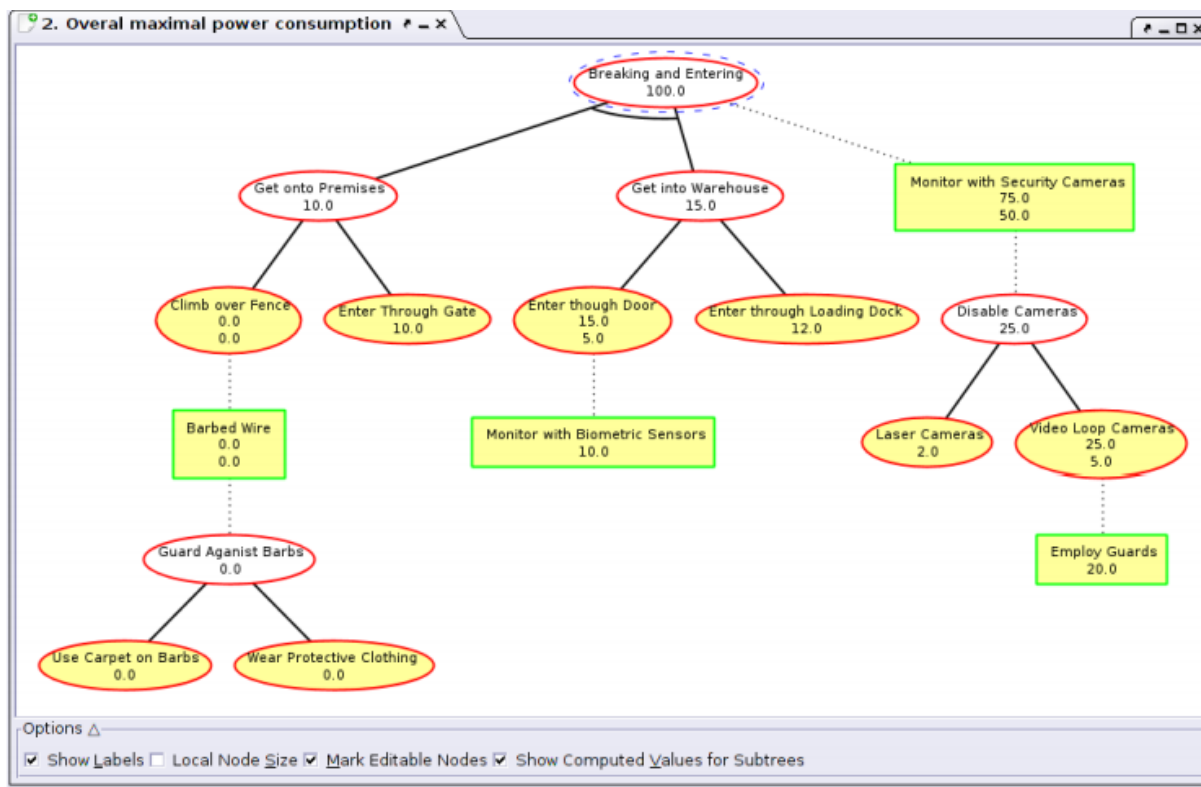


Figure 3 - An ADTree where quantitative values are expressed [7]

2.1.1.2. AttackTree+

AttackTree+ is a tool existing since the 1980s which, according to its developer, Isograph [8], provides an integrated environment for analysing the cybersecurity of automotive systems in keeping with cybersecurity standards ISO 21434 (Road vehicles – Cybersecurity engineering) and SAE J3061 (Cybersecurity Guidebook for Cyber-Physical Vehicle Systems).

It can be considered a suite or toolchain, incorporating different tools for [8]:

- Threat Analysis: provides a graphical interface to construct an asset hierarchy and identify threats to those assets that may be ranked by likelihood, severity and controllability. A template is provided for performing threat analyses in compliance with ISO 26262 (Road vehicles – Functional safety), as well as the HEAVENS methodologies [44];
- Attack Tree Analysis: provides a framework for the construction and analysis of attack tree diagrams. With an attack tree, the user can identify all possible paths to a threat and rank those paths by likelihood. The Attack paths are deduced for the item or component based on historical knowledge of vulnerabilities in similar systems and components;
- Mitigation Tree Analysis: allows the user to identify all possible outcomes of a threat and rank them by their likelihood.

Hence, the tool allows users to model system vulnerability, identify weak spots and improve security using threat analysis and attack trees. It provides to the user various data concerning the attacks [9], as: success probability, cost; impact; source of the attack; attack path; return of Attack.

As shown in Figure 4, the tool allows the user to visualize the path of the minimum attack cost and compare the same path with the defence applied. The figure describes the attack tree for the elevation of privileges attack “A1 Run as Administrator”, and, on the right, the same tree with a mitigation “D3 Malware Download Protection” applied. It can be noticed that in the tree on the right, the defence cost is expressed and the addition of a mitigation has caused the update of probabilities and other existing data.

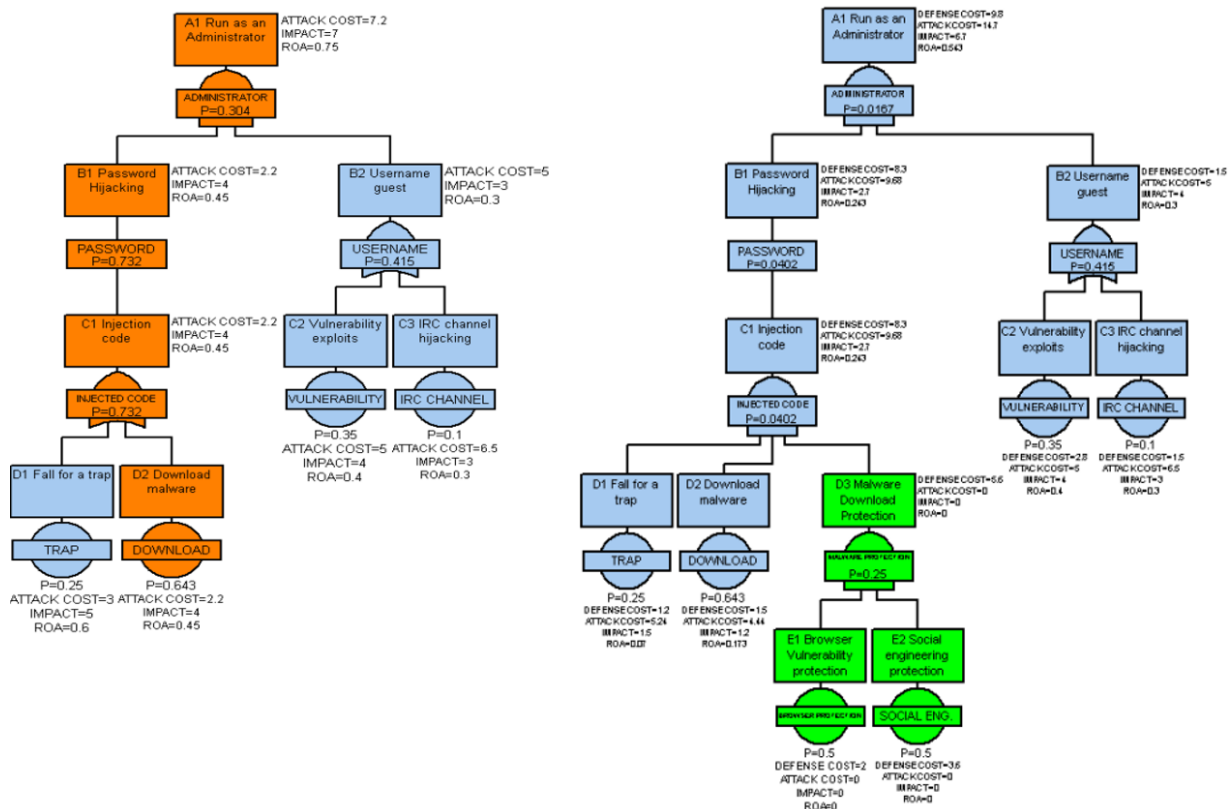


Figure 4 – Attack Tree modelled with Isograph's tool, where the path of the minimum attack cost (on the left, in orange) and the mitigation applied (on the right, in green) are highlighted [9]

Table 1 Attack paths for the tree in Figure 4, where the underlined path has the highest risk of threat occurrence

Attack Source	Attack Path
D1	D1 -> C1 -> B1 -> A1
<u>D2</u>	<u>D2 -> C1 -> B1 -> A1</u>
C2	C2 -> B2 -> A1
C3	C3 -> B2 -> A1

RiskTree

The methodology underlying RiskTree [10] tool is a structured approach for risk management which is adopted by the UK Government [11]. It is based around the concept of attack trees and it provides a systematic way of capturing and prioritizing the risks to business and systems. According to the producers of the tool, its results integrate well with existing business processes, and the risk assessment reports generated can show risks using a variety of data visualizations [10].

The RiskTree process is based on a typical risk management process and provides a structured and systematic way of cataloguing the risks to a system or process [11]. The steps composing the process are:

1. Identification of the risks by means of facilitated workshops and design of the *RiskTree* using the *RiskTree Designer*;
2. Risk Assessment in a consistent way and with consensus from the participants;
3. Prioritization of the risks based on the analysis executed by the *RiskTree Processor*;
4. Management and tracking of the risks, which are placed into a risk register.

The above-described process is supported by an online, cloud-hosted software-as-a-service. The RiskTree software calculates the level for each risk and returns a prioritized list (it can generate a sorted risk table for review). Countermeasures can then be applied, and their effects viewed on both the tree and the risk table. As shown in Figure 5, the tool allows the creation of trees within the browser environment; the progresses can be saved in XML [11].

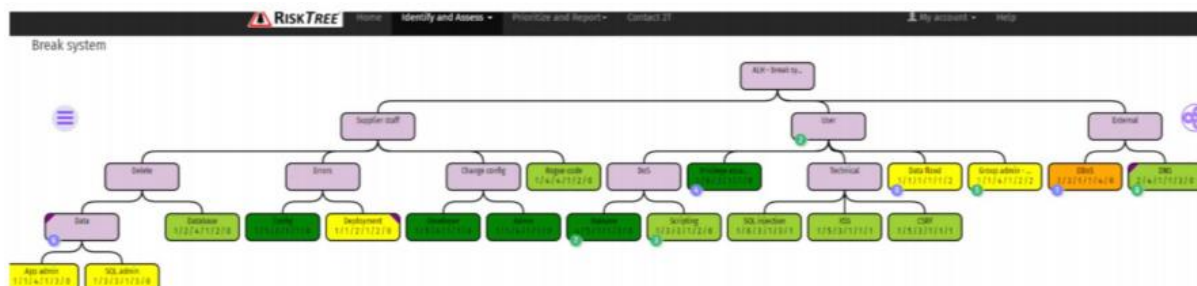


Figure 5 Overview of RiskTree Designer software [11]

The tree embeds indicators for parameters as cost, complexity, consequences, reward, damage or replay. When the tree is built, and those parameters are specified, it can then be submitted to the on-line service for security assessment. The prioritized table of risks is then generated; an example of this table, generated from the above tree, is shown in Figure 6. Risks are sorted on a six-point scale, from Very Low (VL) to Very High (VH).

More details on the process and the tool (e.g., the different type of charts showing risk levels) are available in [11].

Name	Cost	Complexity	Consequences	Reward	Damage	Replay	Risk	Countermeasure
ALH - break system » External » DDoS	3	3	1	1	4	0	MH	
							MH	
ALH - break system » User » Data flood	1	1	3 (1)	1	1	2	MH	
							M	CM2
ALH - break system » Supplier staff » Delete » Data » App admin	1	1	4	1	3	0	M	
							M	
ALH - break system » Supplier staff » Delete » Data » SQL admin	1	3	3	1	3	0	M	
							M	

Figure 6 – An example of prioritized risk table [11]

2.1.2. STRIDE

STRIDE is an acronym and a threat model created by Microsoft engineers, which is meant to guide the discovery of threats in a system [27]. The meaning of the acronym (i.e., the type of threats), security properties and description is shown in Table 2.

Table 2 STRIDE threat model			
Type of Threat	Security property threatened	Description	
S	Spoofing	Authentication	Impersonating something or someone known and trusted
T	Tampering	Integrity	Modifying data on disk, memory, network etc.
R	Repudiation	Non-repudiation	Claim to not be responsible for an action
I	Information Disclosure	Confidentiality	Providing information to someone who is not authorized
D	Denial of Service (DoS)	Availability	Denying or obstructing access to resources required to provide service
E	Elevation of Privilege	Authorization	Allowing access to someone without proper authorization

It is a well-known threat-modelling method used to help reasoning and finding threats to a system, and it is often used in conjunction with a model-based representation of the target system that can be constructed in parallel (see the example shown in Figure 7). STRIDE can be adopted to model not only cyber systems, but also cyber-physical ones. Microsoft is not maintaining STRIDE anymore [37], however, it is included in their Security Development Lifecycle (SDL) [153] and in the Threat Modelling Tool [12] described in Section 2.1.2.1.

The typical steps required for applying the STRIDE model are:

- 1) identification of the assets,
- 2) definition of the trust levels of system users,
- 3) building of the Data Flow Diagram (DFD)³ [46],

³ A Data Flow Diagram provides a graphical representation of the data flow through an information system. The threat modelling produces key artefacts and uses those diagrams as mappings with STRIDE to identify threats.

- 4) identification of the threats based on the six types of threats considered in the STRIDE methodology.

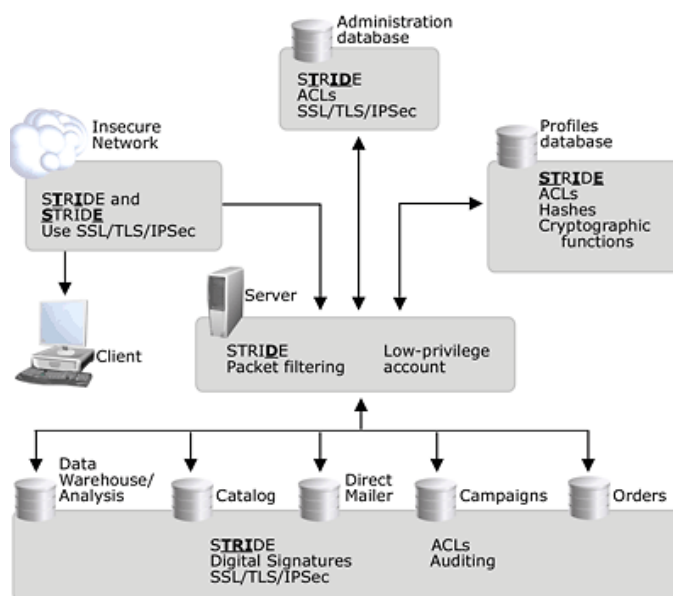


Figure 7 Example of application of STRIDE to Commerce Server installation [63]

An example of the application of STRIDE methodology is shown in Figure 7, where elements of a commerce server installation are described in terms of threats they are exposed to. In this case, those threats are represented as bold and underlined letters of the STRIDE word. The model also integrates possible mitigations.

Different threats affect each type of element





Element	S	T	R	I	D	E
 External Entity	✓		✓			
 Process	✓	✓	✓	✓	✓	✓
 Data Store		✓	✓	✓	✓	✓
 Dataflow		✓	✓	✓	✓	✓

Figure 8 STRIDE Per Element [47]

Several variations of STRIDE have been proposed [64]. One of them, which is known as *STRIDE-per-element* (shown in Figure 8), lists generic elements (external entity, process, data store, dataflow) and depicts them as in data flow diagram notation; the different threats affecting each type of element are represented with a check symbol. In this case, the threat modelling process involves the following steps:

1. Retrieve elements from a Data Flow Diagram (DFD)
2. Find threats from element-STRIDE table
3. Check whether the records in the table are appropriate

A different evolution is known as *STRIDE-per-interaction* and its comparison with STRIDE-per-element can be found in [64].

Finally, a flow diagram with the steps of threat modelling and possible connections between STRIDE, attack tree modelling and patterns is shown in Figure 9.

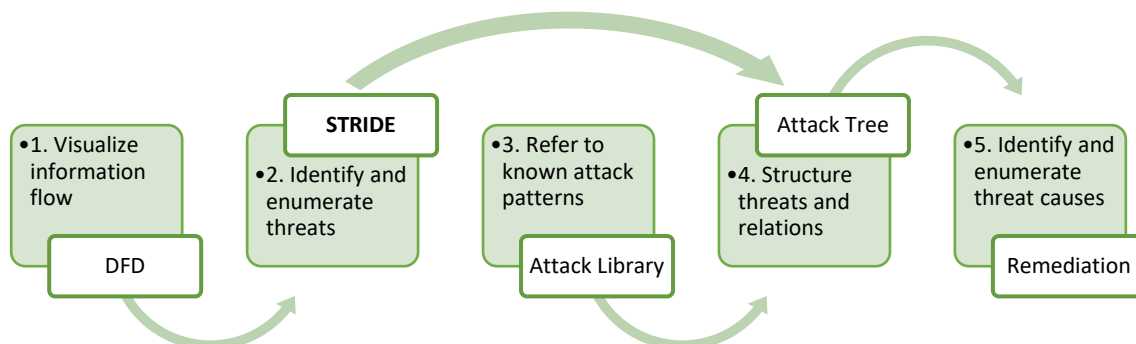


Figure 9 A Possible Threat Analysis process based on STRIDE (image inspired by [64])

Some of the tools that implement STRIDE methodology are described in the following sections.

2.1.2.1. Threat Modelling Tool

Threat Modelling Tool is a core element of the Microsoft SDL, and its underlying modelling methodology is based on STRIDE [12]. The tool can be used to identify threats, attacks, vulnerabilities, and countermeasures that could affect an application. As can be noticed in the overview of Figure 10, the model is a DFD, a standard notation for visualizing system components, and security boundaries. It provides templates which include a pre-defined set of stencils [12].

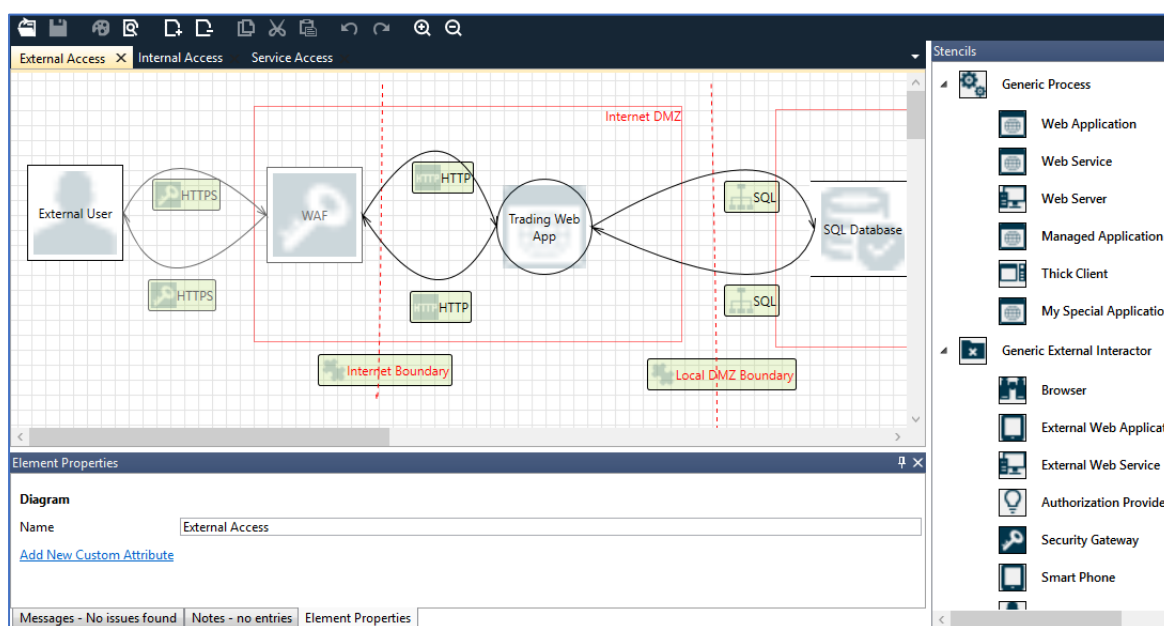


Figure 10 - Model Diagram Example in Microsoft Threat Modelling Tool [48]

The tool helps threat modelers identifying classes of threats they should consider based on the structure of their software design; the threat analysis produces a threat list, also exportable in .xls format, an example of which is provided in Figure 11. Possible mitigations for the detected threats are also provided.

ID	Diagram	Changed By	Last Modified	State	Title	STRIDE Catg	Description	Justification	Interaction	Severity	Possible Mitig	SDL Phase
46	Diagram 1		Generated	Not Started	An adversary m	Elevation of Pri	An adversary m		Request	High	Implement impi	Design
47	Diagram 1		Generated	Not Started	An adversary ca	Information Dis	An adversary ca		Request	High	Do not expose s	Implementation
48	Diagram 1		Generated	Not Started	An adversary m	Information Dis	An adversary m		Request	High	Ensure that the	Implementation
49	Diagram 1		Generated	Not Started	An adversary ca	Information Dis	An adversary ca		Request	High	Implement Cert	Implementation
50	Diagram 1		Generated	Not Started	An adversary ca	Information Dis	If application sa		Request	High	Encrypt sensitiv	Implementation
51	Diagram 1		Generated	Not Started	An adversary ca	Information Dis	An adversary ca		Request	High	Do not expose s	Implementation
52	Diagram 1		Generated	Not Started	Attacker can de	Repudiation	Proper logging		Request	Medium	Ensure that aud	Implementation
53	Diagram 1		Generated	Not Started	An adversary ca	Spoofing	Ensure that TLS		Request	High	Verify X.509 cer	Implementation
54	Diagram 1		Generated	Not Started	An adversary ca	Spoofing	Attackers can e		Request	High	Explicitly disabl	Implementation
55	Diagram 1		Generated	Not Started	An adversary ca	Spoofing	Phishing is attai		Request	High	Verify X.509 cer	Implementation
56	Diagram 1		Generated	Not Started	An adversary m	Spoofing	If proper authen		Request	High	Consider using	Design
57	Diagram 1		Generated	Not Started	An adversary ca	Tampering	An adversary ca		Request	High	Obfuscate gene	Design
58	Diagram 1		Generated	Not Started	An adversary ca	Tampering	SQL injection is		Request	High	Ensure that typ	Implementation
59	Diagram 1		Generated	Not Started	An adversary ca	Tampering	An adversary ca		Request	High	Encrypt section	Implementation

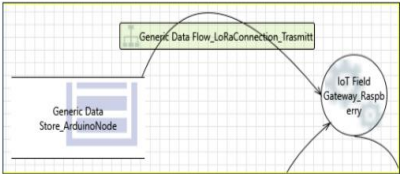
Export Csv 50 Threats Displayed, 50 Total

Figure 11 An example of threat list provided by Microsoft Threat Modelling Tool

Finally, the tool allows to save and print out a threat modelling report, as depicted in Figure 12 Threat Model Report produced with Microsoft Threat Modelling Tool.

Created on 14/05/2020 11:27:33
Threat Model Name:
Owner:
Reviewer:
Contributors:
Description:
Assumptions:
External Dependencies:

Interaction: Generic Data Flow_LoRaConnection_Transmitt



Threat Model Summary:

Not Started	50
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	50
Total Migrated	0

1. An adversary may execute unknown code on IoT Field Gateway_Raspberry [State: Not Started] [Priority: High]

Category: Tampering
Description: An adversary may launch malicious code into IoT Field Gateway_Raspberry and execute it.
Justification: <no mitigation provided>
Possible Mitigation(s): Ensure that unknown code cannot execute on devices. Refer: <https://aka.ms/tmtconfigmgmt#unknown-exe>
SDL Phase: Design

2. An adversary may gain access to the field gateway by leveraging default login credentials. [State: Not Started] [Priority: High]

Category: Spoofing
Description: An adversary may gain access to the field gateway by leveraging default login credentials.
Justification: <no mitigation provided>
Possible Mitigation(s): Ensure that the default login credentials of the field gateway are changed during installation. Refer: <https://aka.ms/tmtconfigmgmt#default-change>
SDL Phase: Design

Figure 12 Threat Model Report produced with Microsoft Threat Modelling Tool

2.1.2.2. OVVL – Open Weakness and Vulnerability Modeler

Open Weakness and Vulnerability Modeler (OVVL) [21] [52], is a tool based on an extension of STRIDE which supports threat modelling in the early stages of the software development lifecycle. The main features of the tool are: model design, threat and vulnerability analysis [52].

An abstraction of complex software systems can be created based on DFDs, where different elements - *interactors*, *processes* and *data-stores* – are placed on a drawing board and connected with data-flows. Figure 13 shows an example of the outcome for the process.

Based on the information provided by the user, OVVL analyses a data-flow diagram for threats and software vulnerabilities, as shown in Figure 14.

Built data-flow diagrams can be analysed for threats following a modified STRIDE. Found threats can be filtered and prioritized. By setting a threats applicable status, a better overview over a systems potential threats can be gained. Additionally, defining this status

creates anonymised data for machine learning purposes, further improving the analysis in the future.

OVVL analyses the data-flow diagram also for searching software vulnerabilities. It queries existing vulnerability databases to identify Common Vulnerabilities and Exposures (CVE) [49] entries, i.e., known and reported real-world vulnerabilities (CVE is further described in Section 4.3.2). The analysis result includes a description of each vulnerability, the date the vulnerability was found and its impact score. Additionally, possible ways to mitigate these security risks are linked in each vulnerability. Data regarding software as Common Platform Enumeration (CPE)⁴ [50] and their corresponding CVE is provided by the NIST through the National Vulnerability Database (NVD) [51] [21].

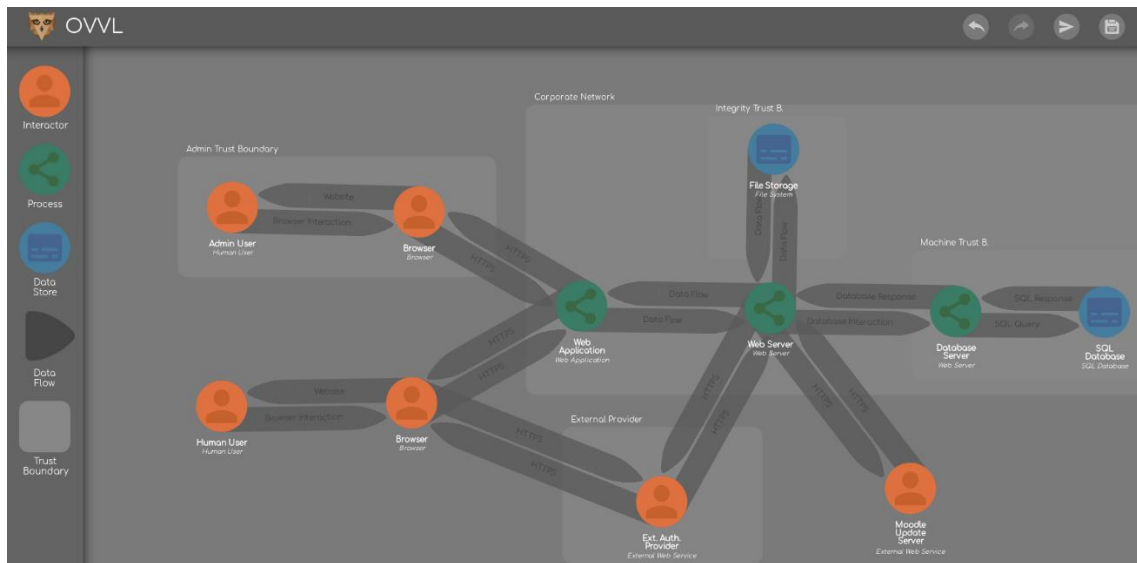


Figure 13 Example of DFD in OVVL [52]

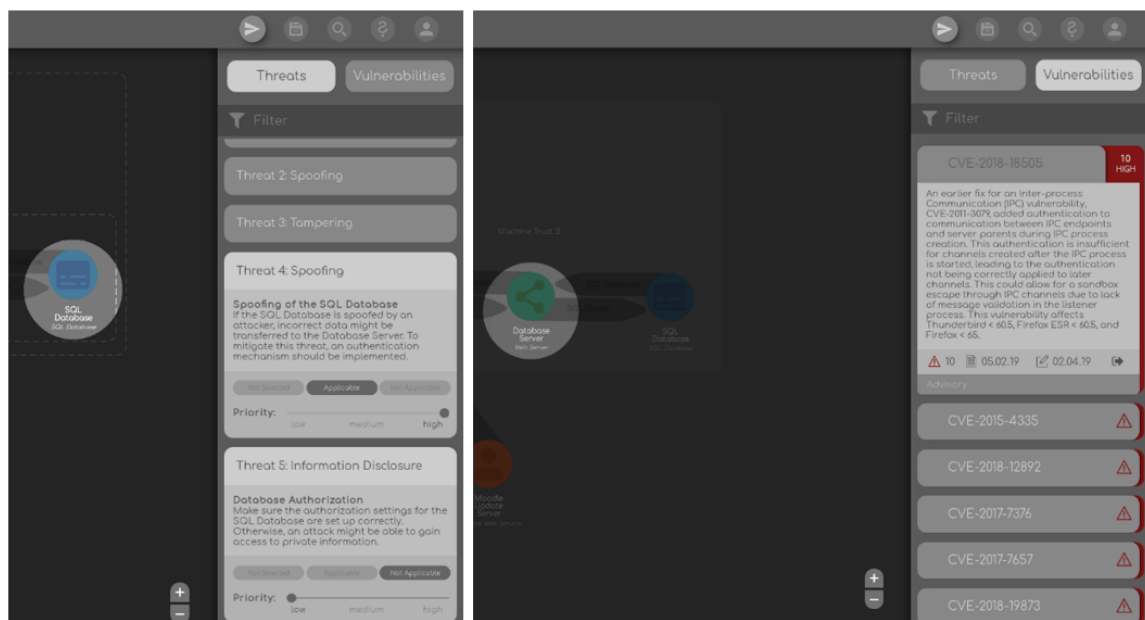


Figure 14 Threat (on the left) and Vulnerability (on the right) analysis within OVVL

⁴ CPE is a structured naming scheme for information technology systems, software, and packages.

2.1.2.3. Threat Dragon

Threat Dragon is an open-source threat modelling tool from OWASP. It comes as a web application or an installable desktop app [20]. The tool allows the users to create a model, load a sample model, or re-open an existing model from GitHub or from the local filesystem and modify it.

The model includes elements as *processes*, *data stores*, *actors*, *data flows* and *trust boundaries*. The elements (apart from boundaries) can be marked as out of scope, and the reason of this marking can be specified as well. An example of model is shown in Figure 15

Threat Dragon provides STRIDE per Element rules to generate the suggested threats for an element on the diagram. When elements of the model have open and unmitigated threats, they are highlighted in red. According to [20], the focus of Threat Dragon is on UX, on a powerful rule engine and on the alignment with other development lifecycle tools.

The tool allows to generate a summary report of the model, listing the diagrams, elements and threats; users can customize the report to show or hide out of scope model elements, mitigated threats or threat model diagrams.

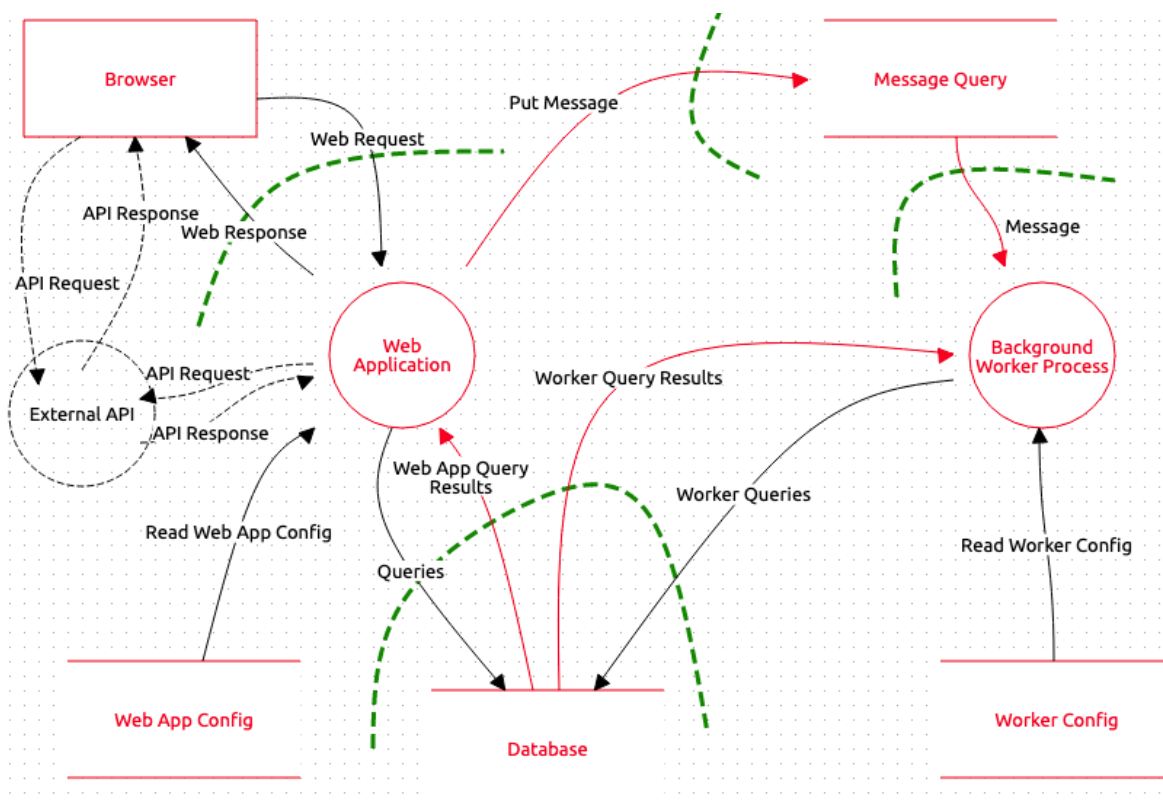


Figure 15 – Data Flow Diagram with Threat Dragon Tool [152]

2.1.2.4. Quantitative Threat Modelling Method

The Quantitative Threat Modelling Method applies a combination of Attack Trees, STRIDE, and Common Vulnerability Scoring System (CVSS)⁵. This method has been used in a case study for a railway communications network [37] [76].

⁵ A description of CVSS is provided in Section 2.2.2

First of all, the method requires the modelling of the attack trees for the system components, one tree for each STRIDE category [27]. Figure 16 shows an example of the first step for the tampering category [76].

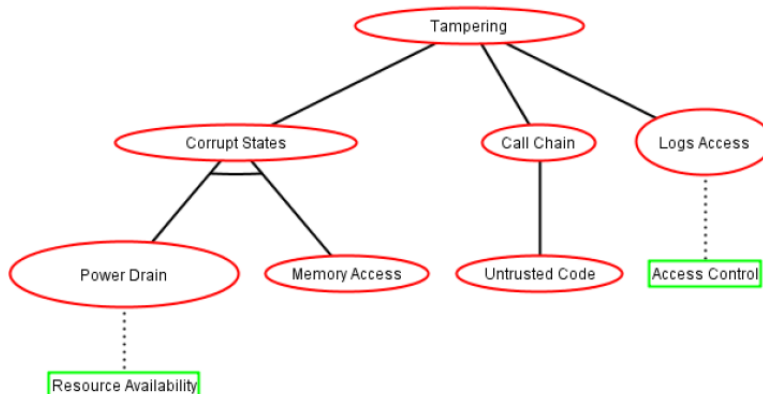


Figure 16 – Example of Attack Tree for tampering category of STRIDE [76]

Then, the QTMM applies CVSS to compute scores for the nodes of the tree, as depicted in Figure 17; the figure shows the risk of occurrence of a tampering attack as 0.78, which means that the CVSS score is high, being 7.8 out of 10. The scores are referring to the situation where mitigations are not in place, and after activating them, the overall risk value, as expected, is reduced [76].

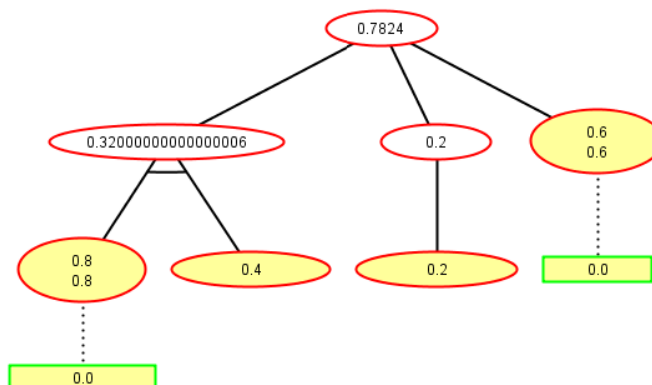


Figure 17 - Example of Tampering Attack Tree with CVSS-based scores and no mitigations [76]

2.1.3. VAST – Visual, Agile, Simple Threat Modelling Method

VAST (Visual, Agile, and Simple Threat) is a modelling method based on project management and agile programming principles, and is the basis for a commercial threat modelling tool called ThreatModeler [38], [13], which will be presented in Section 2.1.3.1.

It divides threat models and corresponding flow diagrams into two categories [38]:

- *Application models*: Process flow diagrams (PFD)⁶ are created that focus on a specific application and represent the architectural viewpoint;
- *Operational models*: end-to-end DFDs are created that incorporate application interactions.

The goal of VAST is to integrate, in a scalable way, threat modelling and risk management in the context of agile development programs. The underlying idea is that threat modelling

⁶ A process flow diagram is a flowchart that helps to describe the general flow of a business process. By extension, a network diagram describes the various components of IT network architecture.

is useful only if it includes the whole SDLC and incorporates the pillars as automation, integration and collaboration.

2.1.3.1. ThreatModeler

ThreatModeler is a threat modelling tool that promotes a collaborative threat modelling process across all SDLC stakeholders. The focus of the tool are web applications, and the underlying modelling methodology, as already described above, is VAST.

The tool uses a so-called Intelligent Threat Engine (ITE) to identify, classify and prioritize the threats in order to reduce the overall risk for the web application. It is synchronized with the threats, security requirements and vulnerabilities, from OWASP, CAPEC⁷ [54], and NVD [51] [13]. Figure 18 shows the interface of the tool during the diagram modelling stage, while Figure 19 shows the dashboard displaying the top threats, listed in descending priority.

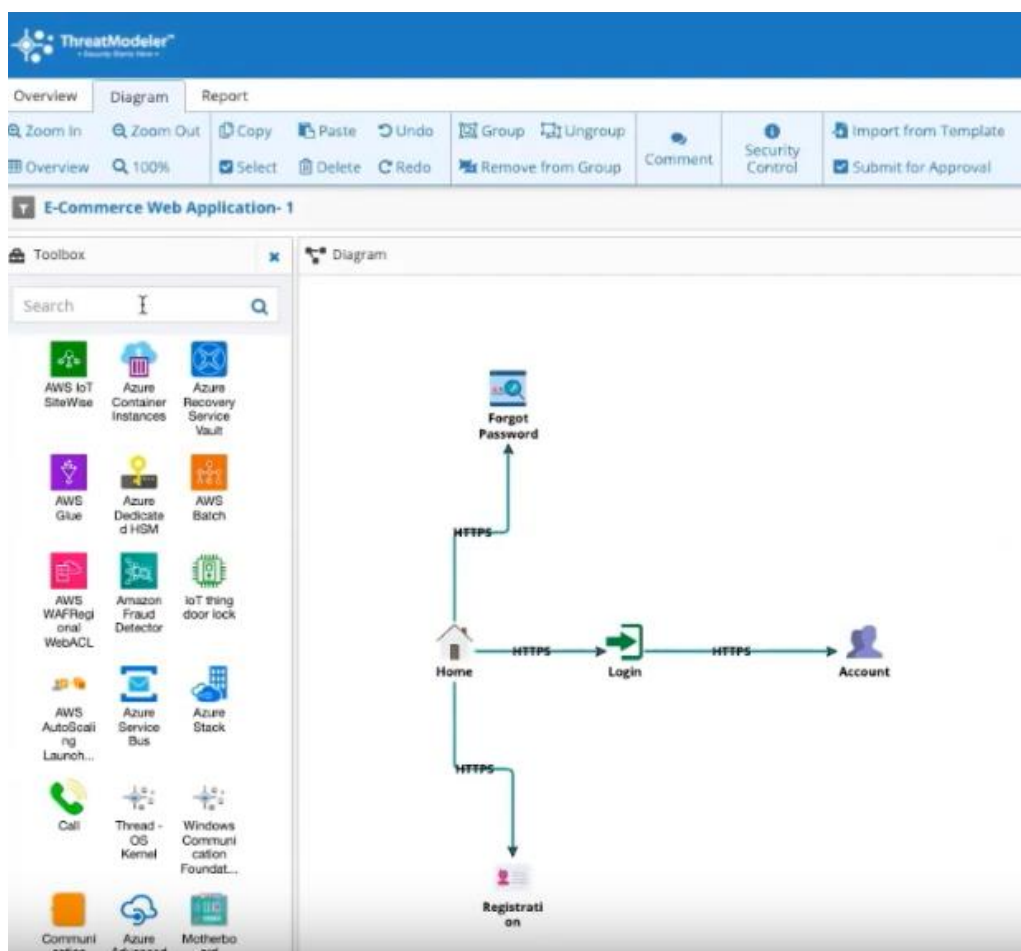


Figure 18 - Example of a Web Application Diagram in ThreatModeler [13]

One of the functionalities of the tool is allowing the user to understand threat information and to take threat mitigation decisions. A report is then generated, allowing user to consult information such as Executive Summary, Threats, Security Requirements and Test Cases.

⁷ CAPEC (Common Attack Pattern Enumeration and Classification) is also described Section 4.3.3

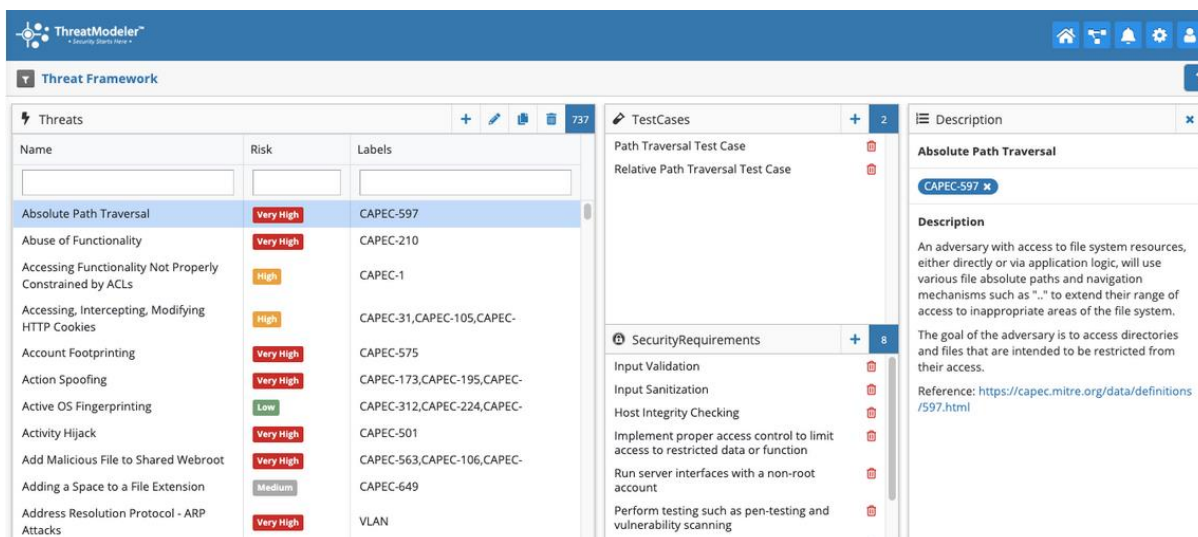


Figure 19 - Example of Dashboard [13]

2.1.4. LINDDUN

LINDDUN⁸ supports analysts in systematically eliciting and mitigating privacy threats in software architectures. As STRIDE, it is a mnemonic for the privacy threat categories which it supports:

- *Linkability*: being able to sufficiently distinguish whether 2 items of interest (IoI) are linked or not, even without knowing the actual identity of the subject of the linkable IoI.
Not being able to hide the link between two or more actions/identities/pieces of information;
- *Identifiability*: being able to sufficiently identify the subject within a set of subjects (i.e., the anonymity set). Not being able to hide the link between the identity and the IoI (an action or piece of information);
- *Non-repudiation*: having irrefutable evidence concerning the occurrence or non-occurrence of an event or action;
- *Detectability*: an attacker can sufficiently distinguish whether an IoI exists or not;
- *Disclosure of information*: exposing information to someone not authorized to see it;
- *Unawareness*: not understanding the consequences of sharing personal information in the past, present, or future;
- *Non-compliance*: not following the (data protection) legislation, the advertised policies or the existing user consents [34].

Its main strength is its combination of methodological guidance and privacy knowledge support.

Figure 20 shows the three steps of LINDDUN methodology [35]:

1. *Modelling*: A good understanding of the system is required in order to analyse its privacy. LINDDUN uses, similarly to STRIDE, a DFD as a model to capture the most relevant system knowledge for the privacy analysis;

⁸ Linkability, Identifiability, Nonrepudiation, Detectability, Disclosure of information, Unawareness, Noncompliance

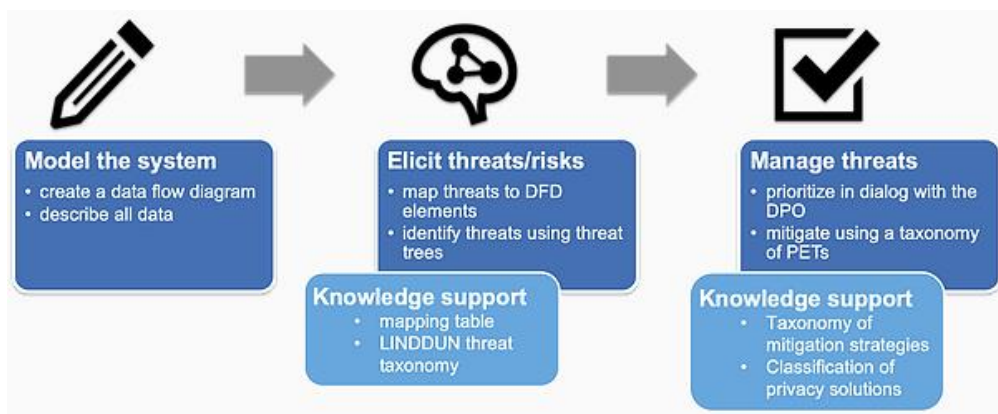


Figure 20 - LINDDUN framework [35]

2. *Elicitation*: Once the system is described, each DFD element should be systematically analysed for privacy threats. First a mapping table will be created to guide this process of systematic privacy threat elicitation;
3. *Management*: identified threats are tackled. So, prioritize threats, select suitable mitigation strategy and select privacy enhancing solution.

2.1.5. PASTA - Process for Attack Simulation and Threat Analysis

PASTA (Process for Attack Simulation and Threat Analysis) is a seven-step, risk-centric methodology. Each step is divided in multiple activities, described in Table 3 [102] [37].

Table 3 - Steps and Activities of PASTA Methodology (sources: [102][37])

Steps	Activities
1. Define Objectives	<ul style="list-style-type: none"> • Identify Business Objectives • Identify Security & Compliance Requirements • Business Impact Analysis
2. Define Technical Scope	<ul style="list-style-type: none"> • Capture the Boundaries of the Technical Environment • Capture Infrastructure Application Software Dependencies
3. Application Decomposition	<ul style="list-style-type: none"> • Identify Use Cases Define App. Entry Points & Trust Levels • Identify Actors Assets Services Roles Data Sources • Data Flow Diagramming (DFDs) Trust Boundaries
4. Threat Analysis	<ul style="list-style-type: none"> • Probabilistic Attack Scenarios Analysis • Regression Analysis on Security Events • Threat Intelligence Correlation & Analytics
5. Vulnerability & Weaknesses Analysis	<ul style="list-style-type: none"> • Queries of Existing Vulnerability Reports & Issues Tracking • Threat to Existing Vulnerability Mapping Using Threat Trees • Design Flaw Analysis Using Use & Abuse Cases • Scorings (CVSS/CWSS) Enumerations (CWE/CVE)
6. Attack Modelling	<ul style="list-style-type: none"> • Attack Surface Analysis • Attack Tree Development Attack Library Mgt. • Attack to Vulnerability & Exploit Analysis Using Attack Trees
7. Risk & Impact Analysis	<ul style="list-style-type: none"> • Qualify & Quantify Business Impact • Countermeasure Identification & Residual Risk Analysis • ID Risk Mitigation Strategies

The objective of this methodology is to identify the threats, enumerate them, and assign a score to each threat. Once the threat model is completed, security experts can develop a detailed analysis of the identified threats and can determine the appropriate

countermeasures that must be deployed to mitigate the risk. This methodology is intended to provide an attacker-centric view of the application and infrastructure from which defenders can develop an asset-centric mitigation strategy [1]. It uses a variety of design and elicitation tools in different stages [37].

2.1.6. TRIKE

Trike is an open-source threat modelling methodology and tool which has been developed as part of a framework for security analysis [77],[33]. It can be considered a fusion of two main models:

- *Requirement Model*: explains the security characteristics of an IT system and assigns acceptable levels of risk to each asset;
- *Implementations Model*: in this model, a DFD is created to illustrate the flow of data and the user performed actions within a system; threats are analysed to enumerate and assign a risk value.

In general, there are four steps composing TRIKE methodology (depicted in Figure 21).

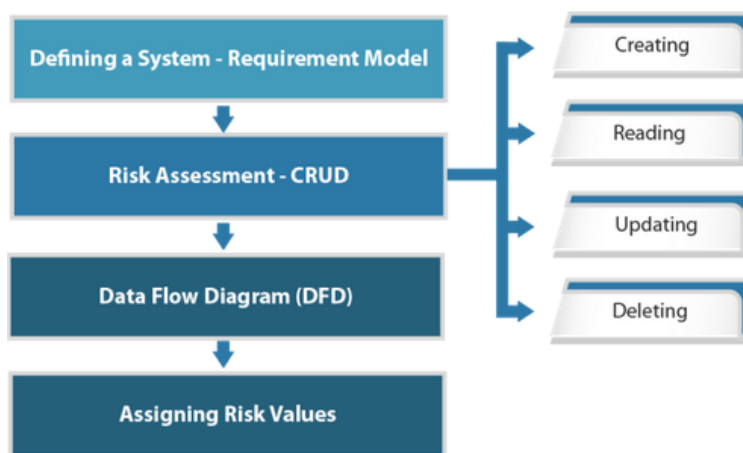


Figure 21 – Steps of TRIKE Methodology [55]

The first step is system definition, where an analyst models the requirements, identifies and lists the system's assets, actors, rules and planned actions. The next step creates a matrix named *actor-asset-action* in which the columns correspond to assets and the rows to actors [37]. The cells of the matrix are then divided into four parts, one for each CRUD⁹ action. The analyst specifies the values of the cells, selecting among: *allowed/disallowed action*, or *action with rules*. A rule tree (see Figure 22) is then generated and associated to each cell [37]. Once the requirement model is ready, DFDs are created. In this step, threats pertaining to *elevation of privilege* or *denial of service* are identified. Each threat is then a root node for an attack tree. Trike also enables the risk assessment for actions (CRUD-based threats) which are targeting the assets; the assessment leverages a rating, on a scale from 1 to 5, based on actions probability (see left side of Figure 23). Actors are also rated, from 1 to 5, according to their potential risks for the asset (where 1 is the highest), see right side of Figure 23 [37].

⁹ CRUD: creating, reading, updating, and deleting

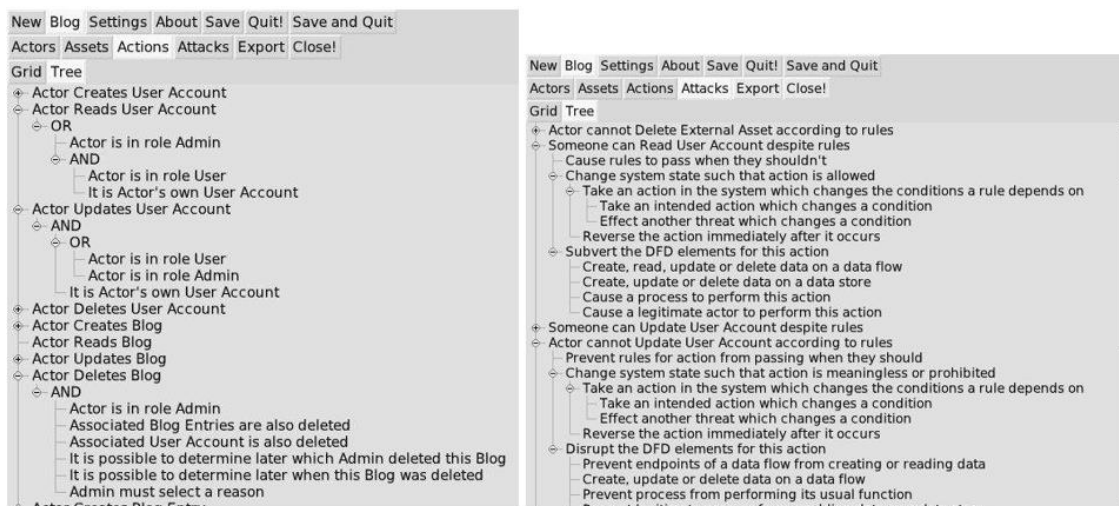


Figure 22 Trike Tool [78]: rules tree for intended actions (left); autogenerated attack tree (right).

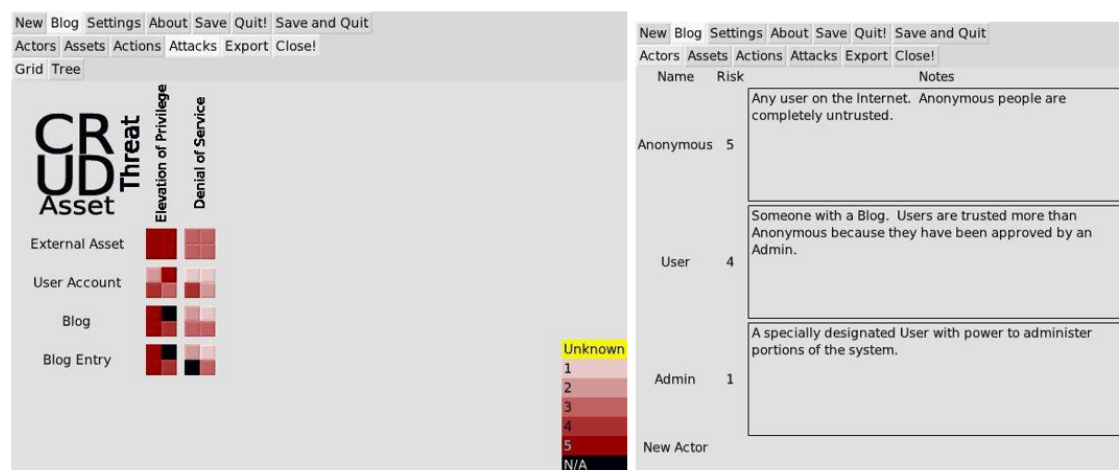


Figure 23 Trike Tool [99]: Risk grid/threat visualization (left); actors view (right).

2.1.7. OCTAVE - Operationally Critical Threat, Asset, and Vulnerability Evaluation

OCTAVE¹⁰ [65], is a heavyweight and self-directed method for strategic cybersecurity risk assessment and plan development.

Unlike most other risk assessment methods, the OCTAVE approach is driven by operational risk and security practices instead of technology [39] [56].

The method is based on eight processes that includes several other processes, but it usually preceded by an exploratory phase (known as *Phase Zero*) to determine the criteria that will be used during the application of the Octave method, and includes evaluating the extent of an impact in a specific area (health, productivity, financial, etc.).

Apart from phase zero, OCTAVE has three main phases, broken down into processes, which can be seen in Figure 24 [39][40].

¹⁰ Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)

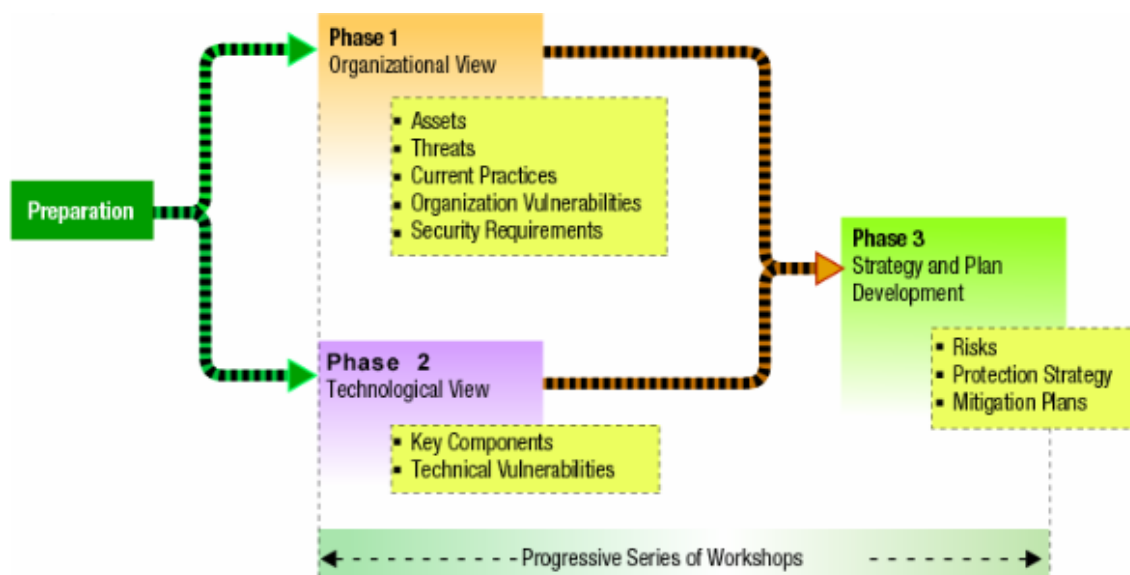


Figure 24 – Phases of OCTAVE methodology [56]

These phases can be synthesized as [39][40][56]:

- *Phase 1: Organizational View and Threats Profiling.* The two major functions of this phase are gathering information from across the organization and defining threat profiles for critical assets;
- *Phase 2: Identification of Vulnerabilities in the Infrastructure.* During this phase, the analysis team evaluates key components of systems supporting the critical assets for technological vulnerabilities;
- *Phase 3: Strategy and Plan Development.* The primary purpose of this phase is to evaluate risks to critical assets and develop an organizational protection strategy and risk mitigation plans.

All aspects of risk (assets, threats, vulnerabilities, and organizational impact) are factored into decision making, enabling an organization to match a practice-based protection strategy to its security risks.

Each process has certain activities that must be completed, and within each of these activities, different steps must be taken in order to achieve the desired outputs. The final result is that risk decisions can be based on is the threat profile of different assets. Each threat profile contains information based on which mitigation decisions can be taken.

However, OCTAVE does not produce a detailed quantitative analysis of security exposure and although the metrics are defined, the mapping with the impact intervals (low, medium and high) is open, making difficult the comparison even between devices evaluated with OCTAVE [65]. In addition, although OCTAVE focuses on speed, since for most businesses time is money, its 18 volumes make it large and complex, to understand with many worksheets and practices to implement. In this sense, OCTAVE has another variant, OCTAVE-S [66] with fewer processes, but still too complex.

2.1.8. ADVISE - ADversary View Security Evaluation

The ADversary View Security Evaluation (ADVISE) method provides quantitative security metrics to system architects. These metrics are used to make informed trade-off decisions involving system security. System architects can use ADVISE to compare the

security strength of system architecture variants and analyse the threats posed by different adversaries [26].

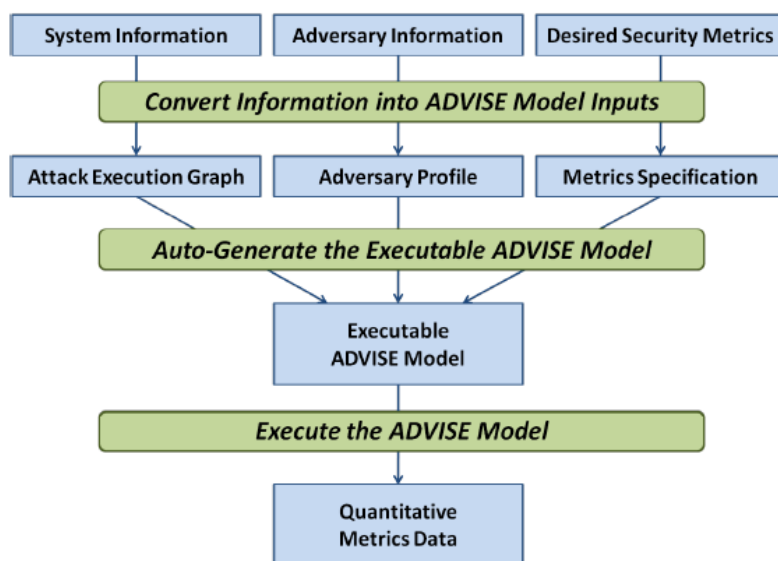


Figure 25 - The ADversary View Security Evaluation (ADVISE) method [26]

The ADVISE method, shown in Figure 25, is implemented in a tool (Mobius [74], briefly described in Section 2.1.8.1) that facilitates user input of system and adversary data and automatically generates executable models. Attacks against a system can be regarded as sequences of smaller attack steps.

In Figure 26, there is an ADVISE model with these attack steps that are precisely defined and organized into an Attack Execution Graph (AEG).

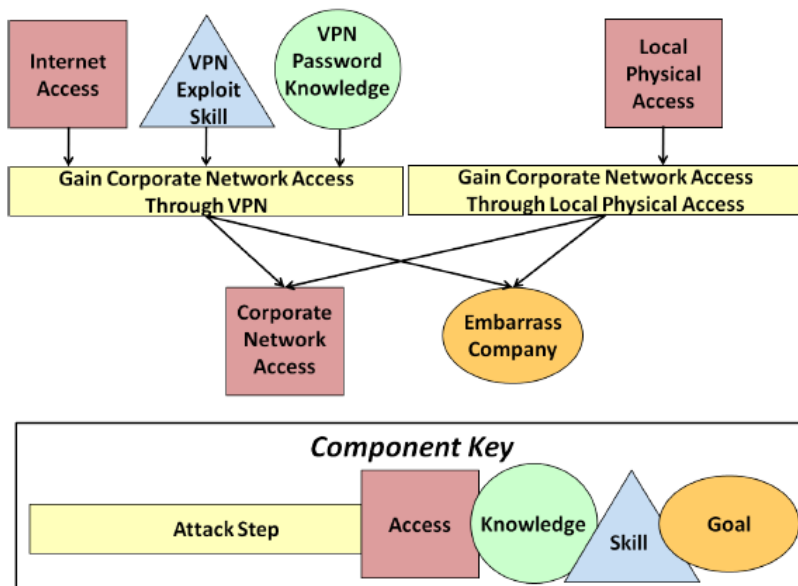


Figure 26 - An Attack Execution Graph (AEG) represents possible attacks [26]

The attack execution graph also contains timing, cost, probabilistic outcomes, and other information about each attack step. This extra information makes it possible to analyse ADVISE models using discrete-event simulation [26].

The user can define the adversary profile, which captures a particular adversary's *attack preferences*, *attack goals*, and *attack skills*; an example is shown in Figure 28.

The ADVISE model execution algorithm uses the adversary profile and the attack steps in the attack execution graph to mimic how the adversary is likely to attack the system. The adversary selects the best next attack step by evaluating the attractiveness of several attack steps, considering *cost*, *payoff*, and the *probability of detection* [26].

This methodology is very well defined and is based on a solid scientific background. However, the threat model must be known a priori by the user, it is not given by the methodology/tool. Another minor issue is that the results of simulations are only in .csv and .txt formats, and the user has to interpret and represent them by themselves.

2.1.8.1. Mobius Framework and ADVISE Implementation

The Mobius discrete-event modelling environment is a framework that supports multiple modelling formalisms and multiple solution techniques and has been often used in system performance and dependability modelling [79].

The ADVISE atomic model formalism implementation in the Mobius framework provides a graphical front-end for creating and modifying ADVISE models (as shown in Figure 27). The model definitions are stored in a textual, XML-based format. Mobius then uses code from the ADVISE implementation to generate C++ code that compiles and links with Mobius framework libraries, creating an executable model [79].

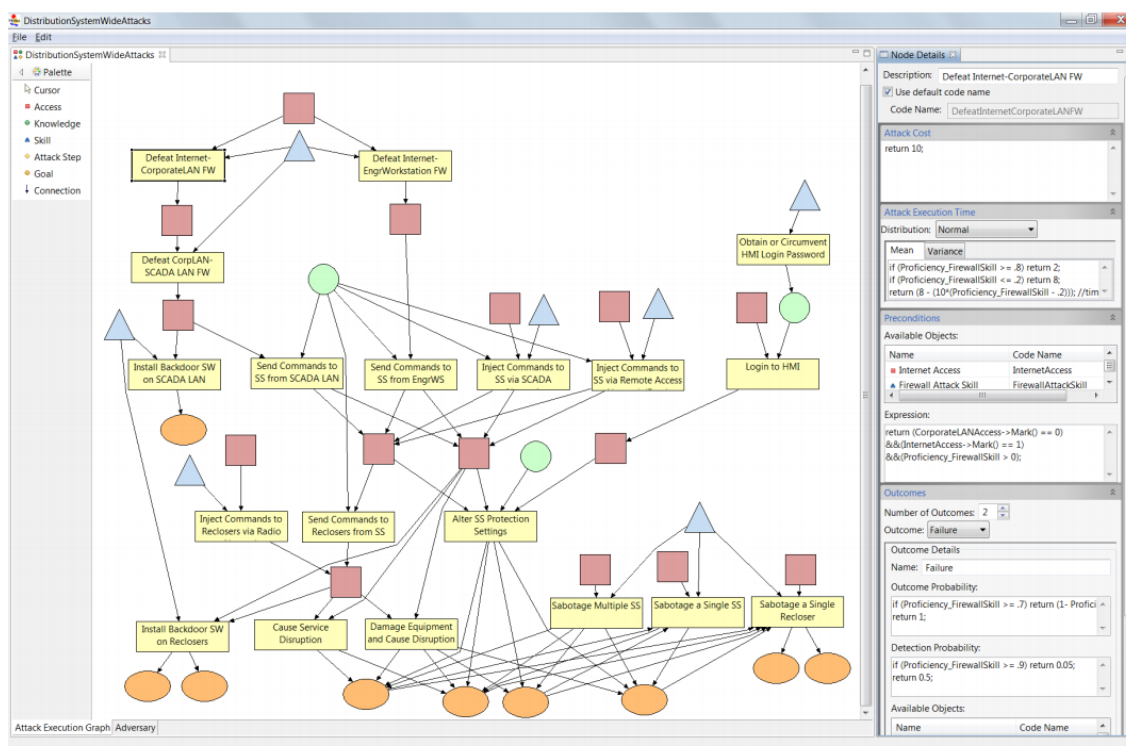


Figure 27 An example of the ADVISE attack execution graph editor page [79]

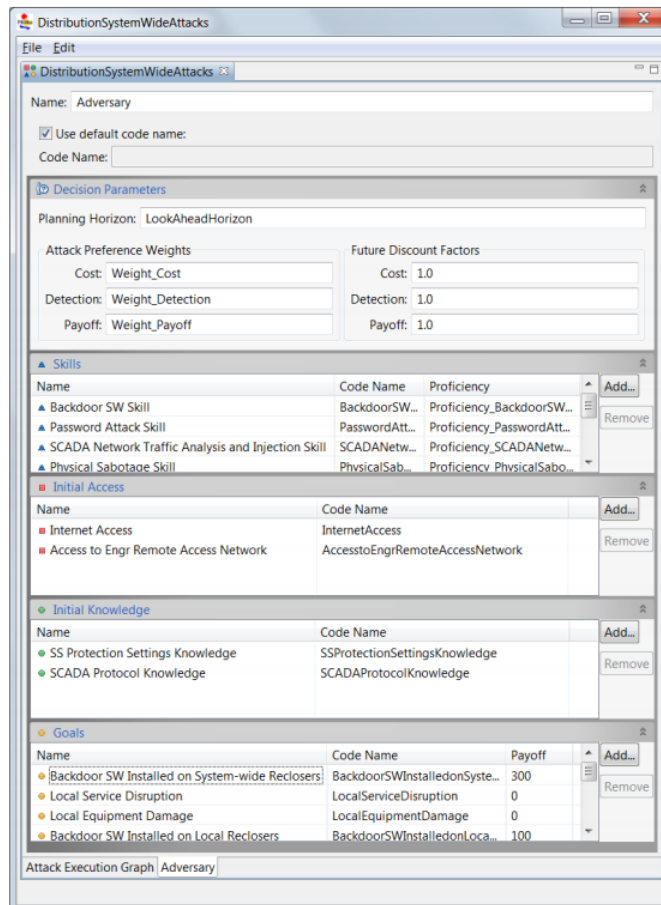


Figure 28 An example of the ADVISE adversary profile editor page in Mobius [79].

2.1.9. Security Cards

Security Cards is an informal method, a kind of brainstorming technique, for the identification of complex and unusual attacks. A security analyst, by using a set of 42 cards, is helped in answering some questions regarding the attack, regarding [32]:

- the adversary;
- the reason why the system can potentially be attacked;
- the assets of interest;
- the way of implementing an attack.

The cards are divided in categories, or dimensions (as shown in Figure 29) [37][41]:

- *Human Impact*: human impact points to the myriad of ways in which human beings can be affected in their lives, from intimate relationships and emotional experience to privacy violations with personal data to widespread societal impacts at the level of the economy, government, and social structure;
- *Adversary's Motivations*: emphasizes the variety of reasons an individual or group might wish to attack a system, from ideological reasons focused on religion, politics, or diplomacy to more self-oriented motivations such as convenience or self-promotion;
- *Adversary Resources*: presents an array of different assets that might be at an adversary's disposal, from hardware and software tools to the ability to influence the actions of groups of people, or access to technical or social expertise;

- *Adversary's Methods*: explores high-level ways that an adversary might approach attacking a system, from the familiar technological attack to manipulating or coercing people, covering up evidence, or leveraging logistical and bureaucratic processes.

In Figure 29 all the options covered in each of the four dimensions are shown.

<p>Human Impact</p> <ul style="list-style-type: none"> • the biosphere • emotional well-being • financial well-being • personal data • physical well-being • relationships • societal well-being • unusual impacts 	<p>Adversary's Motivations</p> <ul style="list-style-type: none"> • access or convenience • curiosity or boredom • desire or obsession • diplomacy or warfare • malice or revenge • money • politics • protection • religion • self-promotion • world view • unusual motivations
<p>Adversary's Resources</p> <ul style="list-style-type: none"> • expertise • a future world • impunity • inside capabilities • inside knowledge • money • power and influence • time • tools • unusual resources 	<p>Adversary's Methods</p> <ul style="list-style-type: none"> • attack cover-up • indirect attack • manipulation or coercion • multi-phase attack • physical attack • processes • technological attack • unusual methods

Figure 29 - Security Card Dimensions [57]

2.1.10.PnG - Persona non Grata

The *Persona non Grata* (PnG) is a method based on skills and motivations of the adversaries, and in particular of the insider attackers. It helps the security expert to analyse the system from the perspective of an insider attacker, and proposes characters representing possible users aiming at maliciously use the system for their purpose. This method can be particularly useful during early prototyping, when a security expert can analyse possible insiders of the system and their characteristics, such as skill, motivation, and goal [37].

Modelling PnGs can therefore help to think about the ways in which a system might be vulnerable to abuse and use this information to specify appropriate mitigating requirements. The PnG approach makes threat modelling more tractable by asking users to focus on attackers, their motivations, and abilities [31].

The theory behind this approach is that if engineers can understand what capabilities an attacker may have and what types of mechanisms, they may use to compromise a system, the engineers will gain a better understanding of targets or weaknesses within their own systems and the degree to which they can be compromised.

PnG is suitable for the agile approach, which uses *personas* to define archetypical users of a system [80]. In fact, each PnG includes an image of the persona, his or her name, a description, the assumed role and a moniker, including a set of relevant goals and skills, and a set of misuse cases that describe specific ways in which the PnG intends to attack the system, as in the example of Figure 30 .

From this, it is possible to construct a threat model that includes the actor (i.e., the PnG) and the attack mechanism and target specified in the misuse cases [32].

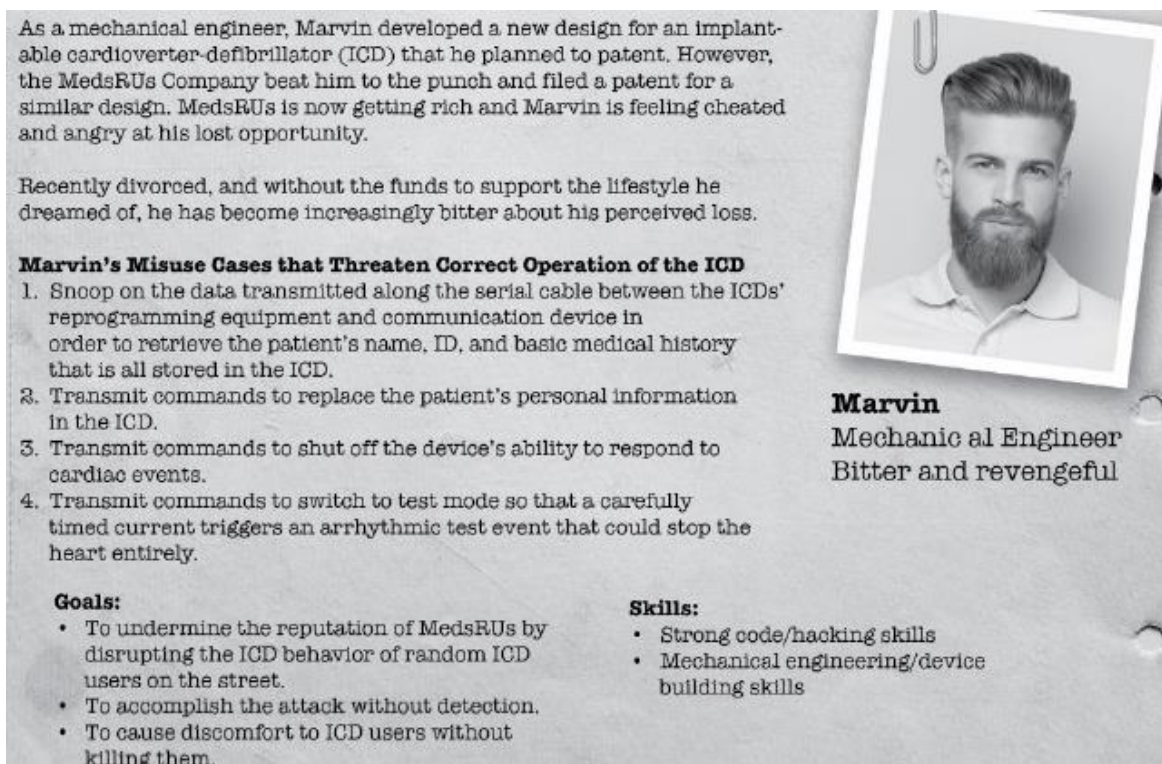


Figure 30 - Example of Persona non Grata [32]

2.1.11.hTMM - hybrid Threat Modelling Method

The hTMM¹¹ is a method which combines the afore-mentioned STRIDE (Section 2.1.2), Security Cards (presented in Section 2.1.9), and PnG (Section 2.1.10). It aims at excluding false positives, considering all the possible threats, giving cost-effective and repeatable results, not depending on the modeller.

The method is composed of five steps [32] [37]:

1. System identification: it executes Steps 1-3 of SQUARE¹² [58] or a similar security requirements method, as:
 - Agree on definitions;
 - Identify a business goal for the system, assets and security goals;
 - Gather as many artifacts as feasible;
2. Application of Security Cards:
 - Distribute the Security Cards 2.1.9 to participants either in advance or at the start of the activity;
 - Have the participants look over the cards along all four dimensions: Human Impact, Adversary's Motivations, Adversary's Resources, and Adversary's Methods;
 - Use the cards to support a brainstorming session and consider each dimension independently and sort the cards within that dimension in order of how relevant and risky it is for the system overall;
3. Removal of PnGs with low likelihood:
 - Itemize their misuse cases;
 - This expands on how the adversary attacks the system;

¹¹ hybrid Threat Modelling Method

¹² Not to be confused with ISO 25000 "Systems and Software Quality Requirements and Evaluation" SquaRE standards

- The misuse cases provide the supporting detailed information on how the attack takes place;
- 4. Tool-supported Result Summarization:
 - Actor (PnG): Who or what instigates the attack?
 - Purpose: What is the actor's goal or intent?
 - Target: What asset is the target?
 - Action: What action does the actor perform or attempt to perform? Here you should consider both the resources and the skills of the actor;
 - Result of the action: What happens as a result of the action? What assets are compromised? What goal has the actor achieved?
 - Impact: What is the severity of the result (high, medium, or low);
 - Threat type: e.g., denial of service, spoofing.
- 5. Formal risk assessment: using these results, and the additional steps of a security requirements method such as SQUARE [58].

2.1.12. CORAS

CORAS is a method for conducting security risk analysis which provides a customized language for threat and risk modelling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis [22].

The CORAS methodology integrates aspects from partly complementary risk analysis techniques, like HAZOP (see Section 2.1.13), FMEA, and FTA, with state-of-the-art system modelling methodology based on UML 2.0. A graphical UML-based language has been developed to support documentation and communication of security analysis results [23].

CORAS is a model-driven approach to risk analysis that follows the process defined by the ISO 31000 risk management standard. The approach consists of three tightly integrated artifacts, namely the CORAS method, the CORAS language and the CORAS tool [24].

The eight steps of CORAS method for a security risk analysis are:

1. Preparation for the analysis
2. Customer presentation of the target
3. Refining the target description using asset diagrams
4. Approval of the target description
5. Risk identification using threat diagrams
6. Risk estimation using threat diagrams
7. Risk evaluation using risk diagrams
8. Risk treatment using treatment diagrams

2.1.12.1. CORAS Tool

The CORAS method provides a computerized tool designed to support documenting, maintaining and reporting analysis results through risk modelling [22].

In Figure 31 an example of risk model obtained with CORAS tool and the legend for the typical model elements.

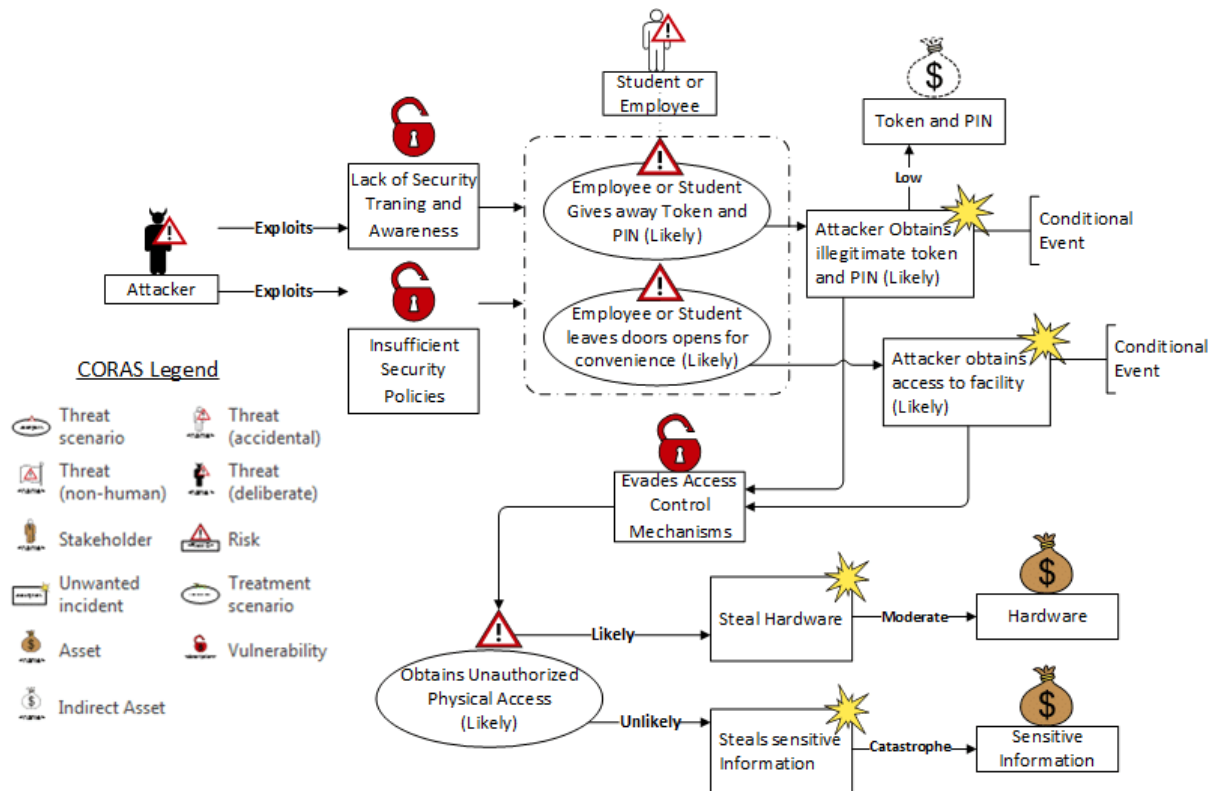


Figure 31 – Example of Risk modelled with CORAS, from [67]

2.1.13. HAZOP – HAZard and OPerability Study

HAZard and OPerability study (HAZOP) [81] is a structured and systematic technique for system examination and risk management. The HAZOP technique is qualitative and aims to stimulate the imagination of participants to identify potential hazards and operability problems.

It is based on a theory which assumes that risk events are caused by deviations from design or operating intentions. Identification of such deviations is facilitated by using sets of guide words as a systematic list of deviation perspectives. HAZOP guide words are key supporting elements in the execution of a HAZOP analysis. Figure 33 shows instead a sample worksheet of an HAZOP study for an automatic train protection system, where in the fourth column we can see the selected guidewords. Guidewords according to IEC Standard 61882 (Hazard and operability studies - Application guide), are presented in Table 4.

Table 4 – Generic HAZOP Guide Words

Guide Word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN/ INSTEAD	Complete substitution
EARLY	Related to the clock time
LATE	Related to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

As represented in Figure 32, the HAZOP analysis process is composed of four phases [25]. A key phase is the *Examination*, where the system is divided into *parts* or elements and each of them is analysed with the help of the guidewords.

Considering the process, the similarities with regard to a risk assessment may be noticed: this point will be addressed in the following of the document, especially in Sections 3 and 4.

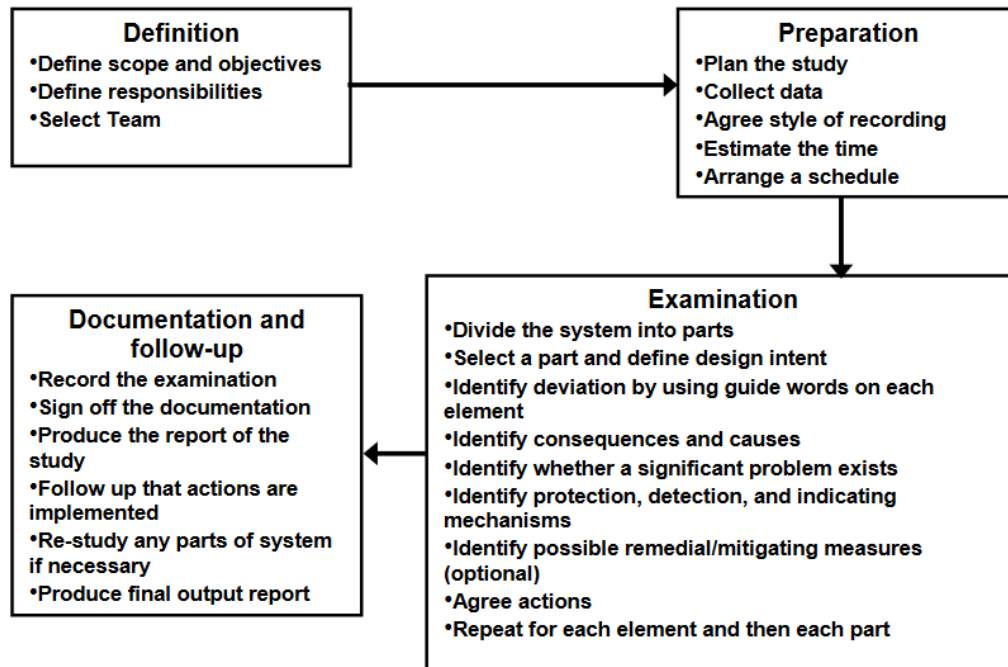


Figure 32 - HAZOP Analysis Process [25]

STUDY TITLE: AUTOMATIC TRAIN PROTECTION SYSTEM								SHEET: 1 of 2		
REFERENCE DRAWING No.: ATP BLOCK DIAGRAM				REVISION No.: 1				DATE:		
TEAM COMPOSITION: DJ, JB, BA								MEETING DATE:		
PART CONSIDERED:				INPUT FROM TRACKSIDE EQUIPMENT						
DESIGN INTENT:				TO PROVIDE SIGNAL TO PES VIA ANTENNAE GIVING INFORMATION ON SAFE SPEEDS AND STOPPING POINTS						
No.	Element	Characteristic	Guide word	Deviation	Possible causes	Consequences	Safeguards	Comments	Actions required	Action allocated to
1	Input signal	Amplitude	NO	No signal detected	Transmitter failure	Considered in separate study of trackside equipment			Review output from trackside equipment study	DJ
2	Input signal	Amplitude	MORE	Greater than design amplitude	Transmitter mounted too close to rail	May damage equipment	Checks to be carried out during installation		Add check to installation procedure	DJ
3	Input signal	Amplitude	LESS	Smaller than design amplitude	Transmitter mounted too far from rail	Signal may be missed	As above		Add check to installation procedure	DJ
4	Input signal	Frequency	OTHER THAN	Different frequency detected	Pick up of a signal from adjacent track	Incorrect value passed to processor	Currently none		Check if action is needed to protect against this	DJ
5	Antennae	Position	OTHER THAN	Antennae is in other than the correct location	Failure of mountings	Could hit track and be destroyed	Cable should provide secondary support		Ensure that cable will keep antennae clear of track	JB
6	Antennae	Voltage	MORE	Greater voltage than expected	Antennae short to live rail	Antennae and other equipment become electrically live			Check if there is any protection against this occurring	DJ

Figure 33 Sample HAZOP Worksheet [82]

2.1.14. Other Tools

2.1.14.1. IriusRisk

IriusRisk is a threat modelling tool which includes templates and risk pattern-based functionalities that allows the user to quickly create a model of a system or software architecture [14]. An example of architecture is given in Figure 34.

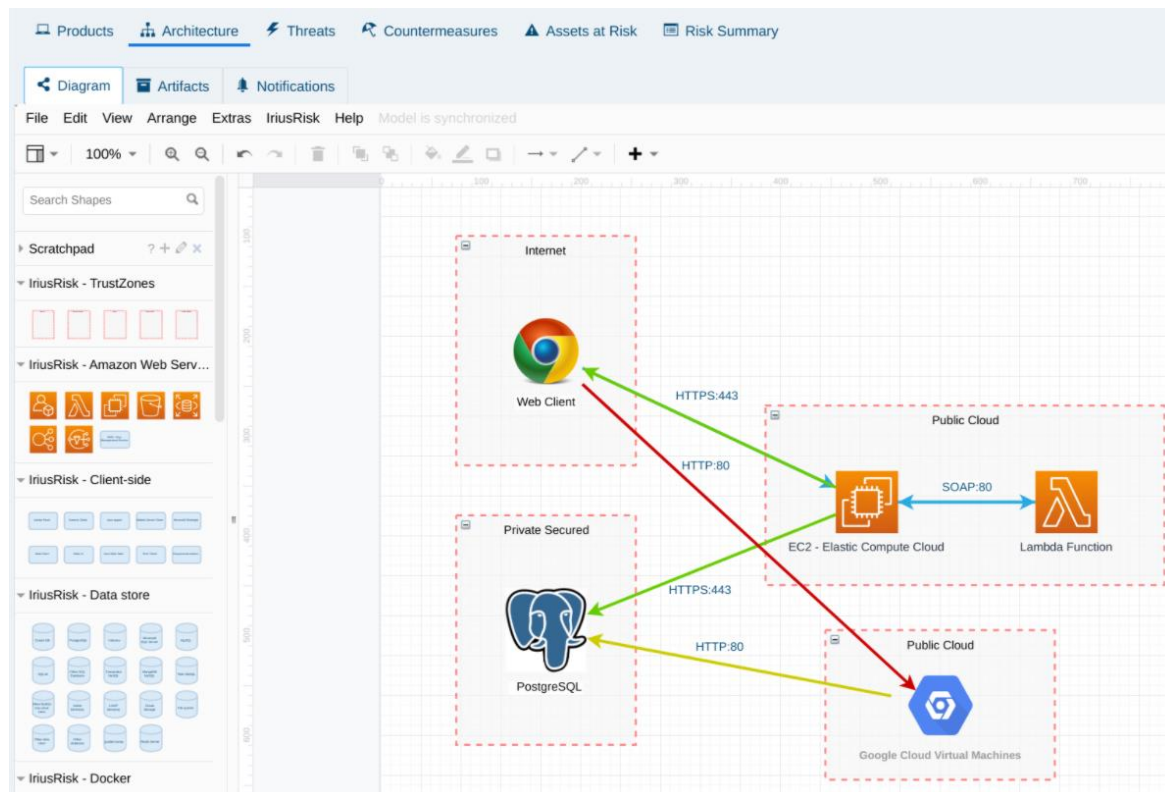


Figure 34 - Example of Architecture [14]

As the architecture and components are selected, the rules engine calculates a threats list. Each threat is linked to potential weaknesses and recommended countermeasures from an extensive application risk database, which includes the CAPEC. In fact, it provides threats or potential weaknesses (CWE¹³) and applicable countermeasures [14], as shown in Figure 35.

Weaknesses are regarded as potentially present, until their presence or absence has been verified through security testing. Tests can be automatically imported from external test sources like JUnit, JBehave, Cucumber, OWASP ZAP and Irius Risk's own BDD-Security framework. Security test results (e.g., negative testing, such as vulnerability assessments and penetration tests, or positive security control testing, such as code reviews and audit) can be recorded against the listed Weaknesses, or Countermeasures. Confirmed weaknesses are highlighted as vulnerabilities [14].

The user can then make an informed decision about the appropriate risk response: *Mitigate, Avoid or Accept*. For example, a countermeasure can be applied to mitigate the risk, or a risk can be accepted, and the risk decision justified [14].

Security testing is supported both from a control and a vulnerability perspective.

¹³ Common Weakness Enumeration - A Community-Developed List of Software & Hardware Weakness Types (CWE), more details are in section 4.3.1 and in [86].



Figure 35 Threats view in Irius Risk

2.1.14.2. securiCAD

The securiCAD tool allows the users to define models of ICT infrastructures, composed by objects and connections between them, and then enable cyber security analysis with by simulating potential attacks. The defined objects can represent both real items and conceptual/representative items (networks, routers, hosts, user accounts, services and so on) [17].

The tool securiCAD allows to the user to simulate potential attack paths from the attacker to all the assets in the modelled infrastructure by simply clicking the Simulate button. The tool highlights the assets risk level with different colours. For each asset, the users can click on the Critical Path icons, and the tool shows the most likely attack path that securiCAD has found for that asset [59].

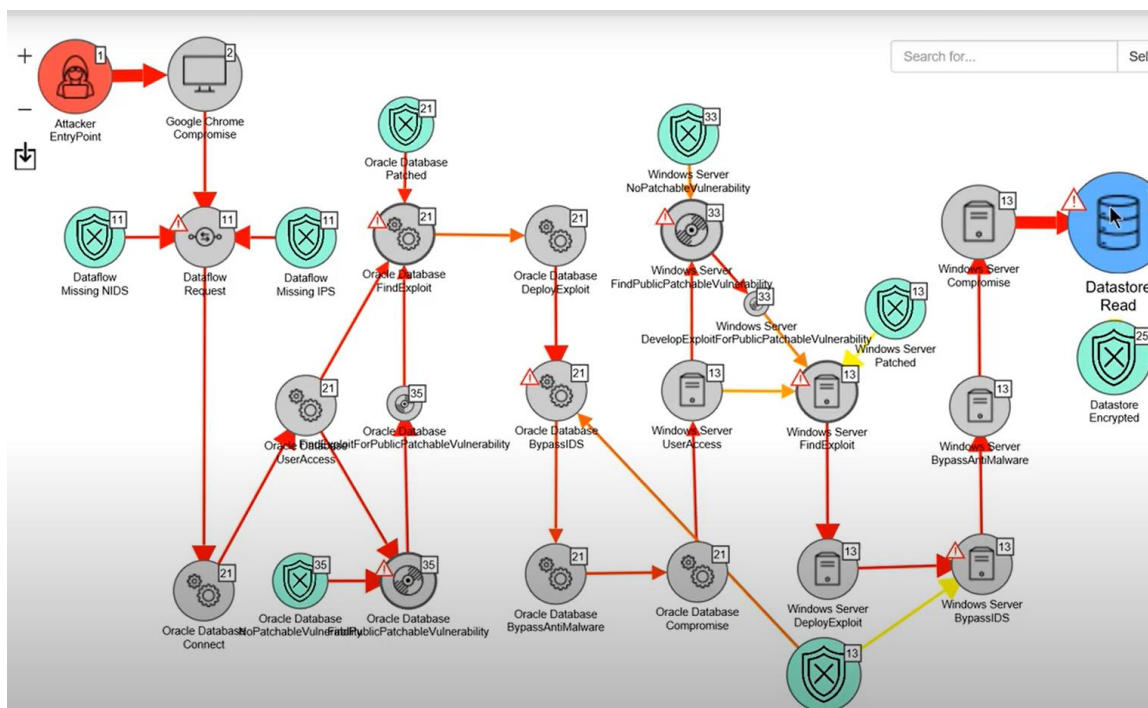


Figure 36 - Example of Critical Path in securiCAD [59]

Simulations can be run both in an online cloud service and locally in the securiCAD software. As soon as the simulation is ready, the results will be shown in two ways; the frames of the objects in the model will be given colours based on the attack success rate, as in the example of Figure 36, and the results will also be presented in an online report [17]. An overview of these results is given in Figure 37.



Figure 37 – Overview of Results given by securiCAD [59]

2.1.14.3. PyTM

PyTM is a Pythonic framework and Python-based library for threat modelling that allows to the users to model a system in Python using the elements and properties described in the pytm framework. It allows the creation of system models as Python objects, with properties as annotations [15]. PyTM can generate DFD, Sequence Diagram and threats to the modelled system, as shown in Figure 38. It allows users to export the report, the images resulting from the generation of the DFD and the Sequence Diagram, which are also given in output as Dot and PlantUML.

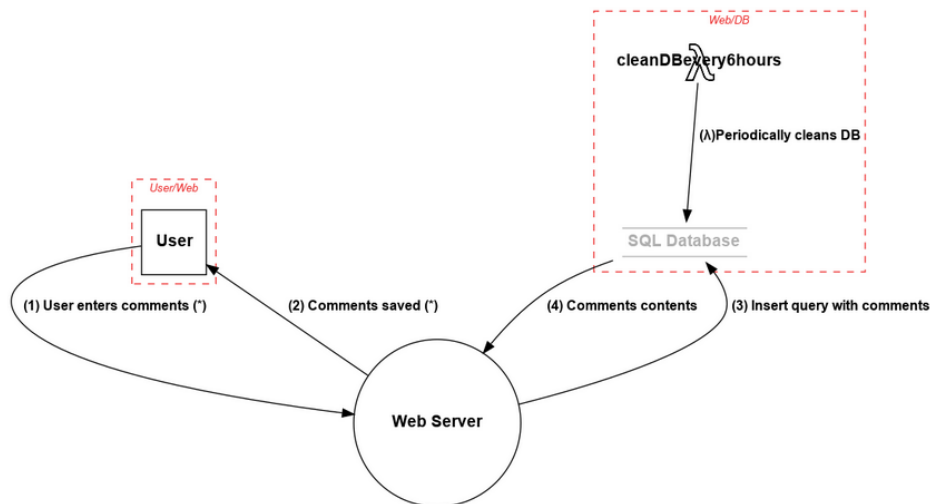


Figure 38 - Example of Diagram realized with PyTM [15]

PyTM uses CAPEC to inform the rule set with descriptions, mitigation and other references. In addition, CAPEC entries are translated as rules and can generate properties for description objects as needed [16].

There is a Threats database that can be set in TM.threatsFile and it is possible to generate a final report, in which diagrams can be included.

2.1.14.4. SD Elements

SD Elements is a tool and software security requirement management platform that allows automation of the security process, such as threat modelling, risk assessments, and implementation of secure coding and deployment guidelines [19]. SD Elements provides step-by-step test cases to help non-security experts test relevant cases to their application [18].

The four steps of SD Security Process are:

1. **Information Gathering:** initiates the secure development process by collecting information about the applications through an adaptive survey;
2. **Expert Assessment:** the tool automatically identifies risks or potential weaknesses of the user's applications. It then classifies the overall risk according to user's pre-defined security and compliance policies, as shown in Figure 39;

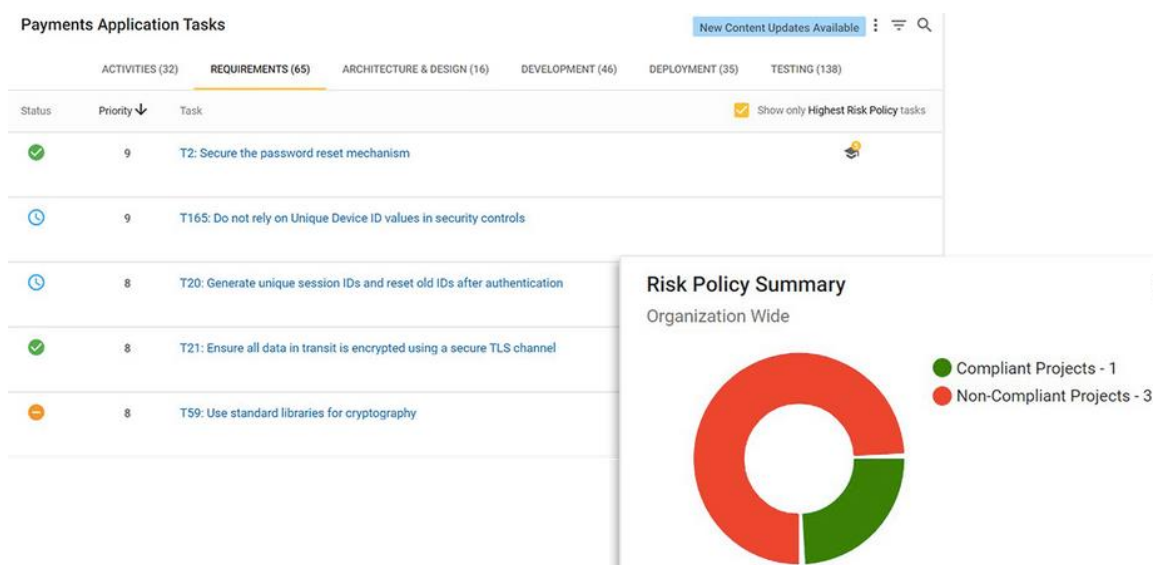


Figure 39 - SD Expert Assessment [19]

3. **Recommendations:** it translates requirements into recommendations and controls.
In the next step, the tool maps recommendations and controls to the risks identified for the applications. These controls can be seen directly to DevOps teams on their issue trackers such as Jira, VersionOne, or Azure Boards;
4. **Validation and Reports:** automatically tracks the status of security activities through robust integrations with security testing tools such as Veracode, Checkmarx, or Fortify; allowing security experts to focus on critical issues. It can instantly generate reports to view identified risks and their current mitigation status.

2.2. Risk Rating and Security Scoring Systems

This Section presents some of the main risk rating and security scoring systems which emerged during the research of threat modelling methodologies and tools. They are presented in a separate section considering that risk rating and security scoring are somehow different but still central and tightly related activity with respect to threat modelling.

2.2.1. CWSS - Common Weakness Scoring System

The Common Weakness Scoring System (CWSS) provides a mechanism for prioritizing software weaknesses and assign a numerical risk to them. To do so, CWSS combines three groups of metrics that are used to calculate the risk: Base Finding, Attack Surface, and Environmental, (shown in Figure 40 [61]).

Each factor in the metric groups is assigned a value, which is converted to its associate weight. The metrics of each group is calculated and combined with the other groups (multiplication) in order to obtain a complete risk measure, which ranges between 0 and 100. The Base Finding sub score is between 0 and 100, whereas the other ones can range between 0 and 1.

However, metrics such as likelihood are difficult to compute [68].

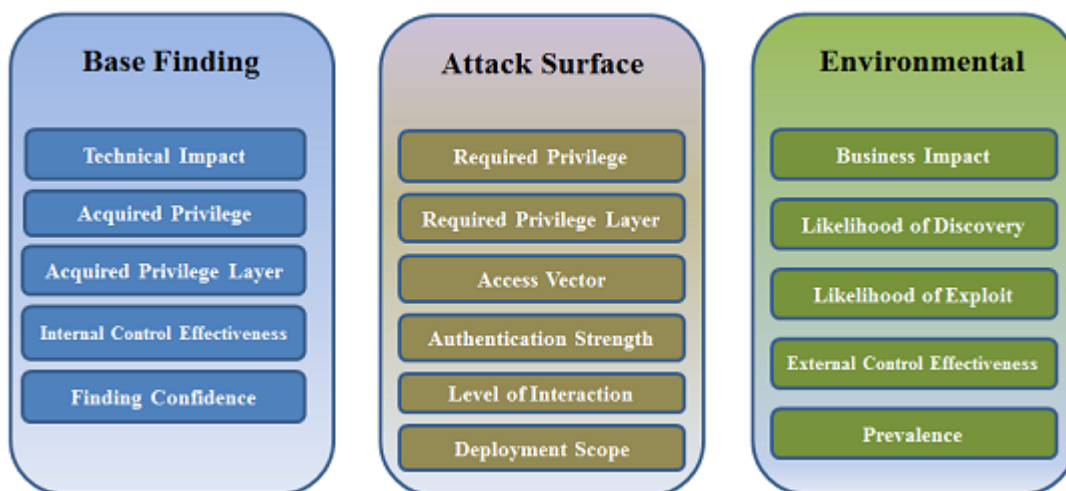


Figure 40 - CWSS Metric Groups [61]

The *Base Finding*: is focused on the inherent risk of the weakness, the confidence in the accuracy of the finding, and strength of controls. It consists of the following factors:

- *Technical Impact (TI)*: is the potential result that can be produced by the weakness, if it can be successfully reached and exploited. This is expressed in terms of confidentiality, integrity, and availability (CIA);
- *Acquired Privilege (AP)*: identifies the type of privileges that an attacker can have when he can successfully exploit the weakness;
- *Acquired Privilege Layer (AL)*: identifies the operational layer where the attacker gains privileges, if he can successfully exploit the weakness;
- *Internal Control Effectiveness (IC)*: measures the ability of the control that it makes the weakness unable to be exploited by an attacker. The Internal Control is a control, protection mechanism, or mitigation that has been explicitly built into the software;
- *Finding Confidence (FC)*: is the confidence that the reported issue, as weakness and it can be triggered or utilized by an attacker.

If the set of values proposed for the TI metric, is not precise enough, CWSS users can use their own quantified methods to derive a sub score. One of the methods uses the Common Weakness Risk Analysis Framework (CWRAF)¹⁴ to define a vignette and a Technical Impact Scorecard. Here, vignette-specific Importance ratings are used to calculate the Impact weight. CWRAF and CWSS allow users to rank classes of weaknesses independent of any particular software package, in order to prioritize them relative to each other (e.g., "buffer overflows are higher priority than memory leaks"). This approach, sometimes referred to as a "Top-N list," is used by the CWE/SANS Top 25 and OWASP Top Ten [69].

The *Attack Surface*: includes factors representing the barriers that an attacker must exceed to exploit the weakness. It consists of the following factors:

- *Required Privilege (RP)*: identifies the type of privileges that an attacker must have to reach the code/functionality that contains the weakness;
- *Required Privilege Layer (RL)*: identifies the operational layer where the attacker must have privileges to try to attack the weakness;

¹⁴ <https://cwe.mitre.org/cwraf/>

- *Access Vector (AV)*: identifies the channel thanks to which an attacker must communicate to reach the code/functionality that contains the weakness (the value it is similar to the ones in CVSS, but here there is a difference between physical and local access;
- *Authentication Strength (AS)*: covers the strength of the authentication routine that protects the code/functionality that contains the weakness;
- *Level of Interaction (IN)*: includes the actions that are required by the human victim(s) to allow a successful attack;
- *Deployment Scope (SC)*: identifies if the weakness is present in all deployable instances of the software, or if it is limited to a subset of platforms and/or configurations.

The *Environmental*: groups characteristics of the weakness that are specific to a particular environment or operational context. It consists of the following factors:

- *Business Impact (BI)*: describes the potential impact to the business or mission if the weakness can be successfully exploited;
- *Likelihood of Discovery (DI)*: represents the likelihood of an attacker that he can discover the weakness;
- *Likelihood of Exploit (EX)*: represents the likelihood of an attacker with the required privileges/authentication/access would be able to successfully exploit it, if the weakness is discovered first by the attacker;
- *External Control Effectiveness (EC)*: is the capability of controls or mitigations outside of the software that may make the weakness more difficult to reach and/or trigger by an attacker;
- *Prevalence (P)*: identifies how frequently this type of weakness appears in software

2.2.2. CVSS - Common Vulnerability Scoring System

The CVSS¹⁵ is a widely adopted methodology which helps a user in specifying some of the main characteristics of a vulnerability and provides a resulting score representing the severity (of impact) of a vulnerability. This method, which current version (CVSSv3.1) was released in June 2019, is often combined with other threat modelling methods [36].

Similar to CWSS, it is composed of three groups of metrics (also shown in Figure 41).

The Base metric group: represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments; the *exploitability metrics* reflect the ease and technical means by which the vulnerability can be exploited; the *impact metrics* measure how a vulnerability, if exploited, will affect the vulnerable component. Based on the Base Metric Group, CVSS produces a numerical severity score ranging from 0.0 to 10.0, which can be modified by scoring the optional Temporal and Environmental metrics (they include a metric value that has no effect on the score).

The Temporal metric group: reflects the characteristics of a vulnerability that may change over time but not across user environments.

The Environmental metric group: represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment [36].

¹⁵ Common Vulnerability Scoring System

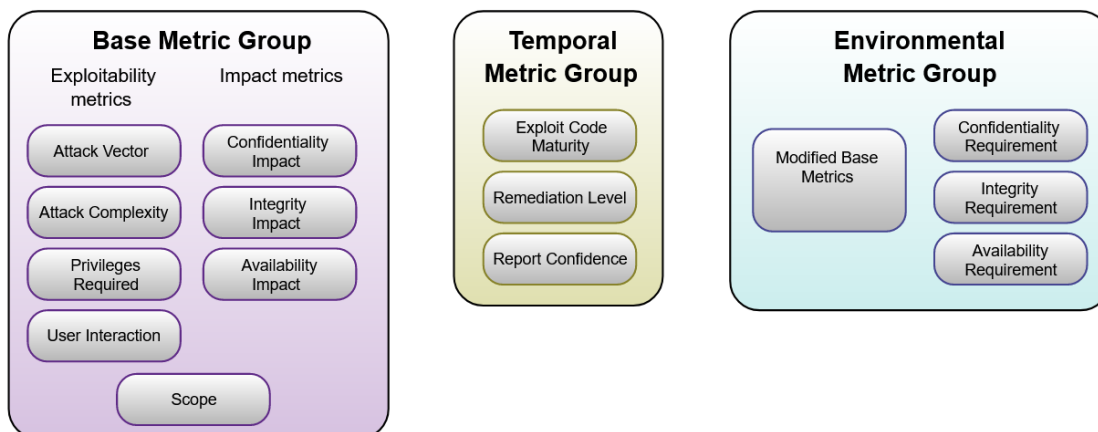


Figure 41 – The groups of CVSS metrics [36]

Although CVSS is similar to CWSS, some metrics like likelihood have been removed, leading to simpler to calculate metrics.

CVSS has been widely adopted, especially the use of base scores from the Base metric group and it represents a widely established approach; for example, it is used in the CVE and in the National Vulnerability Database (NVD) [51] created by the NIST.

2.2.3. VERACODE

The Veracode Security Quality Score, is a scoring system based on CWE dictionary of security flaws, used to map the flaws found in its static and dynamic scans, and on CVSS for the calculation of severity based on the potential Confidentiality, Integrity and Availability impact of a flaw CWE if exploited [62]. Each severity level reflects the business impact if a security breach occurs in these three security aspects.

It is part of the Veracode Platform which uses static and *DynamicDS* analysis (for web applications) to inspect executables and identify security flaws in applications.

Veracode assigns a severity level to each flaw type based on [62]:

- **Confidentiality Impact:** measures the impact on confidentiality if on a system vulnerability is exploited. At the weakness level, the Confidentiality is measured at three levels of impact: None, Partial and Complete, in according CVSS;
- **Integrity Impact:** measures the potential impact on integrity of the application. Integrity measures are needed to protect data from unauthorized changes. In fact, when the integrity of a system is solid, it is protected from unauthorized modifications of its contents.
- **Availability Impact:** measures the potential impact on availability if an attack is successful on the vulnerability. The Availability means to the accessibility of information resources. Typically, in this domain, the vulnerabilities are the Denial of Service. For example: Attacks that compromise authentication and authorization for application access, application memory, and administrative privileges.

The overall Security Quality Score, based on its associated CWE entry, is computed by aggregating impact levels of all weaknesses within an application. It enumerates the security weaknesses and their impact levels within the application code, but it does not predict the potential for vulnerability. It is a single score ranging from 0 to 100, where 0 is the insecure application and 100 is an application where the flaws have been discovered. The score calculation includes non-linear factors so that, for instance, a single Severity 5

flaw is weighted more heavily than five Severity 1 flaws, and so that each additional flaw at a given severity contributes progressively less to the score.

Weights of the Raw Score formula is exponential and are determined by empirical analysis by Veracode's application security experts.

The assurance levels follow a three-letter rating system (from A to F). The first letter is used for the results from binary analysis, the second for automated dynamic analysis, and the third for human testing. They are used to determine the extension of the testing (e.g., higher assurance levels could imply more testing techniques) and the overall acceptance criteria (e.g., a lower assurance level can be accepted with lower security scores if it does not pose a high business risk).

2.2.4. DREAD

DREAD (Damage, Reproducibility, Exploitability, Affected users, Discoverability) is a methodology which is part of a system for risk-assessment of computer security threats. It was used at Microsoft and currently it is used by OpenStack¹⁶.

It is similar to STRIDE as it provides a mnemonic for risk rating security threats using five categories [29], as shown in Figure 42:

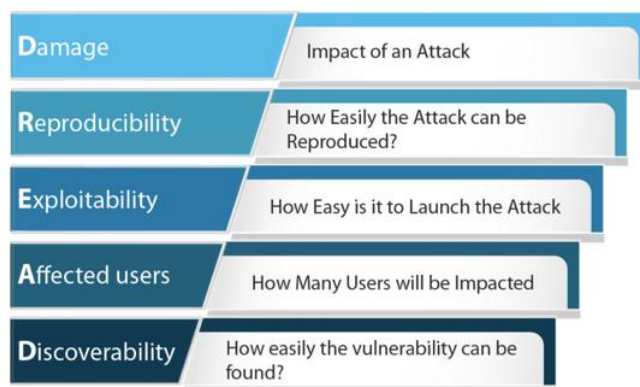


Figure 42 - DREAD Mnemonic

When a given threat is assessed using DREAD, each category is given a rating. The DREAD algorithm [70] is then used to compute a risk value, which is an average of all five categories. The calculation always produces a number between 0 and 10; the higher the number, the more serious the risk. The risk rating is obtained by adding rating values for all items and comparing the results with categories shown in Table 5. The sum of all ratings for a given issue can be used to prioritize among different issues [28].

Table 5 - Risk rating category-impact

Risk rating	Result
High	12 – 15
Medium	8 – 11
Low	5 - 7

However, there is not a consensus on how the actual risk point scale should be, since it all depends on the individuals performing the threat modelling [71]. DREAD requires scoring each of the five categories on a scale from zero to ten, which leads to discussions on the fine differences between consecutive numbers, e.g., five and six. This problem is still bigger in larger organizations with multiple teams. One solution to this problem, as

¹⁶ Openstack, «Security/OSSA-Metrics». <https://wiki.openstack.org/wiki/Security/OSSA-Metrics#Calibration>.

remarked in [72], is using scores of High, Medium, or Low, that are easy to agree, instead of using Microsoft's eleven-valued scale. For example, a simple scheme would be: High (10 points), Medium (5 points), and Low (0 points) when it comes to Damage potential, and Hard (0 points), Medium (5 points), Easy (0 points) when it comes to Reproducibility.

The context is not considered itself, but it can be taken into account when assigning the mark to each category. The same happens with the multilayer and complex systems, although an aggregation is not directly considered, it can be taken into account in the scale, as a global value.

2.2.5. OWASP Risk Rating

The OWASP Risk Rating Methodology [73] is part of the OWASP project, which provides a basis for testing web application technical security controls. The risk rating methodology estimates the risk in terms of likelihood and impact following several steps. The first one consists on identifying a risk to be rated, analysing and gathering information about it. The second step analyses factors for estimating likelihood. It is not necessary to be over-precise in this estimate. Generally, identifying whether the likelihood is low, medium, or high is sufficient. There are a number of factors that can help determine the likelihood, such as the ease of discovery and exploit or the skills of the attacker. The third step is about identifying factors for estimating Impact, divided in technical impact on the application, the data it uses, and the functions it provides and in business impact on the business and company operating the application. In this sense, the context factor can be considered through this metric. The fourth step determines the risk severity. The likelihood and impact estimate are put together to calculate an overall severity for this risk, obtaining none, low, medium, high or critical. Finally, it is decided what to Fix. It is also possible to customize the Risk Rating Model, for example adding factors, customizing options or weighting the factors.

The main limitation of OWASP is that it is only focused on web applications, domain in which there is no current standard [75]. As in the other schemes, the scale used (low, medium and high) based only on the consensus of the testers make the result subjective and variable depending on the person that is measuring the risk, and although it is not required a high precision on calculating the likelihood, this is one of the metrics more difficult to calculate based on the discussion of the challenges at the beginning of this section.

2.2.6. Cenzic HARM - Hailstorm Application Risk Metric

The Cenzic Hailstorm Application Risk Metric (HARM) [60] is a quantitative metric for the risk is associated with a web application. It is split into 4 impact areas: *Browser*, *Session*, *Application*, and *Infrastructure* (server environment). It also takes into account two additional factors, a *complexity* factor and the *precision* associated with detection of a given vulnerability and a modifier called weight, which users can use to modify the obtained risk.

Mathematically, the Base Risk Equation is $10 * 2^I$, where I is the impact area value. Any vulnerability can impact a Web application in up to 4 different ways (4 *impact areas*). Within those 4 areas, the degree of the risk can be 1 ("low") to 5 ("Critical"), represented as rings inside a circle.

To determine the application risk level (*impact value*) for a vulnerability, HARM uses security values with five degrees of risk such as confidentiality or access. The vulnerability

risk is the sum of the risk score from each of the four impact areas, which can be modified by the weights from other metrics (i.e., attack complexity, detection precision, asset value).

The Vulnerability Risk Equation (using $\alpha, \beta, \sigma, \varepsilon$ for the 4 different impact areas) is [60]:

$$\sum \{\alpha, \beta, \gamma, \varepsilon\}$$

the final Vulnerability Risk Equation (using χ, δ, ω for the other three factors respectively is):

$$\sum \{\alpha, \beta, \gamma, \varepsilon\} \cdot \chi \cdot \delta \cdot \omega$$

Finally, the HARM rating is calculated by multiplying all of the identified vulnerabilities (that can include different components and layers) within an application by the level of importance managers give to that application, so it gives the possibility of indirectly considering the context changing the weights.

However, this method does not account for the relationship of vulnerability properties, which are also important in the evaluation of the distribution of exploitation, and it is focused only on web applications [60].

2.3. Comparison of the Threat Modelling Methodologies and Risk Rating Systems

This document presented several different threat modelling methods and tools, and some security scoring and risk rating systems. Some are typically used standalone, some other is usually used in conjunction with others, and some are examples of combination of different methods.

Table 6 (which is our evolution of what listed in [37]) summarizes the main features of each threat modelling method, while risk rating and security scoring methods are summarized and compared in Table 7. Finally, a comparison of the tools is provided in Appendix A.

Table 6 – Summary of the Threat Modelling Methods

Threat Modelling Method	Main Features	Main Issues
Attack Tree	<ul style="list-style-type: none"> Helps identifying relevant mitigation techniques Has consistent results when repeated Is easy to use if the user already has thorough understanding of the system architecture, the threats and their possible combination 	<ul style="list-style-type: none"> Requires thorough understanding of the system where defensive measures are not modelled, attacker/defender interactions and evolutionary aspects are not considered;
STRIDE	<ul style="list-style-type: none"> Helps identifying relevant mitigating techniques Is the most mature Is easy to use 	<ul style="list-style-type: none"> Time Consuming No longer maintained Using DFDs as the only input to threat modelling is limiting because it does not provide a means for representing security-related architectural decisions [116]
VAST Modelling	<ul style="list-style-type: none"> Helps identify relevant mitigation techniques Directly contributes to risk management Contains built-in prioritization of threat mitigation Encourages collaboration among stakeholders Has consistent results when repeated Has automated components 	<ul style="list-style-type: none"> Has little publicly available documentation

	<ul style="list-style-type: none"> Is explicitly designed to be scalable 	
LINDDUN	<ul style="list-style-type: none"> Helps identify relevant mitigation techniques Contains built-in prioritization of threat mitigation 	<ul style="list-style-type: none"> Can be labour intensive and time consuming Limited to privacy threats
PASTA	<ul style="list-style-type: none"> Helps identify relevant mitigating techniques Directly contributes to risk management Encourages collaboration among stakeholders Contains built-in prioritization of threat mitigation Has rich documentation 	<ul style="list-style-type: none"> Is laborious
Trike	<ul style="list-style-type: none"> Helps identify relevant mitigation techniques Directly contributes to risk management Contains built-in prioritization of threat mitigation Encourages collaboration among stakeholders Has automated components 	<ul style="list-style-type: none"> Has vague, insufficient documentation
OCTAVE	<ul style="list-style-type: none"> Helps identify relevant mitigation techniques Directly contributes to risk management Contains built-in prioritization of threat mitigation Encourages collaboration among stakeholders Has consistent results when repeated Is explicitly designed to be scalable 	<ul style="list-style-type: none"> Time Consuming Vague, complex and large documentation
ADVISE	<ul style="list-style-type: none"> Considers adversaries and their characteristics The related tool (Mobius) provides simulation features The Atomic Formalism can be used in conjunction with other formalisms (e.g., SANs) 	<ul style="list-style-type: none"> Time Consuming Requires thorough understanding of the system Threat model must be known a priori by the user
Security Cards	<ul style="list-style-type: none"> Encourages collaboration among stakeholders Targets out-of-the-ordinary threats 	<ul style="list-style-type: none"> Leads to many false positives
Persona non Grata	<ul style="list-style-type: none"> Helps identify relevant mitigation techniques Directly contributes to risk management Has consistent results when repeated It produces few false positives and has high consistency Fits well into the agile approach, which incorporates personas 	<ul style="list-style-type: none"> Tends to detect only some subsets of threats
hTMM	<ul style="list-style-type: none"> Contains built-in prioritization of threat mitigation Encourages collaboration among stakeholders Has consistent results when repeated 	<ul style="list-style-type: none"> Has little documentation
CORAS	<ul style="list-style-type: none"> Customised language for threat and risk modelling UML-like modelling 	<ul style="list-style-type: none"> The methodology is a result of a research project, and the related tool is old and not maintained since years¹⁷
HAZOP	<ul style="list-style-type: none"> Systematic and comprehensive Examines the consequences of failures 	<ul style="list-style-type: none"> Safety-oriented (security-oriented variant exists and is called THROP) Time consuming and expensive Requires detailed design drawing to perform the full study Additional guidewords are required for unusual hazards

¹⁷ http://coras.sourceforge.net/coras_tool.html

		<ul style="list-style-type: none"> • Requires experienced practitioners • Focuses on one-event causes of deviation only
--	--	---

Table 7 Summary of the Risk Rating and Scoring Systems

Risk Rating / Security Scoring System	Features	Main Issues
CVSS	<ul style="list-style-type: none"> ▪ Recommended by the ITU-T ▪ Used in several databases (e.g., CWE or OWASP top ten) ▪ Can be applied early in the process ▪ Built-in support for incomplete information 	<ul style="list-style-type: none"> • Metrics as likelihood are difficult to compute
CVSS	<ul style="list-style-type: none"> ▪ Widely adopted (e.g., in CVE and NVD) ▪ Contains built-in prioritization of threat mitigation ▪ Has consistent results when repeated ▪ Automated components ▪ Has score calculations that are not transparent ▪ Metrics calculation is simpler w.r.t. CWSS 	<ul style="list-style-type: none"> • Assumes that a vulnerability has already been discovered and verified • Does not account for incomplete information • Large bias towards the impact on the physical system
VERACODE	<ul style="list-style-type: none"> ▪ Based on VWE and CVSS 	<ul style="list-style-type: none"> • Unknown
DREAD	<ul style="list-style-type: none"> ▪ Similar to STRIDE ▪ The context can be taken into account when assigning the mark to each category 	<ul style="list-style-type: none"> • There is not a consensus on risk point scale • Requires scoring each of the categories
OWASP Risk Rating	<ul style="list-style-type: none"> ▪ Provides a basis for testing security controls ▪ Multilayer and aggregation can be included by considering a global mark 	<ul style="list-style-type: none"> • It focuses on web applications only • Scale is based only on consensus of the testers • Result is subjective and variable • Likelihood is difficult to calculate
Cenzic HARM	<ul style="list-style-type: none"> ▪ It gives the possibility of indirectly considering the context changing the weights 	<ul style="list-style-type: none"> • Does not account relationship of vulnerability properties • Focused only on web applications

2.4. Reference Security Standards for Threat Analysis and Risk Assessment

This section reports the results of a research, analysis and comparison of the reference security standards for threat analysis and risk assessment, which guided the development of the risk assessment methodology and its application for the threats and hazard identification within the use cases and the BIECO framework.

2.4.1. Introduction

Standards are specifications that establish the fitness of a product for a particular use by guiding the development or assessment of systems and systems components. In some cases, from a user/operator perspective, standards define the function and performance of a device or system. On top of providing best practices, standards are key facilitators of compatibility and interoperability, and define specifications for languages, communication protocols, data formats, linkages within and across systems, interfaces between software applications and hardware devices and much more.

The ICT supply chain, in particular, is part of complex dynamic and globally interconnected ecosystem that encompasses the entire life cycle of ICT hardware, software, and

managed services together with a wide range of actors (entities, considered ecosystem components)—including third-party vendors, suppliers, service providers, and contractors. Certain actors, such as governments and industries purchase products and services and use them to power and enable critical infrastructure systems. Other actors, such as OEMs (Original Equipment Manufacturer) integrate different products provided by multiple other organizations or developers (actors within an ecosystem) in the process of building a final product for the end-user. However, a supply chain is only as strong as its weakest link. Foreign adversaries, hackers, and criminals seeking to steal, compromise or alter, and destroy sensitive information can target their victims at all tiers of the ICT supply chain [93]. Attacks can be caused by intruders e.g., through direct physical attacks on systems [97], or cyber-attacks on the digital parts of a system [98], or even insider attacks through human involved actors, part of the developing ecosystem [99].

Securing the supply chain within a dynamic ecosystem is a complex activity, since *vulnerabilities* may be introduced and exploited during any phase of the product life cycle: from design and development to production, and further on during distribution, acquisition and deployment, maintenance.

For addressing the vulnerability of systems, there are several, different, security standards, specific to each phase of the lifecycle, the type of data, or system and user to be protected, by considering the application domain, the type of protection, and all the important aspects that need to be considered in case of attack.

Section 2.4 presents an overview of standard concepts and methodologies which will guide the development and application of a risk assessment process. Based on this, the use case systems can be analysed for identifying potential threats, vulnerabilities, weaknesses, starting with the early prototyping stage and until maintenance.

2.4.2. Basic Concepts and Risk Model

This Section provides a brief overview of the basic concepts that are fundamental for the definition of risk assessment process and for its application to BIECO use cases. Most of the definitions are originating from NIST¹⁸ SP 800-30 [43], since it is a widely adopted¹⁹ guide for risk assessment, even if good alternatives exist, e.g., the ENISA glossary [126].

Figure 43 gives an example of a *risk model* including the key *risk factors* which will be discussed in the following sections. A glossary with a more comprehensive set of definitions for security terminology can be found in [43].

2.4.2.1. Risk

According to NIST SP 800-30 [43], *Risk* is: *a measure of the extent to which an entity is threatened by a potential circumstance or event, and is typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence.*

In the context of ICT security, risk arises from to the loss of confidentiality, integrity or availability of information or of the ICT system. In the NIST 800-30 [43], the *Risk Assessment* is defined as a *process of identifying, estimating and prioritizing risk.*

¹⁸ NIST: National Institute of Standards and Technology and it is a government agency of the United States of America that deals with technology management.

¹⁹ several studies and assessments conducted following this guide have been published

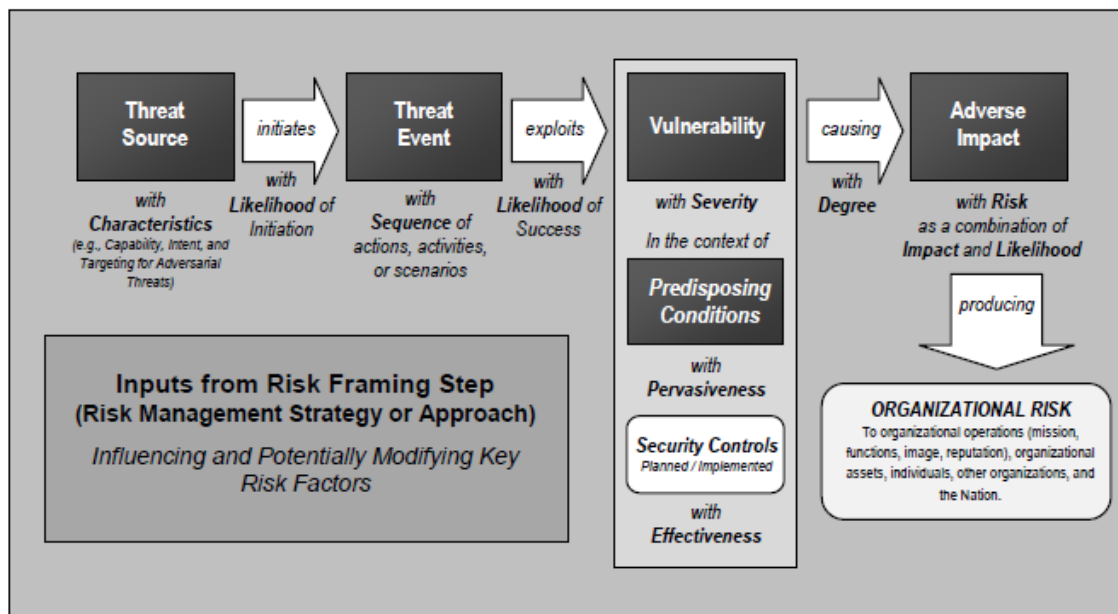


Figure 43 Generic risk model and key risk factors from NIST SP 800-30 (source: [43])

Assessing risk requires the careful analysis of *threat* and *vulnerability* information to determine the extent to which circumstances or events could adversely *impact* an organization and the *likelihood* that such circumstances or events will occur.

2.4.2.2. Vulnerabilities and Weaknesses

ICT systems may be prone to cyber-attacks both inside and outside the system network, boundaries or premises. In order to analyse and discover the vulnerabilities associated with systems, it is often necessary to have a thorough understanding, e.g., to know the types of communications and operations associated with the system, and many other technical, architectural and procedural details. Doing so, it is possible to understand how attackers may make use of the vulnerabilities of the system, system components, processes and architecture to their advantage, to carry out intrusions, and achieve malicious goals.

A vulnerability, as defined by the NIST is [43]: *a weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source*. For example, hardware vulnerabilities can be exploited by special crafted software components that take advantages of it into expressing malicious behaviours. In particular, intended faults can be inserted within software components that during operation can express in a range of malicious behaviour, ranging from overheating a system through fast execution or steal information from shared memory locations.

According to the CVE [49] a widely used²⁰ list of publicly known vulnerabilities, a vulnerability is a flaw resulting from a weakness that can be exploited, causing a negative impact to the security of impacted components. For example, hardware weaknesses such as proximity of memory cells can create a vulnerability that can be exploited by software components specially crafted to take advantage of a device hardware structure.

But most information system vulnerabilities can be associated with *security controls* which are meant to protect the threatened CIA security properties (*confidentiality, integrity,*

²⁰ Among those products and services, the CVE also *feeds* the NVD (US National Vulnerability Database) which builds upon the information included in CVE to provide enhanced information such as fix information, severity scores, impact ratings, and searching features.

and *availability*). In this sense, *preventive control* mechanisms such as the locking out unauthorized intruders can be applied before an intrusion event. During the intrusion event, *detective controls* can identify and characterize the intrusion, and based on its specifics, trigger notifications alarms. The process of building trust in system operations, requires triggering of operational fail-over behave. Therefore, within BIECO, following the runtime detection of malicious intrusions that made their way into the operational phased of a system, fail-over behaviours are triggered for assuring the ultimate trustworthy operation of a system. Typically, after an intrusion event has been detected, *corrective controls* are put in place for limiting the extent of damage. When security controls are not applied, or the systems retain some weaknesses, emergent vulnerabilities can arise over time. For example, system evolution, changes in environment, proliferation of new technologies and new threats create vulnerabilities along the full lifecycle of a system [43].

In general, *risk* materializes as a result of a series of *threat events*, each of which takes advantage of one or more vulnerabilities. For example, the risk of leading to a physical crash in a potential hazardous situation, results from hidden undetected malicious behaviours that made the way from the design through the operational phase of a system.

Generally, the development of threat scenarios is analytically useful, since some vulnerabilities may not be exposed to exploitation unless and until other vulnerabilities have been exploited. Within a supply chain, system and system components (including software components) are shipped sequentially, malicious attacks can manifest based on strategically introduced faults that coordinate operation of multiple components. Analysis that illuminates how a set of vulnerabilities, taken together, could be exploited by one or more threat events w.r.t one component as well as multiple inter-related components is therefore, in a supply chain, much more useful than the analysis of individual vulnerabilities.

The *severity* of a vulnerability is an assessment of the importance of/ the required effort in/ mitigating or correcting the vulnerability itself. It can be determined by the extent of the potential adverse *impact*, if it is exploited by a threat source. Typically, it is context-dependent. For example, within an ICT supply chain, the severity of leaving a software (considered vulnerable) not updated is analysed with regard to the importance of having a wireless channel for delivering a software component or to the required effort of applying software updates in-house.

The *assessment* of vulnerabilities is intended as a systematic examination of an information system or product to determine the adequacy of security measures, identify security gaps, provide data from which to predict the effectiveness of proposed security measures and confirm the adequacy of these measures after implementation. The NIST 800-53 [84], and the ISO/IEC 27005 [83] provide guidelines for the assessment of vulnerabilities. Within BIECO, based on assessed vulnerabilities, safe fail-over behaviour is triggered. For example, a system considered vulnerable to security attacks, which manifest into malicious behaviour of software components during runtime operation, is designed with a failure prediction mechanism in place that is triggered in specific technical situations. These technical situations describe the scenarios in which systems operate.

According to the fundamentals of risk assessment from NIST 800-30 [43], an *analysis approach* can be *vulnerability-oriented* when it starts with a set of predisposing conditions or exploitable weaknesses/deficiencies, and identifies threat events that could exercise those vulnerabilities together with possible consequences of vulnerabilities being exercised. Other approaches (i.e., threat-oriented and asset/impact oriented) are

described in Section 2.4.3.1. Examples of vulnerability-oriented analysis of a system within BIECO will start with a set of identified architectural and logical weaknesses, based on which threat events are deduced.

In the perspective of CWE, a community-developed list of common software and hardware weakness types that have security ramifications, weaknesses are [86]: *flaws, faults, bugs, vulnerabilities, or other errors in software or hardware implementation, code, design, or architecture that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack.*

The concept of weakness is tightly related to the notion of *predisposing condition* of NIST SP 800-30 [43], which meaning is broader: *A predisposing condition is a condition that exists within an organization, a mission or business process, enterprise architecture, information system, or environment of operation, which affects (i.e., increases or decreases) the likelihood that threat events, once initiated, result in adverse impacts to organizational operations and assets, individuals, other organizations.*

Vulnerabilities (including those attributed to predisposing conditions) are part of the overall security posture of organizational information systems and environments of operation that can affect the likelihood of occurrence of a threat event.

2.4.2.3. Threats

As described by NIST SP 800-30 [43] and 800-82 [85], a *threat* is: *any circumstance or event with the potential to adversely impact organizational operations and assets [...]*. For example, an undetected malicious behaviour of a software component that holds the control of a system.

Threat Source

A *Threat Source* is an initiator of an attack, and is characterized by the intent and method targeted at the exploitation of a vulnerability or by a situation and method that may accidentally exploit a vulnerability. An example of threat source is a maliciously intended employee which inserts faults in a software component, endangering the system that executes the software, as well as the whole ecosystem around it, including other systems, system components and actors.

Further types of threat sources included in NIST 800-30 [43] are:

1. hostile cyber or physical attacks;
2. human errors of omission or commission;
3. structural failures of organization-controlled resources (e.g., hardware, software, environmental controls);
4. natural and man-made disasters, accidents, and failures beyond the control of the organization.

A useful taxonomy of threat sources has been developed by [43], and is shown in Table 8.

Table 8 - Taxonomy of Threat Sources [43]

Type of Threat Source	Description	Characteristics
ADVERSARIAL - Individual <ul style="list-style-type: none"> o Outsider o Insider o Trusted Insider o Privileged Insider - Group <ul style="list-style-type: none"> o Ad hoc o Established - Organization <ul style="list-style-type: none"> o Competitor o Supplier o Partner o Customer - Nation-State	Individuals, groups, organizations, or states that seek to exploit the organization's dependence on cyber resources (i.e., information in electronic form, information and communications technologies, and the communications and information-handling capabilities provided by those technologies).	Capability, Intent, Targeting
ACCIDENTAL - User - Privileged User/Administrator	Erroneous actions taken by individuals in the course of executing their everyday responsibilities.	Range of effects
STRUCTURAL - IT Equipment <ul style="list-style-type: none"> o Storage o Processing o Communications o Display o Sensor o Controller - Environmental Controls <ul style="list-style-type: none"> o Temperature/Humidity Controls o Power Supply - Software <ul style="list-style-type: none"> o Operating System o Networking o General-Purpose Application o Mission-Specific Application 	Failures of equipment, environmental controls, or software due to aging, resource depletion, or other circumstances which exceed expected operating parameters	Range of effects
ENVIRONMENTAL - Natural or man-made disaster <ul style="list-style-type: none"> o Fire o Flood/Tsunami o Windstorm/Tornado o Hurricane o Earthquake o Bombing o Overrun - Unusual Natural Event (e.g., sunspots) - Infrastructure Failure/Outage <ul style="list-style-type: none"> o Telecommunications o Electrical Power 	Natural disasters and failures of critical infrastructures on which the organization depends, but which are outside the control of the organization. Note: Natural and man-made disasters can also be characterized in terms of their severity and/or duration. However, because the threat source and the threat event are strongly identified, severity and duration can be included in the description of the threat event (e.g., Category 5 hurricane causes extensive damage to the facilities housing mission-critical systems, making those systems unavailable for three weeks).	Range of effects

Threat Event and Attack Pattern

Threat events are caused by threat sources, and can be defined as [43]: *-single or sets of events, actions, or circumstances that can potentially cause undesirable consequences or impact.*

When a set of discrete threat events, attributed to a specific threat source or multiple threat sources, are partially ordered in time and result in adverse effects, they originate a

so-called *threat scenario*. For example, an adversarial threat source in form of a trusted insider, such as a developer of software components, causes the threat event of hiding a malicious code within a software component that control an actuator of a cyber-physical system. Then during, runtime, in specific situations e.g., when it is most likely to reach a target destruction impact, the malicious behaviour causes an unwanted commission of the actuator.

Knowing the *intent* and targeting aspects of a potential attack, helps organizations narrow the set of threat events that are most relevant to consider. In the previous example, active considerations of catastrophic events rely on deployment of safety mechanisms that can trigger fail-over behaviour based on predicted malicious intentions of control software. In fact, multiple threat sources can initiate or cause the same threat event, e.g., a provisioning server can be taken off-line by a denial-of-service attack, a deliberate act by a malicious system administrator, an administrative error, a hardware fault, or a power failure.

So, when threat events are identified with great specificity, threat scenarios can be modelled, developed and analysed.

Threat events for cyber or physical attacks are characterized by the Tactics, Techniques, and Procedures (TTPs) employed by adversaries. Understanding adversary-based threat events gives organizations insights into the capabilities associated with certain threat sources.

In this regard, the Appendix E of NIST 800-30 [43], provides representative threat events initiated by threat sources, and includes:

- a description of potentially useful inputs to the threat event identification task;
- representative examples of adversarial threat events expressed as tactics, techniques, and procedures (TTPs) and non-adversarial threat events;
- an exemplary assessment scale for the relevance of those threat events;
- templates for summarizing and documenting the results of the threat identification.

The level of detail of TTPs is established as part of the organizational risk frame and it is intended to support risk assessments at all three tiers (organization level, business process level, information system level), and to be tailorable to include additional details, as necessary. The standard NIST SP 800-30 [43] refers to the CAPEC [54] as a reference for detailed descriptions of threat events that exploit software. CAPEC is a publicly available catalogue of common *attack patterns* that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities. Examples of well-known attack patterns from CAPEC are: SQL injection, buffer overflow, http response splitting, etc.

In this catalogue, some attack patterns are defined as descriptions of the common attributes and approaches employed by adversaries to exploit known weaknesses in cyber-enabled capabilities. Attack patterns define the challenges that an adversary may face and how they go about solving it. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples. Each attack pattern captures knowledge about how specific parts of an attack are designed and executed, and further on, gives guidance on ways to mitigate the attack's effectiveness. Attack patterns help the development of applications, or administrating cyber-enabled capabilities to better understand the specific elements of an attack and how to stop them from succeeding [54].

Hazard vs Threat

The various views on safety and security result in different terms being used for similar concepts or even the same term being used with slightly differing meaning. In the field of systems with safety-critical functionalities, the term *hazard* is widely used and often preferred to *threat/threat source* terms.

Thus, a methodology for risk assessment aimed at assuring the ultimate trust should be sufficiently general to be applied not only to the cybersecurity domain but to cyber-physical-security and safety domains. Therefore, we report here a definition of *hazard* concept, a technique for *hazard identification* and some considerations on its relation with the *threat* concept. A *hazard* is a potential source of harm, and it can be a constituted or produced by deviations from design or operational intent. It should be noted that a single *hazard* could potentially lead to multiple forms of *harm*.

A *threat* is a very similar concept from the security domain where the undesirable consequences will primarily affect the security properties of the system under consideration [95]. However, security-threats may have consequences that go beyond security. Security-threats can have adverse impact on safety and dependability in general and therefore on the ultimate level of trustworthy operation of a system. It is not surprising, in fact, that one recent safety standard as the EN 50129:2018 [96] explicitly names security-threats as causes of functional safety hazards, and IT-security as a field that can affect functional safety. Explicit reference to security standards as, in example, the ISA/IEC 62443 [89] is given in the context of [96].

In order to identify potential hazards and operability problems, a well-known standard [94] technique named HAZOP (described in Section 2.1.13) is often adopted.

Among the activities typically performed in the field of systems with safety-critical functionalities, the *hazard analysis* enables identification of potentially dangerous situations that could occur with consequences from a safety point of view. In order to proceed with the identification of potential hazards, the approach followed in the *risk analysis* involves the creation of a functional logical model of the system analysed. This model divides the system into functional blocks, where a part of the functions performed by the system are assigned to each block. In addition, the internal and external *interfaces* of the system are also defined, identifying the data/signals that the functional blocks exchange with each other and with external systems.

Typically, starting from the system model, the *hazard analysis* is carried out by applying appropriate *keywords* to the identified functions and interfaces. The use of keywords has the dual purpose of providing guidance in the analysis of hazards and ensuring maximum coverage in identifying hazard scenarios. HAZOP guide words, are key supporting elements in the execution of a HAZOP analysis according to IEC Standard 61882 [94].

2.4.2.4. Likelihood

According to NIST SP 800-30 [43], the *overall likelihood*, combines an estimate of the *likelihood of initiation/occurrence* of a threat event with an estimate of the *likelihood of impact*.

The *likelihood of occurrence* is a weighted risk factor based on an analysis of the probability that a given threat is capable of exploiting a given vulnerability (or set of vulnerabilities). For adversarial threats, an assessment of likelihood of occurrence is typically based on [43]:

- i. adversary *intent*;

- ii. adversary *capability*; and
- iii. adversary *targeting* (e.g., goal).

For other than adversarial threat events, the likelihood of occurrence is estimated using historical evidence, empirical data, or other factors. The likelihood of threat occurrence can also be based on the state of the organization—taking into consideration predisposing conditions and the presence and effectiveness of deployed security controls to protect against unauthorized/undesirable behaviour, detect and limit damage, and/or maintain or restore mission/business capabilities.

The *likelihood of impact* addresses the probability (or possibility) that the threat event will result in an adverse impact, regardless of the magnitude of harm that can be expected.

Organizations typically employ a three-step process to determine the *overall likelihood* of threat events:

1. assess the likelihood that threat events will be initiated (for adversarial threat events, mainly subject to security attacks) or will occur (for non-adversarial threat events, such as random or sporadic failures).
2. assess the likelihood that the threat events once initiated or occurring, will result in adverse impacts or harm to organizational operations and assets, individuals, other organizations, or the Nation.
3. Finally, assess the overall likelihood as a combination of likelihood of initiation/occurrence and likelihood of resulting in adverse impact.

2.4.2.5. Impact

The NIST SP 800-30 [43] defines two types of impact: *level of impact* and *impact values*.

- The level of *impact* from a threat event is: *the magnitude of harm that can be expected to result from the consequences of unauthorized disclosure of information, unauthorized modification of information, unauthorized destruction of information, or loss of information or information system availability*.
- The *impact value* is: *the assessed potential impact resulting from a compromise of the confidentiality, integrity, or availability of an information type, expressed as a value of low, moderate, or high*.

This is clearly related to key security properties (i.e., the CIA triad), while in other domains, or in a more general perspective, the set of compromised properties may be wider. Other approaches, e.g., the EVITA [127] or HEAVENS [128] for smart vehicles, also consider impact dimensions (operational, safety, privacy and financial).

2.4.3. Overview of the Selected Standards

There are several security standards that can be applied, also depending on the domain of application. Many factors may guide the choice of one standard rather than another, e.g., the goal of the activities, the target system which has to be modelled and analysed, and so on. The standards that have been considered for this analysis and which will be the main reference for the risk assessment methodology of BIECO use cases are analysed in this Section. In any case, the list of standards is not exhaustive and does not constitute a comprehensive review: they were chosen based on our expertise and on the interest for the BIECO project and use cases.

The details of the comparison of different security standards belonging to various domains is given in Section 2.4.3.6 and Appendix B.

2.4.3.1. NIST SP 800-30

Risk Assessment Methodology

The NIST SP 800-30 [43], proposes a risk assessment methodology, which includes:

- *risk model*: which explicitly defines key terms and risk factors (partially already introduced in Section 2.4.2 and Figure 43 of this report);
- *assessment approach*: either *quantitative* or *qualitative*;
- *risk assessment process*;
- *analysis approach*: which describes how risk factors are identified and analysed in order to increase the coverage of the "problematic" space.

According to NIST SP 800-30, the documentation of a risk model includes:

- identification of risk factors (description and scales of values).
- identification of the relationships between risk factors (both conceptual relationships, presented descriptively, and algorithms for combining values).

The same document provides also a wide set of appendices with examples of application of a risk assessment.

Risk Assessment Approaches

As reported in NIST 800-30, risk and its factors, can be assessed in multiple ways, including quantitative, qualitative or semi-qualitative approaches. Typically, each approach has its own advantages and disadvantages

Quantitative assessments usually employ a set of methods, principles or rules for risk based on the use of numbers, and where necessary integrated with some explanations relating to the assumptions and constraints on the use of the results.

Qualitative assessments, instead, typically employ a series of methods, principles or rules are used for risk assessment, based on non-numerical categories or levels, such as *very low, low, moderate, high and very high*. This type of assessment supports communicating risk results to decision makers. However, the range of values in qualitative assessments is comparatively small in most cases, making the relative prioritization or comparison within the set of reported risks difficult. Additionally, unless each value is very clearly defined or is characterized by meaningful examples, different experts relying on their individual experiences could produce significantly different assessment results. The repeatability and reproducibility of qualitative assessments are increased by the annotation of assessed values (e.g., "this value is high because of the following reasons") and by the use of tables or other well-defined functions to combine qualitative values. *Semi-quantitative* assessments typically employ a set of methods, principles, or rules for assessing risk that uses bins, scales, or representative numbers whose values and meanings are not maintained in other contexts. This type of assessment can provide the benefits of quantitative and qualitative assessments. The bins (e.g., 0-15, 16-35, 36-70, 71-85, 86-100) or scales (e.g., 1-10) translate easily into qualitative terms that support risk communications for decision makers (e.g., a score of 95 can be interpreted as very high).

Examples of qualitative and semi-qualitative assessment scale from [43] are given in Table 9 and Table 10.

Table 9 Assessment scales for the level of risk (source: [43])

Qualitative Values	Semi-Quantitative Values	Description
Very High	96-100	10
Very high risk means that a threat event could be expected to have multiple severe or catastrophic adverse effects on		

			organizational operations, organizational assets, individuals, other organizations, or the Nation.
High	80-95	8	High risk means that a threat event could be expected to have a severe or catastrophic adverse effect on organizational operations, organizational assets, individuals, other organizations, or the Nation.
Moderate	21-79	5	Moderate risk means that a threat event could be expected to have a serious adverse effect on organizational operations, organizational assets, individuals, other organizations, or the Nation.
Low	5-20	2	Low risk means that a threat event could be expected to have a limited adverse effect on organizational operations, organizational assets, individuals, other organizations, or the Nation.
Very Low	0-4	0	Very low risk means that a threat event could be expected to have a negligible adverse effect on organizational operations, organizational assets, individuals, other organizations, or the Nation.

Table 10 Assessment scales for the level of risk – Combination of Likelihood and Impact (source: [43])

Likelihood (Threat Event Occurs and Results in Adverse Impact)	Level of Impact				
	Very Low	Low	Moderate	High	Very High
Very High	Very Low	Low	Moderate	High	Very High
High	Very Low	Low	Moderate	High	Very High
Moderate	Very Low	Low	Moderate	Moderate	High
Low	Very Low	Low	Low	Low	Moderate
Very Low	Very Low	Very Low	Very Low	Low	Low

Risk Assessment Process

The risk assessment process described in NIST 800-30 focuses on assessing information security risk and it is composed of four steps, which are described in Table 11, while Step 2 “conduct risk assessment” is shown expanded in Figure 44.

Table 11 – Steps of Risk Assessment Process from [43]

Steps	Description
1. Prepare for Risk Assessment	Identify: purpose and scope, assumptions and constraints, information sources, the risk model and analytic approaches to be employed during the assessment
2. Conduct Risk Assessment	Identify: threat sources and threat events, vulnerabilities and predisposing conditions correlated. Determine: the likelihood that the identified threat sources would initiate specific threat events and the likelihood that the threat events would be successful, adverse impact and information security risks as a combination of likelihood of threat exploitation of vulnerabilities and the impact of such exploitation, including any uncertainties associated with the risk determinations
3. Communicate and Share Risk Assessment Results	Communicate the result of risk assessment and to share the related information;
4. Maintain Risk Assessment	Monitor the risk factors and update the risk assessment

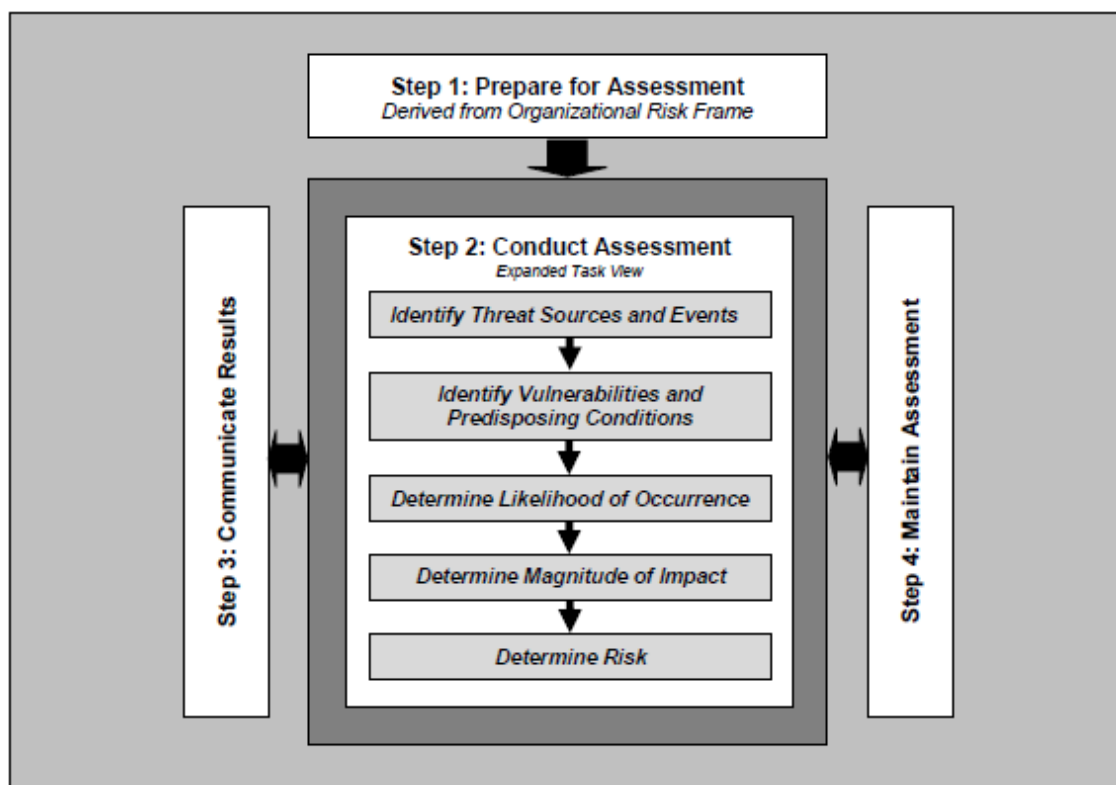


Figure 44 Risk Assessment Process – Step 2 Conduct Assessment Expanded View (source: [43])

Analysis Approach

According to the fundamentals of risk assessment from NIST 800-30 [43], an *analysis approach* can be:

- *vulnerability-oriented*: starts with a set of *predisposing conditions* or exploitable weaknesses/deficiencies in organizational information systems or the environments in which these systems operate, and identifies threat events that could exploit those vulnerabilities together with possible consequences of exercised vulnerabilities;
- *threat-oriented*: starts with the identification of threat sources and threat events, and focuses on the development of threat scenarios; vulnerabilities are identified in the context of threats. For adversarial threats, impacts are identified based on adversary intent;
- *asset/impact-oriented*: starts with the identification of impacts or consequences of concern and critical assets, possibly using the results of a mission or business impact analyses and identifying threat events that could lead to and/or threat sources that could seek those impacts or consequences;

Each analysis approach takes into consideration the same risk factors, and thus entails the same set of risk assessment activities, albeit in different order. Differences in the starting point of the risk assessment can potentially bias the results, causing some risks not to be identified. Therefore, identification of risks from a second orientation (e.g., complementing a threat-oriented analysis approach with an asset/impact-oriented analysis approach) can improve the rigor and effectiveness of the analysis.

In addition to the orientation of the analysis approach, organizations can apply more rigorous analysis techniques (e.g., graph-based analyses) to provide an effective way to account for the many-to-many relationships between:

- (i) threat sources and threat events (i.e., a single threat event can be caused by multiple threat sources and a single threat source can cause multiple threat events);
- (ii) threat events and vulnerabilities (i.e., a single threat event can exploit multiple vulnerabilities and a single vulnerability can be exploited by multiple threat events); and
- (iii) threat events and impacts/assets (i.e., a single threat event can affect multiple assets or have multiple impacts, and a single asset can be affected by multiple threat events).

For example, graph-based analysis techniques (e.g., functional dependency network analysis, *attack tree analysis* for adversarial threats, fault tree analysis for other types of threats) provide ways to use specific threat events to generate threat scenarios. Graph-based analysis techniques can also provide ways to account for situations in which one event can change the likelihood of occurrence for another event.

Attack and fault tree analyses, in particular, can generate multiple threat scenarios that are nearly alike, for purposes of determining the levels of risk. With automated modelling and simulation, large numbers of threat scenarios (e.g., attack/fault trees, traversals of functional dependency networks) can be generated. Thus, graph-based analysis techniques include ways to restrict the analysis into defining a reasonable subset of all possible threat scenarios [43].

2.4.3.2. NIST Cybersecurity Framework

Other standards of the same institute, NIST 800-53 rev4 [90] and NIST 800-53 rev5 [91], focus on how to prevent an “incident” and on possible mitigations in case of its occurrence.

NIST 800-53 Rev5 [91], introduces a *cybersecurity framework*, which integrates industry standards and best practices to help organizations manage risk.

The Cybersecurity Framework provides a systematic methodology for risk management, envisaging five macro-processes (functions):

1. *Identify*: assists in developing an organizational understanding to managing cybersecurity risk to systems, people, assets, data, and capabilities;
2. *Protect*: outlines appropriate safeguards to ensure delivery of critical infrastructure services. The Protect Function supports the ability to limit or contain the impact of a potential cybersecurity event;
3. *Detect*: defines the appropriate activities to identify the occurrence of a cybersecurity event. The Detect Function enables timely discovery of cybersecurity events;
4. *Respond*: includes appropriate activities to take action regarding a detected cybersecurity incident. The Respond Function supports the ability to contain the impact of a potential cybersecurity incident;
5. *Recover*: identifies appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity incident.

An example of application of the Cybersecurity Framework is NIST.IR 7628 [87] which is described in the following section.

2.4.3.3. NIST.IR 7628

NIST.IR 7628 *Guidelines for Smart Grid Cybersecurity* is a standard composed of three volumes which presents an analytical framework that organizations can use to develop effective cybersecurity strategies tailored to their particular combinations of smart grid-related characteristics, risks and hazard, vulnerabilities and weaknesses [87].

The document is a companion to the NIST Framework and Roadmap for Smart Grid Interoperability Standards (NIST SP 1108) [88] and describes a high-level conceptual reference model for the Smart Grid, identifying standards that are applicable to the ongoing development of an interoperable Smart Grid, and specifies a set of high-priority standards-related gaps and issues.

The report contributes to an increased understanding of the key elements critical to realization of the smart grid, including standards-related priorities, strengths and weaknesses of individual standards, the level of effective interoperability among different smart grid domains, and cybersecurity requirements.

The guidelines are intended primarily for individuals and organizations responsible for addressing cyber security for Smart Grid systems and the constituent subsystems of hardware and software components [87]. The contents of the three volumes are summarized in the following.

Volume 1 – *Smart Grid Cybersecurity Strategy, Architecture and High-Level Requirements*:

- discusses the cybersecurity strategy for the smart grid and the specific tasks within this strategy;
- includes a high-level diagram that depicts a composite high-level view of the actors within each of the smart grid domains and includes an overall logical reference model of the smart grid, including all the major domains;
- describes the approach, including the *risk assessment process* to identify the high-level security requirements.
- concludes with a discussion of technical cryptographic and key management issues across the scope of Smart Grid systems and devices.

Volume 2 – *Privacy and the Smart Grid*: provides awareness and discussion of topics regarding privacy issues. Additionally, the second volume provides recommendations, based on widely accepted privacy principles, for entities that participate within the Smart Grid, including an overview of some existing *privacy risk mitigation standards* and frameworks.

Volume 3 – *Supportive Analyses and References*: it is a compilation of supporting analyses and references used to develop the high-level security requirements and other tools and resources presented in the first two volumes, including classes of potential vulnerabilities for the smart grid that are classified by category and it identifies a number of specific security problems in the smart grid.

Smart Grid Risk Assessment

The smart grid risk assessment process described in NIST.IR7628 [87] is based on existing risk assessment approaches developed by both the private and public sectors and includes identifying assets, vulnerabilities, and threats and specifying impacts to produce an assessment of risk to the smart grid and to its domains and subdomains, such as homes and businesses. Between the documents used in developing the risk assessment process of [87], there are NIST SP 800-30 and ANSI/ISA 62443.

Because the smart grid includes systems from the IT, telecommunications, and electric sectors, the risk assessment process is applied to all three sectors as they interact in the smart grid.

The risk assessment in [87] has been undertaken from a high-level, overall functional perspective. The output was the basis for the selection of security requirements and the identification of gaps in guidance and standards related to the security requirements.

Vulnerability classes: The initial list of vulnerability classes was developed using information from several existing documents and web sites, e.g., NIST SP 800-82, *Guide to Industrial Control Systems Security*, CWE vulnerabilities, and OWASP vulnerabilities list. These vulnerability classes ensure that the security controls address the identified vulnerabilities. The vulnerability classes may also be used by smart grid implementers, e.g., vendors and utilities, in assessing their systems. The vulnerability classes are included in Chapter 6 of [87].

Both bottom-up and top-down approaches were used in implementing the risk assessment of this standard.

The *bottom-up* approach focuses on well-understood problems that need to be addressed, such as authenticating and authorizing users to substation intelligent electronic devices (IEDs), key management for meters, and intrusion detection for power equipment. Also, interdependencies among smart grid domains/systems were considered when evaluating the impacts of a cybersecurity incident. An incident in one infrastructure can potentially cascade to failures in other domains/systems.

Top-down analysis: In the top-down approach, logical interface diagrams were developed for six functional priority areas—Electric Transportation, Electric Storage, Wide Area Situational Awareness, Demand Response, Advanced Metering Infrastructure, and Distribution Grid Management. The report [87] includes a logical reference model for the overall smart grid, with logical interfaces identified for the additional grid functionality. Some examples of the logical interface categories are (1) control systems with high data accuracy and high availability constraints; (2) business-to-business (B2B) connections; (3) interfaces between sensor networks and controls systems; and (4) interface to the customer site.

A set of attributes was defined and the attributes allocated to the interface categories, as appropriate. This logical interface category/attributes matrix is used in assessing the impact of a security compromise on confidentiality, integrity, and availability.

The *level of impact* is denoted as low, moderate, or high. This assessment was done for each logical interface category. The output from this process was used in the selection of security requirements. An example is given in Figure 45, where the interface of between control systems and equipment with high availability and with computation and/or bandwidth constraints (e.g., Transmission SCADA and substation equipment) is depicted. The figure also shows the level of impact, on the top right, and the technical high-level security requirements.

As with any assessment, a realistic analysis of the inadvertent errors, acts of nature, and malicious threats and their applicability to subsequent risk-mitigation strategies is critical to the overall outcome. The smart grid is no different. Table 12 summarizes the categories of adversaries to information systems, which, according to this NIST.IR report need to be considered when performing a risk assessment of a smart grid information system.

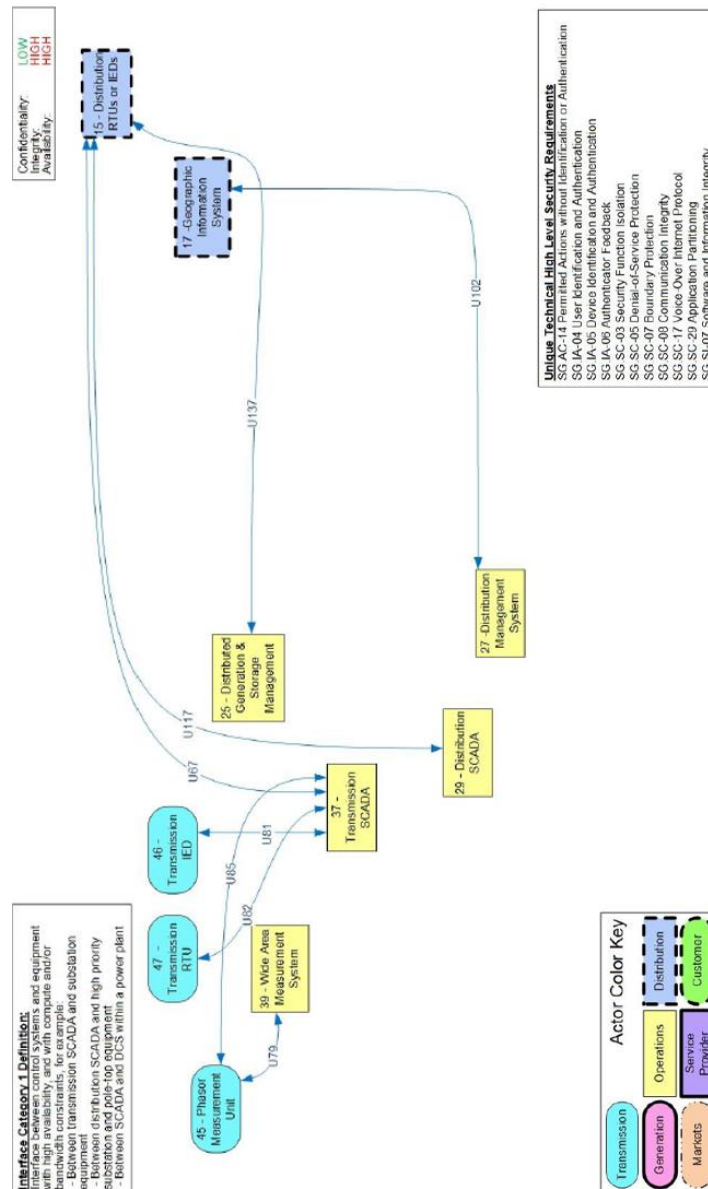


Figure 45 Assessment for the interface between control systems and equipment, including level of impact and security requirements

Table 12 Categories of Adversaries to Information Systems in NIST.IR 7628 (source: [87])

Adversary	Description
Nation States	State-run, well organized and financed. Use foreign service agents to gather classified or critical information from countries viewed as hostile or as having an economic, military or a political advantage.
Hackers	A group of individuals who attack networks and systems seeking to exploit the vulnerabilities in operating systems or other flaws.
Terrorists/ Cyberterrorists	Individuals or groups operating domestically or internationally who represent various terrorist or extremist groups that use violence or the threat of violence to incite fear with the intention of coercing or intimidating governments or societies into succumbing to their demands.
Organized Crime	Coordinated criminal activities including gambling, racketeering, narcotics trafficking, and many others. An organized and well-financed criminal organization.
Other Criminal Elements	Another facet of the criminal community, which is normally not well organized or financed. Normally consists of few individuals, or of one individual acting alone.

Industrial Competitors	Foreign and domestic corporations operating in a competitive market and often engaged in the illegal gathering of information from competitors or foreign governments in the form of corporate espionage.
Disgruntled Employees	Angry, dissatisfied individuals with the potential to inflict harm on the smart grid network or related systems. This can represent an insider threat depending on the current state of the individual's employment and access to the systems.
Careless or Poorly Trained Employees	Those users who, either through lack of training, lack of concern, or lack of attentiveness pose a threat to smart grid systems. This is another example of an insider threat or adversary.

2.4.3.4. ISA/IEC 62443

The ISA/IEC 62443 standard [89] is of international standard for the *security of industrial automation and control systems*. It is actually a group of standards, initiated by the ISA, carried worldwide and is being further developed by the IEC.

The scope of the ISA/IEC 62443 Series is the Security of Industrial Automation and Control Systems (IACS). An IACS is defined as a: *collection of personnel, hardware, software, and policies involved in the operation of the industrial process and that can affect or influence its safe, secure, and reliable operation* [92].

Since IACS includes not only technology, but also people and work processes needed to ensure the safety, integrity, reliability, and security of the control system, the term security here assumes a broader meaning, and, somehow rewrites the security triad, from confidentiality, availability, integrity, (typical for the IT-security) to people, process, technology (for the OT-security) [92].

The security life cycle of the IACS is composed of three phases:

1. **Assessment:** includes activities to identify high-level risks, to carry out vulnerability and low-level risk analyses, to allocate the minimum IT security requirements for each component of System. In detail, this phase includes:
 - Risk Assessment;
 - Vulnerability Assessment;
 - Penetration Test;
 - Threat Modelling;
 - Security Level Allocation.
2. **Implementation:** it is necessary to structure the entire Cyber Security Management System (CSMS) which represents the set of activities needed to identify IT risks and define the related mitigations that make up the security strategy, to protect its own industrial systems. This phase includes: defence Strategy; IT CSMS; Security Level Verification.
3. **Maintenance:** includes maintenance actions that constitute a process of constant monitoring of the security level of components, which allows the transmission of data to be shared safely to the outside. In detail, this phase includes: Control (Auditing); Subsequent Checks (Follow-up).

This standard is arranged in four groups, and each of them is composed of parts [89], [92].

Table 13 shows the complete list of ISA/IEC 62443 standards and technical reports, where document types acronyms, where available, indicate International Standard (IS), Technical Report (TR), and Technical Specification (TS). Part 3-2, presented in bold in the table, is summarized in the following Section.

Table 13 ISA/IEC 62443 [92]

	Part	Type	Title	Date
Overview	1-1	TS	Terminology, Concepts, and Models	2007
	1-2	TR	Master glossary of terms and abbreviations	
	1-3		System cybersecurity conformance metrics	
	1-4		IACS security lifecycle and use cases	
Policies & procedures	2-1	IS	Establishing an IACS security program	2009
	2-2		IACS security program ratings	
	2-3	TR	Patch management in the IACS environment	2015
	2-4	IS	Security program requirements for IACS service providers	2018
	2-5	TR	Implementation guidance for IACS asset owners	
Systems	3-1	TR	Security technologies for IACS	
	3-2	IS	Security risk assessment for system design	2020
	3-3	IS	System security requirements and security levels	2013
Component	4-1	IS	Product security development life-cycle requirements	2018
	4-2	IS	Technical security requirements for IACS component	2019

IACS Risk Management

ISA/IEC 62443-3-2, which is entitled *Security risk assessment for system design* describes the requirements for addressing the cybersecurity risks in an IACS, including the use of Zones and Conduits, and Security Levels. While Part 3-2 includes the requirements for the risk assessment process, it does not specify the exact methodology to be used. The methodology used must be established by the Asset Owner and should be consistent with the overall risk assessment methodology of the organization. Examples using the risk matrix methodology are included as informative content [92]. Figure 46 shows the risk assessment process.

A key step in the Risk Assessment process is to partition the System Under Consideration into separate *Zones* and *Conduits*. The intent is to identify those assets which share common security characteristics in order to establish a set of common security requirements that reduce cybersecurity risk [92].

A Zone is defined as *a grouping of logical or physical assets based upon risk or other criteria such as criticality of assets, operational function, physical or logical location, required access, or responsible organization*.

A Conduit is defined as *a logical grouping of communication channels that share common security requirements connecting two or more zones*.

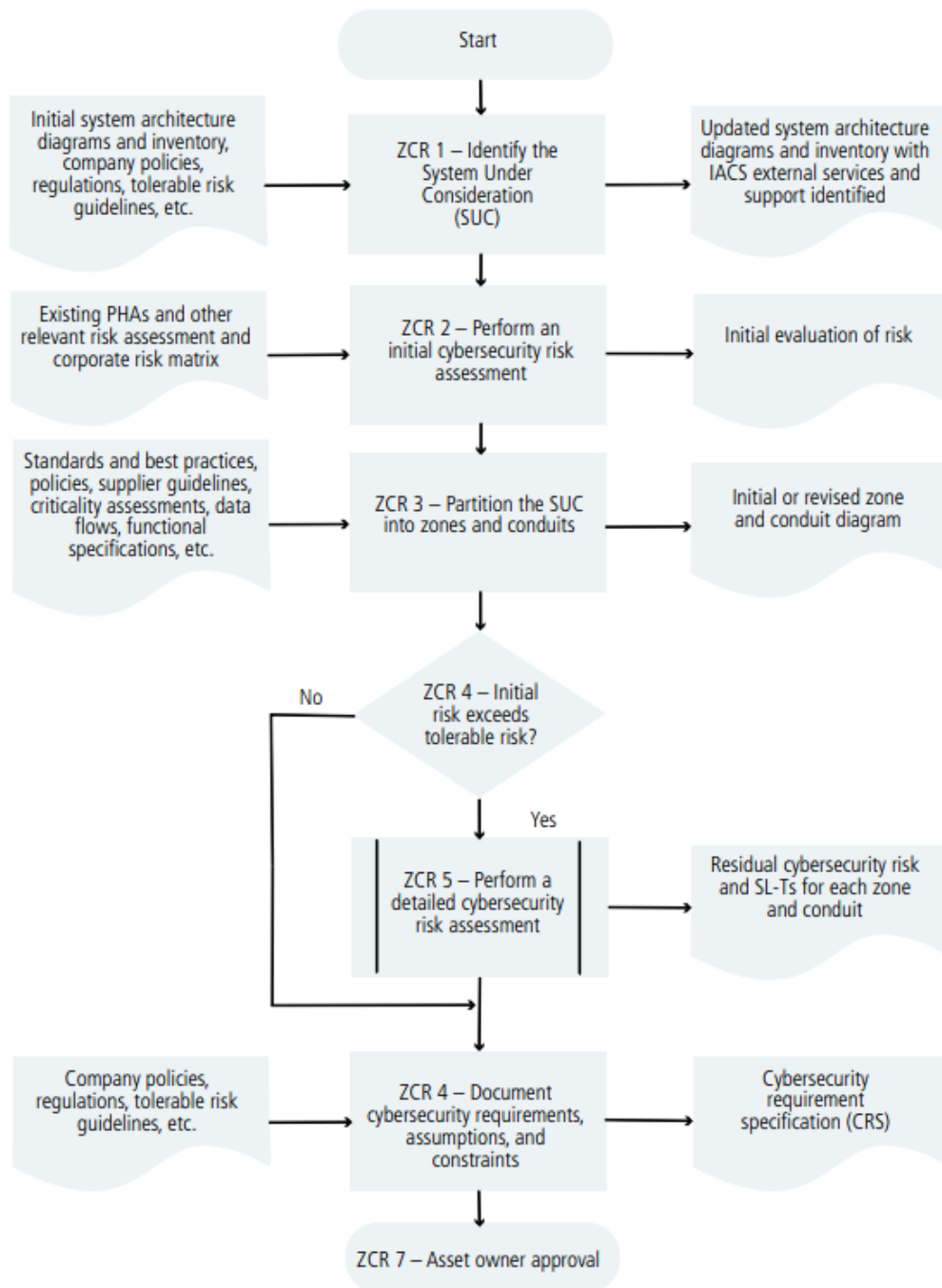


Figure 46 Flow diagram of the ISA 62443-3-2 Risk Assessment Process (source: [92])

Partitioning the System Under Consideration into Zones and Conduits can also reduce overall risk by limiting the scope of a successful cyber-attack. Part 3-2 requires or recommends that some assets are partitioned as follows [92]:

- Shall separate business and control system assets
- Shall separate safety related assets
- Should separate temporarily connected devices
- Should separate wireless devices
- Should separate devices connected via external networks

Part 3-2 also requires that required security countermeasures from the Risk Assessment as well as *security requirements* based on company or facility-specific policies, standards,

and relevant regulations are documented in a Cybersecurity Requirements Specification (CRS). The CRS does not have to be a standalone document; it can be included as a section in other relevant IACS documents. The CRS includes information such as a description of the System Under Consideration, Zone and Conduit drawings, threat environment, and countermeasures from risk assessments [92].

Part 4-1 describes the requirements for the Security Development Lifecycle (SDL) of Control System and Component products. One of the key processes in the product SDL is *threat modelling* which is a systematic process to identify data flows, trust boundaries, attack vectors, and potential threats to the control system. The security issues identified in the threat model must be addressed in the final release of the product and the threat model itself must be periodically updated during the product's lifecycle [92].

Part 3-3 further defines the Security Level in terms of the means, resources, skills, and motivation of the *threat actor*, as shown in Table 14.

Table 14 Security Level and Threat Actors Definition in ISA/IEC 62443-3-3 (source: [92])

Security Level	Definition	Means	Resources	Skills	Motivation
1	Protection against casual or coincidental violation	simple	low	generic	low
2	Protection against intentional violation using simple means with low resources, generic skills, and low motivation	simple	low	generic	low
3	Protection against intentional violation using sophisticated means with moderate resources, IACS-specific skills, and moderate motivation	sophisticated	moderate	IACS-specific	moderate
4	Protection against intentional violation using sophisticated means with extended resources, IACS-specific skills, and high motivation	sophisticated	extended	IACS-specific	high

2.4.3.5. ETSI EG 203 251

The ETSI EG 203 251 V1.1.1 standard methodology for risk assessment [129] combines an extended security assessment derived from ISO 31000 and typical security testing activities following the standard ISO 29119. This methodology was initially developed and evaluated in the RASEN [100] and ARMOUR research projects [101].

The proposal distinguished two main perspectives, a test-based risk security assessment and a risk-based security testing one. In the test-based risk security assessment, testing is used to guide and improve the risk assessment, adjusting risk values and providing feedback, whereas in the risk-based security testing, risk assessment results are used to guide the testing, prioritizing the areas to be tested according to their risk.

The main purpose of integrating the testing process into the risk assessment process is to use testing to extend certain activities of the risk assessment process, thereby improving the overall process results. This can be achieved by ensuring that the test results are used as input for the risk assessment. Risk assessment includes the identification of assets, threats and vulnerabilities, as well as the identification, designation and realization of risk treatment, that is, security control and other countermeasures. Risk itself is a measure that relates the frequency and/or likelihood of accidents to their impact.

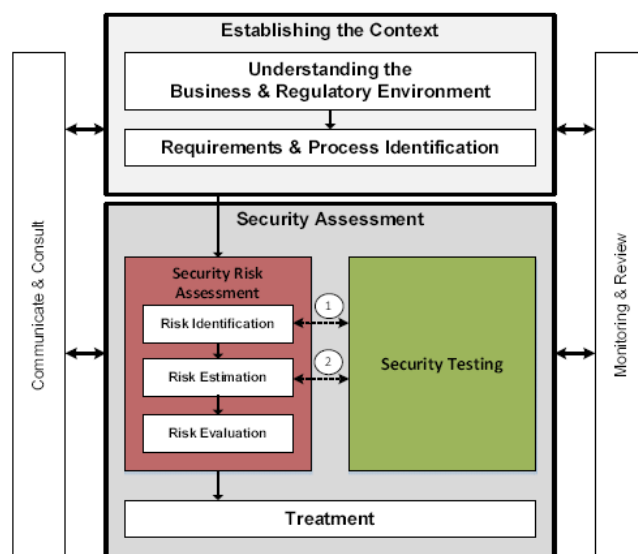


Figure 47 ETSI test-based risk security assessment [129]

The overview of the test-based risk security assessment process is shown in Figure 47. In this process, the risk assessment activity is composed by three activities:

Risk identification is the process of discovering, identifying and describing risks. This involves determining the source of risk (such as threats and vulnerabilities), areas of impact (such as assets), events (including changes in circumstances), causes, and potential consequences. It should disclose the analysis of potential threats or attack surfaces, the identification of potential threats and vulnerabilities, and the derivation of complete threat scenarios, which should cover the relationship between threats, vulnerabilities, and unwanted events. Risk identification can involve historical data, theoretical analysis, informed and expert opinions and the needs of stakeholders.

Risk estimation is the process of determining the level of risk. This involves understanding the nature of the risk, its source and its consequences.

Risk assessment is the process of comparing the results of the risk estimation with risk criteria to determine whether the risk and/or its degree is acceptable or tolerable. Risk assessment helps to make decisions about risk treatment and the most appropriate risk treatment strategies and methods.

These three, together with the "Establishing the Context" and "Treatment" activities, form the core of the ISO 31000 risk management process. As shown in Figure 48, especially in two specific activities, testing can enhance the risk assessment process.

On the one hand, testing can enhance risk identification, as shown in Figure 48. During the risk assessment process, risk identification activities are performed against a target of analysis described in the "Establishing the Context". However, in a test-based risk assessment setting, risk identification is not only based on the documentation of the target, but also on its related test results. Test-based security risk identification can improve security risk identification providing information about the system. Security testing can identify/indicate potential actual vulnerabilities or vulnerable areas of the system.

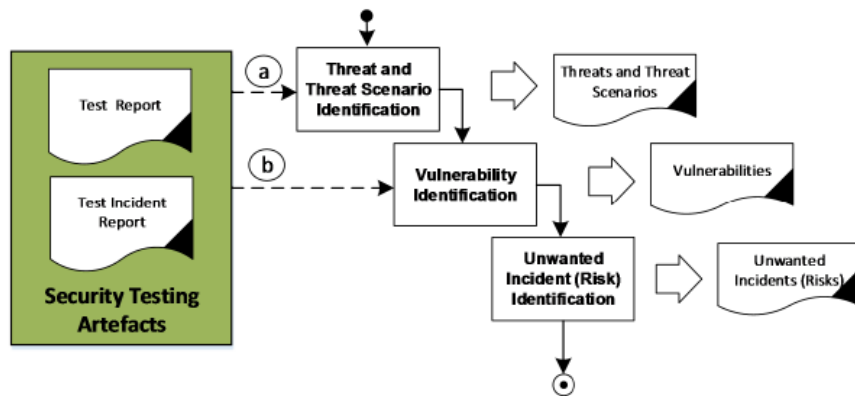


Figure 48 Test based risk identification [129]

On the other hand, testing can enhance the risk estimation activity. As shown in Figure 49, risk estimation activity is comprised by the three sub-activities: Likelihood Estimation, Consequence Estimation, and Estimate Validation. The last sub-activity refers to checking and/or gaining confidence in the correctness of the risk estimates. There are in particular two activities that can be integrated with testing:

- Test-based likelihood estimation
- Test-based estimate validation

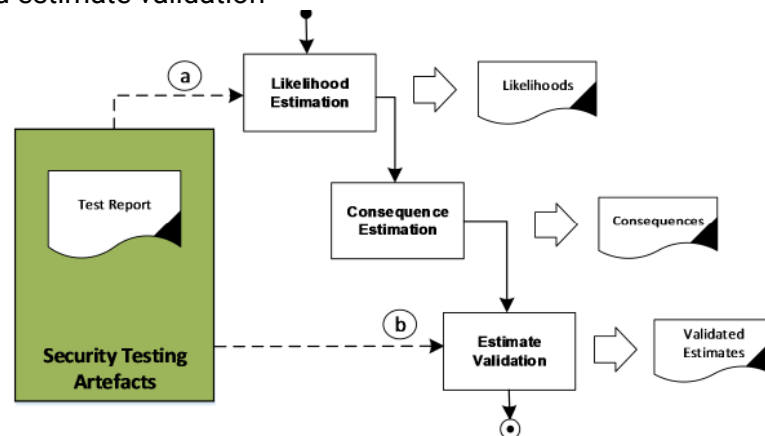


Figure 49 Test-based security risk estimation [129]

Likelihood Estimation is the activity of estimating the likelihood of risks and their causes. In security setting, this involves estimating the following possibilities: a security attack will be launched; an attack will be successful; a successful attack will result in an established risk. In this sense, testing is particularly relevant to obtaining information that can support the estimation of the likelihood of success if an attack is launched. This is because security testing is most commonly used to identify vulnerabilities, and the existence of these vulnerabilities directly affects its likelihood.

The main difference between test-based likelihood estimation and test-based likelihood verification is that in the first one, the test is used to obtain the likelihood first, while in the second activity, the purpose is to verify or gain confidence on the likelihood estimated. In addition, Figure 47 shows other support activities, such as "communication and consult" and "monitoring and review", which are designed to establish a management perspective to continuously control, react and improve all relevant information and results of the process. From a process perspective, these activities are meant to provide context and management-related information for the combined security assessment, and are considered to be the common denominator of the security risk assessment workflow and the test-based risk assessment workflow.

2.4.3.6. Other Standards Considered

The standards presented so far are a good reference for designing and developing a risk assessment methodology and tool (Task 6.1) and for its application to the project use cases (T6.2 and WP8). However, the set of standards analysed is not limited to them: in Table 15 there is the complete list of standards studied in the context of this task. Most of the standards pertains to IT-security domain, while some others address (also) OT-security, safety and privacy.

Table 15 Full list of standards analysed in the context of this activity

Standard	Title
NIST 800-30	Information Security - Guide for Conducting Risk Assessments
ISA/IEC 62443	Security for Industrial Automation and Control Systems
NIST.IR 7628	Guidelines for Smart Grid Cybersecurity (Rev.1)
NIST 800-37	Risk Management Framework for Information Systems and Organizations (A System Life Cycle Approach for Security and Privacy Rev 2)
ISO/IEC 15408-1:2009	Common Criteria for Information Technology Security Evaluation (Part 1: Introduction and general model Rev.5)
ISO/IEC 18045:2008	Common Methodology for Information Technology Security Evaluation
EUCC	EUCC (Common Criteria based European candidate cybersecurity certification scheme)
EN 50129:2018	Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signalling
SAE J3061	Cybersecurity Guidebook for Cyber-Physical Vehicle Systems
ISO 21434	Road vehicles – Cybersecurity engineering
ISO/WD PAS 5112	Road vehicles – Guidelines for auditing cybersecurity engineering (Version 1)
ISO/IEC 27001:2018	Information technology – Security techniques – Information security management systems – Overview and vocabulary (Fifth edition 2018-02)
ISO/IEC TR 19791:2010	Information technology – Security techniques – Security assessment of operational systems (Rev.2)
NIST 800-53	Security and Privacy Controls for Federal Information Systems and Organizations (Rev.4 and Rev.5)
NIST 800-82	Guide to Industrial Control Systems (ICS) Security (Rev.2)
EN 50159:2010	Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems
ISO 26262	Road vehicles – Functional safety
IEC 61508	Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)
ETSI EG 203 251 (ISO 31000)	Methods for Testing & Specification; Risk-based Security Assessment and Testing Methodologies

2.4.4. From Standards to BIECO Risk Assessment Process

After having reviewed the standards and identified some possible overlaps in the phases, a set of nine phases for the risk assessment, from 0 to 8, has been drafted and is shown in Table 16. It integrates the common steps and similarities in the security life cycles of the standards. The phases can be considered an extension of the risk assessment process from NIST SP800-30 described in Table 11.

For step 3 in Table 16 “*Identification of vulnerabilities / hazards / threats*”, it is possible to improve the activity of vulnerability identifications by means of supporting test-based methods described in ETSI EG 203 251. In particular two testing activities can be performed:

- Tests for obtaining information that can support the estimation of the likelihood of success if an attack is launched;
- Tests to verify or gain confidence on the likelihood estimated.

Similarly, for step 4 in Table 16 “*Likelihood determination*” tests in compliance with ETSI EG 203 251 can be executed for improving security threats identification providing information about the system. Security testing can identify/indicate potential vulnerabilities or vulnerable areas of the system.

In Appendix B, we provide more details on the association between steps and descriptions with the reference standards.

Table 16 – BIECO Risk Assessment Process

Step No.	Name	Description
0	Preparation	<p>Identification of purpose, scope, assumptions - constraints, information sources, risk model. Establishing the context, understanding the regulatory environment, requirements and processes identification.</p> <p>Cyber security requirements specification, SUC description.</p> <p>[For IACS context] Produce zone and conduit drawings, identify zone and conduit characteristics, operating environment assumptions, threat environment, organizational security policies, tolerable risk, regulatory requirements</p> <p>For safety-critical context] a safety goal is to be determined for each hazardous event evaluated in the hazard analysis</p>
1	Identification of assets	<p>1: Definition of a list of information assets</p> <p>2: Identification of assets and potential damage resulting from a breach of security features</p> <p>3: SUC (System Under Consideration) identification</p> <p>4: [For IACS context] Partition the SUC into zones and conduits</p>
2	Identification of vulnerabilities /hazards /threats	<p>Identification of threats, attacks and vulnerabilities that apply to each asset</p> <p>[for safety-critical/automotive/railway domain]</p> <p>Identification and description of operational situations and operating modes in which a vehicle may malfunction. Determination and evaluation of potential hazards.</p>
3	Attack path analysis / Impact determination	<p>Identification and linking of potential attack paths to one or more threat scenarios</p> <p>Analysis of threats and vulnerabilities</p> <p>Determination of consequence and impact</p> <p>[for safety-critical/automotive/railway domain]</p> <p>Classification of the identified potential hazards (also) based on the estimation of severity and controllability</p>
4	Likelihood determination	<p>Rating of the feasibility of attack paths based on the ease of exploitation</p> <p>Determination of unmitigated likelihood</p> <p>Analysis of likelihood and associated uncertainty</p> <p>[for safety-critical/automotive/railway domain]</p> <p>Classification of the identified potential hazards (also) based on the estimation of probability of exposure</p> <p>Determination of probability of successful attacks</p>
5	Determination of Risk, uncertainty, target level, and prioritization	<p>Determination of the risk value of a threat scenario</p> <p>Determination of unmitigated cyber security risk, SL-T (Target Security Level), comparison of unmitigated risk with tolerable risk</p> <p>Communication of risk assessment results (e.g., reports, dashboards).</p> <p>[for safety domain]</p>

		<p>Determination of SIL for each hazardous event using the estimation parameters severity, probability of exposure and controllability</p> <p>Production of a list of information security risks that can be prioritized by risk level and used to inform risk response decisions.</p> <p>Provision of uncertainty associated with the risk assessment process</p>
6	Selection of Countermeasures /Mitigations /Controls	<p>Identification of an initial set of controls for the system</p> <p>Tailoring of controls as needed to reduce risk to an acceptable level based on an assessment of risk</p> <p>Addressing identified risks by selecting a suitable risk treatment option</p> <p>Compare unmitigated risk with tolerable risk, identify and evaluate existing countermeasures</p>
7	"Implementation" of Countermeasures /Mitigations /Controls and assessment of effectiveness	<p>Taking of countermeasures until the remaining risk is acceptable</p> <p>Description of how the controls are employed within the system and its environment of operation</p> <p>Assessment of whether the controls are implemented correctly, operating as intended, and producing the desired outcomes with respect to satisfying the [security and privacy] requirements</p> <p>Re-evaluate likelihood and impact, determine residual risk, compare residual risk with tolerable risk, identify additional cyber security countermeasures</p>
8	Maintenance & Communication of assessment results / Monitoring	<p>Documentation and communication of results.</p> <p>Keeping the specific knowledge of the risk organizations current.</p> <p>To support the ongoing review of risk management decisions, maintain risk assessments to incorporate any changes detected through risk monitoring</p> <p>Documenting changes and reporting the [security and privacy] posture of the system</p> <p>Reporting of asset lists, damage scenarios, attack reports or risk reports</p> <p>Comparison of initial risk to tolerable risk</p>

2.5. Modelling of CPSoS

ICT systems and solutions developed by different companies, once integrated in a single system give birth to a so-called System-of-Systems (SoS). SoS are typically deployed on very large geographic scales, comprise a very large number of components, are organized in a hierarchical structure, are driven by complex interactions, and their correct operation and availability is essential. However, the efforts and investments required for their design, implementation and maintenance are enormous. Therefore, new methodologies, principles and reliable tools are needed to manage their evolution and address the growing complexity.

In the traditional modelling environments, large UML [106] or SysML [107] models, which constitutes the widely adopted standards in this field, may become difficult to design and maintain, and often lead to *spaghetti diagrams*, composed of many relationships very complex or even impossible to be visualized and maintained.

Therefore, during the design and modelling of a SoS, many issues have to be faced, as the time required for early prototyping, the cost of modelling large and complex SoS due to their intrinsic complexity, as well as scalability, readability, manageability of the model.

This section reviews some of the main contributions of AMADEOS project [104] which addressed and solved the above challenges and constitutes the starting point on which the solution described in this deliverable is built and which evolves the AMADEOS results, especially for addressing security and risk related concepts.

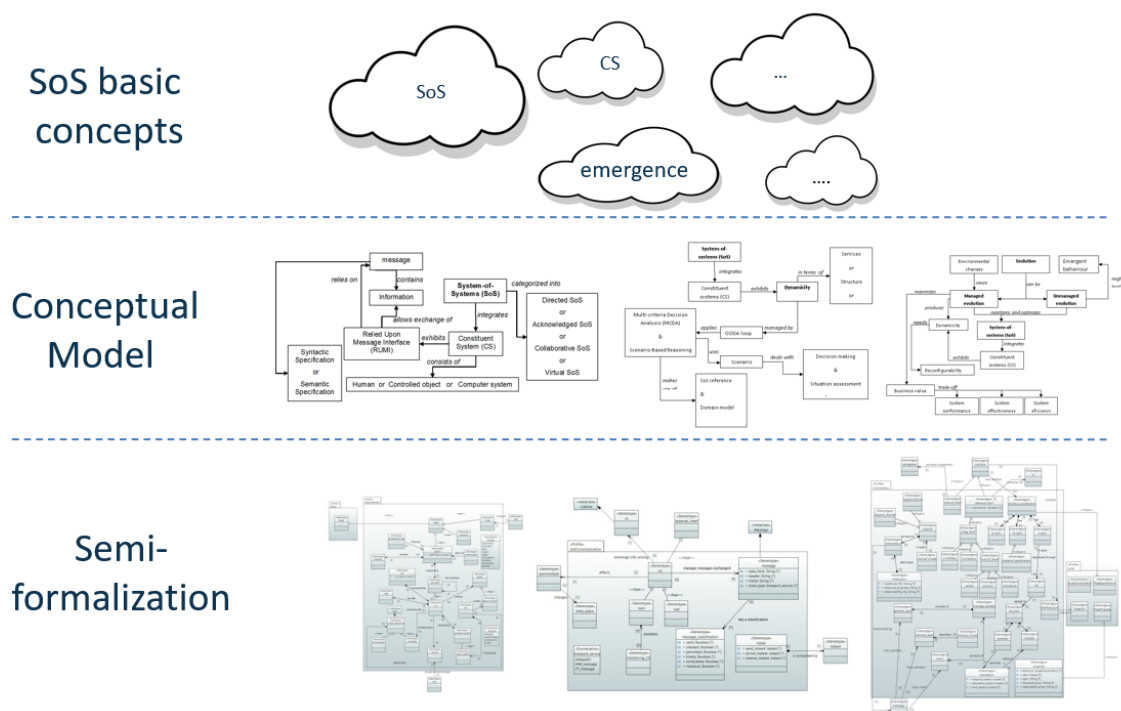


Figure 50 Overview of SoS conceptualization in AMADEOS [104]

AMADEOS collected²¹ and reviewed²² the *SoS basic concepts*, and further described them through:

- a *conceptual model*, thus a high-level graphical representation where concepts and relationships has been grouped in different *Viewpoints*²³ representing the key perspectives of SoS, namely: Structure, Dynamicity, Evolution, Dependability and Security, Time, Emergency and Multi-criticality (described in Section 2.5.2);
- a semi-formal representation, in SysML, of the conceptual model and the viewpoints, which has been organized in a profile composed of several packages; the profile aimed at supporting the understanding and further analysis activities that can be carried out on modelled SoS instances through such a profile (described in Section 2.5.3);
- the development of a supporting facility, called *Blockly4SoS*; leveraging the above concepts and conceptual model, Blockly4SoS has been introduced, and constitutes an important solution for modelling SoS as it reduces the cognitive complexity, introduces an ad-hoc domain-specific SoS profile, provides continuous model validation, includes different model viewpoints, enables the embedded specification of system components behaviour, and automatically generates source code from a model. The tool is further described in Section 2.5.4.

2.5.1. SoS Basic Concepts

In this section, the set of definitions of the relevant concepts for understanding the SoS is provided and some parts of the AMADEOS conceptual model are presented, which constitute the starting point for the design of the methodologies defined and then

²¹ with a detailed analysis of the existing literature for the domain (e.g., from the projects DANSE [123], DSoS [125] and COMPASS [124])

²² overview of the process is in Figure 50

²³ different perspectives, each of which is focused on different SoS concerns **Error! Reference source not found..**

implemented in the context of BIECO WP6 that are presented in this deliverable. Hence, the objective here is to propose a shared vocabulary and define an implicit theory about the SoS domain.

System: *An entity that is capable of interacting with its environment and may be sensitive to the progression of time.*

Universe of Discourse (UoD): *the set of entities and relationships between them that are of interest when modelling the selected world view.*

Environment of a System: *The entities and their actions in the UoD that are not part of a system but have the capability to interact with the system.*

System Boundary: *A dividing line between two systems or between a system and its environment.*

Autonomous System: *A system that can provide its services without guidance by another system.*

System Architecture: *The blueprint of a design that establishes the overall structure, the major building blocks and the interactions among these major building blocks and the environment.*

Subsystem: *A subordinate system that is a part of an encompassing system.*

Constituent System (CS): *An autonomous subsystem of an SoS, consisting of computer systems and possibly of controlled objects and/or human role players that interact to provide a given service.*

Cyber-Physical System (CPS): *A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans.*

An interacting human can be:

- **Prime Mover:** *A human that interacts with the system according to his/her own goal; or*
- **Role Player:** *A human that acts according to a given script during the execution of a system and could be replaced in principle by a cyber-physical system.*

System-of-Systems (SoS): *An SoS is an integration of a finite number of constituent systems (CS) which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.*

The behaviour of a system is of maximum interest to a user, so it is important to define, also:

Function: *A function is a mapping of input data to output data [103]. An alternative definition of function is specification of the intended behaviour of a system.*

Behaviour: *The timed sequence of the effects of input and output actions that can be observed at an interface of a system.*

Service: *The intended behaviour of a system.*

The service specification must specify the intended behaviour of a system.

2.5.1.1. Communication Concepts

In an ICT system, it is essential that the transport of a message from a sender to one or more recipients occurs within a given duration and with *high dependability*, which means

that: within a finite and specified period of time, the message is delivered to the recipients with a high probability, the message is not damaged, and the security of the message (confidentiality, integrity, availability) has not been compromised. So, in a SoS, the communication between the CSs through the exchange of messages is a central mechanism that realizes the integration of the CSs [103].

All the entities involved in the communication, senders and recipients, must share the “rules of the game”, and their interpretation should be unambiguous. These rules are defined as follows.

Communication Protocol: *The set of rules that govern a communication action.*

Message: *A data structure that is formed for the purpose of the timely exchange of information among computer systems.*

A fundamental role for the interaction between the CSs is played by the interfaces, which represent their points of interaction with each other and the environment over time and allow the exchange of information among connected entities. They are defined as follows:

Interaction: *An interaction is an exchange of information at connected interfaces;*

Relied Upon Interfaces (RUI): *interface of a CS whose services are offered to other CSs. A RUI can be of two types:*

- **Relied Upon Message Interface (RUMI):** *interface for the exchange between CS of messages containing information;*
- **Relied Upon Physical Interface (RUPI):** *interface for the exchange of things or energy between CSs.*

In the context of AMADEOS, interfaces have a more detailed and different sub-division (e.g., *Internal, External, Utility Interfaces* and so on); a dedicated viewpoint has been developed, but they are omitted here since not particularly interesting for the risk assessment for the following of the deliverable.

2.5.2. AMADEOS SoS Conceptual Model

AMADEOS represented the main concepts regarding SoS (some of which have been reported above, while many others can be found in [103]) in a conceptual model, with a graphical high-level representation, organized in different viewpoints.

Then, the conceptual model has been semi-formalized in SysML (details in Section 2.5.3), and finally the resulting viewpoints have been imported into Blockly4SoS (described in Section 2.5.4). The conceptual model is available in [103] and is not reported here for the sake of brevity.

2.5.3. AMADEOS SoS SysML Profile

This section focuses on the basic SoS concepts belonging to the different viewpoints and on their semantic relationships, and it describes how the SoS concepts are formally translated using a semi-formal SysML language, organized in a profile composed by viewpoint-related packages [110].

2.5.3.1. Brief introduction to SysML

A semi-formal modelling language is useful for improving the comprehensibility of a problem, because it abstracts a problem thus focusing on particular points of interest through the description of a system using independent visions and levels of abstraction.

It also supports the reduction of development risks and defects through analysis and experimentation processes carried out in the early stages of the design cycle.

The semi-formal language used to describe SoS concepts in AMADEOS is SysML [107] which provides an extension to UML in order to support the modelling and analysis of system-level elements using specific stereotypes (i.e., blocks) and their associations (e.g., generalization). SysML reuses a subset of UML [106] and at the same time extends it: hence it is defined as a UML profile.

2.5.3.2. The AMADEOS Viewpoints

The aspects related to the structure viewpoint have been deeply considered to identify the SoS internal structure, its boundaries with the environment through well-defined interfaces, SoS functionalities and how interactions occur by exchanging messages. The profile diagrams contain the SoS basic concepts distributed in sub-packages [110].

In this document, only four packages are described in detail, since they are the starting point for the developments of evolved and new features in ResilBlockly:

SoS Architecture: describes the basic architectural elements and their semantic relationships;

SoS Communication: provides the basic elements to describe the behaviour of a SoS as a sequence of messages exchanged between CSs;

SoS Security: provides the basic concepts related to SoS security.

SoS Dependability: provides the basic concepts relating to the reliability of SoS.

Instead, other packages are the following (details are in [110]):

SoS Interface: describes all the integration points that allow the exchange of information between the connected components;

SoS Evolution: provides the main elements to describe the gradual and progressive change process of a SoS;

SoS Dynamicity: provides basic concepts relating to the dynamism of a SoS;

SoS Scenario-based reasoning: provides the basic concepts to support the generation, evaluation and management of different scenarios resulting from SoS dynamics, thus supporting the decision-making process in a SoS;

SoS Time: provides the basic elements to describe time concepts;

SoS Multi-Criticality: provide the basic concepts to describe the multi-criticality aspects of a SoS;

SoS Emergence: provides the main elements to describe the SoS emergency concepts.

SoS Architecture, SoS Communication and SoS Interface all together implement the Structure Viewpoint.

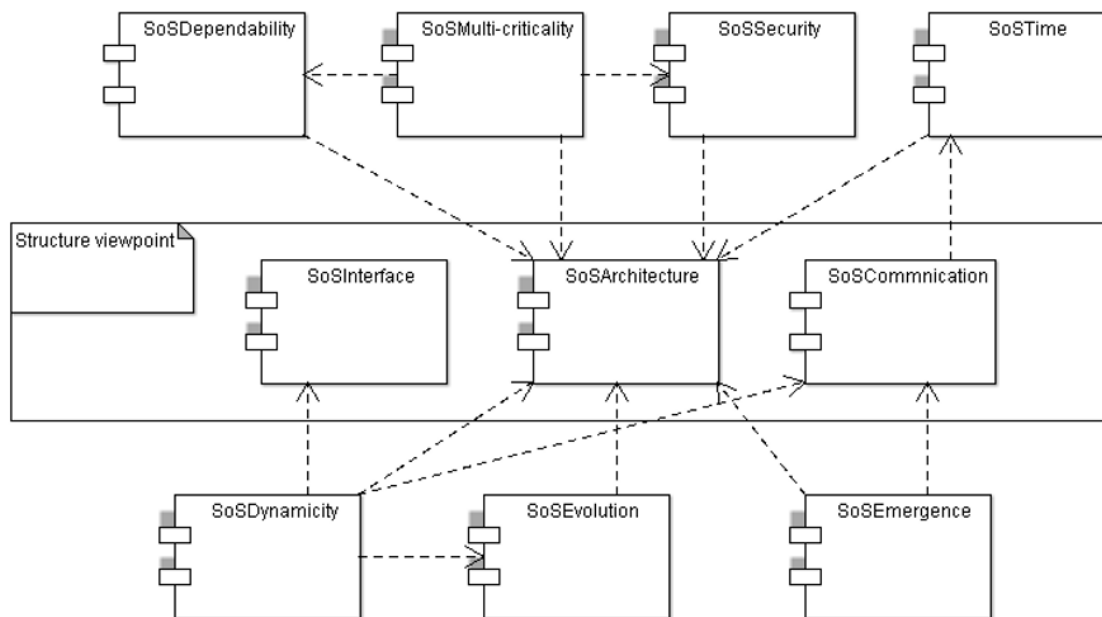


Figure 51 Overview of AMADEOS SysML profile and viewpoint-related packages

2.5.3.3. SoS Architecture Package

This package extends the SysML Block Definition Diagram (BDD) to model the topology and relationships of an SoS. Blocks in SysML BDD are used to represent: systems, system components (hardware and software), elements, conceptual entities and logical abstractions.

Figure 52 represents the static structure of an SoS in terms of its constituent system and relationships and it can represent the topology of any System of Systems (more details are in [103]):

- A *System* is a type of entity (thereby a Block) and is expressed by the *sys_type* Enumeration: autonomous, monolithic, open, closed, legacy, homogeneous, reducible, evolutionary, periodic, stateful and stateless;
- A system can be influenced by an *Architectural style*;
- A system provide communication *Interfaces* and it has a *boundary*;
- A *Subsystem* is a subordinate system that is part of a system and is related to System by a composite relation;
- a Constituent System (CS) is an autonomous subsystem of an SoS;
- CS consisting of human machine interfaces *hmi* and possibly of *physical controlled_object*;
- CS provides a given *Service* by interacting with *role_player* through the RUI (that is introduced in SoS Communication package);
- A *wrapper* to a *legacy_system* and a *prime_mover* are CSs;
- CS extends the property of *System*, which contains multiple *sub_system*, which in turn can be CS;
- System has a *state_space* composed of states described by the variables that may be accessed by the CS service;
- CS interacts with *cyber physical systems*;
- SoS represents the integration of systems;

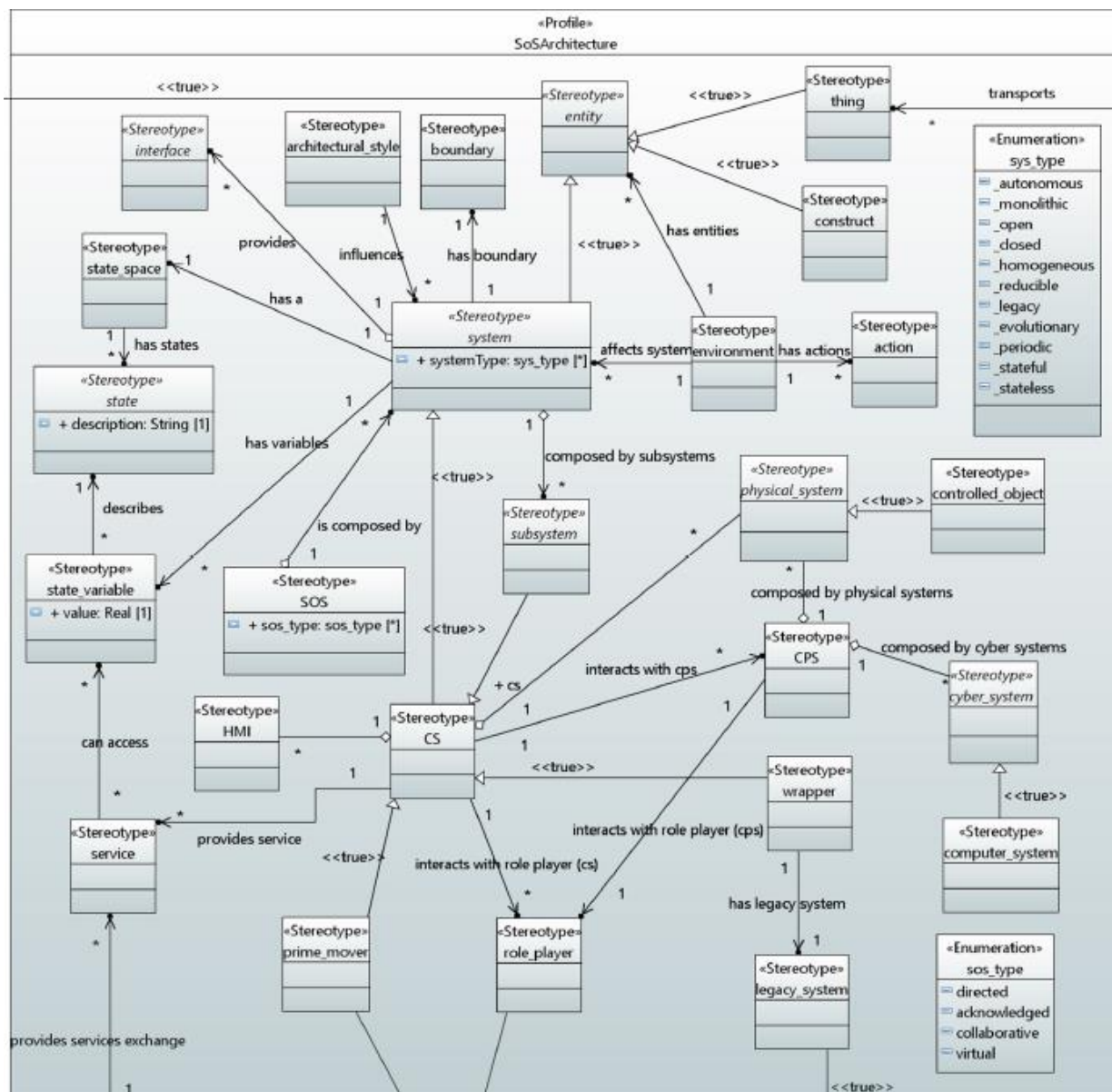


Figure 52 SoS Architecture Package [103]

- An SoS can be directed, acknowledged, collaborative or virtual as it is expressed by the *sos_type* Enumeration;
- Cyber-Physical System (CPS) is composed by a set of *cyber_system* (i.e., computer systems), and *physical_system* (i.e., controlled objects).

2.5.3.4. SoS Communication Package

In a SoS, the communication among the CSs by the exchange of messages is the core mechanism that realizes the integration of CSs.

Figure 53 provides the SysML stereotypes in order to describe the main characteristics of communication protocols among CSs (details are in [103]):

- A *RUI* represents an external interface of a CS where the services of a CS are offered to other CSs;
- *RUI* extends *external_interface* and guarantees the exchange of information among CSs;

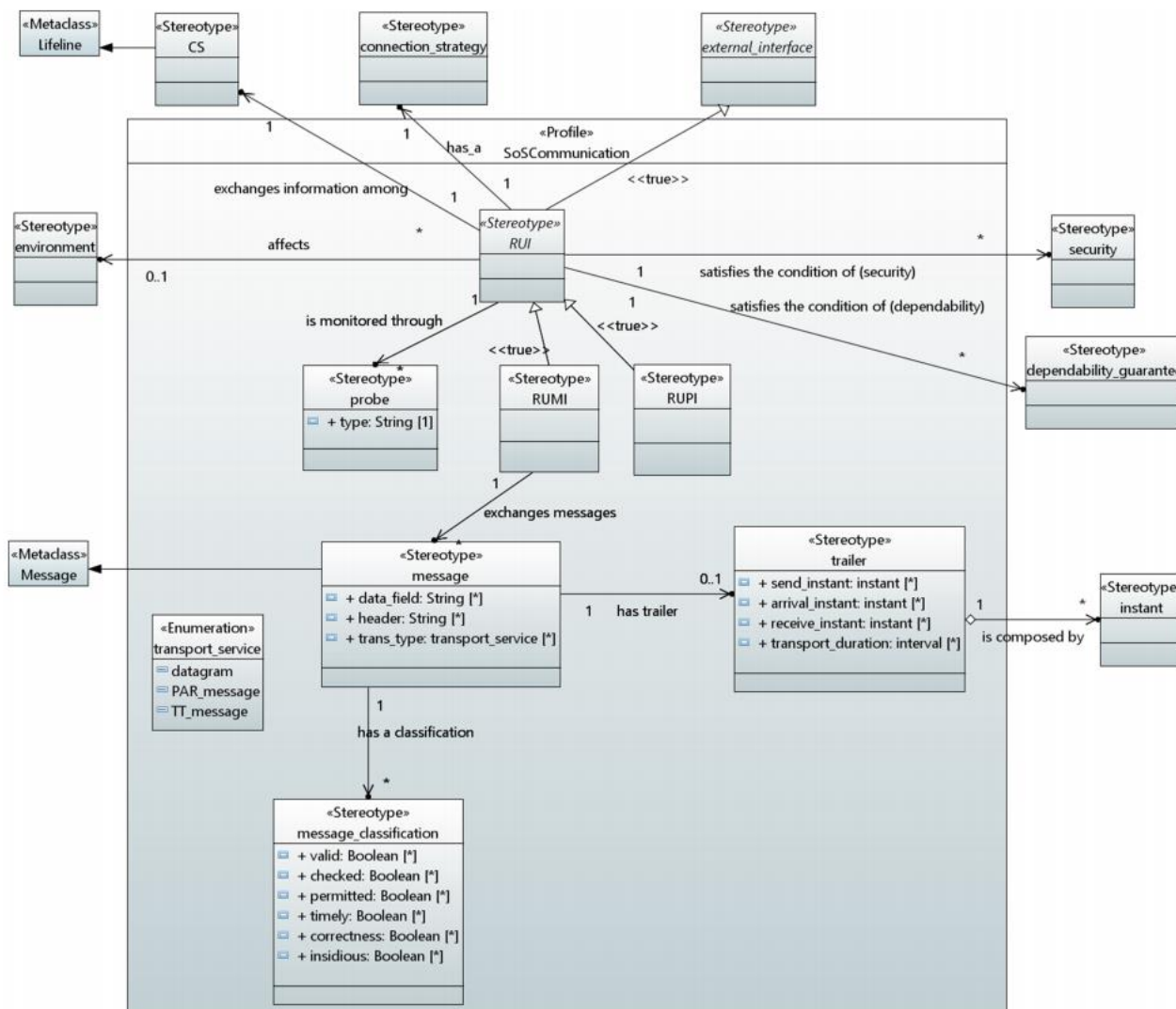


Figure 53 SoS Communication package [103]

- A RUI, can be either a *RUMI* or a *RUPI* and it is monitored through *probes*;
- A RUI, having a *connection_strategy*, is instantiated complying to possibly multiple *dependability_guarantees* and satisfying *security* constraints;
- A *RUMI* represents a message interface for the exchange of information among two or more CSs;
- Physical elements are exchanged among the CSs of an SoS through the *RUPI*;
- Physical elements are *things* or *energy*;
- The *environment* is affected by the RUI;
- A *message* is a data structure that is composed by a *data_field*, a *header* and a *trailer*;
- Message flows through a *transport_service*;
- The main transport protocol classes are listed in the *transport_service* Enumeration data type (i.e., *datagram*, *PAR-Message* and *TT-Message*);
- A message can be classified as valid, checked, permitted, timely, correct or insidious.

2.5.3.5. SoS Dependability Package

Dependability and Security are important properties for SoSs since they affect its availability, reliability, maintainability, safety, data integrity, data privacy and confidentiality.

Figure 54 shows the key concepts (details are in [103]):

- A CS or a whole SoS may require possible multiple *dependability_guarantee*;
- *dependability_guarantee* is achieved through different *dependability measure* by means of possible different *technique*;
- A technique is exploited to reduce the occurrence of faults: *fault_prevention*, *fault_tolerance*, *fault_removal*, *fault_forecast*;
- A *measure* represents a property expected from a dependable system expressed in terms of a quantitative *target_value*: *availability*, *reliability*, *maintainability*, *safety*, *integrity*, *robustness*.

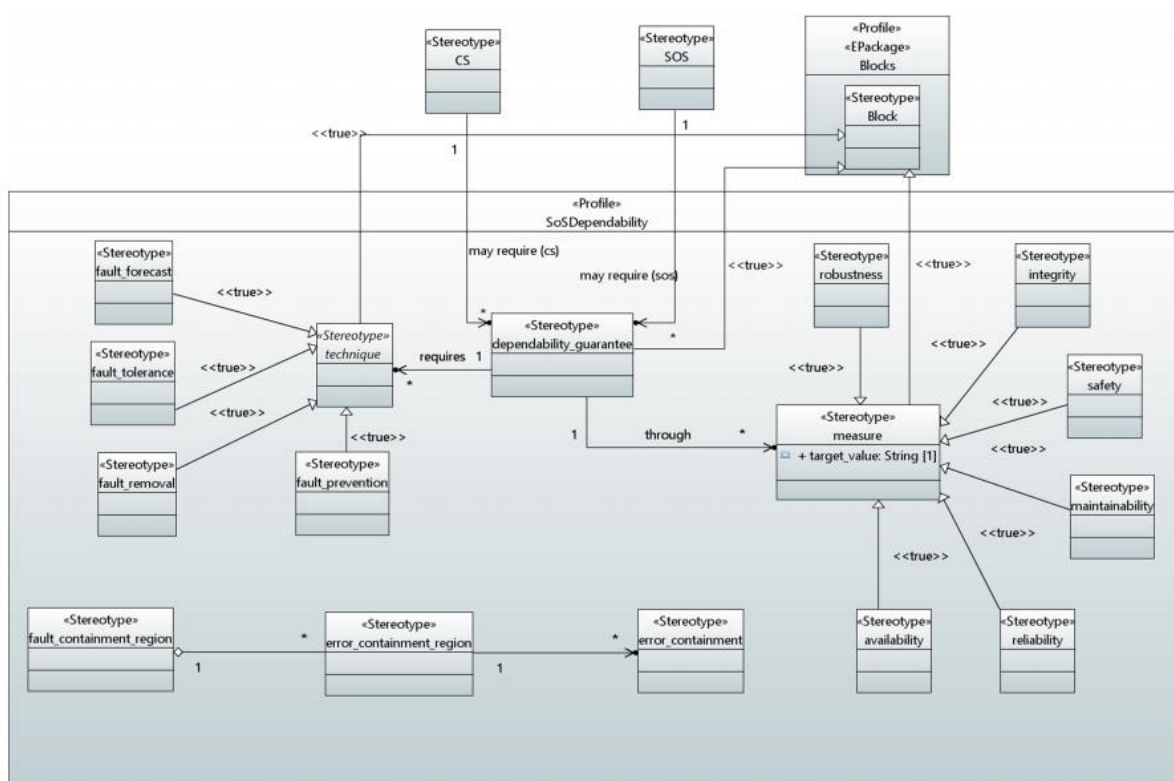


Figure 54 SoS Dependability package [103]

2.5.3.6. SoS Security Package

This package describes the fundamental elements used by a system designer to represent security aspects of an SoS.

Figure 55 shows a set of security concepts (details are in [103]):

- A CS or a SoS are connected to Security Stereotype to satisfy the security conditions of an SoS;
- *Cryptography* based on symmetric (*symmetric_cryptography*) or public key (*public_key_cryptography*) infrastructure;
- *Encryption* represents the process of encoding information or data;

- Three types of keys have been represented: *symmetric_key*, *private_key* or *public_key*;
- Data (or the information exchanged) can be encrypted (*ciphertext*), or not encrypted (*plaintext*)
- *Decryption* represents the process of turning ciphertext to plaintext;
- *AccessControl* consists in a set of actions that are permitted or not allowed by the system;
- *Subject* that represents an active user, a process or a device that causes information to flow among objects or changes the system state;
- A subject may have attributes *permission* that describe how the subject can access to objects;

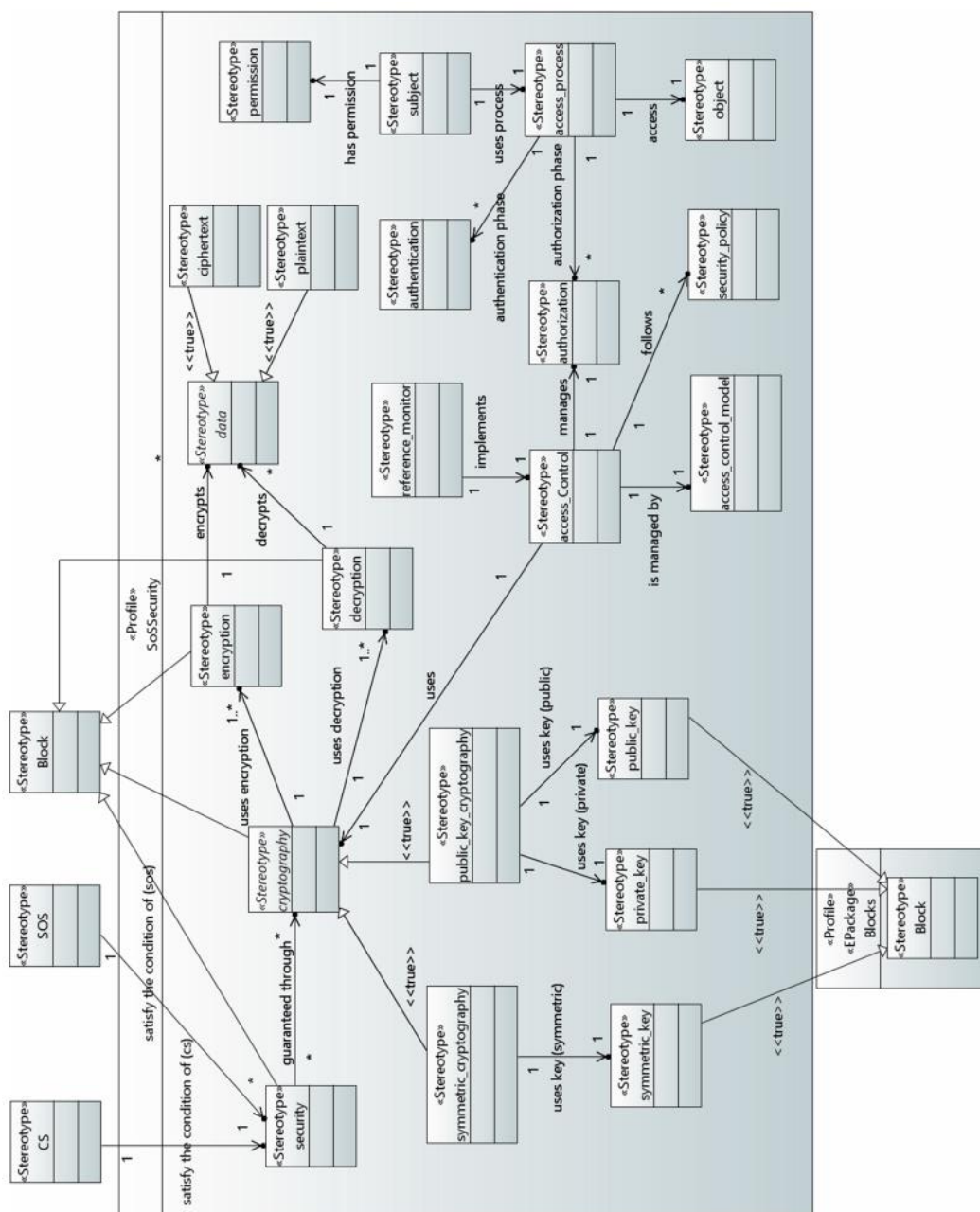


Figure 55 SoS Security package [103]

- An *object* is a passive system-related devices, files, records, tables, processes, programs, or domain containing or receiving information;
- *AccessProcess* is composed by the *authentication* and the *authorization*;
- *ReferenceMonitor* represents the mechanism that implements the access control model;
- *AccessControlModel* captures the set of allowed actions within a system as a policy;
- *SecurityPolicy* represents a set of rules that are used by the system to determine whether a given subject can be permitted to gain access to a specific object.

2.5.4. Blockly4SoS

The AMADEOS SoS profile [110] can be adopted to support a Model-Driven Engineering (MDE) activity. MDE is an approach for system development which leverages models for the understanding, design, construction, deployment, operation, maintenance and modification of systems.

Defining a SoS profile can be difficult for individuals inexperienced with SysML language. Moreover, there can be scalability and readability issues when the complexity of the SoS to be modelled increases. Designers should be able to realize a SoS without having specific and strong skills in modelling languages such as UML or SysML. Furthermore, in traditional modelling environments, large models are known to be difficult to design and maintain and often lead to *spaghetti diagrams*, which is a very complex diagram that leads to a worst visualization of the model, with the use of many relationships (lines) between the intersecting blocks, making it illegible [103].

Blockly4SoS²⁴ is used to model, validate, query, and simulate SoSs by leveraging the integrated Google Blockly²⁵, which is an open-source JavaScript library for building visual programming editors or a visual Domain Specific Language (DSL), using blocks. Blockly4SoS is a valid alternative that facilitates the design of a SoS, thanks to the use of an intuitive interface and in any case compatibility with the MDE approach. In fact, its modelling syntax is defined by a set of blocks available as a puzzle, which are composed to model a SoS.

Blockly4SoS is thus a *supporting facility tool* for SoS designers which enables them to perform MDE leveraging the AMADEOS conceptual model [103] and the integrated Blockly library. The meta-model in Blockly is represented by all the rules and specifications that define the meaning of the various blocks, the relationships that can be created between blocks and all the constraints that allow them to join or not. In particular, Blockly has two ways to define blocks, the JSON and the JavaScript format. The goal of the tool is to simplify and provide means to rapid modelling of SoS using the SysML profile (meta-model).

The SysML meta-model is first transformed into Blockly blocks, then these blocks could be used within the tool to create a SoS model.

2.5.4.1. Overview of Blockly4SoS Flow, Pros and Cons

The main features of Blockly4SoS are [103]:

- It speeds up the modelling and only a modern web browser is required;

²⁴ <http://blockly4sos.resiltech.com>.

²⁵ <https://developers.google.com/blockly/>.

- Its intuitive and simpler user interface than SySML Language;
- the ability to check constraints at design time (user defined and pre-defined constraints) and to warn user when they make mistakes or violations.

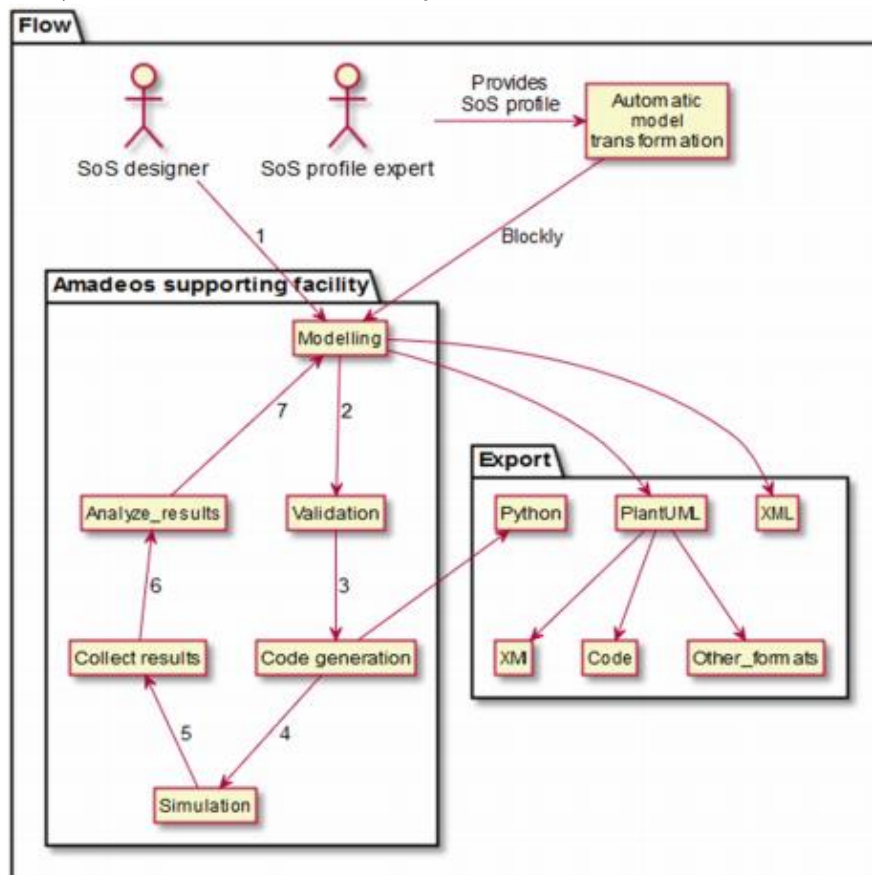


Figure 56 Flow of MDE using the Blockly4SoS [103]

Figure 56 shows the flow and outputs of MDE activity with Blockly4SoS [103]. A preliminary step is carried out by a *SoS profile expert*, which transforms the AMADEOS SoS SysML profile into Blockly blocks; in the case of Blockly4SoS is a one-time activity that has been performed by using an external tool²⁶.

Then, a SoS designer is able to performs modelling with Blockly4SoS as follows:

1. the user is provided with a facility for modelling the SoS, where only compatible blocks can be connected, implementing a validation functionality: the model can be transformed into PlantUML²⁷ or exported as an XML²⁸
2. in addition, the tool enables the coding of behaviour of components directly within the modelling environment (in Python Language), and this source code can be later on exported from the model;
3. The exported code allows the simulation of the components behaviour and of their interactions; simulated constituent systems can also be complemented with real implementations;
4. Simulation results are sequence diagrams and log files which can be analysed in order to refine and update the model.

²⁶ <https://www.eclipse.org/papyrus/>

²⁷ <http://plantuml.com>

²⁸ a specific Blockly version of XML

Together with its features, some cons of Blockly4SoS have been identified: Blockly4SoS only supports coding only in python and only in a window within the modelling environment, so without the features usually available to a programmer when coding in an IDE. Then, the generated XML is possible only in the XML proprietary version of Google Blockly, that can only be reimported within Blockly4SoS itself. Then, Blockly4SoS allows only modelling according to the AMADEOS SysML profile, and no other meta-models are available; potential modifications to the profile have to be performed outside the tool. This could be a problem if the user wants to create an ad-hoc profile for a different domain or to modify some details.

A description of the improvements to Blockly4SoS introduced in the context of BIECO is given in Section 6.1

2.5.4.2. Introduction to Blockly4SoS Main Features

In Blockly4SoS, all the blocks required to build a SoS can be found in the toolbox on left hand side as they have been provided by the SoS profile expert and according to the AMADEOS SysML profile [110]. As example, Figure 57 shows all the blocks related to the Architecture Viewpoint.



Figure 57 – Architecture viewpoint related blocks in Blockly4SoS [111]

The model, in order to be usable in a simulation, must have service blocks with a coded behaviour. So, after having defined a CS and its RUMI, in order to provide a defined service through a RUMI, the user clicks on the plus symbol, located on the right of *Provides exchange of – Service (S)* in a RUMI block, and select from the dropdown menu the service (s), as shown Figure 58:

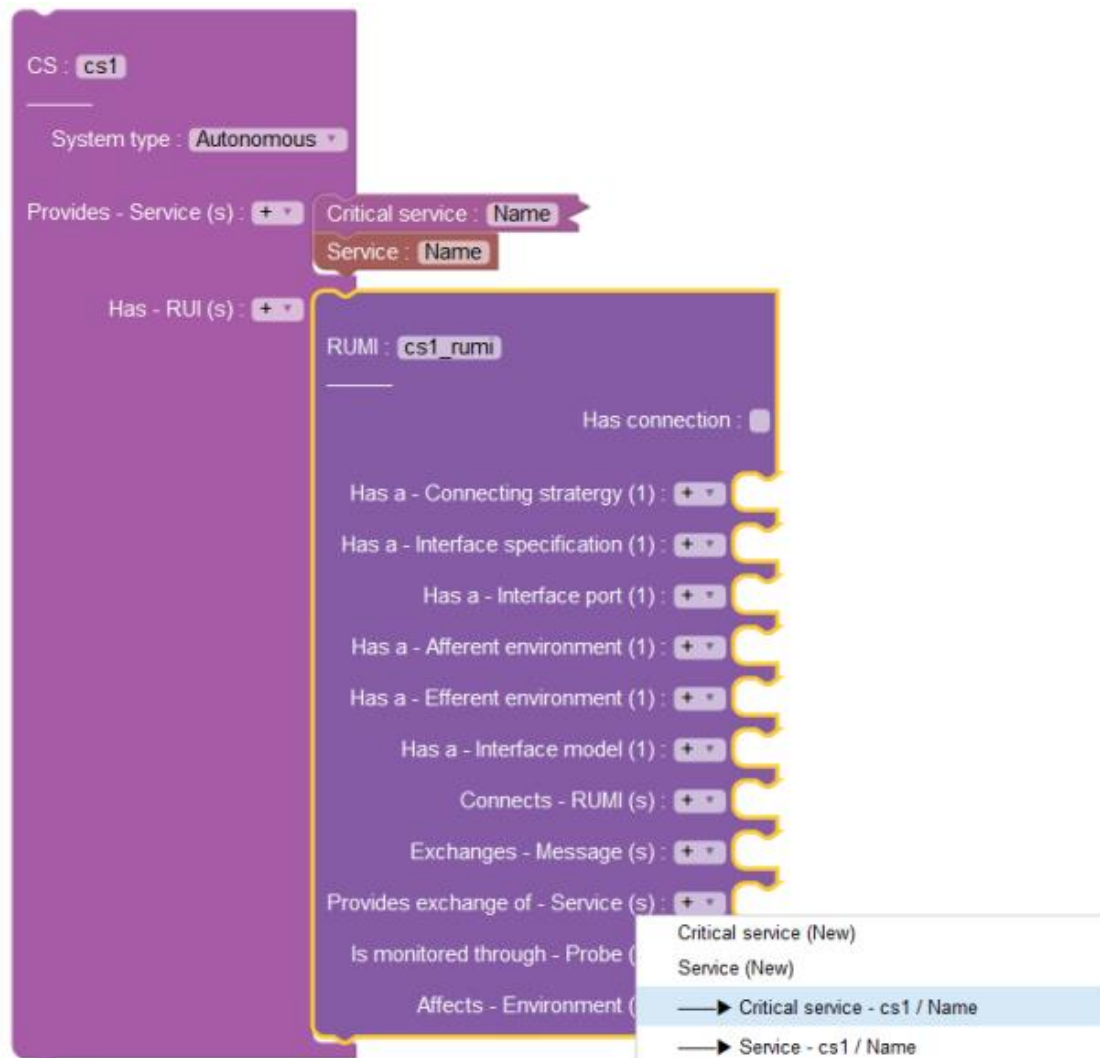


Figure 58 Providing services through a RUMI [111]

To add a behaviour, the user can right click on the interested *Service* block, and to select the *@Add behavior* item; by doing so, a blue icon button appears on the left of the selected service. The designer can click on the icon, opening a text box, in which can insert the python code to define the behaviour of the Service, as shown in the example of Figure 59.

To generate the code related a SoS model, the designer must click on the button *Generate code* and an archive, named *SoS-Simulation.zip*, is generated.

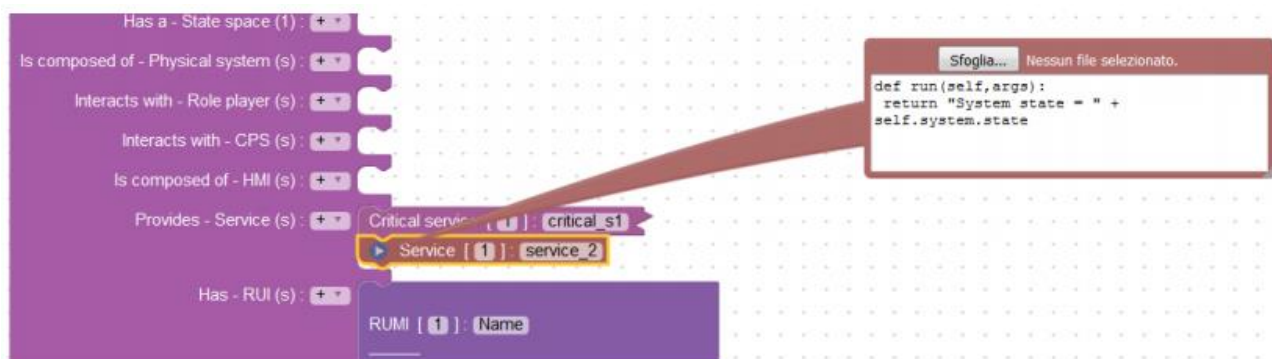


Figure 59 Example of behaviour of a service [111]

The file includes:

1. a folder *src* that contains the Python source code:
 - *amadeos.py* (contains all the constructor code);
 - *sos.py* (the main code for running SoS simulation);
 - *sos_gui.py* (which interacts with the user);
 - *model_behaviour.py* (which contains all the behaviours defined by user).
2. two files for running the simulation on UNIX or windows, respectively.

The simulator is a set of Python programs meant for executing the desired scenarios created by designer. At the end of the simulation, a log file is generated, from which quality metrics may be computed.

Other features have been introduced out of the context of AMADEOS project, one of which is enabled by clicking on the *generate analysis* button: this, even if in a draft version, supports a functional and interface analysis and generates a template for the analysis directly retrieving functions and RUMIs from the model. This is one of the features that have been redesigned and are described in the following of the deliverable (Section 3).

3. Definition of a HAZOP-based Risk Assessment Methodology

The HAZOP technique (described in Section 2.1.13), consists in examining a system by dividing it into *parts* and analysing the potential hazards matched to a system *part* (Figure 32) with the help of guidewords. This Section introduces a methodology, derived from the HAZOP, for a systematic application of an HAZOP-based risk assessment to specific parts of the system model: functions and interfaces, which will be presented in Sections, 3.1 and 3.2 respectively.

In the context of BIECO, the methodology described here in Section 3 (overview of the process is given in Figure 60), together with the approach introduced in Section 4, constitute a basic but effective strategy towards the integration of safety and security analyses, where results from the HAZOP-based risk assessment (typically applied in the safety domain) can be fed into the security risk assessment. In fact, as an example, hazards or low-level component failure that are safety-relevant and that have been identified during the HAZOP, can be further analysed during the security risk assessment from a different point of view.

In other words, this methodology is being proposed and applied as an alternative or preliminary risk assessment approach with regard to the *BIECO Risk Assessment Process* whose steps have been identified in Section 2.4.4, and whose details are given in Section 4. Therefore, a user which has a safety background may feel more comfortable in following the HAZOP approach, while another may prefer to adopt the process originating from the integration of common steps of security standards, and a third user may instead be interested in applying both the methodologies to analyse the same ecosystem from different perspectives.

Moreover, even if the HAZOP methodology is typically used in safety domain for hazard analysis, it is possible to use this approach for conducting also security (threat) analysis and risk assessment, e.g., by using some specific keywords, as discussed in Section 3.3. In this case the two methodologies may be seen as alternative each other. The HAZOP-based methodology is the underlying approach of a feature already existing in Blockly4SoS²⁹ before BIECO, even if in a draft status, introduced after and out of the context of AMADEOS project. In BIECO, the functionality has been re-designed to enable the application also to profiles different from the SoS one, and its implementation completely refactored (description of the latter is given in Section 6.3).

Thus, the contribution of this section is a novel methodology for systematically applying the state-of-the-art HAZOP technique and some typical standard guidewords (Table 4) to the functions and interfaces of a modelled system (e.g., Table 17 and Table 19). Then, leveraging a template for the risk analysis (e.g., Table 18 and Table 20), the methodology originates a HAZOP report. This methodology is practically assisted by the implementation described in 6.3, thanks to which a user automatically obtains the list of functions and interfaces identified from the modelled ecosystem, the automatic generation of a pre-filled HAZOP/THROP report (as depicted in Figure 60). The latter is a downloadable worksheet that can be completed offline originating an actual assessment report. Moreover, the user can specify custom keywords and parsing rules for determining their meaning for the functional/analysis (details in Section 6.3).

²⁹ This feature of Blockly4SoS, as already described in Section 2.5.4.2, is enabled by clicking on the *generate analysis* button

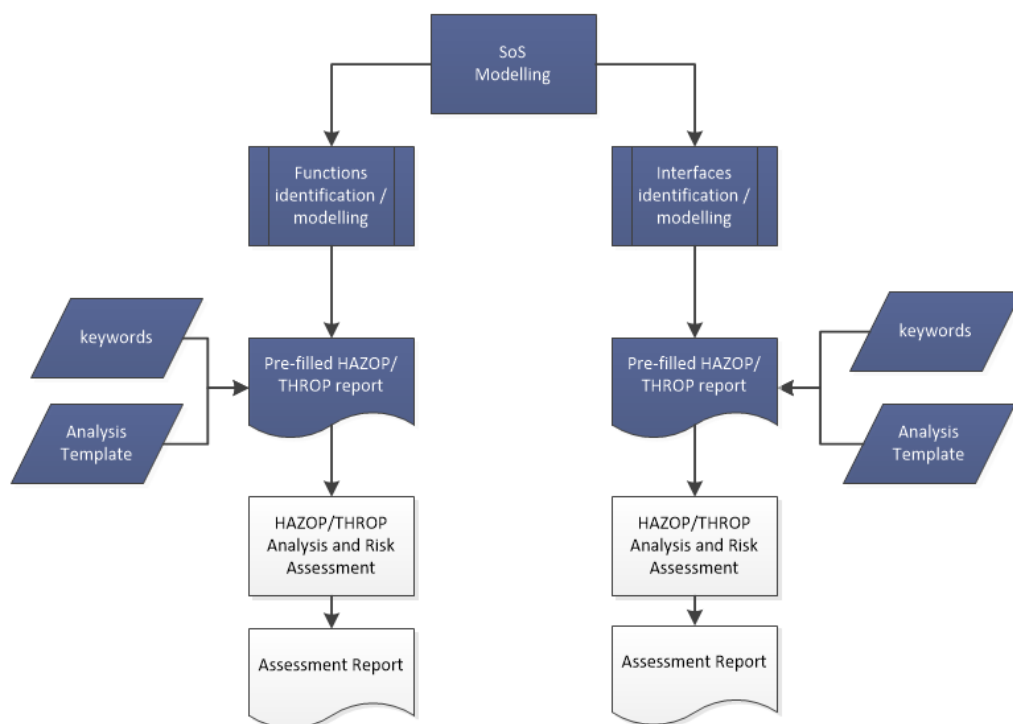


Figure 60 Process view of the HAZOP-based methodology (in blue the steps assisted by ResilBlockly, in white the ones to be addressed offline)

3.1. Functional Hazard Analysis

The *Functions* are a *part* of the model that are systematically analysed and examined, based on guidewords. An example of guidewords is derived from the HAZOP technique, as shown in Table 17, which is the application of the guidewords proposed by IEC Standard 61882 (Hazard and operability studies - Application guide), already presented in Table 4, to the functions and the meaning is updated accordingly.

Table 17 Possible HAZOP Keywords and their meaning for the Functional Analysis

Keyword	Meaning for the Functional Analysis
NO OR NOT	Complete negation of the function outcome
MORE	Quantitative increase in function outcome
LESS	Quantitative decrease in function outcome
AS WELL AS	Qualitative modification/increase in function outcome
PART OF	Qualitative modification/decrease in function outcome
REVERSE	Logical opposite of the function outcome
OTHER THAN/ INSTEAD	Complete substitution in function outcome
EARLY	Function outcome anticipates the intended clock time
LATE	Function outcome is given after the intended clock time
BEFORE	Function outcome is produced before than expected with respect to the order or sequence of events
AFTER	Function outcome is produced after than expected with respect to the order or sequence of events

The analysis can be customized by specifying the list of keywords and the consequent type of deviation from the intended behaviour.

Once the keywords are determined, they are systematically applied to each function, thus generating a table where the number of rows is the product of the number of functions and the number of keywords. On the columns, instead, there are the dimensions of the

HAZOP analysis as represented in Table 18; the first five (*Analysis ID*, *Block*, *Function description*, *Keyword*, *High level description of the scenario to be analysed*) can be filled automatically, without the user intervention, by retrieving the set of functions and related information from the model, and by applying specific parsing and substitution rules. The rules can be also defined by the user as a template (implementation details and examples are in Section 6.3.).

Table 18 Columns in the HAZOP Functional Analysis Template

Column in the template	Meaning for the Functional Analysis
<i>Analysis ID</i>	Unique Identifier Number of a system <i>Function</i> (typically, a <i>relation</i> block that is identified as <i>Function</i>)
<i>Block</i>	Name of the block (e.g., a system component providing the <i>Function</i>) as defined in the model
<i>Function description</i>	Name of <i>Function</i> as defined in the model
<i>Keyword</i>	The keyword that is being applied for the analysis (e.g., one of the guidewords of Table 17)
<i>High level description of the scenario to be analysed</i>	The description of the unexpected behaviour of the function (e.g., according to the meaning of functional analysis, in second column of Table 17)
<i>Causes</i>	Possible causes of the deviation from expected behaviour of the function
<i>Consequences (Local Level)</i>	Impact of the deviation at the local level (if applicable e.g., the function is provided by a subsystem or component)
<i>Consequences (System Level)</i>	Impact of the deviation at the system level
<i>Severity (Pre-Mitigation)</i>	Severity of the impact of the deviation (without considering the introduction of new mitigations)
<i>Probability/Frequency (Pre-Mitigation)</i>	Likelihood of the deviation (without new mitigations in place)
<i>Risk (Pre-Mitigation)</i>	Risk of the deviation (determined considering the above severity and probability and without new mitigations in place)
<i>Mitigation</i>	Possible countermeasure or safeguard to be introduced
<i>Severity (Post-Mitigation)</i>	Updated severity of the impact, considering the mitigation introduced
<i>Probability/Frequency (Post-Mitigation)</i>	Likelihood of the deviation, considering the mitigation introduced
<i>Risk (Post-Mitigation)</i>	Risk of the deviation, considering the updated severity and probability after the introduction of the mitigation

<i>Status</i>	The status of the hazard; e.g., it can assume a value based on categories depending on the system, the domain, or standards (open, pending verification, closed, deleted, covered, etc.)
<i>Note</i>	A field that can be used for commenting the analysis

The other columns in the table (from Causes to Note) require the user intervention and expert analysis in order to be filled.

3.2. Interface Hazard Analysis

Analogously to the Functions, the *Interfaces* are another part of the model that can be analysed for hazards in a systematic way, based on keywords.

An example of keywords for the interface hazard analysis, again derived from the HAZOP technique, is shown in Table 19, where the meaning of each keyword is also explained.

Table 19 Possible HAZOP Keywords and their meaning for the Interface Analysis

Keyword	Meaning for the Interface Analysis
NOT	Complete negation of the transmission over an interface
PART OF	Qualitative modification/decrease in the object transmitted
EARLY	Transmission over an interface anticipates the intended clock time
LATE	Transmission over an interface happens after the intended clock time
BEFORE	Transmission over an interface happens before than expected with respect to the order or sequence of events
AFTER	Transmission over an interface is produced after than expected with respect to the order or sequence of events

As with the functions, the analysis can be customized by specifying the list of keywords and the consequent type of deviation from the intended communication or transmission over an interface.

Similarly, once the keywords are established, the systematic application to each interface generates a new table. On the columns, there are the dimensions of the HAZOP Interface analysis as represented in Table 20; the first six (*Analysis ID*, *Message*, *Source Block*, *Destination Block*, *Keyword*, *High level description of the scenario to be analysed*) can be automatically filled by retrieving from the model the set of interfaces and related information, and by applying specific parsing and substitution rules. Once again, the rules can be specified according to templates, e.g., depending on application domain, standards or specific system details (examples are in Section 6.3.).

Table 20 Columns in the HAZOP Interface Analysis Template

Column in the template	Meaning for the Functional Analysis
<i>Analysis ID</i>	Unique Identifier Number of a system Interface (typically, a triple of <i>relation</i> blocks that is identified as <i>Interface</i> , composed of source, destination and "message")
<i>Message</i>	Name of the "message" block as defined in the model; in principle, it could
<i>Source Block</i>	Name of the block (e.g., a system component) that, leveraging the interface, originates the transmission of a message or "thing",

<i>Destination Block</i>	Name of the block (e.g., a system component) that, leveraging the interface, receives a transmitted message or “thing”
<i>Keyword</i>	The keyword that is being applied for the analysis (e.g., one of the guidewords of Table 19)
<i>High level description of the scenario to be analysed</i>	The description of the unexpected behaviour involving an interface (e.g., according to the meaning of interface analysis, in second column of Table 19)
<i>Causes</i>	Possible causes of the deviation from expected behaviour on the interface
<i>Consequences (Local Level)</i>	Impact of the deviation at the local level (if applicable e.g., it is the interface of a subsystem or component)
<i>Consequences (System Level)</i>	Impact of the deviation at the system level
<i>Severity (Pre-Mitigation)</i>	Severity of the impact of the deviation (without considering the introduction of new mitigations)
<i>Probability/Frequency (Pre-Mitigation)</i>	Likelihood of the deviation (without new mitigations in place)
<i>Risk (Pre-Mitigation)</i>	Risk of the deviation (determined considering the above severity and probability and without new mitigations in place)
<i>Mitigation</i>	Possible countermeasure or safeguard to be introduced
<i>Severity (Post-Mitigation)</i>	Updated severity of the impact, considering the mitigation introduced
<i>Probability/Frequency (Post-Mitigation)</i>	Likelihood of the deviation, considering the mitigation introduced
<i>Risk (Post-Mitigation)</i>	Risk of the deviation, considering the updated severity and probability after the introduction of the mitigation
<i>Status</i>	The status of the hazard; e.g., it can assume a value based on categories depending on the system, the domain, or standards (open, pending verification, closed, deleted, covered, etc.)
<i>Note</i>	A field that can be used for commenting the analysis

3.3. THROP: HAZOP for Security Assessment

Along with typical HAZOP guidewords, one interesting variant that has been proposed in literature is the so-called Threat and Operability Analysis (THROP), which considers threats for a particular feature from a functional perspective. Thus, the THROP first identifies the primary functions of a feature, second applies guidewords to identify potential security threats, and third determines potential worst-case scenario outcomes from the potential malicious behavior [117]. The same approach can be adopted also for interfaces.

Adopting this methodology, and with the help of additional specific guidewords, it is thus possible to extend our HAZOP-based risk assessment methodology for identifying additional threats, and then conducting security analysis and risk assessment.

Some examples of THROP keywords that we identify are given in Table 21, but many other can be specified by the assessor.

Table 21 Additional Keywords identified for the THROP functional and interface security analysis

Keyword	Meaning for the Functional Analysis
DENIAL	The function is not carried out or not executed.
Keyword	Meaning for the Interface Analysis
CORRUPTED	The message arrives corrupted/modified
REDIRECTED	The message is redirected to a wrong/different recipient

Since the implementation described in Section 6.3 enables the user to define custom keywords and rule for determining their meaning for the functional/analysis, the THROP can be realized as well.

4. Definition of a Threat Modelling and Security Risk Assessment Methodology

As described in Section 2.4, the reference security standards for threat analysis and risk assessment have been analysed, and the analysis has driven the definition of an BIECO Risk Assessment Process, which steps (from 0 to 8) listed in Table 16, are depicted also in Figure 61 for reader's convenience. The process integrates the common steps and similarities in the security life cycles of the standards, with a fundamental contribution from the NIST SP 800-30 [43].

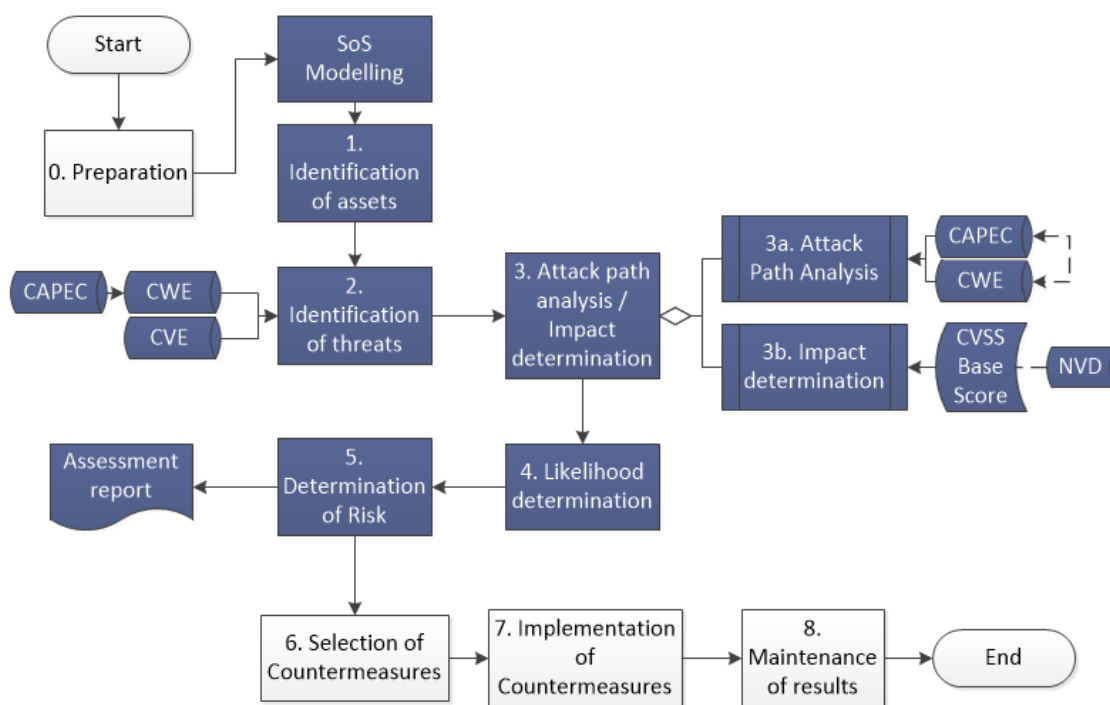


Figure 61 Overview of the Methodology (in blue the steps object of this deliverable and assisted by ResilBlockly, as well as databases or external data integrated within it).

This section provides details about the methodology shown in Figure 61, and in particular about the steps represented with blue rectangles, which can be assisted by ResilBlockly. These steps are:

- step 1 (Identification of assets) which is addressed in section 4.2;
- step 2 (Identification of threats) in section 4.3, where the lists of known weaknesses from CWE [86], vulnerabilities from CVE [49] and NVD [51], and attack patterns from CAPEC [54], are analysed and the process for identifying and associating vulnerabilities and weaknesses to the system components is determined;
- step 3a (Attack path analysis) in section 4.4, where different graphical representations built leveraging the above listed catalogues and their existing relations are proposed and discussed;
- step 3b (Impact determination) in section 4.5, where approaches for impact determination (which in the case of vulnerability is based on CVSS) are presented;
- step 4 (Likelihood determination) in section 4.6, and
- step 5 (Determination of Risk) in section 4.7.

Their description, given herein in Section 4, will be further extended in Section 6 with the tool's implementation details.

Steps 0 (preparation), and 8 (maintenance of results) are briefly addressed in sections 4.1 and 4.8 respectively. Finally, step 6 (Selection of Countermeasures) and eventually also step 7 (implementation of countermeasures) are not in the scope of this deliverable and will be eventually addressed within D6.4 *"Mitigations identification and their design"*.

4.1. Preparation

In our context, the purpose of the risk assessment is determining the risk information about a particular ICT solution, part of the supply chain, that constitutes the system under analysis and is examined for identifying and potential weaknesses, vulnerabilities, and attack patterns, as well as impacts to the system components that could, later on, constitute an issue especially in case of propagation over for the supply chain.

The main preparatory activity consists in obtaining the useful information sources (e.g., the technical documents describing the system architecture, assets, functionalities, and so on), either the system is still at the design stage, or is already implemented and running.

The scope of the assessment and the operating environment have also to be identified: in our case we can focus on a risk analysis of a specific system under analysis, however, e.g., by adopting the SoS concepts, the risk assessment can be extended to other systems and components in the ICT ecosystem.

The proper definition of terminology and the concepts is also a fundamental part of preparation phase; in our case, we refer to the concepts introduced in Section 2 and we will later on clarify them, where needed, in order to resolve possible ambiguities.

Assumptions on the threat environment, and events need to be made by the assessor: in this case the main focus is on cyber and malicious, human-made threats. However, the methodology and also the tool could be adapted to address also other kind of threats (e.g., physical or cyber-physical). Regarding the threat sources, the assessment provided in this case is not differentiated depending on threat agents and their skills; in other words, we do not provide different assessments for different attackers. Anyway, this variable of the assessment could be introduced later on, both in the methodology as well as in the tool.

Finally, the risk model has to be identified; it is detailed within Section 4.7 (as general methodology) and 6.6 (with specific details regarding the risk analysis in ResilBlockly).

4.2. Identification of the Assets

As summarized in Table 16, the identification of the assets consists in listing the components that are going to be analysed; if necessary, a partitioning of the system under analysis or its division in hierarchical levels can be provided.

In the case of ResilBlockly, the association of weaknesses or vulnerabilities to elements of the model will be considered, implicitly, as asset identification. Moreover, this identification can take place also during the profiling, -thus even before the modelling of a specific instance of a system-, that is during the definition of a meta-model, applicable to a category of systems of the same domain.

4.3. Identification and Modelling of Threats

This step consists in the identification of threats, attacks and vulnerabilities that apply to each asset. More in detail, for each system component, that as described before is

implicitly identified as asset during this step, the methodology requires the association of potential weaknesses or vulnerabilities.

In order to achieve the goal and reduce the intrinsic difficulty³⁰ of this process, the methodology leverages the MITRE lists of known threats, and in particular weaknesses from CWE [86], vulnerabilities from CVE [49] and NVD [51], and attack patterns from CAPEC [54], which provide a common baseline and understanding of the threats. These catalogues have been chosen as they are widely adopted and referenced in industry, academia, standards, etc.³¹. In principle, however, other threats originating from different platforms and datasets (e.g., the ones introduced in D3.1 [105]), as well as user-defined weakness or vulnerabilities can be identified and associated similarly.

Sections 4.3.1, 4.3.2 and 4.3.3 are mainly the report of our analysis of the catalogues and their schemas, conducted to identify the most interesting fields for the purpose of threat identification and for the final goal of defining the threat modelling and security risk assessment methodology. Section 4.3.4 describes instead the threats identification and association process.

4.3.1. CWE - Common Weakness Enumeration

The Common Weakness Enumeration (CWE) [86] is a community-developed list of typical and well-known software and hardware weakness types. In this catalogue, the weaknesses are defined as flaws, faults, bugs, vulnerabilities, or other errors in software or hardware implementation, code, design, or architecture that if left unaddressed could result in systems, networks, or hardware being vulnerable to attack. One of its main merits is that has been built over the years with the help of a number of external sources³² and is updated about four times a year. At the time of writing, the CWE List latest version is the 4.4, and the number of total weaknesses is 918.

The CWE weaknesses are organized and can be consulted according to three main views:

1. *Software Development*³³ (composed by 418 weaknesses in 40 categories, e.g., authentication errors, state issues, API/Function errors, etc.) organizes weaknesses around concepts that are frequently used or encountered in software development;
2. *Hardware Design*³⁴ (composed by 95 weaknesses in 12 categories, e.g., memory and storage issues, debug and test problems, etc.), organizes weaknesses around concepts that are frequently used or encountered in hardware design;
3. *Research Concepts*³⁵ (composed by all the 918 weaknesses in CWE), It is mainly organized according to abstractions of behaviours and is intended to facilitate the research.

4.3.1.1. Description and analysis of CWE schema

This Section describes and analyses some of the typical fields in CWE entries according to CWE version 4.4 and mostly focusing on the fields that are relevant for the following of

³⁰ i.e., it is impossible to imagine every potential threat existing, it is difficult to determine whether a threat may or may not exist for a system, and the model will never be complete

³¹ <https://cwe.mitre.org/community/citations.html>

³² <https://cwe.mitre.org/about/sources.html>

³³ <https://cwe.mitre.org/data/definitions/699.html>

³⁴ <https://cwe.mitre.org/data/definitions/1194.html>

³⁵ <https://cwe.mitre.org/data/definitions/1000.html>

the deliverable. The description of the fields is thought to be read with an example at hand. The full list of fields, including some optional ones can be found in the schema documentation^{36 37} or in the glossary³⁸. Optional fields are delimited by squared brackets.

ID: unique identifier for the entry, e.g., *CWE-262*.

Name: a string that identifies the entry, without mentioning the attack that exploits the weakness or the consequences of exploiting it. Example: *Not Using Password Aging*.

Status: defines the different status values that an entity (view, category, weakness) can have, e.g., *stable*, *draft*, *incomplete*, etc.









Symbol	Meaning
	View
	Category
	Weakness - Class
	Weakness - Base
	Weakness - Variant
	Compound Element - Composite
	Compound Element - Named Chain
	Deprecated

Figure 62 Symbols for Weaknesses abstractions and types in CWE

Abstraction: defines the different abstraction levels, among the following five, that apply to a weakness: A **Pillar** (the most abstract type), a **Class** (also very abstract, but more specific than a Pillar Weakness), a **Base** (a more specific type of weakness that is still mostly independent of a resource or technology, but with sufficient details to provide specific methods for detection and prevention), **Variant** (is a weakness that is linked to a certain type of product, typically involving a specific language or technology).

There is also another abstraction that in our case and in the interest of determining weakness and attack tree structures, deserves a more in-depth description. The **Compound** weakness is a meaningful aggregation of several weaknesses, which can be, which can vary depending on the structure and are currently known as either a *Chain* or *Composite*.

Structure: the structure of a weakness, either *Simple*, *Chain*, or *Composite*:

- A *chain* is a sequence of two or more separate weaknesses that can be closely linked together³⁹. One weakness, X, can directly create the conditions that are necessary to cause another weakness, Y, to enter a vulnerable condition. When this happens, CWE refers to X as "*primary*" to Y, and Y is "*resultant*" from X. Chains can involve more than two weaknesses, and in some cases, they might have a tree-like structure.

The **CanPrecede** relationship is used to identify when the weakness is primary to others, and **CanFollow** is used to identify when a weakness is resultant from others.

³⁶ <https://cwe.mitre.org/data/archive.html>

³⁷ <https://cwe.mitre.org/documents/schema/>

³⁸ <https://cwe.mitre.org/documents/glossary>

³⁹ https://cwe.mitre.org/data/reports/chains_and_composites.html

Named Chains: while CWE primarily contains "implicit" chaining relationships, there are several chains that are so common that they were assigned their own CWE identifiers. Named chains possess the optional *Chain Components* field, where the nature of relations **StartsWith** and **FollowedBy** are detailed similar to other relationships (see below).

- A *Composite* is a combination of two or more separate weaknesses that can create a vulnerability, but only if they all occur all the same time. One weakness, X, can be "broken down" into component weaknesses Y and Z. The **Requires** relationship is used by a composite to identify its component weaknesses, and the **RequiredBy** relationship is used by the components of that composite.

At the time of writing, the number of Chains is 78, while there are 3 Named Chains, and 4 Composites.

Description: short definition of the weakness and its key points.

[Related Weaknesses]: is used to refer to other weaknesses that and high-level categories that are related to the weakness. It contains one or more *Related_Weakness* elements, each of which is used to link to the CWE identifier of the other Weakness. Additional attributes included here can be:

- *Nature*: the nature of the relation (e.g., ChildOf, ParentOf, PeerOf, MemberOf, CanPrecede, CanFollow);
- *Type*: one of the symbols and types as in Figure 62
- *ID*:
- *Name*: the optional Ordinal attribute is used to determine if this relationship is the primary ChildOf relationship for this weakness for a given View_ID.
- This attribute can only have the value "Primary" and should only be included for the primary parent/child relationship.

[Applicable Platforms]: the list possible areas for which the given weakness could appear. These may be for specific named Languages, Operating Systems, Architectures, Paradigms, Technologies, or a class of such platforms. The platform is listed along with how frequently the given weakness appears for that instance. They could also be language-independent.

[Likelihood of Exploit]: contains a list of values corresponding to different likelihoods.

[Common Consequences]: is used to specify individual consequences associated with a weakness, and can be a very important field to be considered during a risk assessment. At the time of writing, this optional field recurs in 870 CWE entries.

- **Scope** identifies the security property (or properties) that is violated (e.g., Integrity).
- **[Impact]** is a textual description of the negative technical impact that arises if an adversary succeeds in exploiting this weakness (e.g., *Bypass Protection Mechanism*).
- **[Likelihood]** element that identifies how likely the specific consequence is expected to be seen relative to the other consequences.

[Detection Methods]: is used to identify methods that may be employed to detect this weakness, including their strengths and limitations.

- **Method** identifies the particular detection method being described.

- **Description** is intended to provide some context of how this method can be applied to a specific weakness.
- **[Effectiveness]** says how effective the detection method may be in detecting the associated weakness. This assumes the use of best-of-breed tools, analysts, and methods.
- **[Effectiveness_Notes]** element provides additional discussion of the strengths and shortcomings of this detection method.

[Potential Mitigations]: it contains one or more potential Mitigation elements, which each represent individual mitigations for the weakness.

- *Phase* indicates the development life cycle phase during which this particular mitigation may be applied.
- *Strategy* describes a general strategy for protecting a system to which this mitigation contributes.
- *Effectiveness* summarizes how effective the mitigation may be in preventing the weakness.
- *Description* contains a description of this individual mitigation including any strengths and shortcomings of this mitigation for the weakness.

[Observed Examples]: specifies references to a specific observed instance of a weakness in real-world products. Typically, this will be a CVE reference, that should contain the identifier for the example being cited in the standard CVE identifier format, such as CVE-YYYY-XXXX.

This field is one of the most interesting ones according to our purposes and from the perspective of representing threats with a tree or graph structure, as in the objectives of WP6. Details are provided in the following, especially in Section 4.4 and Section 6.4.

[Affected Resources]: is used to identify system resources that can be affected by an exploit of this weakness. If multiple resources could be affected, then each should be defined by its own Affected_Resource element.

[Taxonomy Mappings]: is used to provide a mapping from an entry in CWE to an equivalent entry in a different taxonomy. Examples of taxonomies are in the CWE sources⁴⁰.

This could be interesting in the context of an assessment especially for connecting an ongoing analysis with possible already existing ones conducted by other entities according to different taxonomies. However, this optional field is quite rare, since it is currently appearing in 190 CWE entries only.

[Related Attack Patterns]: contains references to attack patterns associated with this weakness. The association implies those attack patterns may be applicable if an instance of this weakness exists. Each related attack pattern is identified by a CAPEC identifier.

This is one of the most important fields according to our purposes and from the perspective of building an attack tree. Details are provided in the following, especially in Section 4.4 and Section 6.4.

⁴⁰ <https://cwe.mitre.org/about/sources.html>

4.3.2. CVE - Common Vulnerabilities and Exposures Catalogue

The Common Vulnerabilities and Exposures Catalogue (CVE) is a dictionary of publicly known cybersecurity vulnerabilities which purpose is to uniquely identify and name publicly disclosed vulnerabilities pertaining to specific versions of software or codebases. The CVE list is increased daily, and, at the time of writing, the number of total vulnerabilities is 162473.

4.3.2.1. Description and analysis of CVE schema

This section describes and analyses the typical fields in CVE entries, similarly to what has been done and presented for CWE. The description of the fields is meant to allow understanding the following of the deliverable.

Each CVE Record includes the following fields⁴¹.

CVE-ID. Unique identifier number of CVE vulnerabilities, with four or more digits structured as CVE-YYYY-XXXX; having a unique identifier is particularly important in ecosystems that encompass different constituent systems and stakeholders, as addressed in BIECO, and need to share information about vulnerabilities in an unambiguous way.

Description. Unique description which provides the relevant details to help users in finding the CVE Record for a specific vulnerability, and/or to distinguish between similar-looking vulnerabilities⁴². It may include (but not all the Descriptions do include them) details such as:

- name of the affected product and vendor,
- summary of affected versions,
- vulnerability type,
- impact,
- access that an attacker requires to exploit the vulnerability, and
- important code components or inputs that are involved.

References. Pertinent references (i.e., vulnerability reports and advisories) are provided to help distinguish between vulnerabilities. Each reference used in CVE identifies the source e.g., with an URL to source's website

Assigning CNA. The CVE Numbering Authority (CNA) that assigned the CVE ID to the vulnerability⁴³.

Date Record Created. It indicates when the CVE ID was issued or the CVE Record published on the CVE List. This date does not indicate when the vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE. That information may or may not be included in the description or references of a CVE Record, or in the enhanced information for the CVE Record that is provided in the NVD.

The CVE List feeds the NVD or National Vulnerability Database (see section 4.3.2.2), which then builds upon the information included in CVE Records to provide enhanced information for each record such as fix information, severity scores, and impact ratings (as also introduced in D3.1 [105]). Table 22 describes the possible states of CVE records, according to NVD⁴⁴.

Table 22 Status of CVEs

Status	Description
--------	-------------

⁴¹ <https://cve.mitre.org/cve/identifiers/index.html>

⁴² https://cve.mitre.org/about/faqs.html#cve_record_descriptions_created

⁴³ <https://cve.mitre.org/cve/cna.htm>

⁴⁴ <https://nvd.nist.gov/vuln/vulnerability-status>

<i>PUBLISHED</i>	It is populated with details. These are a CVE Description and reference links regarding details of the CVE
<i>RESERVED</i>	It has been reserved for use by a CNA or security researcher, but the details of it are not yet published. A CVE Record can change from the RESERVED state to being published at any time based on a number of factors both internal and external to the CVE List. Once the CVE Record is published with details on the CVE List, it will become available in the NVD
<i>DISPUTED</i>	When one party disagrees with another party's assertion that a particular issue in software is a vulnerability, a CVE Record assigned to that issue may be designated as being <i>Disputed</i> .
<i>REJECT</i>	A CVE Record that is not accepted as a CVE Record. The reason of the rejection is often stated in the description. Possible examples include it being a duplicate CVE Record, it being withdrawn by the original requester, it being assigned incorrectly, or some other administrative reason. As a rule, REJECT CVE Records should be ignored.

4.3.2.2. NVD – (US) National Vulnerability Database

The NVD is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). It is a reference for vulnerability management, security measurement, and compliance and includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics [51].

When a vulnerability is identified, and CVE IDs are assigned, the information in NVD is updated permanently, is typically available in the NVD within an hour, and is fully synchronized with the CVE List so that any future updates to CVE appear immediately in NVD.

The NVD ingests from the CVE List and in turn performs analysis to determine and associate impact metrics (based on the CVSS), vulnerability types (i.e., CWE weaknesses), and applicability statements (CPE⁴⁵, the Common Platform Enumeration), as well as other pertinent metadata. The NVD does not actively perform vulnerability testing, relying on vendors and third-party security researchers to provide information that is then used to assign these attributes.

In any case, for the interest of BIECO and our methodology, the NVD is an important source of information, especially for the association between CVE and CWE, where the latter can be retrieved starting from the former, and also for the CVSS base score information. Regarding this last point, some NVD records contain the CVSS v2 while others contain the score computed with both versions of the scoring system.

The NVD Dashboard⁴⁶ contains updated information about the CVEs received and processed, as well as the CVSS score distribution between CVSS V3 and CVSS V2.

4.3.2.3. Description and analysis of NVD schema

Each CVE that is registered in NVD Database, includes the following fields:

⁴⁵ <https://nvd.nist.gov/products/cpe>

⁴⁶ <https://nvd.nist.gov/general/nvd-dashboard>

ID. The unique identifier of the CVE.

Description. A summary of the vulnerability, which can include information such as the vulnerable product, impacts, attack vector, weakness or other relevant technical information. In some cases, CVEs may display a *Current Description*, available at the time of viewing, and *Analysis Description*, that were available at the time of NVD analysis. Descriptions are maintained by the CVE Assignment Team.

Severity⁴⁷. Each vulnerability is associated a CVSS v2 and CVSS v3 vector string. CVSS vector strings consist of exploitability and impact metrics. These metrics can be used in an equation to determine a number ranging from 0.0 to 10.0. The higher the number, the higher the severity of the vulnerability.

Figure 63 shows an example of Severity field for an NVD where CVSS v3.x is selected.

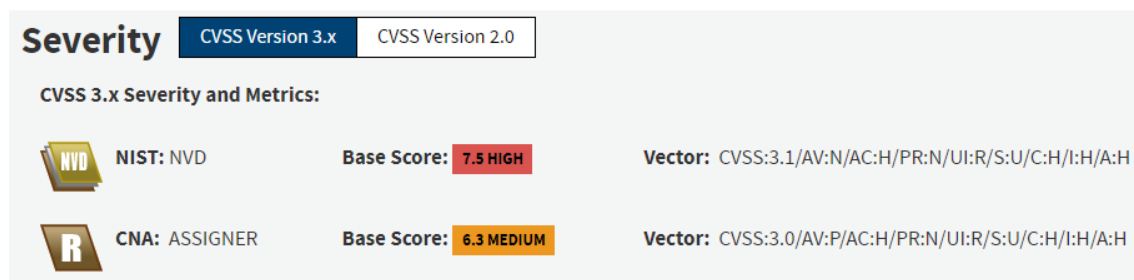


Figure 63 Example of NVD Severity

[References]: these URLs are supplemental information relevant to the vulnerability, which include details that may not be present in the CVE Description. References are given resource tags such as third-party advisory, vendor advisory, technical paper, press/media, VDB entries, etc. The NVD does not have direct control over them.

[Weakness Enumeration]: the common software security weakness from CWE related to the CVE. The NVD uses the CWE-1003 view when associating CWEs to vulnerabilities⁴⁸.

[Known Affected Software Configurations]: this section of the vulnerability detail page is used to show what software or combinations of software are considered vulnerable at the time of analysis. The NVD uses the CPE Specification [50] when creating these applicability statements⁴⁹.

It is important to clarify that the vulnerabilities within the NVD are derived from the CVE List which is maintained by processes upstream of the NVD. The comparison between statuses of CVE and NVD as well as the process involving CVE analysis is in [118].

4.3.3. CAPEC - Common Attack Pattern Enumeration and Classification

The CAPEC [54] is a public catalogue of typical attacks patterns, which are descriptions of common attributes and typical approaches employed by attackers to exploit known weaknesses. The purpose is to help users understand how adversaries exploit weaknesses in their applications, how attacks are designed and executed, with the clear aim of giving guidance on mitigate the attacks.

⁴⁷ <https://nvd.nist.gov/vuln/vulnerability-detail-pages>

⁴⁸ <https://cwe.mitre.org/data/definitions/1003.html>

⁴⁹ <https://nvd.nist.gov/vuln/vulnerability-detail-pages>

Currently, the CAPEC lists 553 attack patterns. They are organized and can be consulted according to two main hierarchical views:

4. *Mechanisms of Attack*⁵⁰: composed by 9 categories, based on the mechanisms and different techniques that are often used when exploiting a vulnerability to attack a system;
5. *Domains of Attack*⁵¹: composed by 6 categories, representing the attack domain (e.g., *software, hardware, supply chain, physical security, etc.*).

Additional external mappings are: the view by WASC⁵², ATT&CK⁵³ and OWASP⁵⁴.

4.3.3.1. Description and analysis of CAPEC schema

This Section describes and analyses some of the typical fields in CAPEC entries according to CAPEC version 3.4 and mostly focusing on the fields that are relevant for BIECO and the purpose of risk analysis which is in object of WP6.

The full list of fields, including some optional ones, can be found in the schema documentation⁵⁵, XML schema definition⁵⁶ or in the glossary⁵⁷.

ID. A unique identifier for the attack pattern

Name. A string that identifies the entry, without mentioning the weaknesses that exploits or the consequences of exploiting it.

Status. Defines the different status values that an attack pattern - but also a view or category-, can have.

Abstraction. Defines the abstraction level that apply to an attack pattern:

- *Meta* level attack (indicated with M, is the most abstract type;
- *Standard* level attack pattern (indicated with S, is also very abstract but more specific than a Meta level attack);
- *Detailed* level attack pattern (indicated with D, is a low level of detail, typically leveraging a specific technique, targeting a specific technology, expresses a complete execution flow and often leverages a number of different standard level attack patterns chained together to accomplish a goal).

Description. High level definition of the attack pattern. It should include how malicious input is initially supplied, the weakness being exploited, and the resulting negative technical impact.

[Relationships]: is used to refer to other attack patterns and give insight to similar items that may exist at higher and lower levels of abstraction. It contains one or more *Related_Attack_Pattern* elements, each of which is used to link to the CAPEC identifier of the other attack pattern. The nature of the relation is also capture by the Nature attribute of a related attack pattern. Additional attributes included here can be:

- *Nature*: the nature of the relation (see *Related Nature* below);
- *Type*: one of the types abstractions (M, D, S);

⁵⁰ CAPEC VIEW *Mechanisms of Attack* <https://capec.mitre.org/data/definitions/1000.html>

⁵¹ CAPEC VIEW *Domains of Attack* <https://capec.mitre.org/data/definitions/3000.html>

⁵² CAPEC VIEW WASC Threat Classification 2.0 <https://capec.mitre.org/data/definitions/333.html>

⁵³ CAPEC VIEW ATT&CK related patterns <https://capec.mitre.org/data/definitions/658.html>

⁵⁴ CAPEC VIEW OWASP related patterns <https://capec.mitre.org/data/definitions/659.html>

⁵⁵ CAPEC Schema Documentation: <https://capec.mitre.org/documents/schema/index.html>

⁵⁶ CAPEC Schema xsd https://capec.mitre.org/data/xsd/ap_schema_latest.xsd

⁵⁷ CAPEC Glossary: <https://capec.mitre.org/about/glossary.html>

- *ID*: ID of related attack pattern;
- *Name*: name of related attack pattern.

This is one of the most interesting fields.

[Related Nature]: defines the different values that can be used to define the nature of a related attack pattern:

- A ChildOf nature denotes a related attack pattern as a higher level of abstraction;
- A ParentOf nature denotes a related attack pattern as a lower level of abstraction;
- The CanPrecede and CanFollow relationships are used to denote attack patterns that are part of a chaining structure.
- The CanAlsoBe relationship denotes an attack pattern that, in the proper environment and context, can also be perceived as the target attack pattern.
- The PeerOf relationship is used to show some similarity with the target attack pattern which does not fit any of the other types of relationships.

This field is one of the most interesting ones according to our purposes and from the perspective of representing threats with a tree or graph structure, as in the objectives of WP6. Details are provided in the following, especially in Sections 4.4 and 6.5.

[Likelihood of Attack]: contains a list of values corresponding to different likelihoods. It is used to capture an average likelihood that an attack that leverages this attack pattern will succeed, but “with the understanding that it will not be completely accurate for all attacks”.

[Typical Severity]: is used to capture an overall average severity value for attacks following this pattern, again not aiming at being completely accurate for all of them.

[Execution Flow]: indicates the steps by the attacker to reach the goal. For each step is contained Phase, Description and/or Techniques.

[Common Consequences]: is used to specify individual consequences associated with an attack pattern, and similarly to CWE, can be a very important field to be considered during a risk assessment.

- **Scope**: identifies the security property (or properties) that is violated (e.g., Integrity).
- **[Impact]**: describes the technical impact that arises if an adversary succeeds in their attack.
- **[Likelihood]**: element that identifies how likely the specific consequence is expected to be seen relative to the other consequences.

[Potential Mitigations]: is used to describe actions or approaches to prevent or mitigate the risk of an attack that leverages this attack pattern. Each individual mitigation approach should help in improving the resiliency of the target system, reduce its attack surface, or reduce the impact of the attack if it is successful.

[Prerequisites]: indicates one or more prerequisites for an attack and is used to provide a description of the conditions that must exist in order for an attack of this type to succeed.

[Skills Required]: is used to describe the level of skills or specific knowledge needed by an adversary to execute this type of attack.

[Resources Required]: is used to describe the resources (e.g., CPU cycles, IP addresses, tools) required by an adversary to effectively execute this type of attack.

[Taxonomy Mappings]: is used to provide a mapping from an entry (Attack Pattern or Category) in CAPEC to an equivalent entry in a different taxonomy.

As described for the CWE, also here this similar mapping can be interesting in the context of an assessment especially for connecting an ongoing analysis with possible already existing ones conducted by other entities according to different taxonomies.

[Related Weaknesses]: contains references to weaknesses associated with this attack pattern. The association implies a weakness that must exist for a given attack to be successful. If multiple weaknesses are associated with the attack pattern, then any of the weaknesses (but not necessarily all) may be present for the attack to be successful. Each related weakness is identified by a CWE identifier.

This last field is one of the most important fields according to the purposes described in this deliverable and from the perspective of building an attack tree which may highlight the steps and paths of an attacker in exploiting a weakness of the system or of one of its components. Details are provided in Section 4.4.

4.3.4. The Weaknesses and Vulnerabilities Identification Process

This step consists in identifying and associating the weaknesses and vulnerabilities to a component. The process has to be considered a following step with regard to the preparation and identification of assets, thus it builds on the assumption that the previous phases have been completed: i.e., the technical documentation of the system and its components has been retrieved and the assets have been identified.

The sub-steps required for the identification of weaknesses and vulnerabilities can be summarized as follows.

1. **Similarities identification** – Identification of similarities between components is suggested in order to reduce the effort required for this whole phase of the methodology. In fact, as some of the components may serve the same functionality, identifying weaknesses (vulnerabilities) may be required only once. After that, it could be added also to the rest of the components. If the functionality presents differences, then a base common characteristic can be identified, limiting the number of weaknesses (vulnerabilities) to be analysed.
2. **Keywords extraction** – Since the catalogues are relatively big, the extraction of key descriptive words for each component has to be performed in order to simplify and improve the retrieval of results. In example, the component name, function or category can be some examples of keywords.
3. **CWE** - Search weaknesses on the CWE catalogue by using the keywords identified and associate them to the component.
4. **CVE** - Search vulnerabilities on CVE by using the keywords identified and associate them to the component.
5. **CWE from CAPEC** - search weaknesses starting from the CAPEC, which means adopting the same keyword-based approach to retrieve relevant attack patterns, and, leveraging the “*related weaknesses*” field in CAPEC, retrieve the CWE weaknesses to be associated to the component.
6. **CVE from CWE** - After having identified and associated the weaknesses from CWE, the “*Observed Examples*” field in CWE can be used in order to retrieve additional and potentially relevant, vulnerabilities from the CVE catalogue. These CVE vulnerabilities may be an extension of the set retrieved in sub-step 4.

7. **CWE from CVE (through CWE)** – Exactly as in the above sub-step, after having identified and collected the vulnerabilities from CVE, the “*Observed Examples*” field in CWE can be inspected in order to identify the vulnerabilities and retrieve the related CWE weakness that is referring to it.
8. **CWE from CVE (through NVD)** – A different way to obtain the same result described in the previous step involves the NVD, where to look for an identified vulnerability, and leveraging the *Weakness Enumeration* field, retrieve the connected CWE weakness.

This process should offer a relatively wide list of weaknesses and vulnerabilities associated to the system components. Moreover, the list can be appropriately integrated with custom weaknesses and vulnerabilities which may be retrieved from different sources.

However, in all the above steps, results may need to be filtered (e.g., based on the CWE *applicable platforms* field, *programming language*, *technology affected*, as well as by relevance to the system/scenario with a manual checking).

This can be seen as an example of trade-off between precision and recall, typical in the information retrieval. On the one hand, the choice of good keywords is fundamental to tune the identification process towards the retrieval of relevant threats only, and this saves time in case the user has to manually analyse several weaknesses and vulnerabilities for each interface of a complex system. On the other hand, some relevant threats may be missed if the choice of keywords is not ideal.

As a future improvement, we plan the implementation of a threat identification algorithm which, leveraging the attributes of the system profile, can support the user and automatically propose CWE weaknesses and CVE vulnerabilities to be associated. Other strategies for the filtering of Weaknesses can leverage the graphical representation of the following sections (e.g., limiting the association to the CWEs in the first level of an HWT), or the views existing in CWE catalogue (already introduced in Section 4.3.1).

4.4. Graphical Representation and Attack Paths Analysis

A graphical representation may be helpful in order to assist the identification of threats and can serve as means for identifying additional weaknesses or vulnerabilities. Moreover, when assuming the form of an attack tree, the structure can be used to understand the possible paths an attacker may follow to exploit a weakness, and can be a smart way to understand where to place mitigations.

The purpose of the graphical representations is thus to give insights that can be useful during the risk assessment process.

Leveraging the information in CWE, CVE and CAPEC catalogues, several different type of trees can be designed which help this phase. In the following we present some of the most interesting ones that have been devised, designed and a subset of which developed within ResilBlockly.

Some of the following example trees have been built by: i) leveraging Talend⁵⁸ tool for extracting the information from the CWE, CVE, and CAPEC catalogues and generate relationships XML files; then ii) using the ADTree tool (introduced in Section 2.1.1.1) for building the graphical tree models by importing the XML.

⁵⁸ <https://www.talend.com/products/talend-open-studio/>

4.4.1. Hierarchical Weakness Tree

This type of tree shows the hierarchical structure of weaknesses and is based on *ParentOf* and *ChildOf* relations often available in the optional CWE field called related weaknesses. From a *starting* weakness identified during the keyword-based search (Section 4.3.4), a child weakness can be retrieved which may better specify the previous, more abstract one, by following the *ParentOf* relation. Based on the relationships in the catalogue, when available, it is possible to generate a Hierarchical Weakness Tree (HWT) for a target weakness associated with the component.

The HWT can generated in different ways, and categorized accordingly, i.e.,

- 1-Direction HWTs: where the tree includes either 'child' or 'parent' weaknesses related to starting weakness, where available;
- 2-Directions HWTs: the tree that includes both 'child' and 'parent' weaknesses related to the starting weakness, where available;

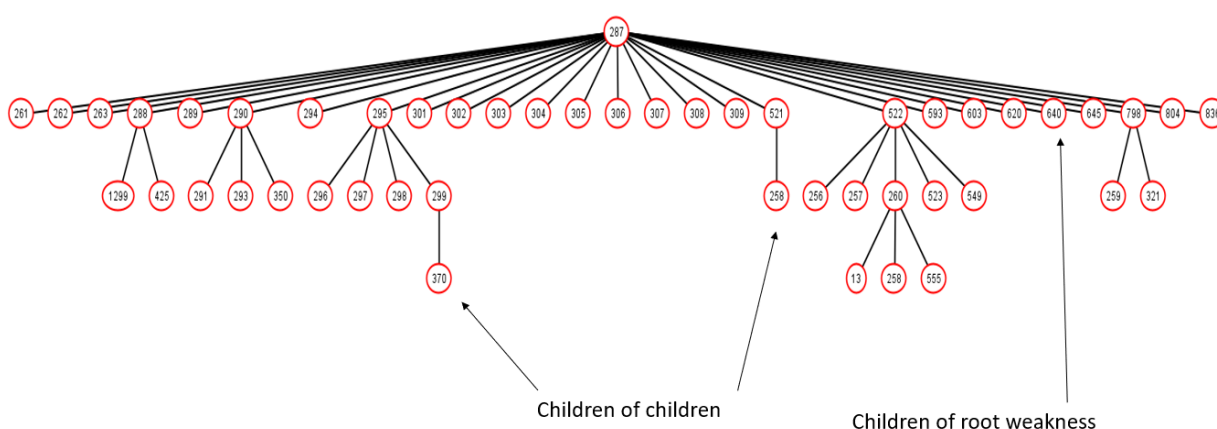


Figure 64 Example of HWT with all the child weaknesses for the “CWE 287: Improper Authentication”

Figure 64 shows an example of HWT tree for the CWE-287 named “Improper Authentication”. As can be seen, the dimension of the resulting tree is quite high, thus in order to be useful for the threat identification, from this tree, weaknesses that do not apply, e.g., because of applicable platform, or any other reason, should be filtered out by the user.

A possible approach to reduce the dimension of this tree and improve its usability is thus to build it only with CWE weaknesses that have been found during the keyword-based search: this can still provide a useful information, that is the representation of weaknesses of different abstraction levels which are connected with parent-child relations and are all associated to the target component.

Moreover, a possible hint for the user here is to consider only the more specific weaknesses associated and discard the more abstract ones: this means focus on the leaves of the tree. The resulting weaknesses, on the leaves of the HWT, can then be used as starting element for further analysis and could be root of other trees, as the ones presented in the following.

4.4.2. Weakness Chains Tree (WCT)

As discussed in Section 4.3.1, in some cases, weaknesses can be reached through *chain* connections, thus from a preceding weakness which directly create the conditions that are necessary to cause another weakness.

It is possible to build a graphical tree leveraging the *CanPrecede* and *CanFollow* relations existing in the CWE catalogue. This approach gives birth to the so-called Weakness Chains Tree (WCT), as in example the one shown in Figure 65, having as a root weakness the CWE-289 “Authentication Bypass by Alternate Name”.

So, considering to have modelled the system and associated e.g., the CWE-178 “Authentication Bypass by Alternate Name” to one component, this tree can inform the user and visually show which other weaknesses can be existing in the component, (in this case the CWE-289).

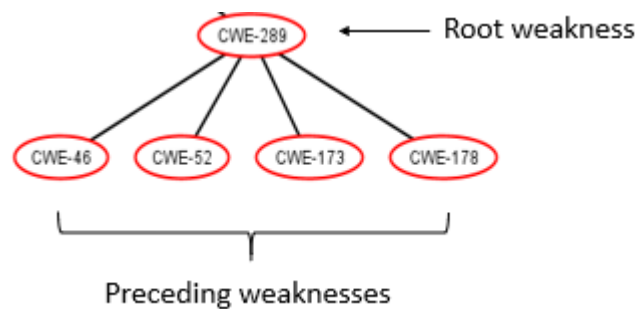


Figure 65 Example of WCT having as root the CWE 289: Authentication Bypass by Alternate Name

Table 23 Chains leading to CWE 289: Authentication Bypass by Alternate Name

Chains	1st weakness	2nd weakness
Chain_1	CWE-46	CWE-289
Chain_2	CWE-52	CWE-289
Chain_3	CWE-173	CWE-289
Chain_4	CWE-178	CWE-289

This type of tree is relatively useful, but has the main drawback that the number of chains of weaknesses in CWE is quite limited, as also discussed in Section 4.3.1.1.

4.4.3. Attack Path Tree and Attack Path Graph

Based on CWE-CAPEC relationship, and in particular on the *related attack pattern* and *related weakness* fields existing in them, respectively, it is possible to build a useful graphical representation having as a root a weakness identified during the keyword-based search, and associated to a system component, and having as its children attack patterns that are *related to* it, and potentially have been as well identified during the keyword-based identification. Then, connecting these attack patterns with additional patterns that *can Precede* them, we are going to obtain a structure that we call Attack Path Tree (APT) as the one shown in Figure 66.

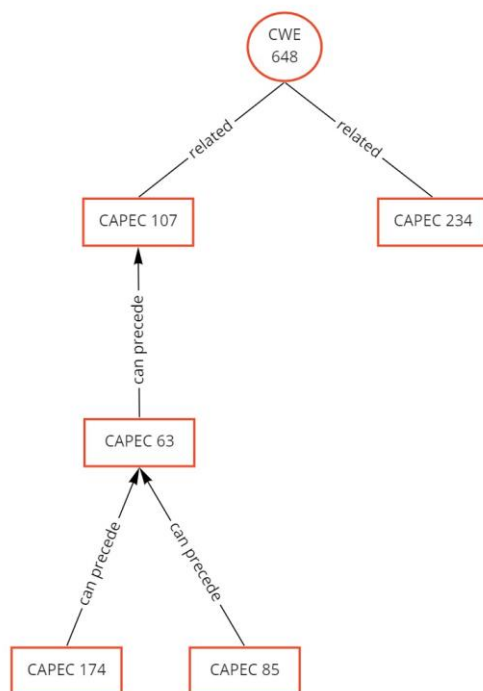


Figure 66 Example of Attack Path Tree for CWE-648

The tree identifies alternative attack paths that can potentially bring an adversary to exploit the CWE-648 weakness. Chains are also listed in Table 24

Table 24 Attack Paths represented with APT having CWE-648 as root weakness

Attack Path	1 st attack pattern	2 nd attack pattern	3 rd attack pattern	Exploited Weakness
Path_1	CAPEC-174	CAPEC-63	CAPEC-107	CWE-648
Path_2	CAPEC-85	CAPEC-63	CAPEC-107	CWE-648
Path_3	CAPEC-234	-	-	CWE-648

By extending an Attack Path Tree with the *related weaknesses* for all the attack patterns in the paths, it is possible to obtain a structure that can be called Attack Path Graph (APG). The APG is a useful structure which gives hints to the user and in assists the threats identification phase of the risk assessment. Figure 67 shows the APG for CWE-648: obtained by extending the APT of Figure 66 with the related weaknesses.

We believe that APG is particularly useful since it allows some early reasoning on which weaknesses could be more critical, e.g., based on the type and quantity of attack patterns to which they are related and on their position in the path. By definition, related weaknesses must exist for a given attack pattern to be successful: this means that if none of the weaknesses related to an attack pattern are considered existing in the system/component, and have not been associated to it, the attack pattern is unlikely to be successfully executed. On the contrary, an existing and associated weakness that has multiple related attack patterns, is something to be taken in careful consideration.

This graphical structure can be further improved by following some other rules, e.g.,

- by differentiating (e.g., greying out) or eliminating weaknesses not applicable for the system or component under evaluation;
- by adding mitigations information (e.g., with green squares, to adopt the formalism of ADtool, as introduced in section 2.1.1.1);

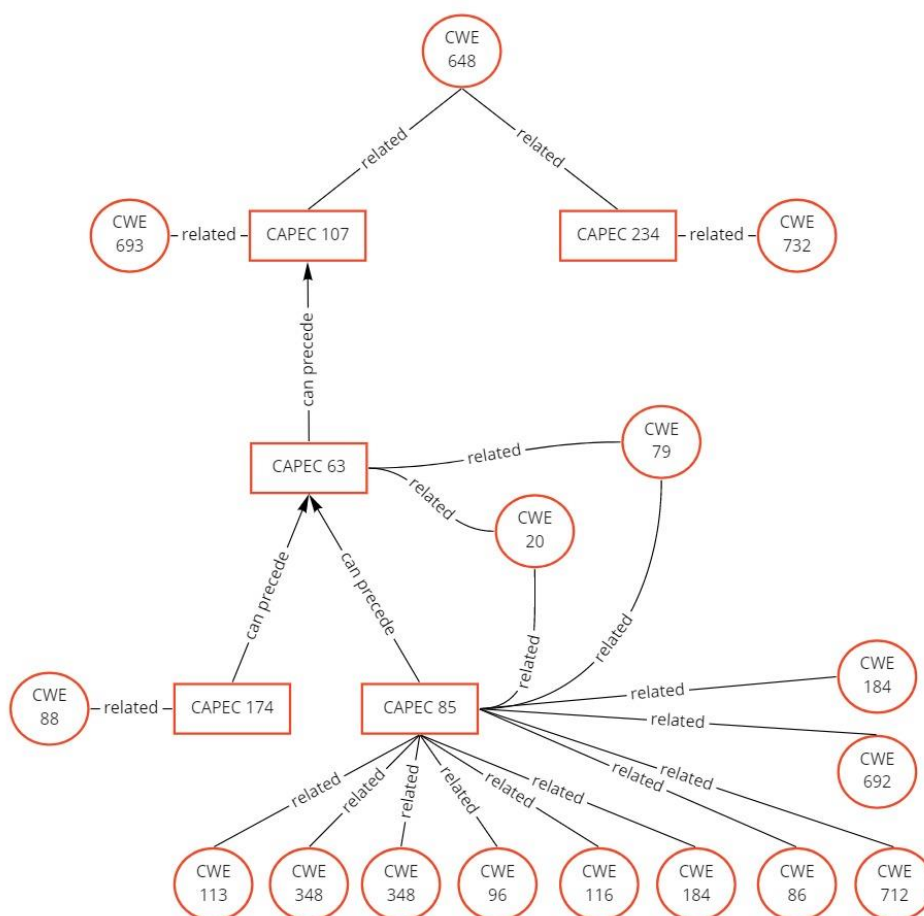


Figure 67 Attack Path Graph example related to CWE-648

4.4.4. Other Trees

Other structures that have been studied and are not described in detail here because considered less interesting for the BIECO risk analysis, are:

- *Weakness-Vulnerability Tree* (connecting a root CWE entry with “observed examples” CVE entries)
- *Hierarchical Attack Pattern Tree* (connecting a more general attack pattern with its children, more specific, attack patterns)
- *Weakness-Vulnerability-Attack Pattern Tree* (a unique tree with a CWE weakness as root node and as children both CVE vulnerabilities and CAPEC attack patterns)

There is also an additional tree, called *Weakness-Attack Pattern Tree*, that can be considered being incorporated in the Attack Path Tree (and Attack Path Graph).

The graphical structures described here are built by leveraging only the weaknesses, vulnerabilities, and attack patterns in the catalogues; however, completely custom or hybrid solutions can also be designed with similar approaches.

4.5. Impact and Severity

After having identified and associated weaknesses and vulnerabilities to system components and having eventually improved the identification by leveraging relations and their structured graphical representation, which enables a first preliminary risk (or criticality) analysis of the threats and attack paths, the risk assessment process continues with the step 3.

This phase of the risk assessment process, introduced in Section 2.4.4, includes an impact determination (step 3b), that is determination of the consequences of a vulnerability or weakness exploit and the estimation of the severity. The following sections discuss how the methodology distinguish the impact and severity of impact determination for vulnerabilities and weaknesses, respectively.

4.5.1. Vulnerabilities Impact and CVSS Base Score

As resulting from the studies of CVE (Section 4.3.2) and NVD (Section 4.3.2.2), as well as from the analysis of the state of the art risk rating and scoring systems which highlighted the features of CVSS (Section 2.2.2), the severity of impact for CVE vulnerabilities is an information that can be considered already existing and given, at least in a base form.

In fact, the widely adopted CVSS, includes in its outputs the *Base Score* a numerical value indicating the severity of a vulnerability according to its intrinsic characteristics, which are constant over time, and assumes the reasonable worst-case impact across different deployed environments [36]. Moreover, the base score, originating from the *base metric group*, includes information about the impact of a vulnerability in terms of security properties, that is, confidentiality, integrity and availability.

Therefore, the methodology introduced in this section bases the impact and related severity determination for vulnerabilities, on the CVSS Base Score, which can be retrieved from NVD (Section 4.3.2.3). The base score is usually available in NVD in two different formats, according to CVSS version 3.x or version 2. This requires that one of the two versions has to be chosen and applied for the whole assessment. At the time of writing, the total number of CVE scored based on CVSS v3 is 79344 while CVSS v2 is available for 153420 CVEs⁵⁹. Thus, according to the current numbers, the conservative choice would be the CVSS v2, due to its higher availability. Vice versa, if in some cases a CVE (NVD) entry possesses only the CVSS v3 base score, appropriate equivalence and conversions should be adopted. Finally, if the scoring is not available for none of the two versions, the underlying equation⁶⁰ could be still used for computing the base score, e.g., adopting the online calculator⁶¹. In case in the future the situation will be inverted, that is, having the CVSS v3 base score increasingly available or even replacing the v2, the choice could be taken accordingly.

The severity of impact score is retrieved in a quantitative value, ranging from 0.0 to 10.0, from which a corresponding a qualitative rating can be derived, and actually is also provided in NVD (as shown in Figure 63), according to the CVSS version, as show in Table 25.

⁵⁹ <https://nvd.nist.gov/general/nvd-dashboard>

⁶⁰ <https://www.first.org/cvss/specification-document>

⁶¹ <https://www.first.org/cvss/calculator/3.1>

In any case, the base score must be evaluated by the assessor and confirmed or updated according to parameters depending on the system under analysis, the environment, and so on. A useful information in this sense, where available, is the *Description* field of CVE and especially the *impact* information that it may include.

Table 25 Qualitative and quantitative severity rating scale in CVSS

CVSS v2 rating	CVSS Base Score	CVSS v3.x rating
Low	0.0	None
	0.1 - 3.9	Low
Medium	4.0 - 6.9	Medium
High	7.0 - 8.9	High
	9.0 - 10.0	Critical

The score has to be mapped with the NIST SP 800-30 *level of impact* scale (as in Table 10), as described in Section 4.7.

4.5.2. Weaknesses Impact and Severity

Regarding the weaknesses identified and associated with the system under analysis, a first approach considered for the impact determination and the related severity estimation, has been the adoption of CWSS (introduced in Section 2.2.1).

However, to our knowledge there is not a public database of weaknesses reporting also an already computed CWSS score (as it happens instead for CVSS within NVD). This is reasonable, since a weakness represent a type of mistake that may be even very abstract, that could contribute to the introduction of vulnerability, i.e., they represent a mistake that has been identified but not verified and proven and which did not occur in the specific product of the manufacturer. Thus, scoring the impact without knowing the context, the type of product, specific language or technology would be questionable and not representative.

Independently from that, conceptually, CVSS and CWSS are very similar, and it would be logical to apply it for weaknesses, since we adopt CVSS for vulnerabilities. However, while the CVSS seems to be widely adopted and recognized, this seems to be not true for the CWSS, which appears still at an early stage, even if the MITRE itself, when comparing CVSS v2 with CWSS, seems to emphasize only the pros of the latter⁶².

For these reasons, in general, the methodology designed in this WP and described here is not leveraging CWSS.

Thus, this phase of the risk assessment requires a research and study about the impact of a weakness, which in the case of CWE weaknesses may also leverage the *common consequences* field, where available, (which specifies consequences associated with a weakness and the security property that is undermined), while the scoring of the severity the impact is left as responsibility of the user. The user-defined severity of impact score for the weaknesses, will adopt the NIST SP 800-30 scale, thus from very low, to very high (as in Table 10).

⁶² https://cwe.mitre.org/cwss/cwss_v1.0.1.html#appendixA

4.6. Likelihood determination

The likelihood is probably the most delicate attribute in a risk assessment, since it is very much a matter of opinion, especially at design phase, when usually the feasibility and ease of exploitation of a vulnerability cannot be proven by security/penetration testing or similar methods.

We believe that this value cannot be retrieved from a catalogue or scoring system, but requires a deep analysis and knowledge of the system. Thus, both for vulnerabilities and weaknesses, the methodology has to rely on, historical data on successful cyber-attacks on similar systems, existing assessment reports, vendor/manufacturer vulnerability reports (for OTS system components), and, moreover, the assessor experience.

Still, all the information eventually available in the catalogues should be taken into account for deriving this value, i.e.,

- for the CWE the *likelihood of exploit* field, and/or the likelihood included in the *common consequences* field, and
- for the CVE the *description* field, especially when reporting the *access that an attacker requires to exploit the vulnerability*, as well as the exploitability metrics composing the CVSS base metrics.

The likelihood determined by the assessor is a value expressed according to the NIST SP 800-30 *likelihood* scale (as in Table 10), and it is the last missing input for determining the risk.

4.7. Risk

Once the impact and likelihood have been determined, the risk is easily deducted adopting the NIST SP 800-30 risk matrix of Table 10. The methodology for risk determination that is applied to CVE (and NVD) vulnerabilities is depicted in Figure 68. This, as briefly considered above, may require conversion of input values from quantitative to qualitative scales, or some careful equivalence when mixing different versions of CVSS base scores. The Risk for CWE weaknesses is computed in a similar way, apart from the fact the the impact is not converted from CVSS.

It is important to notice that the risk assessment described in this deliverable targets the potential weaknesses and vulnerabilities in the system, enabling their subsequent mitigation, but does not directly target the attack patterns. Attack patterns are mainly used as starting point that could be used for identifying more relevant weaknesses. Anyway, a similar methodology for identification, association and risk estimation could be also easily applied to them (e.g., to CAPEC entries, which possesses the *consequences* and *typical severity*, or also the *likelihood of attack* fields).

Moreover, the risk assessment here is not addressing the combination of risks determined for different weaknesses (vulnerabilities), either associated to the same asset, or to different assets in the same system. This could be for sure an interesting future development, e.g., starting from and evolving existing research results [119][120][121][122].

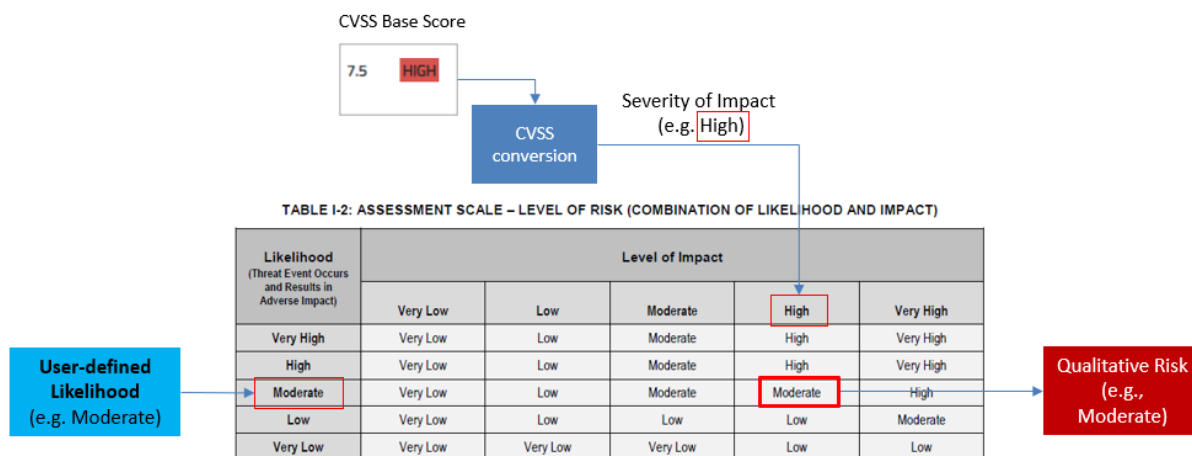


Figure 68 Overview of the (vulnerability) risk determination methodology, integrating CVSS and the NIST SP 800-30 risk matrix

4.8. Assessment Report

This step consists in documenting the results of the assessment. It is a step required and which applies not only to the methodology of Section 4 but also the the HAZOP-based methodology of Section 3.

The assessments should produce the necessary reporting documentation: in the case of the HAZOP-based methodology, a pre-filled HAZOP/THROP report containing the list of assets (functions, interfaces), to which the hazards systematically identified by applying the guidewords with the analysis template are mapped is generated; as shown in Figure 60 this report is partially completed within ResilBlockly, and the assessment has to be finalized offline. The format is CSV, and an example of this report and the fields contained will be given in Appendix of Deliverable D6.2.

Instead, in the case of the methodology of Section 4, as shown in Figure 61, the report contains the list of weaknesses and vulnerabilities that have been identified either with a search in the catalogues, or autonomously defined by the user, as well as the additional ones discovered thanks to the analysis of relations in the catalogues and attack paths, the report of the severity of impact, the likelihood and the resulting risk for each hazard, vulnerability or weakness.

Even if it is not in scope of this deliverable, the report contains a placeholder for suggested or potential mitigations and could be extended with an additional assessment repeated after the mitigation have put in place.

5. Applicability of MUD Standard in Modelling Systems and Interfaces

This section addresses the integration of the Manufacturer Usage Description (MUD) standard [130] in the methodology described, linking the security information that we can obtain from the design phase (modelling) with the runtime phase, when the system is deployed in a specific context. The main objective is to provide useful security information to the deployment network, so it can be used not only to know the security features of the system, but also to decide whether it is secure enough to take part of the system or to apply some security policies to reduce the attack surface and protect both the system and the network from the beginning.

In the next subsections, we describe the characteristics of the MUD standard, as well as the standardized format used to specify all the security information (Section 5.1). Then, Section 5.2 analyses the main flaws of the MUD model, especially those related with the expressiveness.

In the following of this deliverable (Section 6.2), there is our proposal to integrate the usage of the MUD within the modelling phase, allowing the user not only to create an original MUD file to enrich the model, but also to generate an extended MUD to be used during runtime. The inclusion of the MUD in the modelling phase and, moreover, in ResilBlockly, as well as the proposed extension will be further developed in Deliverable D6.2.

5.1. The Manufacturer Usage Description Standard

The MUD was standardized in 2019 within the scope of the Internet Engineering Task Force (IETF). The MUD specification's major goal is to limit the threat and attack surface of a certain IoT device by allowing manufacturers to establish network behaviour profiles for their devices. Each profile is built around a set of policies, or Access Control Lists (ACLs), that specify the communication's endpoints. MUD represents a scalable and flexible approach to the definition of network access policies beyond the use of IP addresses to enable communications with other services. A manufacturer could, for example, declare that access to particular cloud services, as well as connection with other manufacturers' devices, should be permitted. MUD also allows to specify protocols and ports for each communication to provide a more fine-grained configuration of access control rules. The standard also possibilities the extension of the scheme, allowing manufacturers to express other types of conditions or policies based on their needs. For example, while the MUD is focused on network access control regulations, MUD model expansions are being considered for Quality of Service (QoS) aspects of the communications. However, despite this flexibility, the MUD model does not provide mechanisms to describe more fine-grained aspects and additional security restrictions beyond the network layer.

One of the key advantages of the MUD approach is that the manufacturer is responsible for defining the devices' behavioural profiles (instead of the typical network administrator). Indeed, the MUD design and format make it possible to automate the creation of network access policies based on the manufacturer's MUD profile. It should be noted, however, that the instantiation of these profiles may be influenced by the network domain in which the device is deployed. The standard also defines an architecture to allow the network domain where the device is deployed to obtain and enforce this profile.

Since its adoption, MUD has received a significant interest from the research community and standardization bodies. In particular, the National Institute of Standards and Technology (NIST) proposes the MUD standard as a promising approach to mitigate security threats, and to cope with denial-of-service (DoS) attacks in IoT environments, including home and small-business networks. Additionally, the European Union Agency for Cybersecurity (ENISA) considers the use of MUD as part of IoT security good practices to improve, allowing devices to advertise their supported and intended functionality.

5.1.1. The MUD Model

The MUD standard restricts IoT device connections by defining Access Control Lists (ACLs), using the Yet Another Next Generation (YANG) standard to model network restrictions and using JavaScript Object Notation (JSON) for serialization. It's worth noting that the MUD model includes Network ACL extensions to the YANG data model, which are augmented by the MUD standard to specify more expressive ACLs.

The MUD file contains two main blocks: the “mud” and “acls” containers, as shown in Figure 69.

```
module: ietf-mud
  +--rw mud!
    +--rw mud-version          uint8
    +--rw mud-url              inet:uri
    +--rw last-update           yang:date-and-time
    +--rw mud-signature?       inet:uri
    +--rw cache-validity?      uint8
    +--rw is-supported          boolean
    +--rw systeminfo?          string
    +--rw mfg-name?            string
    +--rw model-name?          string
    +--rw firmware-rev?        string
    +--rw software-rev?        string
    +--rw documentation?       inet:uri
    +--rw extensions*          string
    +--rw from-device-policy
      | +--rw acls
      | | +--rw access-list* [name]
      | | | +--rw name -> /acl:acls/acl/name
      +--rw to-device-policy
        +--rw acls
          +--rw access-list* [name]
            +--rw name -> /acl:acls/acl/name
```

Figure 69 MUD standard model of the “mud” container

The “mud” container specifies several features related with the MUD file itself. The property *mud-version* specifies the current version of the MUD file, whereas the *last-update* defines when the current version of the file was generated. The MUD is identified by the *mud-url*, that is, the URL that can be used to retrieve the MUD file. The *mud-signature* (optional) verifies the integrity and authenticates the MUD file to avoid security issues. Furthermore, this container also allows to define optional aspects such as the model of the device (*model-name*), the firmware and software revision (*firmware-rev*, *software-rev*), minimum period of time before checking for updates (*cache-validity*), if the device will receive MUD/software/firmware updates (*is-supported*), additional information (*systeminfo*), link to the device documentation (*documentation*) and additional extensions. Finally, the containers *to-device-policy* and *from-device-policy* containers represent access lists references by indicating the appropriate direction of a specific flow to define the communication pattern of the device.

Consequently, the “acls” container defines those ACLs. Each ACL has a *name*, a set of conditions to apply the rule (*matches*), and the *actions* to apply in case the conditions are satisfied (e.g., forwarding accept or deny). By default, the MUD specifies only the allowed connections, but in certain cases, it can be also useful to define an explicit access restriction to a service (for example, if it has been compromised). The ACL extensions to the YANG data model adds additional keywords that facilitates the definition of high-level policies without the need to know the associated IP addresses. These keywords are *manufacturer*, *same-manufacturer*, *model*, *local-networks*, *controller* and *my-controller*. For example, the keywords *manufacturer* and *same-manufacturer* enable the definition of policies to allow or deny the interaction with devices from the same manufacturer. This way, MUD files define the type of communications and access of a certain device in the form of policies or ACLs. Some examples of these restrictions could be “allow the communication to devices of the same manufacturer”, “allow the access to a specific DNS service”, or “deny the access for a specific port”.

5.2. Limitations of the MUD standard

The MUD model enables a standardised and flexible way to specify network policies. However, as discussed before, one of the main limitations associated with the MUD standard is the lack of expressiveness for the definition of access restrictions beyond the network layer. Indeed, the definition of enriched behavioural profiles could be used to detect/avoid a broader range of potential security attacks, including application layer threats such as slow DDoS attacks³¹.

Based on the BIECO use cases analysis, we identified a set of characteristics that, although they are known at design time, the original MUD file is not able to represent and detail them.

- Application layer protocols, which also define restrictions on the communications.
- Communications with application layer entities such as databases, which are also part of the ecosystem in which the device is deployed.
- Cryptographic algorithms, which not only add restrictions to the communications, but also specify the supported algorithms of the device and its preferences.
- Authorization policies, which are a step forward to access control.
- Exposed resources (HTTP/CoAP API) that the device offers to other entities of the ecosystem, which can be restricted by other security policies and conditions.
- Restrictions on the number of communications, which can help to avoid denial of service (DoS) attacks.

Based on the analysed limitations here and taking into account additional aspects of the other use cases within BIECO project, D6.2 will provide a MUD model extension including at least the following aspects:

- Specification of any protocol at any TCP/IP stack layer to be used when communicating with another entity.
 - The user will be able to select any number of protocols to be used in each communication (e.g., CoAP, HTTP, MQTT, TCP, UDP, IPV6, IPV4, etc.).
 - The list of protocols will be fixed (at least including those used in the use cases) to provide harmonization.
- Communications with databases

- This type of communication will include as specific attributes the host (using a domain name, as usual in the MUD model), port and schema (name of the database).
- Cryptographic algorithms nation)
 - Each one-directional communication (source/destination) will have the possibility of specify the cryptographic algorithms required to establish it.
 - Additional parameters that will be added are the purpose of the cryptographic algorithm (integrity, ciphering, authentication, authorization), the algorithm used (e.g., AES, RSA, SHA, etc.), and the length used (for the keys, HASH, etc.).
 - As in the protocols, the set of algorithms will be fixed, at least including those used in the BIECO use cases, to provide harmonization.
- Exposed resources (HTTP/CoAP API) and authorization conditions
 - In case the device uses HTTP(s)/CoAP(s), the extended MUD will be able to model the exposed resources, indicating the required method (POST, GET, PUT, DELETE, etc.), the resource itself (e.g., /temperature, /update), and an associated authorization condition to access to it.
 - The authorization conditions will define the role needed to access to each resource, following an attribute-based authorization. For example, a user can be allowed to access to /OpenDoor of the smart door device if he has the attribute professor.
- Restrictions on the number of communications
 - The communications to the device can establish a limit on the simultaneous number of connections, from which the performance of the offer service will be affected or even from which the service could fail due to a DoS attack.
- Known vulnerabilities
 - As the MUD file is not completely static, and it can include modifications from the manufacturer, we foresee that the integration of the known vulnerabilities could be highly useful for the deployment domain, e.g., to apply mitigations or to disconnect the affected device from the network until a patch is available.
 - The included vulnerabilities will follow the format of a CVE and CWE entry, containing the ID and a description.
 - The vulnerabilities will include additional information from the ResilBlockly model, in particular the impact (CVSS base score in case of CVE entries), likelihood and risk (NIST matrix SP-800-30).

6. Implementation of the Methodologies in ResilBlockly

This Section introduces ResilBlockly and in particular section 6.1 describes the main technical and functional improvements that it brings with regard to Blockly4SoS.

Section 6.2 contains our proposal for the integration of MUD within the modelling phase in ResilBlockly, and the generation of an extended MUD.

Then, Section 6.3 briefly outlines how the HAZOP-based analysis template can be generated within the tool and according to the methodology of Section 3.

Finally, sections 6.4, 6.5 and 6.6 provide an explanation of how ResilBlockly supports the risk assessment methodology already introduced in Section 4, and assists in the application of steps from 1 to 5 of Figure 61.

The present document is not specifically addressing the systems modelling activity within ResilBlockly, which instead is given in the context of D6.2 “Blockly4SoS user guide”, together with the preliminary validation over one of the BIECO use cases and the detailing of all the ResilBlockly features.

6.1. From Blockly4SoS to ResilBlockly

This Section describes the main differences and new functionalities introduced within *ResilBlockly* which is the name given to the tool designed and implemented as an evolution of Blockly4SoS.

The refactoring of existing features and the introduction of completely new ones has been driven by the need for a having a comprehensive tool, not only capable of modelling the main concepts of cyber-physical systems (that was the main challenge addressed by Blockly4SoS and in AMADEOS), but also -in order to reach the BIECO project goals-, to model threats, hazards, and risk related concepts, to visually match risks to system components, to represent attack paths, and so on.

6.1.1. General Improvements

Blockly4SoS has been developed as a prototype, hence its project structure and software architecture needed important improvements.

Therefore, the following technical improvement points were identified and addressed:

1. Software modularity, to abandon a monolithic architecture.
2. Software maintainability: the code is easier to maintain thanks to the decoupling of the front-end layer (UI) from the back-end layer (Business), and it is now much more developer-independent.
3. Software reliability: the main algorithms were developed in JavaScript language and this does not allow a fast and easy extensibility and maintainability of the core software features⁶³.
4. Software extensibility: the layout adapts automatically to different resolutions (and devices).
5. Software validation: code is more testable.

⁶³ ResilBlockly is developed in Java and Angular languages for the backend and frontend respectively.

Furthermore, Blockly4SoS is domain-specific, since it was designed and intended for the SoS domain. In the following section, instead, it is described how ResilBlockly is made able to address and model many other domains.

6.1.2. Introduction of Profiling and Modelling Features

Two important concepts are being stressed here, since they are required for understanding the following of this document:

- *Profile*, sometimes also referred as *metamodel*, is an abstraction of components and relations for a specific domain;
- *Model*, is an instance of a profile.

As described in Section 2.5, Blockly4SoS allows the users to create model instances using the *AMADEOS* SoS profile, that is an ad-hoc and specific profile for the SoS domain.

However, it may be useful to evolve the SoS profile in order to better fit on users' interests and desires, and to get specialized for specific domains; one of them, for example, is the software and ICT ecosystem domain. Thus, in order to address this requirement, ResilBlockly introduces a profiling functionality, i.e., the so-called *Profile Designer* (the GUI for the choice between this feature and the modelling, i.e., the *Model Designer*, is depicted in Figure 70).

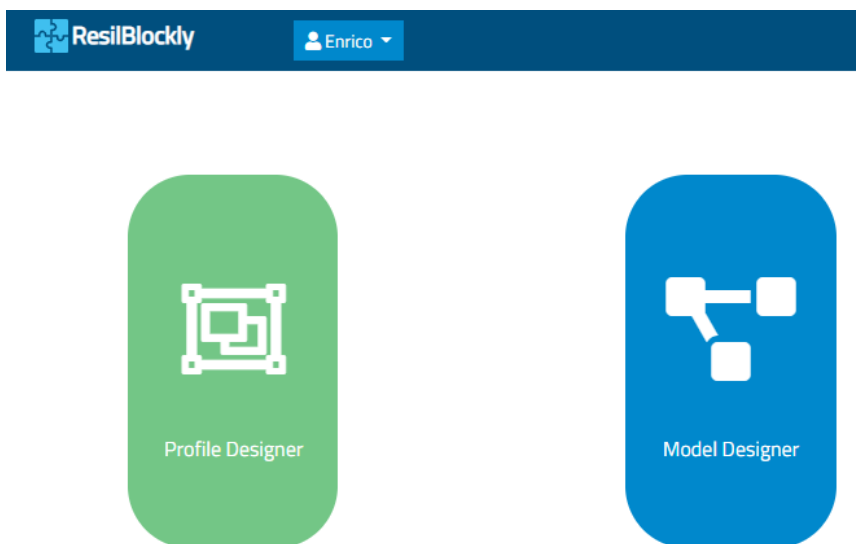


Figure 70 The GUI of ResilBlockly for the choice between Profiling and Modelling Features

The main differences between Blockly4SoS and ResilBlockly, with regard to the introduction of the Profile Designer, are summarized in Table 26.

Table 26 Comparison of Profiling and Modelling in Blockly4SoS and ResilBlockly

Tool version	Profile	Model
Blockly4SoS	Pre-defined (not modifiable)	User defined
ResilBlockly	User defined	User defined

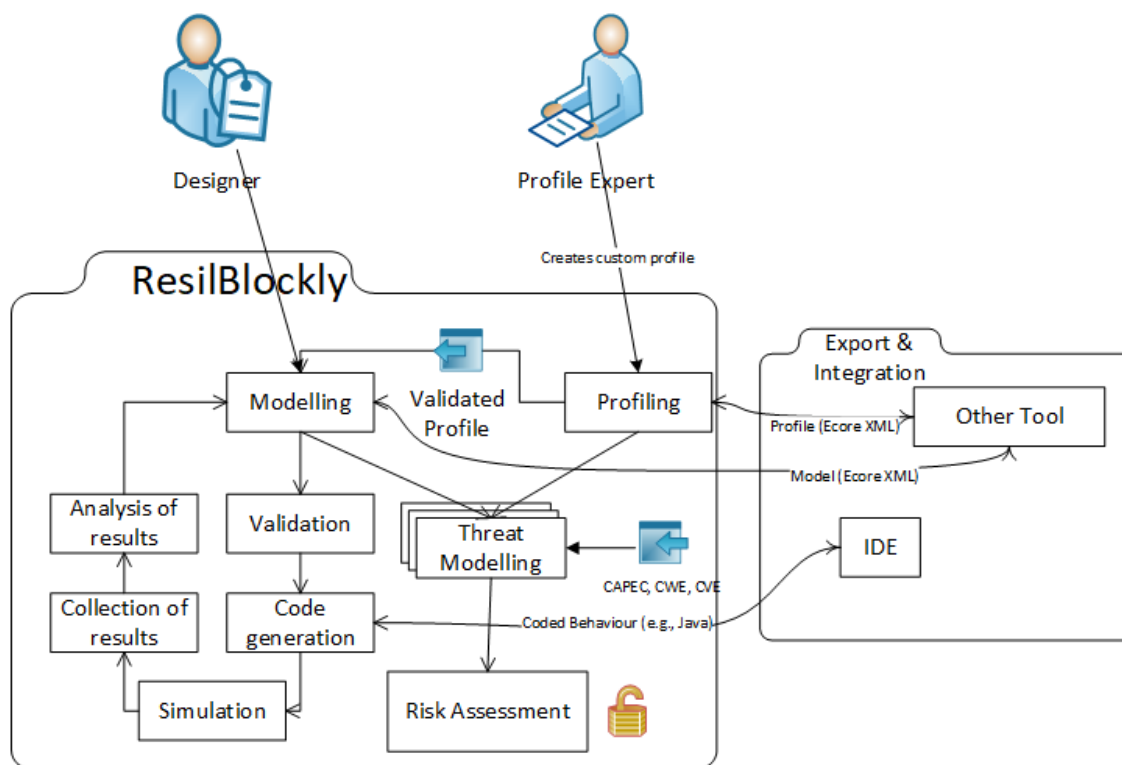


Figure 71 The ResilBlockly flow and categories of users (to be compared with Blockly4SoS flow in Figure 56)

Another important distinction that the profiling and modelling features are addressing, is the different type of users that ResilBlockly is thought for (also depicted in Figure 71):

- *Profile Expert* (mainly a Profile Designer user)
- *System Designer* (mainly a Model Designer user)

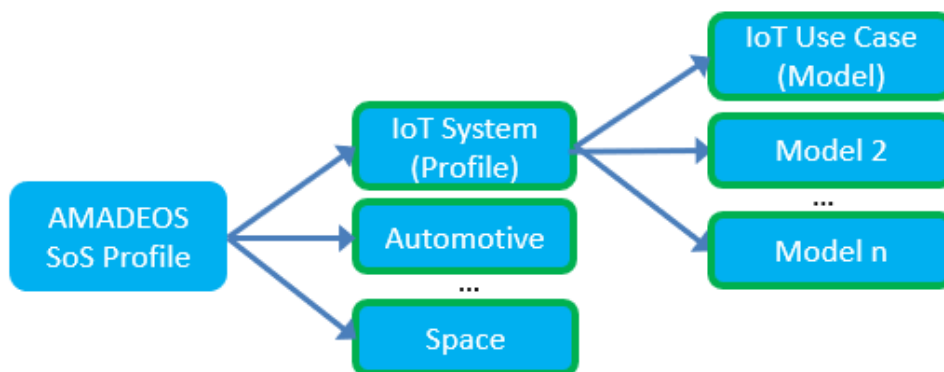


Figure 72 Example of Derivation of Profiles and Models

With ResilBlockly, an existing profile can be adopted (e.g., the AMADEOS SoS Profile) and, thanks to the Profile Designer, a *Profile Expert* can specialize n different profiles derived from it (as shown in Figure 72). As an alternative, new profiles can be created from scratch by the expert. This is one of the main values added by ResilBlockly with regard to Blockly4SoS.

The System Designer, instead, can choose one of the profiles and instantiate it within a model specific for their use case system; as in Blockly4SoS, also here the System Designer is not required to have deep knowledge about the domain (as, instead, the Profile Expert

is). Furthermore, the model is enriched with information (e.g., cybersecurity information as typical weaknesses) inherited from the profile.

The functionalities shown in the flow of Figure 71, and especially other important differences with Blockly4SoS, are going to be explained in the following sections. One of them is the threat modelling, which can be started in the Profiling, thus allowing the profile expert to identify and associate weaknesses and/or vulnerabilities that, in principle, may apply to an element of the profile (e.g., the most relevant weaknesses of a generic database). This activity is meant to be addressed and refined by the model designer, which, by knowing more details about the technologies adopted in the system under modelling (e.g., the specific type of database and its interfaces) is able to introduce new weaknesses and vulnerabilities or to remove previously identified ones.

Another important feature is the code generation, which has been completely redefined and reimplemented; details are given in section 7 and further in D6.2 “Blockly4SoS user guide”.

6.1.3. Interoperability, Ecore and EMF

In Section 2.5.4, the flow of MDE with Blockly4SoS (Figure 56), together with phases and outputs, have been described. As discussed in the previous section, one of the main features introduced in ResilBlockly, is the possibility to create different profiles (metamodels) already within the tool; in order to do so, the selected approach and file format enabling this activity has been the widely adopted Ecore, the base metamodel of the *Eclipse Modeling Framework* (EMF)⁶⁴ [112] [113][114].

EMF is a framework for modelling applications and generating customizable source code or other different outputs. It distinguishes the meta-model, which describes the structure, and the actual model instance. Users can create an *Ecore Model* by importing existing files e.g., Ecore, UML, XML schema, XMI or Java annotations, or creating it directly within EMF.

An *Ecore Model* is thus composed of two main description files [112]: *ecore* and *genmodel*.

- `modelname.ecore`: constitutes the meta-model, in XMI format. It is the representation of the modelled domain and contains the information about the defined classes.
- `modelname.genmodel`: specifies information for the code generation.

As depicted in Figure 73, between the main components of *ecore* we can find the following main elements [115]:

- `EClass` a representation of a class with optional attributes and references.
- `EAttribute` a representation of an attribute, with name and type
- `EReference` a representation of an association between two classes. It has flags to indicate if it represents a containment and a reference class to which it points.
- `EDataType` a representation of the type of an attribute, e.g., int, float, String.

⁶⁴ a project initially developed by IBM and then transferred to the Eclipse Foundation, which provides code generation and model manipulation tools.

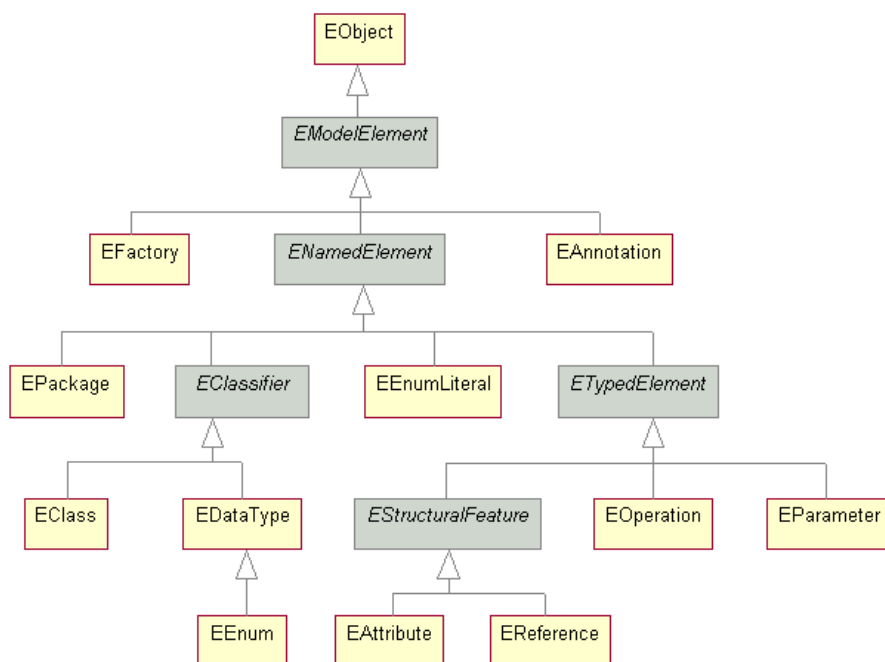


Figure 73 Hierarchy of Ecore components

The above listed elements can be considered the key ones based on which ResilBlockly enables the creation of profiles. It has to be noticed that, as in EMF, the Reference is one of the *Relation* types, and the same group includes also the Composition (a portion of EMF palette is shown in Figure 74).

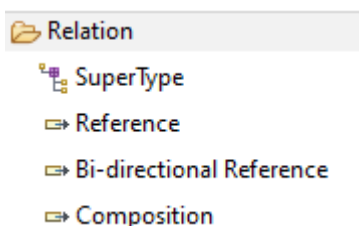


Figure 74 Types of Relation in EMF

The key elements in ResilBlockly Profile Designer are *Class*, *Attribute* and *Relation*, as shown in Figure 75⁶⁵. With a ResilBlockly *Relation* block it is possible to model relations of different types (*Reference*, as default type and as depicted in the figure, or *Composition*).

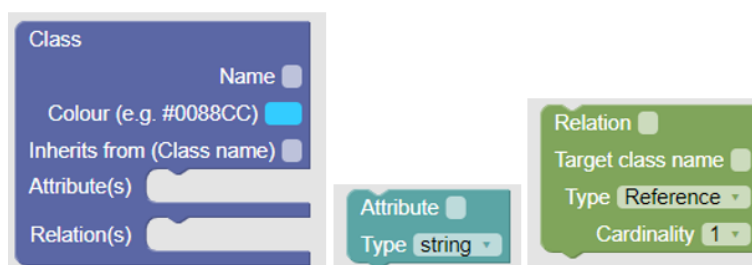


Figure 75 Key elements in ResilBlockly Profile Designer

⁶⁵ actually, there are two additional blocks, namely *Menu* and *Menu Item*, which are not reported in the figure.

6.1.4. Conversion of SoS profile and Import of the ecore

One important activity that has been conducted in order to evolve the tool and complete the transition from Blockly4SoS to ResilBlockly, is the conversion of the AMADEOS SoS profile. All the AMADEOS viewpoints have been reproduced in EMF, generating a single output ecore file. During the process, the Security viewpoint has been deeply reviewed and extended.

Figure 76 shows an example of a small part of the Architecture viewpoint as it appears in EMF, where two of its classes SoS as CS are modelled and connected with a *Composition* relation.

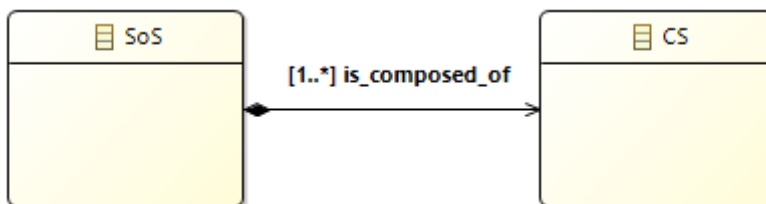


Figure 76 Two of the classes composing the Architecture viewpoint reproduced in EMF

These viewpoints can be imported as ecore into ResilBlockly Profile Designer and are automatically transformed into Blockly blocks of type classes, attributes and relations. A portion of the SoS Profile imported as ecore into ResilBlockly, where the SoS and CS classes have been detailed with other relations, is in Figure 77.

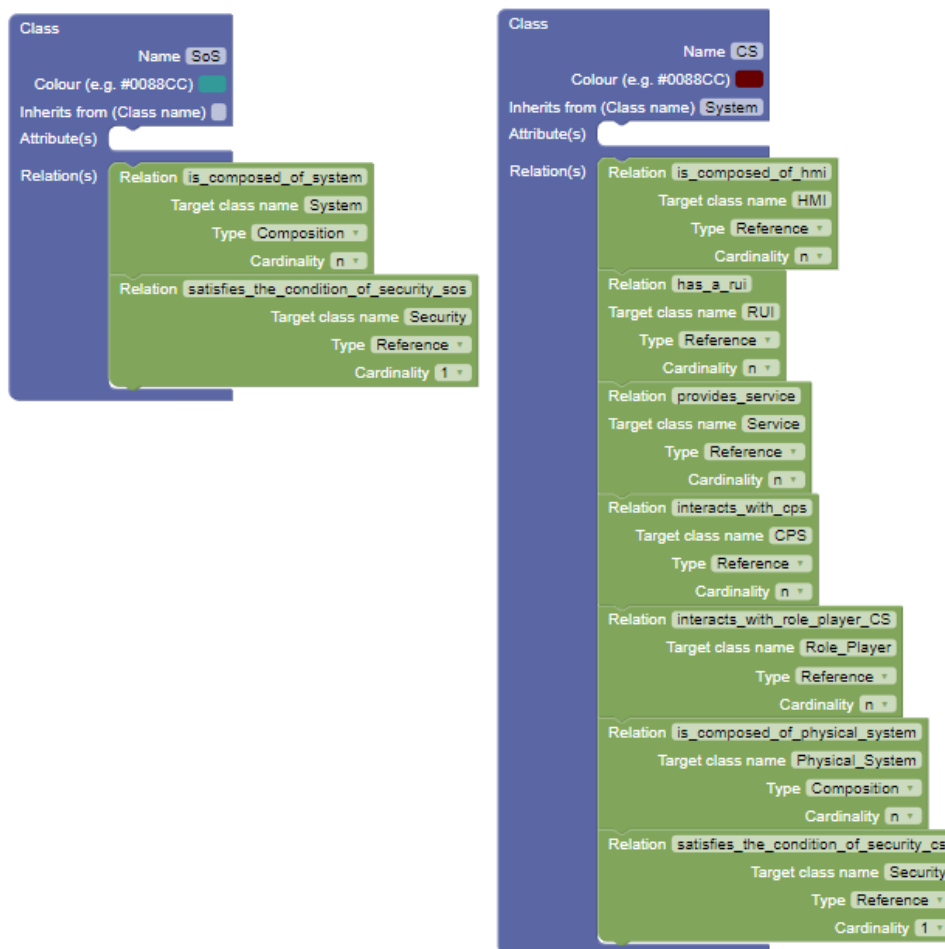


Figure 77 A portion of the SoS Profile imported as ecore into ResilBlockly

The *import ecore* for importing metamodels into the Profile Designer is not the unique functionality implemented that significantly improves the interoperability of ResilBlockly. In fact, the *export ecore* feature in XMI format has been implemented as well, which allows to export the profile as shown in Figure 78.

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="sosProfile"
  nsURI="http://com.sosProfile" nsPrefix="sosProfile">
  <eClassifiers xsi:type="ecore:EClass" name="SoS">
    <eStructuralFeatures xsi:type="ecore:EReference" name="is_composed_by_system"
      upperBound="-1" eType="##/System" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="satisfies_the_condition_of_security_sos"
      eType="##/Security"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="CS" eSuperTypes="##/System">
    <eStructuralFeatures xsi:type="ecore:EReference" name="is_composed_by_hmi" upperBound="-1"
      eType="##/HMI"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="has_a_rui" upperBound="-1"
      eType="##/RUI"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="provides_service" upperBound="-1"
      eType="##/Service"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="interacts_with_cps" upperBound="-1"
      eType="##/CPS"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="interacts_with_role_player_cs"
      upperBound="-1" eType="##/Role_Player"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="is_composed_by_physical_system"
      upperBound="-1" eType="##/Physical_System" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="satisfies_the_condition_of_security_cs"
      eType="##/Security"/>
  </eClassifiers>
```

Figure 78 A portion of the exported ecore XMI showing SoS and CS

Finally, importing and exporting the ecore makes ResilBlockly very interoperable with other tools and it allows the management of different profiles. In fact, on one hand this feature allows a user to design its meta-model within EMF or simply retrieve an existing profile, and then import it into ResilBlockly for later on instantiating the profile and performing analysis and simulations; on the other hand, the user can also design its model directly within ResilBlockly and then export to other tools compatible with ecore XMI.

To test the functionality, an existing ecore metamodel⁶⁶ has been successfully imported into ResilBlockly profile designer.

Other functionalities regarding interoperability of are also implemented (i.e., import/export of Profile Designer workspace and of models created in Model Designer) will be described in the D6.2 together with all the other feature and within the tool's user guide.

6.2. Using the MUD standard for modelling

In order to facilitate both the generation of the MUD and the modelling, ResilBlockly is being extended to include mechanisms both to import the original MUD and to export the extended one. This will imply several benefits for the user:

- The possibility of manually introduce the original MUD aspects
- The possibility of importing the original MUD file just providing the file.
- The possibility to generate the extended version of the MUD with additional aspects considered within ResilBlockly model.

⁶⁶ <https://github.com/DEIS-Project-EU/ODEv2>

Next subsections detail the overview of the import/export of the MUD files that will be further developed and detailed in D6.2.

6.2.1. Importing the original MUD file in ResilBlockly

The user will have two different options to import the original MUD information. On the one hand, ResilBlockly will provide an interface to manually indicate all the information that the standardized MUD file should contain. In particular the user will need to specify the name of the access control rule, the source and destination interfaces for the communication, the source and destination ports and the transport layer protocol (TCP/UDP), as shown in Figure 70. On the other hand, the user can directly import the original MUD file of the components of the model. This information is later integrated in the model to enrich it.

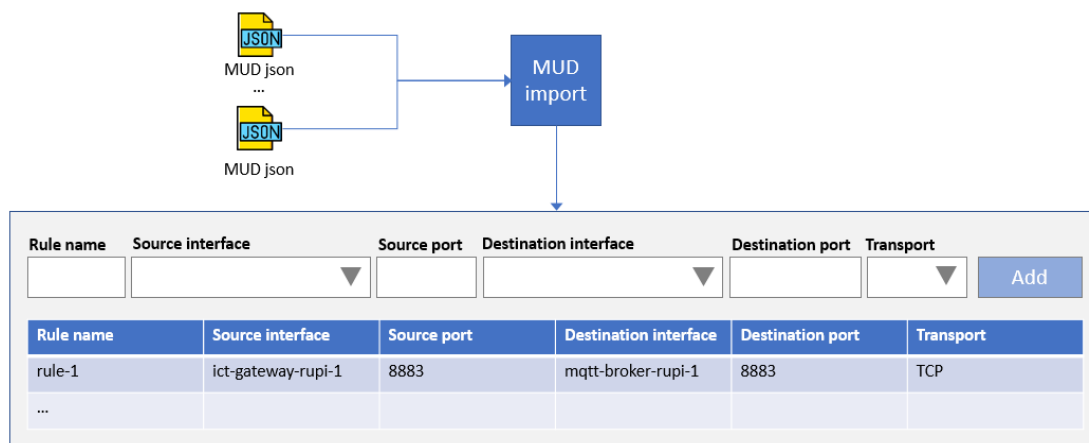


Figure 79 ResilBlockly user interface to import MUD information

6.2.2. Exporting the extended MUD file from ResilBlockly

As discussed before, the MUD model will be enriched to specify additional features from the design phase that can be obtained both from the user of the tool and from the model. In this sense, ResilBlockly will include a mechanism with a user interface that allows the user to add part of the information that compose the extended MUD model and that can be also used to enrich the model of the system. In particular, the cryptographic algorithms used for each communication, a higher set of protocols, the exposed resources in case of HTTP/CoAP, authorization restrictions and limit of the number of communications. The CVE and CWE associated vulnerabilities, as well as their risk, is obtained also from ResilBlockly tool.

6.3. Hazard Analysis in ResilBlockly

A feature that has been implemented to evolve the tool ResilBlockly and to make it able to meet the project goals is the HAZOP Analysis. The HAZOP methodology described in Section 3 has been implemented, starting from a functionality⁶⁷ partially already existing in Blockly4SoS before BIECO, which has been completely re-designed and whose implementation has been deeply refactored in the context of BIECO project.

In order to perform the Functional Analysis or the Interface Analysis, first the Profile Expert user has to create a custom profile, thus meta-modelling components and relations of a

⁶⁷ the so called "Generate Analysis" functionality that is available at the following link and that has been introduced after AMADEOS <https://blockly4sos.resiltech.com/latest/demos/amadeos/i.html>

specific domain by means of *class* and *relation* blocks, together with their *attribute* block, where needed. Then, in order to perform the analyses, the Profile Expert is required to clearly mark which class and relation blocks realize and represent functions or interfaces.

This is another main difference with regard to Blockly4SoS: since ResilBlockly has introduced the Profile Designer functionality, it is not known from the beginning which are the elements composing a profile and, most importantly, which of them have to be considered the target of functional or interface analyses, as instead was possible with the Blockly4SoS SoS profile, which was the only possible profile to be chosen. This is an effort that is required only once, during the design of a profile, and only to the Profile Expert, and is justified by the increase of modelling power of the tool.

6.3.1. Functional Hazard Analysis in ResilBlockly

In the Profile Designer, the Profile Expert starts the so-called *Risk designer* functionality (shown in Figure 80). Then, in the *Functions* tab, the preliminary activity required for enabling the functional analysis, can be performed. Basically, the profile expert user has to appropriately select a Class block and determine which, among all its relation blocks, has to be considered the Functions.

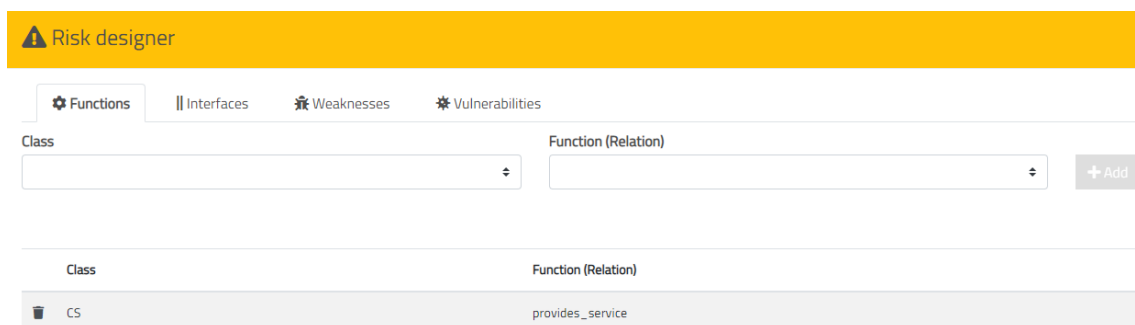


Figure 80 The ResilBlockly Profile Designer - Risk designer GUI with the Functions tab selected and an example of function identified

As said, this is a configuration step, while the actual functional analysis is carried on subsequently by the system designer user. This happens in the Model Designer, with a functionality named *Risk Assessment* (Figure 81). Here, one or more templates for the analysis can be specified, following substitution rules and based on regular expressions. Similarly, a set of keywords can be specified. At this point, the functional analysis is automatically applied to all the functions (relation blocks) that have been specified during the profile design, and for all the keywords in the template.

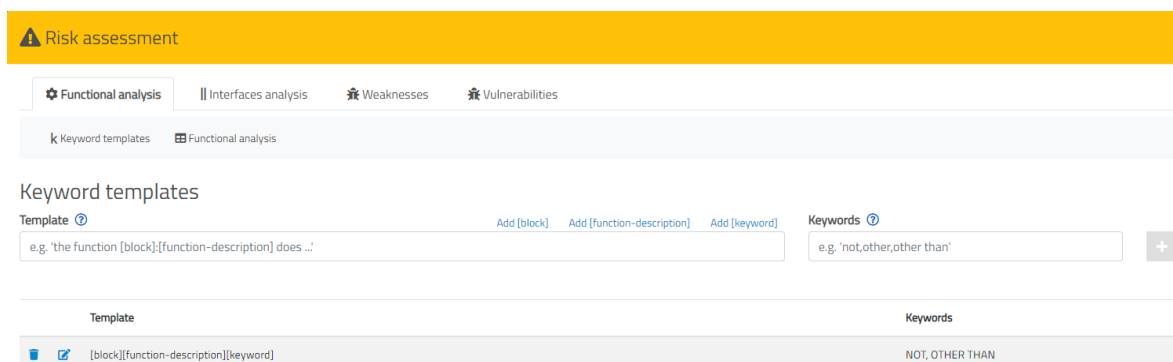
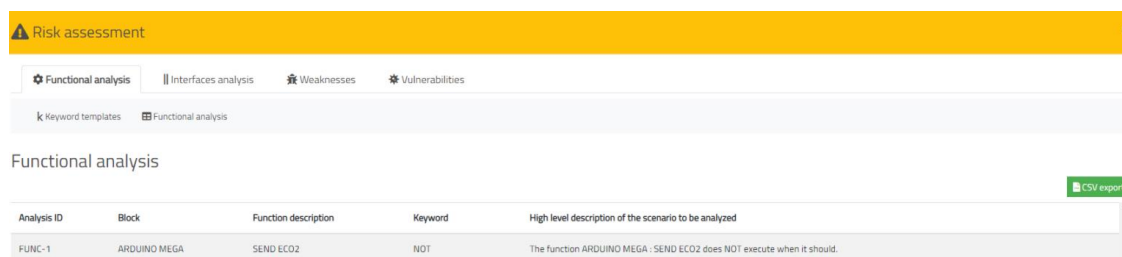


Figure 81 The ResilBlockly Model Designer - Risk Assessment GUI with the functional analysis tab selected and the interface for specifying the template

Figure 82 shows a simple example of this functional analysis. The result of the analysis is provided not only as a visual result but also as a CSV file that can be exported and downloaded. Then, the functional analysis can be completed offline by filling the fields already listed in Section 3 (Table 18 and Table 20), thus concluding the hazard and risk analysis.

This functionality, thanks to the automatic and systematic application of the keywords to all the functions (and interfaces), becomes particularly useful when the modelled system grows in dimension and complexity, and the HAZOP activities that a user typically performs manually, helped only by spreadsheets, is significantly speeded up. Moreover, the risk of forgetting some of the functions to be analysed, is brought to zero if the profile is realized correctly.



Analysis ID	Block	Function description	Keyword	High level description of the scenario to be analyzed
FUNC-1	ARDUINO MEGA	SEND ECO2	NOT	The function ARDUINO MEGA : SEND ECO2 does NOT execute when it should.

Figure 82 The ResilBlockly Model Designer - Risk Assessment GUI with the functional analysis tab selected and the result of a functional analysis

6.3.1. Interface Hazard Analysis

The Interfaces are another type of element of the model that can be systematically analysed for hazards, adopting the Methodology of Section 3.2, which is based on the HAZOP technique introduced in Section 2.1.13. Before to describe how to conduct this analysis within the tool, it is necessary to introduce the recommended approach for designing the interfaces.

6.3.1.1. Interfaces in ResilBlockly Profile Designer

In ResilBlockly, each interface is unidirectional and can be identified by the user with the triple *<Source, Destination, Message>*.

Figure 83 shows a general example where two elements, namely a sensor and a sensor node, respectively can send and receive messages over a channel identified by two interface ends, the Sensor-SensorNode and the SensorNode-Sensor interfaces.

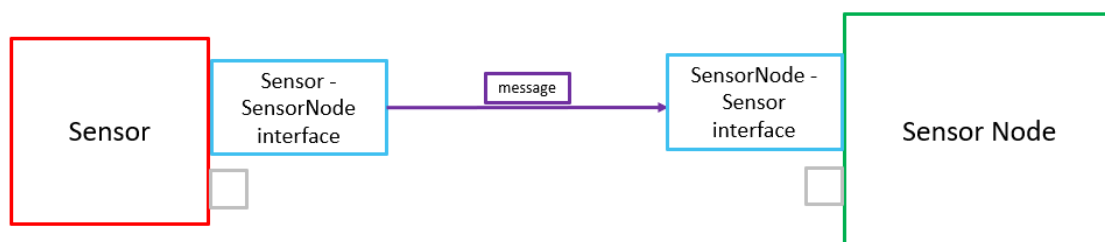


Figure 83 Logical representation of a Sensor Network example with two interfaces

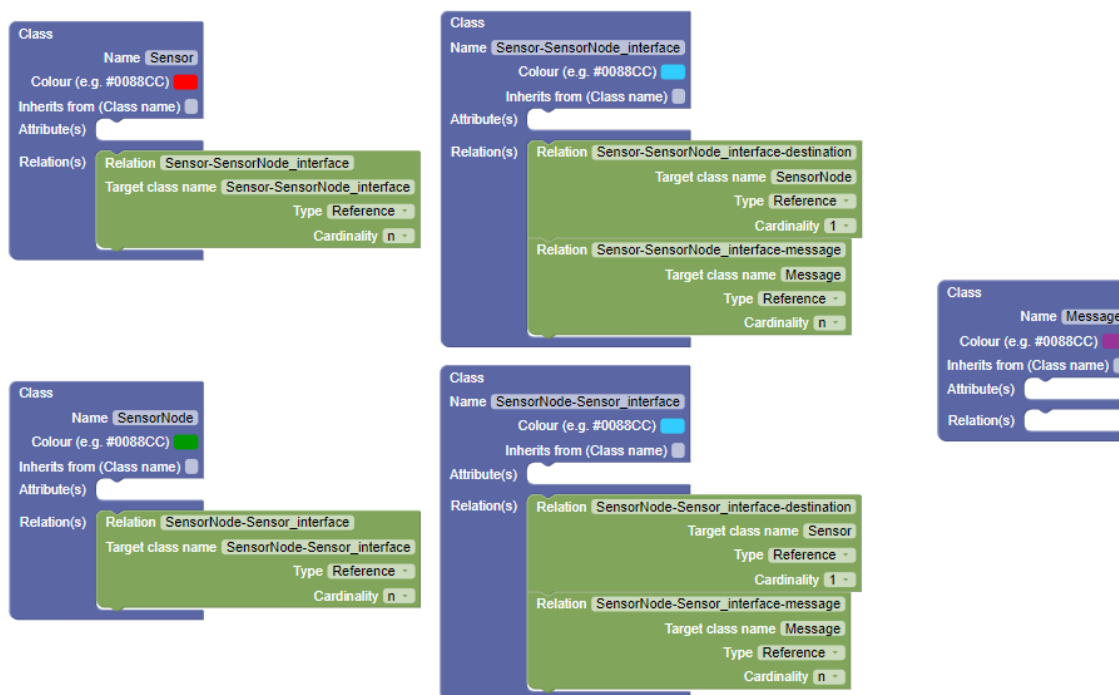


Figure 84 Example of profile with the meta-modelling of interfaces

Figure 84 shows an example of profile which represents the aforementioned sensor network in ResilBlockly making use of Class and Relation blocks.

Analogously to the Functions, in the Profile Designer, after having clicked on Risk designer, the second tab named *Interfaces* (shown in Figure 85) allows the configuration of the interfaces.

Figure 85 The ResilBlockly Profile Designer - Risk Designer GUI with the Interfaces tab selected and the interfaces definition process ongoing

Thus, before being able to conduct the analysis, the profile expert user has to select between the available class blocks which is the *Source*, and then the *Destination* and *Message* both between the available relation blocks. With regard to the above example, Figure 85 shows the process of definition of the Sensor-SensorNode interface, which is the centre-top in Figure 84.

Once the interfaces have been added by clicking on Add, the configuration is concluded and the actual Interface Analysis can be carried out in the Model designer *Risk Assessment*. In Model designer, the system designer user instantiates the profile in a model, e.g., as shown with Figure 86, where a block named *s1*, which is the instance of a Sensor, exposes an interface named *s-sn* which allows it to communicate with a SensorNode called *sn1*, by sending a Message named *msg* towards a destination interface named *sn-s*.



Figure 86 Example of model with modelling simple interfaces

Once a model has been created, the Interface analysis can automatically be generated just after having defined a template for the analysis with a set of appropriate keywords. Figure 87 shows the result for the interface analysis of the above example, where a template composed of two simple keywords, *not* and *corrupted*, has been defined.

Risk assessment					
<div> ⚙️ Functional analysis Interfaces analysis ⚠️ Weaknesses ⚠️ Vulnerabilities </div>					
Analysis ID	Message	Source block	Destination block	Keyword	High level description of the scenario to be analyzed
INTERFACE-1	MSG	S1	SN1	NOT	SN1 does NOT receive MSG.
INTERFACE-2	MSG	S1	SN1	CORRUPTED	SN1 receives CORRUPTED MSG.

Figure 87 The ResilBlockly Model Designer - Risk assessment GUI with the Interfaces tab selected and the result of the analysis

The interfaces hazard analysis functionality of ResilBlockly, as well functional one, does not automatically discover hazards. However, it helps the user in listing all the possible elements (functions/interfaces) of the model and to systematically map them to the customizable set of HAZOP guidewords, according to a customizable template. As for the functional analysis, also here the result is provided as a visual outcome as well as a downloadable CSV, so allowing the system designer to complete the analysis offline.

The pre-filled report in CSV format can be further analysed and completed offline. This step corresponds to the white rectangles of Figure 60.

6.4. Identification of Assets and Threat Modelling in ResilBlockly

This Section describes how the asset identification and threat identification/modelling phases of the methodology described in Section 4.2 and Section 4.3 respectively, have been implemented in ResilBlockly. However, the detailed user guide along with the application of the full methodology to one of the BIECO use cases will be included in deliverable D6.2.

A preliminary and required step for the asset identification is the existence of a profile, either created within the tool or imported as ecore file. Then, in the Profile Designer, the profile expert is allowed to perform the first round of asset identification, which will then be complemented within the Model Designer.

Starting the Risk designer, in the *Weaknesses* and in the *Vulnerabilities* tabs, the user selects a Class block which is implicitly identified as asset under analysis. Then, weaknesses can be searched in the CWE by pressing the *add weakness* button shown in Figure 88 which opens a search interface Figure 89 and allows to find weaknesses, read short descriptions, jump to the CWE catalogue, study the details and finally add the CWE to the asset. Custom weaknesses can be also defined by pressing the dedicated button and filling the required fields.

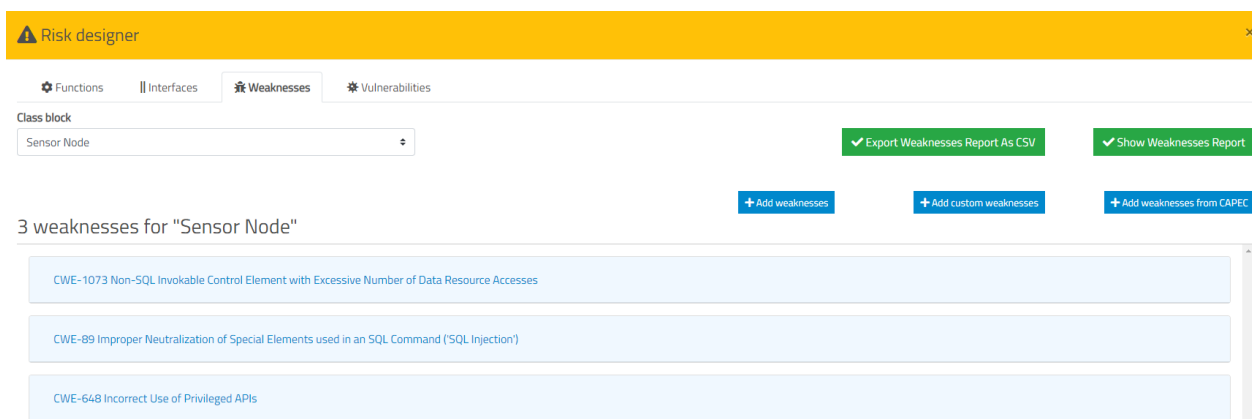


Figure 88 The ResilBlockly Profile Designer - Risk designer GUI with the Weaknesses tab selected and some random weaknesses associated to a sample class block

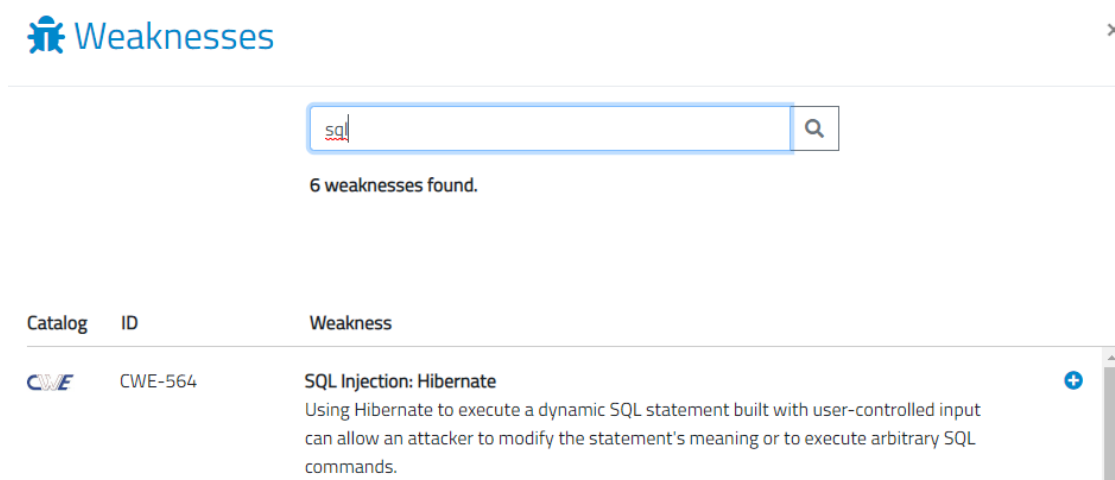


Figure 89 The CWE search interface with a sample example

In addition, CWE weaknesses can be indirectly searched by looking for CAPEC attack patterns (implemented functionality is shown in Figure 90). This can be considered a different approach and starting point for the identification of weaknesses.

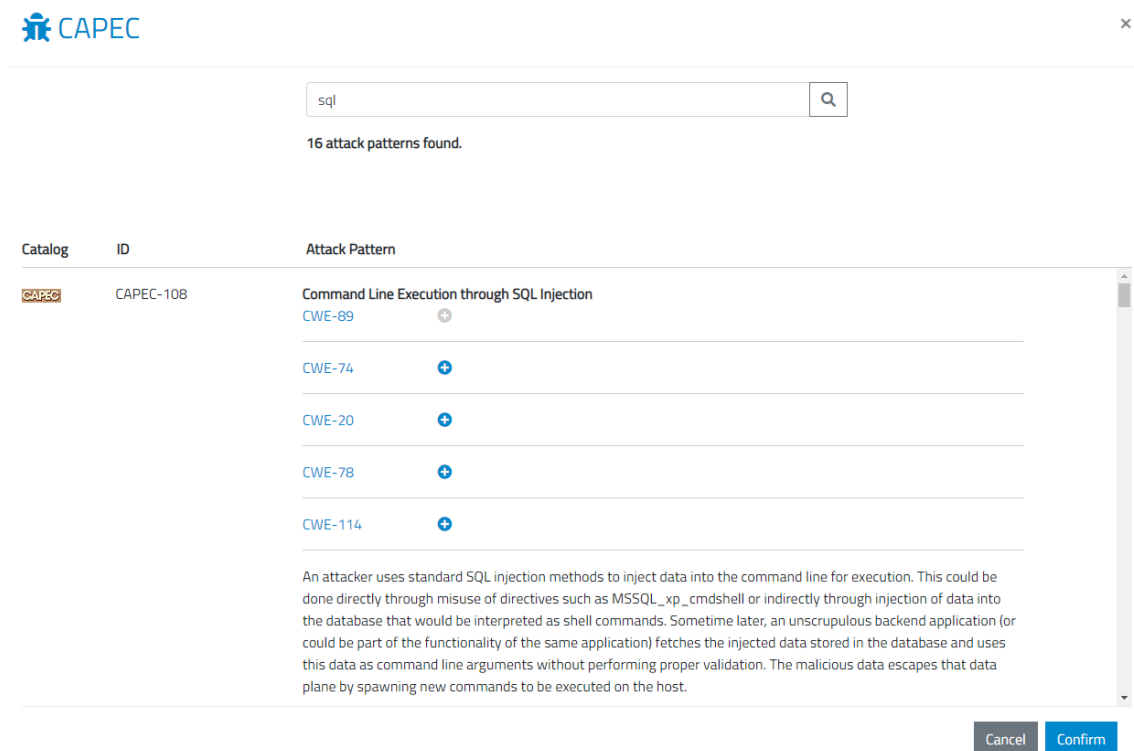


Figure 90 The CAPEC search interface for the retrieval of attack patterns and association of related weaknesses

With a similar approach, the vulnerabilities tab (as shown in Figure 91) enables the identification of assets (class block), research and association of CVE vulnerabilities, or addition of custom defined ones.

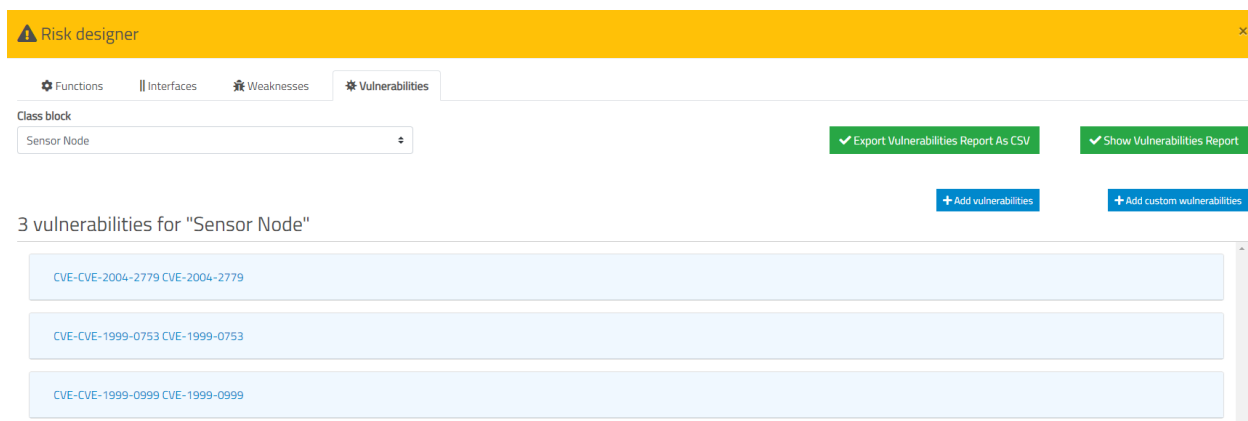


Figure 91 The ResilBlockly Profile Designer - Risk designer GUI with the Vulnerabilities tab selected and some random vulnerabilities associated to a sample class block

Once a profile has been designed, and the weaknesses (vulnerabilities) have been associated to some of its class blocks by the profile expert, they are automatically inherited by each model that instantiates the profile.

The system designer user, in the *Model designer*, by leveraging the so-called Risk assessment functionality, is then able to see weaknesses and vulnerabilities pertaining to

the above-mentioned class blocks and, eventually, to identify new ones. This means that an expert may identify typical threats for a component (interface) of class, and then when the general component is specialized, additional threats which may be specific of the particular type of component, can be identified by the modeller (shown in the third and fourth tab from the left of Figure 87).

As a future improvement, we plan the implementation of a threat identification algorithm which, leveraging the attributes detailed in the profile, can support the user and automatically propose CWE weaknesses and CVE vulnerabilities to be associated.

6.5. Attack Paths and retrieval of additional Threats

Selected graphical representation trees among the ones described in Section 4.4 have been implemented within ResilBlockly in order to enable the visual representation of attack paths and improve the retrieval of weaknesses.

Among the others, the *attack path tree (APT)*, - which becomes an *attack path graph (APG)* in case of weaknesses related to multiple attack patterns, creating loops -, is the best candidate in this sense. In fact, starting from the set of identified CWE weaknesses that have been associated to a system component (interface), and adopting the following algorithm (which refers to the generic APG of Figure 92), the user is able to identify related attack patterns, preceding attack patterns, and additional weaknesses to be considered, analysed and eventually associated:

1. Both in the risk designer and risk assessment, an identified weakness (e.g., CWE-n) is selected;
2. The tool visually represents this weakness (e.g., CWE-n) as root node (Level 0) of an APT;
3. The tool automatically retrieves the *related attack patterns*, (e.g., CAPEC-j, CAPEC-k) where existing, and places them on the Level 1 of the tree.
4. The tool automatically retrieves the *preceding attack patterns* (e.g., CAPEC-i) where existing, thus creating an attack path, and places them one on each level of the three (e.g., Level 2 and subsequent ones)
5. The tool automatically retrieves the *related weaknesses* (e.g., CWE-m, CWE-p, CWE-q, CWE-r) places them on the tree, connecting them to all their related attack patterns. The tree becomes a graph in case loops are created, thus CWE related to multiple CAPECs.

Some of the benefits of the introduction of this feature are:

- to have a visual representation of attack paths potentially connected to and leading to the starting weakness (e.g., CWE-n). It is important since the catalogues structure is very complex, and it may happen to get lost while consulting them.
- being able to directly retrieve and inspect each attack pattern and weakness shown in the graphical representation, leveraging the details from the catalogues.
- identifying and marking, among all the CAPEC and CWE entries in the APT (APG), which ones are relevant for the component (interface) under analysis, that is the one originally associated to CWE-n. In example, the user can grey-out the CWEs or CAPECs considered not relevant for the component (interface) after a careful analysis.

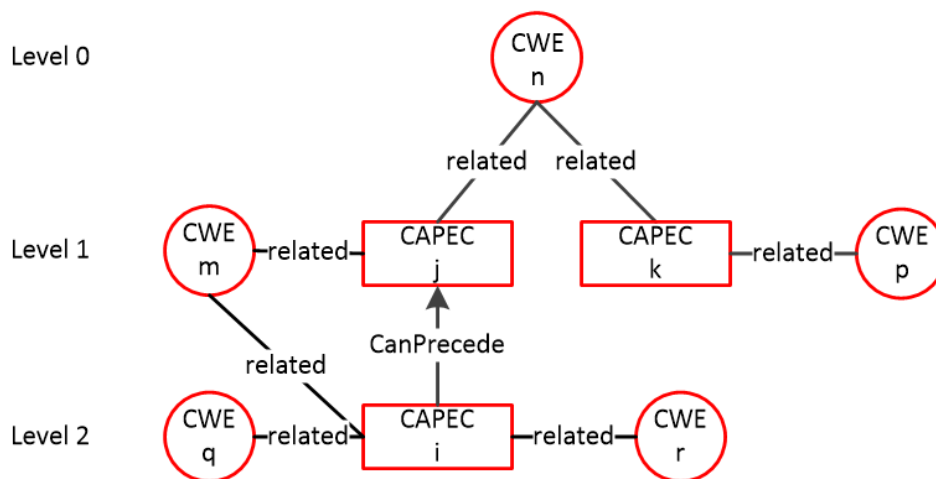


Figure 92 A generic Attack Path Graph

- associating the additional CWEs (remaining not greyed-out) to the component (interface) under analysis.
- according to the definition⁶⁸ of *related weakness* in CAPEC, attack patterns in the APT (APG) have to be considered a threat for the component (interface) if at least one of the related weaknesses exist in the component (interface). That is, a CAPEC whose weaknesses are all excluded (greyed-out) from the APT (APG), can be considered not applicable for the component (interface)

This feature is preliminary to the identification of mitigation strategies and, in particular, it enables to have a first idea of where to place the mitigations, and which weakness, with high priority, have to be eliminated from the component (interface) in order to make unsuccessful the related typical attacks and break the paths towards the exploit.

6.6. Risk Assessment in ResilBlockly

Along with the HAZOP-based Risk assessment that can be conducted by leveraging the functional and interface analysis described in Sections 3 and 6.3, an additional functionality for conducting a risk assessment has been implemented within the tool, according to the methodology introduced in Section 4.

The user, both in the Profile designer (as shown in Figure 93) and in the Model designer, visualizes the CVSS base score for the vulnerabilities associated to the component (interface). This constitutes the severity of impact, which is the first parameter required for the risk assessment (as explained in Section 4 and shown in Figure 68).

Then, in a dedicated window, the likelihood can be inserted, selecting the value from the qualitative scale (i.e., according to the NIST SP800-30 risk matrix of Figure 68; in the future developments, different qualitative scales could be selectable). Finally, the corresponding risk is determined.

The approach is analogous for the weaknesses, with the difference that also the severity of impact is inserted by the user in a dedicated field, and is not determined by retrieving a CVSS base score.

⁶⁸ “[...] each association implies a weakness that must exist for a given attack to be successful. If multiple weaknesses are associated with the attack pattern, then any of the weaknesses (but not necessarily all) may be present for the attack to be successful [...]”.

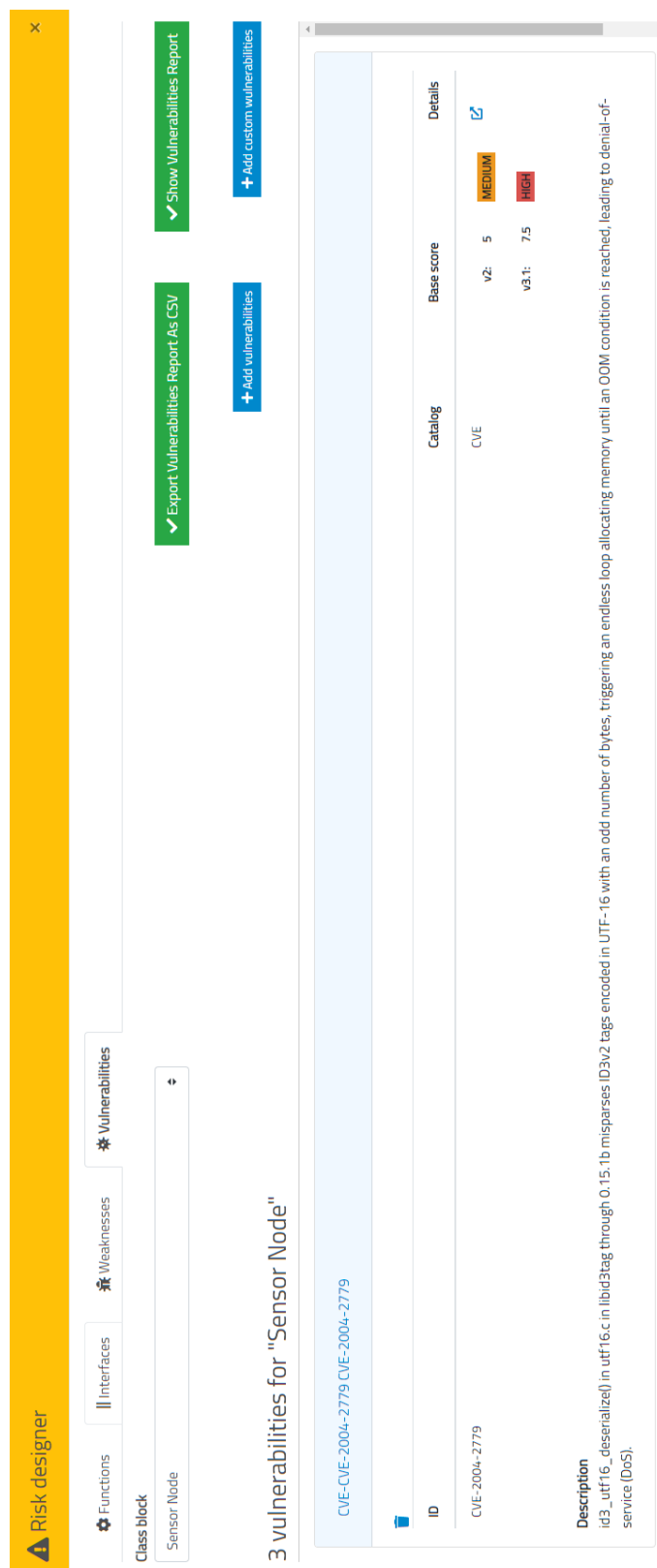


Figure 93 The ResilBlockly Profile Designer - Risk designer GUI with the Vulnerabilities tab with the CVSS base score(s) from the NVD integrated

7. The ResilBlockly Simulation Engine

This section introduces the new simulation engine that has been devised, implemented and integrated with ResilBlockly thanks to which it is possible to represent interactions between system components (e.g., both under normal conditions and during attacks). An overview of the simulation process and of its integration with external IDE and Simulation engine is given in Figure 94.

As introduced in Section 2.5.4, Blockly4SoS was provided with the possibility to simulate the behaviour of model components by specifying some snippets of python code directly into the Blockly4SoS web UI. However, this choice had some drawbacks, i.e., the difficulty in writing the python code into a text area, hence without any code validation, compilation check, import helps, or in general any type of validation that an IDE (Integrated Development Environment) typically offers.

Hence, in the context of the refactoring of Blockly4SoS, a completely new simulation engine has been designed and implemented, and enables to simulate models realized with and exported from ResilBlockly. The engine simulates the behaviour and interactions of model components, based on the messages⁶⁹ exchanged between the interfaces exposed by each component.

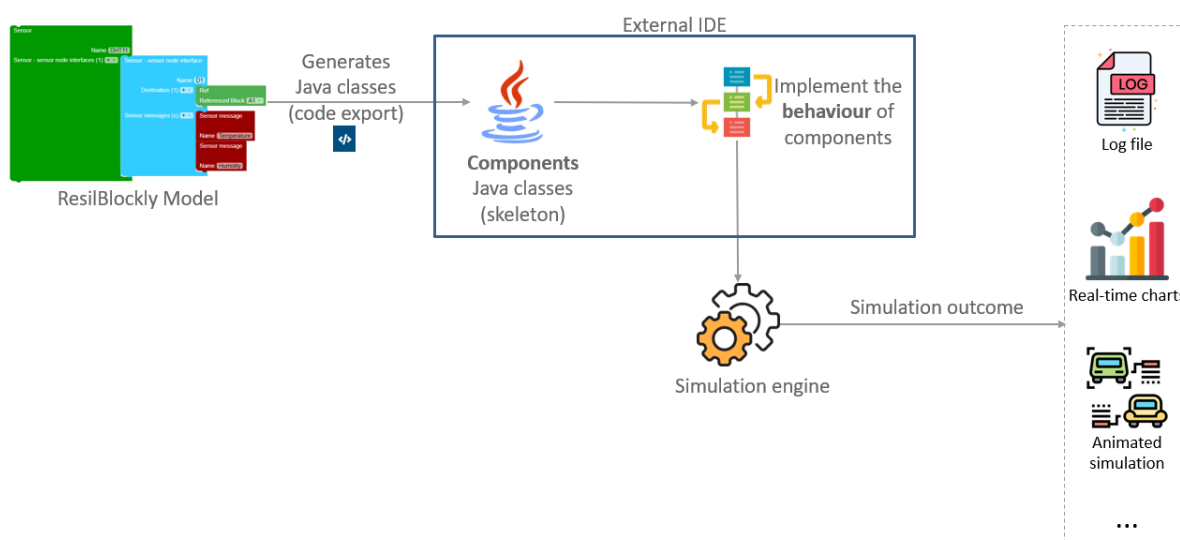


Figure 94 Overview of the simulation process and integration of ResilBlockly model with external IDE and simulation engine

As depicted in Figure 94, the model realized within ResilBlockly can be transformed and exported as Java source code, i.e., a skeleton of the classes corresponding to model elements, relations, attributes, and so on. Then, the code can be imported in an external IDE to be further on elaborated in an environment that offers all the typical features a programmer may need. In the IDE, the behaviour of the components can also be specified.

Then, the simulation engine takes in input the following elements:

- the Java skeleton source code generated from the ResilBlockly Model,
- the behaviour of each ResilBlockly Model Component,

⁶⁹ in principle, the same approach applies also to *things* exchanged over physical interfaces, i.e., RUPIs, in accordance with SoS concepts and AMADEOS profile.

and is able to provide the simulation outcome in many different formats, like:

1. Log file
2. A real-time changing chart
3. A 2D/3D animated simulation
4. Other formats.

The Simulation Engine itself generalizes and standardizes the base structure and behaviour of a model component, and can be used in an external Java Project as a common software interface to implement the specific behaviour of each model component. Indeed, the engine is characterized by an abstract Java class, named `BaseComponent.java`, which constitutes the generic abstraction of a Model Component. A portion of this class is given in Figure 95, while more details will be given in D6.2.

```
public abstract class BaseComponent {
    private final Logger logger = Logf

    protected String id;
    protected String name;
    protected String type;
```

Figure 95 The `BaseComponent.java` class declaration

Each specific Model Component has to extend the `BaseComponent.java` class and override some specific methods; an example of extension of the base component is given in Figure 96, where a class called `DHT11` extends it.

The Java skeleton code of a ResilBlockly Model can be generated and exported from the Model Designer as a compressed archive (.zip file), and is composed of a set of automatically generated Java classes, each of them representing the abstraction of components and interfaces.

```
/**
 * DHT11 component abstraction
 *
 * Auto-generated class, please don't change it manually!
 *💡/
public abstract class DHT11 extends BaseComponent {

    protected String model = "value";

    protected Interface interfaceD1;
```

Figure 96 An example of auto-generated Java Class for a specific Model Component named "DHT11"

The extension of the classes allows to implement specific behaviour for each Model Component. In Figure 97 it is shown the example of a class named `DHT11Behaviour` that extends `DHT11`, and, consequently, also the `BaseComponent`.


```
public class DHT11Behaviour extends DHT11 {
    private static String status;

    @Override
    public void executeBehaviour() {
```

Figure 97 An override example of the executeBehaviour() method

Each auto-generated Java class is provided with some instance attributes -of type Interface (as shown in Figure 96)- which represent and correspond to the interfaces that the related Model Component exposes. The auto-generated Java classes provided by the simulation engine already contain attributes and business logic useful to connect the interface itself with other interfaces, exactly as represented into the ResilBlockly Model.

Figure 98 shows a simple example of a ResilBlockly model with a simple component, a sensor called DHT11, and its interface for communicating with a sensor node.

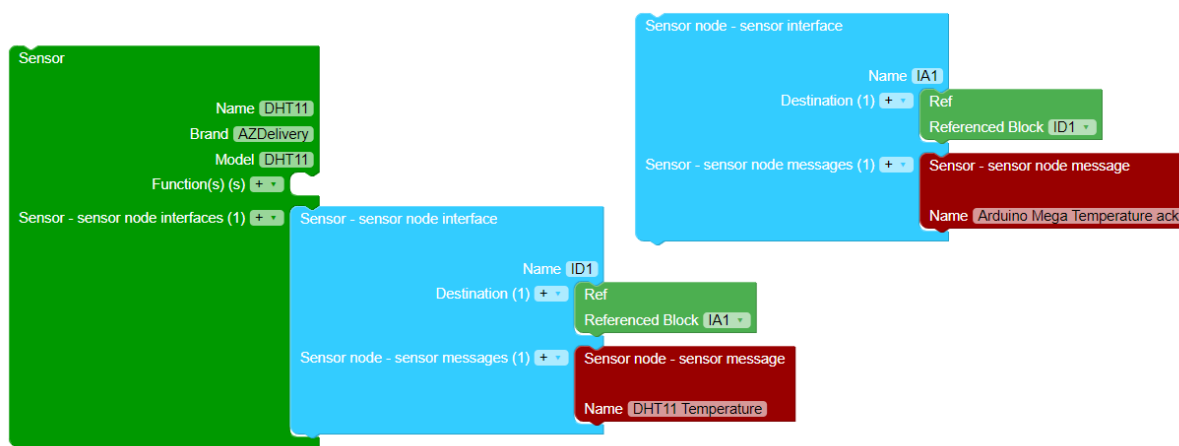


Figure 98 An example of ResilBlockly model

This feature of auto-generating interfaces within the Java code, inheriting information from the model, facilitates the programmer that has in charge the implementation of the component behaviour, since it already provides the interfaces that can be called from a specific component. As shown in Figure 99, each interface object contains information as the `interfaceId` and `interfaceName`, as well as the target unique identifier of the destination interface, i.e., `targetInterfaceId`.

```
@Override
public void initInterfaces() {
    try {
        interfaceD1 = new Interface(
            interfaceId: "b41d4439-0bb2-400a-94bb-f481acf68124",
            interfaceName: "D1",
            targetInterfaceId: "0304b27d-db77-4572-8c53-c3ad41e15a61",
            (String topic, String message) -> {
                onMessageArrived(topic, message);
            }
        );
    } catch (MqttException e) {
    }
}
```

Figure 99 An example of auto-generated interface instance

In the Simulation Engine, the communication between interfaces is based on the MQTT protocol⁷⁰ which is able to decouple them thanks to the Publish/Subscribe paradigm. Indeed, each interface is uniquely identified by a *topic* and at the start of the Simulation Engine all interfaces are *subscribed* to topics related to interfaces to which they are connected into the ResilBlockly Model. Hence, a *publish* is performed every time a *source* interface sends a *message* to the *destination* interface.

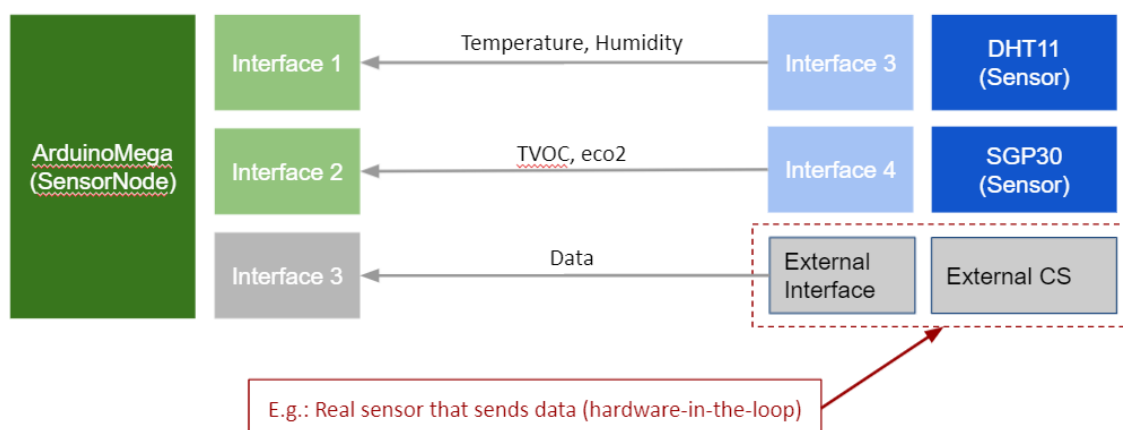


Figure 100 Example of interfaces between simulated and real systems

Thanks to the MQTT protocol, the simulation engine also supports the “hardware-in-the-loop” technique, thus it is possible to simulate a Model in which some components are simulated and others are real ones (as in the example of Figure 100). The unique condition required is the *External Interface* to be connected with a real *External Component* that is able to publish messages on a MQTT broker.

⁷⁰ an OASIS standard messaging protocol for the Internet of Things (IoT) <https://mqtt.org/>



Figure 101 An example of a real time chart, developed as behaviour of a “Dashboard” component, with third-party dependencies

From the behaviour point of view, the Simulation Engine has been designed to be highly flexible, as the User is able to implement every type of behaviour, also using third-party dependencies.

The architectural and functioning concepts behind the simulation engine allow to simulate the interactions between components, e.g., both during nominal behaviour as well as when under particular conditions.

In example, the simulated entity can be an attacker, modelled within ResilBlockly, and whose behaviour is specified within the external IDE; as an alternative, a *real* attack can be introduced targeting one of the components (or better, its interfaces) interacting in the context of the simulation. This allows to observe the interactions between components of a system when attacks are exploited.

Moreover, the intended usage of the ResilBlockly simulation engine within BIECO is simulating the attacks modelled during the threat modelling step (described in Section 6.4), thus targeting the exploit of the weaknesses or vulnerabilities identified and associated to system components (interfaces), to observe the behaviour of the system or components under attack. It is also possible to connect the results of the different simulations with the attack paths graphical representation described in Section 6.5.

Then, by comparing the simulation results obtained from different system models, e.g., according to different system configurations, it will be possible to analyse and understand whether and where to introduce mitigations or which mitigation is more effective.

As stated before, as future improvement for the threat identification and modelling, we plan the implementation of an algorithm which, leveraging the attributes detailed in the profile, can support the user and automatically propose CWE weaknesses and CVE vulnerabilities to be associated.

8. Conclusions

This document reviewed the key concepts regarding modelling of complex systems-of-systems, targeting the representation and analysis of ICT ecosystems and supply chains components. It reviewed and compared many existing solutions, tools and methods for modelling, identifying and representing potential threats, as well as for analysing the intrinsic security risks according to the reference standards. The results of these review, analysis and comparison of the state-of-the-art constitute the first important set of artefacts provided by D6.1.

It has been proposed and described a HAZOP-based hazard analysis and risk assessment methodology thanks to which a user can systematically identify, represent and later on analyse, already in the early prototyping, the safety hazards matched to system functions and interfaces.

Then, a threat identification and modelling methodology has been devised in order to support the phases of a security risk assessment process, from the identification of assets and threats (potential weaknesses and vulnerabilities affecting system components and interfaces), to the determination of impact, likelihood and risk, going through the analysis of attack paths.

In addition, it has been presented how the MUD-compliant specification of network policies has been integrated in ResilBlockly, analysing its limitations and providing specific characteristics and features from the modelling and analysis activities that could be integrated in an extended MUD file. The details of the extension, as well as its application to a specific use case, will be further provided in D6.2. Moreover, the extended MUD will be generated from the information provided in ResilBlockly, so it can be linked to the ICT component and obtained during the runtime.

Furthermore, a preliminary description of ResilBlockly is provided, highlighting the main differences and improvements with regard to its previous version, Blockly4SoS, and how the proposed methodologies are implemented within the tool. A more detailed user guide will be given in deliverable D6.2 in the context of an early validation over the ICT Gateway use case. The ResilBlockly tool itself, together with the methodologies that it integrates, are doubtless the main artefacts of T6.1. Finally, another significant artefact is the completely new simulation engine that has been designed and implemented, and which enables to simulate the models realised with and exported from ResilBlockly.

Thanks to these results, it is possible to identify, determine and associate, already during early prototyping, the weaknesses, vulnerability and safety-hazards of an ICT system or component and to identify which are the weakest elements in the supply chain. The results of the risk analysis enable to understand where attackers will likely try to conduct an attack, and which path would let them to achieve their goal or exploit other weaknesses.

The methodologies, assisted by the tool, constitute an effective solution for the risk determination and analysis. The effectiveness will be demonstrated with the validation activities conducted within T6.2 and WP8 over the BIECO use cases.

Once the risk analysis is performed, these activities can later on and further complemented with the definition of mitigation strategies and measures (addressed in T6.3) which can eliminate the identified weaknesses and significantly reduce the risks.

9. References

- [1] Threat Model, Wikipedia
https://en.wikipedia.org/w/index.php?title=Threat_model&oldid=983957746
- [2] B. Schneier - *Dr. Dobbs's Journal*, December 1999 - "Modelling security threats" -
https://www.schneier.com/academic/archives/1999/12/attack_trees.html.
- [3] T. R. Ingoldsby "Attack Tree-based Threat Risk Analysis"
<https://www.amenaza.com/downloads/docs/AttackTreeThreatRiskAnalysis.pdf>.
- [4] Clemen, RT. 1996. Making hard decisions. 2nd Ed. Duxbury Press
- [5] Modarres, M. 1993. What every engineer should know about reliability and risk analysis. Marcel Dekker, Inc., New York, New York
- [6] ADTool, Université du Luxembourg - Security and Trust of Software System
<https://satoss.uni.lu/members/piotr/adtool/>
- [7] Kordy, B., Mauw, S., Radomirović, S., & Schweitzer, P. (2010, September). Foundations of attack-defense trees. In *Int. Workshop on Formal Aspects in Security and Trust* (pp. 80-95). Springer
- [8] Isograph Attack Tree <https://www.isograph.com/>.
- [9] Wang, P., & Liu, J. C. (2014). Threat analysis of cyber-attacks with attack tree+. *Journal of Information Hiding and Multimedia Signal Processing*, 5(4).
- [10] RiskTree, 2T Security Ltd <https://risktree.2t-security.co.uk/>
- [11] Risk Management with RiskTree, 2T Security Ltd https://www.2t-security.com/papers/RiskTree_overview.pdf
- [12] Getting started with the Threat Modelling Tool, Microsoft Corporation <https://docs.microsoft.com/it-it/azure/security/develop/threat-modelling-tool-getting-started>
- [13] ThreatModeler - <http://threatmodeler.com/>
- [14] IriusRisk - Getting Started <https://support.iriusrisk.com/hc/en-us/articles/360021517751>
- [15] I. Tarandach, PyTM: A Pythonic framework for threat <https://github.com/izar/pytm>
- [16] CAPEC, Organization Usage, <https://capec.mitre.org/community/usage.html>
- [17] securiCAD, foreseeti <https://foreseeti.com/>
- [18] Elahi, G., Aratyn, T., Sivarajan, R., Sethi, R., & Eric, S. K. (2011). SD Elements: A Tool for Secure Application Development Management. In *CAISE Forum* (pp. 81-88).
- [19] Security Compass <https://www.securitycompass.com/sdelements/how-it-works/>
- [20] Threat Dragon, OWASP <https://docs.threatdragon.org/>
- [21] Schaad, A., & Reski, T. (2019). "Open Weakness and Vulnerability Modeler"(OVVL)-An Updated Approach to Threat Modelling.
- [22] The CORAS Method and Tool, CORAS <http://coras.sourceforge.net/>
- [23] Vraalsen, F., Den Braber, F., Lund, M. S., & Stølen, K. (2005, May). The CORAS tool for security risk analysis. In *Int. Conf. on Trust Management* (pp. 402-405). Springer, Berlin, Heidelberg
- [24] Kristian Beckers, Ketil Stølen - ISMS-CORAS: A Structured Method for Establishing an ISO 27001 Compliant Information Security Management System - Chapter · January 2014 DOI: 10.1007/978-3-319-07452-8_13
- [25] Manufacturing Technology Committee -Risk Management Working Group - Hazard & Operability Analysis (HAZOP) - https://pqri.org/wp-content/uploads/2015/08/pdf/HAZOP_Training_Guide.pdf
- [26] LeMay, E., Ford, M. D., Keefe, K., Sanders, W. H., & Muehrcke, C. (2011, September). Model-based security metrics using adversary view security evaluation (advise). In *2011 Eighth International Conference on Quantitative Evaluation of SysTems* (pp. 191-200). IEEE.
- [27] STRIDE Threat Model: Example & Overview <https://study.com/academy/lesson/stride-threat-model-example-overview.html>.
- [28] DREAD risk assessment model, Wikipedia
[https://en.wikipedia.org/wiki/DREAD_\(risk_assessment_model\)](https://en.wikipedia.org/wiki/DREAD_(risk_assessment_model))
- [29] A. Lamba, A., Singh, S., Balvinder, S., & Rela, S. (2015). To Classify Cyber-Security Threats in Automotive Domains Using Different Assessment Methodologies. *International Journal For Technological Research In Engineering*, 3(3).
- [30] Morana, M. M., & Vélez, T. U. (2015). *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. John Wiley & Sons, Incorporated.
- [31] Cleland-Huang, J. How Well Do You Know Your Personae Non Gratae? *IEEE Software*. Volume 31. Number 4. July 2014. Pages 28-31.
- [32] Mead, N. R., Shull, F., Vemuru, K., & Villadsen, O. (2018). A hybrid threat modeling method. *Carnegie Mellon University-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002*.
- [33] P. Saitta, B. Larcom, and M. Eddington - Trike v.1 Methodology Document.
https://www.octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf.
- [34] What is LINDDUN - <https://www.linddun.org/>
- [35] LINDDUN Framework <https://www.linddun.org/linddun>.
- [36] Common Vulnerability Scoring System version 3.1: Specification Document, FIRST- Forum of Incident Response and Security Teams <https://www.first.org/cvss/specification-document>

- [37] Shevchenko, N., Chick, T. A., O'Riordan, P., Scanlon, T. P., & Woody, C. (2018). Threat modelling: a summary of available methods. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.
- [38] Mead, N. R., Shull, F., Vemuru, K., & Villadsen, O. (2018). A hybrid threat modelling method. Carnegie Mellon University-Software Engineering Institute-Technical Report-CMU/SEI-2018-TN-002.
- [39] OCTAVE Method of Security Assessment, The University of Kansas <https://technology.ku.edu/octave-method-security-assessment>
- [40] Caralli, R. A., Stevens, J. F., Young, L. R., & Wilson, W. R. (2007). *Introducing octave allegro: Improving the information security risk assessment process*. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.
- [41] Denning, T., Friedman, B., & Kohno, T. (2013). The Security Cards: A Security Threat Brainstorming Toolkit. *Univ. of Washington*, <http://securitycards.cs.washington.edu>.
- [42] Joyce, James (2003), "Bayes' Theorem", in Zalta, Edward N. (ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2019 ed.), Metaphysics Research Lab, Stanford University, retrieved 2020-01-17
- [43] Ronald S. Ross. NIST, Guide for conducting risk assessments. NIST Special Publication 800-30 revision 1. Technical report, US Dep. Of Commerce, 2012
- [44] Lautenbach, A., & Islam, M. (2016). HEAVENS-HEALING Vulnerabilities to ENhance Software Security and Safety. *The HEAVENS Consortium (Borås SE)*.
- [45] Microsoft Security Development Lifecycle (SDL) - <https://www.microsoft.com/en-us/securityengineering/sdl/>
- [46] Kozar, K. A. (1997). The technique of data flow diagramming.
- [47] The STRIDE per Element Chart, 2007 <https://www.microsoft.com/security/blog/2007/10/29/the-stride-per-element-chart/>.
- [48] M. Rohr Microsoft's new Threat Modeling Tool, 2016. <https://blog.secdisc.com/2016/07/06/microsofts-new-threat-modelling-tool/>.
- [49] CVE - Common Vulnerabilities and Exposure. MITRE Corporation. <https://cve.mitre.org/>
- [50] Official Common Platform Enumeration (CPE) Dictionary, National Vulnerability Database, 2018d.- <https://nvd.nist.gov/products/cpe>
- [51] NVD - National Vulnerability Database, 2018a - <https://nvd.nist.gov/>
- [52] OVVL <https://ovvl.org/index.html#home>
- [53] VAST Modelling Methodology <https://threatmodeler.com/threat-Modelling-methodologies-vast/>
- [54] CAPEC - Common Attack Pattern Enumeration and Classification - <https://capec.mitre.org/>
- [55] Cyber Threat Modelling - <https://www.eccouncil.org/threat-modeling/>.
- [56] Alberts, C.; Dorofee, A.; Stevens, J; & Woody, C. Introduction to the OCTAVE Approach. Software Engineering Institute, Carnegie Mellon University. August 2003. <https://resources.sei.cmu.edu/library/Asset-view.cfm?assetid=51546>
- [57] Security cards information sheet <http://securitycards.cs.washington.edu/assets/security-cards-information-sheet.pdf>
- [58] Mead, N. R., & Stehney, T. (2005). Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes*, 30(4), 1-7.
- [59] securiCad <https://community.securicad.com/using-the-example-model/>
- [60] CENZIC, HARM, The OWASP Foundation - Approaches to Quantitative Risk Analysis for Web Applications.
- [61] CWSS - Common Weakness Scoring System - Scoring CWEs - https://cwe.mitre.org/cwss/cwss_v1.0.1.html
- [62] Veracode <https://help.veracode.com/reader/DGHxSJy3Gn3gtuSIN2jkRQ/oOthQ0PflS7hcwPMwRhYRw>
- [63] Microsoft Documentation, STRIDE Threats in Commerce Server [https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee810587\(v=cs.20\)](https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee810587(v=cs.20))
- [64] FFRI Inc., STRIDE variants and security requirements-based threat analysis https://www.ffri.jp/assets/files/monthly_research/MR201610_STRIDE_Variants_and_Security_Requirements-based_Threat_Analysis_ENG.pdf
- [65] R. A. Caralli, J. F. Stevens, L. R. Young, y W. R. Wilson, «Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process», CERT, 2007.
- [66] C. J. Alberts, A. J. Dorofee, J. F. Stevens, y C. Woody, «OCTAVE-S Implementation Guide, Version 1», 2005.
- [67] Hellesen, N., Torres, H., & Wangen, G. (2018). Empirical case studies of the root-cause analysis method in information security. *International Journal On Advances in Security*, 11.
- [68] J. R. C. Nurse, S. Creese, y D. D. Roure, «Security Risk Assessment in Internet of Things Systems», *IEEE Comput. Soc. IT Pro*, 2017.
- [69] OWASP Top Ten Project <https://www.owasp.org/index.php/Category:OWASP\ Top\ Ten\ Project>
- [70] Microsoft, «DREAD scheme», 2010. [https://docs.microsoft.com/en-us/previous-versions/msp-np/ff648644\(v=pandp.10\)#dread](https://docs.microsoft.com/en-us/previous-versions/msp-np/ff648644(v=pandp.10)#dread).

- [71] Baron, A., Babiceanu, R. F., & Seker, R. (2018, April). Trustworthiness requirements and models for aviation and aerospace systems. In *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)* (pp. 1B3-1). IEEE.
- [72] NCCgroup, «Threat prioritisation: DREAD is dead, baby?», 2016 <https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2016/march/threat-prioritisation-dread-is-dead-baby/>.
- [73] OWASP, *OWASP Application Security Verification Standard (ASVS) Project*
- [74] Mobius, Model-Based Environment for Validation of System Reliability, Availability, Security, and Performance <https://www.mobius.illinois.edu/>
- [75] R. M. R. K., «Security risk assessment of Geospatial Weather Information System (GWIS) using integrated CVSS approach», *Int. J. Adv. Comput. Sci. Appl.*, vol. 1, n.º 3, 2010.
- [76] Potteiger, B., Martins, G., & Koutsoukos, X. (2016, April). Software and attack centric integrated threat modeling for quantitative risk assessment. In *Proceedings of the Symposium and Bootcamp on the Science of Security* (pp. 99-108).
- [77] Trike website <https://www.octotrike.org/>
- [78] Trike implementation <https://sourceforge.net/projects/trike/>
- [79] Ford, M. D., Keefe, K., LeMay, E., Sanders, W. H., & Muehrcke, C. (2013, June). Implementing the ADVISE security modeling formalism in Möbius. In *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (pp. 1-8). IEEE.
- [80] Personas: An Agile Introduction, Agile Modeling <http://www.agilemodeling.com/artifacts/personas.htm>
- [81] IEC-International Electrotechnical Commission. (2001). IEC 61882, Hazard and operability studies – Application guide.
- [82] Appicharla, S. K. (2015). Application of Cognitive Systems Engineering Approach to Railway Systems (System for Investigation of Railway Interfaces). *Railway Research: Selected Topics on Development, Safety and Technology*, 81.
- [83] ISO/IEC, "Information technology – Security Techniques-Information security risk management" ISO/IEC FIDIS 27005:2008.
- [84] Joint Task Force - SP 800-53 Rev. 5 Security and Privacy Controls for Information Systems and Organizations - <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>
- [85] Stouffer, K. A., Falco, J. A., & Scarfone, K. A. (2011). NIST SP 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc).
- [86] Common Weakness Enumeration - <https://cwe.mitre.org/>
- [87] NISTIR 7628 Rev. 1 - The Smart Grid Interoperability Panel–Smart Grid Cybersecurity Committee - Guidelines for Smart Grid Cybersecurity
- [88] NIST Special Publication 1108r3 - The Smart Grid Interoperability Panel–Smart Grid Cybersecurity Committee - NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0
- [89] ISA Intech magazine. New ISA/IEC 62443 standard specifies security capabilities for control system components - <https://www.isa.org/intech-home/2018/september-october/departments/new-standard-specifies-security-capabilities-for-c>
- [90] NIST - SP 800-53A Rev. 4 - Joint Task Force Transformation Initiative - Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans
- [91] NIST Special Publication 800-53 Rev. 5 - Joint Task Force - Security and Privacy Controls for Information Systems and Organizations
- [92] Quick Start Guide: An Overview of ISA/IEC 62443 Standards Security of Industrial Automation and Control Systems – ISA.org <https://qca.isa.org/hubfs/ISAGCA Quick Start Guide FINAL.pdf>
- [93] Cybersecurity & Infrastructure Security Agency (CISA), Information and Communications Technology (ICT) Supply Chain Risk Management (SCRM) <https://www.cisa.gov/supply-chain>
- [94] IEC61882: 2002 Hazard and operability studies (HAZOP studies)-Application Guide
- [95] Bartnes, M. et al (2006). Safety vs. security? Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management May 14-18, 2006, New Orleans, Louisiana, USA.
- [96] CENELEC EN 50129:2018: Railway Applications -Communication, signaling and processing systems, Safety related electronic systems for signalling.
- [97] Miller, C., & Valasek, C. (2015). Remote exploitation of an unaltered passenger vehicle. *Black Hat USA, 2015*(S 91).
- [98] Nie, Sen, Ling Liu, and Yuefeng Du. "Free-fall: Hacking tesla from wireless to can bus." Briefing, Black Hat USA 25 (2017): 1-16.
- [99] S. Alvarez 2020, "Tesla employee foregoes \$1M payment, works with FBI to thwart cybersecurity attack" <https://www.teslarati.com/tesla-employee-fbi-thwarts-russian-cybersecurity-attack/>
- [100] RASEN Compositional Risk Assessment and Security Testing of Networked Systems FP7-ICT G.a. ID: 316853 <http://www.rasenproject.eu/>

- [101] ARMOUR Large-Scale Experiments of IoT Security Trust H2020-EU.2.1.1. G.a. ID: 688237 <http://armour-project.eu/>
- [102] UcedaVélez, T. Threat Modeling w/PASTA: Risk Centric Threat Modeling Case Studies. Technical Report. Open Web Application Security Project (OWASP). 2017.
- [103] Bondavalli, A., Bouchenak, S., & Kopetz, H. (Eds.). (2016). Cyber-physical systems of systems: foundations—a conceptual model and some derivations: the AMADEOS legacy (Vol. 10099). Springer.
- [104] AMADEOS EU FP7-ICT-2013.3.4 Project: Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems <http://amadeos-project.eu/>. GA no. 610535.
- [105] BIECO D3.1 "Report on the State of the Art of Vulnerability Management", Feb. 2021.
- [106] OMG® Unified Modeling Language® (OMG UML®). Version 2.5.1. formal/2017-12-05. December 2017.
- [107] OMG Systems Modeling Language (OMG SysML), <http://www.omg-sysml.org/>.
- [108] J.S. Dahmann and K.J. Baldwin, "Understanding the Current State of US Defense Systems-of-Systems and the Implications for Systems Engineering," 2nd IEEE International Systems Conference (SysCon), pp. 1-7, 2008.
- [109] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Trans. on Dependable and Secure Computing, vol. 1, no. 1, Jan. –Mar. 2004.
- [110] <https://github.com/AMADEOSConceptualModel/SysMLProfileAndApplication.git> - GitHub public link to the AMADEOS SysML profile and application.
- [111] AMADEOS Supporting Facilities User GUIDE <https://blockly4sos.resiltech.com/user-guide.pdf>
- [112] EMF documentation www.eclipse.org/emf/docs.php
- [113] Fundamentals of EMF https://www.eclipse.org/modeling/emf/docs/presentations/EclipseCon/EclipseCon2008_309T_Fundamentals_of_EMF.pdf
- [114] EMF Ecore Javadoc <https://download.eclipse.org/modeling/emf/emf/javadoc/2.10.0/org.eclipse.emf.ecore/package-summary.html>
- [115] EMF tutorial <https://www.vogella.com/tutorials/EclipseEMF/article.html>
- [116] Sion, L.; Yskout, K.; Van Landuyt, D.; & Joosen, W. Solution-aware data flow diagrams for security threat modeling. Pages 1425-1432. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. April 2018. DOI 10.1145/3167132.3167285.
- [117] Macher, G., Armengaud, E., Brenner, E., & Kreiner, C. (2016, September). A review of threat analysis and risk assessment methods in the automotive context. In *International Conference on Computer Safety, Reliability, and Security* (pp. 130-141). Springer, Cham.
- [118] NVD vulnerability status <https://nvd.nist.gov/vuln/vulnerability-status>
- [119] Aksu, M. Ugur, et al. "A quantitative CVSS-based cyber security risk assessment methodology for IT systems." *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2017.
- [120] NISTIR 7788 Singhal, A., & Ou, X. (2017). Security risk analysis of enterprise networks using probabilistic attack graphs. In *Network Security Metrics* (pp. 53-73). Springer, Cham.
- [121] Kotzanikolaou, P., Theoharidou, M., & Gritzalis, D. (2013, March). Cascading effects of common-cause failures in critical infrastructures. In *International Conference on Critical Infrastructure Protection* (pp. 171-182). Springer, Berlin, Heidelberg.
- [122] Stergiopoulos, G., Kouktzoglou, V., Theoharidou, M., & Gritzalis, D. (2017). A process-based dependency risk analysis methodology for critical infrastructures. *International Journal of Critical Infrastructures*, 13(2-3), 184-205.
- [123] DANSE Consortium: DANSE Methodology V2-D_4.3. <https://www.danse-ip.eu>
- [124] COMPASS, Guidelines for Architectural Modelling of SoS. Technical Note Number: D21.5a Version: 1.0, September 2014. <http://www.compass-research.eu>
- [125] Jones, C., et al.: Final version of the DSoS conceptual model. DSoS Project (IST-1999-11585) (2002)
- [126] ENISA Threat and Risk Management Glossary <https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary>
- [127] B. Weyl, O. Henniger, A. Ruddle, H. Seudié, M. Wolf, and T. Wollinger: Securing vehicular on-board IT systems: The EVITA Project. In 25th Joint VDI/VW Automotive Security Conference, Ingolstadt, Germany, October 2009
- [128] Lautenbach, A., & Islam, M. (2016). HEAVENS—HEaling Vulnerabilities to ENhance Software Security and Safety. *The HEAVENS Consortium (Borås SE)*.
- [129] ETSI EG 203 251 V1.1.1 (2016-01) Methods for Testing & Specification; Risk-based Security Assessment and Testing Methodologies.
- [130] E. Lear, D. Romascanu, and R. Droms, "Manufacturer Usage Description Specification (RFC 8520)," 2019. [Online]. Available: <https://tools.ietf.org/html/rfc8520>

Appendix A – Comparison Between Threat Modelling Tools

Table 27 Tools based on Attack Tree Methodology

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
ADTool	threats, attack tree, numerical and label values (e.g., probability of success, difficulty, time, skill level, etc.) of atomic actions (leaves)	attack tree creation, defence tree creation, security analysis, "risk assessment", quantitative analysis, large-scale printing	model, numerical and label values (e.g., probability of success, difficulty, time, skill level, etc.) of non-atomic actions (internal nodes)	pdf, png, jpeg, tex, XML	none	Latest update on GitHub: 2017. Latest publication: 2016	Attack-Defense Trees Tool	Open	University of Luxembourg	1 2 3
AttackTree+	threats, attack tree	attack tree modelling and analysis, mitigation tree, threat analysis, definition of indicators (e.g., attack cost, difficulty, frequency, cut set, etc.), modelling and analysis of consequences	attack tree, fault tree, and mitigation tree diagrams, multiple type of analysis outcomes	pdf, XML, Access, SQL Server, Excel, CSV, etc. (also as input)	"ISO 26262" J3061	up to date (2020)	A tool included in a wide suite of softwares. Coming from dependability/safety area. <i>Existing since 1980s and applied in thousands of companies. Certified according to ISO 9001, Cyber Essentials, and SAP</i>	Commercial	Isograph	1 Attack Tree Webinar (from 14:56): 2
RiskTree	threats, attack tree	risk tree modelling, risk assessment, risk prioritization	risk tree, risk charts (e.g., pyramid, disk+, spider diagram+, riskmap+, prioritized risk table), threats charts	JSON, (also for input), Mindmap XML (tree), CSV, pdf, (full report)	ISO 27001 (if extended in a more complete risk management tool)	up to date (2020)	can be extended in the so-called RiskWiki. Countermeasures can be mapped against controls (such as ISO27001),	Commercial	2T Security	1

Table 28 Tools based on STRIDE methodology

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
Threat Modeling Tool	DFD	template, identify and mitigate potential security issues, intuitive user-interface	threat list, report	xls (threat list)	NIST CFS ID.RA 3 -> NIST SP 800-53 Rev. 4 RA-3 (ISA 62443-2-1:2009)	11 February 2020	The Threat Modeling Tool enables any developer or software architect to: Communicate about the security design of their systems. Analyse those designs for potential security issues using a proven methodology. Suggest and manage mitigations for security issues	Commercial	Microsoft	1 2
OVVL	DFD, threats knowledge	threat analysis, vulnerability analysis, suggested mitigations; threat definitions are the same as in Microsoft's TMT and are based on a modified STRIDE methodology; Angular as frontend framework, in the backend Spring Boot, data stored in MongoDB (easy handling of big datasets, such as the CVE and CPE data); differs from existing open-source tools by distinguishing between design threats and technical vulnerabilities. The features of the product include threat and vulnerability detection, risk mitigation, project integration, secure storage, analysis & automation, etc.	report: information about the identified threats (at design level) and known vulnerabilities (at operational level) can then be pushed to available tooling in the software development lifecycle	n.a.	n.a.	2019/2020	OVVL is a new open-source framework and tool called "OVVL -Open Weakness and Vulnerability Modeler" to facilitate the integration of threat modeling into the development lifecycle for software teams of any size. Its core functionality is derived from the analysis of the current state and existing solutions	Commercial	developed at University of Applied Sciences Offenburg and part of the BMBF KMU-Innovation Project "CloudProtect" (Förderkennzeichen 16KIS0850)	1 2 3
Threat Dragon	DFD, STRIDE methodology, threats, threats prioritization	web application/desktop application, demo model, support for LINDDUN and CIA as well as STRIDE (Threat Dragon provides STRIDE per Element rules to generate the suggested threats for an element on the diagram) and desktop command line interface	report	pdf	n.a.	Version 1.3 (3 Sep 2020)	a tool used to create threat model diagrams and to record possible threats and decide on their mitigations. TD is both an online threat modelling web application and a desktop application. It includes system diagramming as well as a	Open	OWASP® Foundation	1

							rule engine to auto-generate threats and their mitigations. The focus of TD is on great UX, a powerful rule engine and alignment with other development lifecycle tools			
--	--	--	--	--	--	--	---	--	--	--

Table 29 The Tool based on VAST methodology

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
ThreatModeler	DFD	template, identify, classify and prioritize threats, combines threat modeling with abuse case, intuitive user-interface, suggested countermeasures	threat list, view of security requirements, report	all diagrams can be exported as a JSON, PDF or .png file	NIST CFS ID.RA 3, CIS AWSCIS GCPCIS AzureOWASP NIST 800-53AWS Security Epics; MITRE, CAPEC, OWASP (Mobile, IoT, AppSec), WASC, CSA, NVD	n.a.	ThreatModeler™ is an innovative enterprise threat modeling platform that helps organizations fully integrate security into their SDLC and realize sustainable ROI on their security resources. The centralized threat framework automatically and seamlessly integrates security within existing agile and DevOps workflows	Commercial	ThreatModeler Software, Inc.	1 2

Table 30 The Framework which includes an implementation of ADVISE

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
Mobius Framework	threats	attack execution graph, adversary profile, simulation of construction of composed models (e.g., with SANs)	attack execution graph, simulation results (e.g., defining a performance variable and creating a set/range study)	html, csv, txt	none	up to date (2020)	ADVISE, ADversary View Security Evaluation, is an atomic formalism available in Mobius tool and can be used in conjunction with other formalisms	Commercial	Mobius Illinois	1

Table 31 The tool based on Trike methodology

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
Trike	requirement model, DFD, requires a person to hold a view of the entire system to conduct an attack surface analysis	generate threat (semi-)automatically (no brainstorming), security-inexperienced developers reliably find issues, it is clear what to analyse (and what doesn't need to be analysed), attack chaining (no attack trees), immediate feedback (design), start earlier (requirements, not architecture), include intended system behaviour		xls (non-standalone version)		2012 no longer maintained	Trike threat modelling is a unique, open-source threat modelling process focused on satisfying the security auditing process from a cyber risk management perspective. It provides a risk-based approach with unique implementation, and risk modelling process. The foundation of the Trike threat modelling methodology is a "requirements model." The requirements model ensures the assigned level of risk for each asset is "acceptable" to the various stakeholders	Open	n.a.	1 2 3

Table 32 The CORAS Tool

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
------	--------------------	---------------	--------	---------------	----------	----------------	-------------	----------	-------	------

CORAS	risk management	CORAS method has eight steps: preparation for the analysis, customer presentation of the target, refining target description (using asset diagrams), approval of target description, risk identification using threat diagram, risk estimation using threat diagrams, risk evaluation using risk diagrams, risk treatment using treatment diagrams. CORAS language: graphical modelling language for communication, documentation and analysis of security threat and risk scenarios in security risk analyses. CORAS tool is a diagram editor	"Security risk analysis"	n.a.	ISMS-CORAS: A Structured Method for Establishing an ISO 27001 Compliant Information Security Management System. CORAS method is based on the ISO 31000 risk management standard, which is also the basis for the information security risk management process of ISO 27005	CORAS news --> 2014-12-19: A new version (v1.4) of the Eclipse-based CORAS tool is now released. It is available for both the 32-bit and the 64-bit versions of Windows and Java, as well as for other major platforms such as Mac and Linux	it provides a customised language for threat and risk modelling, and comes with detailed guidelines explaining how the language should be used to capture and model relevant information during the various stages of the security analysis. In this respect CORAS is model-based. The Unified Modelling Language (UML) is typically used to model the target of the analysis. For documenting intermediate results, and for presenting the overall conclusions we use special CORAS diagrams which are inspired by UML. The CORAS method provides a computerised tool designed to support documenting, maintaining and reporting analysis results through risk modelling. The CORAS language is a graphical modelling language for communication, documentation and analysis of security threat and risk scenarios in security risk analyses. The language is an integral part of the CORAS method, which is based on the use of structured brainstorming	Open	SINTEF ICT	1 2 3 4
-------	-----------------	--	--------------------------	------	--	--	--	------	------------	------------------

Table 33 Other Tools

Name	Required Knowledge	Main Features	Output	Export format	Standard	Latest updates	Description	Openness	Owner	URLs
IriusRisk	n.a.	real-time risk management, browser-based user-interface, threat linked with weakness and recommended countermeasures	report	doc, xlsx, pdf, xls	NIST SP 800-53, ISO/IEC 27002, PCI-DSS, OWASP ASVS, OWASP MASVS	5 August 2020 (IriusRisk 3.2)	IriusRisk is primarily a risk management tool that helps you identify, mitigate and track security risks during the software development process. It includes templating and risk pattern-based functionality that allows you to quickly create a threat model based on the answers of a series of questionnaires	Commercial	IriusRisk SL	1 2

securiCAD	DFD	threat modelling, virtual attack simulation, automated model generation, risk assessments and suggested mitigations, integration with external tools (report), non-disruptive (virtual attack), find the structural weaknesses in the architecture; you can import external data	report of attack simulation	n.a.	n.a.	Version 1.6.2 (08/05/2020)	securiCAD is a unique tool for decision-making and risk management in IT security. securiCAD conducts automated attack simulations to models of current and future IT architectures, identifies and quantifies risks holistically including structural vulnerabilities, and provides decision support based on the findings	Commercial	foreseeti	<u>1</u>
PyTM	Python, command line, DFD, Sequence Diagram	generate a Data Flow Diagram (DFD), a Sequence Diagram and threats to your system	DFD, Sequence Diagram, html report	png, html	n.a.	Version 1.1.2 - 24 Sep 2020	Define your system in Python using the elements and properties described in the pytm framework. Based on your definition, pytm can generate, a Data Flow Diagram (DFD), a Sequence Diagram and threats to your system	Open	n.a.	<u>1</u>
SD Elements	n.a.	web-based, threat modelling, risk assessment, identify risk and potential weakness, classify risks, track status of security activities, can instantly generate report (risk -> mitigation); SD Elements focuses on vulnerability prevention instead of detection	report	n.a.	OWASP, WASC threat classification, ISA 62443-4-2 (version 4.21 - Improved content for industrial control systems based on ISA 62443-4-2), NIST 800-171 compliance regulation report (version 4.20), ANSI/ISA-62443, Part 4-2 (version 4.20),	Version 4.23 (Latest Release): May – June 2019	secure application development management tool, called SD Elements, which provides a set of core values to application developers, system analysts, and quality assurance teams. SD Elements is a web-based knowledge repository of security guidelines, empowered by a retrieval tool. SD Elements surveys users to learn about the nature of the project, platform, language, and technologies, and then it tailors security knowledge. — (Notes: Provides requirements, implementation, and testing guidelines, in situations that compliance with PCI DSS and HIPAA (HIPAA) is needed; regulatory compliance including PCI DSS, HIPPAHITECH, GLBA, NERC CIP, and international privacy laws) —	Commercial	Security Compass	<u>1</u> <u>2</u> <u>3</u>

Appendix B – Risk Assessment Steps Descriptions and Reference Standards

Step No.	Topic	Step Name in the Standard	Description	Reference Standard
0	Risk Assessment	Preparation	purpose, scope, assumptions - constraints, information sources, risk model identification.	NIST 800-30 NIST 800-82 NIST 800-53 SAE J3061 EN 50129:2018
0	Risk Management Process	Establishing the Context	Understanding the regulatory environment, identification of requirements and process.	ISO 31000 ETSI EG 203 251
0	Security Risk Assessment (Security for IACS)	Document cyber security requirements, assumptions and constraints	cyber security requirements specification, SUC description, zone and conduit drawings, zone and conduit characteristics, operating environment assumptions, threat environment, organizational security policies, tolerable risk, regulatory requirements	IEC 62443-3-2:2020 NIST.IR 7628
0-1	Hazard/Risk Assessment	Safety goal formulation	a safety goal is to be determined for each hazardous event evaluated in the hazard analysis	ISO 26262 NIST.IR 7628 SAE J3061 EN 50129:2018 EN 50159:2010
1	Risk Assessment	Information assets	define a list of information assets	ISO 27001 NIST.IR 7628 SAE J3061
1	Risk Assessment	Asset identification	identification of assets and potential damage resulting from a breach of security features.	ISO 21434
1	Security Risk Assessment (Security for IACS)	SUC (System Under Consideration) identification	defining a system under consideration (SUC) for an IACS	IEC 62443-3-2:2020
1	Security Risk Assessment (Security for IACS)	Partition the SUC into zones and conduits	establish zones and conduits, separate business and IACS assets, separate safety related assets, separate temporarily connected devices, separate wireless devices, separate devices connected via external networks	IEC 62443-3-2:2020 ISO/IEC 15408-1:2009 EN 50129:2018
2	Hazard/Risk Assessment	Situation analysis and hazard identification	operational situations and operating modes in which a vehicle may malfunction are to be considered as the malfunctioning behaviour may trigger potential hazards. These situations and the corresponding potential hazards are to be described and evaluated. Then the potential hazards are determined	ISO 26262 NIST.IR 7628 SAE J3061 EN 50129:2018 EN 50159:2010

Step No.	Topic	Step Name in the Standard	Description	Reference Standard
2	Risk Assessment	Analyse risk -> Threats and Vulnerabilities identification	identify the threats and vulnerabilities that apply to each asset	ISO 27001 NIST.IR 7628 SAE J3061 EN 50159:2010
2	Risk Assessment	Threat Scenario identification	identification and analysis of possible threats, attacks and vulnerabilities	ISO 21434 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008 NIST.IR 7628 SAE J3061
3	Risk Assessment	Attack Path analysis	identification and linking of potential attack paths to one or more threat scenarios	ISO 21434 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008
1-4	Security Risk Assessment (test-based)	Risk Identification	Determining areas of impact (such as assets), the source of risk (e.g., threats, vulnerabilities, attack surfaces), events, causes and potential consequences. Can involve historical data and security testing.	ETSI EG 203 251
3-4	Hazard/Risk Assessment	Hazard classification	identified potential hazards are to be classified based on the estimation of three factors: severity, probability of exposure, and controllability	ISO 26262 SAE J3061 EN 50129:2018 EN 50159:2010
3-4	Risk Assessment	Impact rating	determination of risk levels based on damage scenarios and the probability of successful attacks	ISO 21434
4	Risk Assessment	Attack Feasibility rating	the rating of the feasibility of attack paths based on the ease of exploitation	ISO 21434
1-5	Risk Assessment	Conduction	produce a list of information security risks that can be prioritized by risk level and used to inform risk response decisions. Organizations analyse threats and vulnerabilities, impacts and likelihood, and the uncertainty associated with the risk assessment process	NIST 800-30 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008 NIST.IR 7628 SAE J3061
1-5	Security Risk Assessment (Security for IACS)	Initial Cyber Security Risk assessment	Perform initial cyber security risk assessment	IEC 62443-3-2:2020 NIST.IR 7628
5	Hazard/Risk Assessment	ASIL determination	SIL is to be determined for each hazardous event using the estimation parameters severity, probability of exposure and controllability	ISO 26262 SAE J3061 EN 50129:2018
5	Risk Assessment	Evaluation / Prioritization	use a risk assessment matrix to identify which risks are worth treating and prioritise them	ISO 27001 ISO/IEC 15408-1:2009

Step No.	Topic	Step Name in the Standard	Description	Reference Standard
				ISO/IEC 18045:2008 SAE J3061 EN 50129:2018
5	Risk Assessment	Risk determination	determination of the risk value of a threat scenario.	ISO 21434 NIST.IR 7628
5	Security Risk Assessment (test-based)	Risk Estimation	Understanding the value of risk, its source and consequences, also involving security testing	ETSI EG 203 251
6	Security Control	Selection	it provides an initial set of controls for the system and tailors the controls as needed to reduce risk to an acceptable level based on an assessment of risk	NIST 800-37 NIST 800-53
6	Risk Assessment	Risk Treatment (decision)	addressing identified risks by selecting a suitable risk treatment option	ISO 21434 ISO 31000 ETSI EG 203 251
7	Security Control	Implementation	it helps describing how the controls are employed within the system and its environment of operation	NIST 800-37 NIST 800-82 NIST 800-53 ISO/IEC 15408-1:2009 SAE J3061
7	Security Control	Assessment of effectiveness	it helps assessing whether the controls are implemented correctly, operating as intended, and producing the desired outcomes with respect to satisfying the security and privacy requirements	NIST 800-37 NIST 800-53 NIST.IR 7628 SAE J3061
7	Risk Assessment	Countermeasures	take countermeasures until the remaining risk is acceptable	ISO 21434 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008 NIST.IR 7628 EN 50129:2018 EN 50159:2010
1-8	Security Risk Assessment (Security for IACS)	Perform a detailed Cyber Security Risk Assessment	identify threats, identify vulnerabilities, determine consequence and impact, determine unmitigated likelihood, determine unmitigated cyber security risk, determine SL-T (Target Security Level), compare unmitigated risk with tolerable risk, identify and evaluate existing countermeasures, reevaluate likelihood and impact, determine residual risk, compare residual risk with tolerable risk, identify additional cyber security countermeasures, document and communicate results	IEC 62443-3-2:2020 NIST 800-82 NIST.IR 7628 SAE J3061

Step No.	Topic	Step Name in the Standard	Description	Reference Standard
6-8	Risk Assessment	Risk treatment selection	eliminate entirely, apply security control, retain risk, share with third party	ISO 27001
8	Risk Assessment	Communication of Results	organizations can communicate risk assessment results in a variety of ways (e.g., executive briefings, risk assessment reports, dashboards). Such risk communications can be formal or informal with the content and format determined by organizations initiating and conducting the assessments share risk-related information produced during the risk assessment with appropriate organizational personnel	NIST 800-30 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008
8	Risk Assessment	Maintenance	keep current, the specific knowledge of the risk organizations incurs. The results of risk assessments inform risk management decisions and guide risk responses. To support the ongoing review of risk management decisions, organizations maintain risk assessments to incorporate any changes detected through risk monitoring	NIST 800-30 SAE J3061 EN 50129:2018 EN 50159:2010
8	Security Control	Monitoring	it helps documenting changes and reporting the security and privacy posture of the system	NIST 800-37 NIST 800-53 SAE J3061
8	Risk Assessment	Result of risk assessment	such as asset lists, damage scenarios, attack reports or risk reports	ISO 21434 ISO/IEC 15408-1:2009 ISO/IEC 18045:2008 SAE J3061
8	Security Risk Assessment (Security for IACS)	Risk comparison	compare initial risk to tolerable risk	IEC 62443-3-2:2020
8	Security Risk Assessment (test-based)	Communicate & Consult, Monitoring & Review	supporting activities meant to provide context and management-related information. Common denominator of the security risk assessment workflow and test-based risk assessment workflow	ISO 31000 ETSI EG 203 251