

Joint safety and security modeling for risk assessment in cyber physical systems

Siwar Kriaa

▶ **To cite this version:**

Siwar Kriaa. Joint safety and security modeling for risk assessment in cyber physical systems. Other. Université Paris Saclay (COmUE), 2016. English. NNT : 2016SACLC014 . tel-01318118

HAL Id: tel-01318118
https://theses.hal.science/tel-01318118

Submitted on 19 May 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : 2016SACLC014

THESE DE DOCTORAT
DE
L'UNIVERSITE PARIS-SACLAY
PREPAREE A
CENTRALESUPELEC

ECOLE DOCTORALE N° 573
Approches Interdisciplinaires : Fondements, Applications et Innovations

Spécialité de doctorat : Sciences et technologies industrielles

Par

Mme Siwar KRIAA

Modélisation conjointe de la sûreté et de la sécurité pour l'évaluation des risques dans les systèmes cyber-physiques

Joint Safety and Security Modeling for Risk Assessment in Cyber Physical Systems

Thèse présentée et soutenue à Châtenay-Malabry, le 11 Mars 2016 :

Composition du Jury :

Mr, Totel, Eric	Professeur, CentraleSupélec	Président
Mr, Dacier, Marc	Directeur de Recherche, Qatar Computing Research Institute	Rapporteur
Mr, Kâaniche, Mohamed	Directeur de Recherche, LAAS - CNRS	Rapporteur
Mr, Blanquart, Jean-Paul	Ingénieur de Recherche, Airbus Defense and Space	Examinateur
Mr, Poisson, Pascal	Ingénieur de Recherche, Alstom Transport	Examinateur
Mr, Bouissou, Marc	Professeur, CentraleSupélec & Electrité De France	Directeur de thèse
Mr, Laarouchi, Youssef	Ingénieur de Recherche, Electrité De France	Co-directeur de thèse

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Modélisation conjointe de la sûreté et de la sécurité pour l'évaluation des risques dans les systèmes cyber-physiques

Mots clés : Systèmes cyber physiques, sûreté, sécurité, modélisation, évaluation des risques

Résumé: Les Systèmes Cyber Physiques (CPS) intègrent des composants programmables afin de contrôler un processus physique. Ils sont désormais largement répandus dans différentes industries comme l'énergie, l'aéronautique, l'automobile ou l'industrie chimique. Parmi les différents CPS existants, les systèmes SCADA (Supervisory Control And Data Acquisition) permettent le contrôle et la supervision des installations industrielles critiques. Leur dysfonctionnement peut engendrer des impacts néfastes sur l'installation et son environnement. Les systèmes SCADA ont d'abord été isolés et basés sur des composants standards et propriétaires. Afin de faciliter la supervision du processus industriel et réduire les coûts, ils intègrent de plus en plus des technologies de l'information et de communication (TIC). Ceci les rend plus complexes et les expose à des cyber-attaques qui exploitent les vulnérabilités existantes des TIC. Ces attaques peuvent modifier le fonctionnement du système et nuire à sa sûreté.

Nous associons dans la suite la sûreté aux risques de nature accidentelle provenant du système, et la sécurité aux risques d'origine malveillante et en particulier les cyber-attaques. Dans ce contexte où les infrastructures industrielles sont contrôlées par les nouveaux systèmes SCADA, les risques et les exigences liés à la sûreté et à la sécurité convergent et peuvent avoir des interactions. Une analyse de risque qui couvre à la fois la sûreté et la sécurité est indispensable pour l'identification de ces interactions ce qui conditionne l'optimalité de la gestion de risque.

Dans cette thèse, nous classifions d'abord les approches de état de l'art qui traitent la sûreté et la sécurité des systèmes industriels et nous soulignons leurs carences par rapport aux quatre critères suivants que nous jugeons nécessaires pour une bonne approche basée sur les modèles : formelle, automatique, qualitative et quantitative, et robuste (i.e., intègre facilement dans le modèle des variations d'hypothèses sur le système).

Nous proposons ensuite une nouvelle approche orientée modèles d'analyse conjointe de la sûreté et de la sécurité : S-cube (SCADA Safety and Security modeling), qui satisfait les critères ci-dessus. Elle permet une modélisation formelle des CPS et génère l'analyse de risque qualitative et quantitative associée. Grâce à une modélisation graphique de l'architecture du système, S-cube permet de prendre en compte différentes hypothèses et de générer automatiquement les scenarios de risque liés à la sûreté et à la sécurité qui amènent à un évènement indésirable donné, avec une estimation de leurs probabilités.

L'approche S-cube est basée sur une base de connaissance (BDC) qui décrit les composants typiques des architectures industrielles incluant les systèmes d'information, le contrôle et la supervision, et l'instrumentation. Cette BDC a été conçue sur la base d'une taxonomie d'attaques et modes de défaillances et un mécanisme de raisonnement hiérarchique. Elle a été mise en œuvre à l'aide du langage de modélisation Figaro et ses outils associés. Afin de construire le modèle du système, l'utilisateur saisit graphiquement l'architecture physique et fonctionnelle (logiciels et flux de données) du système. L'association entre la BDC et ce modèle produit un modèle d'états dynamiques : une chaîne de Markov à temps continu. Pour limiter l'explosion combinatoire, cette chaîne n'est pas construite mais peut être explorée de deux façons : recherche de séquences amenant à un évènement indésirable ou simulation de Monte Carlo, ce qui génère des résultats qualitatifs et quantitatifs.

Nous illustrons enfin l'approche S-cube sur un cas d'étude réaliste : un système de stockage d'énergie par pompage, et nous montrons sa capacité à générer une analyse holistique couvrant les risques liés à la sûreté et à la sécurité. Les résultats sont ensuite analysés afin d'identifier les interactions potentielles entre sûreté et sécurité et de donner des recommandations.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Title : Joint safety and security modeling for risk assessment in Cyber-Physical Systems

Keywords : Cyber Physical Systems, safety, security, modeling, risk assessment

Abstract: Cyber physical systems (CPS) embed programmable components in order to control a physical process or infrastructure. CPS are henceforth widely used in different industries like energy, aeronautics, automotive, medical or chemical industry. Among the variety of existing CPS stand SCADA (Supervisory Control And Data Acquisition) systems that offer the necessary means to control and supervise critical infrastructures. Their failure or malfunction can engender adverse consequences on the system and its environment.

SCADA systems used to be isolated and based on simple components and proprietary standards. They are nowadays increasingly integrating information and communication technologies (ICT) in order to facilitate supervision and control of the industrial process and to reduce costs. This trend induces more complexity in SCADA systems and exposes them to cyber-attacks that exploit vulnerabilities already existent in the ICT components. Such attacks can reach some critical components within the system and alter its functioning causing safety harms.

We associate throughout this dissertation safety with accidental risks originating from the system and security with malicious risks with a focus on cyber-attacks. In this context of industrial systems supervised by new SCADA systems, safety and security requirements and risks converge and can have interactions. A joint risk analysis covering both safety and security aspects would be necessary to identify these interactions and optimize the risk management.

In this thesis, we give first a comprehensive survey of existing approaches considering both safety and security issues for industrial systems, and highlight their shortcomings according to the four following criteria that we believe essential for a good model-based approach: formal, automatic, qualitative and quantitative and robust (i.e. easily integrates changes on system into the model).

Next, we propose a new model-based approach for a safety and security joint risk analysis: S-cube (SCADA Safety and Security modeling), that satisfies all the above criteria. The S-cube approach enables to formally model CPS and yields the associated qualitative and quantitative risk analysis. Thanks to graphical modeling, S-cube enables to input the system architecture and to easily consider different hypothesis about it. It enables next to automatically generate safety and security risk scenarios likely to happen on this architecture and that lead to a given undesirable event, with an estimation of their probabilities.

The S-cube approach is based on a knowledge base that describes the typical components of industrial architectures encompassing information, process control and instrumentation levels. This knowledge base has been built upon a taxonomy of attacks and failure modes and a hierarchical top-down reasoning mechanism. It has been implemented using the Figaro modeling language and the associated tools. In order to build the model of a system, the user only has to describe graphically the physical and functional (in terms of software and data flows) architectures of the system. The association of the knowledge base and the system architecture produces a dynamic state based model: a Continuous Time Markov Chain. Because of the combinatorial explosion of the states, this CTMC cannot be exhaustively built, but it can be explored in two ways: by a search of sequences leading to an undesirable event, or by Monte Carlo simulation. This yields both qualitative and quantitative results.

We finally illustrate the S-cube approach on a realistic case study: a pumped storage hydroelectric plant, in order to show its ability to yield a holistic analysis encompassing safety and security risks on such a system. We investigate the results obtained in order to identify potential safety and security interactions and give recommendations.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

To my mother…

Acknowledgments/ Remerciements

I would like to express my sincere appreciation to the jury members: Marc Dacier and Mohamed Kâaniche for accepting to review this thesis and for their precious feedback on my work. I also warmly thank the examiners: Eric Totel, Jean-Paul Blanquart and Pascal Poisson, for their interest in my work and their insightful comments. I am pleased and honored to have presented my work in front of such a highly qualified jury.

****

Cette thèse a été effectuée dans le cadre d'un contrat CIFRE (Conventions Industrielles de Formation par la REcherche) entre EDF R&D et CentraleSupélec (anciennement, Ecole Centrale Paris).

Je souhaite, tout d'abord, exprimer ma profonde reconnaissance à mon directeur de thèse : Marc Bouissou, et le remercier de m'avoir accompagné pendant cette thèse et avant, lors de mon stage de fin d'études. Grâce à ses compétences industrielles et académiques, Marc m'a apporté à la fois son expertise en tant qu'ingénieur senior en sûreté de fonctionnement à EDF R&D; mais aussi les notions théoriques et formelles associées, en tant que professeur à l'Ecole Centrale Paris. J'ai beaucoup appris de Marc qui était toujours disponible pour me conseiller, mais aussi pour m'encourager et me conforter pendant les moments les plus difficiles.

Je tiens ensuite à remercier les trois encadrants industriels coté EDF, que j'ai côtoyés pendant cette aventure : mes chaleureux remerciements s'adressent en premier à Youssef qui s'est chargé de l'encadrement de ma thèse à partir de la fin de la deuxième année. Son aide était précieuse et efficace, car il a pu encadrer mes travaux et m'apporter son recul dans le domaine de la recherche. Je remercie ensuite Yoran qui m'a accompagné en début de la thèse, pour son expertise en cyber-sécurité, ses conseils et son soutien. J'adresse enfin mes remerciements à Ludovic, qui a cru en moi dès le début car c'est grâce à lui que cette thèse a été lancée. Ludovic m'a apporté une aide inestimable avec ses conseils, tout au long de la thèse, et son recul sur le sujet. Youssef, Yoran et Ludovic, chacun de son coté, m'ont apporté trois visions des choses aussi différentes qu'enrichissantes.

Les travaux de cette thèse ont été effectués au sein du groupe Infrastructures de Calcul, Communication et Sécurité du département SINETICS à EDF Clamart. Je commence ainsi par remercier le chef du département Stéphane Bugat de m'avoir accordé sa confiance pour mener à bien mes travaux de recherche. Je tiens ensuite à remercier tous les membres du groupe I2D, y compris les anciens: d'abord les deux chefs de groupe Stéphane Ploix et avant, David Bateman, pour leur confiance, leurs conseils et leur soutien ; et ensuite tous les autres collègues pour leur accueil et la bonne ambiance de travail. A plusieurs moments, les échanges que j'ai eus avec eux (Fredéric, Pascal, Christophe, Alia…) étaient toujours agréables, enrichissants et constructifs. Les pauses café étaient toujours un moment agréable (surtout quand c'est accompagné de croissants chauds) pour parler de tout, raconter des blagues et rire ensemble.

J'aimerais également remercier tous les membres du laboratoire LGI: Bill, Bernard, Vincent, Wassila, Delphine, Sylvie, Corinne, Carole, … et tous les doctorants : Oumaima, Manel, Imen, Elisa, Hakim … qui ont rendu mes passages au labo, pourtant peu fréquents, tellement agréables et conviviaux.

Je tiens à exprimer toute ma gratitude à mes parents qui m'ont toujours soutenu, avec leurs encouragements et leur amour, pour aller jusqu'au bout de mes rêves. Je leur dédie cette thèse, qui est aussi la leur, car c'est grâce à eux que j'en suis là aujourd'hui.

J'adresse mes profonds remerciements à mon cher mari Wassim qui m'a encouragé à faire cette thèse et qui a participé à son aboutissement. Merci d'être toujours présent à mes côtés pour me soutenir, m'encourager, me conseiller, et m'aimer…

Je remercie également mes beaux-parents pour leur soutien, ainsi que toute ma famille : mes grands-parents, mes tantes et mes oncles, ma sœur et mon frère, et mes amis Emna et Mohamed … de m'avoir toujours encouragé et donné de l'énergie positive.

Je dédie enfin cette thèse à mon adorable fils Yassine, mon trésor, qui est venu au monde pendant la thèse pour l'agrémenter de bonheur et de défis.

Contents

Abbreviations and Acronyms 5

Introduction 9

Chapter 1 13

Safety and Security in ICS: issues & challenges 13
1.1 Safety and Security definitions and interdependencies 13
1.1.1 Merging safety and security: a common challenge for numerous industries 13
1.1.2 Terminology 15
1.1.3 Similarities and differences between safety and security 16
1.1.4 Interdependencies between safety and security 17
1.2 Industrial Control Systems: specificities and requirements 19
1.2.1 ICS: definitions and Enterprise architecture topology 19
1.2.2 ICS specificities and security challenges 20
1.3 Safety and security standards for Industrial (Control) Systems 21
1.3.1 Safety standards 21
1.3.2 Security standards 22
1.3.3 Safety and Security standards initiatives 24
1.4 Conclusion 25

Chapter 2 27

Design and operational approaches integrating safety and security: state of the art, classification and critical analysis 27
2.1 Classification criteria 27
2.1.1 Unification vs. Integration approaches 28
2.1.2 Design vs. Operational approaches 28
2.1.3 Qualitative vs. quantitative approaches 28
2.2 Process-oriented approaches 28
2.2.1 Unification approaches 29
2.2.2 Integration approaches 30
2.3 Model based approaches 36
2.3.1 Graphical modeling approaches 36
2.3.2 Non-graphical modeling approaches 44
2.4 Critical Analysis 46
2.4.1 A canonical life-cycle integrating safety and security 46
2.4.2 Analysis of the different approaches identified 47
2.4.3 Discussion 50

Chapter 3 53

Modeling safety and security with Boolean logic Driven Markov Processes 53
3.1 Previous work 53
3.1.1 Modeling safety with BDMP 53
3.1.2 Modeling security with BDMP 55
3.1.3 Modeling safety and security with BDMP 56
3.1.4 The KB3 workbench 56
3.2 Modeling real attacks and complex systems 58
3.2.1 Modeling the Stuxnet attack with BDMP 58

3.2.2 Modeling safety and security interdependencies 65
3.3 Comparison of BDMP with the CHASSIS method Modeling 72
3.3.1 Preparing the comparison 72
3.3.2 Comparing the model elements 73
3.3.3 Qualitative comparison of the sequences 73
3.3.4 The BDMP model 73
3.3.5 The CHASSIS model 74
3.3.6 Results of the comparison 77
3.3.7 Experiences with applying the two approaches 80
3.3.8 Conclusion 81
3.4 Discussion on BDMP 81

Chapter 4 83

The S-cube approach: a model-based approach for SCADA Safety and Security joint modeling 83
4.1 Existing safety and security domain specific languages 84
4.1.1 Security domain DSLs 84
4.1.2 Safety domain DSLs 85
4.2 The S-cube approach: principle and stakeholders 86
4.2.1 Principle of the S-cube approach 86
4.2.2 Stakeholders 87
4.3 The S-cube Knowledge Base 90
4.3.1 Rationale 90
4.3.2 Metamodel 93
4.3.3 Taxonomy of attacks 95
4.4 Qualitative aspects in the S-cube KB 99
4.4.1 Failure modes and repair 99
4.4.2 Attack steps 99
4.4.3 Attack and failure scenarios generation 101
4.5 Quantitative aspects in the S-cube KB 103
4.5.1 Safety metrics 103
4.5.2 Security metrics 105
4.5.3 Discussion 111
4.6 Implementation & Tool chain 111
4.6.1 The Figaro language 111
4.6.2 Tool chain 114
4.7 Conclusions 116

Chapter 5 117

S-cube application on case studies 117
5.1 Modeling corporate networks 117
5.1.1 Description of the case study 117
5.1.2 Qualitative and quantitative risk analysis 120
5.1.3 Conclusions and enhancement 122
5.2 Modeling a hydroelectric ICS: variant 1 122
5.2.1 Overview on the Taum Sauk upper reservoir failure 123
5.2.2 Description of the case study architecture 124
5.2.3 The graphical model 125
5.2.4 Pure safety risk analysis 127
5.2.5 Safety and security joint risk analysis 129

5.2.6 Conclusions on case study variant 1 133
5.3 Modeling a hydroelectric ICS: variant 2 133
5.3.1 The graphical model 134
5.3.2 Pure safety risk analysis 135
5.3.3 Joint safety and security risk analysis 135
5.3.4 Conclusions on variant 2 138
5.4 Comparison between the two variants 138
5.4.1 Pure safety analysis 139
5.4.2 Joint safety and security analysis 139
5.4.3 Safety and security interdependencies 139
5.5 Conclusion 140
Chapter 6 141
Conclusions & Perspectives 141
6.1 Conclusions 141
6.2 Perspectives 143
Annex 1: BDMP models of the Stuxnet attack 145
Annex 2: Individual description of classes 146
Annex 3: Assumptions on the quantitative metrics for the use case: pumped storage plant 151
References 153

Abbreviations and Acronyms

BBN	Bayesian Belief Networks
BDMP	Boolean logic Driven Markov Processes
BPM	Basic Parametric Model
DBPM	Dynamic Basic Parametric Model
CCF	Common Cause Failure
CHASSIS	Combined Harm Assessment of Safety and Security for Information Systems
COTS	Commercials Off The Shelf
CVSS	Common Vulnerability Scoring System
D-MUC	Diagrammatical MisUse Case
DoS	Denial of Service
DSL	Domain Specific Language
D-UC	Diagrammatical Use Case
EBIOS	Expression des Besoins et Identification des Objectifs de Sécurité
FMEA	Failure Mode and Effects Analysis
FMVEA	Failure Mode, Vulnerabilities and Effects Analysis
FSD	Failure Sequence Diagram
HAZOP	HAZard and OPerability
ICS	Industrial Control Systems
ICT	Information and Communication Technologies
IEC	International Electrotechnical Committee
ISMS	Information Security Management Systems
ISO	International Organization for Standardization
IT	Information Technologies
KB	Knowledge Base
MTTF	Mean Time To Failure
MTTS	Mean Time To Success
MUC	MisUse Case
MUSD	MisUse Sequence Diagram
NIST	National Institute of Standards and Technology
OS	Operating System
PERA	Perdue Enterprise Reference Architecture
PDF	Probability Distribution Function
PRM	Probabilistic Relational Models
SCADA	Supervisory Control And Data Acquisition
S-cube	SCADA Safety and Security joint modeling
SD	Sequence Diagram
SIL	Safety Integrity Level
SIS	Safety Instrumented System
SPN	Stochastic Petri Nets
SSI	Safety Security Interdependencies
T-MUC	Textual MisUse Case
TTC	Time To Compromise
TTF	Time To Failure
TTFC	Time To First Compromise
TTS	Time To Success
T-UC	Textual Use Case
UML	Unified Modeling Language

"*All models are wrong, but some are useful*"
George E. P. Box (1919-2013)

"*Tous les modèles sont faux, certains sont utiles*"
George E. P. Box (1919-2013)

Introduction

Context

Industrial systems like power plants, industrial factories, airplanes or cars address the daily and vital needs of society. Their safety has been for a long time given a careful consideration as their failure or malfunction can engender adverse consequences on humans and the environment. Industrial Control Systems (ICS) offer the necessary means to control and supervise these critical systems and infrastructures.

Traditional industrial control systems (ICS) were based solely on mechanical and electrotechnical devices and proprietary standards which were well mastered. These systems have however become expensive to deploy, maintain and operate, and it is becoming difficult to follow the innovation trend in the industrial context. To address these challenges, new information and communication technologies are being increasingly integrated into modern control systems: radio-based services from commercial providers, commercial off-the-shelf (COTS) products (e.g., Windows operation systems), TCP/IP based communications, etc. This migration towards standardized communication technologies and open protocols has facilitated the deployment of highly connected systems and enabled remote control and supervision of infrastructures. For instance, Supervisory Control and Data Acquisition (SCADA) systems are largely deployed in various industries. Although this has increased efficiency and reduced costs for industrial operators, the overall infrastructures have become vulnerable to external malevolence. Indeed, with their increasing complexity and interconnection, modern industrial control systems are exposed to new security-related threats like cyber-attacks.

We adopt in this dissertation the following definitions for safety and security (cf. section 1.1.2 for terminology). We associate safety with accidental risks originating from the system that could result in unacceptable consequences on the system's environment. Security is related to malicious risks and we are mainly interested in cyber-security.

As a matter of fact, since the Stuxnet attack [1] has targeted the Iranian nuclear enrichment facilities and gained a lot of attention from media and press, the number of cyber-attacks targeting industrial facilities has considerably increased. According to [2], attacks doubled on SCADA systems in 2014 compared with 2013. Software vulnerabilities and external networks can be back doors for attacker to access industrial control systems, reach some critical components within the system and alter its functioning causing potential safety-related harms.

Problem

For a long time, much attention was focused on safety concerns related to highly critical systems with large impacts on their environments. However, only accidental components failures or software errors were traditionally addressed in safety analyses.

Today, in this context characterized by the migration of industrial infrastructures towards digital control systems, system safety can also be compromised by security breaches and electronic attacks. It is consequently no longer sufficient to address accidental threats of such systems, threats of intentional origin need to be covered as well.

As Information and Communication Technologies (ICT) have recently been integrated into Industrial Control Systems, security communities working on the security of industrial information systems are also relatively recent.

Although both safety and security communities deal with risks and share the same goal of protecting industrial infrastructures, they are still working separately. Yet, for industrial infrastructures having safety issues and being supervised and controlled by modern ICS such as SCADA, safety and security requirements and risks converge and can have mutual interactions [3]. Indeed, security related requirements and risks can influence the system safety and inversely safety related requirements and risks can influence the system security.

To address these challenges, a joint risk analysis framework considering both safety and security aspects has become crucial. It enables to cover exhaustively risks related to safety and security and identify their potential interdependencies. It consequently conditions a thorough and optimal risk management as well as cost and resource optimization.

Contributions

In this thesis, we propose a new model based approach that we called S-cube, for safety and security joint modeling for Industrial Control Systems in general and SCADA based ICS in particular. The S-cube (for SCADA Safety and Security modeling) approach enables to model an industrial system architecture with the associated safety and security aspects and to automatically generate the possible risk scenarios that lead to an undesirable event with safety issues. It can be applied either in the design phase of new safe and secure systems or in the operational phase of existing systems to optimize and master their safety and security.

In addition to this main contribution, this thesis incorporates other contributions which we summarize as follows:

- The classification of existing approaches combining safety and security for industrial control systems, with a critical and comparative analysis. This classification led us to investigate first the BDMP (Boolean logic Markov Process) approach for safety and security risk modeling;
- The application of the BDMP approach on realistic use cases and its comparison with the CHASSIS (Combined Harm Assessment of Safety and Security for Information Systems) method. The limits of the BDMP and CHASSIS approaches have been identified which led to the development of the S-cube approach;
- The implementation of the S-cube approach;
- The application of the S-cube approach on realistic use cases;

This thesis will be structured as follows: the first chapter will address safety and security issues and challenges in new industrial control systems. The second chapter gives a survey of approaches

integrating safety and security and highlights their main limitations. Chapter 3 investigates the Boolean logic Driven Markov Processes formalism, its application on real case studies and compares it to another complementary approach (CHASSIS). The S-cube approach is depicted in Chapter 4 where its principle and implementation are described. Chapter 5 illustrates S-cube on different case studies and underlines its different uses. Chapter 6 concludes finally this dissertation and gives perspectives.

Chapter 1

Safety and Security in ICS: issues & challenges

This first chapter addresses first the new security risks related to modern Control Systems in different industries and the safety challenges they result into. It also clarifies the definitions of safety and security, their similarities and differences and their possible interdependencies. Secondly, the Industrial Control Systems specificities and requirements are outlined. Thirdly, we present some emergent standardization initiatives that consider safety and security coordination for ICS.

1.1 Safety and Security definitions and interdependencies

In the recent years, there has been an increase in both the frequency and seriousness of cyber-attacks targeting critical infrastructures (e.g. Stuxnet in 2010 and Flame in 2012). Moreover security experts regularly demonstrate control systems vulnerabilities at conferences such as Black Hat or DefCon. For instance, the presentations entitled "Compromising Industrial Environments from 40 Miles Away" and "Out of Control: Demonstrating SCADA Exploitation" given during the BlackHat 2013 conference [4] are two recent and representative examples, among numerous others. The result is increased awareness of new security-related risks that threaten industrial infrastructures and of the growing importance of considering safety and security jointly.

1.1.1 Merging safety and security: a common challenge for numerous industries

Numerous industrials have been affected by the modernization and digitalization of their control systems. These industries include manufacturers that produce critical systems that directly impact their users such as cars, airplanes, trains, medical implants, etc. We find also industrials that hold large infrastructures such as energy power plants, oil refineries, sewerage, etc.; that have safety issues. Although different, all these industries share today the same challenge of protecting their critical systems from security risks that can engender safety impacts. The following paragraphs provide some domain-specific considerations or citations.

The **aerospace industry** is the pioneer domain in which safety and security risks have been addressed in an integrated manner [5]. A wide range of software and hardware contribute to high levels of equipment interconnectivity in new aircraft platforms, associated with both aircraft and the operating environment; these include highly critical items associated with controlling aircraft, non-critical items that inform and entertain the passengers, and others that help airlines operate and maintain their fleets [6]. Moreover, the deployment of Ethernet-based networks and COTS into aerospace systems has increased risks of intentional misuse of aircraft information systems and made them targets of security breaches that could have an impact on aircraft safety and lead to human losses.

In [7], Kosher *et al.* outlined recent dramatic changes in the **automotive industry**: "Modern automobiles are no longer mere mechanical devices; they are pervasively monitored and controlled by dozens of digital computers coordinated via internal vehicular networks. While this transformation has driven major advancements in efficiency and safety, it has also introduced a range of new potential risks." A thorough review of the associated attack potential, including safety related vulnerabilities, is provided in [8][9].

Smith *et al.* [10] described major evolutions in the **railway industry**, especially the migration to remotely controlled trains and the trend towards radio transmissions, or communications based train control (CBTC), which have created new safety and security challenges.

Novak *et al.* described new trends in the **construction industry** related to building automation and control systems (BACS), such as remote access via the internet and shared use of networks for advanced control operations such as safety, security, HVAC (heating, ventilation and air conditioning) and other functions. "These trends break up the isolated structure of existing networks and therefore introduce new risks and threats to working systems" [11].

In the **oil and gas industry**, authors of the SINTEF report [12] emphasized external vendors' growing need for remote access to safety instrumented systems (SISs) in order to ensure the safety of offshore installations. However, remote access requires the use of different networks, including the Internet, which decreases security and threatens SISs. Likewise, Johnsen [13] highlighted the need to improve safety and security in distributed process control systems used in the oil and gas industry. This need emerges from the increasing connection of SCADA systems to networks and their migration towards standardized information and communication technologies which increases their exposure to security threats.

In the **electricity industry**, complex systems control the generation of nuclear, thermal, and renewable (wind, hydraulic) power. In the different electricity domains: production, transport network and distribution, information and communication technologies (ICT) are being deployed into the control systems that use open protocols and COTS.
For electricity production, new nuclear reactors like European Pressurized Reactor (EPR) are modernizing their control systems and hydroelectric plants can be remotely supervised or even controlled.
The evolution of electricity distribution networks towards more energy efficient and environmentally-friendly solutions also has radically changed infrastructures and their associated control systems. Smart grids have evolved in response to the development of decentralized wind and photovoltaic production, the emergence of new uses of electricity (e.g., to power vehicles), the installation of smart meters to ensure data collection and communication with substations, the deployment of sensors and communication devices for better network management, etc. While smart grids have facilitated network management and energy optimization, there has also been an alarming convergence between safety and security stakes due to digitalization and the growing interconnections within these new infrastructures [14]. New situations in which cyber security requirements and safety issues would affect the same systems must be considered when deploying these systems.

To meet this increasing need for securing critical infrastructures against cyber-threats, some European and international collaborations have been formed among industrial and academic partners to develop new safety and security solutions. Security Safety Modeling (SESAMO), for instance, is a European project completed in 2015 that involved 20 academic and industrial partners from the aerospace, automotive, energy, rail transportation and medical domains. The SESAMO partners have addressed safety and security convergence in embedded systems by investigating different methodologies for mastering interdependencies between safety and security and ensuring compatibility. The ultimate goal of this project was to develop an overall methodology and toolkit for integrating safety and security and to apply them to cases that are representative of industrial infrastructures. Some public deliverables [15][16] are on the project website[1].

The meanings of safety and security may vary according to the context and the technical communities. We clarify in the following section the terminology of these two terms and the definitions adopted in the remainder of this dissertation in order to avoid ambiguities.

1.1.2 Terminology

Surprisingly, the definitions of the terms *safety* and *security* [17] vary widely in different contexts and technical communities. In some languages, such as German and Spanish, the same word is used for both terms: "Sicherheit" resp. "Seguridad". However, even in languages that offer two distinct words, the meaning of each term varies considerably from one context to another [18]; for example, security and safety mean different things to electrical engineers than to members of the nuclear community. As a consequence, in order to better define the scope, motivations and objective of this paper, it is of paramount importance to clearly define exactly what is meant by safety and security in this review.

Both safety and security deal with risk. According to ISO 31004 [19] risk is defined as "the effect of uncertainty on objectives, regardless of the domain or circumstances, therefore an event or a hazard (or any other risk source) should not be described as a risk. Risk should be described as the combination of the likelihood of an event (or hazard or source of risk) and its consequence." According to this standard, two important parameters are considered for the risk definition: the source of risk and its consequence. The source of risk can be either accidental or deliberate, and the consequences can be of different types: financial, environmental, human, etc.

Beyond their primary objective of controlling and supervising the industrial process, Industrial Control Systems aim also at protecting the industrial infrastructure, its users and the environment. They are consequently required to be highly dependable. **Dependability** is a more general concept associated to critical systems. Avizienis *et al.* [20] defined system dependability as its ability to deliver a service that can justifiably be trusted. The dependability attributes traditionally addressed are called the RAIMS: Reliability, Availability, Integrity, Maintainability, and Safety.

- Reliability: continuity of the service;
- Availability: readiness to deliver the service;
- Integrity: absence of improper system alteration;
- Maintainability: ability to undergo modifications and repairs;
- Safety: absence of catastrophic consequences on the system users and the environment.

Avizienis *et al.* [20] present in Figure 1 dependability as an integrated concept that encompasses most security attributes. Like dependability, security includes integrity and availability attributes, but requires additionally confidentiality which means the absence of unauthorized disclosure of information.

[1] http://sesamo-project.eu/documents

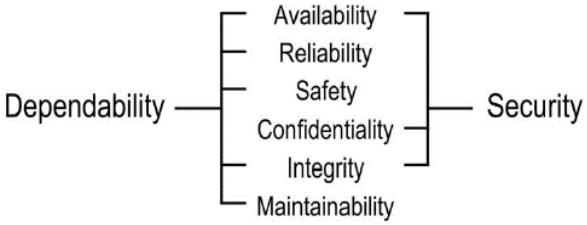


Figure 1: Dependability and security attributes

Availability and integrity as addressed in traditional dependability analyses have been dealing with "accidental" alteration or absence of data that are associated to errors and failures of systems components. Considering in addition security into dependability analyses, data alteration or absence is associated to "unauthorized" actions related to malevolence.

We will be studying in the remainder of this thesis risks that lead to safety issues, i.e. which could result in unacceptable consequences on the system's environment such as human losses, heavy material loss, and nature damage (also called undesirable events). These risks can be either of accidental or malicious origin. In our industrial context, we associate **Safety** with risks having accidental origins. **Security** is associated to risks originating from malicious attacks[2]. These attacks can be accomplished physically (through local access to the system) or by cyber means (through local or remote access to the system). We exclude physical attacks from the scope of this survey (unless specifically mentioned); therefore, security in this paper refers to cyber security and protection from cyber-attacks.

We explain in the following sections the main similarities and differences between safety and security and the rationale for studying interdependencies between them.

1.1.3 Similarities and differences between safety and security

Safety and security, although distinct, share many commonalities. Eames *et al.* [21] argued that both safety and security deal with risks, result in constraints, involve protective measures, and create requirements. These similarities indicate that some of the techniques applicable to one field could also be applicable to the other. "Either accidents or attacks may eventually cause harm to the system assets (in terms of people, property, environments or services)"[22].

Similarities between safety and security have also been recognized by many authors [23][24][25][26][27]. Thus, some authors have explored the proposition that safety and security are a duality, and that much could be gained by one adopting the knowledge, understanding, tools and techniques of the other, and vice versa [25][26]. To highlight these similarities, Eames and Moffett [21] give the general steps of risk analysis common for safety and security : starting by the investigation of the system, then the qualitative and quantitative analysis and finally the definition of the appropriate countermeasures.

Although safety and security share many similarities, it is important to emphasize that each discipline has its own particularities. The main difference between safety and security (as we define them) is the *origin* of risk: safety considers hazards (i.e., how the system may harm the environment due to system failure or some combination of accidental conditions), while security considers threats and focuses on how potential attacks may impact the system's assets and its operation due to vulnerability [22].

The differences between safety and security likewise are reflected by differences in the tools, standards and the way in which risk management is conducted in the two domains, especially during the risk assessment phase. Assessing a security threat is radically different from assessing a safety hazard. In the

[2] We do not intend to provide absolute definitions of the terms safety and security; our aim is simply to have a single word for each central manipulated concept in the context of this report.

first case, the sources of the threats to be assessed are usually not well-known by the analyst, and cover an extremely broad range of possible scenarios. In the second case, the characteristics of the hazards are more accessible, and the number of scenarios to be taken into consideration may also be reduced to a set that is restricted, but sufficient to be considered significant. This is immediately noticeable in quantitative approaches when trying to quantify safety and security metrics using probabilities: safety hazards are relatively stable over time and feedback from analysts on previous accidents is therefore reliable, while security attributes are less predictable [28][29] and depend on many factors (e.g., attacker profile, skills, motivation, etc.), which makes it more difficult for a security analyst to assess and quantify possible scenarios. The use of probabilities in quantitative approaches is widely adopted in the safety field, whereas such approaches are not always accepted in the security field [29].

A more extensive discussion on the differences and similarities between safety and security is provided in [26]. We provide in Section 1.3 an overview of standards specific to each domain in order to show how methods of risk analysis and treatment differ between safety and security communities.

We highlight the fact that in the context of industrial control systems, neither a pure security approach nor a pure safety approach can mitigate risks to the physical infrastructure of the system. Security functions[3] are not meant to cope with physical hazards and failures; likewise, safety functions[4] might not detect and respond to attacks that target the digital components of the system. We infer that safety and security are complementary and should be treated jointly to improve risk management.

1.1.4 Interdependencies between safety and security

Interdependencies between safety and security are studied by examining how requirements and measures from one field may impact the other field.

For a long time, safety and security have been examined independently by separate and distinct communities. However, with the increasing integration of information technologies into industrial control systems, such communities are becoming increasingly aware of the possible dependencies between safety and security issues and the need to treat both disciplines jointly.

Novak *et al.* [30] demonstrated that when safety and security measures are considered jointly, synergy can be identified and leveraged. These authors first analyzed separately Safety goals and Security goals and then compared security goals with the safety procedure. This comparison results into a common procedure between both disciplines that ensures integrity verification, authentication and authorization [11]. Dependencies between them are also outlined, as safety can only be assured by considering the safety-critical system as a whole, meaning that software, which is subject to security threats, must be included in the process.
Likewise, Reichenbach *et al.* [31] stated that concepts in the analysis and development phase are similar and can be reused, which would eliminate overlaps and redundant work; thus, a process including both safety and security in a harmonized toolkit could ease development. Similarly, in several papers [3][26][32], Pietre-Cambacedes *et al*. illustrated the advantages of using models encompassing safety and security when identifying and characterizing risk scenarios.

Novak *et al.* [33] noted drawbacks of separating safety and security when related features are developed and integrated without regard for their dependencies. This separation may increase costs, implementation time and complexity, and dramatically reduce performance. The authors asserted that safety and security issues ought to be considered not only during requirement specification, but also during design, operation, maintenance, and decommissioning (i.e., over the complete lifecycle).

[3] In our context, security functions are system-level processes that protect information and resources from attacks.

[4] In our context, safety functions are processes implemented and delivered by safety devices and measures put in place in order to protect people and the system environment from physical impacts of accidental events (e.g., fire alarms)

In order to better understand the benefits of simultaneously considering safety and security throughout a system's lifecycle, it is important to closely examine the various kinds of interdependencies that can exist between safety and security. Extending previous works [21][34][30], Pietre-Cambacedes summarized [3] safety-security interactions into four kinds:

- Conditional dependency: Fulfillment of safety requirements conditions security or vice-versa.
- Mutual reinforcement: Fulfillment of safety requirements or safety measures contributes to security, or vice-versa, thereby enabling resource optimization and cost reduction.
- Antagonism: When considered jointly, safety and security requirements or measures lead to conflicting situations.
- Independency: No interaction.

These four kinds of interactions cover and extend the different relationships identified in other papers. We discuss and illustrate each kind of interaction with an example:

- Conditional dependency: This interaction is present in the context of many, if not all digital systems that control and monitor safety-critical industrial processes. For example, Smith *et al.* [10] emphasized the security requirements of railway signaling to guarantee train safety. In this context, malicious changes to sensor data or automated device settings may prevent safety systems from protecting an industrial facility in case of an accident. One example of safety conditioning security would be a crisis situation following an accident in which safety functions would prevent efficient intrusion detection because of preemption and priorities on human-machine interfaces and security-related alarms.

- Mutual reinforcement: In some cases, security and safety strengthen each other. Identifying such situations can help optimize resources and even eliminate certain overlaps. Novak [34] identified such synergy in the use of message authentication codes (MACs), a security measure used for authentication purposes, to replace cyclic redundancy checks (CRCs), a safety measure used for transmission error correction. MAC can replace CRC, since the security measure also ensures a safety role by counteracting data integrity errors and uses stronger mechanisms than CRC with respect to malicious modifications.

- Antagonism: Various authors [21][28][35][36] have used the example of an exit door in a limited-access facility and identified a conflict between safety and security designers. The former addresses fire hazards and infer that the door must be kept unlocked in case of fire, whereas the latter addresses potential attacks and infers that the door must be kept locked to prevent unauthorized persons from accessing the facility. A similar and somewhat humorous example was reported by Sun *et al.* [35]: "A European luxury car manufacturer noticed that one of its models was a disproportionately likely target for theft. The mystery was solved when an apprehended thief revealed that this model of car easily opened even when its doors were locked. When jumping on the roof of the car, doors would unlock. The designers of the car included a safety feature whereby the doors of the car would unlock if the car was involved in an accident and rolled over. To check for roll-over situations they verified if enough pressure/weight was being applied on the roof."

- Independence: At first glance, this type of relationship might not seem as important as the other three; in fact, identifying cases of independence between safety and security is valuable for several reasons. First, it can support an objective view of shortcuts that are too often taken, assuming incorrectly that a security risk will "naturally" be detected by a safety risk assessment and resolved as part of the corresponding measures, or vice versa. In some cases, a safety measure will not have any effect on the system's level of security (and vice versa). For example, some approaches to redundancy may have clear benefits for safety, but no effect on security, since vulnerabilities may simply be duplicated.

1.2 Industrial Control Systems: specificities and requirements

We provide in this section an overview on Industrial Control Systems and their safety and security requirements. We particularly underline the specificities related to industrial security compared to security of Information Technology (IT) systems.

1.2.1 ICS: definitions and Enterprise architecture topology

We first define what we call ICS and then give an overview on the Enterprise Architecture topology encompassing these control systems.

The term ICS covers a large variety of control systems among which we find Supervisory Control and Data Acquisition (SCADA) systems used generally for systems with a wide geographical range.

We use in the remainder the term ICS to designate whatever control system used to control an industrial system and the term SCADA particularly for digital systems used to supervise and control industrial infrastructures.

The Perdue Enterprise Reference Architecture (PERA) provides a reference model for Computer Integrated Manufacturing (CIM) [37] that divides the enterprise architecture in different layers based on organizational hierarchy.

Inspired from PERA, IEC 62264-1 [38] provides a model for the enterprise that split the architecture into five key levels[5], based on functional hierarchy:

- Level 0: the physical process;
- Level 1: functions involved in sensing and manipulating the physical process;
- Level 2: functions involved in monitoring and controlling the physical process;
- Level 3: functions involved in managing the work flows to produce the end-products;
- Level 4: functions involved in business-related activities needed to manage the manufacturing organization.

This functional decomposition can be mapped to the real enterprise architecture where components ensuring these functionalities are placed in the corresponding level. Devices that are directly involved in the industrial control process are particularly located at levels 1 and 2. Brun *et al.* [39] depicts in Figure 2 the typical system components of modern control architectures (Level 0 is not presented as it depends on the nature of the industrial system).

Given the evolution of the industrial control infrastructures and the integration of advanced technologies, we can find intelligent sensors that ensure both the functionalities of measurement and process control; and can therefore be considered to belong to levels 1 and 2. Particularly, in smart grids, communicating smart sensors can be used to control the process of electricity distribution. Contrarily to the classical hierarchical architecture, we are talking about "flat" architecture where control is distributed between the different components.

We give in the following sections an overview on ICS specificities and underline the differences between securing traditional IT systems and securing ICS.

[5] This standardized decomposition can, however, differ between different industries and communities.

1.2.2 ICS specificities and security challenges

As Industrial Control Systems integrate new Information and Communication Technologies (ICT) traditionally used in Management Information Systems, they consequently inherit of their vulnerabilities. However ICS have their own characteristics that render their security different from securing traditional IT systems.

We highlight in the following ICS specificities and show the security challenges derived [40], [41]:

- Time-criticality: control systems monitor generally real-time processes. This requires data freshness (e.g., sensor measures are only valid in their designated time) and system responsiveness (e.g., an actuator should execute in real-time the controller orders). Replay-attacks targeting industrial processes enable typically to violate this property. This requirement is not necessary in traditional IT systems were orders can wait for a certain time before being executed.

- Availability: control systems require high availability as they generally control industrial process that are continuous. Unexpected ICS outages can cause unacceptable losses ranging from economic losses and equipment damage, to human losses.

  Unlike IT systems, software updates are not systematically done for ICS as they require to restart the systems. ICS maintainability actions should be scheduled in advance in order to minimize their impacts on the industrial process continuity.

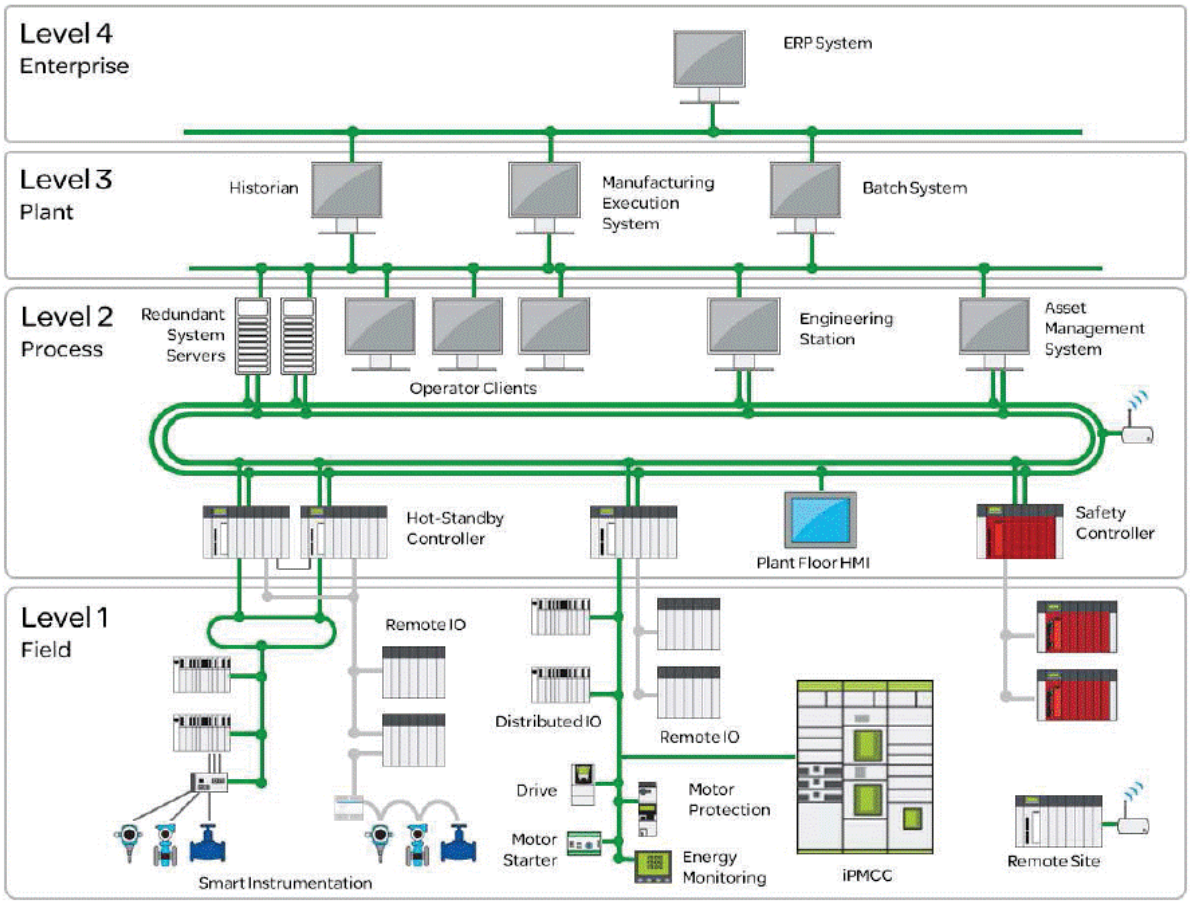


Figure 2: Typical components of modern control architectures [39]

- Fault-tolerance: requires the control systems being capable to continue operating properly in the presence of faults and components failures. The equivalent security property is called intrusion-

tolerance (also called graceful degradation) and requires to keep the attack impact local without escalating into a full system cascading event.

These ICS characteristics that differ from traditional IT systems imply different security risks and priorities. Security requirements for traditional IT systems, ordered with decreasing priority are respectively Confidentiality, Integrity and Availability (CIA). This priority order is inversed when it comes to control systems (AIC):

- Availability: control and process data should always be available to guarantee the good execution of the industrial process;
- Integrity: control and process data (generated, transmitted, displayed and stored) should be genuine and intact;
- Confidentiality: data confidentiality is desirable in the industrial context but not of paramount importance. Unlike availability and integrity problems, data disclosure should not induce safety related issues, but may impact the enterprise image.

Unlike traditional IT systems, ICS are on the first frontier facing human lives and ecological environment. Security properties and requirements applied for IT systems are consequently not completely adapted to control systems and need to be adjusted taking into consideration ICS specificities.

Pietre-Cambacedes *et al.* [42] give a list of myths and realities about ICS. For example, here are four myths:

- Industrial control systems are isolated, not concerned by cyber-attacks and are not understandable by attackers
- Cyber-security incidents do not impact the industrial process
- Classical antiviruses and firewalls are sufficient to protect our systems
- System providers completely master the security of their products

Since safety and security have been for a long time treated separately, the standards related to each discipline have also been distinct. Nevertheless, new emergent standardization initiatives are working on safety and security coordination for control systems of different industries. The next section is dedicated to these standards.

1.3 Safety and security standards for Industrial (Control) Systems

We first give an overview on some safety (only) and security (only) specific standards that are commonly used in the industrial context, and that are the basis for some of the approaches identified in Chapter 2. We next highlight the new standardization works that consider safety and security coordination for industrial control systems.

1.3.1 Safety standards

We recall the main principles of the generic safety standard IEC 61508, from which different industries derived their own standards (e.g., IEC 61513 for the nuclear industry, IEC 61511 for industrial processes, and ISO 26262 for the automotive industry). It covers the complete safety lifecycle including the analysis, realization and operation phases.

IEC 61508, a standard for the functional safety of electrical/electronic/programmable electronic safety-related systems [43], covers important aspects that must be addressed when electrical, electronic, and programmable devices are used in connection with safety functions. The strategy of the standard is to derive safety requirements from a hazard and risk analysis and to design the system to meet those safety requirements, taking all possible causes of failure into account. In essence, all activities related to safety are managed in a planned and methodical way, with each phase having defined inputs and outputs [23]. The standard considers all phases in a safety lifecycle, accompanying the product lifecycle through

initial concept, design, implementation, operation and maintenance, and decommissioning. Figure 3 illustrates the overall safety process associated with this standard.
IEC 61508 specifies a set of methods, measures and procedures that must be respected for claiming a certain safety integrity level (SIL). The standard does not directly address security requirements, but recognizes that if a malevolent or unauthorized action is identified in hazard analysis, a security threat analysis should be carried out. It points towards the IEC 62443 series for dealing with cybersecurity issues (presented in the following paragraph).

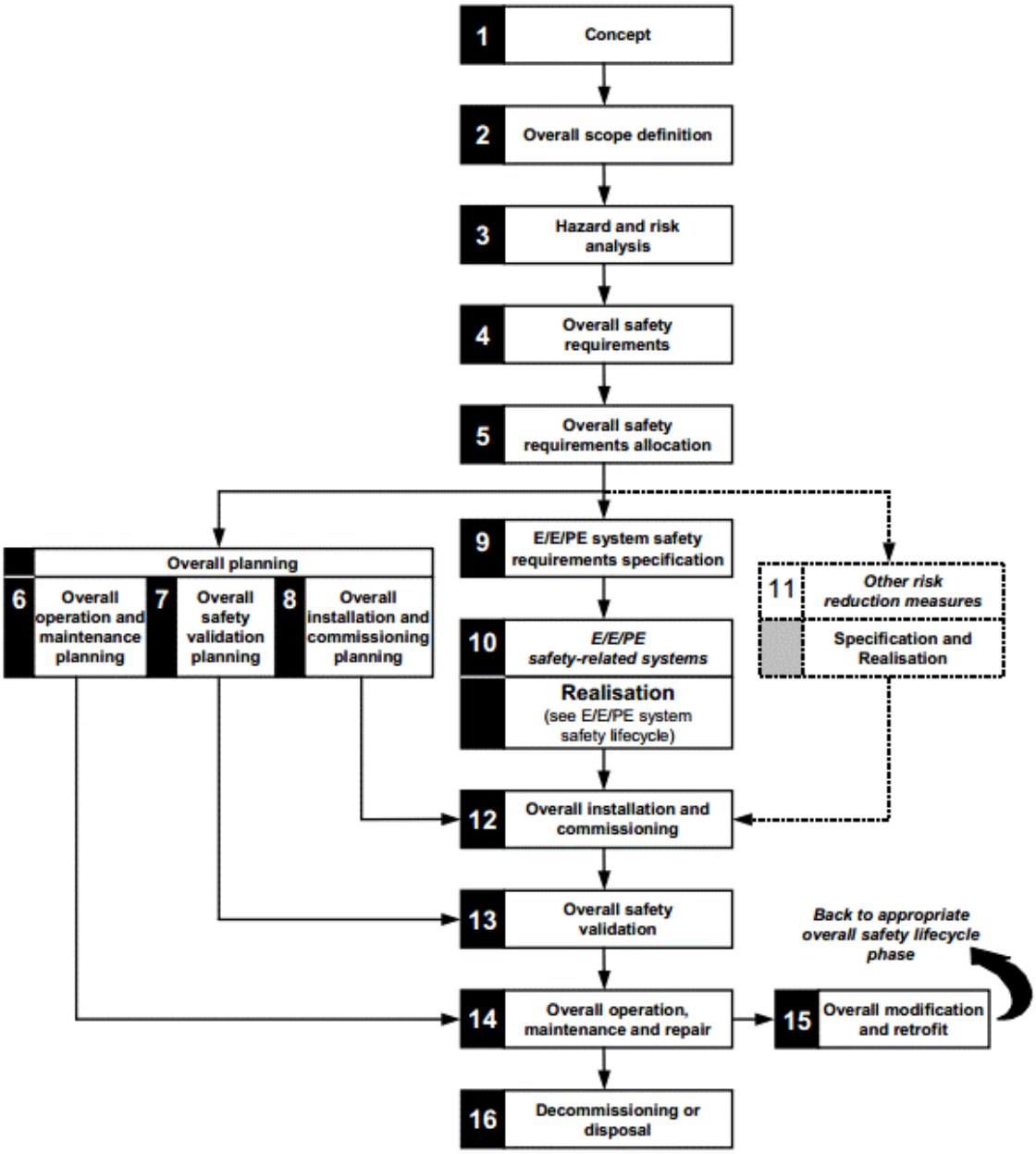


Figure 3: Overall safety lifecycle of IEC 61508 [39]

1.3.2 Security standards

We present in this subsection some security standards that are highly recognized for security design and evaluation.

ISA 99/IEC 62443 is a series of security standards for industrial automation and control systems. In particular, it considers the security of safety systems [31]. For example, to fortify safety instrumented system (SIS) connectivity, Part 2-4 of the standard specifies security measures such as:

- Assuring communications integrity through hard-wiring or logical separation of safety-related communications from the other network control communications;
- Isolating the SIS from layer 3 (network layer) connections;
- Preventing data connection between the distributed control system and the SIS; and
- Restricting allowed functions of the engineering work station (used for maintenance purposes) and its interaction with the SIS.

IEC 62645 is entitled "Nuclear power plants – instrumentation and control systems – requirements for security programs for computer-based systems." It was published in 2014 and provides guidance for the development and management of effective computer security programs at nuclear power plants for instrumentation and control (I&C) systems. The standard defines a graded approach to cyber security by assigning a security level to each I&C system based on the impact of an attack on safety and performance. I&C systems are grouped into security zones [44][45].

The following security standards are related to information systems in general and do not take consequently into account the constraints of industrial control systems. They are however used by security engineers of industrial computing.

Common Criteria/IEC 15408 is an international standard that establishes the general concepts and principles of IT security evaluation and specifies the general evaluation model. It is meant to be used as the basis for evaluating the security properties of IT products. It addresses the assurance levels applied to security management and the related criteria for evaluating and claiming such assurance levels.

ISO/IEC 27001 is an international standard for Information Security Management Systems (ISMS) [46]. It provides a lifecycle model for establishing, implementing, operating, monitoring, reviewing, maintaining and improving an organization's ISMS. This lifecycle model, called "Plan-Do-Check-Act" (PDCA) and shown in Figure 4, illustrates the main actions of the ISMS process that takes information security requirements as inputs and produces information security outcomes that meet these requirements.

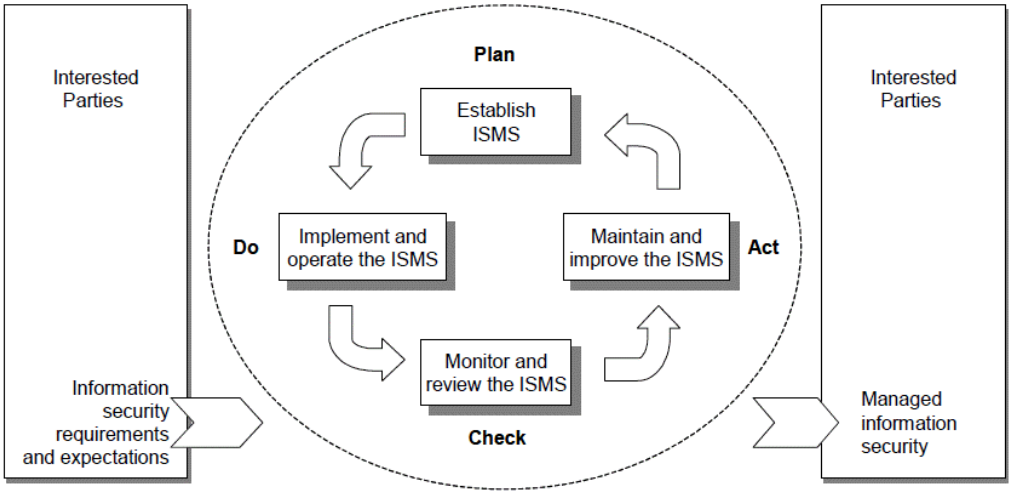


Figure 4: PDCA model applied to ISMS processes [46]

This standard is the top-level document of a series of standards that includes ISO/IEC 27005 [47], which specifically addresses information security risk management. It provides a structured, systematic and

rigorous process for everything from analyzing risks to creating the risk treatment plan. This process is illustrated in Figure 5. Unfortunately, these standards do not mention the safety of the analyzed system.

The **EBIOS** methodology has been formalized and adopted by the French National Agency of Information Systems Security (ANSSI). This methodology has been updated in order to comply with ISO/IEC 27005 and proposes a way to assess and treat risks. It consists of identifying the assets to protect and performing a hazard analysis on each asset by identifying possible attack scenarios, evaluating related risks, and proposing security measures. The methodology is primarily intended to be used to identify security related risks for information systems, but enables the identification of some safety related risks.

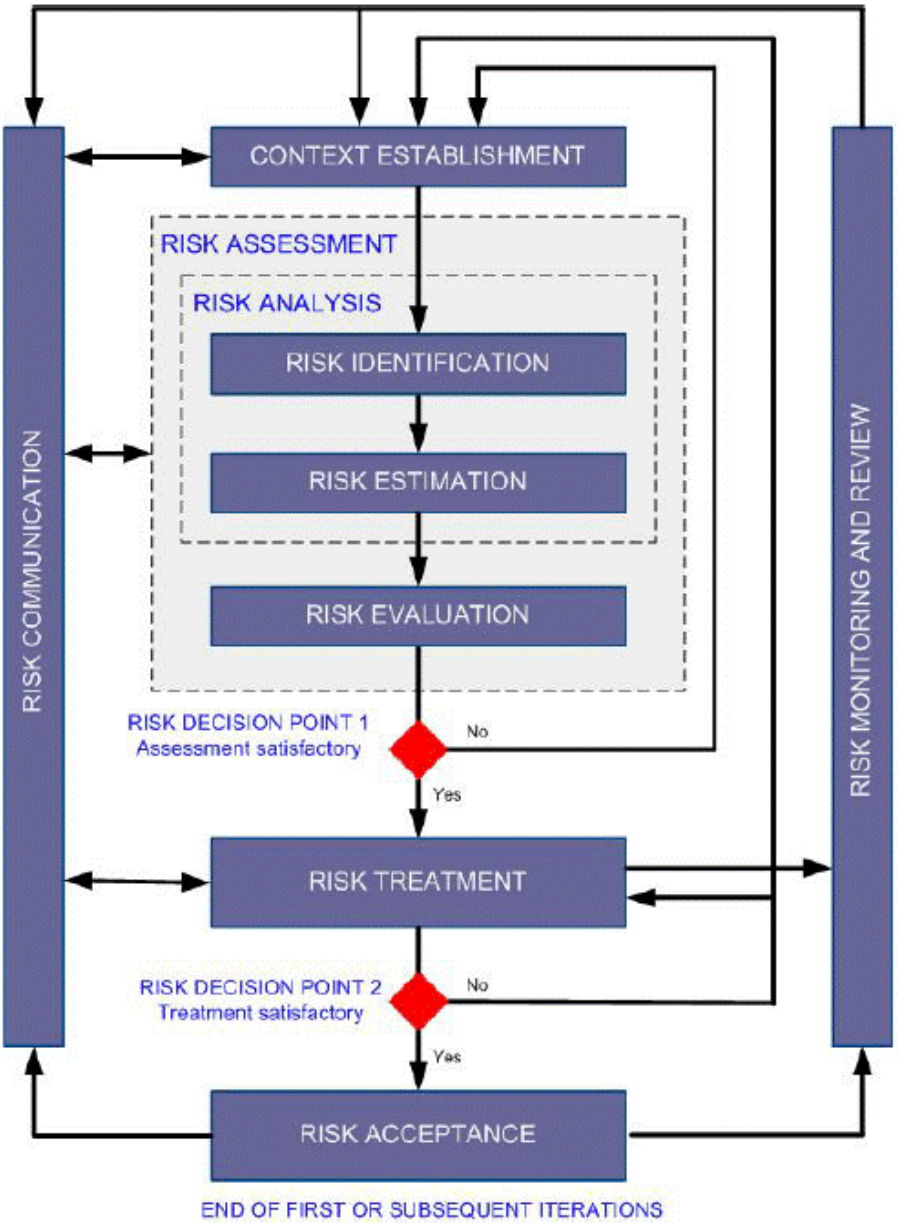


Figure 5: Information security risk management process from ISO/IEC 27005 [47]

1.3.3 Safety and Security standards initiatives

In this section, we describe some emergent standardization initiatives that consider safety and security coordination for industrial control systems.

ISA-99 WG7 is a working group established within the ISA-99 committee to address issues related to the safety and security of industrial automation and control systems. The purpose of this working group is to extend the existing safety lifecycle to consider cyber security aspects at different phases of the industrial process lifecycle (i.e., design, implementation, commissioning, and maintenance) in order to ensure a reliable, efficient, and safe plant.

IEC TC65 AHG1 is a recently formed ad-hoc working group of the IEC, which intends to develop a "Framework Toward Coordinating Safety and Security". It is attached to the same technical committee as the one developing the IEC 61508 and IEC 62443 standards, creating a promising opportunity for the safety and cybersecurity experts involved in their development to meet and actually develop the targeted framework.

IEC 62859, a future standard derived from IEC 62645 (cf. Section 1.3.2), is entitled "Nuclear power plants – instrumentation and control systems – requirements for coordinating safety and cyber security." It aims to establish requirements and recommendations that optimize and coordinate design and operational efforts with respect to safety and cyber security, enhance the identification and resolution of potential conflicts between these aspects throughout the I&C system lifecycle, and aid in the identification and leveraging of potential synergies between safety and security [44]. Although this standard concerns the nuclear domain, many other industrial sectors have shown concrete interest in coordination between safety and cyber security. We mention, for instance, initiatives in the aerospace industry through the SESAR project [6] and in air traffic control management [48].

DO-326/ED-202, first published in 2010 and revised in 2014, is a standard in the avionics domain entitled "Airworthiness Security Process Specification". It is intended to augment current guidelines for aircraft certification to address information security threats to aircraft safety. Successive versions of this document have addressed the positioning of the security assessment process with respect to the system development phase and the safety assessment process in different ways. First, in its first published version (2010), security activities were embedded in safety activities with a bidirectional exchange with the system development phase; in a 2013 draft of its revision (DO-326A draft), security activities were decoupled from safety activities, but with mutual feedback between them; in the revision finally published in 2014 (DO-326A), security activities were further distanced from safety activities with unique feedback from safety to security. The latter does not inform the former directly. The hesitations about DO-326 result from difficulties associated with putting the "ideal" process that was initially devised into practice.

1.4 Conclusion

We clarified first in this chapter the meanings of the terms safety and security as used in the context of this work and underlined their similarities and differences. We also stressed the potential interdependencies between safety and security, in terms of risks and requirements, which can be of a synergetic or conflictual kind. The second part of this chapter addressed these notions in the context of Industrial Control systems (ICS): first by highlighting the specificities and the new challenges of these systems and secondly by identifying the new standardization initiatives that aim at coordinating safety and security issues for critical systems.
We provide in the next chapter a classification of existing approaches that consider safety and security issues for industrial systems. We further give a critical analysis of these approaches and identify their limitations.

Chapter 2

Design and operational approaches integrating safety and security: state of the art, classification and critical analysis

With the growing awareness that safety and security analyses should be coordinated in risk assessment for complex systems, some members of industry and academia have collaborated and initiated work to bridge the gap between safety and security. As illustrated in Section 1.3.3, some industrial sectors particularly exposed to this issue have already started standardization processes; however, there exist numerous other initiatives, either more isolated or more exploratory which deserve interest and should feed in efforts towards a better coordination between safety and security. We describe in this section the different approaches that propose processes or methodologies in which safety and security concerns are jointly considered [49]. These approaches are classified according to the criteria described in Section 2.1; Table 2 summarizes this classification. We provide next in Section 2.4 a critical and comparative analysis of these approaches in which we discuss the advantages and drawbacks of each approach.

2.1 Classification criteria

The different approaches identified in the state of the art have been classified in two main categories: process-oriented approaches and model-based approaches, further refined according to three criteria detailed in paragraphs 2.1.1 to 2.1.3.

Process-oriented approaches: propose new lifecycles and methodological processes that consider safety and security at a very macroscopic level of system design or risk evaluation. They rely on requirements generally specified by safety and/or security standards (cf. Section 1.3) and provide generic descriptions of lifecycles; indicating what kinds of activities should be performed in what sequence.

Model-based approaches: contrarily to process-oriented approaches, model-based approaches rely on a formal or semi-formal representation of the functional/non-functional aspects of system and are generally supported by tools. We further classify the model-based approaches identified according to whether they rely on graphical or non-graphical models.

2.1.1 Unification vs. Integration approaches

We adopt the following distinction made by Eames *et al.* [21] between unification and integration approaches:

- **Unification approaches** are aimed at uniting safety and security techniques into a single methodology. The result of these approaches is a single set of requirements describing the safety and security functions of the system.
- **Integration or harmonizing approaches** are aimed at investigating the similarities and differences of safety and security techniques in order to bring them into alignment. These approaches produce safety and security requirements separately using standard concepts and methodologies, and then show how they interact in order to identify conflicts.

Eames *et al.* [21] emphasized the disadvantages of approaches that attempt to unify safety and security analysis techniques. According to the authors, unification techniques reduce a developer's understanding of the system being analyzed and prevent a thorough analysis of either property, which leads to an incomplete analysis with subsequent safety and security risks going unobserved; global abstraction might obscure requirements conflicts and necessary trade-offs [21]. On the other hand, they outlined the advantages of integrating safety and security by harmonizing techniques from each domain: "[By] applying techniques developed in each domain, conflicts could become more apparent; better understanding of the system and its environment and recognition of risks related to each domain, separation of properties would permit recognition of conflicts and trade-offs and allow judgment-based decisions to be made." The authors inferred that safety; security and their associated risk analysis techniques are closely related and have sufficient similarity to make integration a reasonable and achievable goal.

2.1.2 Design vs. Operational approaches

We classify the approaches identified according to the system lifecycle phase they cover: the design stage, the operational stage, or both stages.

- **Design stage approaches** are aimed at designing new systems in which safety and security constraints/functions are considered jointly;

- **Operational stage approaches** are aimed at studying and evaluating safety and security interactions on existing and operational systems.

2.1.3 Qualitative vs. quantitative approaches

Qualitative approaches address the risks related to a given system and identify their causes and consequences (e.g. via a Failure Mode and Effects Analysis (FMEA)). Quantitative approaches aim at giving a measure of the risk (e.g., score, probability, interval) in order to assess its severity and frequency. For instance, FMEA can be complemented by a score associated to each failure mode by the product of scores supposed to represent its likelihood and its gravity.

2.2 Process-oriented approaches

We classify the process-oriented approaches according to the unification vs. integration distinction provided by Eames *et al.* [21] as described in Section 2.1.1.

2.2.1 Unification approaches

Stoneburner [50] proposed a unified security-safety framework. This framework combines the risk taxonomy for security from the U.S. National Institute of Standards and Technology (NIST) and the risk taxonomy for safety from the Federal Aviation Administration and proposes a common taxonomy by adopting a unified definition for the term "mishap" that includes safety hazards and security threats.

Aven [51] proposed a unified framework for risk and vulnerability analysis covering both safety and security. The risk and vulnerability analysis process is comprised of the eight following steps:

1. Identify the relevant functions to be analyzed.
2. Define the systems to meet these functions.
3. Identify relevant sources of risk (threats, hazards).
4. Perform an uncertainty analysis of these sources.
5. Perform a consequence analysis, addressing uncertainties (using event trees, fault trees).
6. Describe risks and vulnerabilities.
7. Evaluate risks and vulnerabilities.
8. Identify possible measures, and return to 3.

Kornecki *et al.* [22] proposed a combined safety/security engineering process encompassing six subsequent activities, which in turn may require modification of the preceding activity:

1. Identify the assets to protect from harm.
2. Identify harm that can come to the assets.
3. Identify and analyze accidents (safety) and attacks (security) that may cause harm.
4. Identify and analyze hazards (safety) and threats (security).
5. Discover vulnerabilities of the assets.
6. Develop safety/security requirements that ensure the asset(s) can be protected.

Derock *et al.* [36] identified the commonalities between the ISO/IEC 15026 safety standard[6] for system and software integrity levels and the ISO/IEC 27005 security standard; both are "based on a risk management oriented approach." Based on the high level of similarity between the two standards, the authors proposed a generalized process that merges safety and security processes of the two standards. The main phases of this process are:

1. Definition and scheduling of safety activities
2. Preliminary hazard analysis
3. Risk analysis during the specification phase
4. Risk analysis during the design phase
5. Transfer of safety requirements
6. Verification of safety requirements fulfillment on components
7. Verification of the test phases
8. Closure of the safety process

This common analysis process shows the convergence between ISO/IEC 27005 and IEC 15026 and supports the need to merge safety and security analyses for complex systems. The authors pointed out that only one engineer should work on both safety and security, which reduces analysis time and cost and increases efficiency.

Woskowski [52] proposed a risk-based approach to improve the safety and security of critical medical devices by extending the risk management required by IEC14971 beyond device boundaries; the

[6] This standard introduces the concepts of software integrity levels and software integrity requirements (SIL).

objective is to cover interface safety, interface usage and network security aspects, as well as to define appropriate risk mitigation techniques. The adapted IEC 14971-workflow [52] considers both safety and security related hazardous situations in the risk estimation phase.

2.2.2 Integration approaches

Eames and Moffet [21] proposed an integration approach that consists in applying separate risk analysis processes for safety and security for the purpose of determining requirements. Requirements in safety documentation are then cross-referenced to the security analysis and vice versa in order to analyze and identify interactions between safety and security requirements. The resolution of conflicts and inconsistencies enables practitioners to identify changes and measures to be implemented and evaluate their effects.

Johnson [53] proposed a roadmap for "CyberSafety" engineering (which refers to security for safety in our context) aimed at increasing resilience against cyber-attacks on safety-critical systems. The first steps of the roadmap aim at better understanding security threats to safety-critical systems and improving security screening for infrastructure engineers and technical staff across safety-critical industries. The potential threat of insider attacks should be taken into consideration as well as the capabilities of engineering teams to deal with cyber threats (e.g., installing updates, security patches using unverified media). The next steps support multi-party exercises by rehearsing large scale attacks across infrastructures to assess their potential consequences, and sharing lessons learned from previous incidents. This should lead to improving tools, revising security risk assessments in safety-related applications and addressing the consequences of security measures on safety cases (e.g., safety cases can be undermined by the detection of malware or unauthorized access; or it can be difficult to guarantee response times given the potential impact on processor and memory resources). This roadmap is aimed at increasing the resilience of safety-critical infrastructures to cyber-attacks that may induce physical consequences; however, it does not propose a method for capturing safety-security interdependencies.

Kornecki *et al.* [54] proposed a V-shaped development model in which they added security actions to the conventional V-model used in the development of dependable software. The resulting model includes hazard and threat analyses, architectural mitigation, security algorithms, and security and attack tests. The authors stated that "the development of safe and secure systems requires the developers to adhere to a rigorous development process framework that includes analysis of the security threats from the perspective of the industrial control system safety and elaboration of potential countermeasures and mitigation mechanisms."

Novak *et al.* [30] provided a lifecycle model for the pre-design phase of a safe and secure building automation and control system (BACS). The authors reviewed and extended this lifecycle model in further publications [33][34], as illustrated in Figure 6, in order to cover the development and use phases of a system.

This safe and secure lifecycle model uses the safety lifecycle from IEC 61508 (steps 1 and 2) and integrates the approach to deriving security requirements from the Common Criteria security project standardized as IEC 15408 (step 3). The authors claimed that these standards provide good coverage of functional safety and security aspects and created a system classification with four safety integrity levels (SIL) for safety related systems, and seven evaluation assurance levels (EAL) for security related systems. Steps 4 and 5 are additional activities that consider safety and security dependencies resulting from the integration of safety and security.

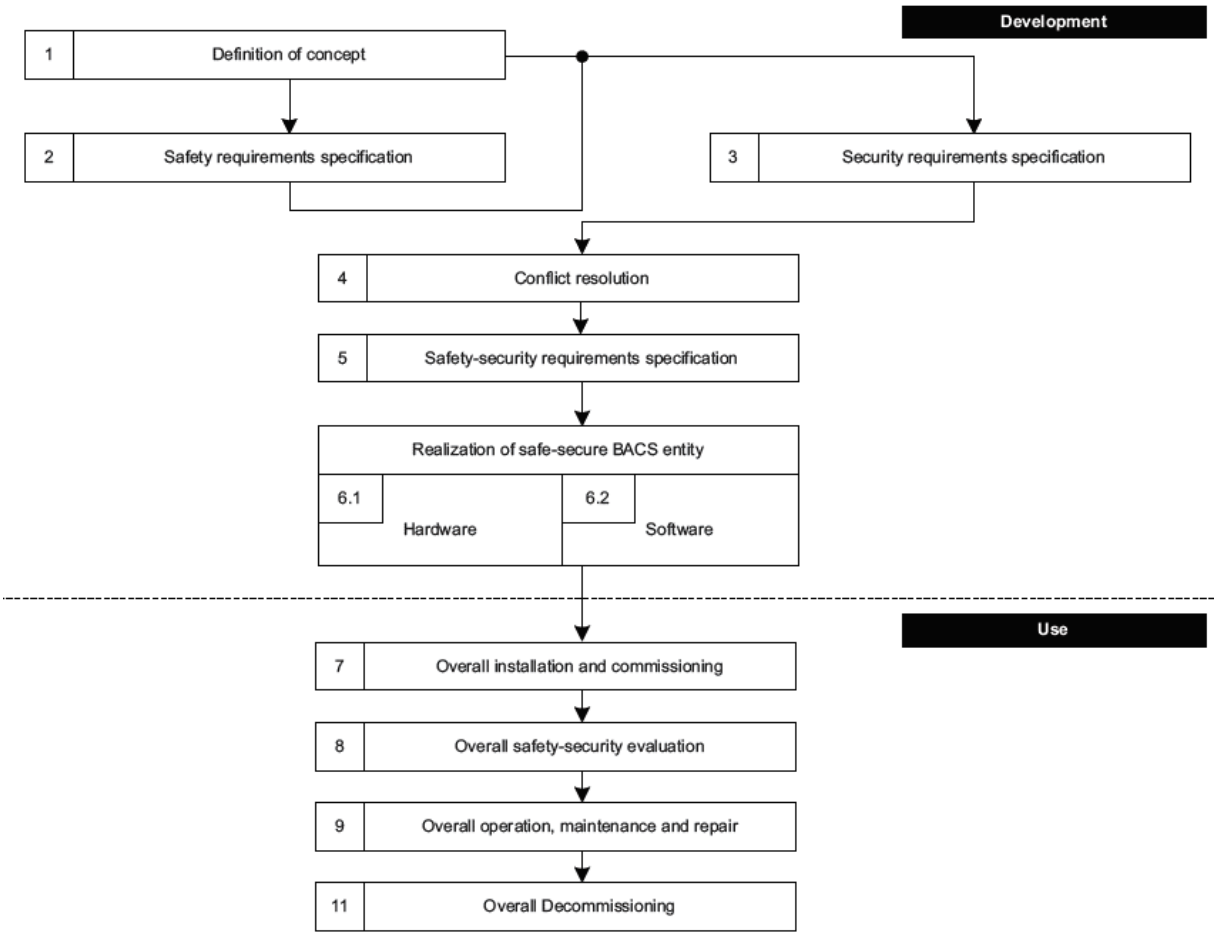


Figure 6: Safety-security lifecycle model

The lifecycle model begins with the definition of the physical environment, boundaries and scope of the system. After typical hazards and associated risks are identified, safety requirements are specified to reduce risks to acceptable levels. After a safety investigation and the identification of assets requiring protection, the list of threats and associated risks and the derived security objectives enable security requirements to be specified. In steps 4 and 5, safety and security requirements are investigated in order to identify commonalities and contradictions: "It is checked whether security requirements lead to new hazards and risks to human health; i.e., are new safety requirements necessary due to security needs and how do they influence security" [30]. At the conflict resolution step, interactions between safety and security requirements are examined by cross-checking the sets of requirements and any conflicts are resolved. The authors introduced the conflict resolution approach at the requirements and functional levels. The conflict resolution approach at the requirements level is shown in Figure 7.

The conflict resolution policy aims at specifying which requirements are preferred in particular situations. It consists of two rules applicable to a subsystem [11]:

1. Prefer safety requirements to security requirements if security has a negative impact on safety;
2. Otherwise, use security requirement.

The resolution of conflicts between safety and security requirements results in a conflict-free set of requirements (step 5).

Conflict resolution at the functional level consists in the assessment of measures that exhibit conflicts (i.e., require different amounts of effort in terms of computational power or consumed memory).

Selection of the adequate measure is influenced by five factors: field of application, safety performance, security performance, hardware environment and software environment.

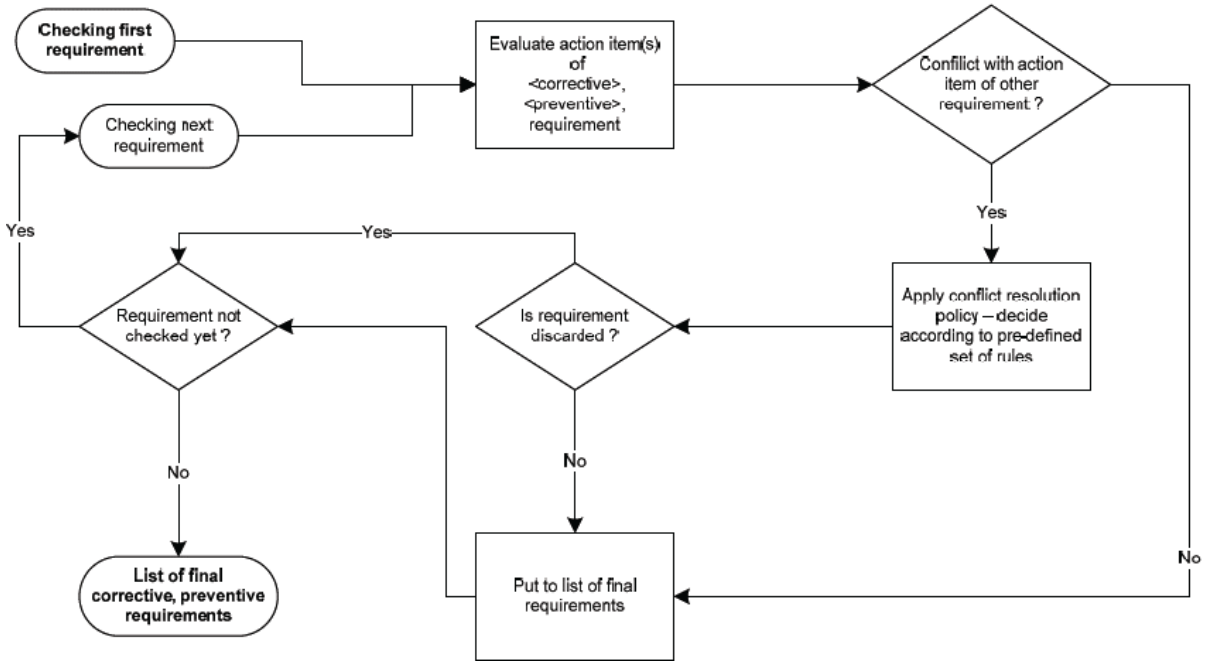


Figure 7: Conflict resolution at the requirements level

The last step of the development phase is constituted by the realization of the safe-secure system, including software and hardware. The use phase is concerned with installation, safety-security validation, operation, maintenance and disposal of the automation system [11]. In this phase, the whole system and the overall interactions among all components must be considered. Safety-security validation is concerned with investigating whether risks have been mitigated properly and whether system safety and security are always provided.

Although the approach and examples (use case of an office building) [33] relate to building automation, the lifecycle model addresses macro level issues and can be applied to other kinds of systems.

Hunter [28], proposed an aligned approach in which safety and security are integrated into different stages of the system's development lifecycle: concept, requirements, qualification and maintenance. This approach, called Lifecycle Attribute Alignment shows lifecycles related to the system's development, system safety management and information security management, and establishes interactions between phases of different lifecycles (see Figure 8). These interactions ensure compatibility between safety and security controls that are established and maintained during the system development; for example, alignment attribute A5 ensures that security updates do not compromise safety.

The control compatibility model proposed in [28] enables to manage any conflicts and incompatibilities between safety and security objectives that may arise. Security objectives are divided into value objectives (i.e., "must," "must not," and "do not care") for each functional aspect. For compatibility to be achieved no "must" control in one aspect may coexist with a "must not" in the other aspect; other matches are compatible.

Sørby [23] provided a development process integrating both safety and security aspects for critical systems inspired by the safety lifecycle from the IEC 61508 standard [43] and the CORAS approach to identifying security risks [55]. This lifecycle, depicted in Figure 9, contrary to the one proposed by Novak in Figure 6, gives more details on how safety and security requirements are integrated in the risk management phase.

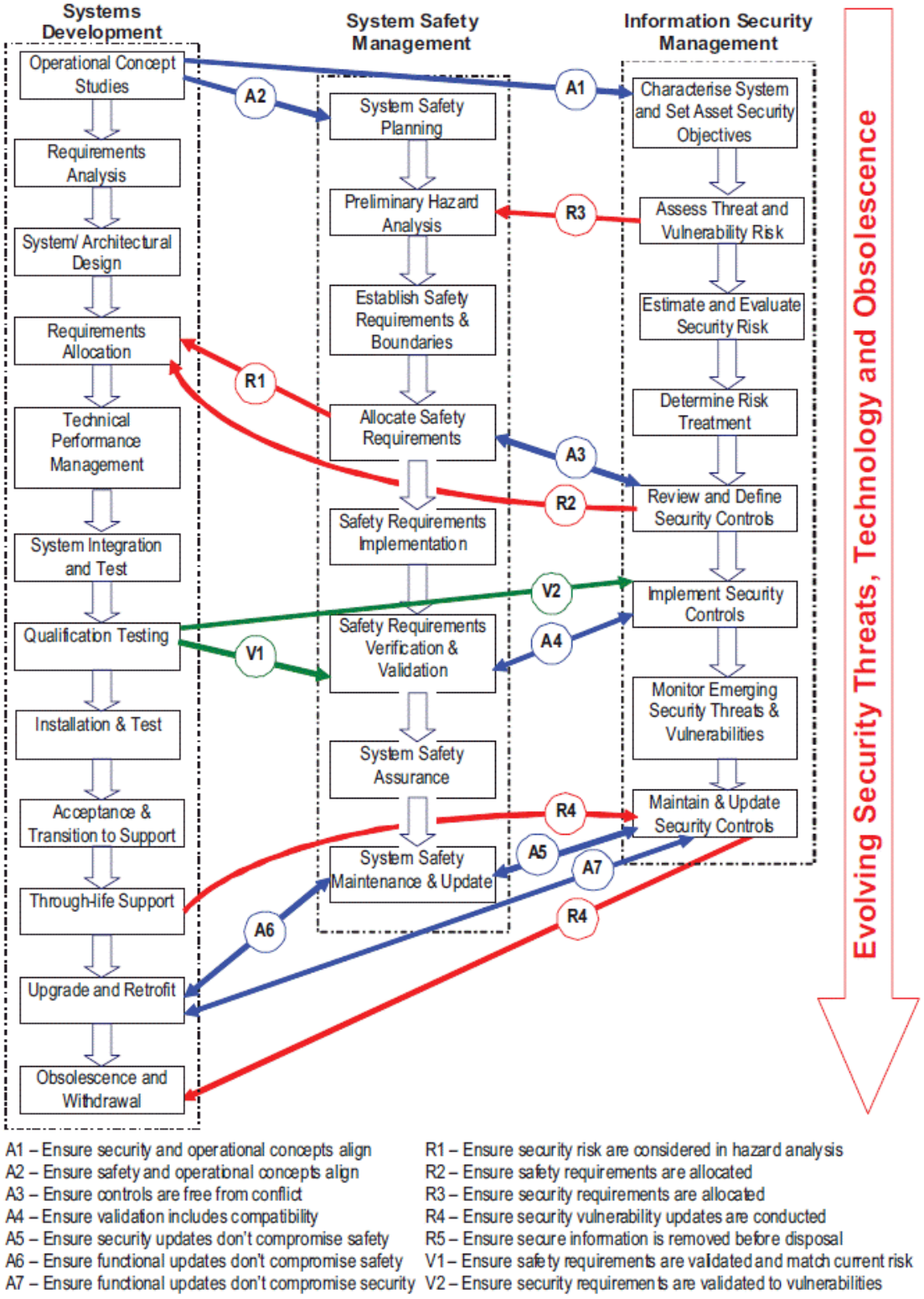


Figure 8: Key lifecycle alignment points

After the system description and functional requirements are developed, the preliminary hazard analysis (PHA) identifies potential hazards of the system, their consequences, causes and countermeasures. The risk management process consists of safety requirements specification; these requirements are used together with existing security policies in order to specify security requirements. Next, security threats and vulnerabilities are identified using the security-HazOp technique issued by the CORAS project. The

aim of the security-HazOp method [56] is to identify critical security-related deviations from intended behavior with a focus on confidentiality, integrity, availability and authenticity (CIA attributes); it uses a brainstorming activity based on a set of keywords and attributes that are the negative counterparts to the CIA attributes: disclosure, manipulation, denial/delay and fabrication/masquerade.

The risk analysis process enables the identification of security threats, their effects on the safety of the system and their associated likelihoods, which enables risk levels to be estimated. Unacceptable safety consequences are treated by introducing safeguards, and the risk management process iterates until the safety of the system is acceptable. This may create new security requirements. The system description is updated by the introduction of new safeguards, which enables the final system to be designed. The system is finally implemented and tested to ensure that all security and safety requirements are fulfilled [23].

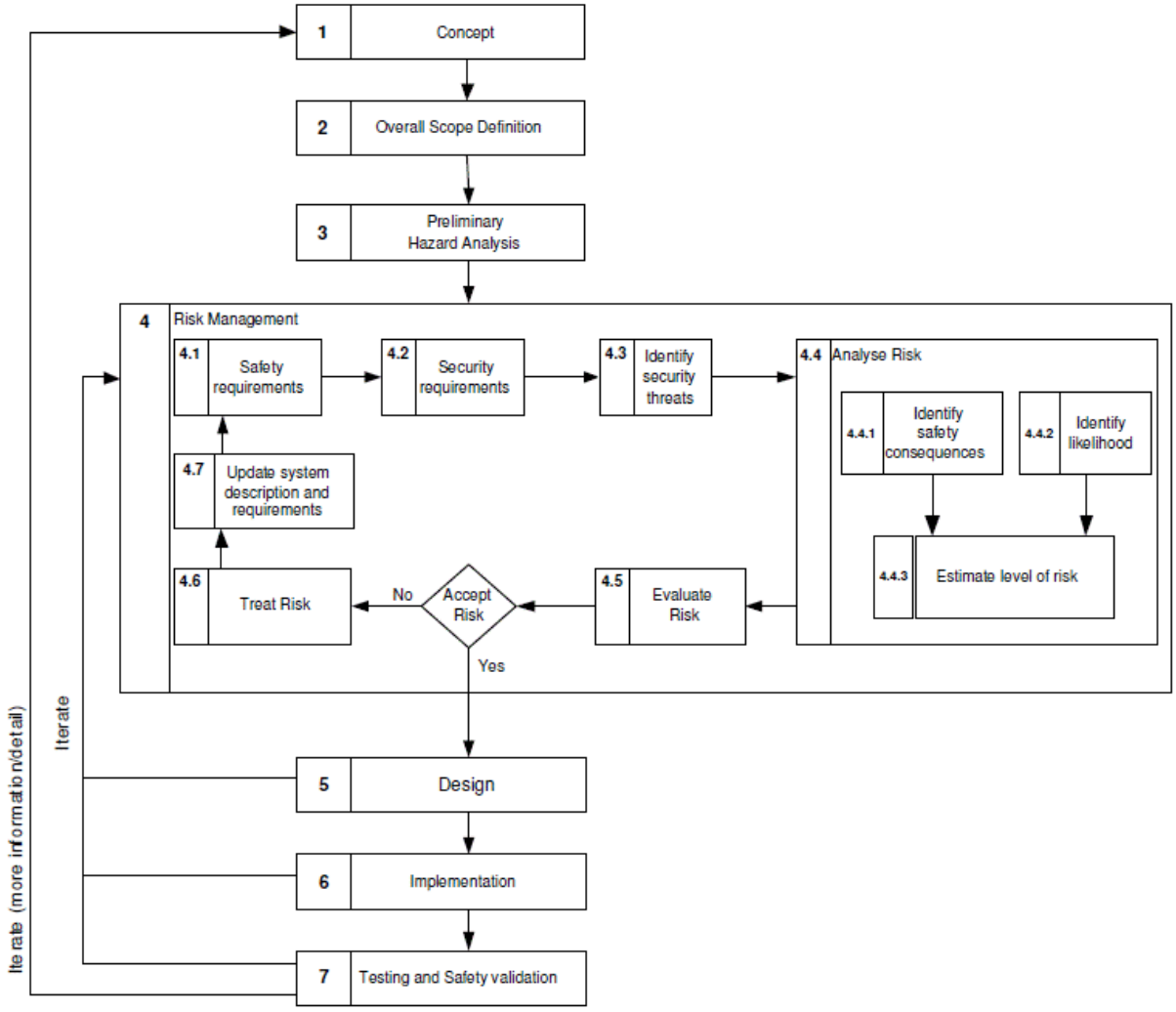


Figure 9: Security-safety lifecycle

This approach was applied to a critical system comprised of a cutting robot that is protected by a gate and supervised by another robot that communicates with a PC controller, thereby proving that the model adequately covers safety and security requirements, hazards and threats. Moreover, the overall architecture of the system was described according to these requirements. The relationship between security threats and safety consequences in typical attack situations was addressed.

Another approach is "an integrated process involving both safety and security allowing achievement of Design Basis Threat (DBT) objectives while ensuring safety" [57]. This process involves collaboration among safety and security professionals and requires a common understanding of the requirements,

concepts and terms on which safety and security stakeholders rely. A flowchart describes the different steps to follow in order to deploy a new or upgraded security system while ensuring compliance with safety requirements. It starts from defining the need for a protection measure, including system selection, system approval, system procurement, installation and deployment.

The security system, selected among the list of alternative security systems that meet the overall objective, must verify the complete list of minimum acceptable requirements (i.e., "musts") and should satisfy as many desirability criteria (i.e., "wants") as possible. It should not only satisfy security needs, but also consider characteristics such as operational efficiency, cost and safety. The safety and security implications of the selected security system are discussed. Safety implications include analyzing facility hazards and accidental or inadvertent discharges caused by human error, faulty security system design, and internal or external hazards that may lead to hazardous material releases, fires, nuclear criticality, leaks, or damage to safety structures, systems and components (SSCs) or process systems. The implications of safety changes to security are also discussed, as security upgrades may impact safety, and conversely, a new or upgraded safety system may have implications in the security domain. The whole process should include stakeholders from the safety and security fields to study impacts of the new or upgraded security module on the whole system and whether it creates conflicting requirements. Finally, security and safety processes should be coordinated to ensure safe and secure operation.

Schmittner *et al*. [58] proposed an approach for the combined analysis of safety and security in which the basic failure mode and effect analysis (FMEA) of cause and effect is extended to include security related aspects. This combined risk analysis method is called Failure Mode, Vulnerabilities and Effects Analysis (FMVEA). The safety analysis flow chart of the FMEA described in IEC 60812 is extended to include security in the analysis. According to the FMVEA analysis flow chart, the failure and threat modes are identified separately and their effects are assessed. The activities of the FMVEA are similar to the activities defined by the EBIOS methodology mentioned in Section 1.3.2.

The SQUALE (Security, Safety and Quality Evaluation for Dependable Systems) project [59] aimed at integrating safety and security concepts in a combined harmonized approach. It provided generic Dependability Criteria that describe how to gain confidence in the correctness and effectiveness of systems with high safety and security requirements. Those criteria had been applied to existing systems from the railway and mobile communication sectors.

Bieber *et al.* [6] published the main results of the Safe and Secure Embedded Aerospace Systems (SEISES) project, in which members of the French aerospace industry collaborated to develop a standard for developing safe and secure embedded systems and leveraging synergies between safety and security. The aim of the project was to integrate safety and security into the lifecycle (i.e., design, development, evaluation, validation, and maintenance) of embedded and communicating systems. Existing aerospace safety standards (ARP-4754, DO-178, DO-254) were used together with security standards (ISO/IEC 27005, Common Criteria). Possible synergies between safety and security also are outlined. In [6], the global process that merges safety and security assurance is described. It defines safety and security activities for assessment and development. Safety and security assurance activities converge at the development level and assessment activities that share the same system description level interact (e.g., security threat identification and functional hazard analysis). The act of risk identification assesses the safety impact of loss of security for each asset.
The main result of the project is the elaboration of a referential that covers safety and security assurance requirements. SEISES assurance objectives were validated by three demonstrators from the aerospace industry.

In the context of the SESAMO project, the Medini Analyze tool supports a methodology for extracting safety-security cross-influences from a system model. It provides a framework for performing a joint qualitative analysis of safety and security by integrating failure modes and effect analysis (FMEA) and fault tree analysis (FTA) techniques with their extensions to security (Security-FMEA and Attack Tree). The approach [60] is built on four different activities: (1) the identification of safety requirements using

FMEA and FTA, (2) the identification of security requirements using Attack Tree analysis and Security-FMEA, (3) the identification of safety functions and their impact on security and vice versa (by several iterations of FTA, FMEA, Security-FMEA, and Attack Tree), and (4) the identification of design synergies in safety and security enhancements (by cross-referencing safety malfunction effects with security violations, which could reveal solutions that serve both safety and security). The combined use of these techniques enables cross-references from security analysis results to safety analysis results which facilitates the identification of possible interdependencies and leads to potential synergies in implementing mitigation/prevention functions and countermeasures.

The process-oriented approaches identified in this section treat safety and security at early phases of the system engineering particularly in the concept and requirements phases. They are then more useful for designing new safe and secure systems. These approaches still deal with safety and security at a very high level. However for existent systems it is more relevant to treat safety and security in details in order to identify possible interactions. We believe that model based approaches are more suitable for this purpose.

We outline that in our work, we are interested in operational systems that have a long service life like power plants, refineries, dams, etc. We elaborate in the following section an exhaustive survey of model-based approaches, identified in the state-of-the-art, that consider safety and security jointly.

2.3 Model based approaches

We classify the model-based approaches identified according to whether they rely on graphical or non-graphical models.

2.3.1 Graphical modeling approaches

The myriad of graphical approaches identified have been classified in eight categories, according to whether they are based on: semi-formal safety/security cases, fault/attack trees, Petri nets, Bayesian belief networks, Unified Modeling Language (UML), model-based system engineering, formal verification; we finally identify approaches that are specific to electrical networks.

2.3.1.1 Semi-formal safety/security cases

Assurance cases include safety cases and security cases. A safety case should communicate a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context [61]. The same definition is applicable to security assurance cases.

Goal structuring notation (GSN) [62], a graphical argumentation notation, is used to represent the individual elements of any safety/security argument (requirements, claims, evidence and context) and the relationships that exist between these elements (i.e., how individual requirements are supported by specific claims, how claims are supported by evidence, and the assumed context that is defined for the argument). The Claims-Argument-Evidence (CAE) notation, developed by the consulting company Adelard as an alternative to GSN, could also be used for security purposes, instead of their initial safety usage.

The SafSec approach [5][63], which originated from the eponymous research project by the UK Ministry of Defense, is a framework for certifying integrated safety and security requirements: "The first state of the process is to identify both safety and security risks through a systematic process, and then to identify suitable control measures. At this stage conflicts and gaps between requirements can be identified and resolved." The SafSec approach enables module boundary contracts[7] to be identified within the specification. Five components of the contract should be considered: the guarantee clause; the reliability clause; the context clause; the assurance level and counter evidence.

[7] A module boundary contract (MBC) specifies verified details of the module properties at a given abstraction level.

To support security accreditation and safety assurance, the SafSec approach uses conventional graphical argument techniques—namely, the GSN syntax. There are three main threads to the argument: the first addresses the security argument and the identification of system threats and vulnerabilities, as well as mitigations to reduce the security risk; the second thread deals with the safety argument and the identification of safety hazards and accidents; the final thread addresses requirements verification and validation to argue that the system is fit to enter service.

Johnson [64] used assurance cases for a high level risk analysis. Relying on semi-formal argumentation techniques (GSN in particular), he identified three ways to integrate safety and security concerns for interactive systems:

- Integration within a single assurance argument: In order to demonstrate the top level goal, the safety and security of a complex system, arguments should first demonstrate that the system is acceptably safe then provide evidence that the system is acceptably secure. However, it is difficult to show that some security evidence has implications for system safety and vice versa.

- Integration of safety concerns into security assurance cases: First, the security assurance case is built, and then nodes are added to distinguish evidence or arguments about security concerns that might undermine the safety of any implementation. This enables the identification of potential safety concerns associated with every threat or vulnerability. Safety hazards that are not related to the security assurance case, however, are not represented.

- Integration of security threats into safety cases: First, a conventional safety case is developed, then threats and vulnerabilities are analyzed in order to identify security concerns that were not identified during the previous step and mitigate them.

This last approach was adopted later by the author. It enables safety and security arguments to be integrated into a single graphical structure. The same approach is used in the SESAMO project, under the name “Security Informed Safety Cases." The main ideas were given at an early stage of the project in deliverable D31; they have been refined thanks to experimentation on use cases. Once the standard safety case is available, the aim is to discover how the existing claims and arguments are affected by security considerations, and whether any new claims and arguments are necessary.

Subramanian *et al.* [65] introduced a similar goal-oriented approach called the non-functional requirements (NFR) approach to evaluate security and safety in an integrated manner. The NFR approach enables the safety and security properties of a system to be represented as goals and uses qualitative reasoning to evaluate whether these properties have been achieved or not. A well-defined ontology enables the system architecture and its safety and security requirements to be represented using “Softgoal” contributions and propagation rules [66]. The graph that captures these elements is called the Softgoal Interdependency Graph. Applying the NFR approach enables the co-evaluation of both safety and security and results can be used to handle tradeoffs between the two to improve the system.

2.3.1.2 Fault/attack tree based approaches

Extended fault trees

The approaches presented in this subsection rely on “static” fault trees or attack trees in which dependencies (e.g., sequences) are not taken into account.

Taking an integration approach, Fovino *et al.* [67] introduced extended fault trees, in which attack trees [68] are integrated into a pre-existent fault tree in order to extend traditional risk analysis (which captures only accidental risks) to include malicious risks. This integration is possible only if attack tree goals exist that also are events of the target fault tree. The approach also proposes quantitative analysis by

assigning probabilities to leaf nodes and calculating the aggregate probabilities of higher level events. An application of this method to a use case was described in [67].

Bezzateev [69] used fault tree analysis (FTA) to model risks introduced by the insertion of a “security module” into the Eurobalise Transmission System, which is a sub-system used in the European Train Control System (ETCS). The author likened “safety hazards of the security module” to accidental failures of this module and “security hazards of the security module” to the attacks initiated on this module. Both safety and security hazards are modeled by the same tree. The security module used in the example is an authentication module used to counter masquerade attacks. The author showed how integrating a new security module could increase the safety level of the whole system.

Kornecki *et al.* [22] also used FTA to develop safety and security requirements of a component used in the next generation air traffic management system and proposed appropriate mitigations. This component, called the Aviation Simulation Network Gateway (ASNG), is a system that acts as an intermediary between two components of the aviation network (the Real Time Distributed Simulation and the Aviation Simulation Network). It provides logic for two-way communication for the transmission of messages and storage of the exchanged data. The authors detailed hazards and threats that may lead to an aviation accident using a fault tree model. They placed a special focus on the ASNG system by detailing accidents or attack scenarios related to communication as the critical aspect in ASNG operations. After associating each basic event of the tree with its probability of occurrence, with the help of associated tools the authors provided quantitative analyses showing probabilities of events, availability, unreliability and other dependability attributes of the system. The fault tree model was used later to support the development of safety and security requirements and to specify mitigations.

Steiner *et al.* [70] extended the component fault trees (CFTs) [71] used in safety analysis by using attack trees (ATs) to model security events that can compromise the safety of a system. CFTs can be roughly described as reusable parts of fault trees that enable the failure modes of one component to be modeled; top level events of a CFT are "out-ports," while “in-ports,” are connected to events coming from other components. According to the authors, CFTs are more suitable for modeling large systems than fault trees.

The authors started by modeling one CFT per system component. In the next step, CFTs are extended to include security concerns that influence system safety. Basic events that also can be caused by malicious intervention are elaborated. Components interfacing with the system environment are the most vulnerable to attacks. The STRIDE classification (spoofing/authentication, tampering/integrity, repudiation/non-repudiation, information disclosure/confidentiality, denial of service/availability, elevation of privilege/authorization) enables possible attacks on such components to be found. The extended CFT contains both safety and security events.

This approach is supported by qualitative and quantitative analysis. Qualitative analysis provides ordered lists of minimal cut sets (MCSs) (sets of basic events which together create the top level event of the tree) sorted according to size (based on the number of basic events in each) and whether they contain only safety events, only security events, or both. Smaller MCSs and those that contain only security events are more critical. Quantitative analysis consists of assigning values to basic events: probabilities for safety events and a simple rating (low, medium, high) for security events, considering that assigning probabilities for security events is not appropriate. A mixed MCS will be assigned a tuple (P, R) where P is the product of all probabilities of the included safety events and R is the minimum of all ratings attributed to basic security events. The tuples of mixed MCSs can be ordered by probability or by rating. The authors illustrated their approach by analyzing an example.

BDMP: Boolean logic Driven Markov Processes

BDMP are a graphical modeling formalism initially designed for safety and reliability assessment [72]. This formalism combines classical fault trees with Markov processes, thereby providing not only good readability and hierarchical representation (like fault trees), but also advanced quantification capabilities. Unlike static fault trees, the BDMP formalism enables dynamic features to be modeled with a special type of link called "trigger." The BDMP formalism has recently been adapted to the security field [73]. In [32], Pietre-Cambacedes shows the ability of this formalism to create models integrating safety and security risks that may lead to the same unwanted event. The qualitative and quantitative capabilities of BDMP enable safety and security risk combinations and interdependencies to be studied. This approach will be further investigated in Chapter 3.

2.3.1.3 Petri net based approaches

Stochastic Petri nets (SPNs) have been used to model the behavior of cyber-physical systems and to analyze the effect of intrusion detection and response on the reliability of a system or to assess vulnerabilities in SCADA systems [74]. Mitchell *et al.* [75] provided a high level SPN model for a CPS with intrusion detection and response. The state of the system is described by the distribution of tokens in the SPN model. Values are then assigned to the model's parameters and used to evaluate the reliability indexes according to the attacker's behavior, the detection level and intrusion response.

Flammini *et al.* [76] introduced a model-based methodology for quantitative estimation of the vulnerability of physical protection systems. The proposed methodology uses generalized stochastic Petri nets (GSPNs) to model dynamic aspects of the system. Petri net patterns are defined in order to create vulnerability models and capture behavioral aspects of assets and actors involved in the attack scenario. The vulnerability model is comprised of the attack model, the sensing model, the assessment model, the intervention model and the supervisor model.

The authors argued that the use of pattern oriented modeling addresses issues associated with the increasing complexity and heterogeneity of systems. They use the ORIS tool that enables SPN modeling and analysis. The effectiveness of the approach was demonstrated using a case study in the mass transit domain. The authors dealt only with physical security in their paper; however, the approach can also model cyber security related risks.

Roth *et al.* [77] proposed state/event fault trees (SEFTs) as an approach to modeling and jointly analyzing aspects of safety and security. SEFTs combine fault trees and state charts into one model, which enables deterministic state spaces and probabilistic failure behavior to be modeled. In the SEFT formalism, communicating components are modeled and failure propagation is facilitated with in- and out-ports. The temporal dependencies between components are modeled with state charts: the state changes can be triggered by exponentially distributed probabilistic events, deterministic events and triggered events. Events are guarded by states and connected to them temporally, while causal dependencies between components are modeled by gates as in typical fault trees, using causal connections.
The ESSaRel modeling tool enables the trees to be modeled and converted into extended deterministic stochastic Petri nets for quantitative analysis; which are then analyzed using the TimeNet tool and steady state analysis or Monte Carlo simulation.

In the SEFT based model, basic vulnerabilities are modeled, such as denial of service (DoS) failures for components and exchanged messages, spoofing, bypassing and reprogramming. The adversary is also modeled by an attack component, subcomponents of which represent attack steps. Attack steps are interconnected and form a logical attack queue. Each attack step can be split in a state chart where different transitions of the attack cycle are detailed based on the quantitative parameters. The authors illustrated their approach on a tire pressure monitoring system; they studied the effect of cyber-attacks

on the safety properties of the system and proved a mutual reinforcement interaction between safety and security in the case of a spoofing attack. Finally, SEFTs yield a modeling and analysis approach that integrates security aspects into a safety model and enables quantitative analysis of safety and security.

2.3.1.4 Bayesian belief network based approaches

A Bayesian belief network (BBN) is a graphical model that represents random variables and their conditional dependencies. BBNs have long been used for safety assessment [78], particularly to include uncertainties [79] or human and organizational factors [80] in risk evaluation and decision making, and recently for security risk assessment [81] [82] [83].

Kornecki *et al.* [84] addressed safety and security jointly using BBNs. They studied interrelationships between safety and security by measuring their impacts on each other and on system reliability. Based on the case study in [65] (i.e., an oil pipeline control system), the authors used BBNs to model system components and the safety and security requirements. The model was then used to evaluate the impact of safety accidents on security aspects and the impact of violating some security requirements on the safety related events.

2.3.1.5 UML based approaches

Misuse cases

Use cases are one of the behavior diagrams of the Unified Modeling Language (UML). They are used to specify the required behavior of a system. In [23], UML use cases were used to illustrate the functional requirements of the system in the third step of the safety-security lifecycle shown in Figure 9.

Misuse cases, first introduced by Sindre and Opdahl [85], are the inverse of use cases. They model prohibited functions or undesirable behaviors caused by either the system or the involved stakeholders. Misuse cases have been used to elicit safety or security requirements [86]. Sindre [87] used misuse case diagrams combining both safety and security threats for the same system.

Chassis

The combined harm assessment of safety and security for information systems (CHASSIS) method [48] defines a unified process for safety and security assessments based on UML notations. The CHASSIS unified process shown in Figure 10 is comprised of three main activities: eliciting functional requirements, eliciting safety/security requirements, and specifying safety/security requirements. These activities rely on the use of UML-based diagrams, i.e., use cases (UCs), misuse cases (MUCs), sequence diagrams (SDs) and misuse/failure sequence diagrams (MUSDs/FSDs); and traditional safety techniques, i.e., HAZOP (hazard and operability) studies and FMEA (failure mode and effects analysis). The CHASSIS process flows from left to right and top to bottom in Figure 10. The double-headed arrows between the boxes symbolize that activities should be performed iteratively [88].

The assessment process begins with the definition of system functions and services (Steps 1-3) and continues through the elaboration of use cases and sequence diagrams. In the second stage, MUCs are created (Step 4) based on UC diagrams from the previous step and with the help of guide phrases composed from a set of HAZOP guidewords. Textual MUCs (Step 5) provide more details on the MUC diagrams. Using FSDs and MUSDs (Step 6), stakeholders refine the harm scenarios by detailing the sequences of events between different actors, and then try to create new mitigations to improve safety and security. Tradeoffs between conflicting safety and security mitigations are performed. Based on T-MUC, HAZOP tables are prepared (Step 7) and safety or security requirements are defined (Step 8) [48].

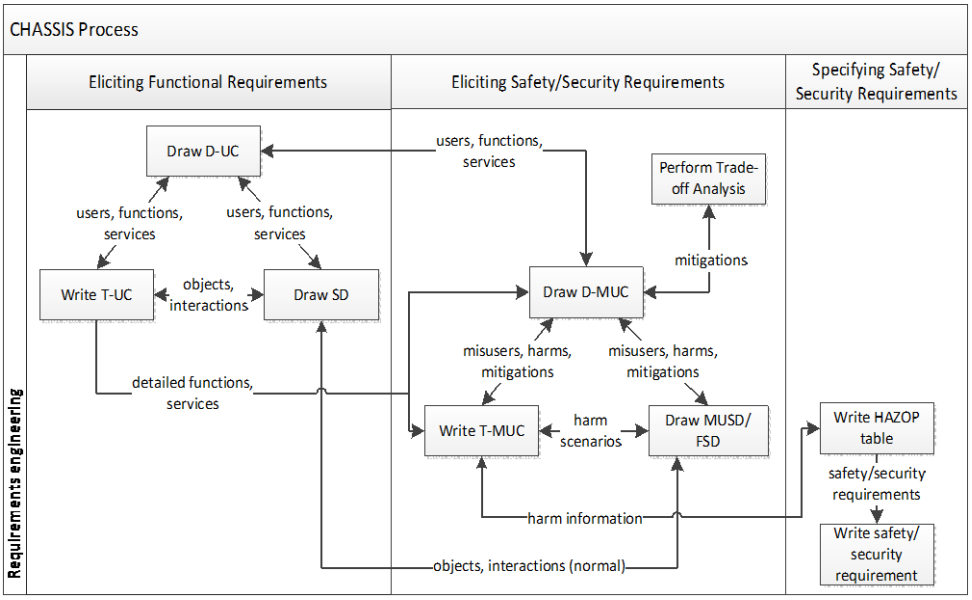


Figure 10: CHASSIS process diagram

UMLsafe/UMLsec

UMLsec [90] extends UML to allow security relevant information to be expressed within UML diagrams in a system specification. The extension is given in the form of a UML profile using the standard UML extension mechanisms. *Stereotypes* define new types of modeling elements, thereby extending the semantics of existing types or classes in the UML metamodel. They are used together with *tags* to formulate security requirements and assumptions about the system environment; *constraints* provide criteria that determine whether the requirements are met by the system design [91]. The approach allows also modeling of potential adversary behavior (including insiders). Similarly, for safety, UMLsafe extends UML for safety systems development [92] by enabling analysis of safety requirements, failure scenarios, fault tolerance, etc.

The UMLsafe/sec [93] approach considers safety/security during early phases of critical system design. It enables modeling of safety/security requirements, failure/attack scenarios, and concepts such as fault tolerance/cryptography; and ensures the UML specification provides the desired level of safety/security. The approach is supported by the UMLsec-Tool framework, which provides automatic verification plugins of UML models for critical requirements. In particular, it includes automated analysis of UMLsec models for the security requirements. This tool was re-implemented and integrated into Eclipse. The new tool, CARiSMA, is based on the Eclipse Modeling Framework (EMF) and enables modeling and code generation. It is also extensible for new languages and enables compliance, risk and security analyses and checks [94].

SysML-sec

SysML-sec, introduced in [95], is a SysML-based model driven engineering environment for designing secure and safe real-time embedded systems, but the approach can also be applied to industrial systems. The SysML-sec methodology enables assessment of the impact of security requirements and mechanisms on system safety based on three SysML stages: system analysis, system design and system validation. In the analysis stage, requirement diagrams are used to model security requirements and their mutual relationships and attack diagrams are used to model attack scenarios. The partitioning of the system is modeled by graphs that represent communicating functions, the architecture, and the mapping of functional elements to assets. In the design stage, security mechanisms are defined and security requirements are refined in security properties. The validation stage provides formal evidence whether

security properties are verified and whether the system is safe and resilient to threats. Apvrille *et al*. used the AVATAR SysML environment [96] for the formal verification of safety and security properties.

2.3.1.6 Model-based system engineering (MBSE) methods

Brunel *et al.* [97] proposed a model-based approach to address safety and security assessment of system architecture. The approach models the system with three different viewpoints: a design viewpoint in which system engineers describe the system architecture using the "Melody" tool [98], a safety/security viewpoint in which safety and security engineers model safety and security properties using the "Safety Architect" tool [99], and finally a viewpoint of the formal model, expressed in the "Alloy" language [100] and used to assess safety and security properties of the system architecture. The safety and security models contain two kinds of information: dysfunctional behavior and the properties to be validated. The former describes how failures or attacks are propagated in the system architecture, while the latter expresses the safety or security requirements that the system architecture must satisfy. The two models are then combined into a single Alloy model and used as input for the Alloy Analyzer, which generates formal validation of safety and security properties. A counter-example is generated in the case of requirements violations. The feasibility of the approach has been tested on a geo-localization system in a case study from the avionics domain.

2.3.1.7 AADL

Delange *et al.* [101] proposed an approach to modeling security and safety concerns in architecture analysis and design language (AADL). The AADL language allows non-functional aspects of components to be expressed, such as security, safety, and interface specifications, and how the components are interconnected. To model a system with safety and security requirements, the authors began by modeling the partitioned architecture then annotated the model with properties dedicated to security and safety policies. The safety and security requirements were then verified and validated.

2.3.1.8 Formal verification methods

Formal methods use mathematical techniques for specification and verification of a system's behavior. Based on logical reasoning, they enable to check whether the functional and non-functional (i.e. safety and security) properties are satisfied by the system design and implementation. Formal methods tools fall into two categories: model checkers that enable to check the design with respect to the specified properties encoded in a modeling language; and theorem provers, also called "proof assistants", which combine automated techniques with manual guidance to prove correctness. These methods are used in industries with safety critical systems like nuclear, railway and aerospace.

Smith *et al.* [10] outlined the use of formal methods in the railway industry for modeling, requirements specification, design and validation of safety critical systems. With the evolution of modern railway infrastructures, the authors discuss some of the interrelationships between safety and security, and stress the need for security in safety-critical systems.

Zafar and Dormey [102] used a genetic software engineering (GSE) method to design an ambulatory infusion pump that must satisfy a number of safety and security properties. The GSE development process consists of the following steps:

1. Specify the system's requirements and integrate the safety and security requirements, then translate each requirement into a behavior tree (BT).
2. Integrate the individual BTs into an IBT (integrated behavior tree) to generate an integrated view of requirements, and resolve integration defects.
3. Systematically refine the IBT into a design behavior tree (DBT), which visually depicts the impact of each design decision on the complete system as the changes are applied to the integrated view of the requirements.

4. Translate the IBT and/or DBT into specification languages (e.g., Communication Sequential Process, Symbolic Analysis Laboratory) for formal verification of the specifications using the SAL-SMC tool.
5. Derive the component-based architecture and individual component behavior from the DBT, which are typically represented in Component Interaction Network (CIN) and Component Behavior Tree (CBT) diagrams.

By applying this method to a case study of an ambulatory infusion pump, the authors uncovered a subtle design flaw which caused them to recognize the importance of using formal methods in design of critical applications.

2.3.1.9 Approaches for electrical networks

Electrical systems require specific methods and tools because their behavior cannot be properly studied without considering the laws of physics. Modeling all possible transient phenomena in an electrical network with precision requires a lot of computing power. This is not feasible with stochastic models, which contribute to this complexity because numerous simulations with various configurations of system failures must be performed, typically in a Monte Carlo simulation. Therefore, in all of the articles cited in this subsection, the physical equations are simplified in order to reduce them to a load-flow problem. This means that after each random event (failure, attack, repair, etc.) the load-flow is recalculated in order to obtain the voltages and intensities at every point of the network. It is then possible to determine the amount of non-distributed energy, a major indicator for measuring the performance of electrical networks. This simplification assumes that the transients consecutive to random events cannot lead to divergent behavior, eventually causing the network to collapse completely.

The CRUTIAL approach [103][104][105] models and quantifies interdependencies in electrical power systems between the electrical infrastructure (EI) and the information infrastructure (II) that implements the EI control and monitoring system. It includes both qualitative and quantitative analysis and evaluation methods [104] and uses stochastic modeling techniques. Beccuti *et al.* [104] [106] used the CRUTIAL modeling framework [107] to model an electrical power system and study the effects of a denial of service (DoS) attack that compromised the communication network used to control the EI remotely. The authors used two different approaches to the modeling and quantification of dependencies between the EI and the II. The first is a stochastic activity network (SAN) model to represent the structure of the power grid and its physical components. The control part behavior is modeled at an abstract level. The second is a stochastic well-formed net (SWN) that models the control system and the behavior of the attacker, and enables stochastic assumptions about the EI behavior. The first model more rigorously represents EI behavior, while the second one is more faithful to the attack scenario and the control algorithms.

Both models represent interdependencies between the II and EI subsystem and the subsequent cascading or escalation failures caused by malfunctions, either at the cyber or the electrical levels.
The interaction between the two models is outlined. The authors also evaluated the effects of a DoS attack on the percentage of the mean power demand that is not met over the interval of time [0, t] and the number of unavailable local control stations (LCSs). The evaluated attack effects cover the panel of dependability attributes, which also includes safety.

Preliminary interdependency analysis (PIA) [108] is a generic methodology for finding and assessing the impact of dependencies between telecommunication infrastructures and electrical grids. It begins with a qualitative analysis to specify the system boundaries and scenarios, and identify the modeled elements and their interdependencies, and is followed by a quantitative analysis based on a Monte Carlo simulation. This simulation relies on an ad-hoc model that can take various forms. A partner in the SESAMO project, City University, developed a simulator from scratch, written in JavaScript (and C++ for the load-flow calculations) in the node.js framework. Thanks to this simulator, it is possible to see the effects of failures and attacks on the amount of non-distributed energy. This approach is very similar

to the CRUTIAL approach, except that the Petri net part of the model is replaced by stochastic state machines.

The domain of electrical networks and their instrumentation and control systems is still largely unexplored and it is likely that the results of different tools, if they were applied to the same use case, would be quite different. This is because every tool has to make assumptions about the reactions of the system when it needs to be reconfigured after failures or attacks. It will take years before the various tools converge more or less towards the same hypothesis and become comparable, like the tools used in safety analyses of simpler systems.

2.3.2 Non-graphical modeling approaches

2.3.2.1 Non-formal approaches

Reichenbach *et al.* [31] proposed a framework for assessing both safety and security for safety critical systems within industrial automation and control systems. The proposed approach combines safety analysis with security analysis by considering the safety integrity level (SIL) of IEC 61508 as an extension of the threat vulnerability and risk assessment (TVRA) method of ETSI TS 102 165-1 [109]. The TVRA method evaluates and calculates factors that are associated with the risks posed by the threats, which are time, expertise, knowledge, opportunity, equipment, asset, impact and intensity. The likelihood is calculated based on the attack potential value, which is calculated using the factors of time, expertise, knowledge, opportunity, and equipment. The impact is calculated from the asset impact value and the attack intensity value. The authors extended the TVRA method to include the safety integrity level as one of the factors affecting risks, which they consider in assessing the impact. The authors inferred that the higher the SIL, the higher the resulting impact will be.

This approach is supported by the ETSI tool, which uses the factors affecting the risks to calculate their occurrence likelihood and impact. The authors illustrated their approach on an example from the automation industry. They considered two attacks with the same occurrence likelihood that compromised two different safety functions with different SILs. They inferred that a higher SIL has a higher impact on the final risk, which shows that the SIL has an influence on the entire TVRA while being compatible with the security threat analysis and the safety standards.

Holstein and Singer [110] described the work led by ISA 99 on safety and security coupling. Four levels of security assurance are defined, with levels 4 and 3 being the most critical: they are assigned if failure of the security protection mechanism could result in loss of life or total failure of the industrial automation and control system's operating capability. The relationship between security assurance level (SAL) and safety integrity level (SIL) is given by a simple correlation; SIL 3 and SIL 4 systems should be related to SAL 3 or SAL 4. To estimate the SAL, the authors use a consequence-based analysis to establish weighting coefficients for contributing security mechanisms. They applied their approach to a chemical truck loading control and emergency shutdown safety system: the SIS components were grouped into separate zones, security metrics were defined and all system components with access to the safety system were mapped to a weighting factor, thereby coupling safety and security. The security assurance level of the system is expressed as the normalized sum of the weighted security levels of the components.

Depoy *et al*. [111] described a methodology that addresses the risk of combined physical and cyber-attacks against critical infrastructure facilities. The authors considered four types of attacks: physical-only, cyber-enabled physical, cyber-only and physical-enabled cyber-attacks. A top-down functional assessment methodology is used for risk assessment. It consists of the following steps:

1. Consequences of concern (CoCs) and the associated unacceptable system states are determined.
2. The engineering process model is used to determine the "cut sets" that cause a given CoC.
3. The vulnerability assessment process defines protective measures, threat model and probability of success of given an attacker.
4. The conditional risk set developer evaluates the risk associated with CoCs and appropriate mitigation measures.

Pieters *et al*. [112] proposed the factor analysis of information risk (FAIR) framework for the integration of safety and security risk assessments. The proposed approach introduces an alternate way to quantify security related metrics from the frequency-based approach used in safety. The authors outlined the key difference between accidental and malicious threats: "Whereas accidental threats occur randomly, the occurrence of malicious threats is based on attacker decisions." In the attacker model, the attacker bases his strategy on the system properties and his effort to perform an attack scenario is a function of time. The security risk includes not only the event frequency but also vulnerability, expected damage, and countermeasures. A theoretical risk assessment framework for accidental and malicious threats is provided in the paper.

2.3.2.2 Formal verification approaches

Sun *et al.* [35] proposed a framework for detecting safety and security conflicts using the Maude rewriting logic language [113]. The proposed framework defines a world composed of a model (classes and objects), propositions, assumptions and requirements. The process of detecting conflicts between safety and security consists in searching all configurations that are safe but unsecure and all configurations that are secure but unsafe. This provides a communication mechanism between different designers that enables them to detect impacts of their requirements on the rest of the system. This method is illustrated by the example of the exit door that must be open is case of fire and locked to prevent unauthorized persons from entering.

Simpson *et al.* [114] used the non-interference concept [115] from security to model properties of safety-critical systems. Using the communicating sequential processes language, the authors modeled some failure modes, particularly fail safe (the occurrence of failures does not affect the safety of the system), fail soft (neither failures nor the associated recovery measures affect the safety of the system), and fail operational (neither failures nor the associated recovery measures affect the functional aspects of the system and its safety).

2.3.2.3 STPA-sec

System theoretic accident model and processes (STAMP) is an accident causality model developed by Leveson [116] that accounts for new causal factors associated with software, human decision making, new technology, social and organizational design, and increasing complexity. Based on the STAMP causality model, system theoretic process analysis (STPA) is a hazard analysis technique that identifies accident scenarios that encompass the entire accident process [117].

In STAMP, the systems are viewed as hierarchical structures in which higher levels control processes at lower levels and the lower levels send feedback to the higher levels. STPA focuses on control actions. It examines each control action under different possible conditions (e.g., providing a control action too late) and identifies whether these conditions lead to hazards. The STPA methodology starts by identifying hazardous states of the system then building its control structure with the fundamental building blocks that involve the control actions. Next a two-steps analysis is done: in Step 1, inadequate control actions are identified using hazardous states and the control structure, and in Step 2, the causal factors leading to the unsafe control actions that violate the safety constraints are determined [117].

The STPA technique emphasizes component interactions and system dynamics. Asare *et al.* [118] confirmed the adequacy of the STPA approach with large complex cyber-physical systems for which traditional hazard analysis is not efficient.

Young *et al.* [119][120] introduced STPA-Sec as a new systems-theoretic approach to safety and security. It consists of a top-down approach that focuses on security analysis as a higher level problem of assuring the overall function of the system which "shifts away from guarding against attacks" towards broader socio-technical vulnerabilities. STPA-Sec identifies required constraints on unsecure control actions that place the system in unsafe states when subjected to intentional and unintentional disturbances. In STPA-Sec, unsafe and unsecure control actions are identified based on the control structure model that provides graphical specification of functional controls in the system. This approach was applied to a nuclear plant example in [120].

2.4 Critical Analysis

Based on the findings in this survey, we first define in Section 2.4.1 a canonical risk analysis process integrating both safety and security—what we call an integrated risk analysis process—that is relevant at different stages of the system lifecycle. In Section 2.4.2, we summarize in Table 2 the classification of approaches previously identified and discuss their pros and cons. In particular, we discuss their capability to identify and treat interdependencies between safety and security. We finally define in Section 2.4.3 criteria that we judge essential for a thorough risk analysis and classify again approaches according to these criteria.

2.4.1 A canonical life-cycle integrating safety and security

We assume a system lifecycle consisting of two main phases: the development phase (including requirements definition and design) and the operational phase. For non-existent systems, the risk analysis process (including safety and security) is performed in the development phase in order to define the appropriate system requirements and design. For existing systems, the risk analysis process should be performed at the operational level and the system should be modified according to the output of this analysis by adding detection and prevention modules. We provide in Table 1 an overview of the main phases of a system lifecycle and the stages at which the integrated risk analysis process could be required.

<table>
<tr><th colspan="4">Development phase</th><th>Operational phase</th></tr>
<tr><td rowspan="2">Concept definition</td><td colspan="2">Requirements definition</td><td rowspan="2">Design</td><td rowspan="2">Implementation, validation</td><td rowspan="2">Operation, maintenance
→ Risk analysis process</td></tr>
<tr><td>Functional requirements</td><td>Non-functional requirements
→ Risk analysis process</td></tr>
</table>

Table 1: System lifecycle

As mentioned in Section 1.3.3 by the DO-326 standard for aeronautical systems, there is a tendency to keep safety and security activities separated for critical systems when it is out of the question to sacrifice some safety requirements for security reasons. This is also true for the nuclear standard IEC 61513, in which the systems architectures are defined based on safety requirements while security concerns are considered relatively late in the system development lifecycle [121].

However, we believe that in the general case, the risk analysis process should combine both safety and security. In Figure 11, we provide a high level view of a safety and security risk analysis process inspired from the generic approaches identified in Section 2.2. The first step of this integrated risk analysis

process is to perform a hazard analysis to identify the hazardous/unsafe states of the system. Considering the definition of safety in the context of this survey, the hazardous states originate from the system and have an impact on the system's environment. Next, safety and a security risk analyses are realized separately by safety and security experts: safety-related scenarios are identified based on failure mode analysis and security-related scenarios are identified based on an analysis of threats and vulnerabilities that lead to unsafe states. Then, the scenarios are ranked according to frequency and impact, and appropriate safety and security requirements are defined. The two sets of safety and security requirements are next integrated and examined together in order to identify possible interactions. The treatment step addresses the different interactions identified (e.g., conflicting requirements). This step requires collaboration of safety and security experts in order to find solutions that satisfy both sides. New safety and security requirements are considered and interactions are then derived. The system modifications resulting from this first pass may introduce new risks; this is why the process iterates until all interactions are identified and no modifications are needed.

This risk analysis process can be applied to the development or the operational phase of the system lifecycle (with items being requirements or design features in the development phase or actual countermeasures in the operational phase).

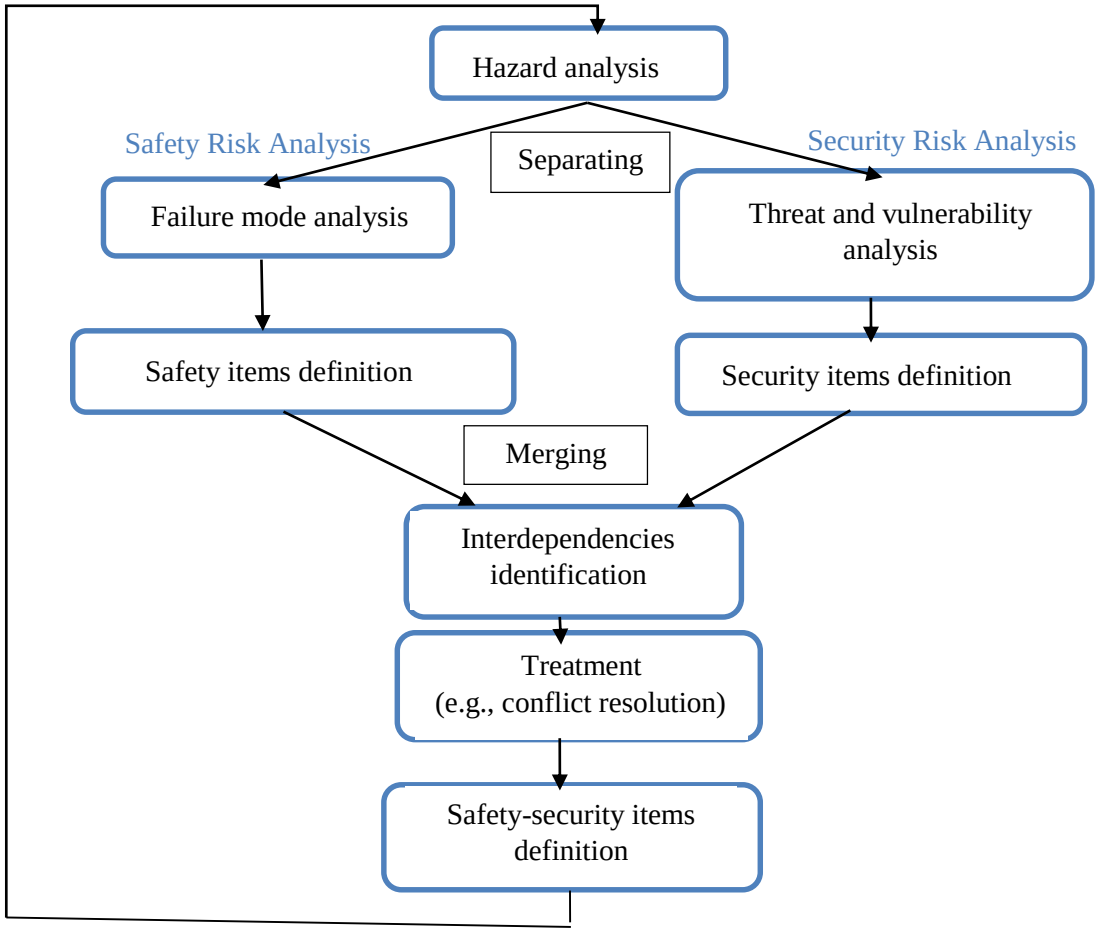


Figure 11: Safety security integrated risk analysis process

2.4.2 Analysis of the different approaches identified

We classify in Table 2 the different approaches identified in this survey according to different criteria: unification vs. integration approaches, coverage of the entire system lifecycle phase, qualitative vs. quantitative approaches, and process-oriented vs. model-based approaches. We later discuss the potential of each approach to identify and treat safety and security interdependencies (SSIs).

Process-oriented approaches rely generally on standards and are used, especially in early phases of a system's development, for the identification and specification of requirements that the system design must satisfy. These approaches aim at combining safety and security aspects in the risk management phase so that requirements will ensure the system is safe and secure. These approaches result in lifecycles or methodological processes considering safety and security together so that one cannot compromise the other, and in some cases, synergies are identified and leveraged. We believe that these approaches remain too macroscopic, purely qualitative in most cases, and require a lot of investment from engineers.

Model-based approaches rely on a formal or semi-formal representation of the functional and non-functional aspects of the system. For existing and operational systems, model-based approaches are more practical for modeling system's components and functionalities. With their qualitative and quantitative capabilities, such approaches are more likely to capture safety and security interdependencies. Actually model-based approaches require more knowledge from the analyst about the system. Moreover, they are generally able to scale up to complex systems and represent different aspects related to safety and security with different viewpoints and levels of detail. We stress in the following paragraphs the advantages and limits of the main model-based approaches identified.

Fault trees have been used for a long time in safety risk analysis, while attack trees, although younger and diverse, are gaining momentum in security modeling [122]. A combination of these modeling approaches, extended fault trees [67], aims to combine safety and security in the risk management process. The main limitation of the fault/attack trees-based methods resides in their static nature: these models are not able to represent sequences and more generally any dependence between basic events.

The BDMP formalism (cf. Chapter 3) overcomes the limits of extended fault trees by enabling the modeling of dynamic behaviors and providing rigorous quantitative capabilities thanks to Markov processes while keeping the advantage of a hierarchical representation which allows to go through successive refinements from an undesirable event related to the system to basic events related to the system components or processes. BDMPs generalize patterns of attack/fault trees but allow additionally a more accurate modeling, taking into account the dependencies, essential in the security field, but also in safety as far as we are dealing with repairable systems with dependencies between components.
Contrarily to what happens with the attack/fault trees, in which one must explicitly include the defenses in the form of sub-trees (the general principle is to represent the success of an action of the attacker by an AND gate with two leaves, representing the action itself and the failure of defenses provided for the attack), defense and response mechanisms are embedded in the BDMPs structure (see Section 3.2.2). A comparison between extended-fault trees and BDMP is provided in [32].

Bayesian belief networks have good quantitative capabilities based on Bayesian inference and a probabilistic framework that can include epistemic uncertainty. This technique, however, has many limits when it comes to dynamic modeling of complex systems related to their readability and computation time. Stochastic Petri nets and similar models such as stochastic activity networks exhibit high potential for qualitatively and quantitatively modeling safety and security risks in complex systems and their interdependencies. UML-based approaches provide good graphical visualization capabilities, and are well suited to qualitative modeling, particularly in the early phases of the system lifecycle. However, they do not provide quantitative analysis capability. We thoroughly compared in Section 3.3 BDMP with the CHASSIS method, which is UML-based.

Based on SysML, SysML-Sec provides, in addition to graphical SysML diagrams, a more formal framework for safety and security requirements verification that covers the analysis, design and validation phases thanks to model checking. Formal methods have potential for treating safety and security interdependencies by enabling requirements specification and verification. However, the use of these methods requires adequate knowledge of associated formalisms and languages, which can be costly in terms of time at the beginning. The application of formal methods is also limited by two factors [10]: they are not practical in the case of highly complex systems; and they cannot guarantee the validity of requirements specification.

Categorization Criteria			Unification vs. Integration		Lifecycle Phase		Qualitative vs. Quantitative	
Approaches			Unification	Integration	Design	Operational	Qualitative	Quantitative
Process-oriented	Stoneburner [50]		x			x	x	x
	Aven [51]		x		x	x	x	x
	Derock [36]		x		x	x	x	
	Woskowski [52]		x		x		x	
	Eames [21]			x		x	x	
	Johnson [53]			x		x	x	
	Kornecki [54]			x	x		x	
	Novak [30], [33], [34]			x	x		x	
	Hunter [28]			x	x	x	x	
	Sørby [23]			x	x		x	
	Ostby[57]			x	x		x	x
	Bieber [6]			x	x		x	
	Schmittner [58]			x	x	x	x	
Model-based	**Graphical Methods**							
	GSN [5], [28], [63], [64]			x	x	x	x	
	NFR [65]			x	x	x	x	
	Extended fault trees	Fovino [67]		x		x	x	x
		Bezzateev [69]		x	x		x	x
		Kornecki [22]		x	x	x	x	x
		Steiner [70]		x		x	x	x
	BDMP [123]		x	x	x	x	x	x
	BBN [84]			x	x	x	x	x
	Misuse cases [87]			x	x	x	x	
	CHASSIS [48]			x	x	x	x	
	UMLsec/UMLsafe [93]			x	x		x	
	SysML-Sec [95]			x	x	x	x	x
	Stochastic Petri nets [67][68][70]		x	x	x	x	x	x
	MBSE [97]			x	x	x	x	
	Formal methods	Zafar[102] GSE method		x	x		x	
		AADL [101]		x	x	x	x	
	Approaches for electrical networks [92][93][95]			x	x	x	x	x
	Non-graphical Methods							
	Informal	Reichenbach [31]		x		x	x	x
		Holstein [110]		x		x	x	x
		Depoy [98]	x			x	x	x
		Pieters [112]		x	x	x	x	x
	Formal	Sun [37]		x	x	x	x	
		Simpson [114]		x	x		x	
	STPA-Sec [119]			x	x	x	x	

Table 2: Classification of the identified approaches

The system-theoretic approach STPA-sec introduces a new way of performing risk analysis based on control actions which is useful in the early phases of the system lifecycle. This purely qualitative approach exhaustively identifies the causal factors in the hazard analysis related to the control structure of the system and the interactions among components. The results generated are still very macroscopic which is not very suitable to identify "the devil in the details" for safety and security interactions.

Model-based system engineering approaches are suitable for addressing increasing system complexity and enable systems to be modeled from different viewpoints. When they are supported by formal languages and analysis tools, they provide a solid framework for eliciting and verifying safety and security requirements. They are probably the most promising approaches: as a consequence, large research efforts are currently invested in their development (e.g. SESAMO and MERGE [110] projects).

2.4.3 Discussion

In the industrial context of the thesis, we are interested in operational systems that have a long service life like power plants, refineries, dams, etc. As discussed in the previous section and inferred from the state of the art, we believe that model-based approaches are more practical to model the systems of our context. We consequently exclude process-oriented approaches from the scope of our exploration and consider only model-based approaches.

The ultimate goal of the thesis is to have a risk analysis approach that encompasses both safety and security risks in order to identify their potential interdependencies into digital control systems. In order to obtain this goal, we believe that the desired approach should satisfy the following criteria:

- C1: safety-security modeling
  The desired approach enables to model both safety and security aspects;
- C2: Formal
  The desired approach is based on mathematical concepts and reasoning methods;
- C3: Qualitative and quantitative
  The desired approach yields both qualitative and quantitative results; indeed such an approach is more likely to capture safety and security interdependencies.

All model-based approaches identified in Table 2 satisfy the first criteria (C1). We evaluate in Table 3 these approaches according to criteria C2 and C3. Non-formal methods are excluded as they do not satisfy C2.

Approach \ **Criteria**		**C2**	**C3**
GSN			
Extended fault trees		x	x
BDMP		x	x
BBN		x	x
UML-based approaches	Misuse cases		
	CHASSIS		
	SysML-sec		x
SPN		x	x
MBSE		x	
Formal verification methods (model-checking, theorem prover)		x	
AADL		x	
STPA-sec			

Table 3: Classification of approaches according to C2 and C3

As shown in Table 3, extended fault trees, Boolean logic Driven Markov Processes, Bayesian belief networks and Stochastic Petri nets based approaches satisfy both criteria. As previously discussed in Section 2.4.2, fault trees approaches are limited by their static aspect that inhibits modeling the dynamics of attacks and failures dependencies. Bayesian Belief Networks and Stochastic Petri Nets have good qualitative and quantitative capabilities however both approaches do not graphically scale up to complex systems. A non-trivial sized system would involve a graphical model which can be easily unreadable.

For these reasons, we have chosen to investigate BDMP as they enable dynamic behavior modeling and provide good readability of models. As the use of BDMP in the security domain has been so far limited to simple and scholar case studies, we propose in the next chapter to use BDMP to model sophisticated attacks. We also illustrate its potential to model jointly safety and security risks and to capture their interdependencies on a realistic case study. We finally outline the limitations encountered with the application of this approach.

Chapter 3

Modeling safety and security with Boolean logic Driven Markov Processes

In this chapter, we first present the previous work on safety and security modeling using the Boolean logic Driven Markov Processes (BDMP) approach. We next investigate its potential to model complex attacks, through modeling Stuxnet. Then, we illustrate BDMP on a realistic case study in order to identify safety and security interdependencies. We finally compare it to the CHASSIS approach.

3.1 Previous work

This section gives an overview on the previous work that consists in using the BDMP formalism for safety modeling, its adaption to security modeling and the introduction of its ability to joint safety and security modeling and to their interactions identification.

3.1.1 Modeling safety with BDMP

Boolean logic Driven Markov Processes (BDMP) are a graphical modeling formalism initially created by Bouissou [72] in 2003 for system safety and reliability analyses.

Visually similar to fault trees, BDMP model the different combinations of events (leaves of the tree) that lead to the top event (associated to the undesirable event, e.g. system failure/damage). BDMP add a new kind of links called “triggers” (represented by red dashed arrows) to fault trees; they enable dynamic behavior modeling. In addition to fault trees, BDMP involve Markov processes. They have interesting mathematical properties that enable to process the model and provide quantitative results. BDMP allow also a dramatic reduction of combinatorial problems related to Markov processes with very large state spaces [72].

We give in the following a quick overview on the BDMP basic properties and the objects used for graphical modeling.

1) The elements of a BDMP:

A BDMP *{A, r, T, P}* is made of: a multi-top coherent fault tree A, a main top event *r* of *A*, a set *T* of triggers and a set *P* of “triggered Markov processes" *Pi* associated to the leaves of *A*. The Markov process *Pi* is said to be “triggered" because it switches instantaneously from one of its modes to the other one according to the state of some externally defined Boolean variable, called “process selector". An important feature of BDMP is the concept of “relevant event". An event is said to be non-relevant if the propagation of its realization effects within the tree only affects the gates already realized. For example if one leaf of an “OR" gate is realized, other leaves under this gate are no longer relevant because the gate is realized. Non relevant failures are trimmed during the processing when exploring the possible sequences. Trimming strongly reduces the combinatorial explosion while yielding accurate results in our assumptions [72]. The process selectors are defined by means of triggers. A trigger, graphically represented with a red dashed arrow, can modify the mode of the processes associated to the leaves of the sub-tree it points at, when the event that is the origin of the trigger changes from FALSE to TRUE (or conversely). The complete definition of the semantics of a BDMP can be found in [72].

2) Modeling objects:

- BDMP leaves: model the basic events corresponding to the system’s components failures that can lead to the undesirable event. The basic BDMP leaves used for safety modeling are given in Table 4;
- Gates and links: the BDMP models use classical logic gates “AND”, “OR” and “k out of n”; and more specific gates (e.g. “PAND” “Aggregate OR”) defined in [123]. In Addition to classical logic links used to connect a gate to its sons (represented as solid black lines), BDMP models contain other specific links described in Table 5.

Each basic event of a BDMP is associated to a Markov Process with two possible modes, corresponding to the fact that the components or subsystems that they model are required or are in standby mode. The mode chosen for a given leaf at a given instant depends on the realization of other leaves, which is modeled with triggers.

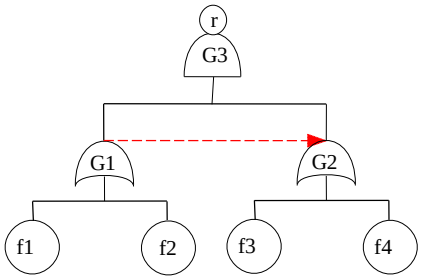


Figure 12: A basic BDMP model

An example of a BDMP is depicted in Figure 12. The top event “r” above the AND gate “G3” is realized if both G1 and G2 are TRUE. The OR gate G1 is TRUE if at least “f1” or “f2” is realized. The OR gate G2 is TRUE if at least “f3” or “f4” is realized. The trigger of this BDMP specifies that whenever G1 is TRUE f3 and f4 are in the required mode. Otherwise they are in the not required mode. This models a standby redundancy.

Representation	Modeled behavior
 “ Failure in operation ”	This leaf is used to model a failure in operation, when the modeled component is active. Failure occurs after a time exponentially distributed (parameter λ) and can also be repaired in a time exponentially distributed (parameter μ).
 “ Instantaneous failure ”	This leaf is used to model a failure on demand likely to arise instantaneously when the leaf changes mode (from not required to required), with a probability γ. Failure can be repaired in a time exponentially distributed (parameter μ).

Table 4: Basic BDMP leaves for safety modeling

Representation	Modeled behavior
“ Trigger link ”	Defines the dynamic aspect of BDMP. The element pointed by the trigger link is not activated until the realization of the origin gate/leaf of the trigger. When this element becomes activated, it transmits the activation signal it receives from its parents to the sub-tree targeted by the trigger.
“ Before link ”	Creates a constraint in the order of realization of instantaneous events (on-demand failure leaves), in the case where they are required simultaneously.

Table 5: Special links used in BDMP models

3.1.2 Modeling security with BDMP

The BDMP formalism was used by Pietre-Cambacedes [73] to model security-related risks. New security leaves were introduced to model attack steps, or in some cases, security events that are not under the direct control of the attacker (e.g., the opening of an email containing a malicious payload by a victim of the attacker). These attack leaves define the different types of events that we can consider in an attack scenario, they are depicted in Table 6. Each attack leaf can be either in “Idle” or “Active” mode. The former is used when nothing is in progress; the latter models an on-going event, generally an attack event in progress.

With these security leaves, BDMPs enable graphical modeling of the different combinations of attack steps that lead to an undesirable event. Detection and response mechanisms against attacks can also be taken into account [124] thanks to the generalization of the concept of mode, allowing three modes instead of just two. The general idea is that each attack step can be detected at various moments: when it begins, during its progress, when it succeeds, or after completion. Whenever detection occurs, this changes all success rates or probabilities for attack steps which are still to be completed. The only thing the analyst has to do to take detection into account in the BDMP model is to change a global option in the model and add in each security leaf the detection rate and the realization rate after detection. This does not require any change in the BDMP structure which enables to represent a complex model in a concise manner. These detection parameters are taken into account in the quantitative processing: once the attack is detected, it makes the following actions of the attacker more difficult or even impossible (cf. example in Section 3.2.2).

Representation	Modeled behavior
 “ AA leaf ”	The "Attacker Action" (AA) leaf models an attacker step towards the accomplishment of his objective. The Idle mode means that the attacker has not yet at this stage tried to do this action. The Active mode corresponds to actual attempts for which the time needed to succeed is exponentially distributed with a parameter λ. The Mean Time To Success (MTTS) for this action is equal to 1/λ.
 “ TSE leaf ”	The "Timed Security Event" (TSE) leaf models an event the realization of which is necessary for the attack success but that is not under the direct control of the attacker. The time needed for its realization is exponentially distributed (MTTS=1/λ). If the leaf comes *back* to the Idle mode, after being active for some time, the leaf state can still become realized which would not be the case with an AA leaf).
 “ ISE leaf ”	The "Instantaneous Security Event" (ISE) leaf models a security event that can happen instantaneously with a probability γ when the leaf switches from the Idle mode to the Active mode. In the Idle mode, the event cannot occur and the leaf stays in the state Potential. In the Active mode, the event is either Realized or Not Realized.

Table 6: Basic BDMP leaves for security modeling

3.1.3 Modeling safety and security with BDMP

Pitere-Cambacedes and Bouissou [32] used the BDMP formalism to create models integrating both safety-related and security-related risks that lead to the same undesirable event. The authors distinguished between “pure models” where purely accidental or purely malicious events are modeled, “hybrid models” where a combination of accidental and malicious events is modeled, and finally “integrated models” which combine pure and integrated models. They additionally underline the potential of such models to catch safety and security interdependencies (cf. Section 1.3.3).

As presented and illustrated on the pedagogical use case given in [32], using BDMP as a unifying formalism for joint safety and security risks modeling and interdependencies remains theoretical and not enough proven. We have consequently chosen to explore in Section 3.2 this approach in depth and illustrate it on real and complex systems in order to identify potential safety and security interdependencies. We also compare in Section 3.3 the BDMP approach to the CHASSIS method (cf. § 2.3.1.5) and show how they complement each other.

We give in the next subsection an overview on the KB3 workbench used for inputting BDMP models and processing them. The information below is necessary to understand the examples given in Sections 3.2 and 3.3.

3.1.4 The KB3 workbench

The KB3 workbench includes the KB3 software used to input graphical models, thanks to a knowledge base, and the quantification tools used to process them.

3.1.4.1 KB3 and Knowledge Bases

KB3 v3 (hereafter named KB3) is a software, developed and used at EDF since 2000, that enables to build probabilistic models for dependability studies. These models can describe structural type (e.g., fault trees) or behavioral type (e.g., dynamic Markovian and non-Markovian models, Monte Carlo simulation models) for the studied system. These models are built from elements described in a knowledge base (KB)[125]. Knowledge bases aim at capitalizing expert’s knowledge about a given system or domain (cf. Section 4.1 for examples), in order to reuse it for building models. They can be abstract (e.g., the KB used to create BDMP) or dedicated to a particular domain (i.e. comprised of

physical objects like pumps, valves, circuit-breakers, etc.). They are written using the Figaro modeling language that will be detailed in Section 4.6.1

Multiple knowledge bases have been developed using the KB3 workbench among which the knowledge bases used to build the BDMP models:

- For “Safety BDMP”, the accidental leaves and the corresponding modeled behavior described in Table 4 are depicted in the “ Dependability KB” [72];
- For “Security BDMP”, the attack leaves and the corresponding modeled behavior described in Table 6 are depicted in the “Security KB” [123];
- For “Safety-Security BDMP”, the “Security-Dependability KB” [126] merges the two previous knowledge bases in order to enable building BDMP models combining both safety and security risk events. Such models will be illustrated in Sections 3.2 & 3.3.

The KB3 tool enables to associate graphical elements with the different components defined in a given knowledge base. These elements are used to build the graphical model (e.g., BDMP model) using the KB3 Human Machine Interface (HMI). KB3 next transforms the model into a Figaro language description, which can be processed thanks to the quantification tools described in the following section.

3.1.4.2 Quantification Tools: Figseq and Yams

Two quantification tools are used in the KB3 workbench (for dynamic models), Figseq and Yams, and enable to process the models input by the user (e.g., BDMP model). We give below a quick overview on each tool. The principle of each tool will be further explained in Section 4.6.2. Both Figseq and Yams use as an input the Figaro model generated by KB3 from the graphical description input by the user in KB3 and the corresponding KB.

Figseq (Figaro Sequences Generator): is relevant in case only exponential distributions are used to model the temporal behavior of the modeled elements. It uses path-based algorithms that explore and quantify the sequences going from the initial state of the system to a failure state. These algorithms are able to deal with large-state space Markov processes while avoiding combinatory explosion [127].

Two different algorithms can be used by the Figseq tool: the “NS” algorithm (Normal Sequences) used for non-repairable systems or in case the mission time is equivalent to the time spent in the first state, and the “NRI” algorithm (No Return to Initial state) used for systems completely repairable (assuming that mean time to repair is very low compared to mean time to failure).). Both algorithms yield a reliability estimation and the list of the most probable sequences leading to an undesirable state. In addition, the NRI algorithm produces an estimation of the asymptotic unavailability of the system.

Yams (Yet Another Monte Carlo Simulator): is used in case Non-exponential distributions are used to model the temporal behavior of the modeled elements. It simulates, based on the Monte Carlo method, a large number of realizations of the random process specified by the model, and then calculates estimators of the variables of interest by statistics. Yams can estimate not only reliability and availability of the system, but also the mean and standard deviation of any random variable linked to the system model, for example its exploitation cost.

We propose in the next section to investigate the ability of Boolean logic Driven Markov Processes firstly to model sophisticated attacks and secondly to model safety and security risks on real system architectures in order to identify their possible interdependencies.

3.2 Modeling real attacks and complex systems

BDMP have not so far been used to model real and sophisticated attacks or to address jointly safety and security issues in complex systems. We propose to deal in this section with these two aspects. We first modeled the famous Stuxnet attack with BDMP [128]. Next, we used the approach proposed in [32] in order to prove its ability to catch safety and security interdependencies on a realistic industrial case study taking into account the system architecture; and gave qualitative and quantitative results obtained from it.

3.2.1 Modeling the Stuxnet attack with BDMP

Since 2010, the Stuxnet worm has been of particular interest for the media and security experts. Not only because of its high degree of sophistication but also because it targeted the control systems of an industrial installation and led, for the first time, to major physical damage. It has since been a trigger to urge industries to protect their critical infrastructures against cyber-attacks and to pay more attention to interdependencies between the cyber and the physical parts of their systems. Many studies explored Stuxnet with more or less details and gave technical explanations about the infiltration and the propagation of this worm into the core network and the control system [1][129]. Yet, very few provided a model enabling a global understanding of the attack. Modeling an attack is a paramount step in the procedure of securing a cyber-physical system for several reasons [130]. First, it enables the identification of the weaknesses and the different access points of the system and makes the attack vectors more evident. Secondly, it makes the search for efficient solutions to mitigate these vulnerabilities easier. Finally, it helps understanding the behavior of the attacker and assessing the effect on the physical infrastructure. The only existing models of the Stuxnet attack are based on attack trees [131] or graphs [132]. We propose in this section to model Stuxnet with BDMP applied to security (cf. Section 3.1.2) in order to: i) better reflect the dynamics of this assault, ii) enable a coarse quantification of the attack success probability and finally, iii) highlight the advantages of BDMP compared to existing models.

The remainder of this section will be organized as follows: first we give a global overview of the Stuxnet attack. Second, we describe the architecture of the industrial site targeted by this attack. Third, we detail the dynamics of the Stuxnet attack. Fourth, we give the BDMP modeling these dynamics and fifth the qualitative and quantitative risk analysis associated with this model. We finally outline the advantages our model compared to existing ones.

3.2.1.1 Stuxnet attack overview

Stuxnet ultimately targeted SCADA systems running a Windows environment that hosts specific Siemens industrial control systems (namely the WinCC, PCS7 and STEP7 platforms) and connected to specific types of Programmable Logic Controller devices (PLCs). It reprograms PLCs in a way that modifies the system operation leading to damage to the physical infrastructure under control. The Stuxnet attack affected mainly Iranian nuclear enrichment facilities and resulted in slowing down the production of centrifugal machines and finally damaging them. The sophistication of the malware and the very specific systems it targeted led to the conclusion that such attack could not be developed by a group of persons but rather by a nation-state.

Considering the sensitivity of the facility targeted by Stuxnet, its SCADA system was not directly connected to the Internet (and presumably non-industrial networks of the facility). Consequently, the best attack path for Stuxnet was to compromise an external device, typically a USB thumb drive, which would be later connected to the control system. So, the first step of the attack was to propagate throughout the Information System of the Enterprise corporate network to increase the probability of reaching the industrial network. To reach this goal, Stuxnet exploits several Windows' vulnerabilities and at least four 0-day exploits [129][132]. Another specificity of this malware is that it injects its entire payload into other legitimate processes and uses several rootkits to escape detection. We give more technical details about Stuxnet dynamics and its life cycle in § 3.2.1.3.

3.2.1.2 Network architecture of the industrial site

Several security consulting services published detailed information about the components of the targeted Siemens platforms and typical network architectures [132]. Based on these studies, we have defined a simplified architecture of what could have been the targeted one. It is represented in Figure 13 and will be the basis of our BDMP model of the Stuxnet attack in § 3.2.1.4.

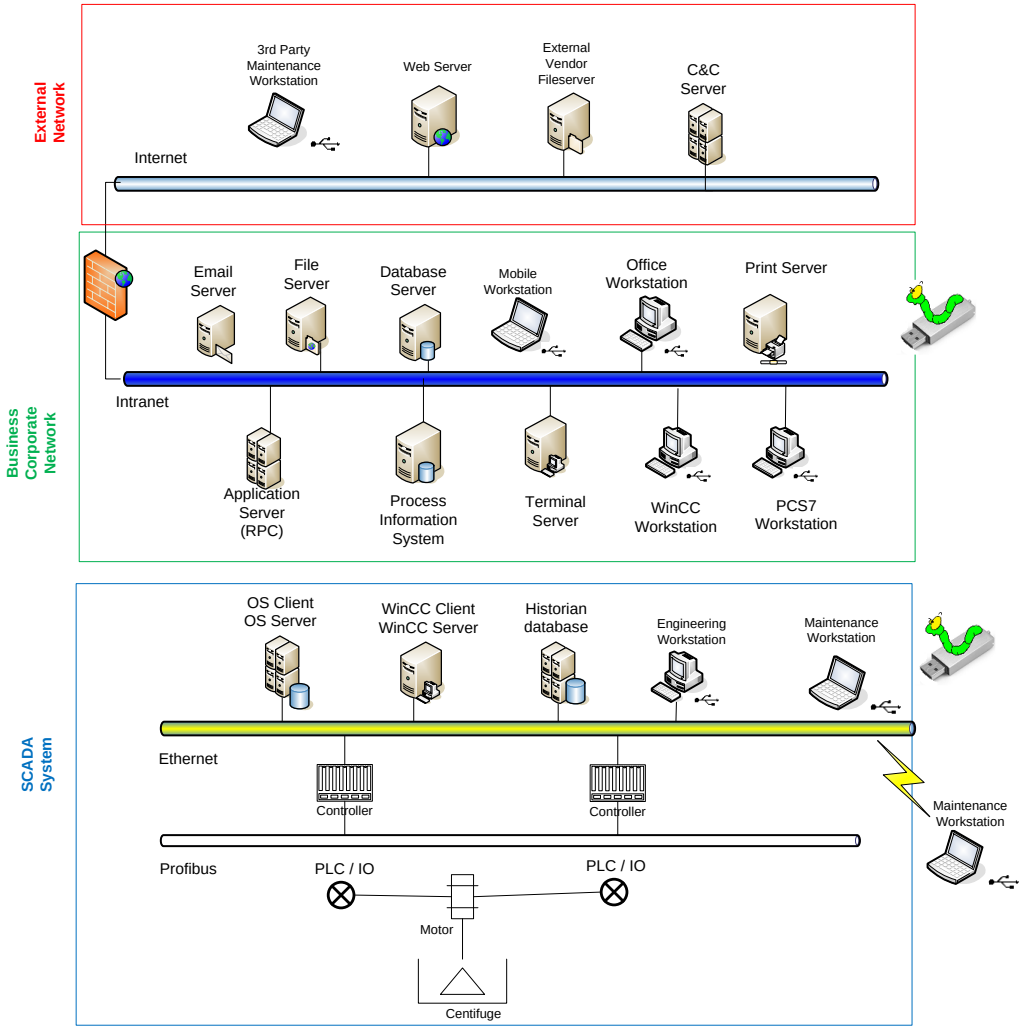


Figure 13: Facility network architecture of an industrial network

The whole facility architecture is composed of two main security zones, the Business Corporate Network and the SCADA system.

The Business Corporate Network hosts the Enterprise usual Information System. It comprises servers and workstations that enable classical daily applications (emails, reporting, accountability...), the Enterprise Resource Planning (ERP) system, etc. It may also host WinCC SQL Server databases that provide high level information to end users and store STEP7 project files, as well as applications that manage PCS7 or WinCC projects. Data can be exchanged between terminals via a local area network that hosts local databases and process information servers. This network can exchange data with external networks connected to the Internet through a “demilitarized zone". The communication is protected by firewalls and other security modules.

The SCADA system includes a Process Control Network and a Control System Network. The Process Control Network consists of WinCC and PCS7 clients and servers which are connected to PLCs and enable communication with them. WinCC machines provide HMI client/server systems used to monitor

the industrial process and visualize messages and real-time data. PCS7 machines include basic data collection functions for project data, process values, archives, alarms and messages. PCS7 servers provide all process data and connect PLCs to the Process Control Network [132]. The control system network includes WinCC/PCS7 servers and PLCs. It controls and supervises the physical process. PCS7/WinCC server and client can be installed on the same hardware which is the case in our model for simplification. PLCs send control signals via a Process Field Bus (Profibus) to speed regulators that control the rotation of motors. This network includes as well WinCC SQL Server databases and other engineering or maintenance workstations. We suppose that, for security reasons, the SCADA system is isolated by an air gap so that no network connection is possible between the two security zones.

3.2.1.3 Stuxnet dynamics

In this sub-section we give more details about the malware main phases and attack steps. Text written in `Typewriter` font denotes the corresponding security leaves (cf. Table 6) in the BDMP model given in Figure 14 and Appendix 1 (not all leaves are mentioned). Basically, we can distinguish two main phases: 1) infiltration and propagation into the corporate Enterprise network; 2) compromising SCADA systems and industrial sabotage.

1) Infiltration and propagation into the corporate Enterprise network:

We assume in our model that the whole network is initially non-infected. It is then very likely that the very first infiltration of the malware was introduced by infected removable media, which represent the main attack vector, into a workstation of the business corporate network.

The user action of inserting an infected removable drive is modeled by a TSE leaf `user USB` key execution as it is dependent on the user and not under the control of the attacker. Stuxnet exploits two Windows vulnerabilities to spread to and from removable drives. One is Windows Shell LNK vulnerability linked to the system handling of shortcuts using ".LNK" and ".PIF" files. The other is autorun.inf file vulnerability which enables self-execution of the removable drive when inserted. The user has just to open a compromised file folder on his USB drive to let the worm do the rest of the work. When the first step is realized, the malware instantaneously exploits one of the two vulnerabilities modeled by ISE leaves `Win LNK vuln` and `autorun.inf vuln` in order to infiltrate the system.

Once introduced into the system, the next attack step is self-installation. The malware loads instantaneously the main dropper which is a dynamic link library (.dll) that contains Stuxnet functions, files and rootkits into a trusted process generally default Windows processes or executable files of security products installed. Then it, `checks Windows config`: it targets particularly 32 bits machines running the operating systems Windows XP/2003/Vista/7 or Windows Server. Next, it `checks admin rights` of the current user. If not found, the malware exploits one of two zero-days, `keyboard layout` and `task planner` vulnerabilities, to elevate its privileges. The worm proceeds then to the execution of its main installer. It comprises two main steps: `install Win rootkit` in order to escape detection then updating the last version of the malware. To install the rootkit, Stuxnet loads a driver file legitimately signed by Realtek certificate and used to scan the main filesystem driver objects. It then creates a new device object and attaches it to the driver chain of the previous driver objects to be the first to receive requests to/from these drivers. This allows the malware to filter out files with ".LNK" and ".TMP" extensions to hide malicious files. To update the last version of the malware, Stuxnet can either establish a `P2P communication` by installing an RPC server on the infected machine and wait for connections from RPC clients or it can directly download the latest updates from Control and Command Server (`C&C server communication`). Stuxnet communicates with remote servers on port 80 via HTTP. It injects itself into Internet explorer or creates another browser process to bypass firewall security rules.

The malware tries to infect as many workstations as possible in the corporate business network to maximize its chances to transit later to the Control Network. The three main ways of infection, self-replication and propagation are: 1) `removable media` connected to compromised machines. 2) the `LAN`; through `network shares`, Print Spooler Service exploit, Windows Server Service exploit or connections to WinCC remote databases. In this last case, Stuxnet searches for WinCC environments. If found, it connects to the database using default Siemens password and sends malicious code via

SQL queries. 3) WinCC/Step7 project files associated to WinCC SIMATIC Manager. Stuxnet searches for and infects files with extensions ".S7P", ".MCP" or ".TMP". It waits then for the user to open the infected file (`user opens file project`) to load its .dll file, decrypt data and execute infection routines.

2) Compromising SCADA systems and industrial sabotage:

The next main step for Stuxnet is to reach and compromise the SCADA Network to attack the industrial system. The network being isolated from the Corporate Business Network for security reasons, the malware waits until it is somehow carried to the Process Control Network by an employee connecting infected removable drives or by a maintenance workstation (`infection of a control PC`). Stuxnet looks first for WinCC/Step7 software on the control PC used to configure the PLC. If found, it installs a rootkit: it loads a library file (s7otbxdx.dll) used for the communication between the control PC and the PLC, renames it (s7otbx**s**x.dll) and inserts malicious code into the new file. After checking connection to PLC as well as other specific configuration (PLC model, Profibus configuration, and speed regulators number), the malware proceeds to infecting and modifying PLC function blocks.

The code executed on PLC differs depending on its CPU type; only 6ES7-315-2 or 6ES7-417 modules are targeted. `Flag sys 300` and `Flag sys 400` enable to choose one of these two options. In the case of 300-series systems, the malware collects data for a period going from 13 days to 3 months before sending falsified data to motors on the communication bus for around 50 minutes. For 400-series systems, the code sequence is more complex. Coarsely, it intercepts input and output signals of PLC and provides false data to the logic code sequence in order to falsify output returned signals (man-in-the-middle attack). The malware operates without being detected or inducing any suspicious signals or abnormal values to be visualized in return to operators.

Through the falsified PLC output signals, the attacker gives instructions to motors to alternate high then low frequency rotation. This phase can last several months causing the physical materiel to wear down slowly and consequently worsen its performance. It can even end into machines self-destruction.

3.2.1.4 Stuxnet modeling with BDMP

We model in Figure 14 the top part of the Stuxnet attack BDMP with its main phases: infiltration, self-installation and attack of the industrial system. The BDMP of the last two phases are detailed in Annex 1 (Figure 41 and Figure 42). The sub-tree modeling the different propagation paths of the malware (Figure 41) is not required later in risk quantification because the attack can succeed from the first chance; that is why it is not linked to the top event of the BDMP.

In this BDMP model, we can parameterize the different leaves following their formal specification by estimating success rates and probabilities (λs and γs). Such parameterization is used later in the quantitative analysis: it enables the computation of the attack success probability as well all possible sequences that lead to the attack success. We list in Table 7 parameter values corresponding to attack leaves that we have chosen in the BDMP model based on our own estimation and writings by security consultants [133].

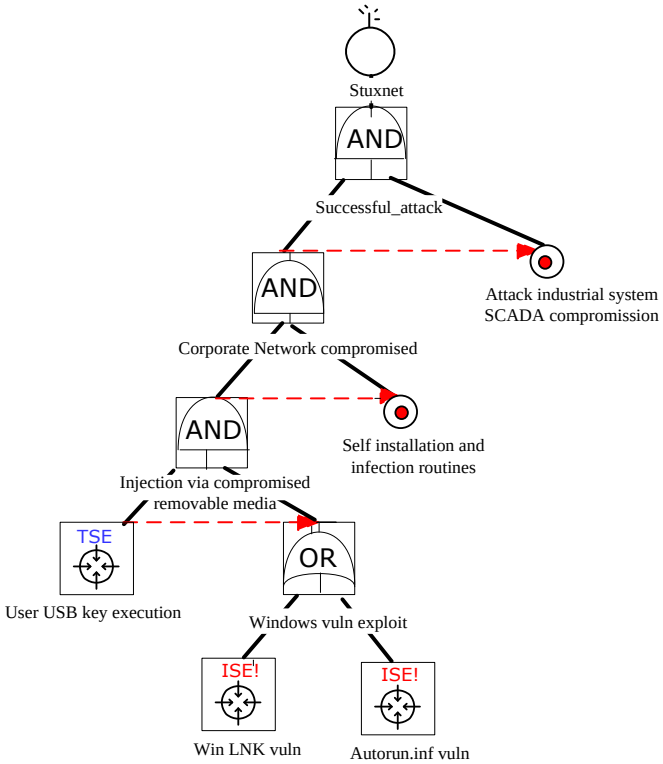


Figure 14: Top part of the Stuxnet model

Subtree	**Leaf label**	**Parameter**[8]
Infiltration	user USB key exec	λ=5.787e-6 (MTTS= 2 days)
	Win LNK vuln autorun.inf vuln	γ=1/2
Installation	Admin rights	γ=0.7
	keyboard layout vuln task planner vuln P2P communication CC server communication	γ=1/2
SCADA compromising	infection of a control PC	λ=7.7e-7 (MTTS= 15 days)
	collect data	λ=3.86e-7 (MTTS= 1 month)
	user opens file project	λ=1.16e-5 (MTTS= 1 day)
	PLC sends false data to motors	λ=3.33e-4 (MTTS= 50 min)
	intercept in out PLC signals	λ=8.36e-7 (MTTS= 1 month)
	modify out signals	λ=1.15e-5 (MTTS= 1 day)
Propagation	removable media	λ=5.79e-6 (MTTS= 2 days)
	network shares	λ=1.39e-4 (MTTS= 2 hours)

[8] Unit for λ is per second

	print servers vuln	λ=9.25e-5 (MTTS= 3 hours)
	service server RPC vuln	λ=2.77e-4 (MTTS= 1 hour)
	cascade centrifuges	γ=0.1 per centrifuge
For all other ISE leaves: γ=0.99 (almost sure)		

Table 7: Parameters of the Stuxnet BDMP model

3.2.1.5 Quantitative and qualitative risk analysis

BDMP enable not only attack representation but yield also quantitative and qualitative results directly usable for risk assessment. KB3 quantification tools (cf. Section 3.1.4) enable BDMP analysis namely the enumeration of all possible attack paths ordered by their probabilities of occurrence and contributions to the final attack success. We summarize in Table 8 all possible attack sequences; vertically readable.

We number steps that are not in common for all sequences and we denote in Table 9 each sequence by the numbers of the steps that differentiate it from other sequences.

User USB key execution			
1-Win LNK vuln autorun.inf vuln (NR)	2-Win LNK vuln(NR) autorun.inf vuln	3-Win LNK vuln autorun.inf vuln	
self injection into process			
check Windows config			
4-Admin rights	Admin rights (NR)		
	5-keyboard layout vuln task planner vuln(NR)	6-keyboard layout vuln(NR) task planner vuln	7-keyboard layout vuln task planner vuln
load driver legitimately signed			
scanning filesystem drivers			
new device object attachmen			
filter out .lnk .tmp files			
8-C&C server communication		9-P2P communication	
infection of a control PC			
check STEP7 or WinCC			
Load library			
rename replace library			
check PLC exists			
Check PLC model			
check Profibus config			
check speed regulators number			
modify PLC function blocks			
rootkit400 activated			
intercept in out PLC signals			
modify out signals			
fail cascade centrifuges			

Table 8: List of successful attack sequences (NR: Not Realized)

According to [133], Stuxnet attacked a group of cascades of centrifuges, each cascade comprises 164 centrifuges. We suppose, in our BDMP model, that the attack is successful when at least 3 centrifugal machines fail. This hypothesis is not important (at least from a qualitative point of view): when Stuxnet infects the PLC, it can make all the machines fail. It is just a matter of time. The succession of failures of a set of identical components is represented by the leaf at the extreme right of the BDMP "cascade centrifuges". The behavior of such a leaf is depicted in FIGURE 15. Each state is defined by the number of failed components.

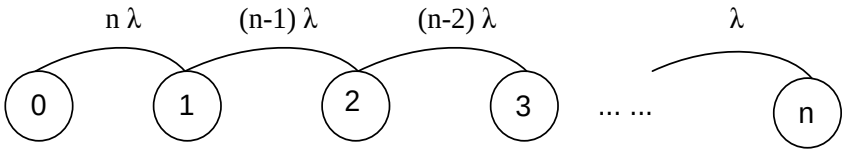


Figure 15: Markov chain modeling the failure process of a set of n identical components (n=164 in the case study)

Using KB3 quantification tools (cf. Section 3.1.4), we compute the probability of each possible sequence and its contribution to the overall probability of the attack success which is estimated to 0.6 for a 400-series system, with the chosen parameters. The results are given in Table 9. We can for example infer from these quantitative results that it is more probable for the malware to update itself through communicating with the Control and Command server that enables the attacker acting remotely on the infected system rather than waiting for RPC clients to connect for Peer-to-Peer communication. We can also notice, but it seems more obvious, that the attack is more likely to succeed when administrator rights are available on the system.

Sequences	1-4-8 2-4-8 3-4-8	1-4-9 2-4-8 3-4-9	1-5-8 2-5-8 3-5-8	1-6-8 2-6-8 3-6-8	1-7-8 2-7-8 3-7-8	1-5-9 2-5-9 3-5-9	1-6-9 2-6-9 3-6-9	1-7-9 2-7-9 3-7-9
Proba per seq	1.06e-1	4.54e-2	1.14e-2			4.86e-3		
Contrib per seq	17.65%	7.56%	1.89%			0.8%		
Sum of contrib	52.95%	22.68%	17.01%			7.2%		

Table 9: Quantification results

Figure 16 plots the evolution of attack success probability according to time. We can see that success probability increases by time and reaches an asymptote after around 6 months. This asymptote is the global attack success probability; equal to 0.6 in the case of a 400-serie PLC. This probability never reaches 1 because we consider all attack failure cases including targeted configuration not found and PLCs not connected to infected configuration machines. The evolution of success probability is also tightly linked to the realization of long-phased attack steps mainly at Attack industrial system phase.

“Stuxnet was discovered in July 2010, but is confirmed to have existed at least one year prior and likely even before"[1]. The value of 6 months that we obtain by our quantification has the same order of magnitude but is clearly inferior for the following reasons: (i) we chose rather high values for the model’s parameters (ii) the malware was discovered after the intended effects had taken place.

3.2.1.6 Comparison with existing Stuxnet models

An attack tree [68] modeling Stuxnet with our hypotheses would be very close to the BDMP without its triggers and precedence (before) links. The attack tree model found in [131] is a good illustration. It gives no indication about the constraints on the order of attack steps. Therefore, it is useless without a detailed textual explanation. On the contrary, once a reader has in mind the correspondence between the necessarily short names of the leaves and the real events, he has a complete description of the attack with the BDMP. This modeling power is what makes the risk quantification of § 3.2.1.5 possible. With an attack tree, the only possible quantification corresponds to the assumption that all timed transitions are enabled from the start of the attack. This gives grossly erroneous results.
This drawback of attack trees is particularly obvious in the case of Stuxnet where many attack steps are highly dependent from one another. The attack graph found in [132] is only a visual representation with no quantification capabilities. Moreover, its non-hierarchical and cyclic structure makes it less readable than a tree structure. It focuses on the propagation aspects of the worm, covering only a subpart of our model.

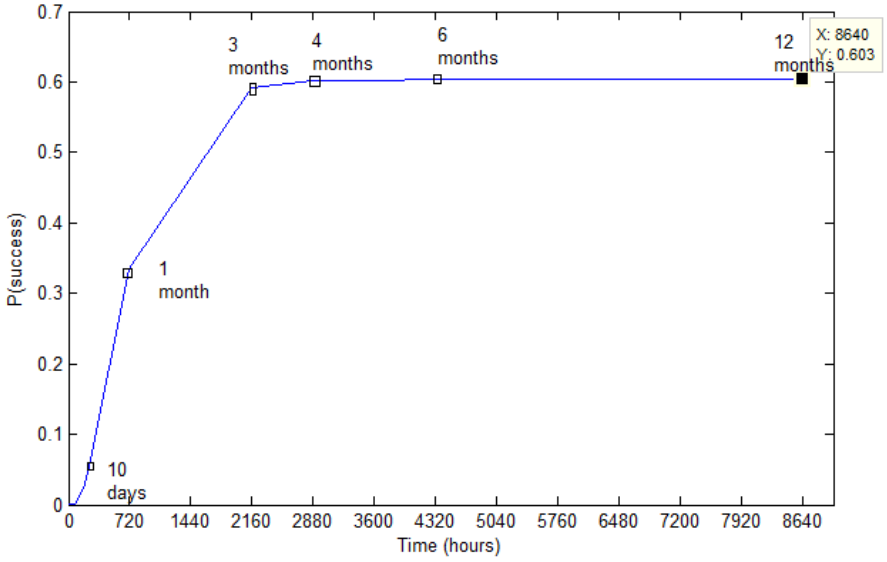


Figure 16: Attack success probability according to time

3.2.1.7 Conclusion

We modeled, in this section, the fundamental mechanisms of the Stuxnet attack in a unique and rigorous graphical representation, and gave quantification results for each possible attack sequence. These results reflect the potential of BDMP for modeling the steps of an attack and its global progress. Quantification tools can process such a model to enumerate possible attack sequences with their probabilities and contributions to the overall attack success. This model also identifies attack vectors and access points that an attacker may exploit in order to infiltrate a system and take control over the main control functions to be represented.

The Stuxnet attack study proves that industrial infrastructures are vulnerable to cyber-attacks. Despite the partitioning (air gap) between the Business Corporate Network and the SCADA system, the malware managed to reach the industrial control and modify the process operation leading to safety issues. We propose in the next section to point out more closely safety and security interdependencies on a realistic case study through BDMP modeling.

3.2.2 Modeling safety and security interdependencies

We propose in this section to model jointly safety and security risks with BDMP in order to identify their possible interdependencies. We have studied in [126] a simple example where safety and security requirements are contradictory. Using BDMP modeling, we have been able to characterize quantitatively the conflict between safety and security measures, and decide what measures should be privileged in order to minimize the global risk.

We consider in the remainder of this section, a detailed case study inspired from the industrial domain. After describing the system architecture, we give a common BDMP model for safety and security risks. We next generate the associated results and demonstrate synergy between safety and security measures.

3.2.2.1 System architecture description

The system considered is a cyber-physical system used for transporting a polluting substance. The architecture of this case study, given in Figure 17, is hypothetical but can be transposed to other industrial systems.

It is composed of a pipeline equipped with pumps used to force the stream and valves used to allow or block the stream. Throughout the pipeline sensors measure the pressure and flow inside each section of

the pipeline. Each equipment (pump or valve) is controlled by a Remote Telemetry Unit (RTU) that communicates with a remote Control Center (CC). RTU tasks are to:

- Collect data from sensors used to measure the pressure and the flow at the neighboring of each pump and valve;
- Control pumps' operation and speed and valves' opening/closing;
- Send data and alarm signals to the CC and receive instructions from it;
- Exchange with neighboring RTUs pressure measures and shutdown signals.

Safety requires RTUs to verify the pressure in the pipeline does not exceed a maximum value P_{max}. Each RTU also calculates the difference of pressure between the one it gets from its own sensors and the one it receives from the neighboring RTU: $\Delta P = |P_n - P_{n-1}|$. If ΔP exceeds a threshold ΔP_{max}, the RTU sends an alarm signal to the CC, which sends back an order to all RTUs to stop pumps and close valves. The pressure threshold is reached generally when the pipeline is broken; this implies that the pressure measured before the break is very high compared to the pressure measured after the break, which makes the difference of pressure very high. A safety requirement enables each RTU to stop the pump or close the valve it controls when ΔP_{max} is reached or when it receives a shutdown order from other RTU without waiting for CC instructions. In addition the RTU sends a shutdown signal to its neighboring RTUs. This action is called later "Reflex Action" and provides redundancy with CC instructions, with a higher priority.

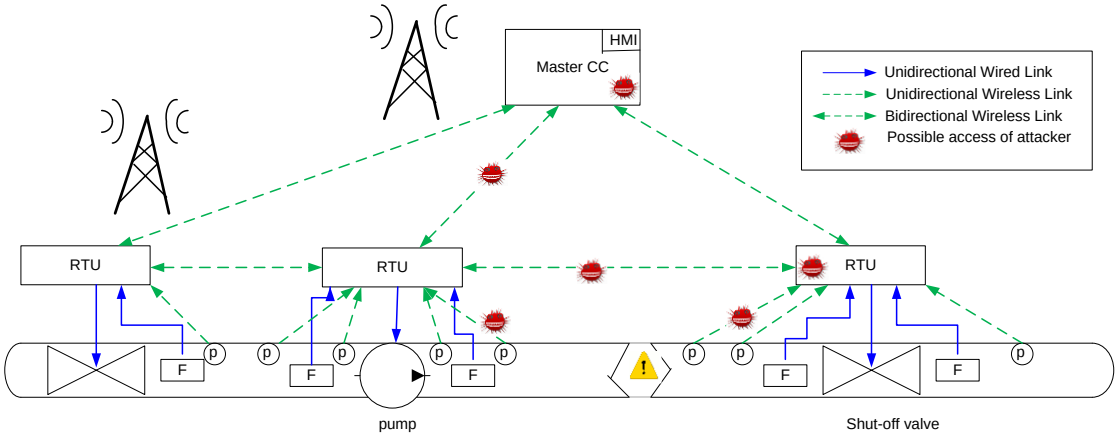


Figure 17: Architecture of the case study

We assume RTUs are locally installed onto pumps and valves and communicate with them via a wired link. Sensors which are relatively distant and scattered all along the pipeline use a wireless link to communicate with RTUs. Supposing that the pipeline goes along hundreds of kilometers and that it is a hundred kilometers distant from the CC, we assume that communication is carried out by a GSM network. The industrial protocols used are Modbus/TCP for RTU-CC communication, Modbus/RTU for inter-RTUs communication and WirelessHART for sensor-RTU communication. These assumptions will be used later to estimate security events parameters.

3.2.2.2 System modeling with BDMP

The BDMP corresponding to this system is given in Figure 18. It models the different scenarios that lead to pollution of the environment (the top event). There are three types of possible scenarios: attack scenarios, accidental scenarios or hybrid scenarios. The first type of scenarios is a successful attack initiated by a malicious person, the second type is based on mere failures of the system's components

and the third type is a combination of attacks and components failures. This latter type best characterizes the possible interactions between safety and security events.

Based on hierarchical reasoning, BDMP cover all the possible scenarios. The top event: pollution can be realized if and only if the pipeline breaks and the protection system fails to react. The protection system refers to the detection of the pipeline break by RTUs and the system shutdown either thanks to the reflex action or by orders sent by the CC. The protection system can fail to react for two different reasons: either it was deactivated *before* the break by an attacker, or it accidentally does not work.

This reasoning corresponds to the top level of the BDMP. The gate named *attack_protection_syst_then_pipeline_break* is a "PAND" gate, which becomes true only if its left input *is* true when the right input *becomes* true. If an attack is perpetrated after a pipe break, this will not worsen the situation. In this model, no defense mechanism against attacks is supposed to exist, but accidental failures can be repaired.

The attack scenario: We suppose that attacks for such an industrial infrastructure follow a Poisson process with an occurrence rate of once every 3 years. The attack scenario starts by deactivating the protection system before provoking the pipeline breach by using the water-hammer phenomenon. In the attack preparation phase the attacker starts by getting access to the SCADA system: either by taking control over the CC (physically or remotely) or accessing physically to the RTU or creeping into the network via the communication link (between the RTU and the CC or between the sensors and the RTU). Secondly, the attacker must understand the system operation in order to be able to deactivate the protection system. Depending on what the attacker has gained access to, he will act differently in order to deactivate the protection. The attack steps in this phase will be quasi instantaneous as the attacker has previously understood the system operation and is able to manipulate it. In order to deactivate the reflex action of RTUs the attacker can simply jam the communication between the RTUs so that the pipeline breach cannot be detected. The house event *No_reflex_action* models the existence or the non-existence of the reflex action as a safety measure implemented locally in the system; this leaf is set either to true or to false prior to any quantification. After preparing for his attack, the attacker is ready to break the pipeline with a water-hammer by provoking a high pumping pressure in the pipeline and closing suddenly the valve downstream which causes a shock leading to a breach at the weakest point in the pipeline.

The accidental scenario: In this case pollution is caused if the pipeline breaks accidentally then the protection system fails to react. The protection failure is realized in two cases: no instructions given by the RTU or the on-demand failure of the equipment (valves and pumps) to react properly. The first case is realized if the RTU fails or if it doesn’t react which implies that it receives no instruction from CC and it does not activate its reflex action. Safety leaves detail the accidental events leading to such scenarios.

The hybrid scenario: This scenario is built up from both accidental and malicious events. We can imagine that the attacker can remotely deactivate the protection system then give up the attack because he does not succeed in creating the water hammer. Then he can just wait until the pipeline breaks accidentally instead of trying another attack. This scenario has a very low probability and supposes that the protection system deactivation is not detected until the pipeline breaks.

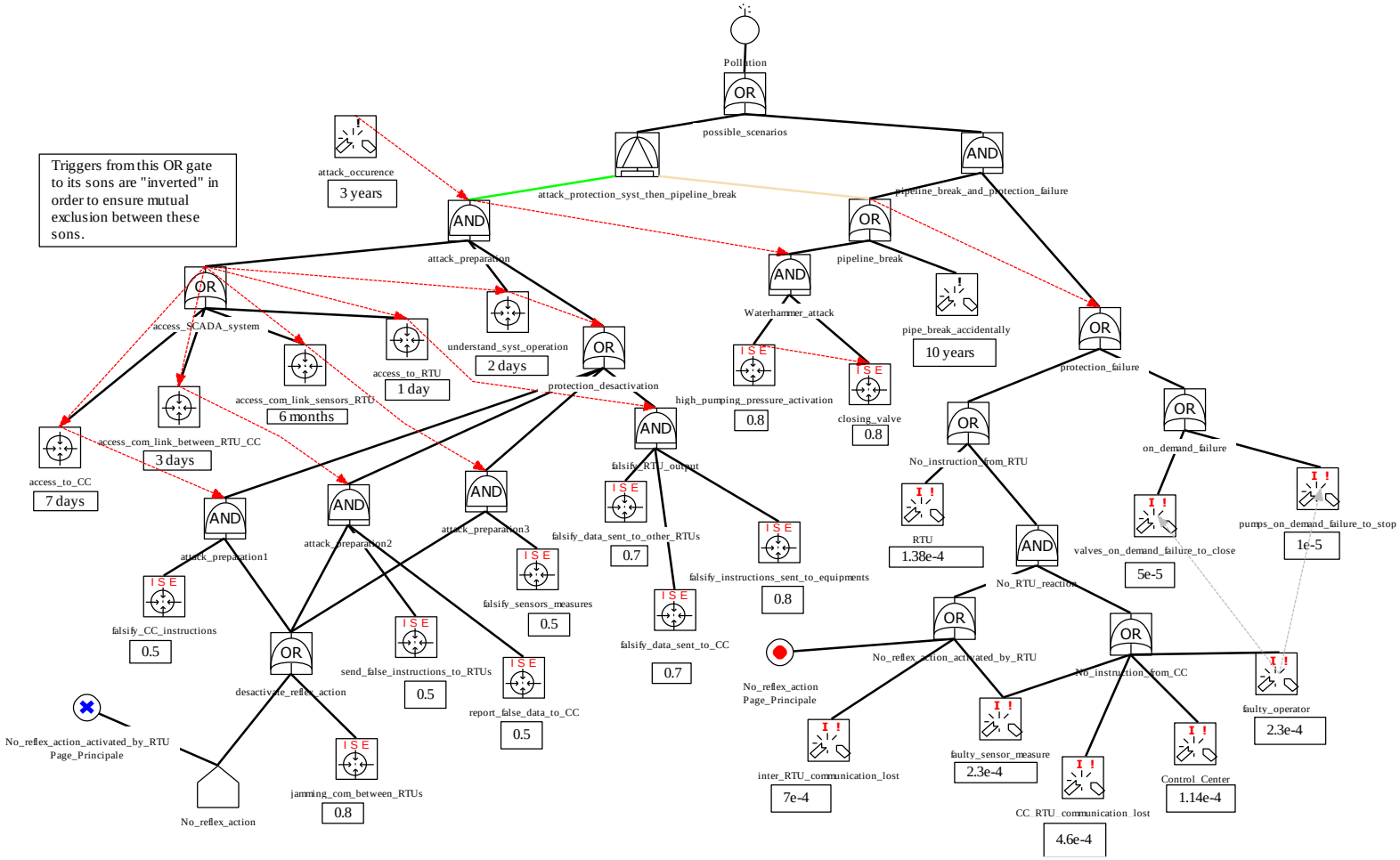


Figure 18: BDMP model of the pipeline and its control system

3.2.2.3 Qualitative and quantitative risk analysis

To make the quantification, we associate the model leaves with parameters based on the estimation of the MTTS for security events, the MTTF for safety events and the probability for *instantaneous events* (see Table 4 and Table 6). Security parameters are estimated based on the assumptions we made on the protocols and the network (see § 3.2.2.1). We also suppose that the attacker has a minimum knowledge of SCADA systems and protocols without necessarily being an insider. These parameters are marked on the model in Figure 18 with comment boxes.

The results given below were obtained with Figseq, as it is explained in section 3.2.1. Based on the given parameters the pollution probability is estimated to 6.18e-2 for a mission time of one year. We can see that attack scenarios are situated at the top of the list of scenarios. The most probable scenario given in Table 10 is the one in which the attacker gets access to the RTU and takes control over the equipment and sends false data to the CC and to the neighboring RTUs.

Transitions		**Proba.**	**Contrib.**
Name	**Rate**		
failF(attack_occurrence)	3.8e-5	4.53e-2	0.733
aa_success(access_to_RTU)	0.04		
aa_success(understand_syst_operation)	0.02		
ise_nd_real(falsify_data_sent_to_CC)	0.7		
ise_nd_real(falsify_data_sent_to_other_RTUs)	0.7		
ise_nd_real(falsify_instructions_sent_to_equipments)	0.8		
ise_nd_real(high_pumping_pressure_activation)	0.8		
ise_nd_real(closing_valve)	0.8		

Table 10: The most probable attack scenario from the BDMP model

We plot in Figure 19 the probability of the most probable sequences according to the type of access. We infer that the RTU is the most critical and vulnerable component in our case study. Being lost in the nature on the pipeline it is easy to attack. These results are of course based on the estimations we give to parameters, for instance we supposed that sensors communicate with RTUs using the WirelessHART protocol which is a secured protocol using authentication, encryption and key checking. The attacker must first find a vulnerability before gaining access to the communication link. On the other hand, the modbus/TCP protocol used for RTUs and CC communication is not secured and data can be clearly read once the attacker accesses the GSM network.

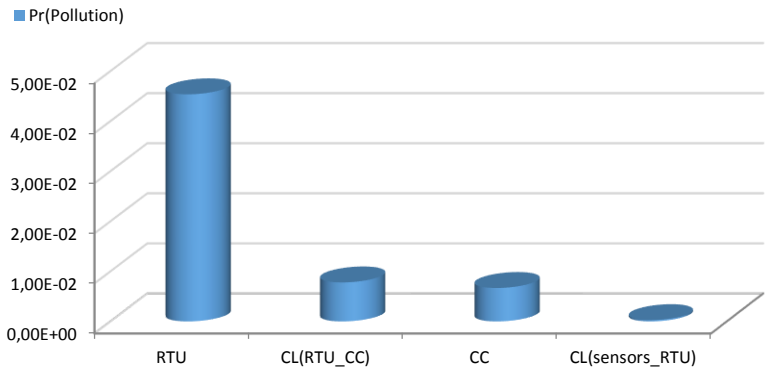


Figure 19: Probability of the most probable sequences according to the type of access

The first hybrid scenario given in Table 11 has a probability of 7.27e-4, in which the attacker deactivates the protection system then gives up the attack before the pipeline breaks accidentally.

Transitions		Proba.	Contrib.
Name	**Rate**		
failF(attack_occurrence)	3.805e-5	7.27e-4	0.117
aa_success(access_to_RTU)	0.0416		
aa_success(understand_syst_operation)	0.0208		
ise_nd_real(falsify_data_sent_to_CC)	0.7		
ise_nd_real(falsify_data_sent_to_other_RTUs)	0.7		
ise_nd_real(falsify_instructions_sent_to_equipments)	0.8		
no_realization(high_pumping_pressure_activation)	0.2		
failF(pipe_break_accidentally)	1.14e-5		

Table 11: The most probable hybrid scenario

The first accidental scenario given in Table 12 appears with a probability of 1.86e-5 and consists of accidental break of the pipeline and failure of the sensors to communicate correct measures to RTUs. This is typically what happened in the case of the Taum Sauk storage tank (cf. Section 5.2). Redundancy among sensors could be considered to prevent such accidental scenarios.

Transitions		Proba.	Contrib.
Name	**Rate**		
failF(pipe_break_accidentally)	1.14e-5	1.86e-5	3.01e-4
good(CC_RTU_communication_lost)	0.99954		
good(Control_Center)	0.999886		
good(RTU)	0.999862		
good(faulty_operator)	0.99977		
failI(faulty_sensor_measure)	0.00023		
good(inter_RTU_communication_lost)	0.9993		

Table 12: The most probable accidental scenario

The results demonstrate that the hybrid scenario is more probable than the accidental scenario. Security events accelerate very much the realization of the undesired event (pollution).

3.2.2.4 Safety and security interdependencies

We propose in this section to highlight the possible interdependencies between safety and security in the use case modeled above.

Mutual reinforcement: The reflex action is a safety module implemented locally at each RTU in order to act in case of accidental pipeline break. In order to assess its influence on the system we calculate the pollution probability with and without reflex action (*No_reflex_action* leaf activated/ deactivated). The results obtained are given in Figure 20.

The pollution probability is higher when no reflex action is implemented at the RTUs. The reflex action represents an additional barrier for the attacker to overcome. If the attacker causes the pipeline breach without deactivating the reflex action this latter would react to prevent pollution as the breach would be detected by RTUs. We can infer consequently that this safety measure reinforces the system security.

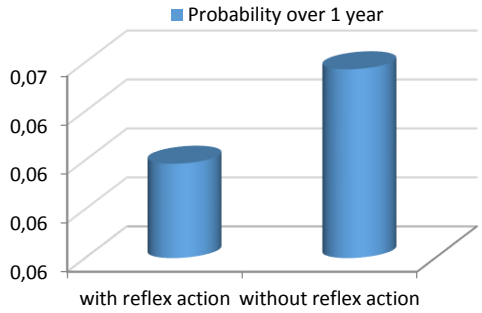


Figure 20: Pollution probability with and without reflex action

Conditional dependency: This kind of interdependency is the most common and implies that the safety level follows the security level. This is more straightforward as attackers' goal is generally to cause safety accidents through compromising the system security. This interaction is illustrated in the two following situations:

- As modeled in Figure 18 the attacker can access the system via the wireless communication link between sensors and RTUs, as no authentication nor encryption is applied. In this case the attacker can manipulate data sent by the sensors to RTUs in order to deactivate the reflex action. The attacker can even exploit the normal functioning of the reflex action to cause the pipeline breach; typically send low pressure measures to the RTU controlling the pump to activate high pumping speed and then when high pressure is reached the attacker can send false low pressure measures to the RTU controlling the valve downstream. This RTU will calculate a high ΔP (high P_{n-1} received from the previous RTU and low P_n given by the attacker) and close the valve leading to a water-hammer. We remind that the reflex action is considered to have a higher priority as a safety module over CC instructions as this latter might detect some inconsistencies in the RTUs measures.

- Enhancing the system security by adding detection and response measures enhances the system safety as it contributes to the reduction of pollution probability. It is possible to include detection aspects in the BDMP. The general idea is that each attack step can be detected at various moments: when it begins, during its progress, when it succeeds, or after completion. Whenever detection occurs, this changes all success rates or probabilities for attack steps which are still to be completed. The only thing the analyst has to do to take detection into account is to change a global option in the model and add in each leaf the detection rate and the realization rate after detection. This does not require any change in the BDMP structure. These detection parameters are taken into account in the quantitative processing. This increases considerably the number of sequences to explore, because each scenario of the model without detection can lead to many variants with detection occurring at various stages. An example of the effect of detection is given in [126], the paper summarized in this Section 3.2.2.

In this example we have been able to put into evidence synergetic interactions between safety and security by modeling safety and security events in an industrial architecture. The qualitative and quantitative analyses enable to rank the scenarios leading to the undesirable event and to identify the most probable scenarios. It is consequently possible to point out the most vulnerable items in the system and take preventive measures accordingly.

3.2.2.5 Conclusion

We illustrated in this section the importance of considering jointly security and safety aspects in the risk evaluation process. Using the BDMP formalism we modeled a realistic industrial case study and put into evidence a synergetic interdependency between safety and security.

Having explored the different possibilities of BDMP, which are representative of formal models widely used in industry, we proposed to explore a radically different approach: the CHASSIS method (cf. §

2.3.1.5), which is based on UML diagrams, in order to better identify the characteristics required for a good modeling approach. We set up a collaboration with the research team that has developed this method, working for the Institute for Energy Technology in Norway. This collaboration was articulated around the comparison of BDMP with CHASSIS in order to see whether an attempt to combine them would be viable and would bring added value. This comparison [134] will be depicted in the next section.

3.3 Comparison of BDMP with the CHASSIS method Modeling

The two methods - Combined Harm Assessment of Safety and Security for Information Systems (CHASSIS) [48] and Boolean logic Driven Markov Processes (BDMP) - are based on extensions of two different approaches, UML and fault/attack trees respectively. While the former method stems from the requirements engineering area and security domain, the latter method is based on risk assessment techniques leveraging quantitative analysis. Still, both approaches are concerned with the combination of safety and security modeling.

We propose in this section to compare BDMP and CHASSIS in order to investigate the two approaches for similarities and differences and to situate them to each other. In particular, we want to investigate whether it is possible to use them together, and whether combining them would be beneficial for addressing safety and security aspects in a more complete and correct manner.

We illustrate in the following the CHASSIS method on the same use case described in § 3.2.2.1 and compare the results obtained from CHASSIS with the results obtained from BDMP given in Section 3.2.2.

3.3.1 Preparing the comparison

We define in this section the preliminary steps preceding the comparison.

3.3.1.1 Pre-study of the approaches

In order to compare the two approaches we initially described them in terms of:

- Background – where is each approach rooted, e.g., other techniques?
- Phase – in which development and risk assessment phases do the approaches apply?
- Input – what inputs are required by the approaches?
- Advantages – what are the advantages of the approaches?
- Limitations – what are the limitations of the approaches?
- Model features and elements – what building blocks are in each model and what features exist to support the modeling?

From this pre-study we can see that the approaches have different backgrounds, but some of the phases and input aspects overlap. The advantages and disadvantages are mainly different, but the model elements and features showed that there could be possibilities for the two approaches to complement each other. With the preliminary results we decided to focus the comparison mainly on the model features and elements.

3.3.1.2 Conducting a case study for the comparison

The cyber-physical system described in § 3.2.2.1 (pipeline and its control system) was used as a case study for the comparison, as it provided a realistic example and gave us the opportunity to collect experiences about similarities and differences of BDMP and CHASSIS. The system had already been modeled with BDMP (cf. § 3.2.2.1), and in order to compare it to an equivalent CHASSIS model, we decided to create a system model of the case study with the CHASSIS method and use the two models as a basis for the comparison.

The system was modeled in several iterations. First, we created an initial model with CHASSIS regarding the architectural description only, not the existing BDMP model. The second iteration included improving some aspects of the CHASSIS model and using the method together with a system expert, taking into consideration also the BDMP model. In the third iteration, the BDMP model was modified in view to the CHASSIS model, thereby harmonizing the two models.

3.3.2 Comparing the model elements

We compared the model elements of the protection system in two ways; taking the BDMP model as starting point and going systematically through all CHASSIS models and vice versa. The model elements, an excerpt of which is given in Table 16, and their relationships were characterized along four possible kinds:

- No – different meaning and naming;
- Indirect – different naming, but implicit meaning;
- Direct – close or different naming, but same meaning;
- One-to-one – the same meaning and naming.

3.3.3 Qualitative comparison of the sequences

After the second iteration of modeling the system (cf. § 3.3.1.2), the sequences created by both models were analyzed. They were identified as important features by both approaches with some resemblance, e.g., outlining the steps in an attack or for accidental system failure. However, there were also differences, e.g., the quantification of sequences generated and the identification of hybrid scenarios mixing attack steps and components failures by the BDMP approach.

In order to undertake a deeper qualitative analysis of the sequences we decided to leverage one of the key features of the BDMP approach: the capability to generate exhaustively the list of sequences leading to the undesired event. The simulation of the BDMP model was done with an observation time of one year as explained in § 3.2.2.3. We compare attack and accidental scenarios generated by the BDMP quantification with the corresponding sequences from the CHASSIS models. For the sequences in Table 17 we investigated the steps involved in the scenarios and their order.

3.3.4 The BDMP model

The BDMP approach was presented in Section 3.1 and illustrated on the use case under study in § 3.2.2. We describe in the following paragraph the BDMP approach according to the terms given in § 3.3.1.1.

BDMP fit in the process of risk evaluation and response for a given system. This process consists in the following steps:

1. Context definition: this step consists in defining: the nature of the risk study, its boundary, objectives and expected results;
2. System description addressing risks: this part aims at providing the reader with the description of the system's architecture, proper functioning and malfunctioning;
3. Risk estimation. This phase includes the following steps:
    a. Analyzing data (statistical data, expert judgments on incident frequencies);
    b. Representing and modeling system related risks;
    c. Exploiting the model (qualitative and quantitative analysis);
4. Choice of prevention and mitigation measures.

BDMP are one possible tool used in the third step of the risk evaluation process for representing and modeling system related risks. Results of the quantitative and qualitative analysis of the BDMP model are directly exploitable in the fourth step.

We select from the quantification results the following attack (Table 13) and accidental (Table 14) scenarios for the comparison previously described in § 3.3.3.

Transitions			Prob.	Contrib.
Steps	**Name**	**Rate**		
1	failF(attack occurrence)	2.28e-5	2.89e-2	1.025e-1
2	aa_success(access comm link between RTU CC)	0.04		
3	aa_success(understand syst operation)	0.08		
4	ise nd real (send false instructions to RTUs)	0.1		
5	ise nd real (report false data to CC)	0.125		
6	ise nd real (reflex action desactivated)	0.067		
7	ise nd real (high pumping pressure activation)	0.7		
8	ise nd real(closing valve)	0.7		

Table 13: Attack scenario from the BDMP model

Transitions			Prob.	Contrib.
Steps	**Name**	**Rate**		
1	failF(pipe break accidentally)	1.14e-5	1.858e-4	6.576e-4
2	good(Control Center)	0.999145		
	good(RTU)	0.99943		
	failI(CC RTU communication lost)	0.0023		
	good(faulty operator)	0.999743		
	good(faulty sensor measure)	0.9989		

Table 14: Accidental scenario from the BDMP model

The next section will address the application of CHASSIS on the use case.

3.3.5 The CHASSIS model

A short presentation of CHASSIS is given in § 2.3.1.5. We give in the following the different models obtained from the application of this method on the use case.

3.3.5.1 Misuse cases

In Figure 21 two D-MUCs for the SCADA system are shown; the safety D-MUC is shown to the left and the security D-MUC to the right. While safety D-MUC represent the misuse cases – threatening the use cases – by grey ovals, security D-MUC show the misuse cases by black ovals.

White ovals show use cases, which are the same for both D-MUCs since based on the same Diagrammatical Use Case (D-UC). However, for security we have added the following mitigating use cases: *Cryptomech. communication* for mitigating *Eavesdropping*, and *Intrusion detection and prevention* for mitigating *Falsify data*. Actors (in white) and "mis-actors"[9] (in grey/black) are surrounding the SCADA system, and what distinguishes humans and external entities are round headed and square headed pin men respectively.

[9] A « mis-actor » is defined as an actor creating adverse effects.

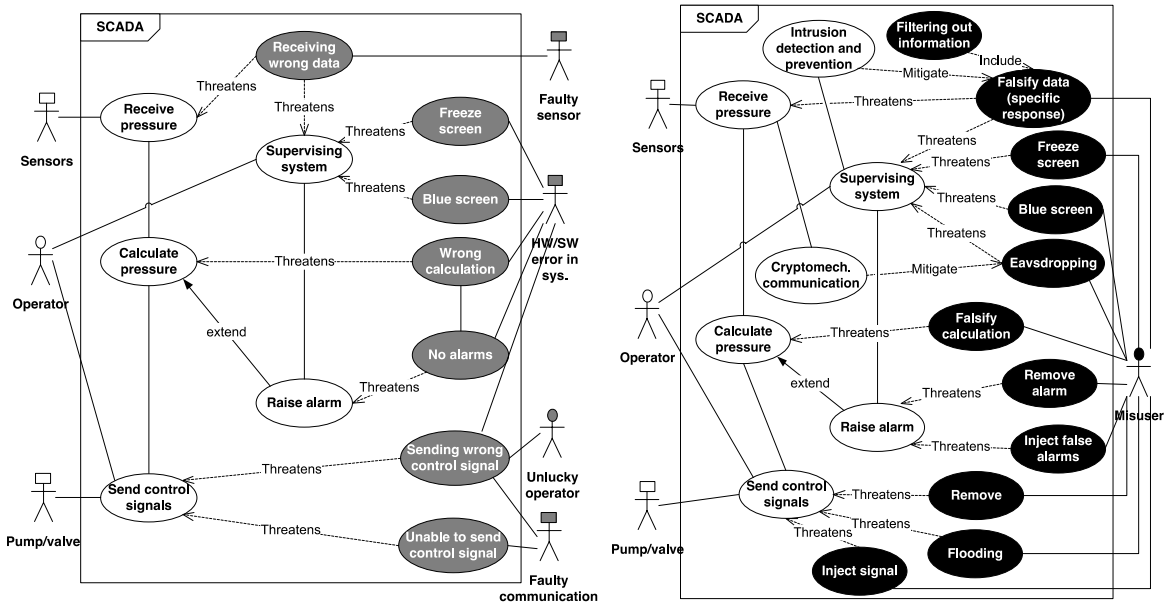


Figure 21: Safety and security D-MUC of SCADA system

Table 15 represents a partial T-MUC[10] describing an attack scenario that builds on the security D-MUC in Figure 21. In the T-MUC more details are added as it is filled, e.g., the basic path describes how the attacker gains access to the SCADA system before he proceeds with misuse such as eavesdropping, injecting control signals at different stages of the attack scenario and removing alarms. The T-MUC is further used to create the MUSD.

Name	Water-hammer attack
Summary	Attacker gains access, manipulates the system to cause the water hammer attack
Basic path	bp1. Attacker gains access to protection system bp2. Attacker eavesdrops communication to understand functioning of system bp3. Attacker injects control signals to increase pressure in pipeline (pump speed) bp4. Attacker removes alarm signals bp5. Attacker falsifies data for supervising system bp6. Attacker intercepts communication data to know when pressure is very high bp7. Attacker injects control signal to close suddenly valve to provoke water hammer attack
Alternative paths	ap1. In bp3, the attacker could falsify pressure data sent to supervising system, to make operator increase pumping speed. ap2. In bp7, attacker allows alarm signal to go through to supervising system, which automatically shuts valves and causes the water-hammer attack.
Mitigation points	mp1. Crypto mechanisms of communication will mitigate bp2 – bp7 mp2. Intrusion detection and prevention system will mitigate bp1, bp3-bp5, bp7
Assumptions	as1. Attacker has minimum knowledge of how system is functioning as2. There is no authentication required within the system as3. The data sent within the system and its network is not encrypted

Table 15: Partial security T-MUC for a water hammer attack

[10] We have only included a few of many fields from the T-MUC due to space reasons. A complete T-MUC is presented in [86].

3.3.5.2 Misuse sequence diagrams

The MUSD shown in Figure 22 corresponds to the security T-MUC in Table 15, as it models the SCADA system and how the attacker proceeds to misuse vulnerabilities of components and their interactions (shown as red/dashed notation) in order to achieve a water hammer attack.

3.3.5.3 Failure sequence diagrams

As seen in the previous subsection the MUSD corresponded to the security T-MUC. However, MUSD and FSD can be used iteratively with T-MUC by adding more details as they emerge while modeling. They can also be used for providing an overview as shown in Figure 23. The component SCADA Ref SCADA DC in Figure 23 refers to the decomposed version of this subsystem in Figure 24 [11].

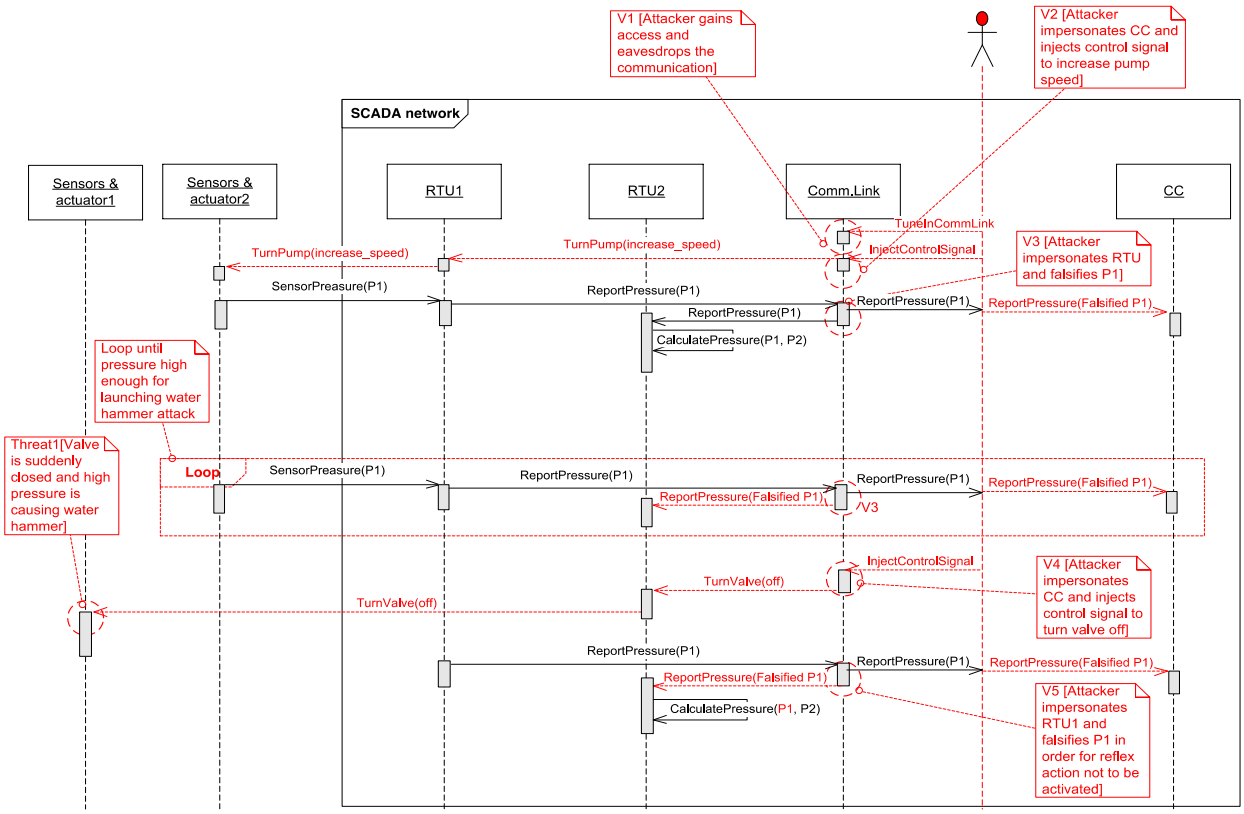


Figure 22: Misuse sequence diagram for a water hammer attack

[11] Note that due to space reason we have not included the complete decomposed FSD in Fig. 24, which excludes the failures 1 and 2, and hazard 1 from Fig. 23. Furthermore, we have not included mitigations in the MUSD and FSD.

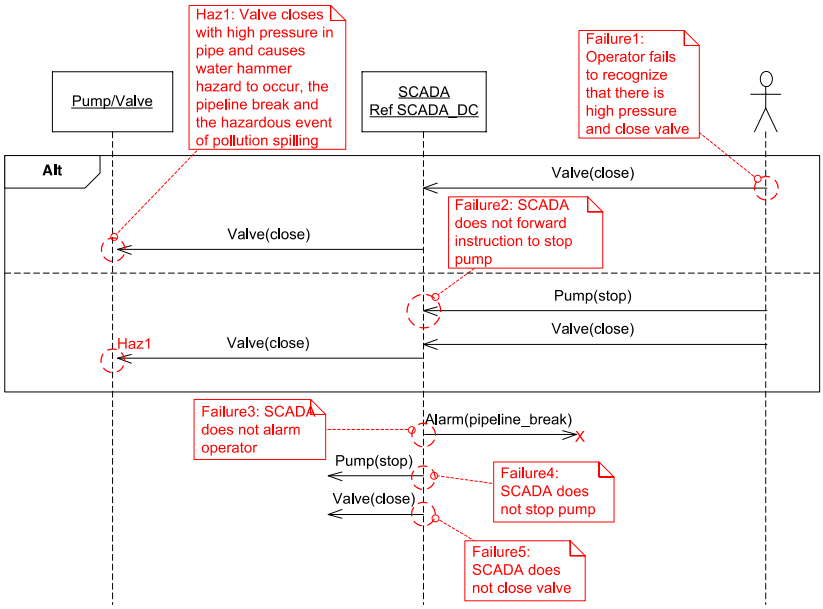


Figure 23: FSD providing the overview of failures in the SCADA system

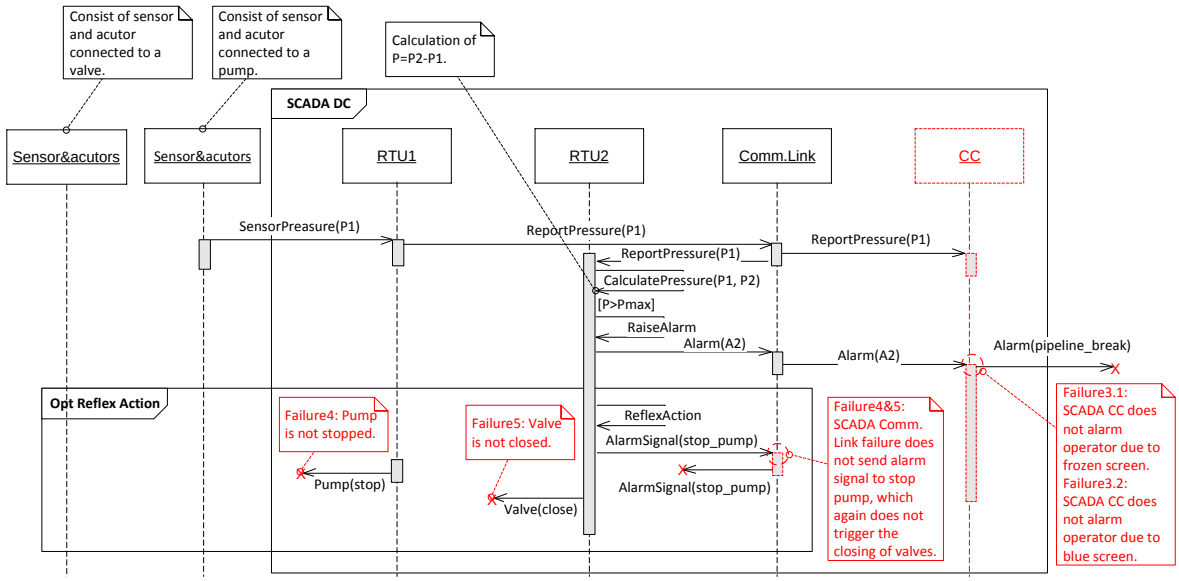


Figure 24: A decomposed FSD providing the details of failures in the SCADA system

Figure 24 provides the details of the SCADA component by decomposing it into subcomponents, and shows where and how the failures occur at the system sub levels.

In the following section, we summarize the main results obtained from the comparison of BDMP and CHASSIS model.

3.3.6 Results of the comparison

For comparing CHASSIS and BDMP models we looked at the coverage of the model elements for each model. As explained earlier, BDMP and CHASSIS are based on different approaches; attack/fault trees and UML diagrams respectively. This makes the two models look quite different at first sight. However, having the same objective of addressing safety and security in a combined manner, the models have a certain degree of resemblance.

3.3.6.1 Comparing the modeling elements

We compared the model elements following the methodology described in Section 3.3.2; the results are given in Table 16.

Comparing BDMP to CHASSIS	**Comparing CHASSIS to BDMP**
BDMP model → Security D-MUC: • No relation for root node *unwanted event* • For some *attacker actions* there are indirect and one-to-one relations • For one of the *AND* and one of the *OR* gates there are indirect relations	Security D-MUC → BDMP model: • For actors and misuser there are indirect relations • For use cases/functions there are mostly no relation (includes mitigations) • For misuse cases/threats there are mostly relation, both direct and indirect
BDMP model → Safety D-MUC: • No relation for root node unwanted event • Only one direct relation for (CC_failure)	Safety D-MUC → BDMP model: • For actors there are indirect relations and for misuser there are direct relations • For use cases/functions there are mostly no relation (includes mitigations) • For misuse cases/threats there are no or indirect relations
BDMP model → MUSD: • For the root node unwanted event there is an indirect relation • For AND and OR gates there are mostly indirect or direct relations, with a few exceptions where there is no relation • For attacker actions there are mostly direct relations, with a few exception where there is no relation	MUSD → BDMP model: • For all objects there are indirect relations, except the attacker object that has direct relation • For the messages the relations are: • No or indirect for internal messages and operator • Mostly indirect for system messages • Mostly direct for attack messages • Direct or one-to-one for notes about attack (vulnerability descriptions)
BDMP model → FSD: • Only indirect relations exist The BDMP *implicitly* defines a large number of sequences or scenarios, which can be automatically made explicit thanks to a tool like FigSeq. An FSD is the explicit description of a single scenario.	FSD → BDMP model: • For all objects there are mostly indirect relations, except the failed CC object that has a one-to-one relation • For the messages the relations are: • No or indirect for internal messages and operator • Mostly indirect for system messages • Mostly indirect for failure messages • Direct or one-to-one for notes about failures (descriptions)

Table 16: Extraction of two way comparison of BDMP and CHASSIS model elements

The comparison of model elements summarized in Table 16 shows that there are more direct and one-to-one relations for the FSD and MUSD, than for D-MUCs, when compared to the BDMP model. This is due to the nature and process of creating D-MUCs, which we will discuss further in subsection 3.3.7. Although BDMP do not in particular aim at integrating system modeling elements with failure and attack modeling elements, as is the case for CHASSIS, we observe that BDMP are often related indirectly to such modeling elements from the CHASSIS model. In particular, this is the case for messages and

objects in MUSD and FSD. The CHASSIS model could support the BDMP model in making this integration more explicit and detailed by using FSD and MUSD, both prior to the BDMP analysis, as an input, but also by summarizing the results from the BDMP model toward system model elements. This applies in particular for integrating the system architecture with attack steps and components failures. On the other hand, BDMP quantification results can guide and focus the FSD and MUSD towards the relevant failure and attack scenarios, leading to the unwanted event. Furthermore, BDMP allows for modeling (implicitly) many alternatives and thereby different scenarios. Although FSD and MUSD from CHASSIS allow modeling of alternatives through the "Alt operator" (shown in Figure 23), in explicitly defined sequences, this does not reach the modeling power offered by BDMP. We will discuss this further in the next subsection when analyzing the sequences generated by BDMP and CHASSIS.

We did not compare the T-MUCs and HAZOP tables to the BDMP model, as our focus was on graphical modeling. T-MUC and HAZOP table provide a structured textual description, supporting both D-MUC and MUSD/FSD. Comparing D-MUCs with BDMP shows that their model elements can be related to each other, i.e., elements like actors, misusers and misuse cases from the D-MUCs relates to the leaves in the BDMP. Furthermore, there is a greater relation between the MUSD and FSD, and the BDMP model, although the system interaction modeled by the CHASSIS model cannot be found, or only indirectly found, in the BDMP model. T-MUC enriches the D-MUC, FSD and MUSD with more details through textual descriptions, and this would also be the case for the BDMP model, wherever there are relations between the CHASSIS and BDMP models.

3.3.6.2 Qualitative comparison of sequences

We discuss and compare in this section the scenarios modeled by CHASSIS and BDMP. Table 17 gives a comparison between a scenario given by the BDMP quantification and the scenario modeled by the MUSD in Figure 22.

Step	BDMP	MUSD
1	failF(attack occurrence)	NA
2	aa_success(access comm link between RTU CC)	V1/TuneInCommLink
3	aa_success(understand syst operation)	V1
4	ise nd real (send false instructions to RTUs)	InjectControlSignal/V2
5	ise nd real (report false data to CC)	ReportPressure(FalsifiedP1)
6	ise nd real (reflex_action_desactivated)	ReportPressure(FalsifiedP1)
7	ise nd real (high pumping pressure activation)	TurnPump(increase_speed)
8	ise nd real(closing valve)	TurnValve(off)

Table 17: The most likely scenario from BDMP compared to corresponding CHASSIS scenario

In the MUSD we recognize the attacker steps of the scenario given by the BDMP model. The order of the attack steps is respected in the first steps but not for the rest of the scenario: the attacker starts by gaining access to the communication link and eavesdropping. Next, he impersonates the CC and sends false instructions to RTU then impersonates the RTU and sends false sensor measures. In the BDMP scenario, the attacker deactivates the reflex action before launching the water hammer attack while in the MUSD he breaks the pipeline then prevents the reflex action from reacting. To deactivate the reflex action the attacker falsifies sensor measures sent between RTUs and removes alarms that may be raised. The water hammer attack is caused by a high pumping pressure then a sudden close of the downstream valve. In the MUSD, the attacker sends the order of increasing pumps speed from the start of the scenario, which is not the case for the BDMP scenario where the attacker starts by deactivating the protection system before launching the water hammer attack. This difference reflects a slightly diverging

view of how an attack would be conducted by the BDMP team and the CHASSIS team. It is by no means an effect of the difference between the approaches themselves.

While BDMP model the different attack steps of this malicious scenario, MUSD provides more details on this scenario by showing the different components involved and the interaction between them. The interactions include the attacker actions and vulnerabilities he exploits (in red/dashed), as well as normal interactions (in black/solid) between the system components.

The scenario modeled by the MUSD is only one possible scenario among all other scenarios modeled by the BDMP model. It is of course unrealistic to represent all the scenarios (3184 scenarios) given by the BDMP with MUSD. Yet MUSD can be used posterior to BDMP to provide more details on the most probable scenarios given by the quantitative analysis of the BDMP.

Concerning accidental scenarios, we note that the scenario given by BDMP is different from the scenario represented by the FSD of Figure 23 and Figure 24. In the BDMP model, details on how pipeline breaks accidentally are not shown for space and readability reasons. In the FSD the pipeline break is the consequence of a water-hammer caused by either a faulty operator or a SCADA system failure. The protection system failure is caused in the BDMP model by absence of instruction from the CC and RTUs due to communication link failure, while in FSD (Figure 23) it is caused by a frozen/blue screen (Failure 3) and communication link failure (Failure 4&5). According to quantification results from BDMP, the FSD scenario has a probability of about 1.6 e-7, while the most probable accidental scenario has a probability of 1.14 e-5.

Finally, the BDMP model yields hybrid scenarios, whereas this is not the intention of CHASSIS, because of the separation of the security and safety parts of the model. CHASSIS does however combine safety and security mitigations during the later trade-off analysis, but we have not included this analysis in our current comparison. In the present case study, the hybrid scenarios were negligible, but it is hard to say that this is always true.

3.3.7 Experiences with applying the two approaches

After having modeled the same system with both approaches and exchanging experiences, we see that CHASSIS and BDMP have many similarities and differences. While CHASSIS applies a methodical process that starts by requirements elicitation facilitated through graphical representation of threats and hazards related to the system, BDMP fits in the risk estimation step of the risk evaluation process. BDMP constitute a very concise graphical representation of all scenarios leading to the unwanted event, and enable an automatic qualitative and quantitative analysis of these scenarios.

3.3.7.1 BDMP and CHASSIS in a risk evaluation process

In Section 3.3.4 a four-step risk evaluation and management process was outlined. BDMP cover the risk estimation (third step) in this process, by representing and modeling system related risks and exploiting the model thanks to advanced quantification capabilities. CHASSIS is on the other hand defined to cover parts of the requirements engineering process, as shown in Figure 10, mainly covering the elicitation activities of functional, safety and security requirements, but also analysis of trade-offs between these requirements and specifications. The first step of the risk evaluation process is associated with requirements elicitation, which is within the scope of the CHASSIS method. Furthermore, CHASSIS provides parts of system descriptions, i.e., functional and architectural by UC and SD, and addresses risk towards these descriptions by integrating safety and security aspects into one common model. Thereby, CHASSIS satisfies the second step in the process, by identifying many attack actions and failures based on system descriptions, and combining those into different scenarios. Its strength is however not in combining all possible sequences of attack actions and failures. CHASSIS does not

provide risk estimation for safety and security aspects. BDMP has risk estimation features that enable qualitative and quantitative analysis of the model, which covers the third step. The fourth and final step of the risk evaluation process corresponds to the trade-off analysis in the CHASSIS process. Both techniques are concerned with the right choice of mitigation and prevention measures for malicious and accidental scenarios. BDMP can by the risk estimation determine which failure or attack sequences should be considered, while CHASSIS can support the identification and trade-off analysis between mitigation/prevention measures.

3.3.7.2 Combining BDMP and CHASSIS

The different comparison iterations explained in Section 3.3.1 between the two modeling techniques made us infer that at different steps of the modeling, both techniques can be beneficial to each other. The combination of both techniques can result in the following process:

1) In a first step, UC and TUC exhibit more knowledge about the system proper functioning;

2) In a second step, MUC provide a large scope on different hazards and threats that can harm the system. All these elements can provide input for the BDMP model that can be conceived in a third step to give a macroscopic representation of the combination of all events that can lead to a given unwanted event;

3) In a fourth step, the BDMP model is quantified and results state the list of different scenarios leading to the unwanted event ordered by their probabilities and contribution;

4) In a fifth step, the most likely scenarios given by the BDMP quantification are represented with more details using MUSD and FSD out of CHASSIS process.

3.3.8 Conclusion

We compared in this section CHASSIS and BDMP, two approaches that consider both security and safety aspects in the risk evaluation process for cyber-physical systems. Our comparison clarified similarities and differences between the two approaches; CHASSIS is a methodical process used for integrating system description with safety and security risks at early stages. It provides good graphical visualization capabilities as it is based on UML. On the other hand, BDMP have powerful quantification capabilities and a potential for exhaustiveness that enable risk estimation regarding an unwanted event related to the system. BDMP also offer good readability and scalability thanks to their hierarchical structure that shows different levels of abstraction of the modeled risk. Nevertheless, the two approaches can provide input to each other and models obtained are complementary by providing different knowledge to the stakeholders.

Results of the comparison have shown that both approaches can mutually strengthen each other: CHASSIS can provide more detailed description of the system and related hazards as an input to BDMP for more thorough and quantitative analysis.

3.4 Discussion on BDMP

We investigated in this section the potential of the BDMP approach to model jointly safety and security for complex and realistic systems and to identify their potential interdependencies. By applying it to a realistic case study of a pipeline, we have proven the ability of this formalism to model, with a compact and fairly readable graphical model, the different risk scenarios associated to both safety and security. The quantification capability of BDMP is a strong side of this formalism as it enables to generate from the BDMP model the risks scenarios ordered by their decreasing probability.

The main limitations encountered with BDMP usage are the following:

1. The BDMP model is manually built by the analyst given the system architecture and an undesirable event. This implies that:
    a. The risk model obtained depends on the analyst's understanding of the system and its related risks. This results in subjective models; i.e. two different analysts modeling the same undesirable event for the same system architecture are likely produce different models;
    b. The analyst has to rebuild manually the BDMP model each time he wants to model a different risk event or each time the system architecture changes, which is time and effort consuming.
2. It is difficult to figure out the system architecture from the BDMP model as this latter does not directly reveal the functional aspects of the system and the relationship between its different components. BDMP models are consequently difficult to understand for readers if they are not provided in advance with a detailed description of the system.

Having these limitations in mind, we believe that an automated approach is required for complex industrial control systems. This approach must be based on safety and security experts' knowledge gathered in a knowledge base, in order to facilitate the model construction, ensure objective modeling and hence the reproducibility of results.

Therefore, in addition to C1, C2 and C3 discussed in Section 2.4.3, the required approach should satisfy the following criteria:

- C4: automatic generation of risk models thanks to a knowledge base; which makes the approach easy to use by engineers and even non-experts. Besides, the models generated, based on the KB and the system description, are the same independently from the analyst and their objectivity is ensured;
- C5: robustness, in the sense that one can bring changes (to model for example different assumptions about the system) without need to revamp the whole model.

We present in the next chapter the S-cube approach that overcomes the limitations of existing approaches and fulfills all the required criteria.

Chapter 4

The S-cube approach: a model-based approach for SCADA Safety and Security joint modeling

Considering the limitations of existing approaches previously mentioned, we propose in this section a new model based approach for SCADA Safety and Security joint modeling.

The S-cube approach satisfies all the criteria required and discussed in Sections 2.4.3 and 3.4:

- C1: it enables a joint safety and security risk analysis for systems having safety challenges and integrating new information technologies and particularly SCADA-based ICS;
- C2: it enables formal modeling of the system architecture, and the related attack and failure modes;
- C3: it yields both a qualitative and quantitative analysis;
- C4: it generates automatically attack and failure scenarios that lead to a given undesirable event, from a description of the system architecture;
- C5: it enables to easily consider different hypotheses about the same system architecture and regenerate the new risk related scenarios.

The S-cube approach relies on a knowledge base (KB) that gathers safety and security expertise on SCADA systems and the associated information systems. In this chapter, we present first some of the already existent “safety only” or “security only” domain specific languages and explain how they inspired the S-cube approach. We detail, next, the S-cube approach principle and knowledge base. Then we address the qualitative and quantitative analysis provided by S-cube. We explain, finally, how this approach has been implemented.

4.1 Existing safety and security domain specific languages

DSL (Domain Specific Languages) aim at capitalizing knowledge on a specific domain. As safety and security have been for a long time treated separately within distinct communities, there are already existing DSLs for modeling either safety or security.

4.1.1 Security domain DSLs

Our exploration has been focused on two main approaches that are based on security DSLs and enable to automatically process system models: CySeMoL [135] and MulVAL[136]. We also had a look at DSLs designed for the formal definition and verification of protocols such as Proverif [137] but we could quickly discard them because they are only limited to the specification and verification of security properties related to cryptographic protocols. Modeling higher level and abstract aspects of the system is not possible with such highly specific languages.

4.1.1.1 The Cyber Security Modeling Language (CySeMoL)

CySeMoL [135] is an attack graph tool that can be used to assess the cyber security of enterprise architectures. It allows users to create models of their architectures and make calculations on the likelihood of different cyber-attacks being successful.

CySeMoL is based on a metamodel that consists of 23 assets, 59 attack steps, 58 defenses and 51 relationships between assets [138]. The attack steps and defenses likelihoods are estimated using Bayesian inference and based on the following assumptions: the attacker profile is a professional penetration tester and he has one week available for the attack.

The CySeMoL approach enables to model only IT components and the security-related risks (criteria C1 not satisfied, cf. Section 2.4.3). The possibility of extending CySeMoL in order to cover both safety and security had been considered but discarded for the following reasons:

- the CySeMoL metamodel is too comprehensive and enriching it with new elements requires to rethink all the dependencies and relationships between the metamodel's elements;
- the quantification process is a Monte Carlo based calculation of a Bayesian network. It does not consequently allow to model dynamic aspects of the attack and particularly its evolution over time (mean time until success). Instead, all parts of the metamodel assume that the attacker has one week to perform the attack;
- the calculation engine is not open source.

We indicate however that some notions modeled in the S-cube KB were inspired from the CySeMoL metamodel and templates.

4.1.1.2 Multihost, multistage Vulnerability AnaLysis (MulVAL)

Another attack graph tool used for security-related vulnerabilities assessment is the MulVAL tool [136]. It exhibits an engine that enables to automatically generate attack graphs given the network configuration and security advisories given by a network scanner that identifies vulnerabilities on each host. This engine uses a set of reasoning rules that specify exploit rules, compromise propagation and multi-hop network access. MulVAL is essentially qualitative: its main output is a logical attack graph, i.e. a logical structure that can be enriched by metrics providing an assessment of the difficulty of various attack steps. Hence one can see that "logical attack graph", as used by Ou *et al.* in [136], is just another name of attack tree.

Like CySeMoL, MulVAL does not consider safety issues (C1 not satisfied). The possibility of extending MulVAL for safety and security joint modeling seems difficult to us for the following reasons:

- MulVAL adopts the Datalog language that is a non-typed language; it is consequently difficult to track the different modeling elements especially for complex systems and to associate them with their characteristics. We believe that an object-oriented language is more appropriate to this purpose;
- MulVAL uses non-user-friendly tools. All inputs have to be in textual form, and the "graphical" output is hardly usable for complex systems;
- The quantification of MulVAL models is too simplistic and could not be extended to safety related parts of a model.

The exploration of MulVAL was however beneficial and inspiring for the S-cube KB.

4.1.2 Safety domain DSLs

In the safety domain, the situation is more complicated than in security, because there are two "levels" of domain specific languages: generic languages that enable to build knowledge bases as libraries of classes and those libraries themselves that enable to build system models.

For generic languages, we present concisely the three languages: Figaro, AltaRica and O3prm, which have been designed for modeling systems and facilitating the safety analysis associated to them.

Figaro will be presented in Section 4.6.1. The BDMP formalism (cf. Section 3.1) and the S-cube KB (cf. Section 4.6.1) are examples of libraries implemented using this language.

Similar to Figaro, AltaRica [139] is a high-level language that enables to create models of systems. A model describes the hierarchy of nodes; each component can embed several sub-nodes. The dynamic behaviors of the system components are represented using a state machine. The state of a component is represented by state variables and their values. The changes of states are possible when an event occurs; which updates the variables values. A complete description of this language and the associated tools is depicted in [140].

In [141], Bouissou *et al.* compare the Figaro and AltaRica languages with an illustration on an electrical case study. The main difference between the two languages resides in the fact that Figaro enables to conceive generic modeling libraries that can be used by engineers through graphical user interfaces, without the need to manipulate the language itself; while AltaRica enables to reuse some elements specified in knowledge bases but requires additionally from the system analyst to add AltaRica code. Figaro is hence more end-user-friendly.

Open Object Oriented Probabilistic Relational Models (O3prm) [142] is a language, created by LIP6 and EDF R&D on the basis of a language developed in the context of the SKOOB ANR project (2008-2011), in order to represent probabilistic relational models (PRM). PRM are a powerful extension of Bayesian belief networks. Inspired by relational languages and improved by object oriented paradigms, they allow consequently the definition of classes of objects and the relations among them. PRM overcome the limitations of the BBN formalism related to the modeling of complex systems as it quickly loses its expressivity due to the large number of the needed variables. In O3prm, variables are gathered into clusters called classes. Classes are generic entities and can be reused in the description of several systems, thereby making this description much easier and faster. On the other hand, PRM inherit from BBN their ability to represent uncertainties related to hazards and make rigorous probabilistic calculations for risk assessment.

To summarize, the three modeling languages Figaro, AltaRica, O3prm are generic DSLs, helping the definition of more specific DSLs in the form of libraries. We note that there are few examples of libraries written in O3prm for safety. On the contrary, there are several examples of Figaro libraries that have

been in use at EDF for many years, to carry out safety studies of systems from the thermo-hydraulic and electrical domains.

Our ultimate goal is to have a DSL that enables modeling SCADA systems regardless of the kind of the physical system they control. Such a DSL can be later coupled with other knowledge bases modeling the real behavior of domain specific system components, and the associated types of failures. In addition, in order to be suitable for modern industrial systems and their rapid evolution, this DSL should also be scalable and reusable if different hypothesis were to be taken into account.

For these reasons, we have chosen to build a new knowledge base (or DSL) that gathers expertise on industrial information and control systems and the associated safety and security aspects. This KB is the core of the S-cube approach: our main scientific contribution in this thesis. It incorporates some notions inspired from the existing DSLs previously mentioned but tries to overcome at the same time some of their limitations. The S-cube approach aims at providing a common framework for dealing with the convergence of safety and security risks in modern control systems in order to capture their mutual interactions.

We provide in the remainder of this chapter the main principles of the S-cube approach and the different stakeholders involved in the process using it. We detail next the main notions modeled in the S-cube knowledge base. We explain finally how this approach has been implemented.

4.2 The S-cube approach: principle and stakeholders

We first present in this section the principle of the S-cube approach [143], then outline the different stakeholders involved in the overall process.

4.2.1 Principle of the S-cube approach

Seen as a black box, the S-cube approach, depicted in Figure 25, takes as input the system architecture and gives as output the attack and failure scenarios that are likely to happen on it and that may lead to a given undesirable event. Undesirable events associated to a given system can be identified in advance by some safety systematic techniques such as FMEA or HAZOP. These events represent risks with intolerable consequences that can happen to the system and lead to safety issues (human losses, environmental and ecological impact, or high economic losses).

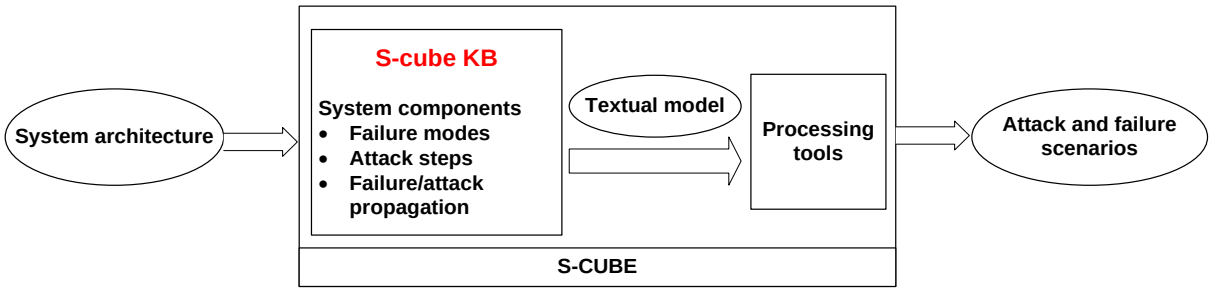


Figure 25: The S-cube approach principle

The S-cube approach relies on a knowledge base, called later S-cube KB (cf. Section 4.3), that gathers expertise on ICS and particularly SCADA systems and their associated safety and security aspects. The S-cube KB can be seen as a Domain Specific Language or a library that enables describing the typical components of digital industrial infrastructures, including corporate enterprise network, industrial control network, field and instrumentation networks; and the related security mechanisms (authentication, access control, etc.) and safety mechanisms (redundancy, voters, etc.). Each component

is associated with the attacks and failure modes that can happen on it (cf. Section 4.4). The effects of these failures and attack steps are described and propagated within the overall system architecture.

The generic models of the S-cube KB are instantiated on the input system architecture. This instantiation results into a textual model which can be processed by calculation engines that generate automatically attack and failure scenarios, with an estimation of their probabilities.

4.2.2 Stakeholders

In order to explain the use cases derived from the S-cube approach and the stakeholders involved in the different steps, we establish a sort of "use cases-flow" diagram, depicted in Figure 26.

We have built during the thesis the S-cube KB, labeled Phase 0 (Ph. 0) in Figure 26. The rationale behind this knowledge base and the different notions modeled will be detailed in the following sections of this chapter. The S-cube KB describes generically the different components present in industrial information and control architectures; this task has been supported by system designers and experts in the industrial domain. Each system component is next associated with its failure modes (cf. Section 4.4.1) and the corresponding safety metrics (cf. Section 4.5.1); this task has been supported by a safety expert. In the same way, each system component is associated with the attacks (cf. Section 4.4.2) likely to target it and the corresponding security metrics (cf. Section 4.5.2); this task has been supported by security experts. The propagation of the failure and attack effects into the whole architecture is also included in the KB; this task was supported by safety resp. security experts.

In order to build the KB, we have been frequently seeking expertise on industrial architectures from system designers and from safety and security engineers of the industrial domain. The knowledge base transcripts consequently the knowledge we gathered from these different stakeholders.

The S-cube KB aims at providing patterns that can be reused in order to model industrial architectures and assess the safety and security risks they are subject to. The task of building the KB (Ph.0) can be seen as the preliminary but fundamental phase of the S-cube approach. The system models and the results obtained are the transcription of what has been modeled in this KB.
The S-cube KB facilitates consequently the risk analysis of industrial architectures and enables automating the other phases of the process (in particular Ph.1 and Ph.2 in Figure 26), as explained below:

Phase 1 (labeled Ph.1 in Figure 26) consists in the system description, which will be input in S-cube. This phase consists in the following steps:

- Step (1.i): First, the system designer/expert, builds the system functional architecture using the modeling elements, that correspond to the classes defined in the S-cube KB (cf. Section 4.3.2). The system functional architecture describes the different network zones of the information system, the machines connected to each zone, the software running on each machine, and the data flows between the different software components. These elements will be detailed in Section 4.3.1.

Once the functional architecture is defined, dysfunctional aspects can be included following the steps (1.ii) and (1.iii) in Figure 26. The order of these steps is not important.

- Step (1.ii): a safety expert defines the safety aspects related to the architecture: for instance, the common cause failure groups (cf. Section 4.4.1). He/she also parameterizes the model with the corresponding safety metrics (cf. Section 4.5.1), for instance the mean time to failure of the physical components (e.g., PLCs, sensors);

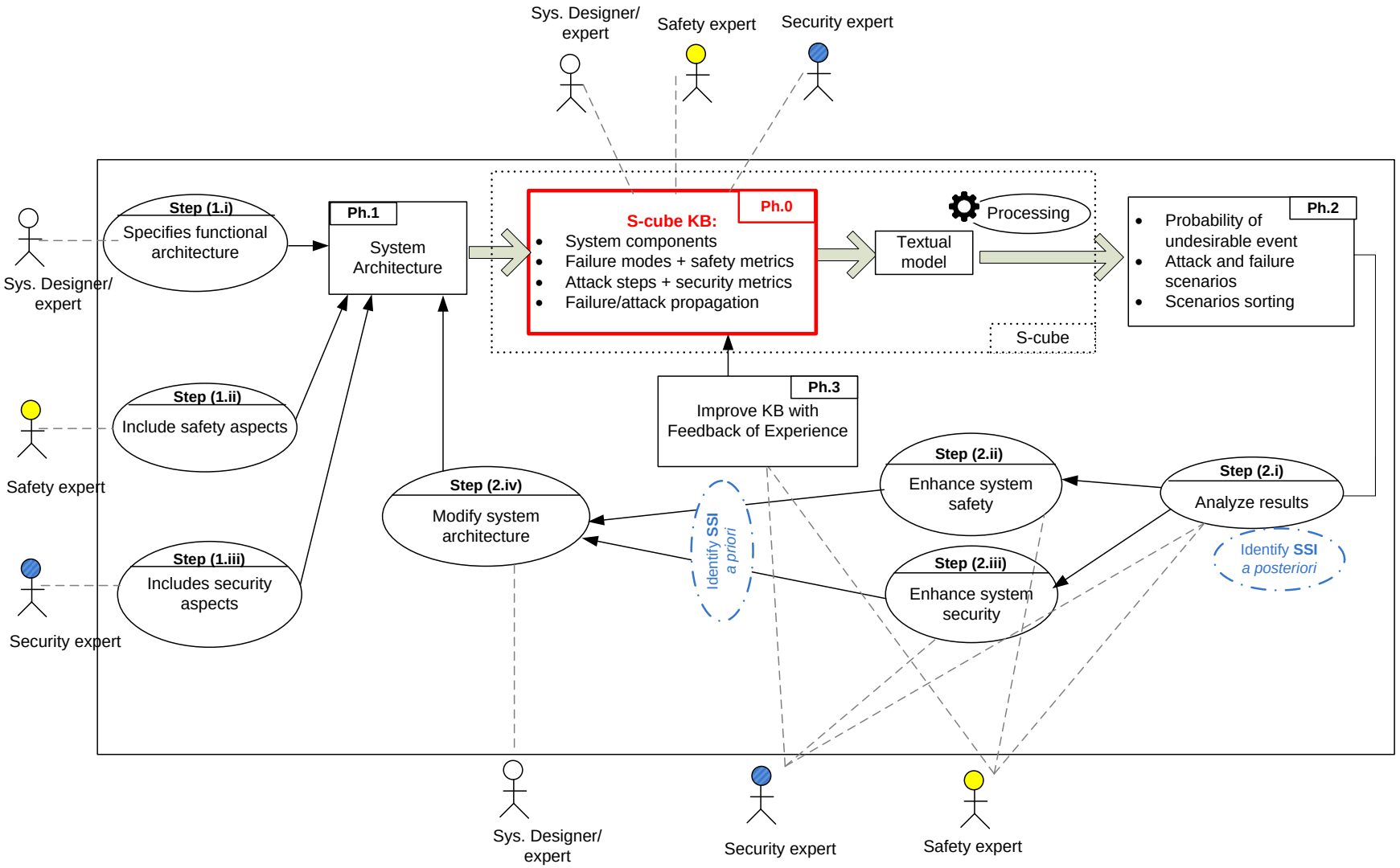


Figure 26: Different steps and stakeholders involved by S-Cube

- Step (1.iii): a security expert defines the security aspects related to the architecture: the vulnerabilities present on the software components, firewalls, authentication mechanisms, etc. He/she also parameterizes the model with the corresponding security metrics (cf. Section 4.5.2).

The three steps of Ph.1 yield finally the system architecture with its functional and dysfunctional (safety and security) aspects. We believe that communication is essential, but unfortunately not always present, between the different stakeholders (Sys. Designer/expert, safety expert, security expert) in order to check the consistency of the architecture. This latter can be input in S-cube, as explained in the previous Section 4.2.1, which outputs the results in Phase 2 (labeled Ph.2 in Figure 26): attack and failure scenarios leading to a given undesirable event, sorted by decreasing probabilities, and the probability of the undesirable event. The steps of this phase are described below:

- Step (2.i): safety and security experts analyze the results yielded by processing the system architecture with S-cube. The most probable scenarios are prioritized. We can see, for instance, whether these scenarios are accidental or malicious, the failure modes/attack steps they are made of and the probability of each scenario;
- Step (2.ii): according to the results obtained, and particularly the accidental scenarios, safety experts decide on the appropriate safety measures to deploy in order to enhance the system safety, which are likely to decrease the probability of the undesirable event;
- Step (2.iii): according to the results obtained, and particularly the attack scenarios, security experts decide on the appropriate security measures to deploy in order to enhance the system security, which are likely to decrease the probability of the undesirable event;

  In steps (2.ii) and (2.iii), safety and security experts should make back-and-forth exchanges in order to identify potential Safety-Security Interactions (labeled SSI in Figure 26). The goal is to examine, *a priori*, whether safety and security measures are synergetic or conflicting (cf. Section 3.2.2 for examples on SSI).

- Step (2.iv): in this step, the initial system architecture is modified according to the measures defined by safety and security experts in steps (2.ii) and (2.iii);

The new system architecture can be again processed with S-cube in order to evaluate the impact of the modifications and the new results of risk assessment in Ph.2.

SSI can be identified, *a posteriori*, while analyzing results in step (2.i) thanks to the qualitative and quantitative risk analysis produced in Ph.2 (cf. examples in Sections 5.1.2, 5.2.4 and 5.3.2).

The third phase labeled Ph.3 in Figure 26, consists in different stakeholders feeding the S-cube KB with new inputs. The system designers/experts can add new components or refine the detail of the modeling if required for a given system. These changes are likely to occur for new systems that change often their architectures (e.g. cars, airplanes). The safety and security experts can refine the qualitative data (e.g. new attacks) and quantitative data using data available from feedback of experience. Ph.3 can also be powered by the results analysis in step (2.i).

The third phase supports the evolutions of the S-cube KB. This knowledge base can be extended with new data, improved and updated with feedback of experience.

We address in the next sections the process of building the S-cube KB and the different notions modeled.

4.3 The S-cube Knowledge Base

We explain in this section the rationale used in building the S-cube KB, Ph.0 in Figure 26, and the main notions modeled.

4.3.1 Rationale

We have chosen to model in the S-cube KB the following aspects related to industrial architectures:

- Enterprise levels: we consider the different levels of the enterprise architecture (cf. § 4.3.1.1);
- Network zones: in each level, we model the existing network zones (cf. § 4.3.1.2);
- Hardware/software: in each network zone, we model the connected components with the distinction between hardware and software (cf. § 4.3.1.3);
- Data flows: we model the data flows exchanged between the different software (cf. § 4.3.1.4).

We detail in the following how each aspect has been addressed in the S-cube approach.

4.3.1.1 Modeling the different enterprise levels

We addressed in a first instance the different levels of the enterprise architecture in the S-cube KB. This latter was initially conceived according to the PERA decomposition given in Section 1.2.1. The five enterprise architecture levels were reviewed and remodeled in a way that guarantees the generic aspect of the knowledge base, and the specificities of each level have been taken into consideration.

The S-cube KB does not model the physical process *per se* (level 0 in the PERA decomposition cf. Section 1.2.1) (e.g., power flow, heat propagation) but aims at modeling risks having impacts on the physical process (safety issues). As mentioned in Section 4.1.2, it can be coupled with other knowledge bases describing a specific industrial domain, for a better visibility on the physical impacts of attack and failure scenarios.

S-cube models the four other enterprise levels:

- **The field level**: (corresponds to the level 1 in the PERA decomposition) comprises devices that are close to the industrial process. These devices can be either sensors or actuators. Sensors are devices used to measure physical quantities like pressure, speed, temperature, etc. Actuators are devices that act directly on the physical process e.g., valves, pumps, breakers;

Level 2 in the PERA methodology is split into the process and the supervision levels:

- **The process level**: comprises automation devices that enable to monitor and control the industrial process. In this level, we find typically Programmable Logic Controllers (PLC) and Remote Telemetry Units (RTU);
- **The supervision level**: comprises SCADA servers and remote supervision devices that enable to have a global view and control the process level;
- **The IT level**: comprises machines integrating information technologies. For traditional industrial architectures, the IT level is associated to the levels 3 and 4 of the PERA decomposition. For modern control systems, information technologies are also integrated in control devices. In the S-cube KB, at first we considered that SCADA servers inherit the characteristics of the IT level, e.g., SCADA servers are installed on classical machines using advanced information technologies. Later, we extended the IT level to process controllers, as modern automata include also information technologies.

This decomposition is based on the functional specificities of each level with respect to control. Each level can consist of one or many networks.

4.3.1.2 Modeling the network zones

We chose to model, in a second instance, the different network zones at each level of the system architecture. A network zone, in S-cube KB, models a set of machines that are allowed to exchange information between one another using either a wired or a wireless communication technology. Examples of network zones could be the field network that connects sensors and actuators to process controllers and carries their data exchange.

4.3.1.3 Modeling the hardware/software system components

The functional architecture, described in § 4.3.1.1, does not necessarily map the physical architecture of the system. For this purpose we have chosen to differentiate in the S-cube KB between software and hardware components. Hardware (called later physical) components correspond to the physical architecture of the system while software components explicit a functional viewpoint of the system.

We model, with the S-cube KB, the different physical machines (hardware) connected to each network zone. We associate next the physical machines with the services (software) running on them. This distinction between software and hardware allows to have an appropriate level of detail for which failures and local attacks like physical access are associated with the physical machines and remote cyber-attacks exploiting vulnerabilities are associated to the software component which houses a specific vulnerability.

We detail in the next section how the S-cube KB includes modeling data flows and especially those used for control, and give details on the specificities of SCADA based control systems.

4.3.1.4 Modeling control data flows

Finally, we included in the S-cube KB, the modeling of data flows between the software components in the system architecture. Acquisition, control and supervision being the fundamental functionalities of a SCADA system, we address more specifically control data flows.

After defining the different levels of the system architecture in § 4.3.1.1 we started modeling the typical components of each level and their contribution to the control dataflow. Figure 27 summarizes the main control components and data flows. This figure was inspired from the control loop used by Leveson [116] in the Step 2 of the STPA method presented in § 2.3.2.3.

Figure 27 illustrates the typical components of each system level:

- At the field level, sensors measure physical quantities of the industrial process and send measures to the process controller. Actuators receive instructions from this latter and act accordingly on the industrial process;
- At the process level, the process controller (e.g., PLC, RTU) receives measures from the sensors, processes data and sends instructions to actuators, if necessary. On the other hand, it sends feedback on the process status to the supervision station and potentially receives instructions from it. In some architectures, process controllers can exchange orders/feedback between one another;
- At the supervision level, the operator station receives feedback from different process controllers, providing a centralized view of the physical process, and sends back instructions;
- The IT level models initially systems that enable the optimization and management of the business process. Such systems are not supposed to have a direct impact on the control process. Yet, as modern controllers and SCADA servers integrate information and communication

technologies (e.g., ftp servers, TCP/IP based communications), we make both the supervision and process levels inherit the IT characteristics.

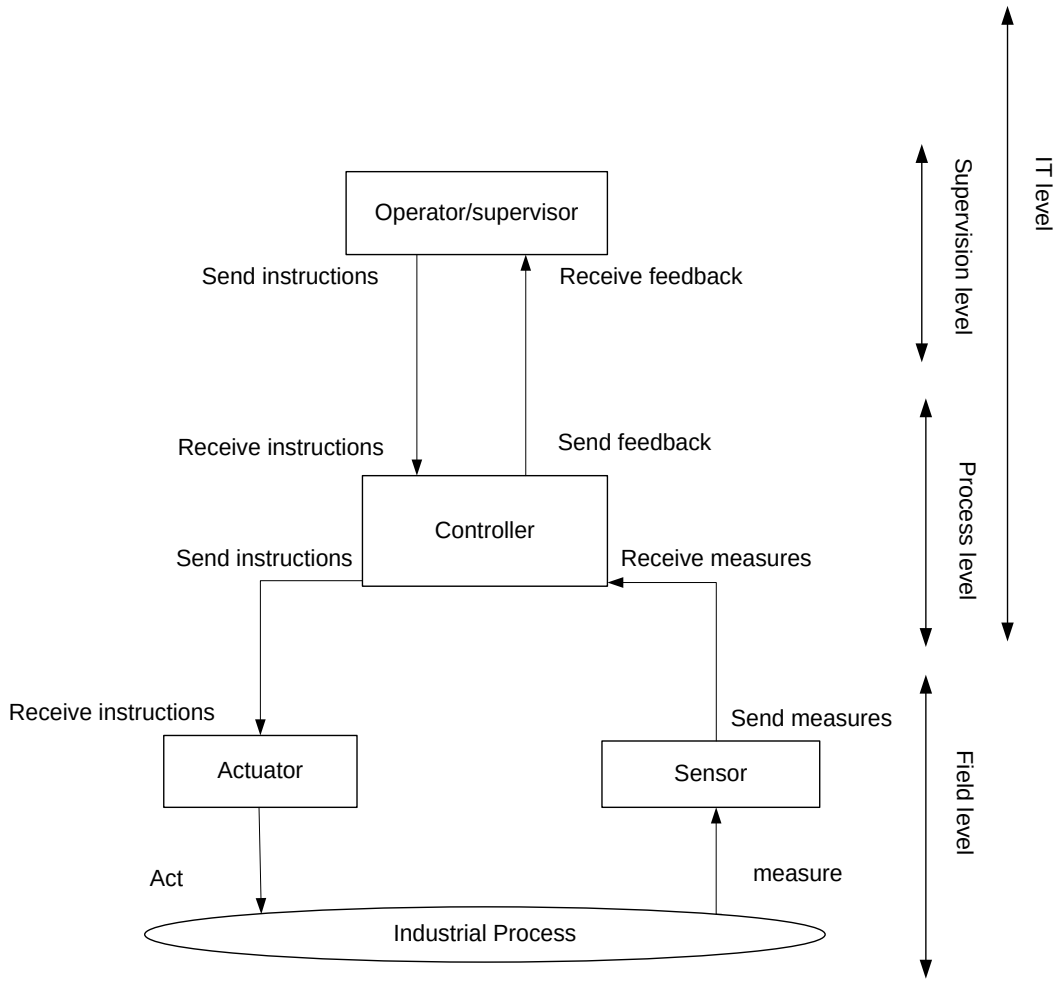


Figure 27: Control components and data flows

As illustrated in Figure 27, we distinguish between two types of control data[12]:

- Instructions: this data is sent by system components having a control functionality (such as process controller, supervisor, etc.) to field devices in order to execute an order;
- Feedback: this data is sent by field devices to acquisition, control and supervision components in order to report a status of the system, it can be either a measure or an alarm.

In addition to this level of detail, the S-cube KB models the data flow direction with respect to a given component; whether it is an input flow (data received) or an output flow (data sent). We summarize in Table 18 the main control data flows, modeled in the S-cube KB, associated to each component of the field, process and supervision levels.

We discussed in Section 1.2.2 the specificities and challenges of ICS in terms of time criticality and stressed the importance of data availability and integrity of control data flows. Indeed, control data flows should be available and unaltered in order to ensure the normal operation of the industrial process.

[12] This distinction was added in the KB for more accuracy on attacks impacting data flows in the process and field levels.

Otherwise, data alteration or unavailability can result into safety-related issues. In the S-cube KB, we study and propagate the effects of attacks and failures on the data flows integrity and availability. For example, a jamming attack on a wireless network would lead to unavailability of all data flows carried by this network.

System level	Component	Control data flow		
		In/Out	**Type**	**Description**
Field level	Sensor	in	_	_
		out	Feedback	sends measures to process controller
	Actuator	in	Instruction	receives instructions from process controller
		out	_	_
Process level	Process controller	in	Feedback	receives feedback from sensors/other process controllers
			Instruction	Receives instructions from operator/supervisor
		out	Feedback	Sends feedback to operator/supervisor or to other process controllers
			Instruction	Sends instructions to actuators/other process controllers
Supervision level	SCADA server/master	in	Feedback	Receives feedback from process controllers
			Instruction	Receives instructions from another SCADA server
		out	Feedback	Sends feedback to another SCADA server
			Instruction	Sends instructions to process controllers

Table 18: The control data flows modeled in the S-cube KB

We show in the next section how the different aspects described above have been aggregated into the S-cube KB by explaining the metamodel used to build this KB.

4.3.2 Metamodel

The S-cube metamodel, depicted in Figure 28, gives an overview on the hierarchy of classes modeled in the S-cube KB. It models the typical components of digital industrial architectures. Each class is represented with a "box" and models a system element template. The S-cube KB adopts the Figaro modeling language (cf. Section 4.6.1). Being object oriented, Figaro allows with the inheritance mechanism to structure knowledge and build progressively the metamodel. This latter can equally be extended in order to refine details about the system.

Each class of the metamodel is associated with its attributes as well as the attacks and failure modes likely to happen on it. Attributes are represented in Figure 28 by small horizontal rectangles. The type

of these attributes is put into brackets or braces for enumerated types. The attributes for which the type is not mentioned are Boolean (can have either the value True or False). The gear wheels icons model the dynamic behavior of the attack steps and failure modes associated with each class. The individual description of each class and the details on the metamodel implementation will be addressed in Annex 2.

We explain in the following paragraphs the main steps followed for building the knowledge base. We stress the key modeling elements of the S-cube metamodel and the assumptions made in order to have the appropriate level of detail. The classes modeled in the knowledge base and the associated attributes are put in *Italic* font.

We first model, with the generic class *component*, a system component which can fail accidentally or be compromised by an attacker. Accidental failures can be repaired by maintenance actions (cf. Section 4.4.1). The classes *network zone* and *physical component* inherit the characteristics of the mother class *component*, and model respectively a network zone (cf. § 4.3.1.2) and a physical component (cf. § 4.3.1.3). The physical component models a machine (hardware) connected to a network zone and that hosts one or many *software components*. Identical physical components that can fail simultaneously due to a common cause are associated with the same *CCF_group* (cf. § 4.4.1.2).

Following the system decomposition presented in § 4.3.1.1, we make the distinction in the S-cube KB between field system components, process system components, supervision system components and finally IT system components. They model generically the physical machines of each system level (cf. § 4.3.1.1). Actuators and Sensors are field system components, while process *controller* (e.g., Programmable Logical Controller) stands among process system components. An *IT system component* models a physical machine integrating advanced ICT typically running an *Operating System* and hosting *software components* from the IT domain. The *process controller* inherits from the *IT system component* class as modern controllers can also run operating systems and services imported from the IT domain.

Furthermore, we model the following software components:

- a *sensor software component* models the software capturing and reporting the physical measures;
- an *actuator software component* models the software receiving and executing the process controller instructions;
- a *process controller software component* models the software receiving and processing sensors measures, and sending orders to actuators;
- a *scada server software component* models the software supervising the process controllers, through receiving feedback and sending instructions;
- an *IT software component* models a software from the IT domain and not directly used for control purposes (e.g., ftp client, http server).

Software components exchange *data flows* (cf. § 4.3.1.4). With S-cube, the user models graphically only the legitimate data flows allowed by firewalls. The firewalling functionality is enabled or disabled by the *gateway* binding two or many networks (cf. CLASS Gateway in Annex 2 for more details).

An *IT software component* can host one or many vulnerabilities. Each vulnerability has one or many consequences among the following: privilege escalation, confidentiality loss, integrity loss or denial of service.

A vulnerability can be associated to a software component or to a physical machine. In the last case, the vulnerability models a bad machine configuration (e.g., system files not write-protected) which is assumed to allow privilege escalation when exploited by an attacker.

We assume, in the S-cube KB, that an IT machine is said to be compromised if an attacker manages to have root privileges on it. If it is the case, he can compromise all software components running on this machine. This assumption is quite credible as compromising one software on a machine does not allow the attacker to compromise all other services unless he/she succeeds in obtaining root privileges.

For IT level networks such as the corporate network, we are interested in the attack propagation (multistage multi-hop) between different IT level machines until reaching some component having a control action on the process. When reaching the control network, we are rather interested in data integrity/availability as the modification or unavailability is generally the main reason leading to undesirable events.

Access Control is modeled by associating an authentication mechanism to a machine (e.g., a login/password is required in order to log into the OS), to a network (e.g., WEP/WPA2 authentication) or to an application (e.g., ftp server needs to authenticate with the ftp server in order to read/write files).

We give in the following section the taxonomy of the different attack vectors embodied in the S-cube KB.

4.3.3 Taxonomy of attacks

Rather than following never-ending vulnerabilities and trying to patch each of them, security experts ought to deal with security at a higher level which would cover all vulnerabilities already known and unrevealed ones. The S-cube KB has been built upon a taxonomy of attack vectors that allows to reason about attacks at a higher level rather than a simple list of vulnerabilities, which guarantees the coverage of all types of attacks. The hierarchical thinking methodology was adopted in building the knowledge base; it consists in starting at first from a high level of abstraction and refining progressively the detail levels. The fundamental mechanism of abstraction allows dealing with the complexity of systems and the myriad of vulnerabilities they are subject to.

Our experience with BDMP applied on the industrial use cases [144] like the example of pipeline (cf. Section 3.2.2) revealed some patterns of the typical attacks on the control and field levels. The attack taxonomy in the S-cube KB was inspired from these patterns, but also from other existing DSLs like CySeMoL, MulVAL (cf. Section 4.1.1) and the Ethical Hacking and countermeasures Courseware (CEH) [145].

The CEH courseware [145] states the five following steps required for a successful attack:

1. Foot-printing and reconnaissance;
2. Scanning;
3. Gaining access;
4. Maintaining access;
5. Clearing track.

We map these steps with what has been modeled in the S-cube KB. The foot-printing and reconnaissance phase can be modeled in two ways: either by the attack step “preparing for the attack” associated with the *attacker* template, or by the attack step “*access*” associated with a physical component in the system. For this second case, the rate of this attack step combines together the frequency of attack occurrence and the time needed for the attacker to collect information about the target network. The scanning step (Step 2) is included in the different paths used by the attacker to access the system. The different ways for the attacker to gain access to the system (Step 3) are depicted in Figure 29. Steps 4 & 5 are not relevant in our context, as we are interested in successful attacks leading to safety issues; these steps are consequently not modeled by S-cube KB.

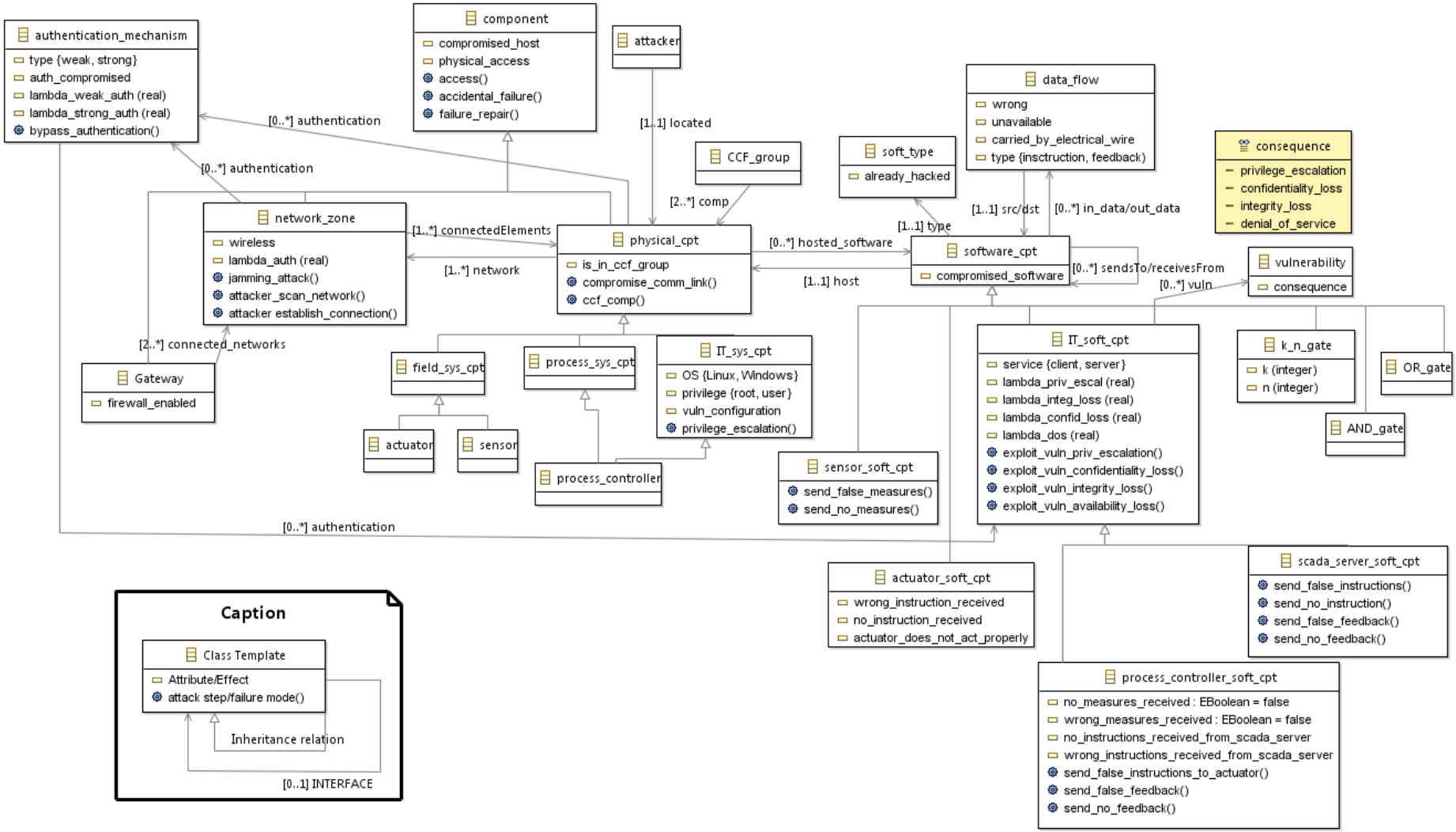


Figure 28: The S-cube metamodel

The taxonomy of attack vectors given in Figure 29 has been used in the S-cube KB. It addresses the different entry points used by the attacker in order to access the system. We believe that this latter should have initially some sort of physical access, whether local or remote, to a machine or a network zone related to the system architecture in order to try some attack scenario. We considered the following entry points (EP) from which an attacker can gain access to the system:

- EP1: physical access to the network. If the network is wired (e.g., Ethernet-based), and the attacker has physical access, he can plug into the network and manage to connect to the switch/hub via the wired link (e.g., Ethernet cable). If the network is wireless, the attacker should be able to capture the network traffic; which may require that he physically moves next to the access point. If the network employs an authentication mechanism, the attacker has to additionally bypass authentication in order to reach other machines connected to the same network. The network is said to be compromised;

- EP2: physical access to a machine connected to the network. If the attacker has physical access to a machine connected to the network and he manages to bypass authentication, if it is employed (by the machine OS and/or by the network), the machine is said to be compromised. If a network hosts a compromised machine it is also said to be compromised, which means that the attacker can reach (i.e. send packets to) any other machine connected to the same network;

- EP3: the network is remotely reachable via another compromised network. If the network is connected to a gateway which does not enable firewalling and which is connected to another compromised network, the attacker can then scan the network in order to identify live systems and open ports (e.g., ICMP scanning). He can then either:

    - EP 3.1: access via a vulnerable service. If there is vulnerable software which is reachable via compromised software communicating with it or which is located on a compromised machine or a compromised network zone, then the attacker can exploit this vulnerability in order to penetrate the system;

    - EP 3.2: access via a vulnerable machine. If a machine is not correctly configured or is running a vulnerable OS and it is reachable by the attacker (it is located on a compromised network or running a compromised software or the attacker can physically access the machine) then the attacker can exploit this vulnerability in order to gain root privileges.

  The network is said to be compromised if the attacker can reach any machine connected to the network zone. We assume that if the attacker can access the network (through the vectors above listed), then he can reach any machine connected to the same network.

We gave in this section the different entry points that can be used by the attacker to access the system. The attack step “access” associated to a physical component of the system (which can be a machine of a network zone) is the initiating vector in attack scenarios generated by the quantitative analysis. The remaining steps describe how the attack propagates given the system configuration.

We detail in the next two sections the qualitative and quantitative aspects included in the S-cube KB.

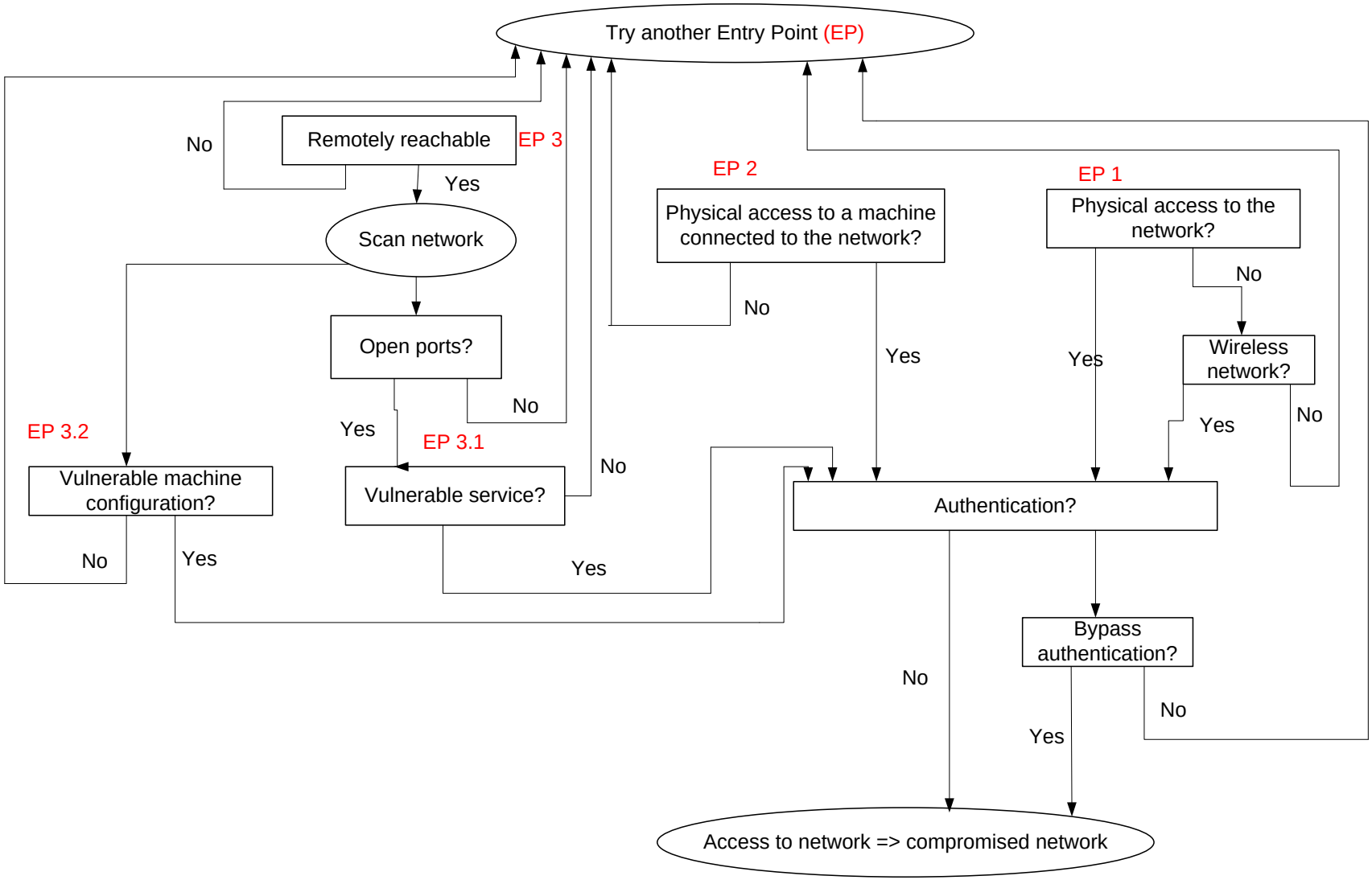


Figure 29: Taxonomy of attack vectors

4.4 Qualitative aspects in the S-cube KB

The S-cube approach has advantages both for building system models and processing them. First the system architecture is modeled using the templates corresponding to classes defined in the S-cube KB (cf. metamodel in § 4.3.2). This model can be either graphical or textual. Next the KB is instantiated on the system model and the resulting instantiation is processed with quantification tools which yields a qualitative and quantitative analysis.

The qualitative part of the analysis consists in generating the attack and failure scenarios likely to happen on the system model and that can lead to an undesirable event initially set by the user. The quantitative part, depicted in Section 4.5, allows sorting these scenarios according to their decreasing probabilities and gives an estimation of the undesirable event probability.

We give in the remainder of this section the types of failures and attack steps modeled in the S-cube KB. The risk scenarios output by S-cube are built from these generic failure modes and attack steps.

4.4.1 Failure modes and repair

We address in the S-cube KB accidental failures which can be either independent or dependent.

4.4.1.1 Failure in operation (independent events)

The S-cube KB models, for each system component, the accidental failure in operation which may occur randomly and independently from other components failures. This failure is associated with network zones and different physical machines (i.e. hardware failures). We assume in the current version of the knowledge base that software always functions in a deterministic and dependable way if not altered by third-parties (which excludes software bugs and crashes from our scope).

4.4.1.2 Common cause failures (dependent events)

A Common Cause Failure (CCF) is the failure of multiple components that result from a single cause, like for example a fire, a flood, an earthquake, etc. This cause is shared by a given set of components and can be related for instance to the design, the software, the environment, etc.

In safety-critical systems, redundancy is often introduced to enhance reliability. However, the intended effect may be reduced when components are subject to common cause failures. According to expert judgment, CCFs account for 1 to 10% of a component's failure rate [146]. It is consequently important to consider this kind of failures in the safety analysis in order not to underestimate the system reliability.

We model in the S-cube KB CCFs which we associate with physical components, and which represent dependent failures that may occur at the same time or within a short time interval, due to a shared cause.

4.4.1.3 Repair actions

The S-cube KB includes the modeling of maintenance actions aiming at repairing the accidental failures of the physical components. Once repaired, these latter resume their normal operation.

We present in the next sub-section the attack steps modeled in the S-cube KB. An attack scenario generated by S-cube will consist of one or several attack steps among the following.

4.4.2 Attack steps

In addition to failure modes previously described, we summarize in Table 19 the attack steps associated to each class as described in the metamodel in Figure 28. Classes describing physical components are in blue and classes corresponding to software components are in green.

Class	Attack steps / Failure modes
Component (generic class)	**Accidental failure:** models an accidental failure, in operation, of a given system component; **Failure repair:** models the repair of the accidental failure of the system component; **Access**: models physical access of the attacker to the component.
Network zone	**Accidental failure:** models an accidental failure of the network (e.g., failure of the switch); which results into its inability to ensure data transmission; **Access network:** models attacker's physical access to the network (cf. Section 4.3.3); **Jamming attack:** models a jamming attack on a wireless network; **Scan network:** models attacker scanning the network in order to identify live systems and open ports; **Establish connection:** models attacker establishing illegitimate connection with an open port; **Bypass authentication:** models attacker bypassing authentication to the network. We distinguish between weak and strong authentication.
Physical component (Generic)	**Accidental failure:** models an accidental failure in operation of a physical component (cf. § 4.4.1.1); **Common Cause Failure:** models the failure of the physical component due to a common cause (cf. § 4.4.1.2); **Access component:** models attacker accessing to a physical machine (cf. Section 4.3.3); **Compromise communication link** (Man In The Middle attack): models an attacker compromising the communication link between two machines; **Bypass authentication**: models attacker bypassing authentication to the OS of a physical machine; which can be either weak or strong.
IT system component	**Accidental failure:** models an accidental failure of a machine from the IT level; **Access (physical):** models the attacker's physical access to a machine from the IT level; **Privilege escalation:** models the attacker exploiting a bad configuration or a vulnerability related to OS of the machine for privilege escalation.
Process Controller	**Accidental failure:** models accidental failure of the process controller; **Access (physical):** models attacker's physical access to the process controller.
Sensor	**Accidental failure:** models accidental failure of a sensor; **Access (physical):** models attacker's physical access to a sensor.
Actuator	**Accidental failure**: models accidental failure of an actuator; **Access (physical):** models the attacker's physical access to an actuator.

Software component (Generic)	
IT software component	**Bypass authentication:** models attacker bypassing authentication required by an IT software component (e.g., ftp server). We distinguish between weak and strong authentication; **Exploit vuln priv escalation:** models attacker exploiting a vulnerability that results in privilege escalation; **Exploit vuln integrity loss**: models attacker exploiting a vulnerability that results in integrity loss; **Exploit vuln denial of service**: models attacker exploiting a vulnerability that results in denial of service; **Exploit vuln confidentiality loss**: models attacker exploiting a vulnerability that results in confidentiality loss.
SCADA server software component	**Send false instructions:** attacker falsifies instructions sent from SCADA server software; **Send no instructions:** attacker removes instructions sent from SCADA server software; **Send false feedback**: attacker falsifies feedback sent from SCADA server software; **Send no feedback**: attacker removes feedback sent from SCADA server software.
Process controller software component	**Send false instructions**: attacker falsifies instructions sent from process controller software; **Send no instructions:** attacker removes instructions sent from process controller software; **Send false feedback:** attacker falsifies feedback sent from process controller; **Send no feedback**: attacker removes feedback sent from process controller software.
Sensor software component	**Send false measures:** attacker falsifies measures sent from sensor; **Send no measures:** attacker removes measures sent from sensor.

Table 19: Failure modes and attack steps modeled in the S-cube KB

The attack steps modeled, so far, in the S-cube KB have been discussed with our security engineers. They are consistent with the level of detail at which we have decided to stop. This list is not, in fact, exhaustive; the knowledge base can be further extended with other "categories" of attacks and existing attack steps can be decomposed into more detailed attack steps.

We explain in the next section how accidental and malicious scenarios are generated (Ph.2 in Figure 26) from the system architecture and the S-cube KB.

4.4.3 Attack and failure scenarios generation

After the system architecture is described (Ph.1 in Figure 26), the S-cube KB is instantiated on it. This instantiation generates a textual model, which constitutes a virtual definition of the state space of the

system (all the states in which the system can be). This state space is defined locally, by the list of possible transitions from any state and the states they lead to.

The textual model can be explored in two ways:

- Using a path-based exploration algorithm, the state space is explored step by step. Starting from the initial state, we explore the tree of all possible paths in a depth left first manner. The exploration of one path is terminated if one of the following cases is reached: the targeted state, an absorbing state or one of the truncating criteria[13]. The principle of this algorithm is illustrated in Figure 30. If the explored state graph is in fact a Continuous Time Markov Chain (CTMC) then probabilities calculated analytically can be associated to sequences; this gives on one hand relevant criteria to eliminate most sequences, which makes the exploration tractable, and on the other hand an estimation of the probability of reaching a target state before a given time.

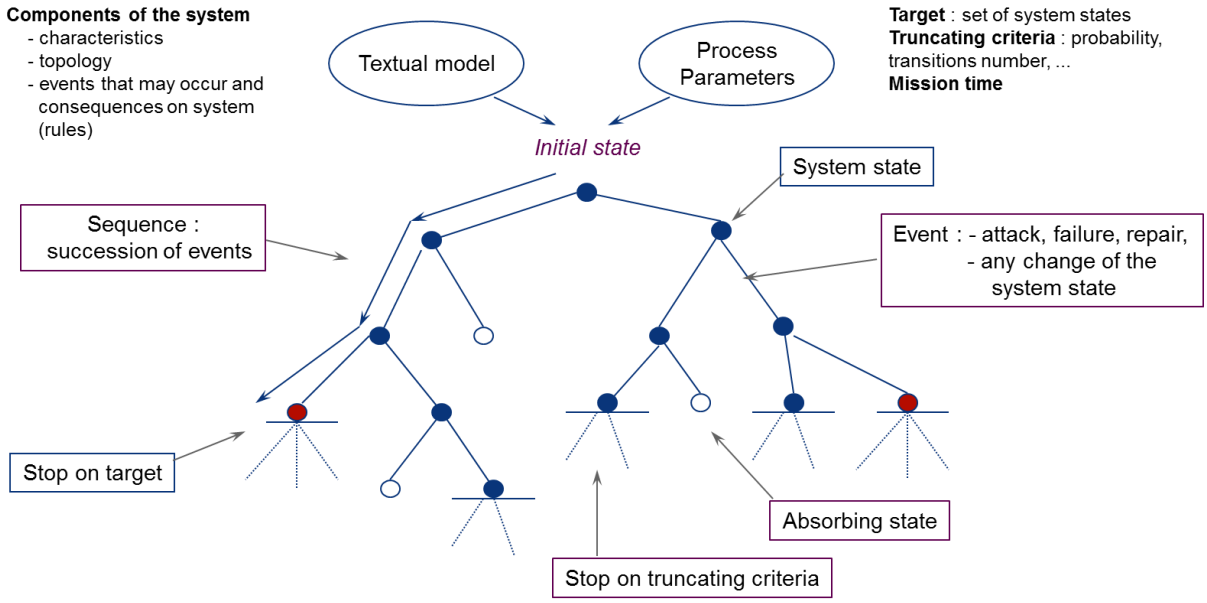


Figure 30: Sequence exploration principle

- Using the Monte Carlo method, which allows processing any problem having a probabilistic interpretation. Based on the law of large numbers, this method simulates many histories of the system using repeated random sampling. These histories yield independent and identically distributed realizations of a numerical variable of interest. By calculating the average of these realizations, one obtains an estimator of the mean of this variable's distribution. If the variable is Boolean, its mean value is equal to the probability that the variable takes the value 1. This is how the probability of the system being in a target state can be estimated.

The qualitative analysis, yielded in Ph.2 in Figure 26, generates exhaustively all the scenarios leading to the undesirable event specified. The number of scenarios can easily be huge and unmanageable especially for large systems. In order to be exploitable, the qualitative results should be associated with some quantitative parameters that enable sorting and prioritizing the most probable scenarios. We explain in the next section the quantitative aspects associated with the S-cube approach.

[13] The user can set truncation criteria which can be for example the minimum probability or the maximum number of transitions of the generated scenarios.

We give in the next section the hypotheses taken for the safety and security metrics associated to failure modes resp. attacks modeled in the S-cube KB and that are the basis of the quantitative results obtained in Ph.2 in Figure 26.

4.5 Quantitative aspects in the S-cube KB

We have already shown in Section 3.2.2 the advantages of building a common probabilistic model for safety and security. In a similar vein, S-cube offers a quantitative framework, based on probabilities, for assessing accidental and malicious risks. Each attack step and failure mode defined in the S-cube KB is associated with a security respectively safety metric as detailed later in this section.

4.5.1 Safety metrics

We explain the safety metrics associated to the failure modes described previously in Section 4.4.1.

4.5.1.1 Independent accidental failures

In dependability analysis, the system (or component) reliability corresponds to its "ability to perform a required function, under given environmental and operational conditions and for a stated period of time". The reliability expression as a function of time (t) is given by the following expression (1):

$$R(t) = \Pr(T > t) = 1 - \Pr(T \leq t) = 1 - F(t); \qquad (1)$$

where T is a random variable corresponding to the Time To Failure (TTF) and F(t) is the cumulative distribution function of T.

The failure rate is given by the expression (2):

$$\lambda(t) = \lim_{dt \to 0} \frac{\Pr(T<t+dt \,/\, T>t)}{dt}; \qquad (2)$$

Intuitively, λ(t)dt represents the probability that the system or component fails between t and t+dt, knowing that it survived until t.

The exponential distribution is commonly used and validated in traditional reliability analyses for the probability distribution of the TTF. In this case, the failure rate λ of the system (or component) is constant and equal to the parameter of the exponential distribution. This assumption (constant failure rate) is valid in the useful part of the system lifetime which excludes the beginning period (premature failure) and the ending period (aging phenomenon). The Mean Time To Failure MTTF = $E\{T\} = \int_0^\infty R(t)dt$) of a system or component is equal, in this instance, to the inverse of the failure rate (λ).

In the S-cube KB, we adopt the exponential approximation for the time to failure of a system component in operation. The quantitative metrics used in this case correspond to failure rates of components.

Data related to the failure rates can be obtained from the experience feedback on the system components failures or from the manufacturer's documentation. Experimental data may take a lot of time before being available (the range of MTTF generally is from years to decades), especially for systems with a long service life.

4.5.1.2 Common Cause Failures

The Common Cause Failures (CCFs) have been addressed in the probabilistic risk analysis with different models, for instance the Atwood model [147] and the α factor model [148]. These models use feedback of experience data in order to quantify the probabilities of events causing the failure of a specific group of identical of components. The transition between these models and the CCF model used in static (i.e. essentially made with fault trees) analyses and called the Basic Parametric Model (BPM) [149] is then

possible. This latter is used to evaluate the probability of the different combinations of components failures within the same CCF group.

In the S-cube KB, we adopted the dynamic generalization of the BPM (DBPM) described in [104]. This latter uses the α-factor model to exploit the feedback of experience failure data. The parameters of this model, called α-factors, are next transformed according to formulae (3) into $\beta_{k/m}$ factors used by the DBPM to evaluate the occurrence rates $\lambda_{k/m}$ of an event leading to the failure of k components included in a group of m components.

$$\beta_{k/m} = \frac{m}{\binom{m}{k}} \frac{\alpha_{k/m}}{\sum_{k=1}^{m} k\, \alpha_{k/m}} \qquad (3)$$

A CCF group is a set of components that can be subject to simultaneous failures due to a common cause (cf. § 4.4.1.2). The number of combinations of components common cause failures (CCF of k among m components, k ranging from 1 to m), can be quickly significant with large groups of components. For simplification reasons, we have chosen to model in the S-cube KB only groups of two or three components. Inspired from Donat *et al.* [150] where the CCFs are integrated into the BDMP formalism, the modeling of CCF in S-cube is depicted in Figure 31.

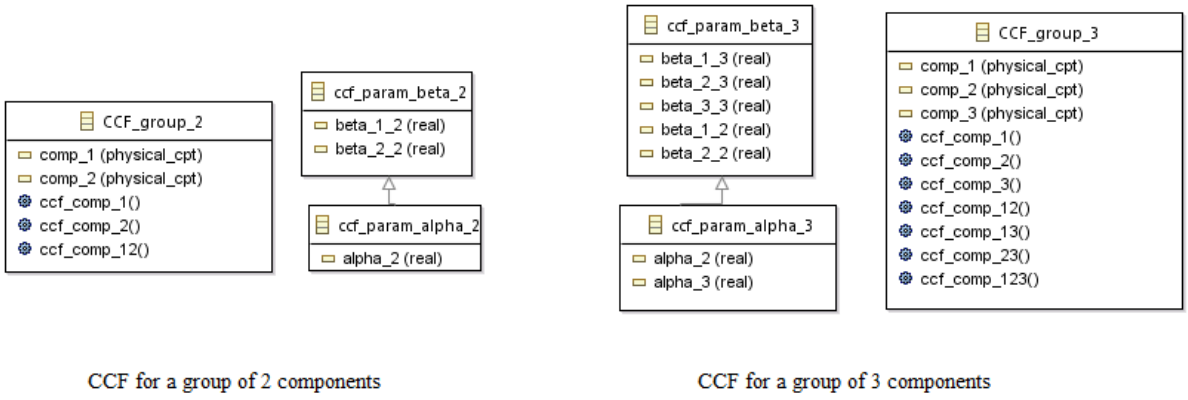


Figure 31: Common cause failures metamodel

CCF_group_2 models a set of two identical physical components that can be subject to one of the following failures due to a common cause:

- Failure of one component (1 or 2) out of two (which corresponds in Figure 31 to failures: *ccf_comp_1* and *ccf_comp_2*);
- Simultaneous failure of the two components 1 and 2 (which corresponds in Figure 31 to the failure: *ccf_comp_12*).

Similarly, CCF_group_3 models a set of three identical physical components that can be subject to one of the following failures due to a common cause:

- Failure of one component (1 or 2 or 3) out of three (which corresponds in Figure 31 to failures: *ccf_comp_1*, *ccf_comp_2* and *ccf_comp_3*);
- Simultaneous failure of two components (12 or 13 or 23) out of three (which corresponds in Figure 31 to failures: *ccf_comp_12*, *ccf_comp_13* and *ccf_comp_23*);
- Simultaneous failure of the three components 1, 2 and 3 (which corresponds in Figure 31 to the failure: *ccf_comp_123*).

ccf_param_alpha_2 and *ccf_param_beta_2* in Figure 31 define the parameters $\alpha_{k/m}$ resp. $\beta_{k/m}$ of the α-factor model resp. the DBPM related to CCF_group_2. The same thing is done with CCF_group_3 and the associated parameters *ccf_param_alpha_3* and *ccf_param_beta_3* in Figure 31.

The user has to specify the value of the parameter *alpha_2* in the case of a group of two components and two parameters *alpha_2* and *alpha_3* in the case of a group of three components. The β parameters values are deduced from the α parameters that can be obtained from failure feedback of experience data.

4.5.1.3 Assumptions for the S-cube KB

In the S-cube KB, the quantitative safety metrics were assigned based on estimations of our safety experts or based on existing studies from the literature. We summarize in Table 20 the default values, in the S-cube KB, assigned to the safety metrics and the references used for these estimations. These values can be overridden and customized by the user with the appropriate values associated with the system under study.

System element	Failure mode	Quantitative data source	1/MTTF or failure rate
Network zone	Accidental failure	Experts estimation	1e-5/h
Physical component	Accidental failure	Experts estimation	1e-5/h
CCF_group_2	Common cause failure	[150][146]	Alpha_2 = 0.05
CCF_group_3	Common cause failure		Alpha_2 = 0.1
			Alpha_3 = 0.05

Table 20: Quantitative safety metrics for S-Cube

We address in the next section the quantitative aspects related to security modeling in the S-cube KB.

4.5.2 Security metrics

We make the assumption that Times To Success (TTS) are, also, exponentially distributed. Hence, security metrics used in the S-cube KB are the success rates (mathematically defined by the same formula as for failure rates) of the attack steps described in Table 19. However, it is easier for experts to reason in terms of Mean Time To Success (MTTS) that are simply the inverse of success rates. In order to support the assumptions we have made for MTTS in S-cube, we give in the following a quick overview on existing work on quantitative evaluation of this security parameter then explain the way we parameterized our attack steps.

4.5.2.1 Related work

Unlike for failures, feedback on attacks is not easy to obtain. Industrials often refuse to communicate about their experience with attacks as this can infringe their image. Quantitative security data are consequently less available and can be subject to large uncertainties. Security metrics are, in addition, intimately linked to the attacker's profile and behavior which is hardly predictable.

Several works have been trying to quantify security metrics and estimate the likelihood of software vulnerabilities and exploits [151][152]. Based on empirical data collected from intrusion experiments, Jonsson *et al.* [153] inferred that during the standard attack phase, the intrusion process could be described by an exponential distribution using the variable attacker working time.

Holm [154] carried out a large-scale study of the time required to compromise a computer system. The author studied the Time To First Compromise (TTFC), which is the time measured from the deployment

until the first malware alarm, and found that the Pareto distribution is best fit. Results also show that the exponential is reasonably good at modeling intrusions that require less than 600 days; while 90% of all detected intrusions require 400 days or less.

The results of the study in [154] show that the log-normal and Pareto distributions provide the best fit for Time To Compromise (TTC) while the exponential, gamma and Weibull distribution are reasonably good at modeling intrusions which require less than 100 days. This study was done on a large enterprise that specializes in IT and has business in various locations all over the world, and consequently many computers; the results might only be valid for an enterprise with the same characteristics. There is no empirical evidence to justify that the Pareto and log-normal distributions might fit also for targeted attacks, the kind of attacks that happen in the industrial domain.

Holm [155] proposed a Bayesian network model that can be used to estimate the likelihood that a professional penetration tester is able to obtain knowledge on critical software vulnerabilities and exploits for these vulnerabilities under different conditions. This model requires input data from the user which are as both uncertain and subjective. Besides, the model yields Probability Distribution Functions (PDF) which are not parametric and cannot be fitted to some known PDF.

Dacier *et al.* [156] [157] propose a method of quantitative security evaluation in which the authors suggest the use of a Markovian model to represent an attack process.

McQueen *et al.* [158] proposes also a Markovian model for estimating the time to compromise (TTC) a computer system through exploiting a given vulnerability. The authors model the TTC, which they define as the measure of the effort expended by an attacker for a successful attack. They also assume that effort is expended uniformly, as a random process composed of three attacker sub-processes:

- Process 1: is when at least one vulnerability is known and the attacker has at least one exploit readily available. The probability that the attacker is in process 1 is given by equation (4):

$$P_1 = 1 - e^{-vm/k} \qquad (4)$$

  Where v is the number of vulnerabilities on the component of interest, m is the number of exploits readily available to the attacker, and k is the total number of vulnerabilities.

  Assuming that the attacker is familiar with at least one of the available vulnerabilities and has experience with at least one exploit associated with the known vulnerabilities, the authors estimate the time required for Process 1 with t_1=8 hours.

- Process 2: is when at least one vulnerability is known but the attacker does not have an exploit readily available. Since Process 1 and Process 2 are mutually exclusive, the probability that the attacker is in process 2 is given by equation (5):

$$P_2 = 1 - P_1 = e^{-vm/k} \qquad (5)$$

  The mean time needed to complete Process 2 is modeled as the expected value of the number of tries (ET) times 5.8 days: $t_2 = 5.8 * ET$; where ET is the expected number of tries;

- Process 3: is when the attacker identifies new vulnerabilities and exploits (zero-days). The time estimated for this process is given by equation (6):

$$t_3 = \left(\frac{V}{AM} - 0.5\right) . 30{,}42 + 5{,}8 \qquad (6)$$

  Where AM is the average number of the vulnerabilities for which an exploit can be found or created by the attacker;

Assuming that the three processes are mutually exclusive (Process 3 only applies if Processes 1 and 2 do not apply or are unsuccessful), the overall TTC is given by equation (7):

$$T = t_1.P_1 + t_2.(1 - P_1)(1 - u) + t_3\, u\, (1 - P_1) \qquad (7)$$

Where $= (1 - (\frac{AM}{V}))^{v}$: the probability that Process 2 is unsuccessful ($u=1$ if $V=0$).

We explain in § 4.5.2.3, how we used McQueen *et al.*'s [158] model in order to assess the MTTS for exploiting a vulnerability in a software system component.

4.5.2.2 The Common Vulnerability Scoring System

The NIST [159] introduced the Common Vulnerability Scoring System (CVSS) an open framework for scoring IT vulnerabilities based on three metric groups: base, temporal, and environmental. Base metrics represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. Temporal metrics represent the characteristics of a vulnerability that change with time but not among user environments. Environmental metrics represent the characteristics of a vulnerability that are relevant and unique to a particular user's environment.

The CVSS score is first calculated according to an equation as a function of the base metrics. It can be next refined based on the temporal and environmental metrics. We only consider the base metrics below. The CVSS score ranges from 0 to 10; the higher it is, the more critical the vulnerability.

Base metrics are:

- Access Vector (AV): reflects how the vulnerability is exploited: locally, with adjacent network access or remotely. The more remote an attacker can be to the target of the attack, the greater the vulnerability score;
- Access Complexity (AC): measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system. The access complexity can be high, medium or low. The lower the required complexity, the higher the vulnerability score;
- Authentication (Au): measures the numbers of times an attacker must authenticate to access a target in order to exploit a vulnerability. This metric can take the following values: multiple, single or none. The fewer authentication instances are required, the higher the vulnerability score.
- Confidentiality Impact (C): measures the impact on confidentiality of a successfully exploited vulnerability. It can be complete, partial or none;
- Integrity Impact (I): measures the impact to integrity of a successfully exploited vulnerability. It can be complete, partial or none;
- Availability Impact (A): measures the impact to availability of a successfully exploited vulnerability. It can be complete, partial or none;

We explain in the next subsection how the CVSS base metrics and some studies from previous work have been exploited to assess the security metrics related to the attack steps modeled in the S-cube KB.

4.5.2.3 Assumptions for the S-cube KB

The S-cube KB includes modeling the CVSS base metrics as described below:

- (AV): the different access vectors are modeled as described in Figure 29;
- (AC): the access complexity is included in the MTTS assessment;

- (Au): authentication is modeled by associating an authentication mechanism to a service, a network or a host. We also model whether the authentication is weak or strong. The mean time to success of the attack step "bypassing authentication" is higher in case a strong authentication is in place;

- (C, I, A): the impact on confidentiality, integrity or availability is associated with each vulnerability, by security experts in step (1.iii) of the system description phase (Ph.1) in Figure 26. As explained in Section 1.2.2, confidentiality is not too important in the industrial context and could not lead to safety issues. Vulnerabilities having as consequences: "privilege escalation", "integrity loss" or "denial of service" are the most relevant when it comes to safety-related risks and their impact on data flows integrity and availability are propagated throughout the system model.

Other metrics that impact the MTTS are the attacker's profile (expert, intermediate, and beginner) and the resources (money) he/she is ready to invest into the attack. In order to meet safety requirements, we make, in the S-cube KB, the pessimistic assumption that the attacker is an expert and holds unlimited resources to achieve the attack.

In S-cube KB, we considered the two different random variables corresponding to TTS:

- TTS associated with the attack step *access* (*TTS_access*); which corresponds to the time until the system is accessed by an attacker. As discussed in Section 4.3.3, the attacker needs first to have some sort of access to the system in order to try some attack scenario (cf. Figure 29 on entry points). *MTTS_access* is the mean time required for an attacker to access the system;
- TTS associated with other attack steps given in Table 19; which corresponds to the time required for the attacker to achieve a given attack step;

We have chosen to use, in the S-cube KB, the exponential distribution to model the TTS for all the attack steps. We argue below this choice.

For the time after which the system is accessed by an attacker (*TTS_access*), the empirical results obtained by Holm in [154] (cf. § 4.5.2.1) show that the exponential distribution is relevant if *TTS_access*<600 days. Furthermore, the Grigelionis theorem given in [160] proves the relevance of the exponential distribution if we make the following assumptions:

- Each attack scenario can be approximated by a point renewal process $T^{n,i} = (T_k^{n,i})_{k\geq 0}$ $(1 \leq i \leq n)$; with an initial delay time $T_0^{n,i}$;
- All delay times $T_0^{n,i}$ are independent from one another; which means that attacks that may target the system are independent from one another;
- On the time scale, these attacks can superpose and each of the n processes (n is large because it corresponds to the number of potential attackers) has a small contribution;

Given these assumptions, the theorem stipulates that the superposition of the *n* independent renewal processes converge towards a Poisson point process. We can consequently infer that the random variable corresponding to the minimum of the delay times $T_0^{n,i}$, corresponding in our context to the *TTS_access*, follows an exponential distribution.

For the other attack steps, the use of the exponential distribution is not always approved for security assessment. However, we have chosen this assumption because it makes it possible to use Figseq for the qualitative analysis (Figeseq works only with markovian models).

As previously mentioned, the McQueen *et al.*'s [158] model (cf. § 4.5.2.1) has been used in order to estimate the MTTS associated to attack steps modeling the attacker exploiting a vulnerability of a

software component in the S-cube KB. In order to use the formula (7) of this model, we extracted statistical data[14] on vulnerabilities from the Common Vulnerabilities and Exposures (CVE) dictionary [161]. The k parameter in McQueen *et al.*'s model corresponds to the total number of vulnerabilities and the m parameter corresponds to the number of exploits publicly available.

As previously discussed, vulnerabilities are categorized in the S-cube KB, into three main categories, according to their consequences and impact on confidentiality (C), integrity (I) and availability (A) (cf. base metrics in § 4.5.2.2):

- Vulnerabilities resulting into privilege escalation (have an impact on C, I and A);
- Vulnerabilities resulting into integrity loss (have an impact on I);
- Vulnerabilities resulting into denial of service (have an impact on A).

We give, in Table 21, the data obtained from the CVE dictionary[15] [161] corresponding to k and m parameters for each type of vulnerability. We explain below how we proceeded to get this data.

Vulnerability type	**Number of total vulnerabilities (k)**	**Number of total exploits publicly available (m)**
Privilege escalation	3388	184
Integrity loss	2222	60
Denial of service	14791	654

Table 21: Statistical data on vulnerabilities sorted by type

For vulnerabilities resulting in privilege escalation (cf. first line of Table 21), data extracted correspond to the type "Gain privilege" in [161].

For vulnerabilities resulting in integrity loss (the second line of Table 21), we consider from [161] data of type "Gain information" and having a complete or partial impact on integrity. We consider particularly vulnerabilities having a CVSS score >5; given that for vulnerabilities with a CVSS <5 the impact on integrity is "None".

For vulnerabilities resulting in denial of service (the third line of Table 21), we took the data corresponding to "DoS" in [161].

We also made the assumptions given in Table 22 when using the McQueen *et al.*'s [158] model.

Assumption	**Rationale**
V = 1	In each attack step "exploit vulnerability", the attacker exploits just one vulnerability
AM/V = 1	We assume the attacker's skill level is "expert"
t_1= 1	The time needed for an expert to exploit a known vulnerability with an exploit readily available is one working day

[14] We took this data on the CVE dictionary in August 2015

[15] Data has been collected since 1999

ET=1	The expected number of tries is equal to one; i.e. the attacker tries to exploit a vulnerability just once and abandons in case of unsuccessful attempt

Table 22: Assumptions taken for MTTS evaluation using McQueen's model

Given these assumptions, the MTTS obtained using the equation (7) and data in Table 21 is approximately five days for all types of vulnerabilities ($1/MTTS \sim 0.01\ h^{-1}$).

For other kinds of attack steps modeled in the KB, the MTTS was estimated by our security experts. Table 23 gathers the sources used to estimate the quantitative data associated with the attack steps described in the S-cube KB.

System element	**Attack step**	**Quantitative data source**	**MTTS**
Physical component	Physical access	Depends on whether we model the frequency of the attack or the time needed to access the system	-
Network zone	Physical access	Same as for physical component	-
	Jamming attack	Security experts eval.	Jamming is almost instantaneous, however the attack requires special equipment, highly expensive, which reduces its frequency
	DoS attack		1 hour
	Scan network		1hour for IT networks 1 - 2 days for industrial networks
	Establish connection		1 day
IT system component	Exploit bad config. vulnerability		2 - 3 days
IT software component	Exploit priv. escal. vulnerability	Mc Queen *et al.* [158] + CVSS DB [161]	1 day if exploit available 5 days else
	Exploit integrity loss vulnerability	Mc Queen *et al.* [158] + CVSS DB [161]	1 day if exploit available 5 days else
	Exploit DoS vulnerability	Mc Queen *et al.* [158] + CVSS DB [161]	1 day if exploit available 5 days else
	Exploit confidentiality loss vulnerability	Mc Queen *et al.* [158] + CVSS DB [161]	1 day if exploit available 5 days else
Physical component Network zone IT Soft cpt	Bypass authentication (MTTS depends on whether authentication is weak or strong)	Security experts eval.	1 day if weak password 30 days if strong password 2 years if strong cryptography (e.g., SSL)

Table 23: Sources used for the estimation of MTTS for attack steps

The feasibility of jamming attacks on wireless networks has been discussed in [162], [163] but the quantitative results could not be used to estimate the MTTS of such an attack. The same holds for DoS and brute force attacks addressed by Sommestad *et al.* in [164].

We discuss in the next section the uncertainties related to this quantitative data associated with the attack steps and failure modes described in the S-cube KB.

4.5.3 Discussion

The results obtained in Ph.2 of Figure 26 are based on the qualitative and quantitative aspects in the S-cube KB previously described. These results are called also later, qualitative and quantitative analysis. The MTTS associated with the initiating "access" attack step (*MTTS_access*) can be parameterized in two different ways for the two following purposes, which can be complementary:

1) To assess, from scratch, a given architecture in order to pinpoint the different access paths privileged by the attacker and identify the most vulnerable components. In this case, *MTTS_access* is set with the inverse of the frequency of the attacks that target the specific kind of system architecture under study (such an information can be obtained in-house from security feedback e.g., log-files). With this kind of study, the qualitative results are the most interesting to analyze.

2) To quantify more precisely the probability of a successful attack scenario given that the attacker has started at t=0 to target the system. In this case, *MTTS_access* is set with the mean time needed for the attacker to have some physical or remote access to the system, assuming that the reconnaissance phase was already done. With this kind of study, more focus is given to the quantitative analysis. The qualitative and quantitative results provide the attack scenarios with a more accurate estimation of the time needed to complete each scenario.

In the second kind of study, using a joint model for assessing safety and security risks leads to results that promote generally attack scenarios. Indeed, the MTTFs used for failure modes are very large compared to security metrics. On the contrary, in the first kind of study, it is possible to have the same order of magnitude for both kinds of risk, thanks to the fact that the frequency of targeting a given industrial architecture is indeed low (fortunately); unfortunately, this metric is even more difficult to predict than other security metrics and highly subjective as it is related to human intention.

The quantitative results are aimed at providing operators of the control systems with a measure of the risk associated with potential attacks in order to effectively manage their resources. The results obtained should not be considered as definitive and accurate values. They are based on the assumptions taken for the different attacks modeled in the S-cube KB and related to the level of detail modeled.

We explain in the next section how the main notions of the S-cube KB above described have been implemented.

4.6 Implementation & Tool chain

We first show how the S-cube KB has been developed using the Figaro language. We present next the tool chain used by S-cube at different phases given in Figure 26.

4.6.1 The Figaro language

Figaro [165] is a general modeling language initially developed for reliability analysis. A quick comparison between Figaro and other safety DSLs was given in Section 4.1.2. We underline below the main specificities of Figaro language and how they have been used to implement the S-cube KB.

Figaro is an object oriented language and implements additionally some artificial intelligence notions. Although specific to safety, Figaro is general enough to be adaptable for other domains related to dependability and especially for security.

Figaro provides an appropriate formalism for developing knowledge bases with generic descriptions of components. It enables thanks to the inheritance mechanism to structure the knowledge and avoid information redundancy. Each system generic component is described with a class. A class can be compared to a mold which, when filled, gives an object having the shape of the mold and all its characteristics.

A class consists of two parts [166]:

- A purely static and declarative part: where one finds the name of the class, the class from which it inherits characteristics, the other classes with which it interacts, the constant characteristics and the state variables with their domains and initial values;
- A dynamic part: where the behavior of the class is described thanks to two kinds of rules: the occurrence rules and the interaction rules.
    - The occurrence rules: describe elementary events with the conditions governing how they are triggered and the associated probability distributions. If the conditions of the occurrence rule are satisfied, the event can occur:
        - Instantaneously: this is used in order to describe the choice between different instantaneous transitions; each transition is associated with a probability, and the sum of transitions probabilities appearing in a given rule must be equal to 1;
        - After a time that follows a given probability distribution: in this case the type of the distribution and its parameters are associated with the transition;
    - The interaction rules: aim to propagate the effects that are the immediate and certain consequences of an event (i.e., the firing of a transition of an occurrence rule) in the system.

The Figaro language is quite legible and can be easily associated with graphical representations. The Figaro language can be used either to define physical components (e.g., pumps, valves, heat-exchangers in a knowledge base dedicated to thermo-hydraulic systems), or to define abstract objects like the places and transitions of a Petri net.

We give below an excerpt of the Figaro description of the class that models a network zone, and show some of the keywords used:

```
CLASS network_zone KIND_OF component; (* declaration of a class modeling a network zone *)
   CONSTANT  (* declaration of the constants related to the class *)
        wireless  (* this Boolean is set to true in an object modeling a wireless network zone *)
        DOMAIN BOOLEAN
        DEFAULT FALSE;
   ATTRIBUTE (* declaration of the attributes related to the class *)
        lambda_auth (* rate of the attack “bypass authentication” *)
        DOMAIN REAL
        DEFAULT 0.01;
   EFFECT  (* declaration of the effects related to the class *)
        network_access
        LABEL "attacker has access on network %OBJECT";
   INTERFACE (* declaration of the interfaces related to the class *)
```

```
        connectedElements (* this name will be used in the rules to designate a set of physical_cpt *)
        KIND  physical_cpt
        CARDINAL 1 TO INFINITY
        LABEL "sys_components connected to the network_zone";
        authentication (* the network_zone uses or not an authentication mechanism *)
        KIND  authentication_mechanism
        CARDINAL 0 TO 1
        LABEL "authentication to the network_zone";
    FAILURE (* declaration of the failure modes and attack steps related to the class *)
    jamming_attack
        LABEL "network jamming attack";
    OCCURRENCE (* description of the dynamic behavior of the failure modes and attack steps *)
    wireless_network_jamming_attack (* name of the rule – used only for traceability in debug tools *)
        IF (wireless)
        MAY_OCCUR
        FAULT jamming_attack
        DIST EXP(0.000001);
    INTERACTION (* propagation of the instantaneous effects of the failure modes and attack steps *)
    network_unavailable (* propagation of the unavailability of data in case of jamming attack*)
        IF failure OR jamming_attack OR denial_of_service_attack
        THEN FOR_ALL x A connectedElements
        DO (
                FOR_ALL y A hosted_software(x) DO (
                                        FOR_ALL z A out_data(y) DO unavailable(z) ) );
```

In Table 24, we detail the meaning of some keywords used in the Figaro language. A complete documentation on the language and its syntax is provided in [167][168].

Keyword	**Signification**
CLASS	Declares a new Class
KIND_OF	Inheritance relationship with other classes
CONSTANT	Declares the constants related to the class
ATTRIBUTE	Declares the attributes related to the class. Contrarily to constants, attributes can change value by the execution of the occurrence/interaction rules
EFFECT	Declares the effects related to the class. An effect is a Boolean that is used to propagate the effects of the attack steps and failure modes. The value of this Boolean is reset to FALSE, then updated each time the interaction rules are executed
INTERFACE	Declares the interfaces related to the class. An interface describes a relationship between the class and other classes
FAILURE	Declares the failure modes and attack steps related to the class
OCCURRENCE	Describes the dynamic behavior of the failure modes and attack steps through associating them with the appropriate probability distribution and the associated rate

INTERACTION	Propagates the instantaneous effects among which the ones resulting from the realization of the attack steps and failure modes

Table 24: Meaning of some keywords of the Figaro language

There are two levels of the Figaro language: order 0 and order 1. The order 1 Figaro, so far represented, is used to write knowledge bases (Ph.0 in Figure 26). Using a variety of keywords including quantifiers (e.g., IT_EXISTS, FOR_ALL), it is a highly expressive and natural language. The order 0 Figaro is the language in which is generated the textual model resulting from the instantiation of the KB on the graphical model. This language includes few keywords which makes it simple and efficiently executable by machines for further processing.

The choice of the Figaro modeling language led us to use the KB3 workbench, described in Section 3.1.4, that enables to build Figaro-based models and process them. We give in the following section details about the tool chain used for the S-cube approach implementation.

4.6.2 Tool chain

In order to explain how the principles of the S-cube approach have been implemented, we reproduce the

Figure 25: The S-cube approach principle

given in Section 4.2, to which we add how each aspect has been implemented in blue italic text. The result is depicted in Figure 32.

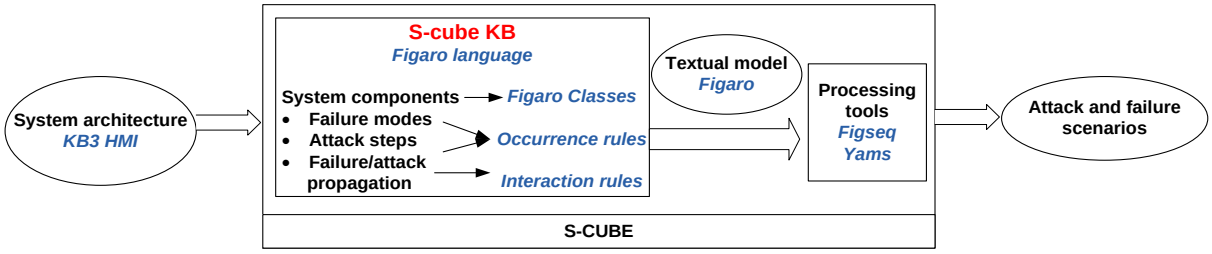


Figure 32: The S-cube tool chain

During the S-cube KB development (Ph.0 in Figure 26), the Figaro classes have been used to describe the generic components comprised in industrial information architectures and their main characteristics as already introduced in Section 4.3.2. For each class, occurrence rules are used to model security (attacks) and safety (failures) events that may happen to each system component (cf. Section 4.4). These rules contain also the probability distribution of the time after which the event will happen (cf. Section 4.5). For each class, the interaction rules model the propagation of the instantaneous effects within the whole system architecture, for instance how the compromising or the failure of one component impacts other system elements (e.g., data no more available). The graphical elements, corresponding to instances of classes defined in the S-cube KB, are specified using the KB3 tool (cf. § 3.1.4.1).

First, the system architecture is input using the KB3 HMI (Ph.1 in Figure 26) where the graphical elements previously defined are loaded into a palette. Automatic verification of the graphical model is done and the correctness and coherence of the model input is checked. For instance, if some graphical element cannot accept a certain type of link the user is notified with an error message. The system model can also be described textually but this requires a basic knowledge of the Figaro language and the S-cube KB.

The formal definition of the Figaro language is given in [169] and allows to detect inconsistencies or ensure the consistency of knowledge bases as they are built. The S-cube KB, respects a set of rules given in [169] that ensure the consistency of this KB. This implies that all the models built with the S-cube

KB, are coherent from their very construction, and can embed no inconsistencies or undesirable properties. In particular for any model built using the S-cube KB, the following properties are satisfied:

- The space of states is finite as all the variables defined have a finite domain;
- The model is not totally repairable, as detection and reaction measures have not so far been modeled. This implies that the space of states includes some states from which the initial state can no longer be reached;
- Monotonous inference: the EFFECTs are only set to true in the interaction rules. Given that all effects are initialized to false each time the interaction rules are executed, this guarantees that whatever the execution order of the interaction rules, the inference converges towards the same state.

The S-cube KB is instantiated on the graphical model of the system architecture, which generates a textual model in order 0 Figaro. This model implicitly defines a Continuous Time Markov Chain (CTMC), since all timed transitions of the model are associated to exponential distributions. Because of the combinatorial explosion of the states, this CTMC cannot be exhaustively built, but it can be explored by the quantification tools, in order to yield qualitative and quantitative results. Initially the state of the system is given by the values of the attributes and constants associated to each class (cf. Section 4.6.1). The interaction rules are first executed in order to initialize the values of EFFECTs, before the occurrence rules are executed. If the conditions of an occurrence rule are fulfilled, the risk event (FAILURE) can occur (instantaneously or in a time exponentially distributed). After the simulation of an event, which changes locally few attributes of a given object, the interaction rules are executed again in order to refresh the effects in the entire model.

In order to yield qualitative and quantitative results, the order 0 Figaro model can be processed using the Figseq or Yams tools. We give below an overview on the principle of each one of those tools (already introduced in § 3.1.4.2):

- The FigSeq tool: explores, step by step (cf. § 4.4.3), the sequences leading to the undesirable event. Given the time mission time and truncation criteria, FigSeq computes an estimated value of the undesirable event probability taking into account the contribution of the explored sequences that led to the undesirable event, and gives also a pessimistic value taking additionally into account the truncated sequences. The truncation criteria are specified in the Figseq tool and can be for instance the minimum probability of the sequence or the maximum number of the sequence branches.
  FigSeq can be used only in case of a purely Markovian model. The use of exponential distributions only in the S-cube KB allows to take benefits from the mathematical properties of the tool, and in particular the qualitative analysis yielding the attack and failure scenarios and the use of the Harrison [170] technique to compute their probabilities;
- The Yams tool: uses the "analog"[16] Monte Carlo simulation [171] on the system model to compute an estimated value of the undesirable event probability. Any kind of probability distribution can be associated to transitions with this tool. Yams is also able to output a selection of simulated scenarios, but the obtained results are much more "noisy" than those obtained with Figseq; in particular, there is no warranty that all scenarios with a probability greater than a given threshold can be obtained.

Before processing the Figaro textual model with quantification tools, the user defines an undesirable event (target state). The model processing generates attack and failure scenarios leading to the undesirable event defined, with an estimation of their probabilities (Ph.2 in Figure 26).

[16] Analog means here: without an acceleration technique. The use of such techniques is not easy in the general case where various kinds of probability distributions are used [171].

The attack and failure scenarios are listed into a table (cf. Table 28 for example) and sorted by decreasing contribution to the undesirable event, whose probability is also calculated. These scenarios are composed of the attack steps and failure modes associated to each system component (cf. Section 4.4) described in the S-cube KB.

We presented in this section the tool chain used in different phases of the S-cube approach (cf. Figure 26). More details on the S-cube KB implementation are given in Annex 2 where each class is individually described.

4.7 Conclusions

We presented in this chapter the main principles of the S-cube approach, related to modeling notions and the associated qualitative and quantitative aspects, and how they have been implemented.

S-cube provides a risk analysis framework (tool-based approach) to assess information and control architecture of industrial systems. Thanks to a taxonomy and hierarchical reasoning, we identified the attacks and failure modes these systems are subject to and associated them with quantitative metrics.

The S-cube approach has been implemented thanks to the Figaro modeling language and its associated tools. The system architecture is first modeled graphically by the user, and then processed with the quantification tools. The qualitative analysis provides the scenarios composed of attack steps and failures that lead to a given undesirable event. We can distinguish three kinds of possible scenarios:

- Purely accidental scenarios: which consist of only accidental components failures;
- Purely malicious scenarios: which consist of only attack steps;
- Hybrid scenarios: which consist of a mix of accidental failures and attack steps.

The quantitative analysis allows to sort these scenarios by their probabilities, which makes it easier to exploit the results, and gives an estimation of the undesirable event probability.

We note that S-cube bas been subject to a European patent filing referenced “B2581-S-CUBE EPO FILING”: Method for assessing safety and security risks of an industrial process.

We illustrate in the next chapter the S-cube approach on realistic and complex case studies in order to show its applicability and its ability to generate joint safety and security risk analysis.

Chapter 5

S-cube application on case studies

We illustrate in this chapter the S-cube approach on different use cases in order to show its ability to model real complex systems and assess the related safety and security risks. The S-cube approach can be used for risk assessment at different levels of the system architecture: the corporate level or the industrial control level, and at different phases of the system lifecycle: the design phase or the operational phase.

In order to show the various purposes for which S-cube can be used, we deal in this chapter with two case studies: the first one addresses the use of S-cube on the corporate level in the operation phase; the second one uses S-cube in the design phase of an industrial control architecture.

5.1 Modeling corporate networks

We give in this section the example of an industrial system with the associated control and corporate architecture, where new information and communication technologies are used. We first describe the architecture of the case study then we give the associated risk analysis using the S-cube approach. In this case study, we particularly show how the S-cube approach can be used during the operational phase in order to assess the emergent risks and vulnerabilities.

5.1.1 Description of the case study

We consider the system architecture, depicted in Figure 33, which consists of four network zones: the corporate network, a demilitarized zone (DMZ), the process control network and the field network. The corporate network, the DMZ and the process control network are separated by firewalls. The field network comprises the sensors and actuators used to sense and manipulate the industrial process, as well as the Process Controller. This latter communicates with an Acquisition Server via the process control network. The Acquisition Server is used for both collecting the process data and supervising the industrial process. The process data are stored in an ftp server (*http_ftp_server* in Figure 33) placed in

the demilitarized zone. An operator workstation, connected to the corporate network, hosts an http client application which uses the data stored in the *http_ftp_server* for statistical and optimization purposes. This system can be considered as a simple example containing all levels of the PERA (cf. Figure 2).

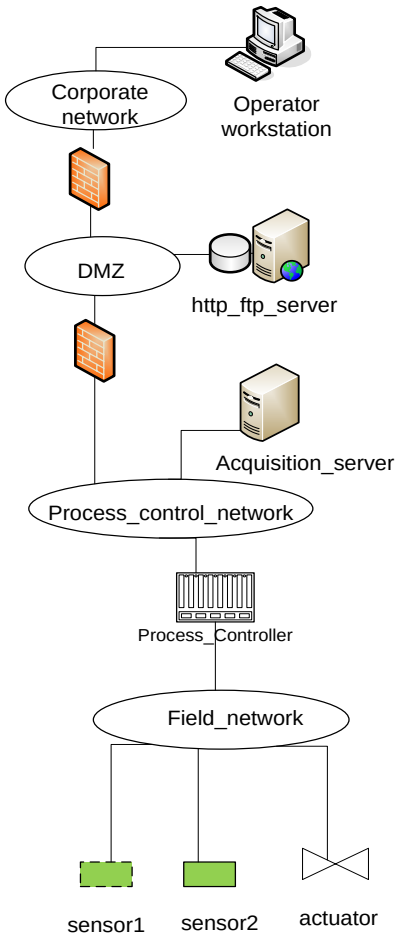


Figure 33: The system architecture under study

We describe the system architecture (Ph.1 in Figure 26), as illustrated in Figure 34 using the modeling elements provided by the S-cube KB. The words in *italic* font refer to classes used in the S-cube KB (cf. metamodel in 4.3.2 and detailed description of classes in Annex 2) or to the modeling elements used in Figure 34.

As previously discussed in § 4.3.1.3, S-cube models both the functional and logical architectures. The functional architecture is described by the different machines, the networks they are connected to, and the software components they are hosting (modeled with circles). The logical architecture is addressed by modeling the data flows between the different software components.

The connection of a physical machine to a network zone is modeled with a dotted black link (*link_machine_network*). The association between the physical components and the software running on them is modeled with dashed black arrows links (*link_machine_soft*). The solid blue arrows model the allowed data flows between the different software components. The firewall models the filtering policy between the two network zones it separates which implies that only the modeled data flows can be exchanged and no other undefined data flow can be initiated.

The field network comprises sensors: *sensor1* and *sensor2* and the *actuator*. The sensors measures are sent from the sensors software components (*sensor1_soft* and *sensor2_soft*), to a voter (*k/n gate*), which sends the measure to the *process_controller_soft* (the process control software running on the *Process_Controller*). This latter sends back instructions to the *actuator_soft* which executes the action on the process. The field network uses a wireless communication to exchange data between the process controller, sensors and actuators.

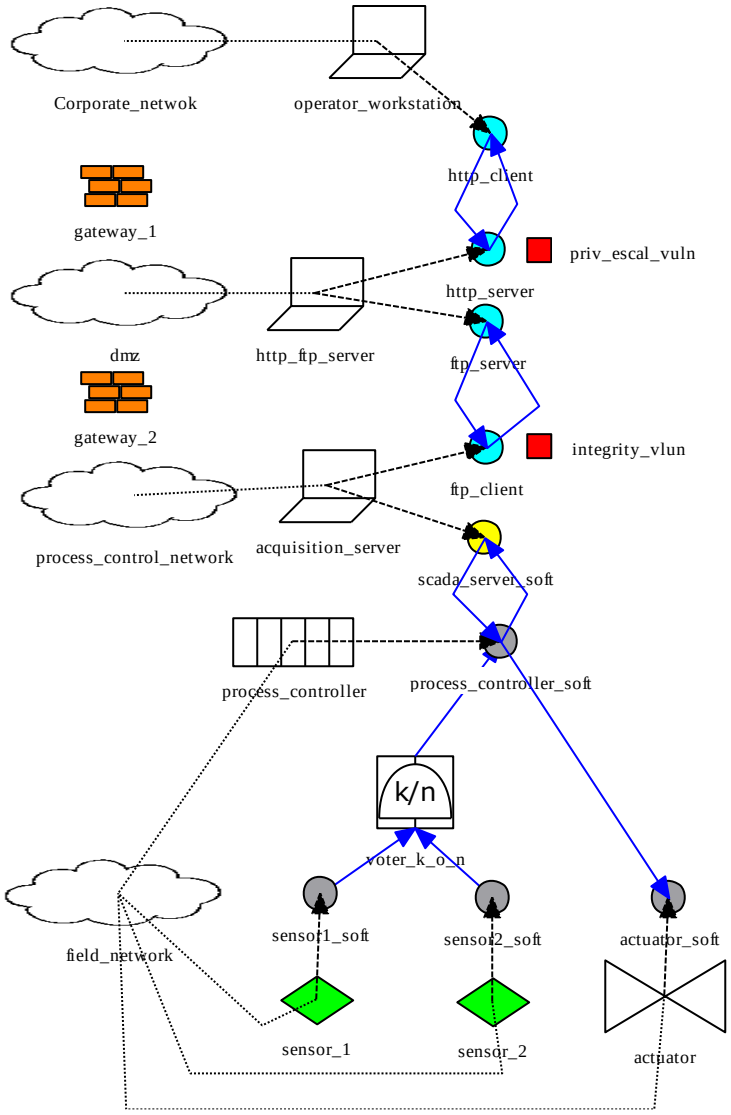


Figure 34: The graphical model as input by S-Cube

The *process_controller_soft* communicates feedback about the process to the SCADA server software (*scada_server_soft*), running on the acquisition server, and receives back instructions from the operator. The acquisition server also hosts an *ftp_client* that communicates with the *ftp_server* running on the *http_ftp_server* placed in the demilitarized zone. The *http_client* application running on the operator workstation communicates with the *http_server* hosted by the *http_ftp_server*.

We make the following assumptions regarding the architecture under study:

- physical access to the operator workstation is possible;
- the *http_ftp_server* is running with user privileges;
- the *acquisition server* is running with user privileges;

We assume that the following vulnerabilities exist on the architecture and have not been patched:

- a vulnerability exists on the *http_server* and results into privilege escalation;
- a vulnerability exists on the *ftp_client* and results into integrity loss;
- a configuration vulnerability exists on the acquisition server and results into root privileges acquisition.

In order to analyze this architecture with the S-cube approach, we first input the graphical model using KB3. The graphical elements corresponding to the different classes defined in the S-cube KB are given in a palette, as described in § 3.1.4.1, and are used to reproduce the system architecture, depicted in Figure 34.

We propose in the next section process this system architecture with S-cube and analyze the results generated (ph.2 in Figure 26).

5.1.2 Qualitative and quantitative risk analysis

We evaluate the described architecture, with the S-cube KB as described in Figure 32, in order to assess the risk related to the undesirable event "*actuator_does_not_act_properly*". For instance we can imagine that this architecture is used to control and supervise a chemical plant and that the process controller sends an instruction to stop heating but the heater does not respond which can lead to exceeding temperature limits and result in safety consequences (explosion, human injuries).

We focus in this example on the qualitative behavior described in the various components of the S-cube KB and take simple hypotheses for the occurrence rates of the events that can affect the security or safety of the system. The quantitative analysis produces the following results: after one year of functioning without maintenance (the components are supposed non-repairable in this example) and without considering detection and prevention measures, the probability of the actuator not acting properly reaches 0,48. Of course, this seems very high, but we made the pessimistic assumption that the undesirable event occurs whenever one actuator in the field network receives wrong or no instruction from the process controller. The malfunction of the heater may not be sufficient to create a safety-related risk and must be combined with malfunctions of other components such as hard-shutdown mechanisms.

The attack and failure scenarios that can lead to this undesirable event are automatically generated using the FigSeq quantification tool (ph.2 in Figure 26). They are generated based on the rules in the knowledge base describing attacks (cf. Section 4.4) failures and the rates (inverse of mean time to compromise resp. mean time to failure) of the exponential distribution associated to each rule (cf. Section 4.5). These scenarios are sorted in a table according to their deceasing occurrence probabilities and hence their contributions to the undesirable event.

Attack scenarios: We extract in Table 25 the first three scenarios; which are purely malicious as they are composed of only attack steps.

Seq. n°	Transition name	Rate (per hour)	Pr.
1	access(operator_workstation)	1e-4	1.12e-1
	exploit_server_vuln_priv_escal (http_server)	1e-3	
	exploit_server_vuln_integrity_loss (ftp_client)	1e-3	
	privilege_escal.(acquisition_server)	1e-3	
	send_false_instruct_to_process_controller(scada_server_soft)	0.1	
2	access(operator_workstation)	1e-4	1.12e-1
	exploit_server_vuln_priv_escal (http_server)	1e-3	
	exploit_server_vuln_integrity_loss (ftp_client)	1e-3	
	privilege_escal.(acquisition_server)	1e-3	
	send_no_instruct_to_process_controller(scada_server_soft)	0.1	
3	jamming_attack(field_network)	1e-5	3.85-2

Table 25: The most probable attack scenarios

We can see for example that the first scenario (Seq. n°1 in Table 25) consists of five attack steps: in the first step the attacker succeeds in having access to the operator workstation (here because the attribute *physical access* of this machine was set to true but the attacker can also have remote access). In the second step, the attacker exploits remotely the existing vulnerability in the *http server* which results into privilege escalation. The *http server* is consequently compromised and the attacker has root privileges on the *http ftp server* machine which enables him to compromise also the *ftp server* running on it. As the *ftp server* communicates with an *ftp client* running on the *acquisition server* (cf. Figure 33), the attacker tries in the third step to remotely exploit the vulnerability in the *ftp client*. The *ftp client* is then compromised. Given that the vulnerability leads only to integrity loss the attacker will also need to make a *privilege escalation attack*, in the fourth step, exploiting the configuration vulnerability related to the acquisition server in order to be able to compromise the *scada server* software. If the attacker succeeds in compromising this latter then he can, finally, send false instructions to the *process controller* which will itself send false instructions to the *actuator*. This latter will consequently not act properly when required which leads finally to a safety related consequences.

The second attack scenario (Seq. n°2) is the same as the first one except for the last step. For this latter, instead of falsifying data, the attacker will deny service so that no instructions will be sent to the process controller which will itself send no instructions to the actuator when needed.

The third attack scenario (Seq. n°3) is a jamming attack at the wireless field network. This attack is less probable as it requires the attacker to be next to the process controller or the actuator to jam the communication which is not that easy.

Accidental scenarios: The next three risk scenarios, given in Table 26, are purely accidental as they are composed of only component failures. These scenarios are with just one failure event (called single point of failure): the failure of the acquisition server, the field network or the process controller will cause instructions not to be sent to the actuator when needed.

Seq. n°	Transition name	Rate (per hour)	Pr.
4	accidental_fail(acquisition_server)	1e-6	3.85e-3
5	accidental_fail(field_network)	1e-6	3.85e-3
6	accidental_fail(process_controller)	1e-6	3.85e-3

Table 26: The most probable accidental scenarios

Hybrid scenarios: The structure of this system is too simple to give birth to hybrid scenarios, where the *combination* of accidental failures and attacks leads to the undesirable event. This is due to the absence of redundancy for the PLC or acquisition server. If there were such redundancies, we could see scenarios where one of these components is lost accidentally and the other one because of an attack.

We conclude from the results obtained that for the case study architecture given in Figure 33, the acquisition server and the process controller are the most critical components and their failure or compromise can lead to safety related consequences. Mitigation measures in this context would be to deploy redundant components with different technologies in order to provide the main functionalities to control the process in case of unavailability. This would also make attack scenarios more difficult to achieve as the attacker would need to find other vulnerabilities and succeed in exploiting them in order to falsify instructions sent to actuators.

5.1.3 Conclusions and enhancement

We have modeled in this example attacks that target the corporate networks and that try next, by multi-stage multi-hopping, to reach the industrial network in order to interfere with the normal operation of the industrial process.

Security enhancement measures recommended by our security experts (step (2.iii) in Figure 26) would be to inhibit any incoming dataflow towards the process control network. This can be achieved by the introduction of data diodes that allow data to travel only in one direction. Classical firewalls can decide about who initiates the connection; but once the communication channel is established, the data can be exchanged in both directions.

We modified the system architecture in Figure 34 by removing the data flow from the ftp_server to the ftp_client (unidirectional communication from the ftp_client to the ftp_server). By processing again the modified architecture with S-cube, no attack scenarios leading to the undesirable event were generated.

We have demonstrated in this case study how the S-cube approach can be used to assess the risks related to operational system architectures. In particular, the impact of the new vulnerabilities to which the system may be subject during its exploitation phase can be assessed.

We show in the next section how S-cube can also be used in the design phase to compare the safety and security of industrial architectures controlled by modern ICS. We study two variants of a hydroelectric ICS. For each variant, we model the system architecture (Ph.1 in Figure 26) and analyze (step 2.i) the qualitative and quantitative analysis generated (Ph.2 in Figure 26) by S-cube.

5.2 Modeling a hydroelectric ICS: variant 1

Using S-cube, we study in the remainder of this chapter a realistic case study: a pumped storage hydroelectric plant in order to show the ability of this approach to model real and complex systems and to yield the associated risk analysis.

This case study is inspired from the Taum Sauk pumped storage plant. In 2005, a famous accident happened at this installation; it resulted in the destruction of a section of the upper reservoir and the

sudden release of a large volume of water down the slopes. We give first a quick overview on the Taum Sauk accident and then we describe the system architecture of our case study that was built from the incident reports and analyses following this accident.

We focus hereafter on modeling the industrial control architecture and show how S-cube can be used in the design phase in order to easily model different hypotheses on the same architecture and generate the attack and failure scenarios for each configuration. We consider in particular two variants of the case study. By comparing the risk analysis generated for each variant, we demonstrate consequently how S-cube helps choosing the best configuration for both the safety and the security of the system.

The remainder of this section will be organized as follows. First, we give an overview on the Taum Sauk dam failure. Second, we describe the system architecture, common for the two variants of the case study. We address next in the remaining sections the first variant (variant 1), while the second variant (variant 2) will be depicted in Section 5.3. For each variant we give the S-cube graphical model (Ph.1 in Figure 26). This latter is next processed with S-cube which generates risk analysis; we make first a pure safety risk analysis, and then a joint safety and security risk analysis. We give finally our conclusions on each case study. The comparison between the two variants is given in Section 5.4.

5.2.1 Overview on the Taum Sauk upper reservoir failure

The Taum Sauk hydroelectric Power Plant is a pumped storage hydroelectric power station; an example of such an installation is given in Figure 35. It consists of two water reservoirs: an upper reservoir and a lower reservoir separated by a penstock. In high electricity demand hours, water is released from the upper reservoir to the lower reservoir in order to generate electricity. In low electricity demand hours, water is pumped back from the lower reservoir to the upper reservoir for energy storage. The reversible pumps are situated in the station.

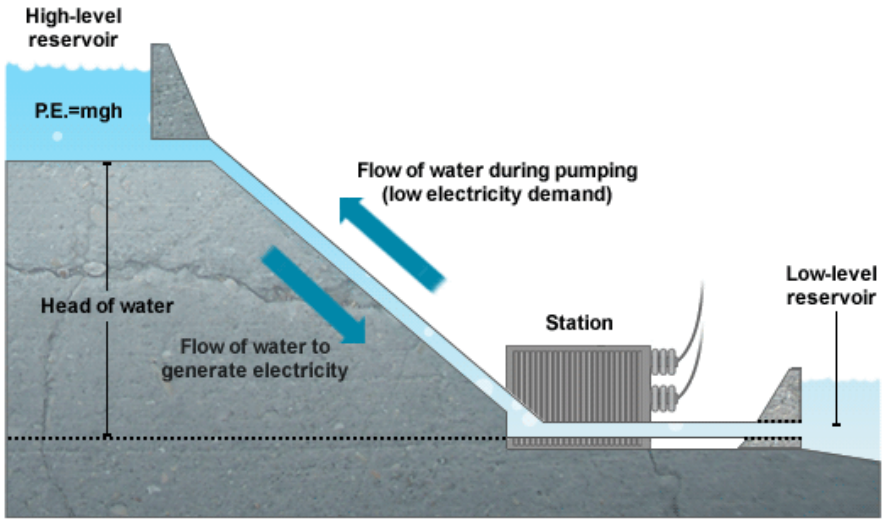


Figure 35: A pumped storage hydroelectric power station [172]

On December 2005, the Upper Reservoir of the Taum Sauk Pumped Storage Project was overtopped when water continued to be pumped from the lower reservoir to the upper reservoir. This led to the failure of a section of the upper reservoir embankment and the sudden and complete evacuation of water. This breach of the upper reservoir had catastrophic consequences: flooding of the surrounding areas including Highways, campgrounds and adjacent properties.

The Taum Sauk upper reservoir breach was caused by a purely accidental failure of the instrumentation used for monitoring the reservoir level. In addition, incident reports have shown that the real system architecture, when the breach occurred, did not respect the initial design, which decreased the system resilience and accelerated the upper reservoir failure. In [173], S-cube was used to model both the designed and the implemented architectures and to compare the risk scenarios related to each architecture. This study showed how the catastrophic failure of the Taum Sauk Upper Reservoir could as well be the result of a cyber-attack.

5.2.2 Description of the case study architecture

The system architecture of our case study is depicted in Figure 36. It was inspired and built from information collected on the instrumentation and control system of the Taum Sauk Upper Reservoir [174][175][176].

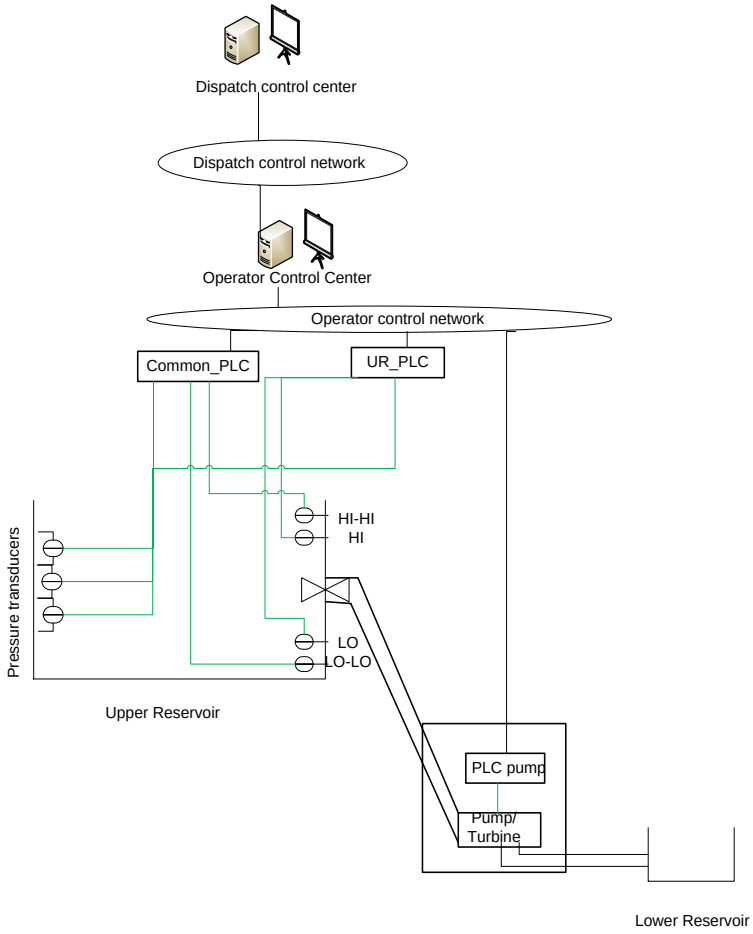


Figure 36: The system architecture under study

The instrumentation system is composed of two sets of sensors:

- Three primary Pressure sensors, placed at a given elevation, convert the pressure into water level. These measures are used to monitor the reservoir level and hence trigger the shutting down of the pumps/generators units;

- Four conductivity sensors placed in pairs above and below the highest and the lowest water level. These sensors are activated if the water reaches the level at which they are placed. The sensors *HI* and *HI-HI* determine whether the water level in the upper reservoir is too high, in which case the

generating mode is activated. The sensors *LO* and *LO-LO* determine whether the water level is too low, in which case the pumping mode is activated.

The control system relies on two main Programmable Logic Controllers (PLC): the Common PLC and the Upper Reservoir (UR) PLC. The Taum Sauk installation was remotely controlled through a microwave system from the Osage Plant, under the direction of the load dispatching in St. Louis [175]. The dispatching control center provides generating Megawatt instructions (with pump start/stop mode) to the Operator control center at the Osage plant; from there, the operator remotely controls the Taum Sauk units (two reversible pumps) at the Lower Reservoir.

The primary pressure sensors send the water level measures to both the Common and the UR PLCs. These measures are transmitted to the Operator and dispatching control center. The HI and the HI-HI sensors are placed at two different elevations and used for emergency shutdown should extremely high water levels occur.

In its normal operation, the plant is controlled by pressure sensors. The average value of the three readings is considered by the controllers. In the pumping mode, the pumps are activated in order to stock the water in the upper reservoir. If for some reasons the pressure sensors did not operate correctly, the HI sensor would be reached which should activate the automatic shutdown of the pumps. If for some reasons, the pumping mode was not terminated, the HI-HI sensor would be encountered which activates a hard emergency stop of the pumps.

The two process controllers and the conductivity sensors HI and HI_HI constitute a Safety Instrumented System (SIS), designed to bring the process to a safe state in case of dysfunction of the pressure sensors.

We consider in the following the system architecture used to control the water level in the Upper Reservoir. We study in the remainder of this section the first variant of this architecture (called later case study variant 1), which we model graphically with S-cube, then analyze in order to get the associated risk analysis. We consider first only safety risks, and second both safety and security risks that lead to the undesirable event: the Upper Reservoir failure. We give finally conclusions based on the results obtained.

5.2.3 The graphical model

We make the following assumptions for variant 1 of the system architecture used to control the water level in the Upper Reservoir as described in Section 5.2.2 (cf. Figure 36):

- The two primary PLCs: Common PLC and UR PLC are identical (same hardware), use the same technologies (same automation software and same communication protocol) and are subject to the same environmental constraints;
- The HI and the HI-HI sensors are connected, with separate conduits, to both PLCs. This means that both the Common PLC and the UR PLC can receive feedback from the sensors when activated.

The graphical model input with the KB3 HMI and modeling the case study variant 1 architecture is depicted in Figure 37. As done for the case study in Section 5.1, we model the physical machines connected to each network zone, the software components running on them, and the data flows between the different software components. Text in *italic* font refers to the modeling objects used in Figure 37.

The primary pressure sensors are modeled with the physical components "*pressure1*", "*pressure2*" and "*pressure3*". The software components associated to each sensor, and that enable to capture and send the measures, are named *P1, P2* and *P3.* Concerning the conductivity sensors, we model only the high level sensors *S_HI_HI* and *S_HI* and the associated software, named *HI_HI* resp. *HI*.

The three water pressure sensors are identical and are subject to the same environmental constraints. They are consequently associated with the same common cause failure group (*CCF_group3_sensors*).

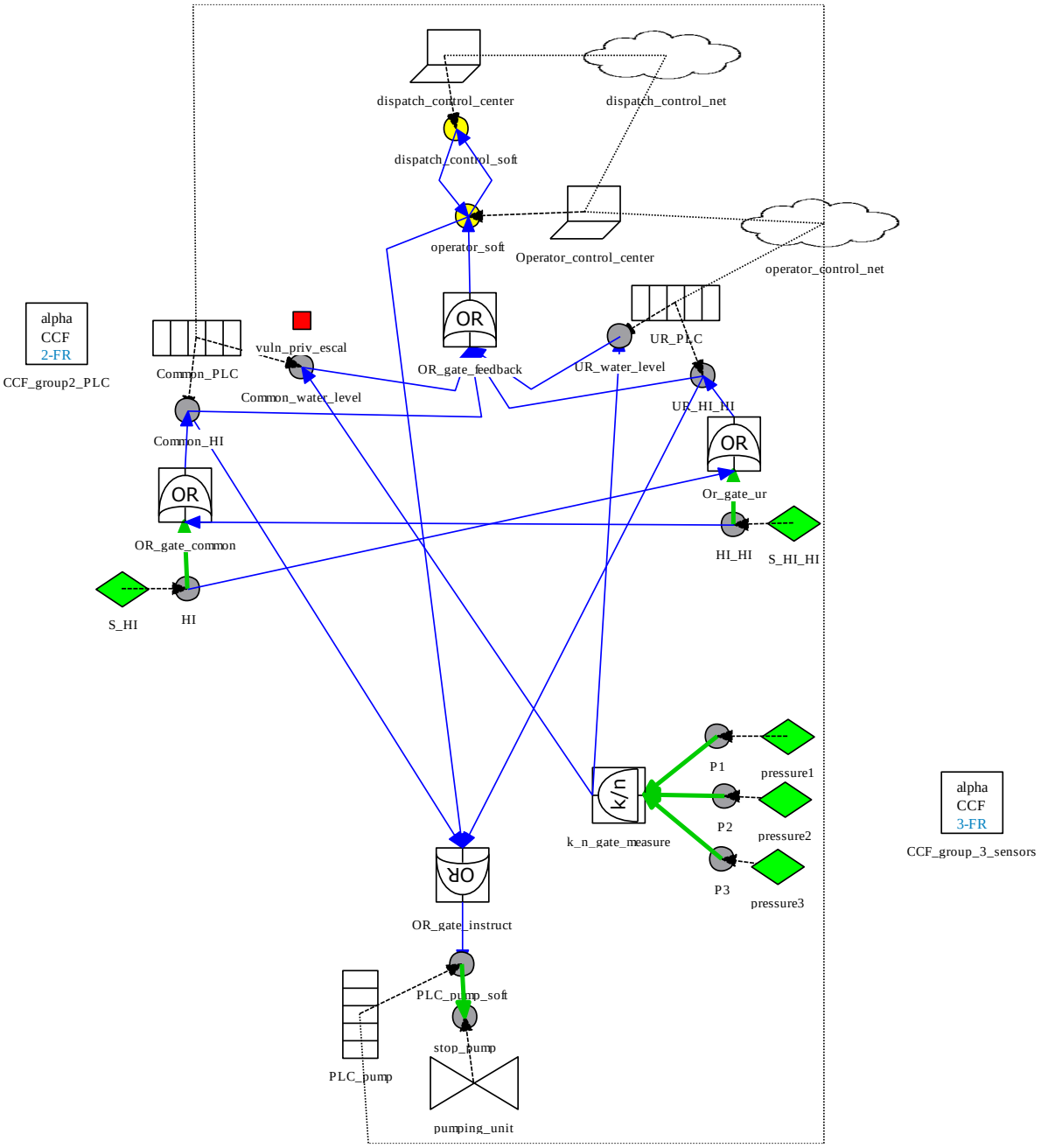


Figure 37: The system architecture of case study variant 1

The three of them send their readings to the *k_n_gate_measure*, which is a 2 out of 3 voter. If two readings out of the three are consistent, the measure will be sent to both process controller software components *Common_water_level* and *UR_water_level* of the *Common_PLC* resp. the *UR_PLC* that control, in normal operation of the plant, whether the water level is within the operational limits. As a safety mechanism, the HI sensor is activated in case water level reaches the elevation at which this sensor is placed. In this case, an alarm is sent from the S*_HI* sensor software component to both *Common_HI* and *UR_HI_HI* software running on *Common_PLC* resp. *UR_PLC* which send an instruction to *PLC_pump*; used to control the *pumping_unit*; to shut down the pump. If for some reason, the pump is not shut down, the water would reach the *S_HI_HI* sensor level and an alarm would be sent, like with the HI sensor to both PLCs which would send an instruction to *PLC_pump_soft* for emergency shutdown of the pump. The *OR_gate_feedback* before the *Common_HI* software models the fact that if this latter receives an alarm from the HI or the HI_HI sensors, it would send emergency shutdown

instruction to the *PLC_pump* in order to stop the pumps. The same goes for the *UR_HI_HI* software on the *UR_PLC*.

The green links model data flows carried by electrical wires, which is the case for the data flows sent by the sensors. These latter are connected to the process controllers by electrical wires. Data flows modeled with blue arrows are carried by a network.

Given the assumption we made on identical PLCs, we associated the *Common_PLC* and the *UR_PLC* with the same *CCF_group2_PLC*. The software components *Common_water_level* and *UR_water_level* have consequently the same software type; this latter has a vulnerability which results into privilege escalation if exploited by an attacker.

All feedback from the *UR_PLC* and the *Common_PLC* is reported to the *operator_control_center* used for the acquisition and supervision of the process. The *operator_soft* receives feedback from the *OR_gate* that receives four input data flows. If one of the four feedbacks indicated high water level values, the operator would send an instruction to *PLC_pump* to shut down the *pumping_unit*. The *PLC_pump_soft* receives the shutdown instruction from the *OR_gate_instruct*, which receives it either from the operator in normal operation or directly from one of the controllers (*UR_PLC* or *Common_PLC* software) in emergency regime.

The operator control center sends feedback on process status to the SCADA software (*dispatch_control_soft*) running on the remote dispatch control center, and receives back instructions on the Megawatts to generate.

We assume that the operator control network that connects the PLCs and the operator control center uses a wireless communication technology and that it uses no authentication and encryption mechanisms.

The system architecture modeling, input using the KB3 HMI as described in Figure 37, corresponds to the first phase (Ph.1) of the S-cube process described in Figure 26. All the assumptions made on the functional architecture (step 1.i) and the related safety and security aspects (steps (1.ii) resp. (1.iiii)) are considered in the graphical model by double clicking on the system components and filling in the appropriate interfaces and attributes.

We propose in the next section to assess the risks related to the case study variant 1. We consider the following undesirable event: the loss of the protection function that shuts down the pumps when the water level reaches the maximum permitted level. It means that the undesirable event happens when the pumps do not stop when required. This undesirable event can lead, in the worst case, to the upper reservoir breach as happened for the Taum Sauk dam (cf. § 5.2.1) and consequently to safety issues.

We process the graphical model in Figure 37, using the S-cube KB and the quantification tools (cf. § 4.6.2), and generate automatically the accidental and malicious scenarios leading to the undesirable event, with an estimation of their probabilities (Ph.2 in Figure 26). We give in Annex 3 the quantitative parameters assigned for this case study and associated with the failure rates of the system components and the mean time to success for the attack steps.

We propose at first to make a traditional safety analysis that considers only accidental failures. We take at a second time additionally malicious risks into consideration in order to assess their impact on the probability of the undesirable event.

5.2.4 Pure safety risk analysis

Using the global object, we set the Boolean *inhibit_attacks* to true (cf. Annex 2). We generate automatically the accidental scenarios likely to happen on this case study and that result into the upper reservoir breach. We also model for this case study maintenance actions that consist in physical components repair in case of accidental failure.

The scenarios generated are given in a table and sorted by decreasing contribution to the overall undesirable event probability (cf. Table 27 for example). Each scenario is composed of one or many transitions, whose rates are also indicated in the table.

Based on the hypothesis taken for types of failures and the corresponding failure rates given in Annex 3, the quantitative analysis yields the following results: after one year of operation, with maintenance in case of accidental failures, the unreliability of the function of shutting down the pumps is estimated to 1.06e-2. This may seem unacceptable, but one must not forget that the inability to shut down the pumps can lead to a dam failure only if it happens in a situation where the water level is very high and the pumps are running.

The scenarios generated are reorganized into minimal cut sets sorted by decreasing contributions to the undesirable event. We give in Table 27 the four most probable scenarios likely to happen for the first case study and resulting into the undesirable event.

Num Seq.	**Transitions**			**Proba.**	**Contrib.(%)**
	Name	**Rate[17]**	**Class**		
1	ccf_12(CCF_group2_PLC)	9.52e-7	EXP	7.3e-3	68
	accidental_failure(Common_PLC)	1	INS		
	accidental_failure(UR_PLC)	1			
2	accidental_failure(operator_control_net)	1e-7	EXP	7.61e-4	7
3	accidental_failure(PLC_pump)	1e-7	EXP	7.61e-4	7
4	accidental_failure(Common_PLC)	1e-5	EXP	5.3e-5	0.5
	accidental_failure(UR_PLC)	1e-5	EXP		

Table 27: The first accidental scenarios likely to happen on case study variant 1

- Scenario 1 with 68% of contribution to the undesirable event: a common cause results into the instantaneous and simultaneous failure of the primary PLCs: *Common_PLC* and *UR_PLC*, belonging to the same *CCF_group2_PLC*;
- Scenario 2: consists of the accidental failure of the operator control network (e.g., unavailability of the switch) which results in the unavailability of all the data flows carried by this network and in particular the instructions for shutting down the pumps that would not be sent when required;
- Scenario 3: consists of the accidental failure of the *PLC_pump* used to control the operation of pumps;
- Scenario 4: the accidental failure of the *Common_PLC* followed by the accidental failure of the *UR_PLC*; the two failures are independent from each other. This scenario assumes that the accidental failure of the second controller would occur before the failure of the first controller would be repaired. This scenario could be mitigated by increasing the rapidity of maintenance actions.

The redundancy between the primary process controllers reinforces the system resilience in case of the accidental failure of one controller. However, as the two controllers have been implemented with the same technology for cost reduction purposes, they can hence be subject to a common cause failure. With a contribution of 68%, common cause failures increase considerably the overall probability of the undesirable event which is relatively high.

[17] Per hour for exponential distributions rates

Single failure points, which are in our case the failure of the operator control network and the failure of the pumps controller, should also be minimized with the use of a more reliable hardware and maybe redundant components.

5.2.5 Safety and security joint risk analysis

The control system architecture of this case study (cf. Figure 36) uses information and communication technologies (remote control and supervision software, wireless networks, etc.). It can be consequently subject to cyber malevolence. We propose here to process the system architecture while taking into account both accidental and malicious risks (both Booleans *inhibit_attacks* and *inhibit_failures* are set to False) and calculate the unreliability of the pumps shutdown function.

For this joint safety and security analysis, we make two studies: the first one simulates the system during a short mission time (four days) after the start of an attack and the second one considers a long mission time (one year), but without attack in initial state.

5.2.5.1 Short mission time with an attack at t=0

We assume for this study that at the beginning of the simulation (t=0) the attacker has intended to attack the system and has already collected some information concerning his/her target. We model maintenance actions for accidental failures but we do not take into account detection and reaction measures for the attacks.

Using the quantification tools Yams and FigSeq (cf. Section 4.6.2), we simulate the system over a short mission time of 100 hours (~ four days) to evaluate the probability of the undesirable event (loss of the protection function).

We process the model first using Yams, the Monte Carlo simulator. After 100 hours of operation the probability of the breach is estimated to 9.22e-3. Second, using the FigSeq tool, we generate automatically the attack and failure scenarios leading to the undesirable event and having a minimum probability[18] of 1e-5. These scenarios are reorganized into four minimal cut sets given in Table 28 and sorted by decreasing probabilities.

N° Seq.	**Transitions**			**Proba.**	**Contrib.** (%)
	Name	**Rate (per hour)**	**Class**		
1	Access (operator_control_net)	0.06	EXP	2.88e-3	31
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	send_no_feedback (Common_water_level)	0.8	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.8	EXP		
	send_no_feedback (UR_water_level)	0.8	EXP		
	send_no_feedback (Common_HI)	0.7	EXP		
	send_no_instructions_to_actuator (Common_HI)	0.8	EXP		
	send_no_feedback (UR_HI_HI)	0.8	EXP		
	send_no_instructions_to_actuator (UR_HI_HI)	0.8	EXP		

[18] This truncation criteria reduces the number of sequences generated by the tool which can quickly explode, but consequently the probability of the undesirable event is not accurate enough because all sequences are not taken into account; that's why we use the Monte Carlo Simulator YAMS to assess the overall undesirable event probability and FigSeq in a second time for assessing the probability of the most probable scenarios.

2	Access (operator_control_net)	0.06	EXP	1.29e-3	14
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	Send_false_feedback (Common_water_level)	0.7	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.7	EXP		
	send_false_feedback (UR_water_level)	0.7	EXP		
	send_false_feedback (Common_HI)	0.7	EXP		
	send_false_instructions_to_actuator (Common_HI)	0.7	EXP		
	send_false_feedback (UR_HI_HI)	0.7	EXP		
	send_false_instructions_to_actuator (UR_HI_HI)	0.7	EXP		
3	jamming_attack(operator_control_net)	1e-6	EXP	8.85e-5	0.96
4	ccf_12(CCF_group2_PLC)	9.52e-7	EXP	8.43e-5	0.46
	accidental_failure(Common_PLC)	1	INS		
	accidental_failure(UR_PLC)	1	INS		

Table 28: Attack and failure scenarios for the variant 1 after 100 hours of operation

First/second scenario: the most probable scenarios likely to happen on this system architecture are attack scenarios. The first two scenarios have the same attack strategy and consist of nine attack steps:

- 1st step: accessing the operator control network; given that this latter uses a wireless technology and no authentication is required to access it, an expert attacker can intercept the data flows transiting through this network and read them (data encryption is generally not employed in industrial networks). For this first study, we assume that the average time needed to carry out this attack step is one week (*MTTS_access* = 168h).
- 2nd step: exploiting the vulnerability in the *Common_PLC* resulting into privilege escalation. Having access to the operator control network, the attacker can reach any machine connected to this network. He will particularly try to remotely exploit the existing vulnerability on the *Common_PLC* software in order to have complete privileges; this would take him an average of 4 days (cf. § 4.5.2.3);
- 3rd step: Denying/falsifying the feedback on the water level reported by the *Common_PLC* to the operator; this is possible as the attacker has full privileges on this controller. This attack step results into the operator not being informed about the real water level: the attacker can for example send false values or freeze the process state on the operator HMI. The MTTS for this attack step is about 1 hour;
- 4th step: exploiting the vulnerability in the *UR_PLC* resulting in privilege escalation. Since both controllers are in active redundancy, the attacker will have also to compromise the *UR_PLC*. Seeing that the attacker has already exploited the same vulnerability of the same software type with the first controller, he will spend much less time to succeed into exploiting it again; the MTTS for this attack step is about 1h ($\lambda = 0.8$ instead of 0.01 in case the software type has not already been hacked by the attacker).
- 5th step: Denying/falsifying the feedback on the water level reported by the *UR_PLC* to the operator. The attacker needs also to falsify the feedback reported by the second controller that receives also the water level from the pressure sensors;
- 6th step: Denying/falsifying the feedback reported by the HI sensor when water encounters the HI elevation, and sent from the *Common_PLC* to the operator;
- 7th step: Denying/falsifying the instruction sent by the *Common_PLC* to shut down the pumps;
- 8th step: Denying/falsifying the feedback reported by the HI_HI sensor, when water encounters the HI_HI elevation, and sent from the *UR_PLC* to the operator;

- 9th step: Denying/falsifying the instruction sent by the UR_*PLC* to shut down the pumps.

Once the attacker has succeeded in accessing the operator control network and exploiting the first vulnerability, the following attack steps (3th to 9th steps) can be done more quickly (just over 1h for the MTTS) as the attacker gets acquainted with the process and data flows. The redundancy between the two controllers will incite the attacker to do the same attack steps twice in order to succeed into provoking the undesirable event. This is why, in the S-cube KB, a Boolean "*already_hacked*" is set to TRUE for all software applications of a given kind when one of them has been hacked. This strongly diminishes the MTTS for the future attack steps concerning the other applications of the same kind. .

We supposed that denying data flows is slightly easier to do (with a success rate of 0.8) compared to falsifying data flows (with a success rate of 0.7). However, the falsifying would be more strategic for the attacker in order not to be easily detected.

For simplification reasons, we refer to the first attack scenario with "denying attack scenario" and to the second attack scenario with "falsifying attack scenario".

Third scenario: this scenario models a jamming attack on the operator control network. This attack is specific to wireless networks and aims at interfering with the other legitimate communications (e.g., by continuously emitting a radio signal, sending regular packets, etc.) in order to deny their access to the medium. This results in the unavailability of all data exchanged via the network and particularly instructions to stop the pumping unit; which would lead to the undesirable event. This attack remains however with a low probability of occurrence (8.85e-5) as it requires for the attacker to have special and very expensive equipment because of the large geographical distribution of the system, and to be at the vicinity of the network access point.

Fourth scenario: this is the accidental scenario that consists in the common cause failure of the two primary controllers (cf. pure safety analysis in Section 5.2.4).

5.2.5.2 Long mission time without attack at t=0

We simulate in this study the system architecture in Figure 37 over a long mission time of one year. The MTTS set for the access attack step models in this study the frequency of attacks that target such an installation; we assume for this study that the frequency of such an attack is once a year (*MTTS_access* = 1 year).

The probability of the breach is estimated to 6.57e-2. The minimal cut sets generated are given in Table 29.

Num Seq.	**Transitions**			**Proba.**	**Contrib.** (%)
	Name	**Rate**[19]	**Class**		
1	jamming_attack(operator_control_net)	1e-6	EXP	5.2e-3	7.9
2	ccf_12(CCF_group2_PLC)	9.52e-7	EXP	4.98e-3	7.5
	accidental_failure(Common_PLC)	1	INS		
	accidental_failure(UR_PLC)	1	INS		
3	Access (operator_control_net)	1e-4	EXP	4.16e-3	6.3
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	send_no_feedback (Common_water_level)	0.8	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.8	EXP		
	send_no_feedback (UR_water_level)	0.8	EXP		

[19] Per hour for exponential distribution rates

	send_no_feedback (Common_HI)	0.7	EXP		
	send_no_instructions_to_actuator (Common_HI)	0.8	EXP		
	send_no_feedback (UR_HI_HI)	0.8	EXP		
	send_no_instructions_to_actuator (UR_HI_HI)	0.8	EXP		
4	Access (operator_control_net)	1e-4	EXP	1.87e-3	2.8
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	Send_false_feedback (Common_water_level)	0.7	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.7	EXP		
	send_false_feedback (UR_water_level)	0.7	EXP		
	send_false_feedback (Common_HI)	0.7	EXP		
	send_false_instructions_to_actuator (Common_HI)	0.7	EXP		
	send_false_feedback (UR_HI_HI)	0.7	EXP		
	send_false_instructions_to_actuator (UR_HI_HI)	0.7	EXP		
5	accidental_failure(PLC_pump)	1e-7	EXP	5e-4	0.7
6	accidental_failure(operator_control_net)	1e-7	EXP	5e-4	0.7

Table 29: Attack and failure scenarios for variant 1 over 1 year of operation time

With a frequency of attack equal to once a year, the probability of attack scenarios has become close to the probability of accidental scenarios (cf. Section 4.5.3). We can see that within a mission time of one year, the scenario of the common cause failure of the two controllers is more likely to occur than the "denying/falsifying" attack scenarios.

The results[20] yield also hybrid scenarios with probabilities around 1e-6. We give in Table 30 an example of a hybrid scenario that consists of two parts: first the accidental failure of the *UR_PLC* and second an attack on the *Common_PLC*. This implies that the failed controller had not been repaired before the second controller would be targeted by an attacker.

Num Seq.	**Transitions**			**Pr**	**Contrib. (%)**
	Name	**Rate (/hour)**	**Class**		
10	accidental_failure(UR_PLC)	1e-6	EXP	1.8e-6	2.7e-3
	Access (operator_control_net)	1e-4	EXP		
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	send_no_feedback (Common_water_level)	0.8	EXP		
	send_no_feedback (Common_HI)	0.8	EXP		
	send_no_instructions_to_actuator (Common_HI)	0.8	EXP		

Table 30: Example of a hybrid scenario

This scenario remains with very low probability as both controllers are operating in active redundancy and the failure of one controller would be detected by the operator station. However in case of standby

[20] These results were obtained for the system architecture of variant 1 without considering repair actions. For a repairable system, the hybrid scenarios are with very low probability and don't appear among the scenarios due to the minimum probability set as a truncation criteria.

redundancy, it would be more probable to have hybrid scenarios where the attacker starts compromising the first controller then occurs the failure on demand of the second controller.

5.2.6 Conclusions on case study variant 1

We can infer from the results that the probability of the undesirable event over one year of operation increases by 519% when security risks are considered in addition to safety risks. Without considering malicious actions and particularly cyber-attacks, the probability of the breach is under-estimated and the risk analysis is incomplete.

Within a short mission time (100 hours), the first joint safety and security study estimates the probability of the attack success given that the attacker intends to attack the system at t=0. Results show that attack scenarios are the most likely to happen. This is due to the fact that failure rates are generally very low compared to the rates associated with attack steps (cf. 2) in Section 4.5.3). Indeed, once determined in making his attack, the attacker spends hours/days on trying to achieve an attack step. On the other hand components failures take several years before happening.

With a mission time of one year, the second joint safety and security study aims at comparing the probabilities of the accidental and the attack scenarios. In this case, the MTTS for the access step (*MTTS_access*) initiating the attack is set to the inverse of the frequency of attacks that target specifically this kind of systems (cf. 1) in Section 4.5.3). The probabilities of attack and accidental scenarios are, in this case, of the same order of magnitude. However, attack scenarios increase considerably the probability of the undesirable event. We summarize in Figure 38, approximately, the distribution of scenarios according to their type (malicious, accidental or hybrid) for variant 1 over one year of mission time.

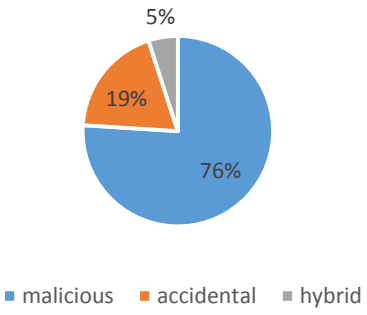


Figure 38: Repartition of scenarios according to their type for variant 1 over 1 year

Common cause failures have a large contribution to the undesirable event for this case study architecture. We propose in the next section to modify the system architecture taking into account these conclusions and re-assess the risk.

5.3 Modeling a hydroelectric ICS: variant 2

We consider in this section a second variant of the system architecture described in the Figure 36; for which we make the following assumptions:

- The primary controller (*Common_PLC* and *UR_PLC*) use different technologies (hardware and software), which implies that they cannot be subject to a common cause failure;
- The HI and the HI-HI sensors are connected, with separate conduits, each one to a different PLC: the HI sensor is connected to the Common PLC while the *HI_HI* sensor is connected to the *UR_PLC*.

Note that while the first change increases the cost of the system, the second one decreases it thanks to a simplification of the architecture.

5.3.1 The graphical model

Given the assumptions taken for this case study we make the following modifications in the graphical model already built in Figure 37. Thanks to the robustness of the S-cube approach, these changes can be done incrementally to the first model without the need to revamp the whole model. The resulting system architecture is given in Figure 39.

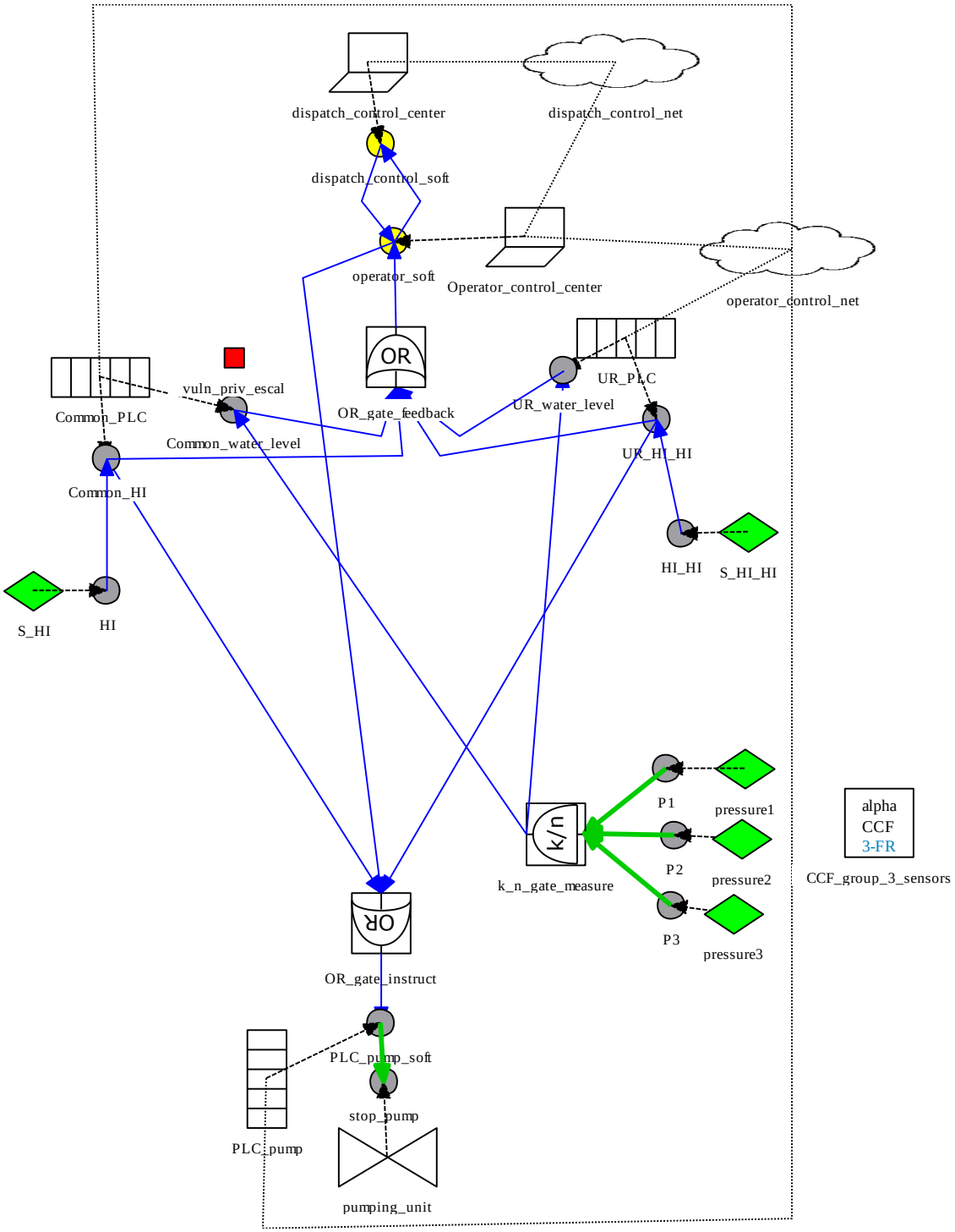


Figure 39: The system architecture of case study variant 2

- We remove the *OR_gates* before the process controller software components: *Common_HI* and the *UR_HI_HI*. The HI sensor sends feedback exclusively to the Common_PLC while the HI_HI sensor reports alarms to the UR_PLC;
- The *Common_PLC* and the *UR_PLC* are no longer associated to a *CCF_group* and the automation software associated with each PLC is different. The *Common_water_level* and the *UR_water_level* are consequently associated with two different software types. Each software type has a distinct vulnerability that can result into privilege escalation if exploited by an attacker.

The other modeling elements and hypotheses are the same as in variant 1.

We carry out exactly the same studies as for variant 1 in order to assess the impact of the architecture modifications on the qualitative and quantitative risk analysis. The same failure and attack rates given in Annex 3 are adopted also for this case study.

5.3.2 Pure safety risk analysis

We process the new graphical model in Figure 39 with S-cube, as previously done for variant 1, in order to assess the impact of the system architecture modifications on the probability of the Upper Reservoir breach and the scenarios leading to this undesirable event.

We first evaluate the system architecture of variant 2 taking into consideration only the accidental failures of the system components. After one year of operation, with repair actions in case of accidental failures, the probability of the upper reservoir breach is evaluated to 1.66e-3. The accidental scenarios generated are reorganized into minimal cut sets given in Table 31.

We have three accidental scenarios with a minimum probability of 1e-5: two single point failures (the accidental failure of the *PLC_pump*, the accidental failure of the operator control network) and the independent failures of the two primary controllers.

The common cause failure of the two primary PLCs is no longer conceivable on this system architecture. Using different software and hardware technologies, the *Common_PLC* and the *UR_PLC* are no more likely to fail simultaneously due to the same cause, contrarily to what was formerly the case in variant 1.

Num Seq.	Transitions			MT Proba	Contrib.(%)
	Name	Rate	Class		
1	accidental_failure(PLC_pump)	1e-07	EXP	7.8e-4	47
2	accidental_failure (operator_control_net)	1e-07	EXP	7.8e-4	47
3	accidental_failure(Common_PLC)	1e-05	EXP	5.68e-5	3.4
	accidental_failure(UR_PLC)	1e-05	EXP		

Table 31: Minimal cut sets of accidental scenarios for variant 2

5.3.3 Joint safety and security risk analysis

We process now the new system architecture taking into consideration both accidental and malicious risks, making the same studies as done for variant 1 in Section 5.2.5.

5.3.3.1 Short mission time with an attack at t=0

Over a short mission time of 100 hours, the probability of the undesirable event is estimated to 2.96e-3. The minimal cut sets obtained are given in Table 32.

Results show that, as for variant 1, attack scenarios are dominating with a contribution exceeding 88% of the overall breach probability. The first attack scenario likely to happen is the "Denying attack scenario", the second one is the "Falsifying attack scenario" and the third one is the "jamming attack" with a significantly lower probability.

We can see that the probability of the first attacks considerably decreased by about 68% for this case study compared to their probability obtained for the same study on the previous architecture (variant 1). This decrease is due to the fact that the two controllers use different technologies and automation software. This is modeled with S-cube (as explained in Annex 2) by associating two different software types with the automation software used for each controller. The vulnerabilities associated with each software are consequently different and it requires as much time and effort from the attacker to find an exploit for the first vulnerability as it requires for the second one. This makes it more difficult for the attacker to compromise both controllers in order to lead to the undesirable event.

Num Seq.	Transitions			MT Seq.	Contrib.(%)
	Name	Rate	Class		
1	Access (operator_control_net)	0.06	EXP	1.8e-3	60
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	send_no_feedback (Common_water_level)	0.8	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.01	EXP		
	send_no_feedback (UR_water_level)	0.8	EXP		
	send_no_feedback (Common_HI)	0.7	EXP		
	send_no_instructions_to_actuator (Common_HI)	0.8	EXP		
	send_no_feedback (UR_HI_HI)	0.8	EXP		
	send_no_instructions_to_actuator (UR_HI_HI)	0.8	EXP		
2	Access (operator_control_net)	0.06	EXP	8.3e-4	28
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	Send_false_feedback (Common_water_level)	0.7	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.01	EXP		
	send_false_feedback (UR_water_level)	0.7	EXP		
	send_false_feedback (Common_HI)	0.7	EXP		
	send_false_instructions_to_actuator (Common_HI)	0.7	EXP		
	send_false_feedback (UR_HI_HI)	0.7	EXP		
	send_false_instructions_to_actuator (UR_HI_HI)	0.7	EXP		
3	jamming_attack(operator_control_net)	1e-6	EXP	8.86e-5	0.3

Table 32: Malicious and accidental scenarios for variant 2 over 100 hours of operation

5.3.3.2 Long mission time without attack at t=0

The simulation of variant 2 over one year of mission time yields an estimation of the probability of the breach of 5.38e-2. We give in Table 33 the first minimal cut sets generated for this study.

During one year of the system operation, we find that the attack scenarios are again the most probable for this system architecture with a probability greater than 1e-3. Accidental scenarios are ten times less probable than attack scenarios; we find first the single points failures and next the accidental failure of the two controllers.

N° Seq.	Transitions			Pr	Contrib. (%)
	Name	Rate	Class		
1	Access (operator_control_net)	1e-4	EXP	9e-3	16
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	send_no_feedback (Common_water_level)	0.8	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.8	EXP		
	send_no_feedback (UR_water_level)	0.8	EXP		
	send_no_feedback (Common_HI)	0.7	EXP		
	send_no_instructions_to_actuator (Common_HI)	0.8	EXP		
	send_no_feedback (UR_HI_HI)	0.8	EXP		
	send_no_instructions_to_actuator (UR_HI_HI)	0.8	EXP		
2	jamming_attack(operator_control_net)	1e-6	EXP	5.3e-3	10
3	Access (operator_control_net)	1e-4	EXP	4e-3	7
	exploit_vuln_priv_escalation (Common_water_level)	0.01	EXP		
	Send_false_feedback (Common_water_level)	0.7	EXP		
	exploit_vuln_priv_escalation (UR_water_level)	0.7	EXP		
	send_false_feedback (UR_water_level)	0.7	EXP		
	send_false_feedback (Common_HI)	0.7	EXP		
	send_false_instructions_to_actuator (Common_HI)	0.7	EXP		
	send_false_feedback (UR_HI_HI)	0.7	EXP		
	send_false_instructions_to_actuator (UR_HI_HI)	0.7	EXP		
4	accidental_failure(PLC_pump)	1e-7	EXP	5e-4	0.9
5	accidental_failure(operator_control_net)	1e-7	EXP	5e-4	0.9
6	accidental_failure(Common_PLC)	1e-5	EXP	4e-5	0.07
	accidental_failure(UR_PLC)	1e-5	EXP		

Table 33: The minimal cut sets on variant 2

Hybrid scenarios composed of accidental and malicious events appear late in the list of the scenarios with a probability around 1e-5. We extract in Table 34 an example of a hybrid scenario obtained for variant 2 (without considering repairs).

This scenario is composed of an accidental part: the accidental failure of the HI_HI sensor, and a malicious part. In the malicious part, the attacker interrupts the feedback from the HI sensor by a DoS. Added to this, the failure of the HI_HI sensor would not trigger the emergency shutdown sent by the *UR_HI_HI* software component on the *UR_PLC*. This scenario can lead to safety issues only if it occurs in the case of water reaching maximum allowed levels; which imply that its probability estimation is pessimistic.

Num Seq.	Transitions			Proba	Contrib.
	Name	Rate	Class		
15	accidental_failure(S_HI_HI)	1e-5	EXP	2.7e-5	4.06e-4
	access(operator_control_net)	0.0001	EXP		
	exploit_vuln_priv_escalation(Common_water_level)	0.01	EXP		
	send_no_instructions_to_actuator(Common_HI)	0.8	EXP		
	send_no_feedback(Common_HI)	0.8	EXP		
	send_no_feedback(Common_water_level)	0.8	EXP		
	exploit_vuln_priv_escalation(UR_water_level)	0.01	EXP		
	send_no_feedback(UR_water_level)	0.8	EXP		

Table 34: A hybrid scenario on variant 2

5.3.4 Conclusions on variant 2

Again, results for this case study show that security related risks are predominant over accidental risks.

Over a short mission time, and considering that at t=0s an attacker intends to target the system in study, we can see that an attack scenario is the most likely to lead to the system breach, compared to accidental scenarios.

After one year of operation, with maintenance in case of accidental failures, attack scenarios remain at the top of the most probable scenarios likely to result into the system breach. We give in Figure 40, approximately, the repartition of scenarios according to their type (malicious, accidental or hybrid) for variant 2 over one year of operation.

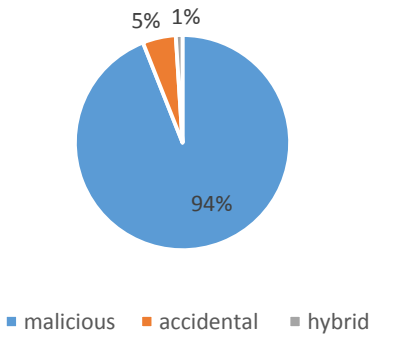


Figure 40: Repartition of scenarios according to their type for variant 2 over 1 year

Again, this demonstrates the vulnerability of industrial architectures to cyber-attacks that can result into safety issues and the importance of a risk analysis encompassing both safety and security issues.

5.4 Comparison between the two variants

We compare in this section the results obtained for the two variants of the case study for the different studies realized. Doing so, we act as if we had to design the system, and were looking for the best solution.

5.4.1 Pure safety analysis

In variant 1, the two primary controllers are identical and use the same hardware and software components. This solution is generally adopted for cost reduction purposes. However, it exposes the identical components to common cause failures. Results generated for variant 1 show that accidental scenarios resulting from common cause failures contribute the most to the undesirable event.

In variant 2, we considered two different types of controllers which eliminated such scenarios of common cause failures (however there are still common cause failures that can affect sensors). The overall breach probability decreased by 84% in this architecture compared to the previous one. This result is sufficiently good to justify the additional costs due to a diversification of the controllers.

5.4.2 Joint safety and security analysis

Over a short simulation time, assuming that at t=0 the attacker intends to target the system, attack scenarios are the most probable scenarios for both variants. The most probable attacks consist of accessing the wireless operator control network, compromising the controllers by exploiting vulnerabilities resulting into privilege escalation and then denying or falsifying the feedback and instructions exchanged between the controllers and the instrumentation ("denying attack" and falsifying attack").

Results show that the probability of attack scenarios for variant 1 are higher (by 37% for the "denying attack" and 35% for the falsifying attack") than for variant 2. This is due again to the fact that for the first architecture the controllers are identical. After succeeding in exploiting the vulnerability on the first software controller, the attacker can easily and quickly exploit the same vulnerability on the second software controller. However, for the second architecture and with two different controllers, the automation software is different and the attacker must find two different exploits, which decreases his chances to achieve a successful attack in a limited time.

Over one year of simulation time, and with a frequency of attack of 1e-4/h (approximately one attack per year), a joint analysis generates attack and failure scenarios with close orders of magnitude for safety and security risks. For the first system architecture, the common cause failure scenario resulting into the loss of both primary controllers is more probable than the denying/falsifying attacks. In the second system configuration, cyber-attacks are the most probable scenarios. The global probability of the undesirable event is 18% lower than in variant 1 given that predominant common cause failures scenarios are discarded and attack scenarios are more difficult thanks to the diversity of the controllers.

5.4.3 Safety and security interdependencies

The redundancy built in the design of the SIS, between the PLCs and the HI and HI_HI sensors, forms a safety mechanism that aims at enhancing the system reliability in case of accidental failure of one controller or one of the conductivity sensors.

When it comes to attack scenarios, and in order to provoke the loss of the protection function (which can result in the dam breach), the attacker must falsify/deny both instructions sent by *Common_HI* and the *UR_HI_HI* software components, in addition to instructions sent by the operator control, for the purpose of compromising the instruction sent to *PLC_pump* to disable the shutdown of the pumping unit. This takes additional effort from the attacker and reduces his chances of success in a given time. We infer that the redundancy between the HI and the HI-HI sensors, which consists in a dependability mechanism initially designed to reinforce the system safety, enables also to reinforce the system security and its resilience against malicious attacks and cyber malevolence.

5.5 Conclusion

We illustrated in this chapter the S-cube approach on different case studies: first on a "canonical" implementation of the PERA and next on a real industrial installation and its associated information and control architecture. For each case study, the system architecture has been graphically modeled and then processed which yields a qualitative and quantitative risk analysis. Thanks to the robustness of the S-cube approach, the different hypothesis and modifications in the system architecture have been incrementally taken into consideration in the graphical model. S-cube automatically generates the attack and failure scenarios related to each system configuration.

The qualitative and quantitative results are obtained based on the different hypothesis taken for types of failures and attacks and the level of detail modeled in the S-cube KB. The quantitative analysis is aimed at providing operators of the control systems with a ranking of various risks associated with failures and attacks in order to help them effectively manage their resources. The probabilities obtained should not be considered as definitive and accurate values.

On the case of the corporate network, we demonstrated how S-cube can be used in the exploitation phase in order to assess the incoming risks related to new vulnerabilities released or discovered.

On the case of the Dam, we showed how S-cube can be used in the design phase of the system in order to evaluate different configurations and decide on the safest and most secure one. We have also demonstrated how mechanisms designed for safety purposes can also enhance the system's security and its resilience against cyber-attacks, which are less predictable than the accidental failures but more likely to result into disastrous consequences.

Of course, S-cube can also be used in the design phase of corporate networks or in the operational phase of industrial control networks for ameliorating existing and operational industrial installations. On this last case, it would be useful for giving recommendations on the best safety and security practices and measures in order to counter or at less decrease the probability of successful attacks. S-cube can in general have the three following uses: evaluating a system architecture from scratch, evaluating the impact of changing something on the architecture (e.g. using an Ethernet based communication instead of a wireless communication for the operator control network) or the impact of adding (or removing) something in the architecture (e.g. adding an authentication mechanism to the operator control network, adding a firewall between the dispatch control network and the operator control network).

Chapter 6

Conclusions & Perspectives

6.1 Conclusions

We have been dealing in this dissertation, with approaches that consider together safety and security risks and requirements in the risk analysis of industrial systems. The research in the frame of this thesis has been undertaken in four main steps, each one bringing an original contribution:

1. The elaboration of a comprehensive survey of existing approaches, developed by both academic and industrial communities, that combine safety and security issues for the design and the evaluation of cyber-physical systems in general and industrial systems with a digital control in particular. A critical analysis has been led in order to classify these approaches and identify their shortcomings according to whether they enable formal modeling of the system and the associated safety and security requirements and whether they provide qualitative and quantitative risk analysis;
2. The investigation of the Boolean logic Driven Markov Processes formalism that satisfies both of the above criteria. The theoretical framework of BDMP for studying jointly safety and security had been already proposed in previous work but its application was limited to some simple didactic examples. We worked on illustrating the safety-security BDMPs on a realistic case study of a pipeline in order to show their ability to model safety and security risks in a common risk model and identify their potential interactions thanks to qualitative and quantitative results. The use of BDMP requires the detailed knowledge of the system architecture; this implies that BDMP are more relevant for assessing existing and operational systems. We compared BDMP with the CHASSIS method, which is used in early phases of designing systems, in order to investigate the complementarity of these two approaches.

   Our work with BDMP revealed three main limitations of this approach: first, the risk models are manually built by the analyst and subject to his/her point of view. Second these models are understood by other readers only if they know well the system architecture; third and finally, considering a different hypothesis about the system or a different undesirable event to study leads

to rebuild each time a new risk model with the same time and effort, which is tedious and error prone. More interest has been consequently given to an automatic and robust approach.

3. The development of the S-cube approach, overriding the limitations of existing approaches including BDMP. S-cube satisfies all the required criteria: formal modeling, qualitative and quantitative analysis, automatic generation of risk models, and robustness. Based on a knowledge base, S-cube allows modeling graphically the industrial architecture and the associated information and control system. This architecture is next processed, which automatically generates the attack and failures scenarios likely to happen on it.
   The S-cube knowledge base gathers knowledge on industrial control systems and includes a taxonomy of the attacks and failure modes likely to happen on these systems. These risk events are associated with probability distributions in order to dynamically model their evolution over time. The S-cube knowledge base has been implemented with the Figaro modeling language and the associated GUI and quantification tools.
4. The illustration of the S-cube approach on case studies. The S-cube approach has been validated on different case studies. First on a corporate network to model a multi-hop multi-stage attack starting by compromising the high level information system and reaching the industrial process. Second on a real case study of a pumped storage hydroelectric plant. For this second case, we have modeled graphically the system architecture. The robustness of S-cube allows to easily consider different configurations related to this architecture. Automating the generation of risk scenarios, encompassing safety-related risks and security-related risks, S-cube facilitates the production of risk analysis associated to each configuration. Attack and failure scenarios are sorted by decreasing contribution to the undesirable event they lead to. By comparing the qualitative and quantitative results associated with each system configuration, we can make conclusions on the safest and most secure one. The results enable also to identify the potential safety and security interactions.

The S-cube approach provides a robust risk analysis framework for modeling industrial information and control architectures and assessing the related safety and security risks. By applying it on a real case study of a hydroelectric plant, we demonstrated its ability to yield a holistic analysis encompassing safety and security risks on such a system and to assess the system resilience against accidental and malicious risks. Thanks to quantitative results, we have been also able to capture safety and security interdependencies.

S-cube aims at identifying the system weaknesses and vulnerabilities in order to support risk management and decision making. It can be used by safety and security engineers in different phases of the system lifecycle:

- Design phase to help designing new safe and secure systems with the appropriate safety and security requirements. S-cube helps the designer to assess and compare different configurations and safety/security mechanisms before converging towards an architecture where safety and security requirements are coordinated;

- Operation phase to assess the risk on existent systems, to help defining new safety and security mechanisms and to support crisis management. Indeed, S-cube helps risk analysts identify vulnerable components in the system architecture, predict undesirable events and avoid paths leading to them by choosing the appropriate patch and defense strategy.

We have been able to demonstrate in this thesis the advantages of S-cube and its applicability on large and complex systems. This approach can however be enhanced with different possible extensions.

6.2 Perspectives

The following extensions of S-cube can be envisaged for future work:

1. The S-cube KB models generically the control and information architectures of industrial systems. It can be coupled with existing knowledge bases that are specific to a given category of industrial systems and that enable to model, with more details, the physical process. There are existent libraries already developed in EDF in order to model hydraulic, ventilation and electrical systems. Coupling S-cube with such libraries would better reveal the impact of attacks on the physical process (e.g. power flow, water level). The possible safety and security interactions would also be better identified.
2. Including detection and reaction measures. The attack steps modeled in the S-cube KB can be associated with the appropriate detection and reaction measures that would prevent the attack step from being successful or just decrease its probability of success. This would be closer to reality, as security mechanisms can be deployed in the architecture in order to counter the possible attacks. This extension would model a completely repairable system, given that repair has already been modeled for accidental failure modes in the KB.
3. The quantitative analysis related to security can be improved in two ways:
    a. By working on the probability distributions associated with each attack step. By a statistical analysis of feedback of experience data, one can identify the probability distribution of the time needed to achieve each attack step (e.g. brute force, DoS, etc.). Exploiting this kind of data would imply the use of Yams instead of FigSeq for processing the model. The qualitative results would be the same as those obtained with the exponential approximation, while the quantitative results would be more accurate;
    b. By including uncertainties related to the quantitative metrics especially those related to security. Using an approach based on possibility theory enables to take into account uncertainties related to quantitative data. Two types of uncertainties are considered: aleatory uncertainty related to randomness of data and epistemic uncertainty related to lack of information. Instead of associating risk with a probability, these approaches consider an interval in which the risk probability is framed. This uncertainty is propagated to the output of the system model, which yields conservative results. Such an approach has already been applied in safety analyses [79][177]. Extended to security, where uncertainties are very present, it enables to associate each attack step with the interval of time required for the attacker to achieve the attack step [*min_time_required, max_time required*]. More details on the attacker profile could be also taken into consideration into security metrics.
4. Automating the filling of the data associated with the system architecture for more ease of use and a better exhaustiveness. These data can be extracted by a network analyzer that scans the network in order to identify the different machines connected and the services running on each host. Associated with a vulnerability data base (e.g. NVD, CVE), the scanner can identify the vulnerable software on the different machines. An add-on module can be developed in order to feed the data output by the scanner into the graphical model of the system architecture.

Finally, this work leverages a close collaboration between safety and security experts and will help them mutualize their efforts to reduce industrial risks and build safe and secure systems. It offers a first building block in the development of new industrial tools for risk management at EDF.

Annex 1: BDMP models of the Stuxnet attack

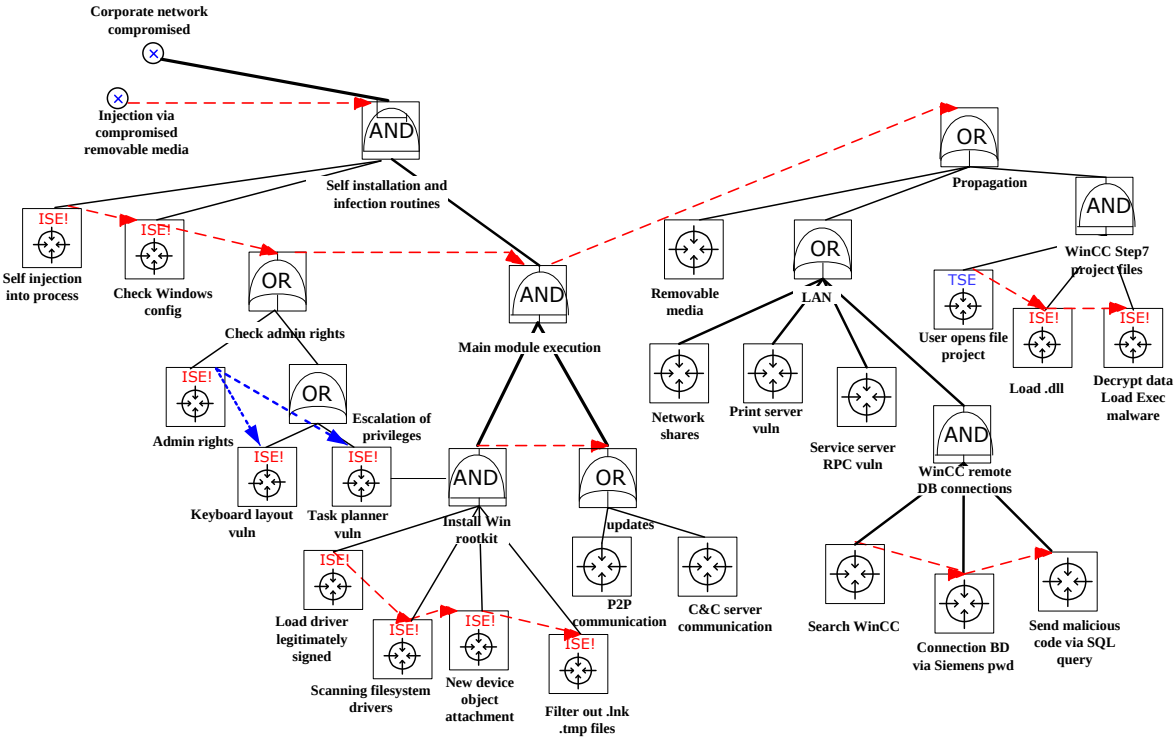


Figure 41: BDMP of the "self-installation and infection routines"

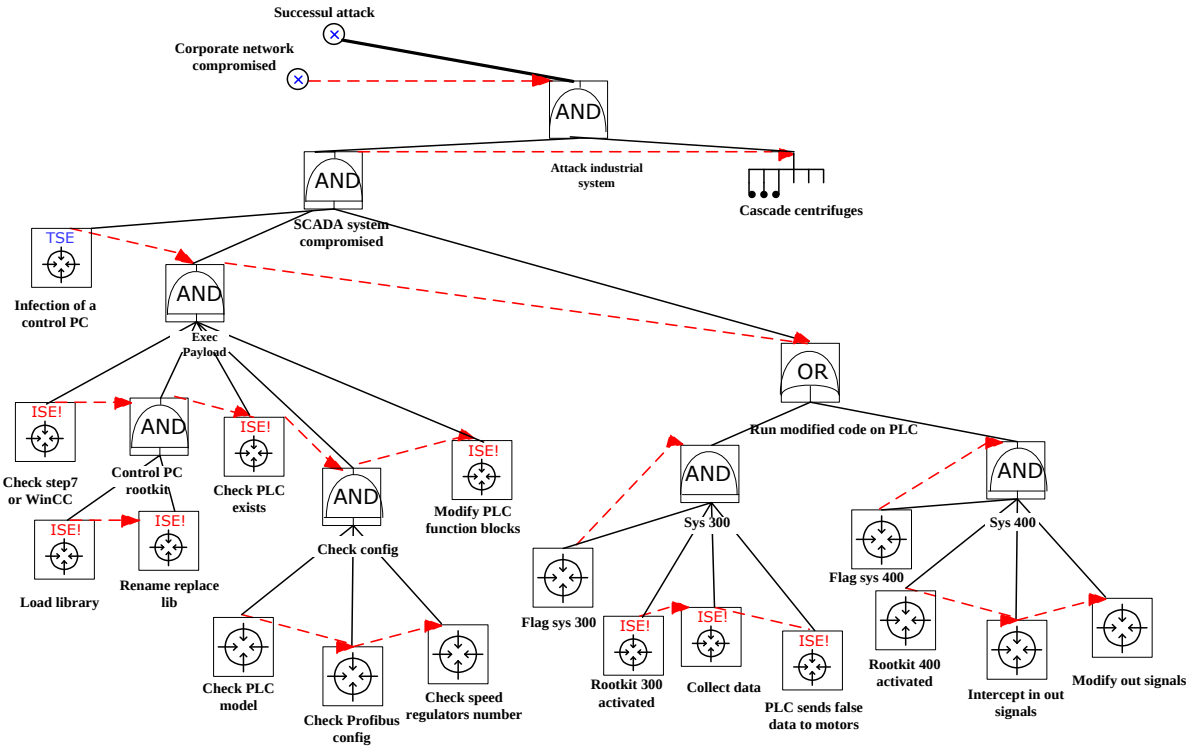


Figure 42: BDMP of the "attack industrial system phase"

Annex 2: Individual description of classes

We describe first the generic system components that can be found in classical information architectures. We focus next on the system components that are specific to SCADA and industrial architectures. The inheritance between the different classes and the other components with which they are interacting are depicted in the metamodel in Figure 28.

1. Generic classes

We model different machines (physical_cpt) connected to a network (network_zone) and hosting different services (software_cpt). Software components exchange different data flows (data_flow) and can host some vulnerabilities (vulnerability) that can be exploited by an attacker. In the following paragraphs, the key words extracted from the Figaro language are written in uppercase and the words in italic font are used in the S-cube KB.

CLASS component: the super class *Component* models a generic system component; which can fail accidentally or be compromised by an attacker. The component class has two types of risk events, expressed in the Figaro language using the key word FAILURE (cf. § 4.6.1). The first is the failure mode *accidental failure* which models an accidental failure of a component that can occur randomly with the rate *lambda_fail* (cf. Section 4.5.1 for quantitative aspects). The event *failure_repair* models the repair of the component's accidental failure which can occur with the rate *lambda_repair*.

The second risk event is the attack step *access*. A Boolean called *physical_access* is initially set by the user to TRUE or FALSE according to whether it is possible or not to physically access the component. If *physical_access* is set to TRUE, the *access* attack step can occur with the rate *lambda_access* and the component is compromised. The two Booleans *failure* and *compromised_host* (associated with EFFECTs in Figaro) model respectively the facts that the component is failed and compromised by an attacker. For instance, the Boolean *failure* is set to true when the risk event *accidental failure* occurs and is reset to false when *failure_repair* occurs.

The classes inheriting the characteristics this class are *network_zone*, *physical_cpt* and *gateway*.

CLASS network_zone: the *network_zone* class models a set of machines connected to one another using either wired or wireless communication technologies. It can be connected to one or many gateways and use an authentication mechanism. The accidental failure of the network zone can be due for example to the switch/hub failure. The *network_zone* class overrides the *access* attack step to condition it on whether the network is wired and can be accessed physically by the attacker or is wireless and no authentication mechanism is required. Other possible attacks defined for this specific class are:

- The jamming attack: can occur randomly for wireless networks. The effects of this attack step realization are propagated by the interaction rules; where all data sent from components connected to the network zone are *unavailable*;
- Bypassing authentication: in case an authentication is required to be connected to the network, the attacker can try to bypass this authentication in order to try to compromise some machines connected to the network.
- Scan network: if the network zone is connected to another network zone by a gateway and no firewalling is enabled between both network, an attacker located at the first network can launch a scan on the other network in order to identify living hosts and open ports;
- Establish connection: if the attacker identifies vulnerable machines and services further to a network scan, he/she can try to establish connection to the network. If he/she succeed this attack step, the network is compromised.

We make the assumption that if there is a compromised host connected to the network, this latter is also said to be compromised. A compromised network models the fact that the attacker can reach any other machine located at this network.

CLASS Gateway: models a gateway linking two or many network zones. The attribute *firewall_enabled* is set to true if the firewalling functionality is enabled. If a network_zone is connected to another compromised network_zone through the gateway and no firewalling is enabled, then the attacker can scan the network, in order to identify living hosts and open ports.

In the S-cube KB, open ports are modeled by server applications, in listening mode, receiving no data flows (cf. CLASS data_flow). If no firewalling is enabled the attacker can try to establish a connection with an open port. If he succeeds in doing so, the network is then compromised.

CLASS physical_cpt: we make the distinction in the S-cube KB between the physical components (*physical_cpt*) that represent the network machines and the software components (*software_cpt*) running on these machines. The **link_machine_soft** is used to associate the machine with the different software components running on it.

A physical component belongs to a network zone, hosts one or many software components and can use an authentication mechanism, required to access its operating system (OS). Being KIND_OF component, the physical component inherits the accidental failure, which once realized results into unavailability for all data flows sent from software components hosted on the failed machine.

Attack steps associated with this particular class are: access, bypassing authentication, and compromising the communication link. This latter models a MITM attack where the attacker interferes with the communication between two machines, in order to falsify or deny the data exchange between them.

We assume that when a physical_cpt is compromised then all the software components running on it are also compromised.

CLASS CCF_group: models a set of identical physical components, subject to the same environmental constraints, that be prone to failures which can occur simultaneously or close in time, due to a common cause (cf. § 4.5.1.2 and Figure 31).

CLASS software_cpt: this class models a software component running on a given host, which can be for instance a server or client application. A software_cpt can send and receive data from other software components. Two interfaces *out_data* and *in_data* model respectively input and output data flows of a given software_cpt.

We assume in our KB that software cannot fail. It can however be altered by an attacker; in this case, the Boolean *compromised_software* switches to TRUE.

CLASS data_flow: this class models the authorized data flows between software components (e.g., client and server applications). A source and a destination is associated with each data flow.

Particularly for control data flows (data having a direct impact on the physical process, cf. § 4.3.1.4), we distinguish in the S-cube KB between two types: "*instructions*" and "*feedback*". For other data flows, the *data_type* is "*other*".

The Boolean "*carried_by_electrical_wire*" is set to true when an electrical wire (e.g., twisted pair) is used to carry a specific data flow, between two components. Indeed, in classical control architectures, each sensor/actuator is linked directly to the process controller via this kind of link. With such a configuration, the attacker has to access physically and individually each link in order to modify/deny the data passing on it.

We propagate the two effects related to data flows: unavailability and alteration, as result of attacks and failures, across the whole system architecture.

In the S-cube KB, the data-flows represented in the graphical system model represent the authorized data flows between software components (client and server applications). Open ports are modeled by server applications without receiving any data flows (in listening mode). If no firewalling is enabled the attacker can try to establish a connection with an open port. If he succeeds in doing so, the network is then compromised.

CLASS IT_sys_cpt: models a machine integrating ICT and providing advanced functionalities; it runs a given Operating System that can be either Windows or Linux in our KB. Used traditionally in high levels of the system information architecture, such kind of machines are being also used in control and process levels in modern architectures. The attribute *privilege* models the default privilege that the legitimate user has on this machine. *Vuln_configuration* models a vulnerability associated with a bad configuration of the machine. An attacker can exploit such vulnerability, if it exists, to gain root privileges on the machine (*privilege_escalation_attack*).

CLASS vulnerability: models a vulnerability related to an IT software; which can have one or many consequences among the following: *privilege escalation, confidentiality loss, integrity loss or service denial*.

CLASS IT_soft_cpt: models a software component running on an *IT_sys_cpt*. IT software component can use an authentication mechanism and can have many or no vulnerabilities. In order to compromise this software, the attacker should first bypass authentication, if it exists. The four other generic attack steps associated with this class model exploiting vulnerabilities with the four possible consequences of a vulnerability. Interaction rules propagate the effect of data being altered in case of privilege escalation or integrity loss and the effect of data being unavailable in case of service denial.

CLASS soft_type: models a category of software type (e.g., http server, ftp client) associated with an IT_soft_cpt. If a software component of type *soft_type* is compromised the Boolean *already_hacked* of this class is set to true. The probability of exploiting vulnerability on another software component belonging to the same *soft_type* decreases considerably. This models the fact that an attacker that have already succeeded in exploiting a vulnerability associated to a given software will spend much less time to succeed into exploiting it again with another software of the same type.

CLASS attacker: models an attacker which is initially located on a physical machine. This latter is consequently a compromised host.

CLASS authentication_mechanism: models an authentication mechanism used by a system component which can be a machine, a network zone or an IT service. The attribute *type* indicates whether the authentication mechanism is weak or strong. The MTTS the attack step "bypassing authentication" is calculated according to the authentication type (higher probability of success in the case of weak authentication).

2. SCADA-specific classes

We describe in this section the components that are specific to SCADA-based systems, cf. Section 4.3.1 for more details on the industrial architecture specificities.

Sensors: CLASS sensor KIND_OF field_sys_cpt and CLASS sensor_soft_cpt KIND_OF software_cpt

The class ***sensor*** models the physical part of the sensor which is placed close to the industrial process. We recall that the main functionality of a sensor is to measure physical quantities and communicate them to the controller. If the attacker physically access the sensor, then the sensor is compromised.

The class ***sensor_soft_cpt*** models the embedded software in a sensor that enables to capture a physical quantity and transfer it to the upper levels of the control process. If the sensor enables to capture more than one physical quantity (e.g., temperature and pressure) then each physical quantity is modeled by a separated *sensor_soft_cpt*.

The attack steps modeled for this class are "*sending false measures"* or "*sending no measures"*. This attack step can occur if the *sensor_soft_cpt* is compromised (e.g., physical access to the sensor) or in case of the communication link on which measures are sent is compromised (e.g., MITM attack on the network or physical access to the electrical wire carrying the measure). If the attacker succeeds in achieving the attack step "*send false measure*" the interaction rules propagate the effects, where all data flows of type "*feedback*" are "*wrong*".

Actuators: CLASS actuator KIND_OF field_sys_cpt and CLASS actuator_soft_cpt KIND_OF software_cpt:

The class ***actuator*** models the physical part of an actuator. It can for example model a valve or a pump. We recall that an actuator is a system component that executes a direct action on the process after receiving a control instruction.

The ***actuator_soft_cpt*** class models the software embedded in the actuator that receives the instruction. The interaction rules in this class model the fact that if the actuator does not act properly it can be because either it receives wrong or no instruction from the controller.

CLASS process_controller: models the physical machine that is used for controlling the process. It can be for example a programmable logical controller (PLC) or a remote telemetry unit (RTU).

The ***process_controller_soft_cpt*** class inherits the characteristics of an IT software component and models the software embedded in a *process_sys_cpt*. It enables to collect sensor measures, send instructions to actuators or receives instructions from higher control levels (each functionality can be modeled by a single *process_controller_soft_cpt*). The attack steps modeled for this class are: "sending false instructions to actuator", "sending no instructions to actuator", "sending false feedback" and "sending no feedback", which can occur either if the *process_controller_soft_cpt* is compromised or the communication link on which the instructions/feedback are sent is compromised. The interaction rules model the fact that the process_controller_soft_cpt can either receive wrong resp. no measures from sensors or receive wrong resp. no instructions from the Scada server. The output data is then *wrong* resp. un*available*.

CLASS scada_server_soft_cpt inherits the characteristics of an IT software component and models the software used to remotely supervise the overall process. IT receives the information from the process controller, processes it and sends back instructions (each functionality can be modeled by a single *scada_server_soft_cpt*). The attack steps modeled for this class are: "sending false instructions", "sending no instructions", "sending false feedback" and "sending no feedback"; which may occur if the scada server software is compromised or the communication link on which instructions/feedback are sent is compromised.

3. Representation of voters

We describe in this section classes graphically represented as logical gates that can be used to represent voters present in the system. Since the inputs and outputs of these gates are not simple Boolean values, but instead data flows containing the two effects wrong and unavailable, the behavior of the voters is a bit more complicated than just logical functions.

CLASS AND gate: makes an AND operation between two or many data flows. The output data flows are correct if all the input data flows are the same. If it exists an input data flow which is *wrong/unavailable* then the output data flows are *wrong/unavailable*.

CLASS OR gate: makes an OR operation between two or many data flows. The output data flows are correct if it exists a correct input data flow. If all the input data flows are *wrong/unavailable* then the output data flows are *wrong/unavailable*.

CLASS k_n_gate: this gate receives n data flows and outputs a correct data flow if k over the n input data flows are the same. This class inherits from the software component class and models a safety mechanism that makes a vote out of many inputs received.

For instance, the *k_n_gate* receives measures from sensors and if at least k measures out of n data are identical it sends the measure to the controller. This voter can also input different instructions from controllers and make a decision on what instruction to send to an actuator. We focus especially, in the S-cube KB, on describing the behavior of the 2/3 and 2/4 voters.

4. Other classes

These classes do not model system components and are not visible in the graphical model but are used by other classes that are used to model the system.

CLASS global: this class is implemented in any model by a single global object that has two Booleans:

- Inhibit_attacks: when set to true by the user, this Boolean inhibits taking into consideration attacks into the risk scenarios;
- Inhibit_failures: when set to true by the user, this Boolean inhibits taking into consideration accidental failures into the risk scenarios;

Consequently, this class allows the user to choose whether to consider in the risk analysis only accidental failures (i.e. pure safety analysis), only malicious actions (i.e. pure security analysis) or both of them (i.e. joint safety and security risk analysis).

Annex 3: Assumptions on the quantitative metrics for the use case: pumped storage plant

Type	Object	Family	Characteristic	Value
CCF_group_2	CCF_group2_PLC	Constant	alpha_2	0.05 (for variant 1)
CCF_group_3	CCF_group_3_sensors	Constant	alpha_2	0.1 (for variant 1)
CCF_group_3	CCF_group_3_sensors	Constant	alpha_3	0.05 (for variant 1)
k_n_gate	k_n_gate_measure	Constant	k, n	2, 3
network_zone	operator_control_net	Constant	lambda_access	0.06 (for short mission time study) 0.0001 (for long mission time study)
process_controller	Common_PLC	Constant	lambda_fail	1e -05
process_controller	PLC_pump	Constant	lambda_fail	1e-07
process_controller	UR_PLC	Constant	lambda_fail	1e-05
IT_sys_cpt	Operator_control_center	Constant	lambda_fail	1e-05
IT_sys_cpt	dispatch_control_center	Constant	lambda_fail	1e-05
sensor	S_HI	Constant	lambda_fail	1e-05
sensor	S_HI_HI	Constant	lambda_fail	1e-05
sensor	pressure1	Constant	lambda_fail	1e-05
sensor	pressure2	Constant	lambda_fail	1e-05
sensor	pressure3	Constant	lambda_fail	1e-05
network_zone	dispatch_control_net	Constant	lambda_fail	1e-05
network_zone	operator_control_net	Constant	lambda_fail	1e-05
actuator	pumping_unit	Constant	lambda_fail	1e-10
process_controller_soft_cpt	Common_water_level	Attribute	lambda_priv_escal	0.01 (if not already hacked) 0.7 (if already hacked)
process_controller_soft_cpt	UR_water_level	Attribute	lambda_priv_escal	0.01 (if not already hacked) 0.7 (if already hacked)

References

[1] N. Falliere, L. O. Murchu, and E. Chien, “W32.Stuxnet Dossier (v1.4).” Symantec report, Feb-2011.

[2] “Dell Security Annual Threat Report,” 2015.

[3] L. Pièrre-Cambacédès, “Des relations entre sûreté et sécurité,” Télécom ParisTech, 2010.

[4] “BlackHat USA 2013.” [Online]. Available: http://www.blackhat.com/us-13/briefings.html#Forner.

[5] T. J. Cockram and S. R. Lautieri, “Combining Security and Safety Principles in Practice,” in *2007 2nd Institution of Engineering and Technology International Conference on System Safety*, 2007, pp. 159–164.

[6] P. Bieber, J.-P. Blanquart, G. Descargues, M. Dulucq, Y. Fourastier, E. Hazane, M. Julien, L. Léonardon, and G. Sarouille, “Security and Safety Assurance for Aerospace Embedded Systems,” in *Proceedings of the 6th International Conference on Embedded Real Time Software and Systems (ERTS2 2012), Toulouse, France*, 2012, pp. 1–3.

[7] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, “Experimental Security Analysis of a Modern Automobile,” in *2010 IEEE Symposium on Security and Privacy (SP)*, 2010, pp. 447–462.

[8] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, “Comprehensive Experimental Analyses of Automotive Attack Surfaces.,” in *USENIX Security Symposium*, 2011.

[9] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, “Survey on security threats and protection mechanisms in embedded automotive networks,” in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, 2013, pp. 1–12.

[10] J. Smith, S. Russell, and M. Looi, “Security as a safety issue in rail communications,” in *Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33*, Darlinghurst, Australia, Australia, 2003, pp. 79–88.

[11] T. Novak, A. Treytl, and A. Gerstinger, “Embedded security in safety critical automation systems,” in *Proceedings of the 26th International System Safety Conference (ISSC 2008)*, Vancouver, Canada, 2008, pp. p. S.1–11,.

[12] *The SeSa Method for Assessing Secure Remote Access to Safety Instrumented Systems*. Trondheim: SINTEF, Technology and Society, Safety and Reliability, 2006.

[13] S. O. Johnsen, “Resilience at interfaces: Improvement of safety and security in distributed control systems by web of influence,” *Inf. Manag. Comput. Secur.*, vol. 20, no. 2, pp. 71–87, Jun. 2012.

[14] A. Lee and T. Brewer, “Smart grid cyber security strategy and requirements,” *Draft Interag. Rep. NISTIR*, vol. 7628, 2009.

[15] “Security and Safety Modelling - D2.1 Specification of Safety and Security Mechanisms,” v01, 29 May 2013 Final.

[16] “Security and Safety Modelling - D3.1 Specification of Safety and Security Analysis and Assessment Techniques,” v01, 29 May 2013 Final.

[17] A. Burns, J. McDermid, and J. Dobson, “On the Meaning of Safety and Security,” *Comput. J.*, vol. 35, no. 1, pp. 3–15, Feb. 1992.

[18] L. Pièrre-Cambacédès and C. Chaudet, “The SEMA referential framework: Avoiding ambiguities in the terms ‘security’ and ‘safety,’” *Int. J. Crit. Infrastruct. Prot.*, vol. 3, no. 2, pp. 55–66, Jul. 2010.

[19] "ISO/TR 31004: 2013: Risk management — Guidance for the implementation of ISO 31000," Oct. 2013.
[20] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, 2004.
[21] D. P. Eames and J. D. Moffett, "The Integration of Safety and Security Requirements," in *Proceedings of the 18th International Conference on Computer Computer Safety, Reliability and Security*, London, UK, UK, 1999, pp. 468–480.
[22] A. Kornecki and M. Liu, "Fault Tree Analysis for Safety/Security Verification in Aviation Software," *Electronics*, vol. 2, no. 1, pp. 41–56, Jan. 2013.
[23] K. Sørby, "Relationship between security and safety in a security-safety critical system: Safety consequences of security threats," NTNU, Trondheim, Norway, MSc thesis, 2003.
[24] N. G. Leveson, *Safeware: system safety and computers*. New York, NY, USA: ACM, 1995.
[25] D. Brewer, F. Redmill, and T. Anderson, "Applying Security Techniques to Achieving Safety," in *Directions in Safety-Critical Systems*, Springer London, 1993, pp. 246–256.
[26] L. Piètre-Cambacédès and M. Bouissou, "Cross-fertilization between safety and security engineering," *Reliab. Eng. Syst. Saf.*, vol. 110, pp. 110–126, Feb. 2013.
[27] D. G. Firesmith, "Common Concepts Underlying Safety Security and Survivability Engineering," Dec. 2003.
[28] B. Hunter, "Integrating Safety And Security Into The System Lifecycle," in *Improving Systems and Software Engineering Conference (ISSEC)*, Canberr, Australia, 2009, p. 147.
[29] T. Aven, "Identification of safety and security critical systems and activities," *Reliab. Eng. Syst. Saf.*, vol. 94, no. 2, pp. 404–411, Feb. 2009.
[30] T. Novak, A. Treytl, and P. Palensky, "Common approach to functional safety and system security in building automation and control systems," in *IEEE Conference on Emerging Technologies and Factory Automation, 2007. ETFA*, 2007, pp. 1141–1148.
[31] F. Reichenbach, J. Endresen, M. M. R. Chowdhury, and J. Rossebo, "A Pragmatic Approach on Combined Safety and Security Risk Analysis," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2012, pp. 239–244.
[32] L. Piètre-Cambacédès and M. Bouissou, "Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes)," in *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, 2010, pp. 2852–2861.
[33] T. Novak and A. Gerstinger, "Safety- and Security-Critical Services in Building Automation and Control Systems," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3614–3621, 2010.
[34] T. Novak and A. Treytl, "Functional safety and system security in automation systems - a life cycle model," in *IEEE International Conference on Emerging Technologies and Factory Automation, 2008. ETFA 2008*, 2008, pp. 311–318.
[35] M. Sun, S. Mohan, L. Sha, and C. Gunter, "Addressing Safety and Security Contradictions in Cyber-Physical Systems," in *1st Workshop on Future Directions in Cyber-Physical Systems Security (CPSSW'09)*, Newark, United states, 2009.
[36] A. Derock, "Convergence of the latest standards adressing safety and security for information technology," in *in On-line proceedings of Embedded Real Time Software and Systems (ERTS2 2010)*, Toulouse, France, 2010.
[37] J. T. Williams, "A Reference Model For Computer Integrated Manufacturing (CIM) A Description from the Viewpoint of Industrial Automation," 1989.

[38] International Electrotechnical Commission, *Enterprise-control system integration. Part 1, Part 1,*. 2013.
[39] J.-M. Brun, L. Platel, and F. Tea, “Cyber Security of Industrial Control System Why ICS specificity lead to Cyber Security Challenge?,” in *C&ESAR*, 2013.
[40] B. Zhu, A. Joseph, and S. Sastry, “A Taxonomy of Cyber Attacks on SCADA Systems,” in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 380–388.
[41] K. Stoufer, J. Falco, and K. Scarfone, “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),” NIST SP 800-82, Jun. 2011.
[42] L. Pietre-Cambacedes, Y. Fourastier, and F. Téa, *Cybersécurité des installations industrielles*, Cépaduès. 2015.
[43] “IEC 61508-1:2010: Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1 : general requirements.” 01-Apr-2010.
[44] P.-C. Ludovic, “Cyber Security of Nuclear Instrumentation & Control Systems: Overview of the IEC Standardization Activities,” 2013, pp. 2156–2160.
[45] “IEC 62645 - DRAFT - Draft Document - Nuclear power plants - Instrumentation and control systems - Requirements for security programmes for computer-based systems (IEC 45A/890/CDV:2012).”
[46] “ISO/IEC 27001:2013, Information technology — Security techniques — Information security management systems — Requirements.” .
[47] “ISO/IEC 27005:2011 - Information technology -- Security techniques -- Information security risk management.” .
[48] C. Raspotnig, P. Karpati, and V. Katta, “A Combined Process for Elicitation and Analysis of Safety and Security Requirements,” in *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2012, pp. 347–361.
[49] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, “A survey of approaches combining safety and security for industrial control systems,” *Reliab. Eng. Syst. Saf.*, vol. 139, pp. 156–178, Jul. 2015.
[50] G. Stoneburner, “Toward a Unified Security-Safety Model,” *Computer*, vol. 39, no. 8, pp. 96–97, 2006.
[51] T. Aven, “A unified framework for risk and vulnerability analysis covering both safety and security,” *Reliab. Eng. Syst. Saf.*, vol. 92, no. 6, pp. 745–754, Jun. 2007.
[52] C. Woskowski, “A Pragmatic Approach towards Safe and Secure Medical Device Integration,” in *Computer Safety, Reliability, and Security*, vol. 8666, A. Bondavalli and F. Di Giandomenico, Eds. Cham: Springer International Publishing, 2014, pp. 342–353.
[53] C. W. Johnson, “CyberSafety: On the Interactions Between CyberSecurity and the Software Engineering of Safety-Critical Systems,” in *In C. Dale and T. Anderson (eds.), Achieving System Safety*, 2012, pp. 85–96.
[54] A. J. Kornecki and J. Zalewski, “Safety and security in industrial control,” in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, New York, NY, USA, 2010, pp. 77:1–77:4.
[55] K. Stølen, F. den Braber, T. Dimitrakos, R. Fredriksen, B. A. Gran, S.-H. Houmb, Y. C. Stamatiou, and J. Ø. Aagedal, “Model-Based Risk Assessment in a Component-Based Software Engineering Process,” in *Business Component-Based Software Engineering*, F. Barbier, Ed. Springer US, 2003, pp. 189–207.

[56] R. Winther, O.-A. Johnsen, and B. A. Gran, “Security Assessments of Safety Critical Systems Using HAZOPs,” in *Computer Safety, Reliability and Security*, U. Voges, Ed. Springer Berlin Heidelberg, 2001, pp. 14–24.
[57] P. A. Ostby and M. S. Strauch, “Topical Report on Security and Safety Integration,” Safety and Security Interface Technology Initiative, Sep. 2006.
[58] C. Schmittner, T. Gruber, P. Puschner, and E. Schoitsch, “Security Application of Failure Mode and Effect Analysis (FMEA),” in *Computer Safety, Reliability, and Security*, vol. 8666, A. Bondavalli and F. Di Giandomenico, Eds. Cham: Springer International Publishing, 2014, pp. 310–325.
[59] Y. Deswarte, M. Kaâniche, P. Corneillie, and J. Goodson, “SQUALE Dependability Assessment Criteria,” in *Computer Safety, Reliability and Security*, M. Felici and K. Kanoun, Eds. Springer Berlin Heidelberg, 1999, pp. 27–38.
[60] “Medini Functional Safety Tool.” [Online]. Available: http://www.kpit.com/engineering/products/medini-functional-safety-tool.
[61] T. Kelly and R. Weaver, “The goal structuring notation–a safety argument notation,” in *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*, 2004.
[62] K. Attwood and P. Chinneck, “GSN Community Standard Version 1.” Nov-2011.
[63] S. Lautieri, D. Cooper et D. Jackson, “SafSec : Commonalities between safety and security assurance,” in *Proceedings of the 13th Safety Critical Systems Symposium (SSS’05)*, Southampton, Royaume-Uni, 2005, pp. p. 65–75.
[64] C. Johnson, “Using Assurance Cases and Boolean Logic Driven Markov Processes to Formalise Cyber Security Concerns for Safety-Critical Interaction with Global Navigation Satellite Systems,” *Electron. Commun. EASST*, vol. 45, no. 0, Dec. 2011.
[65] N. Subramanian and J. Zalewski, “Assessment of safety and security of system architectures for cyberphysical systems,” in *Systems Conference (SysCon), 2013 IEEE International*, 2013, pp. 634–641.
[66] L. Chung and J. C. S. do P. Leite, “On Non-Functional Requirements in Software Engineering,” in *Conceptual Modeling: Foundations and Applications*, A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, Eds. Springer Berlin Heidelberg, 2009, pp. 363–379.
[67] I. Nai Fovino, M. Masera, and A. De Cian, “Integrating cyber attacks within fault trees,” *Reliab. Eng. Syst. Saf.*, vol. 94, no. 9, pp. 1394–1402, Sep. 2009.
[68] B. Schneier, “Attack trees : Modeling security threats,” *Dr.Dobb’s Journal*, pp. 21–29, 1999.
[69] S. Bezzateev, N. Voloshina, and P. Sankin, “Joint Safety and Security Analysis for Complex Systems,” in *PROCEEDING OF THE 13TH CONFERENCE OF FRUCT ASSOCIATION*, 2013.
[70] M. Steiner and P. Liggesmeyer, “Combination of Safety and Security Analysis-Finding Security Problems That Threaten The Safety of a System,” in *Proceedings of Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, 2013.
[71] B. Kaiser, P. Liggesmeyer, and O. Mäckel, “A new component concept for fault trees,” in *Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33*, Darlinghurst, Australia, Australia, 2003, pp. 37–46.
[72] M. Bouissou and J.-L. Bon, “A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes,” *Reliab. Eng. Syst. Saf.*, vol. 82, no. 2, pp. 149–163, Nov. 2003.

[73] L. Piètre-Cambacédès and M. Bouissou, “Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP),” in *Dependable Computing Conference (EDCC), 2010 European*, 2010, pp. 199–208.
[74] C.-W. Ten, C.-C. Liu, and G. Manimaran, “Vulnerability Assessment of Cybersecurity for SCADA Systems,” *IEEE Trans. Power Syst.*, vol. 23, no. 4, pp. 1836–1846, 2008.
[75] R. Mitchell and I. Chen, “Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems,” *IEEE Trans. Reliab.*, vol. 62, no. 1, pp. 199–210, 2013.
[76] F. Flammini, U. Gentile, S. Marrone, R. Nardone, and V. Vittorini, “A Petri Net Pattern-Oriented Approach for the Design of Physical Protection Systems,” in *Computer Safety, Reliability, and Security*, A. Bondavalli and F. D. Giandomenico, Eds. Springer International Publishing, 2014, pp. 230–245.
[77] M. Roth and P. Liggesmeyer, “Modeling and Analysis of Safety-Critical Cyber Physical Systems using State/Event Fault Trees,” in *Proceedings of Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*, Toulouse, France, 2013, p. NA.
[78] D. R. Duran, E. Robinson, A. J. Kornecki, and J. Zalewski, “Safety analysis of Autonomous Ground Vehicle optical systems: Bayesian belief networks approach,” in *2013 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2013, pp. 1419–1425.
[79] C. Simon, P. Weber, and E. Levrat, “Bayesian Networks and Evidence Theory to Model Complex Systems Reliability,” *Journal of Computer, VOL. 2*, pp. 33–43, Feb-2007.
[80] P. Trucco, E. Cagno, F. Ruggeri, and O. Grande, “A Bayesian Belief Network modelling of organisational factors in risk analysis: A case study in maritime transportation,” *Reliab. Eng. Syst. Saf.*, vol. 93, no. 6, pp. 845–856, Jun. 2008.
[81] N. Poolsappasit, R. Dewri, and I. Ray, “Dynamic Security Risk Management Using Bayesian Attack Graphs,” *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012.
[82] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, “Using Bayesian networks for cyber security analysis,” in *2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2010, pp. 211–220.
[83] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, “Measuring Network Security Using Dynamic Bayesian Network,” in *Proceedings of the 4th ACM Workshop on Quality of Protection*, New York, NY, USA, 2008, pp. 23–30.
[84] A. J. Kornecki, N. Subramanian, and J. Zalewski, “Studying interrelationships of safety and security for software assurance in cyber-physical systems: Approach based on bayesian belief networks,” in *2013 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2013, pp. 1393–1399.
[85] G. Sindre and A. L. Opdahl, “Capturing security requirements through misuse cases,” in *In: Proceedings of the 14th Norwegian informatics conference (NIK)*, 2001.
[86] G. Sindre and A. L. Opdahl, “Eliciting security requirements with misuse cases,” *Requir. Eng.*, vol. 10, no. 1, pp. 34–44, Jun. 2004.
[87] G. Sindre, “A look at misuse cases for safety concerns,” in *Situational Method Engineering: Fundamentals and Experiences*, Springer, 2007, pp. 252–266.
[88] C. Raspotnig, “Guideline for applying CHASSIS.” 2012.
[89] “Guideline_for_applying_CHASSIS_draft_BORA.pdf.” .
[90] J. Jürjens, “UMLsec Extending UML for secure Systems Development.” The Unified Modeling Language (2002), 1-9.

[91] J. Jürjens and S. H. Houmb, “Risk-driven development of security-critical systems using UMLsec,” in *Information Technology*, Springer, 2004, pp. 21–53.
[92] J. Jürjens and J. Grünbauer, “Critical systems development with UML: overview with automotive case-study,” in *4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2003)*, pp. 512–517.
[93] J. Jürjens, “Developing safety-and security-critical systems with UML,” in *DARP workshop, Loughborough*, 2003.
[94] S. Wenzel, “CARiSMA - a tool for analyzing compliance-, risk- and security properties on software models.” [Online]. Available: https://www-secse.cs.tu-dortmund.de/carisma/.
[95] L. Apvrille and Y. Roudier, “Towards the Model-Driven Engineering of Secure yet Safe Embedded Systems,” in *Pre-proceedings of The International Workshop on Graphical Models for Security 2014*, 2014.
[96] G. Pedroza, L. Apvrille, and D. Knorreck, “Avatar: A sysml environment for the formal verification of safety and security properties,” in *New Technologies of Distributed Systems (NOTERE), 2011 11th Annual International Conference on*, 2011, pp. 1–10.
[97] J. Brunel, D. Chemouil, L. Rioux, M. Bakkali, and F. Vallée, “A Viewpoint-Based Approach for Formal Safety & Security Assessment of System Architectures,” in *Proceedings of MoDeVVa 2014*, 2014.
[98] “Melody™ - SysML plugin for IBM Rational Rhapsody,” *Intercax*. .
[99] “Safety Architect | Model-Based Safety Analysis.” [Online]. Available: http://www.all4tec.net/Model-Based-Safety-Analysis/safety-architect.html.
[100] “Alloy: a language & tool for relational models.” [Online]. Available: http://alloy.mit.edu/alloy/.
[101] J. Delange, L. Pautet, and P. Feiler, “Validating Safety and Security Requirements for Partitioned Architectures,” in *Reliable Software Technologies – Ada-Europe 2009*, F. Kordon and Y. Kermarrec, Eds. Springer Berlin Heidelberg, 2009, pp. 30–43.
[102] S. Zafar and R. G. Dromey, “Integrating safety and security requirements into design of an embedded system,” in *Software Engineering Conference, 2005. APSEC ’05. 12th Asia-Pacific*, 2005, p. 8 pp.–.
[103] J.-C. Laprie, K. Kanoun, and M. Kaâniche, “Modelling Interdependencies Between the Electricity and Information Infrastructures,” in *Computer Safety, Reliability, and Security*, F. Saglietti and N. Oster, Eds. Springer Berlin Heidelberg, 2007, pp. 54–67.
[104] M. Beccuti, G. Franceschinis, S. Donatelli, S. Chiaradonna, F. Di Giandomenico, P. Lollini, G. Dondossola, and F. Garrone, “Quantification of dependencies in electrical and information infrastructures: The CRUTIAL approach,” in *Fourth International Conference on Critical Infrastructures, 2009. CRIS 2009*, 2009, pp. 1–8.
[105] S. Chiaradonna, F. D. Giandomenico, and P. Lollini, “Case Study on Critical Infrastructures: Assessment of Electric Power Systems,” in *Resilience Assessment and Evaluation of Computing Systems*, K. Wolter, A. Avritzer, M. Vieira, and A. van Moorsel, Eds. Springer Berlin Heidelberg, 2012, pp. 365–390.
[106] M. Beccuti, G. Franceschinis, M. Kaâniche, and K. Kanoun, “Multi-level Dependability Modeling of Interdependencies between the Electricity and Information Infrastructures,” in *Critical Information Infrastructure Security*, R. Setola and S. Geretshuber, Eds. Springer Berlin Heidelberg, 2008, pp. 48–59.
[107] “CRUTIAL - CRitical UTility InfrastructurAL resilience.” [Online]. Available: http://crutial.rse-web.it/.

[108] R. Winther, O.-A. Johnsen, and B. A. Gran, “Security Assessments of Safety Critical Systems Using HAZOPs,” in *Computer Safety, Reliability and Security*, vol. 2187, U. Voges, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 14–24.
[109] J. E. Y. Rossebo, S. Cadzow, and P. Sijben, “eTVRA, a Threat, Vulnerability and Risk Assessment Method and Tool for eEurope,” in *The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007*, 2007, pp. 925–933.
[110] D. K. Holstein and B. Singer, “Quantitative Security Measures for Cyber and Safety Security,” in *ISA Safety & Security Symposium*, 2010.
[111] J. Depoy, J. Phelan, P. Sholander, B. Smith, G. B. Varnado, and G. Wyss, “Risk assessment for physical and cyber attacks on critical infrastructures,” in *IEEE Military Communications Conference, 2005. MILCOM 2005*, 2005, pp. 1961–1969 Vol. 3.
[112] W. Pieters, Z. Lukszo, D. Hadziosmanovic, and J. van den Berg, “Reconciling Malicious and Accidental Risk in Cyber Security,” *J. Internet Serv. Inf. Secur. JISIS*, vol. 4, no. 2, pp. 4–26, 2014.
[113] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, “The Maude 2.0 System,” in *Rewriting Techniques and Applications*, vol. 2706, R. Nieuwenhuis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 76–87.
[114] A. Simpson, J. Woodcock, and J. Davies, “Safety through Security,” in *Proceedings of the 9th international workshop on Software specification and design*, Washington, DC, USA, 1998, p. 18–.
[115] J. Goguen and J. Meseguer, “Security policies and security models,” in *Proceedings of the IEEE Symposium on Security and Privacy (S&P’82)*, Oakland, United-states, 1982, pp. p. 11–20.
[116] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)*. The MIT Press, 2012.
[117] D.-A. Lee, J.-S. Lee, S.-W. Cheon, and J. Yoo, “Application of System-Theoretic Process Analysis to Engineered Safety Features-Component Control System,” in *Proc. of the 37th Enlarged Halden Programme Group (EHPG) meeting*, Gol, Norway, 2013.
[118] P. Asare, J. Lach, and J. A. Stankovic, “FSTPA-I: A Formal Approach to Hazard Identification via System Theoretic Process Analysis,” 2013.
[119] W. Young and N. G. Leveson, “An integrated approach to safety and security based on systems theory,” *Commun. ACM*, vol. 57, no. 2, pp. 31–35, Feb. 2014.
[120] W. Young and N. G. Leveson, “Systems Thinking for Safety and Security,” in *Annual Computer Security Applications Conference*, New Orleans, LA, 9-13 December.
[121] C. Poirier, S. Kriaa, F. Pebay-Peyroula, C. Mraidha, and V. Zille, “A tool for I&C system architecture design: the French Connexion cluster,” in *International Symposium on Future I&C for Nuclear Power Plants / International Symposium on Symbiotic Nuclear Power Systems 2014*, Republic of Korea, 2014.
[122] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, “DAG-based attack and defense modeling: Don’t miss the forest for the attack trees,” *Comput. Sci. Rev.*
[123] L. Pietre-Cambacedes, Y. Deflesselle, and M. Bouissou, “Security Modeling with BDMP: From Theory to Implementation,” in *2011 Conference on Network and Information Systems Security (SAR-SSI)*, 2011, pp. 1–8.
[124] L. Piètre-Cambacédès and M. Bouissou, “Attack and Defense Modeling with BDMP,” in *Computer Network Security*, I. Kotenko and V. Skormin, Eds. Springer Berlin Heidelberg, 2010, pp. 86–101.
[125] M. Bouissou, “KB3 tool: feedback on knowledge bases,” in *Proceedings of the Annual European Safety and Reliability Conference*, 2002.

[126] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, and L. Pietre-Cambacedes, “Safety and Security Interactions Modeling Using the BDMP Formalism: Case Study of a Pipeline,” in *Computer Safety, Reliability, and Security*, vol. 8666, A. Bondavalli and F. Di Giandomenico, Eds. Cham: Springer International Publishing, 2014, pp. 326–341.
[127] M. Bouissou and Y. Lefebvre, “A path-based algorithm to evaluate asymptotic unavailability for large Markov models,” in *Reliability and Maintainability Symposium, 2002. Proceedings. Annual*, 2002, pp. 32–39.
[128] S. Kriaa, M. Bouissou, and L. Pietre-Cambacedes, “Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments,” in *2012 7th International Conference on Risk and Security of Internet and Systems (CRiSIS)*, 2012, pp. 1–8.
[129] A. Matrosov, E. Rodionov, D. Herley, and J. Malcho, “Stuxnet under the microscope (v1.0),” presented at the EST, 2011, pp. 1–85.
[130] M. Ekstedt, G. Dondossola, L. Pietre-Cambacedes, J. McDonald, and Å. Torkilseng, “Modelling of cyber attacks for assessing smart grid security,” presented at the Cigre Study Committee D2 Colloquium. Buenos Aires, Argentina. 19th - 20th October 2011, 2011.
[131] J. R. Nielsen, “Evaluating Information Assurance Control Effectiveness on an Air Force Supervisory Control and Data Acquisition (SCADA) System,” Mar. 2011.
[132] E. Byres, A. Ginter, and J. Langill, “How Stuxnet spreads, a study of infection paths in best parctice systems (v1.0).” Tofino Security, Feb-2011.
[133] D. Helan, “Stuxnet. analysis, myths, realities,” *Actu Secu*, Feb-2011.
[134] S. Kriaa, C. Raspotnig, M. Bouissou, L. Pièrre-Cambacédès, P. Karpati, Y. Halgand, and V. Katta, “Comparing Two Approaches to Safety and Security Modelling BDMP Technique and CHASSIS Method,” in *Proc. of the 37th Enlarged Halden Programme Group (EHPG) meeting*, Gol, Norway, 2013.
[135] H. Holm, T. Sommestad, M. Ekstedt, and L. Nordstörm, “CySeMoL: A tool for cyber security analysis of enterprises,” in *Electricity Distribution (CIRED 2013), 22nd International Conference and Exhibition on*, 2013, pp. 1–4.
[136] X. Ou, S. Govindavajhala, and A. W. Appel, “MulVAL: A Logic-based Network Security Analyzer.,” in *USENIX security*, 2005.
[137] B. Blanchet, “Automatic Verification of Correspondences for Security Protocols,” *J. Od Comput. Secur.*, vol. 17, no. 4, pp. 363–434, 2009.
[138] H. Holm, M. Ekstedt, T. Sommestad, and M. Korman, “A manual for the Cyber Security Modeling Language.” 28-May-2014.
[139] “AltaRica Project | MEthods and Tools for AltaRica Language.” .
[140] T. Prosvirnova, “AltaRica 3.0: a Model-Based approach for Safety Analyses,” phdthesis, Ecole Polytechnique, 2014.
[141] M. Bouissou and C. Seguin, “Comparison of the modeling languages AltaRica and Figaro,” in *In proceedings of the 14th congress on reliability and maintenability (IMDR)*, Lille, France, 2006.
[142] “O3PRM: Open Object Oriented Probabilistic Relational Models.” [Online]. Available: http://o3prm.lip6.fr/.
[143] S. Kriaa, M. Bouissou, and Y. Laarouchi, “A Model Based Approach for SCADA Safety and Security joint Modeling: S-cube,” in *IET System Safety and Cyber Security*, Bristol, 2015.
[144] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, and L. Pietre-Cambacedes, “Safety and Security Interactions Modeling Using the BDMP Formalism: Case Study of a Pipeline,” in *Computer Safety, Reliability, and Security*, vol. 8666, A. Bondavalli and F. Di Giandomenico, Eds. Cham: Springer International Publishing, 2014, pp. 326–341.
[145] *Ethical Hacking and Countermeasures (CEH v7.1)*, EC-Council. .

[146] S. Hauge, P. Hokstad, S. Håbrekke, and M. A. Lundteigen, "Common cause failures in safety-instrumented systems: Using field experience from the petroleum industry," *Reliab. Eng. Syst. Saf.*, vol. 151, pp. 34–45, Jul. 2016.
[147] C. L. Atwood, "The Binomial Failure Rate Common Cause Model," *Technometrics*, vol. 28, no. 2, pp. 139–148, May 1986.
[148] G. Apostolakis and P. Moieni, "The foundations of models of dependence in probabilistic safety assessment," *Reliab. Eng.*, vol. 18, no. 3, pp. 177–195, 1987.
[149] A. Mosleh, K. N. Fleming, G. W. Parry, H. M. Paula, D. H. Worledge, and D. M. Rasmuson, "Procedures for Treating Common Cause Failures in Safety and Reliability Studies: Volume 2, Analytic Background and Techniques: Final Report," Electric Power Research Inst., Palo Alto, CA (USA); Pickard, Lowe and Garrick, Inc., Newport Beach, CA (USA), EPRI-NP-5613-Vol.2, Dec. 1988.
[150] R. Donat and M. Bouissou, "Common Cause Failures in Discrete Dynamic Models: Theory and Applications in the Figaro Modelling Language," in *In proceedings of the 25th European Safety and Reliability Conference (ESREL)*, Zürich, 2015.
[151] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Trans. Softw. Eng.*, vol. 25, no. 5, pp. 633–650, Sep. 1999.
[152] G. V. Marconato, M. Kaâniche, and V. Nicomette, "A Vulnerability Life Cycle-Based Security Modeling and Evaluation Approach," *Comput. J.*, p. bxs112, Sep. 2012.
[153] E. Jonsson and T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior," *Softw. Eng. IEEE Trans. On*, vol. 23, no. 4, pp. 235–245, 1997.
[154] H. Holm, "A Large-Scale Study of the Time Required to Compromise a Computer System," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 1, pp. 2–15, Jan. 2014.
[155] H. Holm, M. Korman, and M. Ekstedt, "A Bayesian network model for likelihood estimations of acquirement of critical software vulnerabilities and exploits," *Inf. Softw. Technol.*, vol. 58, pp. 304–318, Feb. 2015.
[156] M. Dacier, Y. Deswarte, and M. Kaâniche, "Models and tools for quantitative assessment of operational security," in *Information Systems Security*, S. K. Katsikas and D. Gritzalis, Eds. Springer US, 1996, pp. 177–186.
[157] M. Dacier, Y. Deswarte, and M. Kaaniche, "Quantitive assessment of operational security models and tools," Technical Report Research Report 96493, May 1996.
[158] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel, "Time-to-compromise model for cyber risk reduction estimation," in *Quality of Protection*, Springer, 2006, pp. 49–64.
[159] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Published by FIRST-Forum of Incident Response and Security Teams*, 2007, pp. 1–23.
[160] C. Cocozza-Thivent, *Processus stochastiques et fiabilité des systèmes*. Springer Science & Business Media, 1997.
[161] "CVE details (The ultimate security vulnerability datasource)." [Online]. Available: www.cvedetails.com.
[162] Y. W. Law, P. Hartel, J. den Hartog, and P. Havinga, "Link-layer jamming attacks on S-MAC," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, 2005, pp. 217–225.
[163] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 46–57.

[164] T. Sommestad, H. Holm, and M. Ekstedt, “Estimates of success rates of Denial-of-Service attacks,” in *Proceedings of the 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2011)*, 2011, pp. 21–28.
[165] M. Bouissou, H. Bouhadana, M. Bannelier, and N. Vilatte, “Knowledge modeling and reliability processing: presentation of the FIGARO language and of associated tools,” in *proceedings of SAFECOMP 91*, Trondheim, Norway, 1991.
[166] M. Bouissou, “Automated dependability analysis of complex systems with the KB3 workbench: the experience of EDF R&D,” in *Proceedings of the International Conference on ENERGY and ENVIRONMENT, CIEM 2005*, 2005.
[167] M. Bouissou, S. Humbert, and J.-C. Houdebine, “Reference handbook of the Figaro modeling language.” EDF.
[168] M. Bouissou, L. Buffoni, and J.-C. Houdebine, “Syntax of the Figaro modeling language.” EDF.
[169] M. Bouissou and J.-C. Houdebine, “Inconsistency detection in KB3 tools,” in *ESREL 2002*.
[170] P. G. Harrison, “Laplace Transform Inversion and Passage-Time Distributions in Markov Processes,” *J. Appl. Probab.*, vol. 27, no. 1, pp. 74–87, 1990.
[171] M. Bouissou, “A simple yet efficient acceleration technique for Monte Carlo simulation,” in *The 22nd annual European Safety and Reliability Conference ESREL*, Amsterdam, 2013.
[172] “Pumped storage hydroelectric power station.” [Online]. Available: http://www.bbc.co.uk/bitesize/standard/physics/energy_matters/generation_of_electricity/revision/3/.
[173] S. Kriaa, M. Bouissou, and Y. Laarouchi, “SCADA Safety and Security joint modeling (S-cube): case study of a dam,” in *Proceedings of the 22th Computer & Electronics Security Applications Rendez-vous (C&ESAR’2015)*, Rennes, France, 2015, pp. 55–69.
[174] Rogers and Watkins, “Overview of the Taum Sauk Pumped Storage Power Plant Upper Reservoir Failure,” *6th Int. Conf. on Case Histories in Geotechnical engineering*, Arlington, VA, 11-Aug-2008.
[175] Before the public service commission, state of Missouri, “Staff’s initial incident report.,” Case No. ES-2007-0474, Oct. 2007.
[176] FERC Taum Sauk Investigation Team, “Report of Findings on the Overtopping and Embankment Breach of the Upper Dam - Taum Sauk Pumped Storage Project,” FERC No. 2277, Apr. 2006.
[177] N. Pedroni, E. Zio, E. Ferrario, A. Pasanisi, and M. Couplet, “Hierarchical propagation of probabilistic and non-probabilistic uncertainty in the parameters of a risk model,” *Comput. Struct.*, vol. 126, pp. 199–213, Sep. 2013.