UNIVERSITÀ
DEGLI STUDI
FIRENZE

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

Dottorato di Ricerca in Informatica, Sistemi e Telecomunicazioni
Indirizzo: Informatica e Applicazioni
Settore Scientifico Disciplinare: INF/01
Ciclo XXVII
(Anni 2012/2014)

# MODEL-BASED APPROACHES TO DEPENDABILITY AND SECURITY ASSESSMENT IN CRITICAL AND DYNAMIC SYSTEMS

NICOLA NOSTRO

Supervisors:

*Prof. Andrea Bondavalli*

*Dr. Felicita Di Giandomenico*

*PhD Coordinator: Prof. Luigi Chisci*

*December 2014*

Nicola Nostro: *Model-based Approaches to Dependability and Security Assessment in Critical and Dynamic Systems,* Dottorato di Ricerca in Informatica, Sistemi e Telecomunicazioni, Ciclo XXVII, Università degli Studi di Firenze. © December 2014.

## ABSTRACT

Everyday life is more and more dependent on the services provided by highly complex and pervasive critical infrastructures, whose failures might lead to catastrophic consequences in terms of damages to human life, environment, economy.

The thesis presents a work based on the study and research of topics related to resilience and security of complex systems, which present characteristics of heterogeneity, dynamicity, evolvability, interdependencies, interconnections, criticality with respect to the domain of application. Nowadays, the study of such systems and their characteristics is of paramount importance, because they are becoming more and more widespread, affecting our lives and our way of life. For this reason, it is crucial to design such systems with an high level of resilience and security. Indeed, their failures could lead, in the worst case, to catastrophic consequences, for instance if we consider the Critical Infrastructures.

Through stochastic model-based approaches, the thesis addresses the system evaluation to support design decision at pre-deployment and run-time, both from resilience and security point of view. The thesis is structured in three main parts.

Initially the focus is on the Electrical Power Systems, specifically on the modelling and assessment of dependability and performance requirements by means of model-based approaches and on the investigation of techniques and mechanisms to apply in order to tolerate unexpected events, failures, or attacks, thus allowing the continuity of the service. The thesis then addresses the study of new modelling approaches for the assessment of systems showing aspects of evolution and dynamicity. Indeed, current systems are more and more conceived as dynamically adaptable and evolvable sets of components, which must be able to modify their behaviour at run-time to tackle the continuous changes happening in the unpredictable open-world settings. In particular the focus is on new approaches useful to combine the benefits of both classic pre-deployment (off-line) analysis, and run-time analysis, by providing solutions that enable a refinement and an improvement of the system model. Finally, the thesis also focuses on aspects of security related to complex systems. Indeed, due to the increasing presence of automatic controlling operation, the massive use of networks to transfer data and information, and the human operations, new security concerns in such systems have been introduced and cannot be neglected. Security issues do not only have direct impact on systems availability, integrity and confidentiality, but they also can influence the dependability, specifically the safety of critical systems. In this context the thesis mainly focus on security

issue related to the insider threats, which are one of the major cause of security violations.

More in detail the dissertation is organized as follows. Chapter 1 introduces the context of the work, describing the characteristics of complex systems that we consider in the thesis, along with an overview of some real incidents that have affected such systems, in order to highlight their criticality.

Chapter 2 provides an overview of the main model-based approaches to support system design decision with respect to aspects of dependability, performance and security. The last part of the chapter presents the contribution of the thesis.

Chapter 3 describes the first part of the thesis with the focus on critical infrastructures, in particular on Electrical Power Systems. The chapter describes an existing modelling framework to support system design decision and a proposed extension, which allows to perform interesting analysis accounting for heterogeneous aspects of the Electrical Power Systems. The analysis results are also presented in the chapter.

Chapter 4 describes the characteristics of dynamic, evolvable and interoperable systems and provides a model-based approach for the assessment at pre-deployment time. The proposed approach allows, at pre-deployment time, the enhancement of the system model in order to satisfy dependability and performance requirements; and at run-time, the refinement of the model through monitoring of system parameters.

Chapter 5 deals with security aspects of the systems, specifically related to the insider threats. The chapter proposes a specific methodology for the security assessment of the system through model-based approaches.
Finally, the thesis concludes with an overview of the overall work in the study of complex systems by exploiting model-based approaches to support the assessment activities. The concluding chapter also provides a description of the research challenges that deserve to be further investigated.

## RELATED PUBLICATIONS

This thesis is partially based on work included in the following publications:

**I.** S. Chiaradonna, F. Di Giandomenico, and N. Nostro. Analysis of electric power systems accounting for interdependencies in heterogeneous scenarios. In *Ninth European Dependable Computing Conference (EDCC 2012)*, May 2012.

**II.** Silvano Chiaradonna, Felicita Di Giandomenico, and Nicola Nostro. Model-based assessment of multi-region electric power systems showing heterogeneous characteristics. In Frank Ortmeier and Peter Daniel, editors, *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, pages 328-339. Springer Berlin Heidelberg, 2012.

**III.** A. Bertolino, A. Calabrò, F. Di Giandomenico, N. Nostro, P. Inverardi, and R. Spalazzese. On-the-fly dependable mediation between heterogeneous networked systems. In *ICSOFT 2011, CCIS 303*, pages 20-37. Springer-Verlag Berlin Heidelberg, 2013.

**IV.** S. Chiaradonna, F. Di Giandomenico, and N. Nostro. Stochastic Assessment of Power Systems in Presence of Heterogeneity. *International Journal of Critical Computer-Based Systems (IJCCBS )*, Vol.4 Issue 4, pages 326-348, February 2013.

**V.** F. Di Giandomenico, A. Bertolino, A. Calabrò, and N. Nostro. An approach to adaptive dependability assessment in dynamic and evolving connected systems. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, Vol. 4 Issue 1, pages 1-25, 2013.

**VI.** N. Nostro, A. Ceccarelli, A. Bondavalli, and F. Brancati. A methodology and supporting techniques for the quantitative assessment of insider threats. In *Proceedings of the 2Nd International Workshop on Dependability Issues in Cloud Computing*, DISCCO '13, pages 1-6, New York, NY, USA, 2013. ACM.

**VII.** N. Nostro, A. Ceccarelli, A. Bondavalli, and F. Brancati. Insider threat assessment: A model-based methodology. *ACM SIGOPS Operating Systems Review*, Vol. 48 Issue 2, pages 3-12, July 2014.

**VIII.** F. Di Giandomenico, M.L. Itria, P. Masci, and N. Nostro. Automated synthesis of dependable mediators for heterogeneous interoperable systems. International Journal *Reliability Engineering & System Safety*, Vol. 132, pages 220-232, December 2014.

The following publications are related to the topic of the thesis as well, but where published before the beginning of the PhD course:

**IX.** S. Chiaradonna, F. Di Giandomenico, and N. Nostro. Modeling and analysis of the impact of failures in electric power systems organized in interconnected regions. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN)*, pages 442-453. June 2011.

**X.** P. Masci, N. Nostro and F. Di Giandomenico. On Enabling Dependability Assurance in Heterogeneous Networks through Automated Model-Based Analysis. In *Proceedings of the Third International Workshop, SERENE 2011*. pages 78-92, 2011.

Finally, the following work, published within the PhD course, is only marginally related to the topics of this thesis:

**XI.** N. Nostro, A. Bondavalli and N. Silva. "Adding Security Concerns to Safety Critical Certification". *2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 521-526, 2014.

**XII.** N. Nostro, I. Matteucci, A. Ceccarelli, F. Di Giandomenico, F. Martinelli and A. Bondavalli. On Security Countermeasures Ranking through Threat Analysis. In Andrea Bondavalli, Andrea Ceccarelli, and Frank Ortmeier, editors, *Computer Safety, Reliability, and Security*, volume 8696 of *Lecture Notes in Computer Science*, pages 243-254. Springer International Publishing, 2014.

# ACKNOWLEDGMENTS

First of all I would like to thank my supervisors, Prof. Andrea Bondavalli and Dr. Felicita Di Giandomenico, for their continuous support in my research activities. Their experience has always allowed me to identify the right way to pursue my objectives.

I would also thank Prof. Rosario Pugliese for his precious advise since the first day as internal contact.

Thanks go to the current and past members of the Resilience Computing Lab research group, Paolo Lollini, Andrea Ceccarelli, Francesco Brancati, Andreia Rossi for the collaborations and interesting discussions. Especially, I would like to thank Leonardo Montecchi, who often helped me like an older brother.

I thank Nuno Silva for the very interesting, fruitful and constructive discussions we had during the two months I spent at CRITICAL Software in Coimbra.

Thank you very much to all the people of the PhD room for having made the work environment more pleasant and stimulating. I thank all the people that I met during these three years, that in someway have contributed to the achievement of my goal.

Thanks to my friends, Francesco, David, Leonardo, Michele, Ottavio... that make my life more beautiful.

A special thank is form my parents and my sisters that are always close to me in all my life decision.

Finally, I say thanks to Martina who has lived with me every moment, from the simplest to the most difficult, patiently and with love.

— *Nicola Nostro*

# CONTENTS

# 1

# COMPLEX SYSTEMS

The increasing of technological, financial and social development of industrialized countries, is strongly related to the proper functioning of complex technological systems. Such infrastructures are complex and highly interdependent systems, which provide essential services in our everyday life.

We live in a world where systems are more and more composed by networks of heterogeneous devices and components from different producers, thus leading to potential issues due to evolution and connectivity. Such systems are increasingly pervasive, dynamic and heterogeneous, as a consequence several aspects of modern society rely on their continuous availability and interoperability.

The concept of *complex systems* has led to several definitions according to the reference domain (physics, engineering, mathematics, computer science, and so on). In general terms, the common point among the various definitions in literature, is that they are systems made up of interconnected parts, which together constitute the global integrated entity. The aspect of complexity refers to properties of the systems which are not easily and directly understandable from the "simple" description and properties of the single parts.
The study of complex systems implies all the activities aiming at investigating how the relationship among the parts of the system, may lead to proper or improper behaviour of the global system and how it may interact with the environment in which it is utilized [25].

Complex systems contribute to the design, production and delivery of several products and services that we run across in our daily activities, and their relevance will increase in coming years.
Among such systems, characterized by being decisive in our society, we can surely find those systems defined and referred to as Critical Infrastructure (CI). Critical Infrastructures are systems whose failure or disruption could lead to catastrophic consequences from the economic or, in the worst case, from the human safety point of view. Indeed, a malfunctioning of similar infrastructures, although limited to a short period of time, could lead to negative consequences, causing economic losses or even putting at risk the human safety [140]. They are in general represented by facilities and assets, including water supply, wastewater treatment, flood-reduction structures, telecommunications, power grids, nuclear plants, transportation, financial, health, etc.

The increasing dependence upon critical infrastructures, has lead to an unavoidable expansion of their complexity, due to the growing demand for new services and products by a growing population. Moreover, the conditions under which such systems are called to operate are continuously evolving, introducing

possible unexpected problems, and thus requiring a higher resilience. In fact, in the past they were used to provide services most in isolation, without or with limited interconnections with each other, so they could be damaged locally, with no impact or consequences on other infrastructures. Nowadays, the scenario has changed significantly, and several infrastructures, in order to provide services, cooperate through strong networking, mainly based on information technology systems.

Critical infrastructure may cross political boundaries and may be *built*, *natural*, or *virtual*. **Built** critical infrastructure includes energy; water and wastewater treatment, distribution, and collection; transportation; and communications systems. **Natural** critical infrastructure systems include lakes, rivers, and streams that are used for navigation, water supply, or flood water storage, as well as coastal wetlands that provide a buffer for storm surges. **Virtual** critical infrastructure includes cyber, electronic, and information systems.
Critical Infrastructure Protection (CIP) is therefore a priority for most of the countries and several initiatives are in place to identify open issues and research viable solutions in this highly challenging area, especially to identify vulnerabilities and devise survivability enhancements on critical areas.

## 1.1 CHARACTERISTICS OF COMPLEX SYSTEMS

Complexity can be a significant obstacle to successful design of critical systems. It is characterized by various aspects, like dimension of the systems, data, diversity, heterogeneity, etc. As a result, complex systems may be potentially susceptible to failures or attacks. Furthermore, aspects of uncertainty may also be the basis of incorrect design and implementation of the systems, which can often be observed in the non-fulfillment of non-functional requirements such as performance, dependability and security. They are generally characterized by the difficulties of understanding a dynamic integrated set of people, devices, processes, technologies [9].

In order to ensure that these systems continue to operate in a proper way, which means satisfying the functional and non-functional requirements, despite to their aspects of complexity, it is important to provide additional supporting activities, like for instance:

- Modelling and quantitative assessment.
- Processes of improvement in order to deal with evolving aspects of the systems.
- Monitoring processes in order to face their dynamic nature.
- Risk assessment and identification of proper security solutions.

- Management of uncertainty.

## 1.2 CHARACTERISTICS OF CRITICAL INFRASTRUCTURES

The increasing introduction of network management, monitoring and control systems, on the one hand has certainly improved the performance level of such infrastructures, on the other, it has also introduced risk of failure and new vulnerabilities (e.g., new unexpected undesired behaviour or access to cyber criminals). Therefore, the scenario has become more and more complex in recent years, as the introduction of advanced technologies added new sources of potential risk alongside the traditional threats. An effective infrastructure protection includes threats identification, vulnerability reduction and attack source or damage origin identification. This activity aims at service downtime minimization and damage limitation.
Another relevant aspect, characterizing CIs, is represented by the interactions and interdependencies among critical infrastructures.

*An **interdependency** is a bidirectional relationship between two infrastructures through which the state of each infrastructure influences or is correlated to the state of the other*. More generally, two infrastructures are interdependent when each is dependent on the other [112].

Hence, understanding and analyzing interdependencies is an additional critical step during the design and the maintenance of such systems, since they might be a source of threats and contribute to risk uncertainty.
Infrastructure interdependencies can be categorized according to various dimensions in order to facilitate their identification, understanding and analysis. Among the most important dimensions identified in [112], there are:

- **physical:** Two infrastructures are physically interdependent if the state of each is dependent on the material output(s) of the other;
- **cyber:** An infrastructure has a cyber interdependency if its state depends on information transmitted through the information infrastructure;
- **geographic:** Infrastructures are geographically interdependent if a local environmental event can create state changes in all of them;
- **logical:** Two infrastructures are logically interdependent if the state of each depends on the state of the other via a mechanism that is not a physical, cyber, or geographic connection.

Interdependencies increase the vulnerability of the corresponding infrastructures as they give rise to multiple error propagation channels from one infrastructure to another that increase their exposure to accidental, as well as to malicious, threats. Consequently, the impact of component failures in critical infrastructures can be exacerbated due to interdependencies and the overall

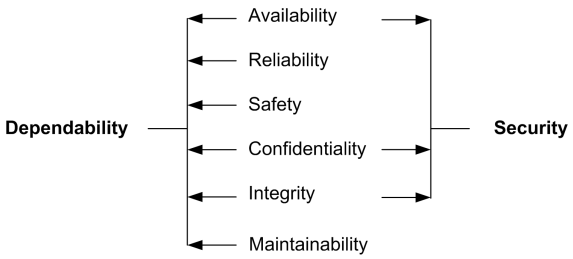


Figure 1.: Relationship between dependability and security.

severity of a failure is generally much larger and more difficult to foresee, compared to failures confined to single infrastructures.

The following subsections provide the basic concepts and definitions regarding dependable and secure systems, in order to make the reader familiar with the concepts covered. Further details can be found in [23], [86], and [87].

### 1.2.1 *Dependability and Security: Basic Concepts and Definitions*

As developed over the past three decades, dependability is an integrating concept that encompasses the following attributes: *availability*, *reliability*, *safety*, *integrity*, and *maintainability*. **Dependability** is defined as *the ability to deliver service that can justifiably be trusted* [87]. When addressing security aspects of a system an additional attribute has great prominence, which is *confidentiality*, i.e., the absence of unauthorized disclosure of information. **Security** is a composite of the attributes of *confidentiality*, *integrity*, and *availability*. Figure 1 summarizes the relationship between dependability and security in terms of their main attributes.

**Resilience** is defined as *the persistence of service delivery that can justifiably be trusted, when facing changes*. Equivalently, referring to the concept of dependability, *resilience* is also defined as *the persistence of dependability when facing changes* [87].

The *changes* can be classified according to three viewpoints, or dimensions:

- Their *nature*, which can be: functional, environmental, or technological (hardware and software).
- Their *prospect*, which can be: foreseen, foreseeable, unforeseen.
- Their *timing*, which can be: short term, medium term, long term.

It has to be emphasized, in the context of dependability, that the changes can concern, or induce changes in the threats the system is facing. The threat changes can have their source in the changes to the system or its environment, taken either *i)* in isolation, such as, for technological changes, the ever increasing proportion of transient hardware faults that goes along with the progress of integration, or

*ii)* in combination, such as the ever evolving and growing problem of attacks both by amateur hackers and by professional criminals, that may result from environmental and technological changes. Finally, the changes can themselves turn into threats, as in the case of mismatches between the modifications that implement the changes and the former status of the system.

### 1.2.2 *Accidental faults*

Analyzing interdependencies allows a greater understanding of the effects of failures. Three types of failures are of particular interest when analyzing interdependent infrastructures: *cascading failures*, *escalating failures*, and *common cause failures*.

CASCADING FAILURES: occur when a failure in one infrastructure causes the failure of one or more component(s) in a second infrastructure.

ESCALATING FAILURES: occur when an existing failure in one infrastructure exacerbates an independent failure in another infrastructure, increasing its severity or the time for recovery and restoration from this failure.

COMMON CAUSE FAILURES: occur when two or more infrastructures are affected simultaneously because of some common cause.

Besides analyzing the types of failures, it is important to understand the different causes that might lead to the occurrence of such failures. Faults and their sources may be very different, as discussed in [23]. They can be classified according to different criteria: the phase of creation (development vs operational faults), the system boundaries (internal vs external faults), the phenomenological cause (natural vs human-made faults), the dimension (hardware vs software faults), the persistence (permanent vs transient faults), the objective of the developer or the human interacting with the system (malicious vs accidental faults), their intent (deliberate vs non-deliberate faults), their capability (accidental vs incompetence faults). Accidental faults remain an important source of failures that may affect both the physical and the cyber aspects of CIs.

### 1.2.3 *Malicious faults*

*Malicious faults* are introduced with the intentional purpose of altering the proper functioning of a system during its employment. The objectives of such faults can be different:

- to disrupt or halt service, causing denials of service;
- to access confidential information;

- to improperly modify the system.

They are grouped into two classes:

1. **Malicious logic faults** that encompass development faults (e.g., Trojan horses, logic or timing bombs, and trapdoors), as well as operational faults (e.g., viruses, worms, or zombies) [86].

2. **Intrusion attempts** that are operational external faults. The external character of intrusion attempts does not exclude the possibility that they may be performed by system operators or administrators who are exceeding their rights, the so-called *insiders*.

## 1.3 CRITICAL INFRASTRUCTURES ACCIDENTS

In the global scenario, the history is sadly full of real events and accidents to critical infrastructures, which have caused great economic losses and/or injuries or death of human beings. In the following we provide a brief overview of some meaningful events occurred in the past. They are split in two sections, considering both the accidental, and intentional failures, in order to make clear what are the real risks that may occur due to a failure or disruption of a critical infrastructure.

### 1.3.1 *Critical Infrastructures - Accidental Failures*

In the telecommunication domain, we can recall the failure of a node of the Telecom Italia[1] infrastructure in the city of Rome, occurred in 2004, which caused the interruption of the mobile and wired telephone traffic of a large area of Rome for several hours. The event had large impacts on several sectors like: banking, postal, transportation, and so on [17].
Most serious was the Italian black-out occurred during the night of September 28, 2003. The event left all the country without electrical power for a period of time ranging from three to twenty hours [118]. It was caused by a change in the power flow from Switzerland to other connections, thus leading to a strong overload on the boundary lines, which caused a domino effect and the interruption of all international connections, causing the separation of Italy from the European network. It is easy to understand how such an event could have a negative impact on several fields: railway, maritime, avionic, telecommunications, health, financial, and so on. Similar black-out caused by different reasons, also occurred in Denmark, in United States and Canada in 2003. Figure 2 shows in a schematic way the several domains that might be affected by a blackout.

More recent is the nuclear disaster at the Fukushima I Nuclear Power Plant on 11 March 2011. The failure occurred when the plant was hit by a tsunami

1 Telecom Italia is one of the main Italian telecommunications company.

triggered by the magnitude 9.0 earthquake. The plant began releasing substantial amounts of radioactive material on 12 March, becoming the largest nuclear incident since the Chernobyl disaster in April 1986 [89].
In the transportation domain, we can recall the railway accident on December 30, 2013, when a westbound Burlington Northern and Santa Fe (BNSF) train, carrying soybeans, derailed approximately one mile west of Casselton, North Dakota (US). An adjacent eastbound BNSF train carrying crude oil struck wreckage from the westbound train. The collision ignited the crude oil and caused a chain of large explosions, which were heard and felt several miles away. The resulting fireball created a massive cloud of black smoke, which prompted authorities to issue a voluntary evacuation of the city and surrounding area as a precaution. The National Transportation Safety Board (NTSB) is currently conducting an investigation into the cause of the incident, which has occurred in proximity to a populated area. Thus renewing safety concerns regarding the transportation of hazardous materials, especially in the wake of the Lac-Mégantic derailment in Canada earlier in the year [105]. Luckily no injuries to human beings were reported, but the damage was estimated at 6.1 Million Dollars [14].
A further case, in the transportation domain, is represented by the Malaysia Airlines Flight 370 that disappeared on Saturday, March 8, 2014, while flying from Kuala Lumpur International Airport to Beijing Capital International Airport. The aircraft was carrying 12 Malaysian crew members and 227 passengers from 15 nations. Currently there is still no explanation of what happened and what are the causes [21].

### 1.3.2 *Critical Infrastructures - Intentional Failures*

This section reports some real and relevant events occurred due to intentional and harmful actions on critical infrastructures. It is worth noting that only in Italy the number of cyber attacks to critical infrastructures grew in 2013 of 245%, compared to 2011 [11].

In 1982, a system controlling the trans-Siberian gas pipeline (allegedly implanted by the CIA) caused the largest non-nuclear explosion in history [136].

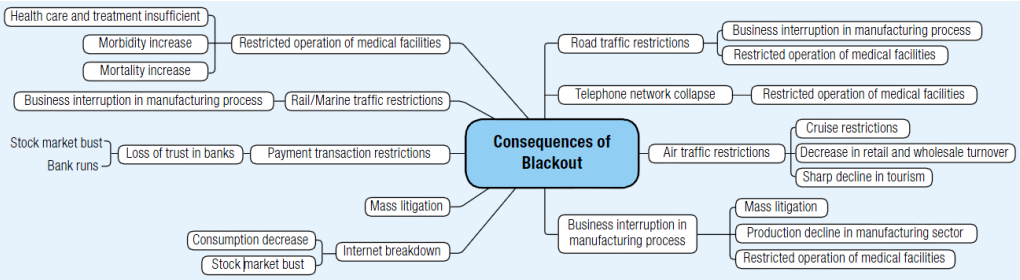


Figure 2.: Blackout consequences on different domains [10].

The CIA, allegedly discovered that Soviet spies planned to purchase secretly a gas pipeline controller developed in Canada, and planted a Trojan horse (logic bomb) in the controller's software. Once it was installed, the controller ran a test of the pipeline's pressure gauges during which the logic bomb reset the gauges to the double of gas pressure in the pipeline. The resulting explosion was the largest non-nuclear explosion ever seen from space.

Recently has been found that more than 1,000 energy companies in North America and Europe have been compromised in a huge malware attack. According to security researchers at Symantec[2], who have named the hacking gang *Dragonfly*, the main interest of the attackers was in compromising industrial control systems [45]. The attack is similar to the Stuxnet computer worm [60], which was designed to attack similar industrial controllers in 2010 and reportedly ruined almost 20% of Iran's nuclear power plants.

In January 2003 during the *"slammer"* virus[3], *Continental Airlines* was forced to shut down flights due to computer problems. Especially, the worm caused flight delays and cancellations for company after it overwhelmed the company's on-line ticketing systems and electronic kiosks that travellers use to check in [85].

2 Symantec Corporation is an American technology company. The company makes security, storage, backup and availability software.

3 Slammer is a computer worm that caused a denial of service on some Internet hosts and dramatically slowed down general Internet traffic, starting at 05:30 UTC on January 25, 2003.

# 2

# MODEL-BASED APPROACHES TO SUPPORT DEPENDABILITY AND PERFORMANCE ANALYSIS

A model is an abstraction of a system that highlights the important features of the system and provides ways of describing its properties, neglecting all those details that are relevant for the actual implementation, but that are marginal for the objective of the study.

Models play a primary role in dependability and performability assessment of modern computing systems. As a fault-forecasting technique, model-based evaluation [102] allows system architects to understand and learn about specific aspects of the system, to detect possible design weak points or bottlenecks, to perform early validation of dependability requirements, or to suggest solutions for future releases or modifications of the systems. Within the domain of critical and complex systems, modelling is a valuable tool since it avoids to perform analysis, e.g., "what-if" analyses, on a real instance of the system, which may be costly, dangerous or simply unfeasible. Modelling is also of primary importance as a support to the design process, in which the real system is not yet available.

Assessing the resilience of composite systems is a difficult task that may require the combination of several assessment methods and approaches. In this perspective, models can be profitably used as support for experimentation and vice-versa. On one side, modelling can help in selecting the features and measures of interest to be evaluated experimentally, as well as selecting the right inputs to be provided for experimentation. On the other side, the measures assessed experimentally can be used as parameters in the models, and the features identified during the experimentation may impact the semantics of the dependability model.

## 2.1 MODELLING FORMALISMS FOR DEPENDABILITY

Research in dependability analysis has led to a variety of modelling formalisms. Each of these techniques has its own strengths and weaknesses in terms of accessibility, ease of construction, efficiency and accuracy of solution algorithms, and availability of supporting software tools. The choice of the most appropriate model depends upon the complexity of the system, the specific aspects to be studied, the attributes to be evaluated, the accuracy required, and the resources available for the study. Modelling formalisms can be broadly classified into combinatorial models and state-space models. In the following, a short

overview of the most common modelling formalisms in model-based valuation of dependable systems is presented.

### 2.1.1 *Combinatorial models*

In contrast with state-space models, combinatorial models do not enumerate all possible system states to obtain a solution. Instead, simpler approaches are used to compute system dependability measures [102]. While being concise, easy to understand, and supported by efficient evaluation methods, such methods require strong assumptions to be made on the system. Typically, realistic features such as interrelated behaviour of components, imperfect coverage, non-zero reconfiguration delays, and combination with performance cannot be captured by these models.

Despite some extensions to "classical" combinatorial models introduce primitives to specify some kinds of dependencies between components (e.g., see [55]), their modelling power is still limited with respect to that offered by state-space models. Major representative formalisms in this category include:

- Fault-Trees (FTs) are a deductive modelling and analysis technique based on the study of the events that may impair the dependability of a system [57, 91, 131, 137]. A Fault Tree (FT) considers the combination of events that may lead to an undesirable situation of the system, typically the delivery of an improper service, if the reliability of the system is the matter of the analysis, or a catastrophic failure for a safety study. It is important to highlight that a fault tree is not in itself a quantitative model. It is a qualitative model that can be evaluated quantitatively, as it is often done. The use of fault trees is of particular industrial relevance in the development of critical systems; their usage, non only for quantitative evaluation, is standardized as the Fault Tree Analysis (FTA) practice [77].

- Reliability Block Diagrams (RBDs) are another popular combinatorial formalism in dependability analysis, due to its resemblance with classical block diagrams describing the physical structure of systems [81, 91, 126]. However, an Reliability Block Diagram (RBD) is a graphical structure which maps the operational dependency of a system on its components, and not the actual physical structure of the system. Blocks in a RBD may be connected together in the most intuitive configuration that can represent the temporal order of usage, or some redundancy management scheme, or the success criteria of the system. The goal is to derive the probability of correct operation or the time-dependent reliability for the overall system. The solution of a RBD model is similar to that of a FT [91, 99, 137].

Other graph-based models which can be classified as combinatorial models exist in literature, e.g., Reliability Graphs (RG) [91], and Attack Tree (AT) [134], a variant of fault trees tailored to security analysis.

### 2.1.2 *State-space models*

More accurate modelling can be obtained with state-based methods, since they allow to relax the independence assumptions needed for combinatorial modelling, and to account for more complex relationships between system components. Major representative formalisms in this category include:

- Markov chains [71, 99, 113, 137] represent a suitable tool for the modelling of a variety of systems of different nature. They combine an extreme versatility with well developed and efficient solution algorithms, which have been implemented in many automated tools. Markov Chains are widely used in different domains, and are also the theoretical basis for the evaluation of more expressive state-based formalisms.

  A Markov Chain [31] is a stochastic process $\{X(t), t \geqslant 0\}$, having a discrete (or countable) state space, and which enjoys the following *memoryless* property, also known as the *Markov property*: given the current state of the model, the future evolution of the model is described by the current state, and is independent of past states. The only continuous probability distribution that satisfies the memoryless property is the exponential distribution.

  Unfortunately, not all the existing systems and their features can be properly described using Markov processes. In some cases, the assumption that the holding time in any state of the system is exponentially distributed may be very unrealistic, and to properly represent the system behaviour more general stochastic processes (e.g., semi-Markov, Markov Regenerative or even non-Markovian processes) must be employed.

- Stochastic Petri Nets (SPNs) modelling paradigm has been developed with the specific purpose of representing in a compact and clear way concurrence, synchronization and cooperation among processes [109]. Very soon, Petri nets have been widely accepted because of their ability to describe the qualitative and quantitative aspects of complex systems, and also because of their intuitive and appealing graphical representation.

  The class of (Markovian) Stochastic Petri Nets ([24, 95]) is a very popular timed extension of the place-transition Petri nets [108].

  Due to their expressiveness, SPN are commonly used to specify Markov processes at a higher abstraction level. Several extensions have been introduced in the literature, adding new primitives to support a more compact

specification of the state-space, or allowing the specification of non-Markov processes.

- The class of Generalized Stochastic Petri Nets (GSPNs) [15] relaxes the assumption that all the transitions have an exponentially distributed delay, and allows for exponential transitions and instantaneous transitions as well (drawn as thin bars), that is transitions that once enabled fire in zero time. The solution of a GSPN model resorts again to that of an associate Markov chain, which can be solved to study the GSPN model evolution over time.

- Non-Markovian models have been developed to overcome some limitations of GSPNs. In fact, using GSPNs to model a system implies that an approximation is introduced whenever an activity with non-exponential duration must be represented. Extensions to basic SPNs have been introduced in the literature, allowing the specification of non-Markovian stochastic processes. Models with generally distributed activities can represent a large class of systems, but in the best case they require complex and costly analytical solution techniques. If analytic solution methods do not exist at all, discrete-event simulation must be used to solve the models thus providing only estimates of the measures of interest. Alternatively, one can approximate an underlying non-Markovian process with a Markov process; the price to pay following this approach is a significant increase in the number of states of the resulting Markov model, and the errors introduced by the approximation.

  Several classes of non-Markovian approaches have been defined [30] such as Semi-Markov Stochastic Petri Net (SMSPNs) [68], Markov Regenerative Stochastic Petri Nets (MRSPNs) [42], Deterministic and Stochastic Petri Nets (DSPNs) [93], Stochastic Reward Nets (SRNs) [43], Stochastic Activity Networks (SANs) [117].

Other modelling formalisms exist that allow a high-level specification of Markov Chain models, e.g., Stochastic Automata Networks [110] or the family of formalisms collectively known as Stochastic Process Algebras [44]. Such formalisms are extensions of basic process algebras (such as PEPA [72]), which are augmented with the ability to associate probabilities and/or time delays to the execution of actions, thus allowing quantitative analysis to be performed on the model. Other formalisms allow the specification of more general probability distributions, e.g., SPADES [70], and thus require more sophisticated numerical techniques or discrete-event simulation.

### 2.1.3 *The SAN Formalism*

Stochastic Activity Networks [117] are an extension of the Petri Nets (PN) formalism [100, 109]. SANs are directed graphs with four disjoint sets of nodes: *places*, *input gates*, *output gates*, and *activities*. The latter replace and extend the *transitions* of the PN formalism. The topology of a SAN is defined by its input and output gates and by two functions that map input gates to activities and pairs (*activity*, *case*) (see below) to output gates, respectively. Each input (output) gate has a set of *input* (*output*) places.

Each SAN activity may be either *instantaneous* or *timed*. Timed activities represent actions with a duration affecting the performance of the modelled system, e.g., message transmission time. The duration of each timed activity is expressed via a *time distribution* function. Any instantaneous or timed activity may have mutually exclusive outcomes, called *cases*, chosen probabilistically according to the *case distribution* of the activity. Cases can be used to model probabilistic behaviours. An activity *completes* when its (possibly instantaneous) execution terminates.

As in PNs, the state of a SAN is defined by its *marking*, i.e., a function that, at each step of the net's evolution, maps the places to non-negative integers (called the *number of tokens* of the place). SANs enable the user to specify any desired enabling condition and firing rule for each activity. This is accomplished by associating an *enabling predicate* and an *input function* to each input gate, and an *output function* to each output gate. The enabling predicate is a Boolean function of the marking of the gate's input places. The input and output functions compute the next marking of the input and output places, respectively, given their current marking. If these predicates and functions are not specified for some activity, the standard PN rules are assumed.

The evolution of a SAN, starting from a given marking $\mu$, may be described as follows: (i) The instantaneous activities enabled in $\mu$ complete in some unspecified order; (ii) if no instantaneous activities are enabled in $\mu$, the enabled (timed) activities become *active*; (iii) the completion times of each active (timed) activity are computed stochastically, according to the respective time distributions; the activity with the earliest completion time is selected for completion; (iv) when an activity (timed or not) completes, one of its cases is selected according to the case distribution, and the next marking $\mu'$ is computed by evaluating the input and output functions; (v) if an activity that was active in $\mu$ is no longer enabled in $\mu'$, it is removed from the set of active activities.

Graphically, places are drawn as circles, input (output) gates as left-pointing (right-pointing) triangles, instantaneous activities as narrow vertical bars, and timed activities as thick vertical bars. Cases are drawn as small circles on the right side of activities. Gates with default (standard PN) enabling predicates and firing rules are not shown.

## 2.2 MODELLING FORMALISMS FOR SECURITY

Cyber attacks are the basis for the evaluation of security systems. Analysis of systems security is an activity that must be done during all phases of the design process, to make design choices, during testing, deployment, operation, and maintenance, to gain confidence that the potential system's vulnerabilities have been properly handled in order to continuously satisfy security requirements characterizing the system.

Traditional approaches of security evaluation focus on avoiding intrusions to the system and violation of predefined *security properties* or *security policies*. Generally such approaches aim at identifying and representing potential attack paths, by providing a generic representation of the attacks and consequently specifying procedures that should be followed during the design of a system [1, 6, 8], but they do not perform a quantitative evaluation. Moreover, from the security analysis point of view, it is essential to integrate knowledge connected to attack paths with knowledge associated to the potential threats of the target system. Security threats are in fact one of the principal issues related to all those systems that make use of information and communication technologies. Such a characteristic make these systems prone to failures and vulnerabilities potentially exploitable by malicious agents, so a threat analysis of the system is of paramount importance in order to understand the main threats the system is exposed to, and the plausible countermeasures. At the same time a threat analysis by itself is lacking in showing how these threats can be realized, so it is important to provide a description of the attack paths that an attacker may follow to carry out the threat.

Unfortunately, security is often in conflict with critical aspects like functional requirements and/or performance of the system; consequently, a perfect security objective is not realistic, or its achievement could result very expensive in terms of costs and time. Accordingly to such a statement, it becomes of paramount importance to understand how much a system is secure and how much security it provides. There exists an extensive literature and several years of work on qualitative analysis, but quantitative methods need to be examined in depth and applied in order to provide a further insight of the system with respect to quantitative security aspects.

A first class of quantitative methods have been based on formal methods, with the intent to prove if specific security properties are still valid given a predefined set of assumptions. Often what such methods are lacking in represents is the probabilistic aspects. In particular, security models shall describe how and when a security violation occurs, its impact on the system under analysis, proper countermeasures to the attack and relative costs and effects on the system. Research in security analysis has developed a variety of models, each focusing on particular levels of abstraction and/or system characteristics. Important classes of modelling approaches are represented by: Attack Trees [120], Privilege

Graphs [51, 52, 107], Attack Graphs [124, 125, 139], and ADVISE [88]. The following sections provide an overview of these model-based approaches useful to perform security analyses.

### 2.2.1 *Attack Trees*

**Attack Trees** (ATs) [120] are closely related to fault trees, they consider a security breach as a system failure, and describe sets of events that can lead to system failure in a combinatorial way [102]. They however do not consider the notion of time. Specifically, attack trees are used to describe how a number of events may lead to a security violation. Events are represented by leafs connected through boolean nodes AND/OR. Systems' security can be modelled with a set of attack trees, where the root of each tree represents an attack that can affect the system. Attack trees can be used to evaluate several aspects of the system security, depending on the kind of value that is assigned to the leaf nodes. It can be possible to assign probabilities or costs to the the leaf nodes, thus providing an evaluation of the probability or costs needed to reach the attack goal.

### 2.2.2 *Attack Graphs*

**Attack Graphs** [124, 125, 139] and **Privilege Graphs** [51, 52, 107], with respect to attack trees, introduce state to the analysis. In a privilege graph a node represents a privilege state. An attacker starts at one node in the graph and works toward an attack goal by gaining privileges and transitioning to new privilege states. Attack graphs (AGs) and privilege graphs enable state-based analysis, but they do not consider the different attack goals and attack preferences of individual adversaries.
The AG analysis aims to determine Attack Paths (APs), which are paths that link entry points of a network to the possible targets, going through the vulnerability spread in the network and linked together. Once APs are computed a system designer is able to understand which are the security weaknesses and where is better to place patches in order to enhance the security of the system.

In a network infrastructure, for example, any component has its own weaknesses. The exploit of a vulnerability can be expressed as an event which occurs if a given set of pre-conditions hold. These conditions may include the set of vulnerabilities that the exploit relies on, sufficient user rights on the target and on the attacked host and network connectivity. When all the pre-conditions hold, an attacker can exploits the vulnerability gaining specific post-conditions, which are simply pre-conditions to another exploit.

### 2.2.3 *ADversary VIew Security Evaluation*

**ADversary VIew Security Evaluation (ADVISE)** [88] extends the concept of attack graph by building executable models driven by the preferences of attack. Analyses performed through ADVISE can be tailored in order to reflect the behaviour of attackers with different goals, preferences, resources, skill, knowledge and access to the system. An attack is composed of a sequence of steps. All the potential attack steps against the system are defined as the *Attack Execution Graph (AEG)*. Each step allows the achievement of a goal or the progress of the attack. With respect to attack graphs, ADVISE introduces the concept of time, probability, and costs associated to each single attack step. One of the big problem of the security evaluation is represented by the undefined behaviour of the adversary, that is defined as "the activity of planning and executing an attack, with fall-back positions and alternatives, is called attacker course of action" [80]. From this perspective, ADVISE provides a profile of the attacker, by specifying its attributes (*adversary preferences*), through an *adversary profile* (**AP**). The combination of the AEG and the AP allows to generate an executable model useful to produce relevant analysis output based on security metrics. The execution of the model permits to determine the sequence of potential state transitions. The method models step-by-step the decisions of the attacker, where the outcome of an attack step has impact on the next decision of the adversary. ADVISE introduces a precise and reproducible technique to create security state-based models. The execution of the model is based on the adversary's profile and on the attack steps of the graph, with the objective to simulate the adversary's behaviour. The adversary selects the best attack step to follow, among the possible next attack steps. The choice is based on the attractiveness of the steps based on costs, rewards and probabilities (to be detected).
An ADVISE model allows to evaluate quantitative metrics in order to provide security informations of a system with respect to a specific adversary. Metrics can assess the probability of compromise within a particular time period and can also give insight on the speed of compromise and the most likely attack steps. It is possible to consider two types of metrics: state metrics and event metrics. State metrics analyze the model state. Event metrics analyze events, namely state changes, attack step attempts, and attack step outcomes.

## 2.3 THE CONTRIBUTION OF THE THESIS

Modelling activity plays a primary role in dependability and performance assessment of modern complex systems. Whatever formalism or tool is used to perform model-based analysis and evaluation, model-based analysis shows an important support to: *i)* early detection of potential design weaknesses and bottlenecks, *ii)* make sound choices among several available alternative solutions

to address dependability and, in general, Quality of Service (QoS) requirements, and *iii)* tuning of dependability mechanisms parameters.
One of the main advantages of adopting modelling approaches is that they do not exercise real instances of the system under analysis, which on the contrary may result costly in terms of time and money or simply unfeasible (e.g., because the system is still under design).

Accordingly to such considerations, the thesis mainly focuses on issues related to off-line and on-line quantitative evaluation of dependability, performance, and security properties of critical infrastructures and complex systems.

The work of the thesis is composed of three main parts having in common the definition of model-based approaches to dependability and security assessment in heterogeneous, dynamic and evolving systems.
The first part of the thesis considers the analysis of critical infrastructures as a support to the design and maintenance phase of the system. Particular attention regards electrical power systems, which are classified as critical infrastructures, providing services highly impacting on our society. They rely on a complex internal organization, where interdependencies among the composing parts increase their vulnerabilities. In this context, the work of the thesis is mainly based on the results of the EU Project CRUTIAL [50], which aimed at developing a modelling framework for the assessment of electrical power systems, considering separately the two constituting infrastructures: the electrical infrastructure and the infrastructure of the control system and taking into account their interdependencies, thus allowing to assess the impact of reciprocal failures. The thesis provides an extension of the modelling framework, by considering new heterogeneous aspects of the components of the electrical grid, and different types of failures, making it possible to go beyond the limitations of the framework, which considered homogeneous characteristics of the components and therefore unrealistic. Such an extension allows us to assess similar infrastructures in more realistic conditions.
The second part of the thesis has been carried out in the context of the EU Project CONNECT [46], which aimed at resolving the heterogeneity barrier and enable the continuous composition of networked systems independently of the deployment and implementation technologies, through the synthesis of CONNECTors. In this context the thesis contributes to provide a novel modelling approach to be used both at design time and at run-time. The approach provides at design time a potential enhancement of dependability and performance requirements of the CONNECTor, based on a pre-defined library of dependability and performance mechanisms, thus allowing the deployment of CONNECTors that satisfy non functional requirements. At run-time, the approach allows to update and refine the model parameters with actual values gathered from on-line observations.
In conclusion, the third part of the thesis, which is carried out in the context of the Italian research Project SECURE! [2], is related to aspects of security assessment of critical and complex system. Specifically, the work provides a

generic methodology for the security assessment with respect to the insider threats, considering as starting point the interactions of legitimate users with the systems.

# 3

# MODEL-BASED APPROACHES TO SUPPORT SYSTEM DESIGN DECISION

The conditions under which Critical Infrastructures operate are continuously evolving. In fact, as mentioned, in the past they were used to provide services mainly in isolation with limited interconnections with other infrastructures. Nowadays, the scenario is quite different, and more and more infrastructures are required to cooperate with each other.
Moreover, in the past, accidental threats were basically the only real threats to the infrastructures, in particular they were mainly related to natural disasters, which were geographically localized and time limited, thus low attention was devoted to intentional and malicious acts targeting such kind of critical infrastructures.
The increasing evolution of Information and Communications Technology (ICT) allowed an efficient and convenient control of CIs remotely (e.g., over the Internet). Hence, industries and governments have increasingly used Information Technology (IT) systems in order to improve the operation of CIs. As consequence, previously isolated worlds (CIs and ICT) have become strongly interconnected, thus leading to a greater *complexity* and to new aspects of *heterogeneity*, where several heterogeneous systems are often called to interact or to be integrated with each other, even though they were not designed with this purpose. Although this correlation brings benefits to the operation of such infrastructures, at the same time it raises new security concerns and increases simultaneously the risk of potential faults. Two aspects that should always be considered together during the planning of the protection of CIs.
Furthermore, CIs are composed of critical components, each of them has to be analyzed from the point of view of possible risks and security aspects. Components which are intended to be used and to operate in safety-critical systems and environments are usually designed to be failsafe. However, new vulnerabilities are introduced, due to the increased connectivity and open design of these infrastructures, the use of Commercial Off-The-Shelf (COTS) components, which were not built with security in mind, thus increasing the attack surface of such systems.

The complexity, heterogeneity, adaptability, and mobility of critical infrastructures impose novel challenges on the design of risk mitigation systems and security mechanisms. Indeed, structures evolve to improve the quality of the provided services as well as to manage possible threats caused by new interactions and new methods of attack.

Therefore, it is of paramount importance that they have to be reliable and resilient to continue providing their fundamental services. Hence, it is critical to:

*i)* build such infrastructures following reasonable engineering design principles; *ii)* protect them against both accidental and malicious faults; and *iii)* assess their degree of resilience and security.

This chapter presents an assessment framework helpful: *i)* to support the detailed modelling of the interdependencies between two different infrastructures, and *ii)* to quantitatively assess the impact of such interdependencies through proper specified metrics, and the presence of heterogeneous characteristics.

## 3.1 STATE OF THE ART

Several research projects have had and still have a focus on the study of critical infrastructures, their interdependencies and related new vulnerabilities and threats. In the following, we provide an overview of some relevant research projects in this context.
The European project IRRIIS [78] (ended in 2009) has devoted significant effort to interdependencies analysis and modelling, with focus on the dependencies of the telecommunication infrastructure from the power supply. They have developed Sim-CIP [138], an integrated simulation environment used for the modelling and simulation, based on the ISE (Implementation, Services and Effects) metamodel [84], [119].
The European Project AFTER [12] has addressed the challenges posed by the need for vulnerability evaluation and contingency planning of the energy grids and energy plants, in presence of natural or man-related hazard scenarios, considering also the relevant ICT systems used in protection and control. The project focused on high impact, widespread multiple contingencies and on cascading events that can cause catastrophic outages of the electric power systems. In particular, AFTER aimed at developing a methodology and tool for the global vulnerability analysis and risk assessment of Electric Power Systems considering interdependencies with ICT systems.
The TCIPG project [4], formed by the partnership of U.S. government, academia, and industry, is currently active in order to protect U.S. power grid by significantly improving the way the power grid infrastructure is designed, making it more secure, resilient, and safe.
The European project SoES [3] aims at facing the ICT security demand of Energy Smart Grids. The objective of the project are mainly related to *i)* the identification of vulnerabilities, threats and countermeasures relevant for the Smart Grid architectures; *ii)* the definition of ICT security best practices for Energy Smart Grids.
EPSRC-NSFC [59] is an active collaborative research initiative between UK and China Institutions, which aims at studying the integration of large scale renewable power parks with Direct Current (DC) networks and at investigating secure

grid topologies and their interface with Alternating Current (AC) grids.
The European project CRUTIAL [50] (ended in 2009) addressed new networked ICT systems for the management of the electric power grids, in which the physical process of electricity transportation need to be connected with information infrastructures, through corporate networks (intranets), which are in turn connected to the Internet. A major research activity of the project focused on the design of a modelling framework which aimed at building generic and reusable models, accounting for: *i)* internal dynamics of the represented control and power grid infrastructures as well as dependencies among them, and *ii)* generic fault and propagation conditions. In particular, the project has been deeply interested in the modelling of interdependencies between the two infrastructures constituting an Electrical Power System (EPS): the power grid and its cyber control. In the context of the project, an innovative, modular EPS modelling framework has been developed, considering separately the two constituting infrastructures and taking into account their interdependencies, thus allowing to assess the impact of reciprocal failures [27, 38, 129, 130]. The CRUTIAL project has started a promising activity of analysis in the context of power grid, thus we believe that it is important to continue the work performed within the project, by proposing an extension of the developed modelling framework. Specifically, the extension we propose allows us to go beyond the limitations of the modelling framework, developed within the project, by considering heterogeneous aspects of the system under analysis, thus allowing to get relevant and concrete measures representative of average trends of the power delivered by an EPS.
The next Section 3.2 provides a detailed description of an Electrical Power System that is the starting point of the thesis.

## 3.2 ELECTRICAL POWER SYSTEMS: AN OVERVIEW

Starting from the results obtained in the context of the EU Project CRUTIAL and the further progresses presented in [128], this section provides a description of the Electrical Power Systems and the modelling framework developed during the project, in order to give a clear overview of the context, thus allowing to understand the improvement and the new results we aim at obtaining with the thesis.

An Electrical Power System (EPS) is generally defined as a network of electrical components used to generate, supply, transmit electric power. The EPS is logically structured in two main interacting parts: the Electrical Infrastructure (EI) and the Information Technology based Control System (ITCS). EI represents the infrastructure necessary to produce and to transport the electrical power towards the final users. ITCS represents the control system based on information technology, whose main objectives are: *i)* minimizing the downtime of generators, power lines and substations, and *ii)* improving the quality of

service through the setting of parameters (e.g., frequency and voltage). Hence, ITCS is able to remotely control the electrical infrastructure, receiving data and sending commands, and to coordinate maintenance and reconfiguration actions on the electrical grid.

The logical structure of the EPS is depicted in Figure 3, where a number of regional transmission grids are connected through a multi-regional communication network. Each region has the same logical structure from the point of view of both electrical grid elements and information infrastructure ones, although the number of the elements and the topology of the grid can be different for each region. A single region is composed by the regional EI (namely the Regional Transmission Grid) and the regional ITCS.

EI includes all the electrical elements that are logically distinguished in: *generators*, which produce energy, the *power lines*, through which the produced energy is conveyed to reach *loads*, which are different types of end-users using the produced energy. *Substations* are structured components in which the electric power is transformed and split over several power lines.
The ITCS is logically composed by two hierarchical subsystems: the Local Control Systems and the Regional Tele-control Systems.

- The Local Control System (LCS) guarantees the correct operation of a node equipment and reconfigures the node in case of breakdown of some apparatus.
- The Regional Telecontrol System (RTS) monitors all the electrical components of its assigned region and takes reconfiguration actions to restore the functionality of the grid in case of breakdowns, involving the whole region and possibly also the RTSs of neighbouring regions, if necessary.

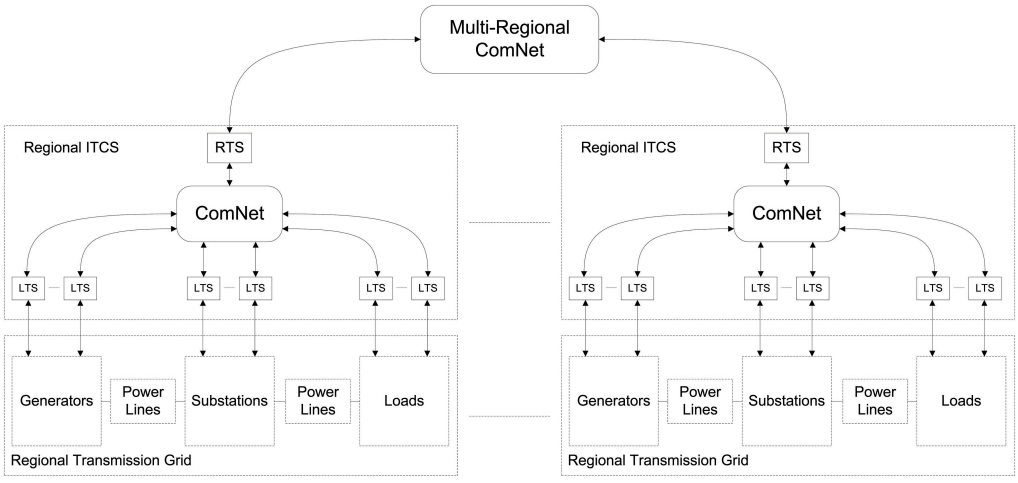


Figure 3.: Logical structure of a multi-region Electrical Power System.

The two considered reconfiguration strategies, performed by LCS and RTS respectively, are:

- $\mathcal{RS}_1()$, to represent the effect of the reactions of ITCS to an event that has compromised the electrical equilibrium of EI when only the state local to the involved EI components is considered. Given the limited information required by this reconfiguration, performed by the LCS controlling the electrical apparatus affected by the failure, it is considered very fast (actually, instantaneous) in the model.
- $\mathcal{RS}_2()$, to represent the effect of the reactions of ITCS to an event that has compromised the electrical equilibrium of the EI portion and when the state global of all the EI system under the control of ITCS is considered. This reconfiguration, performed by the RTS of the affected region, requires knowledge of the global state of the region and therefore reacts in a longer time.

The output values of $\mathcal{RS}_1()$ and $\mathcal{RS}_2()$ (i.e., the new values for the Power Flow **F** and the Active Power **P**) are derived by solving different Linear Programming (LP) problems, based on the overall grid. Typically, $\mathcal{RS}_1()$ is based on the grid configuration immediately before the occurrence of the disruption, while $\mathcal{RS}_2()$ is based on the nominal grid configuration, that is the initial configuration at time *zero* (e.g., in case the power demand is constant). The EPS organisation in multi regions requires a coordination among them when computing a reconfiguration spanning a number of regions. A reconfiguration algorithm considering multiple regions has been developed, based on the principle that first a new electrical equilibrium is attempted inside the region where the failure occurred and, only in case this is not possible, neighbour regions are called to contribute [128].

### 3.2.1 *The Modelling Framework*

#### *SAN implementation*

The implementation of the EPS framework just recalled has been carried out using the SAN formalism [117], which is an extension of the Stochastic Petri Nets, and the Möbius tool [53]. Atomic models for each of the main EI and ITCS components have been developed and connected (*Join* operator) through some shared places of the SAN model, that represent part of the states of EPS. The overall model, obtained by combining such atomic models (properly replicated through the *Rep* operator in accordance with the multiplicity of the represented components in EPS) is shown in Figure 4.

In the following we recall the atomic models which compose the overall EPS model of Figure 4 [128].

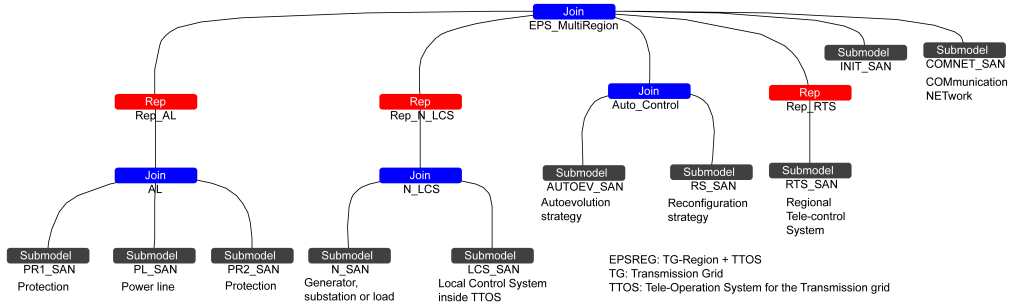


Figure 4.: Composed model for a multi-region EPS

1. INIT_SAN, which represents the initialization of the overall model at initial time 0.
2. PL_SAN, which represents the generic power line with the connected transformers.
3. PR1_SAN and PR2_SAN, which represent the generic protections and the breakers connected to the two extremities of the power line.
4. N_SAN and LCS_SAN, which represent, respectively, a node of the grid (a generator, a load or a substation) and the associated Local Control System.
5. AUTOEV_SAN, which represents the automatic evolution of EI when an event modifying its state occurs.
6. RS_SAN, which represents the local reconfiguration strategy $\mathcal{RS}_1()$ applied by LCS, and the computation of the regional reconfiguration strategy $\mathcal{RS}_2()$ (computation time and application of $\mathcal{RS}_2()$ are modelled in RTS_SAN).
7. RTS_SAN and COMNET_SAN, which represent, respectively, the RTS, where the regional reconfiguration strategy $\mathcal{RS}_2()$ is applied, and the public or private networks (ComNet of Figure 3).

The Multi-Regional ComNet network of Figure 3 and its failures have not been explicitly modelled, but they are accounted for in the RTS_SAN model and in the $\mathcal{RS}_2()$ reconfiguration algorithm.

### 3.2.2 *Heterogeneous context*

Taking advantage of the flexibility of the approach in terms of both data structures and cost functions at the basis of the reconfiguration policies, the thesis manages to adapt it to new heterogeneous context by making modifications and refinements to the reconfiguration strategy, in order to be able to consider

more realistic scenarios and providing a concrete support to the design or the management of EPSs [39, 40, 41].

The new types of heterogeneity we aim at including in the original framework are related to power lines and loads of the grid. Specifically, the thesis extends the EPS modelling framework, which was limited to homogeneous scenarios from the point of view of the entities of the power grid. Indeed, we consider the option of different failure rates for the power lines and the effect of the failure of clusters of power lines. We also consider different repair times for each power line, to explore the impact of this system parameters and get insights on the worthwhileness of acting on it to improve user satisfaction. Moreover, we consider the presence in the grid of loads of different criticality from the point of view of consequences of power loss in case of failures.
In particular, we modify the implementation of the $\mathcal{RS}_2()$ algorithm, by introducing a new cost function which allows us to obtain the desired behaviour of the reconfiguration strategy at multi-regional level in presence of a critical load. In order to account for the higher criticality of load $j$ with respect to the other loads, a maximum cost is associated to power loss of load $j$, while all the other loads of the whole grid have much lower values for the cost parameters. This implies that, in case of malfunctions triggering a reconfiguration by ITCS, redispatch and shedding inside the region $r$ where the problem occurred (except for load $j$ if in Region $r$) are attempted first. If no solution within Region $r$ is found, redispatch and shedding operations in neighbour regions (following the development of the $\mathcal{RS}_2()$ reconfiguration algorithm) are attempted first, always excluding the critical load $j$, whose shedding is performed only as the very last action if no electrical equilibrium has been reached meanwhile. In the following, we show the expression for the new cost function accounting for heterogeneity, related to the reconfiguration strategy $\mathcal{RS}_2()$:

$$\begin{aligned} C = &\sum_{i\in\mathcal{G}'} WG'|P_i - P_i^0| + \sum_{i\in\mathcal{G}''} WG''|F_{l_i} - F_{l_i}^*| \\ &+ \sum_{\substack{i\in\mathcal{L}' \\ i\neq j}} WL'|P_i - D_i| + \sum_{i\in\mathcal{L}''} WL''|F_{l_i} - F_{l_i}^*| + WL'_j|P_j - D_j| \end{aligned} \tag{3.1}$$

where the following items hold:

1. $\mathcal{G}'$ and $\mathcal{L}'$ are the sets of real generators and loads, while $\mathcal{G}''$ and $\mathcal{L}''$ are the sets of dummy generators and loads representing boundary power lines, which are the power lines connecting regions to each other (as developed in [128], boundary power lines are not directly included in the modelling framework, but they are substituted by a *dummy generator* and a *dummy load* in each of the two interconnected regions);

2. $P_i^0$ is the power on generator $i$ at the time $0$;

3. $D_i$ is the power demand (constant over time) on load $i$;

4. $F^{*}_{l_i}$ is the power associated to boundary power line $l_i$, immediately before the occurrence of the disruption;

5. $WG'$ and $WL'$ are the weights associated to real generators and loads, while $WG''$ and $WL''$ are the weights associated to dummy generators and loads;

6. $WL'_j$ is the weight associated to critical load $j$;

7. $WG' << WG'' << WL' << WL'' << WL'_j$; these relations between weights establish the order of shedding and redispatch operations to perform during the reconfiguration.

In the case where all the loads are critical except the load $j$, the items (6) and (7) are replaced by the following:

6. $WL'_j$ is the weight associated to the unique not critical load $j$;

7. $WG' << WG'' << WL'_j << WL'' << WL'_i$, for each critical load $i \neq j$.

It is important to notice that, since the output values of $\mathcal{RS}_2()$ (and $\mathcal{RS}_1()$) are derived by solving an LP problem, the behaviour of the modelled power system may (also significantly) change for different relations and settings of the weights associated to the same cost function.

*The EPS configuration under analysis, measures of interest and scenarios*

This section provides a description of the EPS configuration and the measures of interest we aim at considering in the thesis. The test grid we adopt for our analysis is based on the IEEE Reliability Test System - 1996 (RTS-96) (described in [75], [76]), which was created by a committee of power system experts in order to provide a standardized test grid for different power system reliability evaluation methodologies. In particular, it may include many different configurations and it is typically adopted as reference power grid in several power system reliability evaluation studies. Figure 5 shows the RTS-96 in the configuration we consider for our analyses. It is composed of 56 power lines and 42 nodes, of which 10 are generators (circles in figure), 15 substations (diamonds in figure), and 17 are loads (squares in figure). It is structured into four interconnected regions, which are enclosed into a dashed box in figure, without following any specific criterion, but in a rather arbitrary way just for the purpose of exercising a multi-regional power grid.

In the figure, the label "$P_i/P_i^{max}$" associated to the generators represents the initial (active) power $P_i$ and the maximum power that the generator $i$ can supply $P_i^{max}$. The label "$D_i$" associated to the loads represents the power demand (constant over time) of the load $i$. The label "$F_{ij}/F_{ij}^{max}$" associated to the power lines represents the initial power flow $F_{ij}$ through the power line $(i, j)$ and the maximum power flow that a transmission power line can carry $F_{ij}^{max}$. A

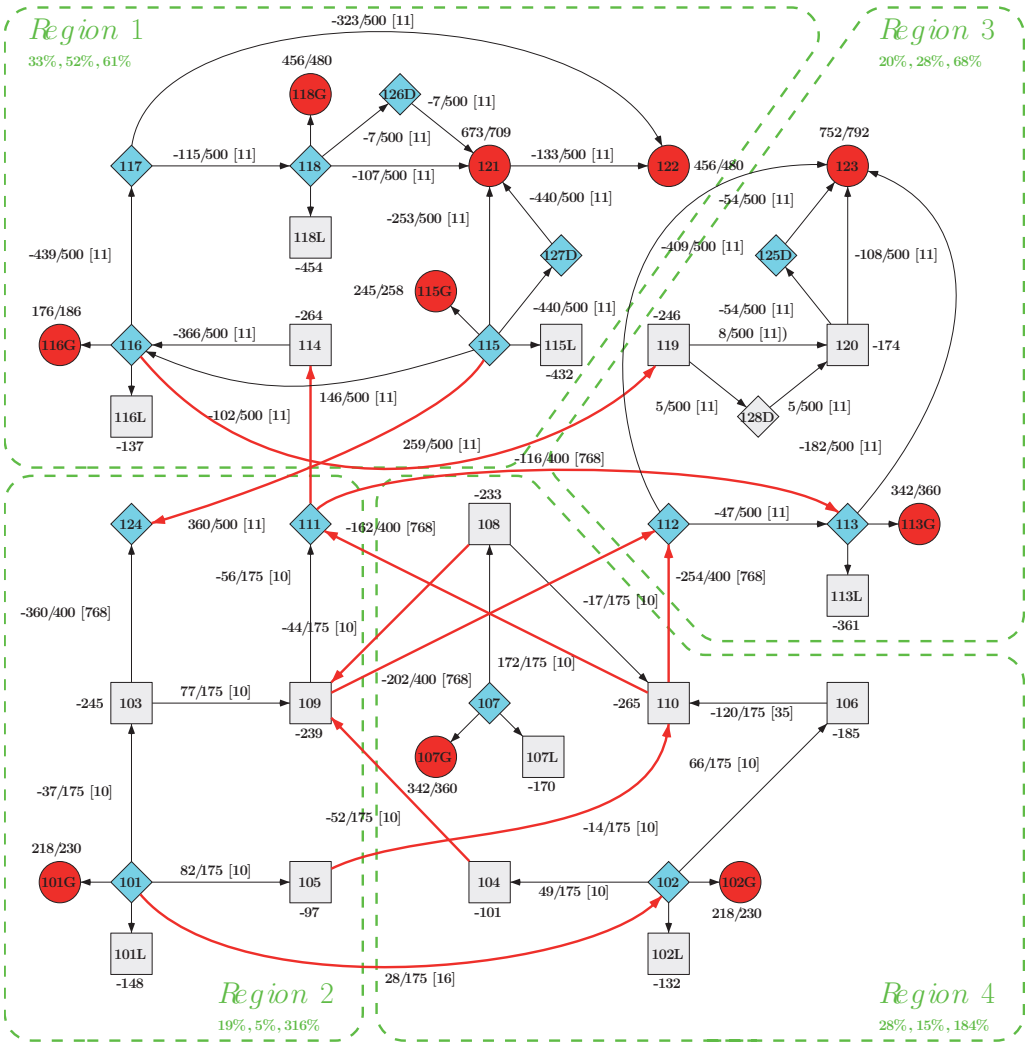


Figure 5.: Diagram of the EI grid corresponding to the RTS96 test grid

negative $F_{ij}$ value means that the current is flowing in the opposite direction of the corresponding arrow. Also, in square brackets we shown the repair time of each power line, in number of hours. The percentages under the region name represent, in the order, the percentage of the (active) power demand of the region with respect to power demand of the overall grid, the percentage of power provided by the region with respect to power demand of the overall grid, and the percentage of the maximum power of the region needed to satisfy the power demand of the region (i.e., the ratio between the whole power demand of the region and the maximum power that can be supplied by the region itself).

With this configuration of the electrical grid, we report in the following the new measures of interest we aim at evaluating in our analyses as indicator of the black-out size:

- $UD_h$, is defined as the mean number of hours of undelivered power demand to load $h$, from the time of disruption until the restoration of the correct state of the power system (that is, after the repair of the failed power lines, condition required for all power demand to be met).
- $UD$, defined as the mean number of hours of power demand that is not met for the whole grid, from the time of disruption until the restoration of the correct state of the power system.

These measures do not express the loss of power in terms of absolute values of power unit, e.g., *Mega Watt (MW)*, but rather in terms of hours of undelivered power demand (until the repair of the failed lines, whose time represents the maximum number of hours of power demand loss that can be experienced). Thus, they provide a very intuitive quantification of the loss of power demand that is independent from a reference time window for the analysis.

With the purpose to perform extensive analyses, we consider several scenarios based on the grid topology in Figure 5, each focusing on specific combinations of loads criticality and power lines failures. We define the load criticality in terms of the cost associated with undelivered power demand with respect to the requested one, which leads to perform shedding operations on the critical loads only after the shedding of the non critical loads involved in the reconfiguration has been performed without resulting in an electrical equilibrium.

The results have been obtained using the simulator engine of the Möbius tool. The failure event triggering the intervention of the ITCS reconfiguration is a disruption of power lines. Therefore, in the experiments performed, the simulation starts just after the failure of one (or more) power line(s), and lasts until the end of the repair of the power line(s). Each result is obtained by executing 20000 simulation runs (batches). The confidence level was set to 0.95. The confidence intervals obtained for the results are shown in the plots, although, for some values, they are very small (similar to points).

### 3.2.3 *Analysis results*

This section presents the analyses we perform on the EPS, applying aspects of heterogeneity on the different scenarios briefly sketched in the previous section. In particular, we aim at examining three major scenarios: *i)* electrical grid composed by loads characterized by different criticality; *ii)* electrical grid composed by the failure of a cluster of power lines; and *iii)* electrical grid with power lines characterized by different repair time.

*Scenario with loads characterized by different criticality*

To exploit heterogeneity on the loads, in this scenario we consider all the loads having the same criticality except one, for which a loss in the requested power demand results in:

- much higher costs: the case where one load is assumed to be the only critical load, or
- much lower costs: the opposite case where all the loads are critical except one load.

This kind of heterogeneity is directly reflected in the algorithm of $\mathcal{RS}_2()$, namely in the cost function we consider when setting up the LP problem, as defined in equation (3.1). For the sake of comparison, we provide the results of the analysis also including the case of homogeneous setting corresponding to the case that all the loads have the same criticality.

Figure 6 shows the results of the expected number of hours UD of undelivered power demand for the whole grid at varying the (single) power line $(i, j)$ that fails (on the x axis). More specifically, Figure 6 illustrates the trend of UD in presence of different criticality of the loads. The loads shown in the figure are chosen so as to include: the one requesting the highest demand (load *118L*), the one requesting the lowest demand (load *105*) and two representatives of a

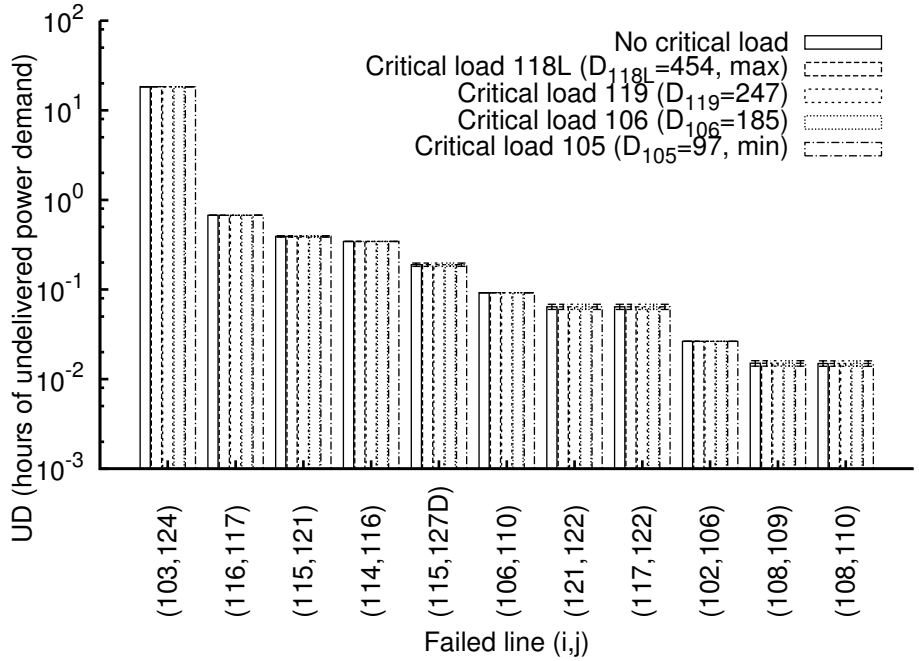


Figure 6.: UD, at varying the single failed power line $(i, j)$, for different critical loads and with different times to repair.

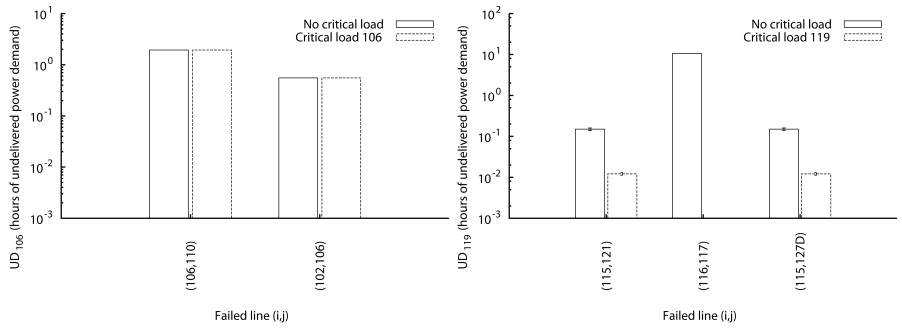


Figure 7.: $UD_{106}$ (a) and $UD_{119}$ (b), at varying the single and failed power line $(i, j)$ and with different times to repair.

medium demand request (loads *119* and *106*). In addition, we also show the basic case where all the loads have the same criticality (indicated, in the figure by the label "No critical load"). We can observe that, whatever be the failed power line, considering a different criticality for a specified load does not impact on the undelivered power demand of the overall electrical grid, whether the load has a maximum, intermediate, or minimal demand.

Instead, looking at Figures 7a and 7b, where we consider the measure of interest relative to a single critical load, we can observe how the failure of the specific power line can impact differently on the different critical load. As expected, when a load is critical its undelivered demand is in general lower (sometimes zero) with respect to the case when it is not critical (e.g., the failure of the power line $(116, 117)$ does not impact the load *119* when treated as a critical node, see Figure 7b). However, there are cases where the loss remains the same: this occurs when, given the location of the failed power line and the critical load, a new equilibrium cannot be restored without affecting the power demand of that critical load. For example, in Figure 7a when the load *106* is critical, the failure of the power lines $(106, 110)$ or $(102, 106)$, affects $UD_{106}$ in the same way as in the case this load is not critical. The corresponding figures relative to loads *118L* and *105* are not shown, since $UD_{118L}$ and $UD_{105}$ are always zero, either when they are considered critical or not and whichever be the failed power line.

Figure 8 shows the analyses results of the overall undelivered demand, when we consider all the loads critical except one, the load *119* or *118L*, that are treated separately as non-critical ones. Likewise in Figure 6, we perform the analyses at varying the failure of a single power line $(i, j)$ (on the x axis). However, contrary to the results of Figure 6, it is interesting to observe that the failure of specific power lines, such as power lines $(103, 124)$ or $(115, 121)$, impacts differently on the measure UD for different non-critical loads. For example, considering the failed power line $(103, 124)$, for the case *no critical load* the overall undelivered

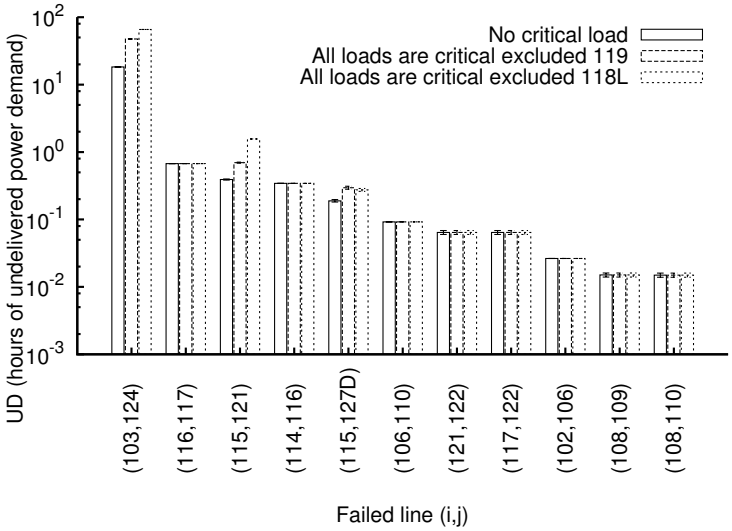


Figure 8.: UD, at varying the single failed power line $(i, j)$, when all the loads are considered critical except one load (the load 119 or 118L), with different times to repair.

demand is UD = 18.3, while for all loads critical excluded *118L* the overall undelivered demand is UD = 65.7. The difference between these values is due mainly to the loss of 47.8 hours of overall demand on the load *118L*, in the case where all loads are critical excluded *118L*. In this case, the relations and settings of the weights associated to the cost function defined in equation (3.1) have a negative impact on UD, with respect to those adopted for the case with no critical load.

Figure 9 shows how the only failure of power line $(116, 117)$ has impact on the $UD_h$, for each load $h$, both in the extreme cases where load $h$ is assumed to be the only critical load in the grid and in the opposite case where all the loads are critical except load $h$. We selected power line $(116, 117)$ as the failed power line since, from Figure 6, it is the one causing the greatest load loss together with power line $(103, 124)$. Not surprisingly, when load $h$ is the only non critical one, its undelivered demand ranges from 5 to 11 hours (this last value indicates complete blackout for the duration of the power line failures, being 11 hours the repair time for this power line). In the opposite case when load $h$ is critical, its power loss is 0.

Figure 10 shows the results of the same analysis but focusing on the specific load *119* and showing how $UD_h$ for all the loads $h$ changes when load *119* is the only critical one or the only non critical one. We can observe that for some loads there is almost no influence, but for others (like loads *103*, *119* and *101L*) the impact is significant.

Figure 11 shows how the only failure of power line $(116, 117)$ has impact on $UD_{119}$, at varying the number of random critical loads (on the $x$ axis), both

when *119* is a non-critical and a critical load. Looking at the figure, when the load *119* is critical, $UD_{119}$ is mostly zero, except when the number of random critical loads is high (12, 13, and 14), meaning that in this configurations no shedding of the other non-critical loads is enough to avoid some power loss of load 119. Moreover, when load 119 is non-critical, load $UD_{119}$ has the same

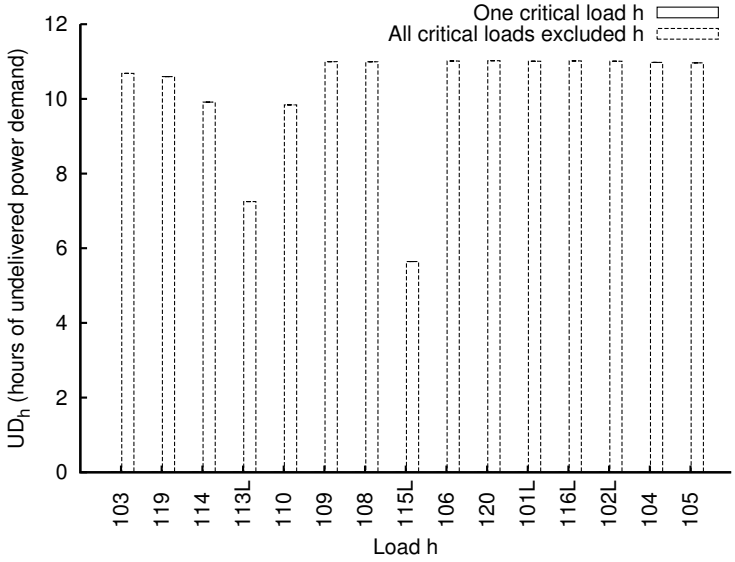


Figure 9.: $UD_h$, at varying the load $h$, when $h$ is the only critical load and when it is the only non-critical load, in case of failure of power line $(116, 117)$ only and with different times to repair.

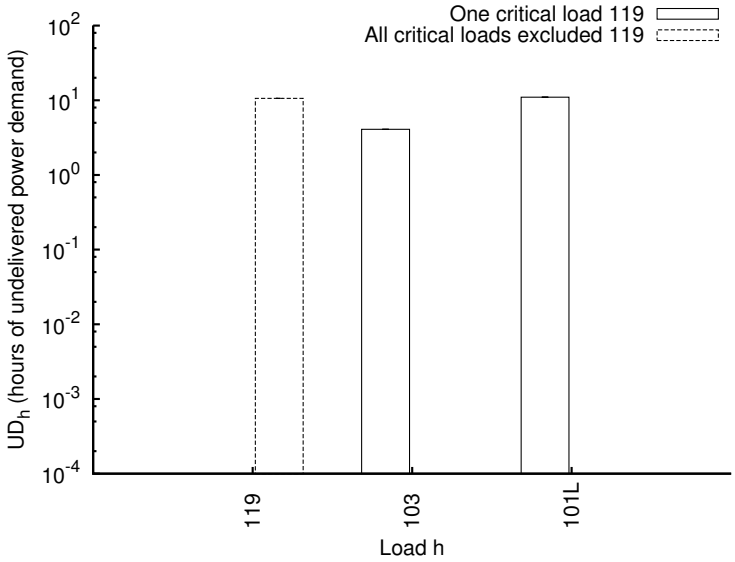


Figure 10.: $UD_h$, at varying the load $h$, when *119* is the only critical load and when it is the only non-critical load, in case of failure of power line $(116, 117)$ only.

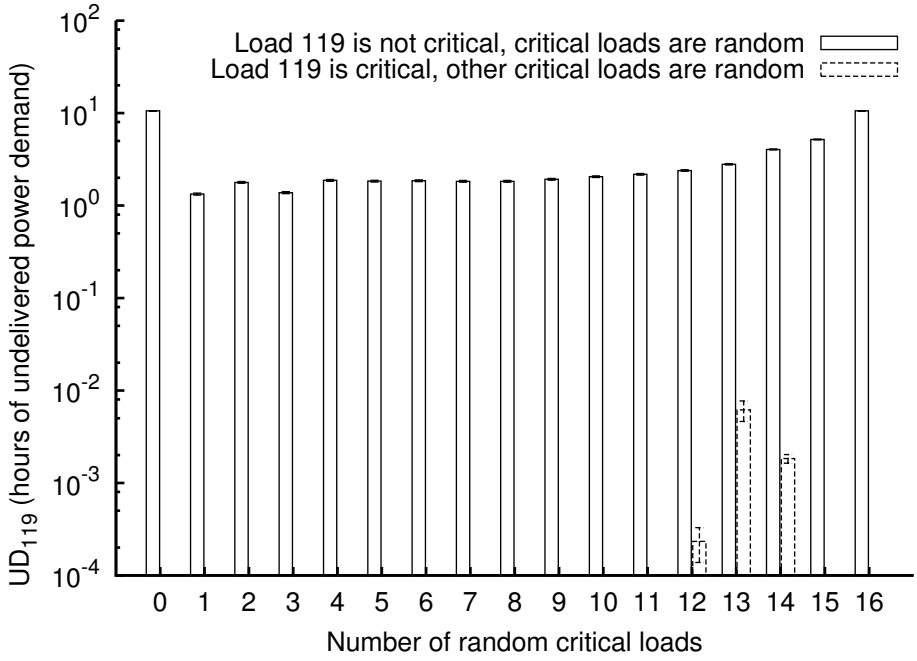


Figure 11.: $UD_{119}$, at varying the number of random critical loads, when *119* is the only non-critical load and when *119* is critical, in case of failure of power line $(116, 117)$ only and with different times.

value in the two extreme cases where no critical loads exist in the power grid and where all the loads other than load 119 are critical. This occurs because all the other loads, having the same criticality level are managed in the same way by the reconfiguration algorithm.

*Scenario characterized by the failure of a cluster of power lines*

In order to move towards more realistic scenarios and to overcome the limitation of the basic EPS modelling framework, we perform analyses in order to evaluate the impact of the failure of a cluster of power lines.

Figure 12 shows UD at varying the failure of power line $(i, j)$, on the x axis, together with its neighbouring power lines (e.g., looking at the grid in Figure 5, the neighbours of power line $(110, 106)$ are: $(110, 112)$, $(108, 110)$, $(110, 111)$, $(105, 110)$, $(102, 106)$).
We perform the analyses considering three cases:

- the grid has no critical loads and only power line $(i, j)$ fails;
- the only critical load is *118L* (chosen since, as a result of the analysis in Figure 6, no single failure of power lines has effect on this load);
- all the loads are critical except load *118L*.


Figure 12.: UD, at varying the failure of power line $(i, j)$, on the x axis, together with the failure of its neighbouring power lines, for different criticality conditions of the loads and with different times to repair.

The last two cases are in presence of simultaneous failure of power line $(i, j)$ and its neighbouring power lines.

The first immediate result from this analysis is that the impact of a single power line on the mean power loss UD is limited to a small set of power lines only (11 out of the 56 power lines included in the grid), while the failure of clusters of power lines has always a relevant effect. The second comment is that the failure of cluster of power lines (whose dimension varies depending on the number of neighbours of each power line) has, in the great majority of the cases, an impact that is independent from the number of critical loads considered. This is due to the fact that the damages caused by the failed power lines are so substantial that the criticality dimension of the loads becomes almost irrelevant.

Figure 13 shows $UD_h$, when load $h$ is the critical one or the specific load *119* is the critical one, in presence of the failure of the whole set of power lines directly connected with $(116, 117)$, including $(116, 117)$ itself. The results in figure also show the comparison with the simplest case in which only power line $(116, 117)$ fails, to better appreciate the effects of the failure of the cluster of power lines. It is interesting to note that the single failure of power line $(116, 117)$ never impacts on $UD_h$, for each load $h$, while the failure of the power line and of all its neighbouring ones determines in general a significant loss (leading to total loss

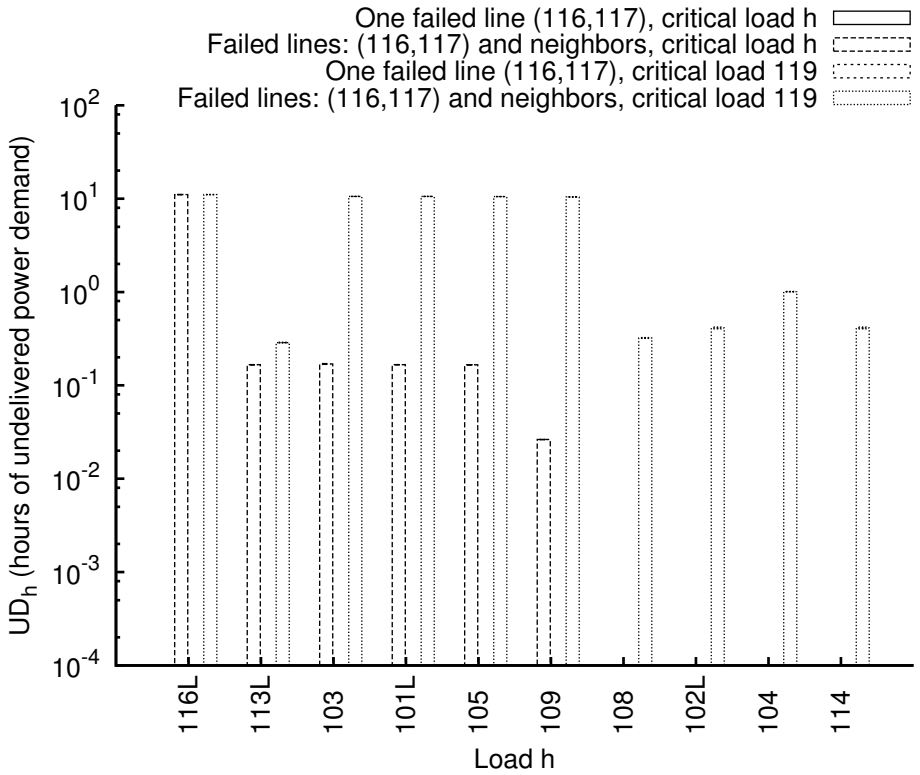


Figure 13.: $UD_h$ when load $h$ is the critical one or the specific load *119* is the critical one, for different criticality conditions of the loads and with different times to repair.

for some loads, e.g., load 116L in presence of the failure of power line $(103, 124)$, which has the highest repair time). Although the trend is not surprising, the results are useful to understand some characteristics of the grid. For instance, when the cluster of power lines fails, $UD_{113L}$ relative to load *113L* is almost the same both in case the load is critical and when it is not critical. This implies that no shedding of the other loads helps in reducing significantly the power loss of load *113L* when it is critical.

*Scenario with power lines characterized by different repair time*

This last set of analyses aims at exploring the effect of the repair time of the power lines on the undelivered demand to loads, so as to understand whether it would be valuable to improve the repair time to suffer less loss or not. Table 1 shows how the default repair times of the power lines defined in Figure 5 are distributed.

Figure 14 displays the extremes power losses incurred by the power system when the single power line $(i, j)$ fails and the repair times have the same extreme values: 10 hours or 32 days. For the sake of comparison, we also consider the

| Number of power lines | Repair time (hours) | Percentage of total |
|---|---|---|
| 25 | 10 | 44.6 |
| 24 | 11 | 42.9 |
| 1 | 16 | 1.8 |
| 1 | 35 | 1.8 |
| 5 | 768 | 8.9 |

Table 1.: Distribution of the repair times of the 56 power lines.

case of an intermediate value and the default setting (shown in Figure 5) for the repair times.

As expected, in general decreasing the repair time of each power line brings benefit to the measure UD. Of course, given a repair time value, its impact on UD varies according to which is the power line showing that value of repair time. For example, it can be observed that the worst repair time considered, that is 32 days, leads to the highest values of UD when it is associated to power line $(116, 117)$. But at the same time, even power lines with high repair time may have negligible impact on the undelivered demand. In fact, it is interesting to observe that only power line $(103, 124)$ among the five having the highest repair

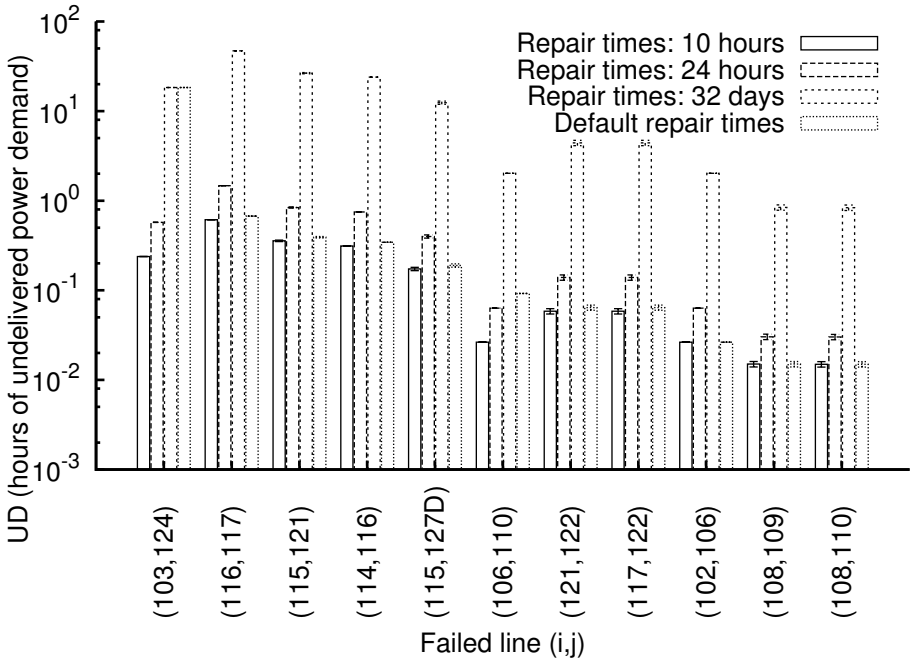


Figure 14.: UD, at varying the failure of power line $(i, j)$, on the x axis, for extreme values of the repair time.

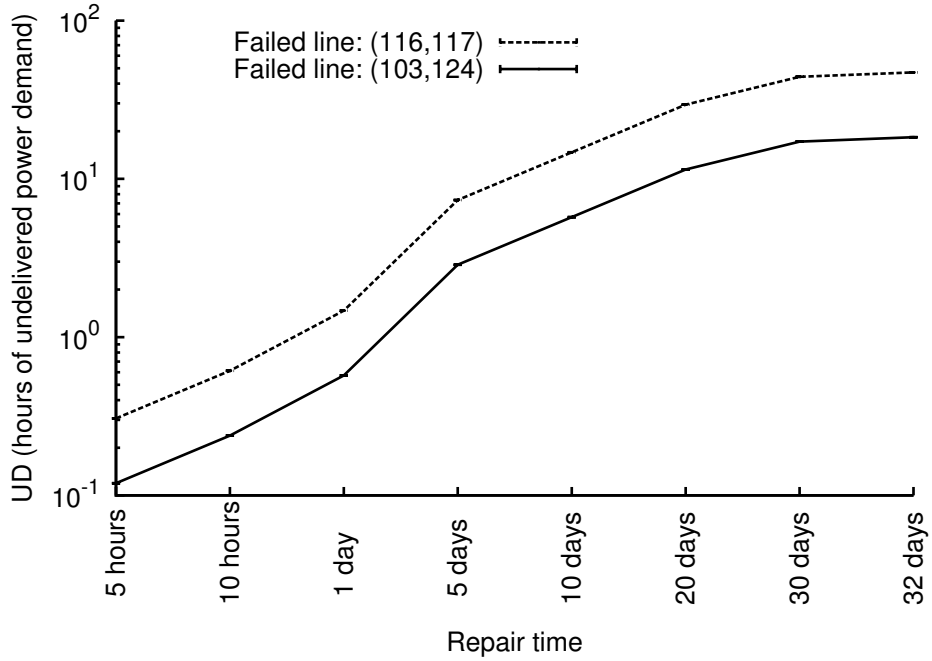


Figure 15.: UD, at varying the repair time, on the x axis, when a single power line $(103, 124)$ or $(116, 117)$ is failed.

time in the default setting, i.e., 768 hours, appears in the figure, meaning that the others have no impact on the loss of demand.

Finally, Figure 15 shows how UD varies as a function of different values of the repair times, when the single power line $(103, 124)$ or $(116, 117)$ is failed. Failure of power line $(116, 117)$ is more critical than the other power line, thus suggesting possible improvement of repair time for the former to get lower values for UD.

## 3.3 CONCLUSIONS

The main objective of this part of the thesis is to show, through stochastic model-based approach, how it is possible to support system design decision or system management with reference to Critical Infrastructures. Specifically, the thesis extends the results of the European Project CRUTIAL, regarding the electrical power systems, aiming at dealing with scenarios characterized by heterogeneity of the system components. By considering heterogeneity aspects, we represent more faithfully in our analyses the variety of processes that are in place in existing system configurations.

A novelty of this extension is mainly related to the new reconfiguration strategy, which allows to account for more realistic configurations of the grid (e.g., different loads criticality, different repair time of failed power lines, failure

of cluster of power lines), thus providing useful analysis results to understand the robustness of the analysed electrical grid, when affected by malfunctions in specific areas of the topology, or to set-up appropriate contractual policies with users requiring specific service conditions.

# 4

# MODEL-BASED APPROACHES TO IMPROVE DESIGN DECISION AND RUN-TIME ADAPTATION

The classic and well understood way of building dependable systems [23] is based on the application of rigorous development methods. Special programming techniques are used for software, such as model-driven development [63], and specific architectures are used for hardware, such as modular redundancy [22]. Dependability-critical domains, such as transportations and power plants, *require by law* the adoption of these techniques, and defined standards that must be followed.

This classic approach to dependability is challenged when critical systems are made up of networks of heterogeneous devices from different manufacturers. We live in the Future Internet (FI) era, which is characterized by unprecedented levels of connectivity and evolution. Software systems are increasingly pervasive, dynamic and heterogeneous, and many, even critical, aspects of modern society rely on their continuous availability and seamless interoperability.

More and more they are conceived as dynamically adaptable and evolvable sets of components that must be able to modify their behaviour at run-time to tackle the continuous changes happening in the unpredictable open-world settings [26]. Operating in the open-world poses a number of unprecedented challenges to software systems, including:

- The reference specification of expected/correct operation is not a-priori available.
- Specifications are learnt/inferred, thus they can be incomplete, unstable, uncertain, with impact on all the software engineering processes built upon system specification.
- System components are assembled dynamically, with potential strong impact on interoperability in presence of heterogeneity.
- Assessment activities must accommodate change (and must be adaptable themselves), therefore special emphasis is on run-time assessment (possibly coupled with off-line analysis techniques, wherever possible), which is a new paradigm with respect to traditional assessment methods.

As a result of such prominent trends, two related needs emerge. On the one side, we observe that the interconnected components, which we refer to as the Networked Systems (NSs), are independently developed. Because of this, the fast pace at which technology advances, along diverging tracks, can form gaps

and establish separately evolving technological islands, between which communication is hampered. Thus the state of practice is that ad hoc bridging solutions need to be continuously developed to fill those communication gaps. On the other side, the everyday life of modern and future society is growingly depending on the services provided by such highly complex and pervasive systems. In some cases their failures might even lead to catastrophic consequences in terms of damages to human life, environment, economy. Therefore, dependability and performance properties of such systems become increasingly critical.

## 4.1 CHALLENGES IN THE ASSESSMENT OF DEPENDABILITY AND PERFORMANCE OF EVOLVING SYSTEMS

The need for research advancement in the assessment of evolving, ubiquitous systems is recognized by the dependability/resilience community, being indicated as one of the prominent research challenges. In fact, it is observed that, since current and future systems result from evolutions of pre-existing systems, as a consequence assessment should move from off-line and pre-deployment, to continuous and automated operational assessment. The traditional approaches to assessment, which dominate the current practice, are: *i)* pre-deployment assessment, i.e. collecting data in a simulated environment (e.g., "model-based analysis", "statistical testing", "dependability benchmarking", etc.), and/or *ii)* processing the measurement data accumulated in real operation at a later stage, e.g., periodic reviews widely used in some safety-critical industries such as the nuclear sector. Both these categories of methods have shortcomings when dealing with evolution and dynamicity of the system under analysis. In fact, dealing with evolution and dynamicity raises two major challenges from the point of view of dependability and performance analysis:

- Pre-deployment assessment is limited by its nature: the impact on system dependability/performance cannot be known for unforeseen environments. Therefore, given the many possible variations occurring during software application lifetime, it would be necessary to analyse beforehand, through off-line analysis, all the possible scenarios which could take place at run-time, to be stored in a look-up table from which to retrieve the correct analysis upon a scenario's occurrence. But this cumbersome activity is in general impossible to conduct at a sufficiently satisfactory level, especially for critical applications subject to strong dependability requirements. Resorting to processing the measurements collected in real operation at a later stage, e.g., in periodic reviews, may be inadequate as well, since by the time the observations are processed the operational environment may have changed to something not yet seen before.
- Pre-deployment assessment, however, plays an important role in providing a priori knowledge about how the system is expected to operate, especially

if the simulated environment is "close" to the operational environment post deployment, and to take appropriate design decisions. Stochastic model-based assessment has been widely recognized as a helpful means to cover this role [20]. Nevertheless, the unavoidable higher chance of inaccurate/unknown model parameters needs to be considered as a weakness that could result in too inaccurate analysis results, thus negatively impacting design decisions.

To contribute to overcome such deficiencies of current methods in assessing dynamic systems, the thesis considers some of the results of the European Research Project CONNECT [46], and presents an approach useful to combine the benefits of both pre-deployment and processing of data obtained from real executions. Specifically, at pre-deployment time we propose an enhancement of the system through a model-based approach that exploits a predefined library of dependability and performance mechanisms to be implemented properly in the system, if dependability and performance requirements are not met after an initial analysis. The second approach, which begins after the deployment of the system, aims at refining the parameters of the models, by collecting the actual values at runtime, thus allowing to verify the correctness of the model and especially the correctness of the analysis results obtained at pre-deployment.
In order to clarify the context and the issues concerning dependability and performance aspects in heterogeneous interacting and evolving systems, the project CONNECT is briefly described in Section 4.3.

## 4.2 STATE OF THE ART

A number of solutions have been presented in the literature to deal with critical applications requiring degrees of fault tolerance or efficiency, including classical architectural mechanisms such as N-Version Programming, Recovery Block, Cooperative Backup and dynamic mechanisms such as dynamic function allocation [83, 90, 111, 135]. There are also studies where the most popular fault tolerance solutions are discussed and classified according to a variety of key characteristics, e.g., the main direct benefits of the mechanism in terms of improved system resilience or assurance of key system properties and the types of system within which this mechanism can be applied [111], with the intention to assist in the selection and configuration of mechanisms at design-time and run-time. Also, modelling and analyses of fault tolerant systems have been widely pursued, to help understanding the adequacy of employed fault tolerance mechanisms in terms of dependability, resilience and security indicators, under a variety of failure modes and critical system scenarios. Applied approaches include reliability block diagrams, fault trees, Markov models, Petri nets, Bayesian models, semantic technologies. A very rich literature has been produced which documents performed studies (such as [56, 61, 62, 66, 90] just to

mention a few). A review of several technologies supporting dynamic selection of components and services, such as Multi-Agent Systems, GRID computing, Web Services, is also included in [111]. In [98] the authors address the interoperability problems raised by semantic heterogeneities in Web services communities by using a context-based approach that separates shared knowledge from the local contexts of Web service providers and developing mediation mechanisms that handle semantic data discrepancies between Web services and communities, thus enabling seamless interoperation at the semantic level. In [101] the authors examine challenges to interoperability; classify the types of heterogeneities that can occur between interacting services and present a possible solution for data interoperability using the mapping support provided by WSDL-S. Here, we are not tight to any specific technology, but rather we are focusing on generally applicable solutions in the specific context of dependable networks interoperability in heterogeneous environments.

Solutions for automatic addition of fault tolerance to fault intolerant programs have been also proposed, both resorting to a set of symbolic heuristics for synthesizing fault-tolerant distributed programs such as in [32] and by developing formal algorithms working in presence of a specific set of faults, a safety condition, and a reachability constraint as in [116], where it is also shown that the synthesis problem in the context of distributed programs is NP-complete in the state space of the given intolerant program. Further investigations on theoretical aspects and the development of a software framework for the synthesis of fault-tolerant programs are in [58].

Also, in the context of *reflective* systems, i.e. systems having the property of enabling observation and control of its own structure and behaviour from outside itself, independent development of fault tolerance libraries and applications has been carried on as a means to enhance the flexibility of dependable systems. The aim was to provide properties like: ease of use and transparency of mechanisms for the application programmer, seamless reuse and extension of both functional and non-functional software and composition of mechanisms [114]. Moving to autonomic systems [73], their control loop requires assistance by a methodology and supporting technique to select and apply countermeasures to faults experienced during the system lifetime, possibly resorting to a library of fault tolerance patterns. However, to the best of our knowledge, there is no general approach in the literature, developed to select from a library of fault tolerance mechanisms, whose efficacy to meet specified dependability and performance requirements is checked through model-based analysis. Part of the thesis aims at exploring this direction.

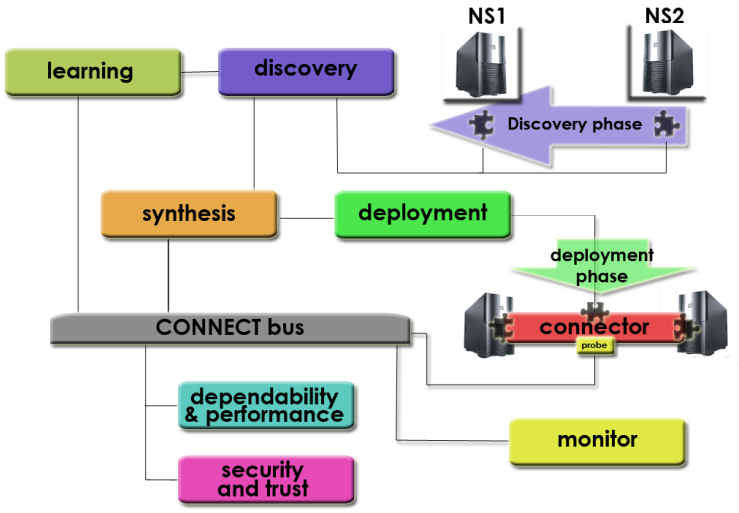


Figure 16.: The CONNECT architecture

## 4.3 CONNECT PROJECT OVERVIEW

The European Project CONNECT addressed the need of enabling seamless and dependable interoperability among Networked Systems (NSs) in spite of technology diversity and evolution. The goal of the project was to have eternally functioning distributed systems within a dynamically evolving open-world context, through the *on-the-fly* synthesis of the CONNECTors by means of which heterogeneous NSs can communicate in dependable and secure way.

Indeed, effective interoperability requires ensuring that such "connected systems" provide the required non-functional properties and continue to do so even in presence of evolution, thus calling for enhanced and adaptive assessment frameworks.

Figure 16 provides an overview of the CONNECT vision and architecture. In brief, the NSs manifest the intention to connect to other NSs. The *Enablers* are networked entities that incorporate all the intelligence and logic offered by CONNECT for enabling the required connection. In the following are briefly described the enablers that are part of the enabling architecture:

DISCOVERY ENABLER: discovers the NSs, catches their requests for communication and initiates the CONNECT process. We tend to make the minimum possible assumptions on the information (called the *affordance*) that NSs must provide.

LEARNING ENABLER: it uses active learning algorithms to dynamically determine the interaction behaviour of a NS and produce a model in the form of a Labeled Transition System (LTS).

SYNTHESIS ENABLER: from the models of the two NSs, this enabler synthesizes a mediator component through automated behavioural matching.

DEPLOYMENT ENABLER: deploys and manages the synthesized CONNECTORS.

MONITOR ENABLER: collects raw information about the CONNECTORS behaviour, filters and passes them to the enablers who requested them.

SECURITY AND TRUST ENABLER: collaborates with the synthesis enabler to satisfy possible security and trust requirements. It also continuously determines if the requirements are maintained at run-time, by receiving monitoring data from the monitoring enabler.

DEPENDABILITY AND PERFORMANCE ENABLER: this enabler assesses dependability and performance properties of the CONNECTOR and is the module on which the thesis mainly focuses. We describe this enabler in detail in Section 4.4.

## 4.4 THE DEPENDABILITY AND PERFORMANCE ENABLER

This section aims at providing a general overview of the Dependability and Performance Enabler (called DEPER), which is part of the CONNECT's architecture previously described. The main purpose of this enabler is to perform analysis of the synthesised CONNECTOR in order to verify whether given non-functional requirements, namely dependability and performance, are met by the CONNECTOR. As already introduced in the previous section, the enabler interacts with other enablers in the CONNECT framework to: *i)* be triggered on the analysis to perform and take in input both the specification of the system to analyse and the metric to assess together with the value required for it; *ii)* provide the feedback from the analysis to Synthesis, which can then proceed with the deployment of the CONNECTOR or refine it according to the feedback; *iii)* check and, if necessary, refine the accuracy of pre-deployment model parameters through on-line observations, by interacting with the Monitor enabler.
The enabler is composed by five modules whose activities start from pre-deployment assessment of the generated bridging CONNECTORS to subsequent refinements based on run-time observations of real networked systems and CONNECTORS executions. Figure 17 shows the logical architecture of the Dependability and Performance enabler and the modules composing it, which are described in detail in the following subsections.

### *Builder Module*

The Builder module takes in input the specification of the CONNECTed system. This specification is given as Labeled Transition Systems (LTSs) annotated with

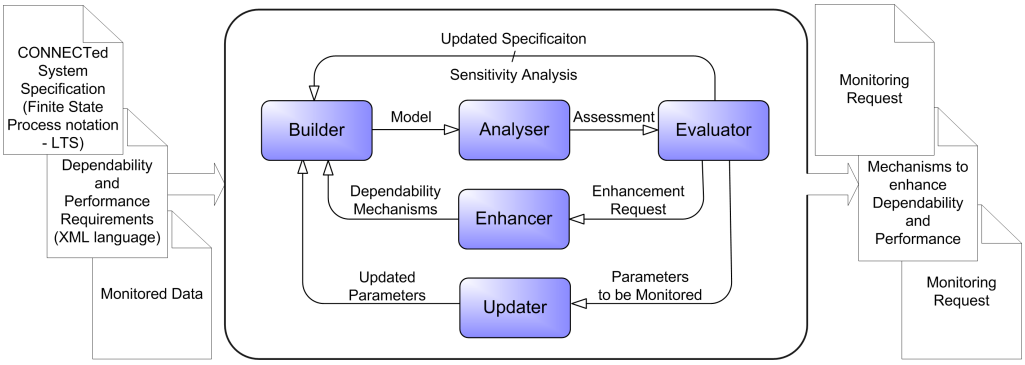


Figure 17.: The Dependability and Performance enabler architecture

non-functional information necessary to build the dependability and performance model of the CONNECTed system. Annotations include, for each labeled transition, the following fields: time to complete, firing probability, and failure probability. The module produces in output a dependability and performance model of the CONNECTed system suitable to assess the given dependability and performance requirements. Such model is specified with a formalism suitable to describe complex systems that have probabilistic behaviour, e.g., stochastic processes.

*Analyser Module*

The Analyser module takes in input the dependability and performance model from the Builder module and the dependability and performance properties required by the NSs from Discovery/Learning. These requirements are expressed as metrics and guarantees. Metrics are arithmetic expressions that describe how to obtain a quantitative assessment of the properties of interest of the CONNECTed system. To allow for automated assessment, they are expressed in terms of transitions and states of the LTS specification of the NSs. Guarantees are boolean expressions that are required to be satisfied on the metrics. The module extends the received model with *reward functions* suitable to quantitative assessment of the metrics of interest. Then, it makes use of a solver engine to produce a quantitative assessment of the dependability and performance metrics.

*Evaluator Module*

The Evaluator module is in charge of checking whether the analysis results match with the guarantee, as requested by the networking systems willing to communicate, or not. Evaluator informs the Synthesis enabler about the outcome of the analysis and, in case of mismatch, it may receive back a request to evaluate

if enhancements can be applied to improve the dependability or performance level of the CONNECTed system, namely:

- To take into account an alternative CONNECTor deployment (e.g., a deployment that uses a communication channel with lower failure rate). A new analysis is triggered, considering the updated specification of the CONNECTor.
- Enhance the specification of the CONNECTor by including dependability mechanisms, which are counter-measures to contrast failure modes affecting performance and/or dependability metrics (e.g., a message retransmission technique). Such mechanisms are then applied by the Enhancer module to model elements that are considered weak from the point of view of the metric under assessment.

In case the analysis results, provided by Analyser, match with the guarantee, the CONNECTor's design is considered satisfactory and ready to be deployed, thus terminating the pre-deployment analysis phase. However, because of possible inaccuracy of model parameters, due to potential sources of uncertainty dictated by the dynamic and evolving context, Evaluator also instructs the Updater module about the events to be observed on-line by the Monitor enabler. Collection of such events allows DEPER to determine whether a new analysis needs to be performed, to properly adapt to changes (or unforeseen circumstances), as better detailed later in Section 4.6.

*Enhancer Module*

The Enhancer module is activated by the Evaluator module when the guarantees are not satisfied and the Synthesis enabler makes a request to enhance the CONNECTor with dependability mechanisms. Enhancer is instructed by the Evaluator module with indications about how to select the dependability mechanism to try and to which elements of the original model the mechanism has to be applied. Then, it performs the following actions: *i)* selects the dependability mechanisms that can be employed, among those available in the category indicated by Evaluator; *ii)* instructs the Builder module on the application of the selected dependability mechanism in the CONNECTed system model, in accordance with indications from Evaluator, and triggers a new analysis. At the end of this new analysis, Evaluator verifies whether the enhanced CONNECTor fulfills the dependability and performance requirements. If the requirements are satisfied, the Evaluator module informs the Synthesis enabler about the mechanism to add to the CONNECTor design and the DEPER's support to the design of this CONNECTor is completed. Otherwise, Enhancer makes a further attempt with the next dependability mechanism (if available), according to some internal predefined policies about the rank of available mechanisms and about how to apply them to model elements provided by Evaluator, and a new cycle

with Builder, Analyser and Evaluator is repeated. This loop ends either when a successful mechanism is found, or when all the mechanisms are exhausted. A library of models for triggering the generation of typical dependability mechanisms suitable to contrast two typical classes of failure modes that may happen during interactions is described in Section 4.5.2.
The thesis extends the Enhancement module by providing an approach for the automatic selection of dependability and performance mechanisms and the weakness of the CONNECTOR where to implement such mechanisms. The enhancement approach, which is based on a pre-deployment stochastic model-based analysis, is fully describe in Section 4.5.

*Updater Module*

The Updater module is activated once the CONNECTOR has been deployed. It is in charge to update the values of model parameters used in the analysis, on the basis of the data related to real executions of the CONNECTED system as gathered through a monitoring infrastructure. Indeed, keeping the analysis model updated is very important in the CONNECT context, to cope with evolution of NSs and possible inaccurate information known at pre-deployment time. Based on the monitored data, this module may trigger a new analysis on the updated model by reactivating the cycle on the Builder, Analyser and Evaluator. This module is fully described in Section 4.6, when focusing on the on-line adaptive assessment.

## 4.5 DEPENDABILITY AND PERFORMANCE ENHANCEMENT

As already mentioned, pre-deployment assessment is a crucial activity to drive the system design towards a realization compliant with the required level for quality of service indicators. In fact, it allows for early detection of design deficiencies, so as to promptly take the appropriate recovery actions, thus significantly saving in money and time with respect to discovering such problems at later stages. Also, it is central to the decision making process among alternative design solutions, again gaining in efficiency and better guarantee to end up with the "right" system. The pre-deployment assessment part of the proposed method, tailored to dynamic and evolving systems assessment, exploits State-Based Stochastic modelling and analysis, which are embedded in an automated process.

In this section, we point the attention on a dependability and performance enabler's module, denominated Enhancer. Such a module, as already described, is responsible for guiding the synthesis process towards enhancements of a CONNECTOR's design whenever the analysis reveals inadequate dependability levels. In brief, this module is in charge of selecting a combination of dependability and performance mechanisms suitable for enhancing the synthesised CONNECTOR so

that it complies with given requirements. Then, the synthesis enabler embeds the selected dependability mechanisms in the CONNECTOR's design and proceeds with its implementation and deployment.

### 4.5.1 *Rationale behind the approach*

The selection of dependability mechanisms for enhancing CONNECTORs is typically driven by: *i)* application constraints imposed by the application domain, e.g., remote-control applications may have different constraints in terms of timing and tolerance to degraded service from video-based applications; *ii)* fault & failure assumptions for devices during their operational life, e.g., transient faults or permanent faults; *iii)* dependability metrics relevant to given dependability requirements that must be met, e.g., message delivery time, coverage of receivers.

***Application constraints*** may have an impact on the system tolerance to degraded service (meaning degradation both in the value and time domains), and the ability to deploy specific solutions. In some cases this may hinder the benefits of certain dependability mechanisms. For instance, consider a dependability mechanism based on error-correction that can be used to detect transmission errors and thus reconstruct the original error-free data. Given that the CONNECT framework aims at modifying only the CONNECTOR, the mechanism can be applied only if the original error-free data are available at the transmitter-side of the CONNECTOR, e.g., either when the CONNECTOR can be deployed on the transmitter, or when a reliable channel exists between the transmitter and the infrastructural element with the deployed CONNECTOR.

***Fault & Failure assumptions*** are at the basis of dependability and performance models. They are typically based on assumptions about the malfunctions that may undermine the dependability or performance indicators of the system under analysis during its operational life. The nature of the assumed faults and/or failures guides the selection of countermeasures. Avizienis et al [23] proposed a taxonomy of the faults and failures in the dependability community. For the generic dependability and performance mechanisms considered we focus mainly in: *i)* the persistence of the fault, that is whether the fault is transient (maybe, in bursts, but lasting a limited amount of time), or permanent; *ii)* the objective of the fault, namely if it is accidental or intentionally introduced in the system; *iii)* the failure domain, that is content failures when the content of the information delivered at the service interface deviates from implementing the system function, or timing failures when the time of arrival or the duration of the information delivered at the service interface deviates from implementing the system function. Moreover, we discuss in the next section how the dependability

mechanisms we adopt relate with these faults and failures assumptions.

***Dependability metrics*** guide the definition of the assessment model, which has to faithfully include all the system aspects with a relevant impact on the measure under evaluation while abstracting or even neglecting all the other behaviours/phenomena with negligible impact, in order to keep the model as much as possible manageable and controllable. The measure also influences the choice of the dependability mechanism, characterised by differing structure and operational behaviours. We mainly address two categories of measure for stochastic quantitative analysis: performance-related one (including, e.g., variants of the latency indicator), and dependability-related one (including, e.g., different forms of coverage as percentage of networked services receiving/offering a service with respect to the full population).

In the literature, there is no evidence that relying just on one of the factors illustrated above is the best choice. Therefore, the selection method we propose, (illustrated further in Section 4.5.3) considers a combination of these identified factors.

The following section describes the basic library of dependability and performance mechanisms for the enhancer. The library builds on classical dependability patterns, and uses the concept of overlay networks for triggering the synthesis of specific dependability mechanisms in the synthesised CONNECTOR from high-level specifications.

### 4.5.2 *Library of dependability and performance mechanisms*

Overlay networks are virtual networks built on top of existing network substrates: in overlay networks, nodes represent logical hosts involved in interactions, and links correspond to paths in the network substrate that are traversed by messages during inter-operations. To date, overlay networks have been used for exploiting peculiar characteristics of network substrates at the application level; for instance, Andersen et al [19] exploit highly redundant networks substrates for defining resilient communication systems as overlay networks.

Thanks to the infrastructure provided by CONNECT, we can use the concept of overlay networks in an alternative way, i.e., rather than using overlay networks for exploiting the characteristics of the network substrate, we use them for defining the characteristics of the network substrate that should be synthesised. The basic idea is to view CONNECTORS as overlay networks, and to exploit their structure for triggering the generation of specific dependability and performance mechanisms in the network substrate during the synthesis process.

In the following, we describe the models for triggering the generation of typical dependability mechanisms suitable to contrast two typical classes of failure

modes that may happen during interactions: *timing failures*, in which networked systems send messages at time instants that do not match an agreed schedule, and *value failures*, in which networked systems send messages containing incorrect information items. We consider timing failures of type omission, i.e., late messages are always discarded, and value failures that cause a networked system to respond within the correct time interval but with an incorrect value. Each model is specified with the Stochastic Activity Network (SAN) formalism. We develop the models according to three basic rules that allow to simplify the automated procedure for embedding the mechanism in the specification of the synthesised CONNECTOR: *i)* each model has an initial place, `s0`, whose tokens enable the first activity of the model; *ii)* each model has a final place, `s1`, which contains tokens whenever the last activity of the model completes; *iii)* the overall number of tokens in `s1` is always less or equal to the number of tokens in `s0`. With the above rules, the behaviour of the model can be seen as an enhanced activity, and can be directly used to replace any activity that moves tokens between two places in the specification of the CONNECTOR (the basic semantics of an activity is always preserved). The next subsections describe the dependability and performance mechanisms, which are: *Retry Mechanism*, *Probing mechanisms*, *Majority Voting*, *Error Correction*, and *Parallel Retry*. For illustrative purpose, we show and describe more in detail the Retry and the Probing mechanisms.

*Retry Mechanism*

The retry mechanism consists in re-sending messages that get corrupted or lost during communications, e.g., due to transient failures of communication links. This mechanism is widely adopted in communication protocols, such as TCP/IP [33] for enabling reliable communication over unreliable channels. A typical implementation of the retry mechanism uses time-outs and acknowledgements: after transmitting a message, the sender waits for a message of the receiver that acknowledges successful communication. If the acknowledgement is not received within a certain time interval, the sender assumes that the communication was not successful, and re-transmits the message. The synthesis of a retry mechanism is triggered with the stochastic activity network shown in Figure 18. On the sender side, the mechanism creates message re-transmission policy for re-sending the message at most $N$ times; on the receiver side, the mechanism creates a policy for avoiding duplicated reception of messages and for sending acknowledgements. In the model, all places initially contain zero tokens, except `p0`, which contains $N$ tokens, where $N$ is a model parameter representing the maximum number of re-transmissions. The activity `send` is enabled when the conjunction of the following conditions is true: `p0` and `s0` contain at least one token, and `ackReceived` contains zero tokens. When the activity `send` completes with success (`case 0`, with probability $\mathrm{pr}_0$), a token is removed from `s0` and `p0`, and the marking of `p1` is incremented by one. The activity

`receive` is enabled when `p1` contains at least one token and `messageReceived` contains zero tokens. When the activity `receive` completes, a token is moved to `s1`, and the marking of `p2` and `messageReceived` is incremented by one. A token in `p2` enables activity `sendAck`, whose aim is to enable the receiving host notify the sender that the message has been successfully received. The sender stops re-transmitting the message as soon as it gets an acknowledgement that the message has been successfully received, or after $N$ attempts.

*Probing Mechanism*

The probing mechanism exploits redundant paths and periodic keep-alive messages for enabling reliable communication in face of path failures. The basic idea is to continuously collect statistics on the characteristics of the communication channels, and to select the best channel on the basis of such statistics. This mechanism has been used for defining communication services with guaranteed delivery and performance levels, e.g., see Akamai's SureRoute [16] and reliable multi-cast protocols for peer-to-peer networks [141]. The synthesis of a probing mechanism that uses two redundant communication channels is triggered with the stochastic activity network shown in Figure 19. The mechanism instruments the sender with a periodic channel probing functionality suitable to feed a monitoring system that collects statistics about the reliability level of the communication channels. In the model, the place `mode` is a state variable that indicates the mode of operation of the mechanism, which can be either probing mode (`mode` contains zero tokens), i.e., the mechanism tests the characteristics of the communication channels through keep-alive messages, or normal mode (mode contains one token), i.e., the mechanism selects the best estimated channel for relaying messages. Initially, all places contain zero tokens, except `ready`, which contains one token. When in normal mode, the activity `select` is enabled when `s0` contains at least one token, and `ready` contains one token. When `select` completes, one token is removed from `s0` and `ready`, and the activity `send0` gets enabled if `monitor0` has more tokens than `monitor1` (`send1` gets enabled in the

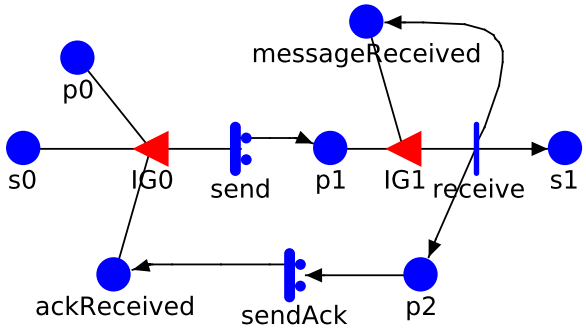


Figure 18.: Retry mechanism

other case). If *send0* completes with success (`case 0`), then a token is added to *s1* and *ready*. Similarly, when *send1* completes with success, a token is moved to *s1* and *ready*. When in probing mode, the model behave as follows: *send0* and *send1* have both the same rate $R_0$ (while their case probabilities depend on the characteristics of the channels, which may vary over time). The activity *select* is enabled when *ready* contains one token; when *select* completes, *ready* contains zero tokens and activities *send0* and *send1* get enabled (by moving one token in both *p0* and *p1*). When *send0* completes with success (case 0), a token is added to *monitor0*. Similarly, when *send1* completes, a token is added to *monitor1*. A token is moved to *ready* when both *send0* and *send1* complete.

*Majority Voting Mechanism*

This is a fault-tolerant mechanism that relies on a decentralised voting system for checking the consistency of data. Voters are software systems that constantly check each other's results, and has been widely used for developing resilient systems in the presence of faulty components. In a network, voting systems can be used to compare message replicas transmitted over different channels. It is rather expensive in redundancy, with benefits from the point of view of enhancements of several properties (integrity, safety, reliability, and maintainability). But the redundancy degree, the synchronization among the multiple executions and the additional voting execution result in a degradation of latency and throughput aspects with respect to the non-redundant Connector solution. The majority voting mechanism that uses three redundant communication channels is triggered with the stochastic activity network shown in Figure 20.

*Error Correction Mechanism*

This mechanism deals with the detection of errors and re-construction of the original, error-free data. A widely used approach for enabling hosts to automatically detect and correct errors in received messages is Forward Error Correction (FEC). The mechanism requires the sender host to transmit a small

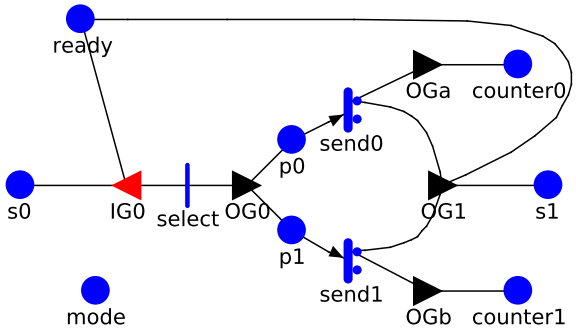


Figure 19.: Probing mechanism

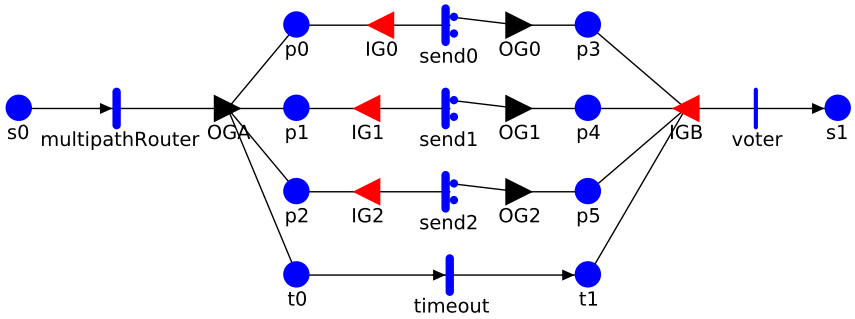


Figure 20.: Majority Voting mechanism

amount of additional data along with the message, thus causing a worsening of latency and throughput, but providing an improvement on reliability and integrity. With respect to potential timing constraints imposed by the applications, this mechanism is classified as suitable to comply with up to a medium level. The error correction mechanism that uses two redundant communication channels is triggered with the stochastic activity network shown in Figure 21.

*Parallel Retry*

This mechanism is similar to the Retry mechanism, but exploits channels redundancy (instead of successive retries on the same channel) to speed up the communication time.

### 4.5.3 *A methodology for the automatic selection of dependability and performance mechanisms*

The CONNECT project uses ontologies [69] as the basis for specifying the behaviour of networked systems. Specifically, a semantic description of the behaviour of the networked systems is used to identify peers that need to be connected to enable a service. Following an ontology-based approach similar to the one used for functional mediation, we define a general method for selecting a

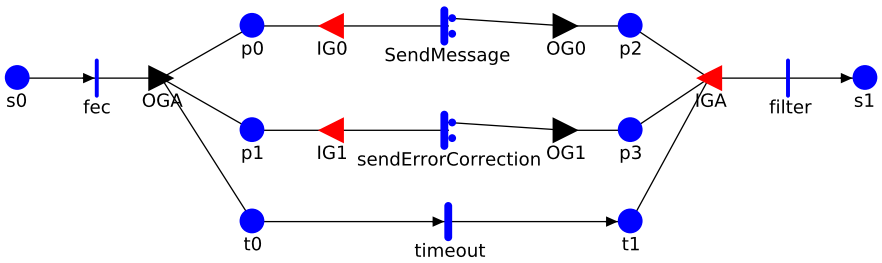


Figure 21.: Error Correction mechanism

dependability mechanism for CONNECTors. The method consists of the following main steps:

STEP 1. An ontology-based characterisation of the dependability mechanisms is created that points out how the mechanism is linked to application constraints, fault & failure assumptions and dependability metrics.

STEP 2. A semantic matching based on the developed ontology is performed when a dependability mechanism is required for the CONNECTor. The aim is to identify and give a rating to those dependability mechanisms that are relevant to the type of mismatch revealed by the performed model-based dependability and performance analysis of the CONNECTor.

STEP 3A. If only one identified mechanism is rated the highest, then this mechanism is selected.

STEP 3B. If more than one mechanism have same (highest) rating, then a prioritisation among application constraints, fault & failure assumptions and dependability metrics are used for ranking the identified mechanisms.

We implement the **Step 1** of the method by creating a table of the ontology of dependability mechanisms for CONNECTors, previously introduced. Table 2 shows such an ontology and defines a mapping relation between dependability mechanisms, mitigated threats, affected dependability metrics (both enhanced and deteriorated), and application constraints to be satisfied. Specifically, Table 2 specifies for threats: type (e.g., *omission* or *value*), duration (*transient* or *permanent*) and nature (*accidental*, i.e. natural or human made but without malicious intent, or *intentional* [23]) of the threat (column *Mitigated Threat*). Concerning the dependability metrics, the table indicates which metrics can be enhanced by the dependability mechanism (column *Enhanced Metric*), and which are potentially deteriorated (column *Deteriorated Metric*). We focus on the classical dependability and performance metrics (e.g., reliability, safety, integrity, latency); however, it is possible to extent this set to include other dependability and performance related metrics, such as the *coverage* metric.

Finally, Table 2 shows, in the last column (*Satisfied Constraint Application*), the application constraints satisfied by the mechanism. As briefly discussed, such application constraints are application specific and are mainly related with the ability of the application to tolerate degraded services and to provide enabling facilities to deploy dependability and performance mechanisms. Since we do not target any specific application, we consider only the classical timing constraint, grading it in *none*, *soft*, *medium*, and *hard*. Again, extensions to consider other kinds of applications constraints can be easily accommodated. The ontology table includes the five basic dependability mechanisms already described in the previous Section 4.5.2 as a suitable core to enhance CONNECTor dependability and performance. Specifically, the *retry* mechanism can be used to enhance

the reliability when non-intentional transient faults result in omitted data or data with wrong values. However, the ontology indicates that the mechanism is likely to deteriorate two other quality parameters of the CONNECTOR: *latency* and *throughput*. Given the longer execution time due to the retry, it complies with applications having none or soft timing constraints only. The *probing* mechanism is useful to enhance the reliability and the latency of the CONNECTOR, and it is compliant with hard timing constraints. However, implementing such mechanism leads to a throughput deterioration of the system, due to additional procedures to select the best channel. The *majority voting* mechanism is expensive in redundancy, with benefits from the point of view of enhancements of several properties (integrity, safety, reliability, and maintainability). But the redundancy degree, the synchronization among the multiple executions and the additional voting execution result in a degradation of latency and throughput aspects with respect to the non-redundant CONNECTOR solution. The *error correction* mechanism requires the sender to transmit a small amount of additional data along with the message, thus causing a worsening of latency and throughput, but providing an improvement on reliability and integrity. With respect to potential timing constraints imposed by the applications, this mechanism is classified as suitable to comply with up to a *medium* level. The *parallel retry*, similar to the retry mechanism, exploits channels redundancy to speed up the communication time. In this way it allows to improve latency, reliability, and maintainability, but at the same time it may reduce the throughput of the system due to the

| **Ontology of dependability mechanisms for CONNECTORS** | | | | |
|---|---|---|---|---|
| **Mechanism** | **Mitigated Threat** | **Enhanced Metric** | **Deteriorated Metric** | **Satisfied Application Constraint** |
| Retry | Transient, Accidental, Omission or Value failure | Reliability | Latency, Throughput | Soft Timing |
| Probing | Transient or Permanent, Accidental, Omission failure, | Reliability, Latency | Throughput | Hard Timing |
| Majority voting | Transient or Permanent, Accidental or Intentional Value failure | Integrity, Safety Reliability, Maintainability | Latency, Throughput | Medium/Hard Timing |
| Error correction | Transient, Accidental or Intentional Omission or Value failure | Reliability, Integrity | Latency, Throughput | Medium Timing |
| Parallel retry | Transient or Permanent, Accidental, Omission or Value failure | Latency, Reliability Maintainability | Throughput | Hard Timing |

Table 2.: Example of ontology of dependability enhancement mechanisms for CONNECTORS.

redundancy. Concerning its ability towards timing constraints, this mechanism appears suitable to satisfy up to *hard* constraints.

It is important to underline that Table 2 considers general characteristics of the selected dependability mechanisms, but of course, a more accurate matching with respect to their ability to enhance (or decrease) dependability and performance metrics, as well as to satisfy application constraints, depends on the specific implementation of such mechanisms, and on the quantification of the grades of timing constraints (here simplified in none, soft, medium, hard), which maybe application dependent.

By using the defined ontology, the proposed approach for automated selection of a dependability mechanism for CONNECTORS, illustrated at the beginning of this section, leads to Table 3. The table presents the mechanisms following general risk assessment principles: it establishes a relationship between threats, application constraints, and risk reduction strategies (Dependability Mechanisms). Table 3 presents a combinations of *Threats* and *Application constraints*, and for each of them selects from Table 2 a set of dependability mechanisms suitable to cope with them. Of course, this table is not exhaustive; thus, it is expected to be extended with additional combinations of *Threats* and *Application constraints*, as well as additional dependability and performance mechanisms to enable CONNECTORS enhancement in different application scenarios.

### 4.5.4 *Where to apply the dependability mechanism*

The most accurate method to identify element(s) of the CONNECTOR that should be enhanced through the selected dependability mechanism is based on a systematic exploration of the benefits of the mechanism to each individual element in the CONNECTOR. The element for which the benefit is the highest is the best candidate. While very accurate, this approach is time consuming. We propose an alternative method that provides a good trade-off between time and accuracy in several practical cases. The idea of the approach is to perform an exploratory investigation based on simple probabilistic formulations for finding out where to apply the dependability mechanism. The approach relies on two tailored specific cases: dependability-related metrics and performance-related metrics.

We illustrate in detail the exploratory investigation by considering a typical situation encountered in model-based analysis of interoperable devices. We specify communication protocols in terms of timed activities. Each timed activity $a_i$ is characterised by a time to complete $t_i$ and by a failure probability $q_i$, independently from the others. For illustrative purpose, we assume that dependability-related metrics are mainly influenced by failure probability, while performance-related metrics are mainly influenced by the time to complete. Un-

| Table for automated selection of dependability mechanisms for CONNECTORS | | |
|---|---|---|
| **Threat** | **Application constraint** | **Mechanism** |
| Transient,<br>Accidental,<br>Omission failure | Hard Timing | Parallel Retry<br>Probing |
| Transient,<br>Accidental,<br>Omission failure | None | Parallel Retry<br>Probing<br>Error correction<br>Retry |
| Transient,<br>Accidental,<br>Omission failure | Medium Timing | Parallel Retry<br>Probing<br>Error correction<br>Retry |
| Permanent,<br>Accidental,<br>Omission failure | Medium Timing | Parallel Retry<br>Probing |
| Transient,<br>Accidental or Intentional,<br>Value failure | None | Majority voting<br>Error Correction<br>Retry |
| Transient,<br>Accidental or Intentional,<br>Value failure | Medium Timing | Majority voting<br>Error Correction<br>Retry |
| Permanent,<br>Accidental,<br>Value failure | Medium Timing | Majority voting<br>Parallel Retry |
| Permanent,<br>Accidental,<br>Value failure | None | Majority voting<br>Parallel Retry |
| Permanent,<br>Intentional,<br>Value failure | Medium Timing | Majority voting |
| Permanent,<br>Accidental,<br>Value failure | Hard Timing | Parallel Retry<br>Majority voting |

Table 3.: Example of table for automated selection of dependability mechanisms for CONNECTORS.

der these assumptions, two automated strategies (one for dependability-related metrics, one for performance-related metrics) can be defined for identifying elements in the CONNECTOR that should be enhanced with dependability mechanisms.

### 4.5.5 *Dependability-related metrics case*

We start with the simple case where: *i)* the metric under analysis is a function of the failure probabilities of all the activities in the model representing the CONNECTOR; *ii)* each activity can either be successfully completed or fails. This means that there is only one path in the model that results in a successful execution of the CONNECTOR. Figure 22a illustrates this case.

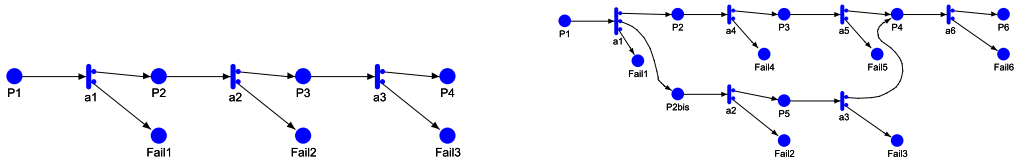


Figure 22.: Example of a basic CONNECTOR model (a), and of a more complex CONNECTOR with two branches (b)

Let us consider that N activities are included in the model of the CONNECTOR under analysis. Then, the selection of the activity to be enhanced is simply determined through the following steps:

1. for each activity $a_i$, determine the new value $\bar{q}_i$ resulting from applying the dependability mechanism to activity $a_i$;
2. for each activity $a_i$, compute the value of the failure probability $Q_i$ of the whole CONNECTOR when using the new value $\bar{q}_i$. The general formulation of the CONNECTOR failure probability Q is given by the expression:
   $Q = q_1 + \sum_{i=2}^{N} q_i \cdot \prod_{j=1}^{i-1} (1 - q_j)$

   For each index i, $Q_i$ is obtained by substituting $\bar{q}_i$ to $q_i$ in the formula of Q.
3. select the activity $a_j$ whose new probability value $\bar{q}_j$ determines a $Q_i$ that is the minimum among the N values of $Q_i$.

To exemplify the approach, let us consider the model in Figure 22a and the failure probabilities $q_i$ for two dependability mechanisms (majority voting and retry), as reported in Table 4. According to the values in the table, activity $a_1$ should be enhanced when applying majority voting, while activity $a_3$ should be enhanced when using retry.

Generalisations of this simple case can be performed in order to take into account that: *i)* not all the activities $a_i$ are involved in the dependability metric under analysis, and *ii)* more than one path is possible within the dependability model to reach $a_i$ from the start activity (e.g., as illustrated in Figure 22b).

| activity | $q_i$ | $\bar{q}_i$(Maj. voting) | $\bar{q}_i$(Retry) | $Q_i$(Maj. voting) | $Q_i$(Retry) |
|---|---|---|---|---|---|
| $a_1$ | 0.20000000 | 0.10400000 | 0.04000000 | 0.66104320 | 0.63683200 |
| $a_2$ | 0.03000000 | 0.00264600 | 0.00090000 | 0.68882555 | 0.68828080 |
| $a_3$ | 0.35000000 | 0.28175000 | 0.12250000 | 0.66558280 | 0.59143600 |

Table 4.: Illustrative example of the selection approach for dependability-related metrics

With respect to point i), the generalisation is easily performed by determining the new values $\bar{q}_i$ only for those activities $a_i$ involved in the formula expressing the metric and consequently restricting the calculation of $Q_i$ only to the $\bar{q}_j$ computed. Then, the last step remains the same.

The second generalisation does not require any actual modification to the described basic method. What changes is just the formula expressing the CONNECTor failure probability Q, which has to take into account the different paths which may lead to failure.

An interesting observation concerns the formula expressing the improvement in reliability gained through the application of the schemes for which a dependability model has been generated (e.g., those in Table 3), necessary to derive the new values $\bar{q}_i$ associated to activities $a_i$. It could be stored together with the mechanism model as a parametric reliability expression, and be efficiently adapted with the current probability failure parameter $q_i$ of the activity at hand when a specific CONNECTor needs to be enhanced. This allows us to speed up the application of the selection procedure.

### 4.5.6 *Performance-related metrics case*

In the case of performance-related metrics, the choice of the model elements to be enhanced is expected to be made among those representing communications with longer time to complete. We recall that performance metrics we are dealing with are classical metrics, typically the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage, as defined in [5]. Therefore, with reference to our context, the dependability mechanisms should act to improve transmission time essentially in presence of possible congestion of the communication channels, responsible for longer times. Among the dependability mechanisms shown in Table 3, Parallel Retry and Probing are the only ones adequate to address performance improvements: by using redundant transmission channels, these mechanisms allow to exploit the most performable ones.

The approach to select the activity to improve is similar to the dependability-related metric case, but specular under the time domain. The steps are:

1. identification of those activities that are involved in the performance metric (unless we are dealing with an end-to-end metric, a subset of activities is typically involved);

2. for each activity $a_i$ in the involved set, re-determine the value of the transmission time parameter $\bar{t}_i$ resulting from applying the dependability mechanism;

3. since it is expected that the performance-related metric is based on the summation of the execution time of the activities under consideration, select the activity $a_j$ for which the difference between the original time parameter value $t_i$ and the new one $\bar{t}_j$, as calculated at step 2., is the largest.

Similarly to Table 4, the following Table 5 elaborates a simple example for the performance-related case, using the Probing and Parallel Retry (using three channels) as dependability mechanisms.

| activity | $t_i$ | $\bar{t}_i$(probing) | $\bar{t}_i$(parallel retry) | $t_i$-$\bar{t}_i$(probing) | $t_i$-$\bar{t}_i$(parallel retry) |
|---|---|---|---|---|---|
| $a_1$ | 4.664 | 4.530 | 1.589 | 0.134 | 3.075 |
| $a_2$ | 2.490 | 2.302 | 0.794 | 0.188 | 1.696 |
| $a_3$ | 1.747 | 1.497 | 0.529 | 0.250 | 1.218 |

Table 5.: Illustrative example of the selection approach for performance-related metrics

From the table, we observe that the Parallel Retry results in much better values than those of the Probing, at the cost of a higher redundancy. The choice of the tuning of the mechanisms parameters (such as the number of additional channels for the Parallel Retry) depends on the level of performance that needs to be guaranteed for the specific application at hand.

## 4.6 ADAPTIVE ASSESSMENT

After having introduced the pre-deployment analysis methods, we focus here on the Updater module in order to show how we deal with the adaptive assessment. Basically, the dynamicity and evolution of the targeted environment lead to potential sources of uncertainty, which undermine the accuracy of the off-line analysis. To cope with this issue, we investigate the adaptive dependability assessment, which exploits run-time monitoring to re-calibrate and enhance the dependability and performance prediction along time. At design time, stochastic model-based analysis is performed as a pre-deployment method to support the synthesis of a CONNECTOR. While the prediction so obtained plays an important role in guiding the building of the CONNECTOR, it might suffer from unacceptable

inaccuracy because of possibly limited knowledge at analysis time or successive context evolution. Through the run-time monitoring of proper events and by collecting them along several executions, we can identify changes that require to be accounted for by a new iteration of the model-based analysis [54].
Updater is the module of the DEPER architecture (Figure 17) in charge of interacting with the Monitor enabler to refine the accuracy of model parameters through on-line observations. Inaccuracy of the non-functional values used in the off-line analysis at design time is mainly due to two possible causes: *i)* limited knowledge of the NSs characteristics acquired by DEPER/Discovery enablers; *ii)* evolution along time of the NSs, as naturally accounted for in the CONNECT context. Updater receives inputs from both internally to DEPER (from the Evaluator module) and externally (from the Monitor enabler). For each CONNECTOR ready to be deployed, the Updater module receives from the Evaluator module the model parameters to convey to the Monitor enabler for run-time observations. The parameters received from Evaluator are obtained through a sensitivity analysis that aims to understand which elements of the CONNECTED system have highest impact on the dependability and performance measure. From the Monitor enabler, the Updater module receives a continuous flow of data of the parameters under monitoring relative to the different executions of the CONNECTOR. Accumulated data are processed through statistical inference techniques. If, for a given parameter, the statistical inference indicates a discrepancy between the on-line observed behaviour and the off-line estimated value used in the model, a new analysis is triggered by instructing the Builder module to update the CONNECTED system model. To improve on efficiency, the Updater module could receive indications not only on the parameters to be monitored, but also on a range of values for each of them, thus setting the variation interval within which the already performed analysis is subject to negligible modifications. Then, should the new values determined via inference techniques on on-line collected data be outside the reference range values, the consequence is that the synthesised CONNECTOR does not meet anymore the stated requirements and re-adjustments at synthesis level are necessary. Of course, the efficiency gained in avoiding repetitions of the analysis triggered by Updater has to be compared with the additional effort necessary at pre-deployment time to assess the ranges for the parameters values via sensitivity analysis. In the current prototype implementation of DEPER, range values have been not accounted for and left as a future extension of the enabler. The activity diagram of the updater phase, described above, is shown in Figure 23.

As reported in [137], methods of statistical inference, applied to a collection of elements under investigation (called population), allow to estimate the characteristics of the entire population. In our case, the collection of values relative to each parameter under monitoring constitutes a subset of the population (called sample) to which we apply such techniques. Parameter estimation is the process by which it is possible to get information, from the observed sample, in order

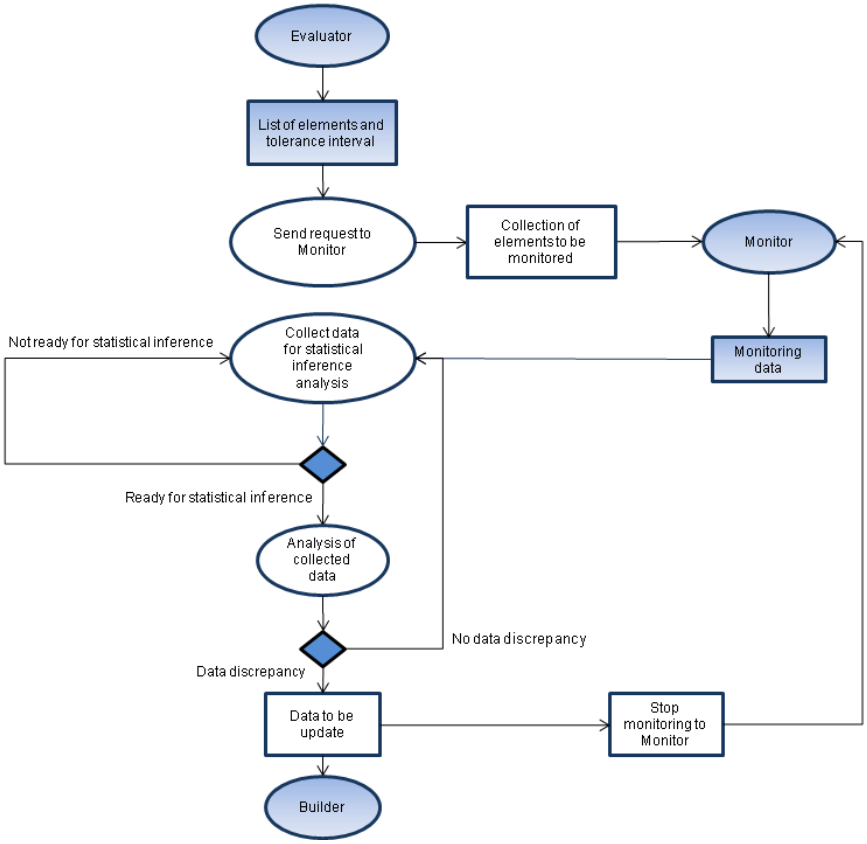


Figure 23.: The activity diagram of the updater phase

to assign a value (point estimate) to the parameter or a set of values (interval estimate). The sampling process represents a significant problem, because it is unknown which is the representative sample size ($n$). It seems intuitive that the precision of the estimates increases with $n$. On the other hand increasing $n$ could lead to excessive increase of time and costs. The methods of parameter estimation rarely produce a point estimate of the desired parameter which coincides with the actual value. Therefore, it is often preferred to find an interval estimate, called confidence interval $\Delta$, which contains the real value of the parameter under analysis with a confidence level $\alpha$. The size $n$ of the random sample affects the confidence interval, therefore it is possible to determine the value of $n$ based on the confidence interval:

$$n \geqslant \left\lceil \left( \frac{z_{\alpha/2} S^2(n)}{\Delta} \right)^2 \right\rceil$$

where the value of $z_{\alpha/2}$ is tabulated. When the sample size is relatively small ($n < 30$), we can use the Student $t$ distribution [137].
To evaluate the sample size $n$ we encounter two difficulties:

1. $S^2$ is not known in advance;
2. $t_{n-1;\alpha/2}$, which can be read from a table, depends on $n$.

These difficulties can be solved by the following two points:

1. using an assumed value of the variance, indicated by $S^{*2}$, from pilot investigations;
2. using an iterative algorithm, to evaluate $n$ using from time to time the degrees of freedom obtained at the previous step. The stop condition of the algorithm is reached when the result of two successive steps is the same.

The iterative algorithm proceeds as follows:

1. $n_0 = \infty$ (initialisation);
2. $n_1 = \left(\frac{2 \cdot t_{n_0;\alpha/2}}{\Delta}\right)^2 \cdot S^{*2}$;
3. $n_2 = \left(\frac{2 \cdot t_{n_1;\alpha/2}}{\Delta}\right)^2 \cdot S^{*2}$;
4. ...
5. until the last two results are the same.

Following this approach and considering fixed values of confidence interval and confidence level, we are able to define the sample size that the Monitor enabler has to send to the Dependability and Performance enabler in order to evaluate the monitored data.

## 4.7 GMES CASE STUDY

In this section, we present a demonstrative scenario based on the GMES (Global Monitoring for Environment and Security)[1] European Programme for the establishment of a European capacity for Earth Observation. The purpose is to show how the methods we propose can be applied to select proper dependability mechanisms and where to apply them at pre-deployment time and how to refine the model parameters with actual values gathered at run-time.
GMES services address six main thematic areas: Land Monitoring, Marine Environment Monitoring, Atmosphere Monitoring, Emergency Management, Security, and Climate Change. The GMES emergency management service provides all actors involved in the management of natural disasters. In particular, it covers different catastrophic circumstances: floods, forest fires, landslides, earthquakes and volcanic eruptions and humanitarian crises.

1 Currently known as *Copernicus* http://www.copernicus.eu/

### 4.7.1 *Forest-fire emergency*

In this work, the focus is on the forest fire emergency situation [47, 67]. The scenario describes the management of forest-fire, close to a border village and a factory between two different countries, Country A and Country B. Forest monitoring and forest fire management in the country A are under the responsibility of Country A Command and Control fire operations centre (C2-A).

The Forest-fire scenario addresses different phases. We focus on the *Reinforcement integration phase*, where Country B provides an Unmanned Aerial Vehicle (UAV) to Country A. The UAV is equipped with various video cameras that allow to get a better view of the fire front close to the village in order to be able to proceed to its evacuation in time. Country B grants access to its weather forecast service, in order to continuously provide information about temperature, humidity and wind of the area interested by the fire.

During the crisis phase, Country A's Command and Control fire operations centre (C2-A) is in charge of the management of the forest fire crisis. This phase involves a number of sensors and human actors. In case the fire goes wider and there is a direct threat to the village and the factory, Country A asks support to Country B (*reinforcement phase*). The reinforcement phase starts when the support resources, provided by Country B, are deployed and controlled by C2-A. Once deployed on the emergency area, resources from Country B are seen by C2-A as resources available and usable during the Fire-fighting phase. Resources provided by Country B have the functionalities similar to those of Country A's resources (e.g., UAVs provide high quality images or weather information of the area interested by the fire), but use different protocols.

Networked Systems involved in this scenario are: C2-A, the Weather Service, and the UAV with integrated video camera. For the illustrative purpose we consider only C2-A and the UAV, as this is sufficient to show the approach. We select the UAV instead of the Weather Service because it provides a scenario more interesting. The behaviour of the Networked Systems is described in the following subsections.

#### *Command and Control Centre of Country A*

The Command and Control Centre represents the first Networked System which needs to use specific resources. When C2-A wants to access the resources equipped with one (or more) video camera(s) and operating in the field, like an Unmanned Ground Vehicles (UGV), it first needs to authenticate with the resource sending a *getToken* message. After that, it receives back a message containing the *token* useful to access the resources. Once C2-A has a token, it can instruct the resource to move forward, backward, left and right, by sending the corresponding message, which is followed by an acknowledgement message confirming the movement. At the same time C2-A can select the camera installed

on the resource through the message *selectCamera* and then it can receive the video stream with a specified zoom level, by sending the message *getVideo*.

*UAV and integrated Video Camera of the Country B*

The user of the UAV first needs to authenticate with the service by sending a `getIdentifier` request. After that it receives back an identifier (`idResp`) and can trigger the flight phase through the `takeOff` message. After the take off, the user can order the UAV to move left, right, front, back, up or down, or to land. For each order of movement, the UAV replies with an acknowledgement message. Moreover, at anytime during the flight phase, the user can get, from the integrated video camera, the video stream showing in real time the specified area of interest; the user can also perform zoom-in and zoom-out commands.

CONNECT*ing C2-A and the UAV*

The application behaviour of the CONNECTed system is depicted by the sequence diagram in Figure 24. Given the heterogeneity between the Command and Control Center and the UAV involved in this GMES scenario, it is necessary to synthesise on-the-fly a CONNECTor to allow the communications and the exchange of information between the two Networked Systems. Therefore, a CONNECTor needs to be synthesised and analysed to assess whether it meets the dependability and performance requirements.

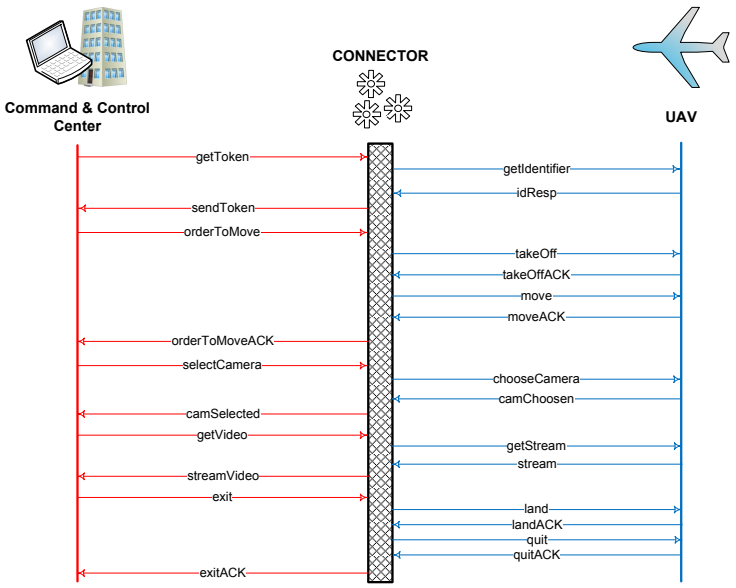


Figure 24.: Sequence diagram of the CONNECTed system

### 4.7.2 CONNECTOR's *model*

From the specification of the CONNECTed system, we generate a model (through the SAN formalism) suitable to assess dependability and performance related metrics (by means of the Möbius tool [53]). The Dependability and Performance unit, mentioned in previous sections, accomplishes this task by the automated process we implement.

SAN models for the two networked systems (that are, the Command and Control Center and the UAV) and of the synthesised CONNECTOR are generated according to the systems' specification. Figure 25 shows the model of the synthesised CONNECTOR useful to allow communication between C2-A, the UAV, and the Weather Service.

In order to simplify the reading, we provide a brief description of the SAN model of the connector. One token in the place `p1` (in the upper left side) means that the CONNECTOR is waiting for the first Networked System, that is C2-A, which can be interested to communicate with the Weather Service or the UAV. The upper part of the atomic model shows the CONNECTion between C2-A and the Weather Service, while the lower and larger part shows the CONNECTion between C2-A and the UAV. The activation of one of the two possible CONNECTion is regulated by the input gates `ig1` and `ig5`, connected to the timed activity `getWeather` and `getToken`, respectively. We focus on the description of the (C2-A - CONNECTOR- UAV) branch, that is the one we analyse in this work. The activity `getToken` becomes enabled to complete when the place `p1` contains at least one token and the condition expressed by the input gate `ig5` is true, that is the place `sCC3` contains at least one token (`sCC3->Mark()>0`). The place `sCC3` represents a shared place between the two atomic SANs: C2-A and CONNECTOR. Tokens are put in `sCC3` by the C2-A in order to activate the communication with the UAV, representing the *getToken* message. Once the activity `getToken` completes, the CONNECTOR sends a *getIdentifier* request to the UAV, represented by the activity `getIdentifier`, when this activity completes, it adds one token in the place `p8` and, through the output gate `og5`, one token in the place `sUAV1`, that is a shared place between the two atomic SANs: CONNECTOR and UAV. At this point the UAV processes the received request and responds with an *identifier*. The identifier is received back by the CONNECTOR, it is represented by the activity `idResp`, which is enabled by predicate expressed in the input gate `ig6`: at least one token in the share place `sUAV2`. The description of the model proceeds following the description of the behaviour of the CONNECTed system, shown in Figure 24. As general rule, we can observe that each activity has two cases the first from the top represents the correct operation, while the second one represents the failure of the activity, with a specified failure probability. The name of each shared place, is preceded by the suffix *s*, followed by the name of the Networked System that shares this place (e.g., UAV), finally an identification number is appended to the name to distinguish the various places.

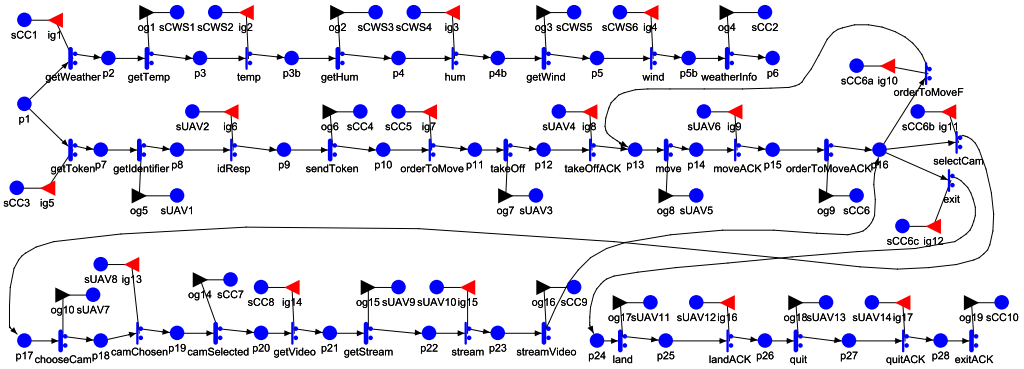


Figure 25.: SAN of the CONNECTOR

### 4.7.3 *Analysis*

In order to cover both performance and dependability aspects, the measures we assess in the evaluation are respectively *latency* and *coverage*. It is important to note that the measures we assess refer to different and independent analyses, which are detailed in the following.

*Pre-deployment Analysis*

*Latency* is evaluated as a performance indicator and is measured from when the Command and Control Center sends one of the possible orders to move (`orderToMove`) to when it receives an acknowledgement back (`orderToMoveACK`).

*Coverage* is defined as the percentage of stream video the Command and Control Center correctly receives from the UAV, with respect to the number of video requests made.

Given the initial parameters values characterising the timing aspects of the elements involved in the CONNECTed system model (namely, the time to communicate between the networked systems and the CONNECTOR), and the latency requirement stated beforehand, the analysis provides a positive feedback on the performance requirement, while it reveals that the CONNECTOR does not satisfy the dependability requirement.

Figure 26 shows the trend of coverage (on the $y$ axis) at increasing values of streams requests from C2-A (on the $x$ axis). Moreover, the coverage threshold is shown in the figure at value of 0.75, as specified in the requirement. The figure shows that the analysis results, obtained considering the basic model of the CONNECTed system, satisfy the requirement only when the number of requests is at most 13.

Table 3 needs to be examined to find the proper mapping between the mechanism and the characterising aspects when the coverage requirement is not met. Here, timing constraints are rather soft, while correctness is mainly relevant.

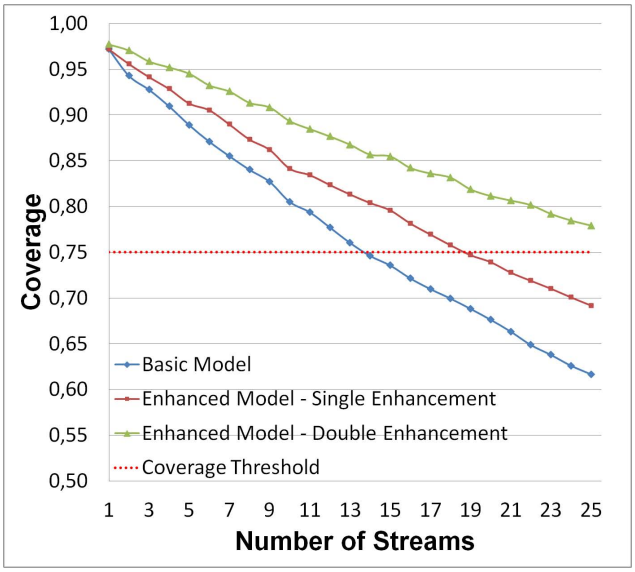


Figure 26.: Coverage assessment at varying of number of streams request

Therefore, the viable candidate mechanisms are Retry and Majority Voting. For the sake of saving in additional resources to employ, we select the Retry mechanism.

By applying the procedure (Section 4.5.4) that selects the model element(s) that needs to be enhanced, we find that `getStream` is the activity that leads to the greatest improvement with the Retry mechanism. This activity models two communication channels between the Connector and the UAV. The results in Figure 26 confirm that Retry applied to that activity provides an improvement on the measure of coverage with respect to the original model. The enhanced Connector meets the requirement up to a maximum of 18 requests.

In order to verify a further possible improvement, we perform analyses by applying the Retry mechanism on a further activity, `stream`, that is the second critical activity obtained through the selection procedure. It is possible to note that the use of this dependability mechanism on two different activities allows the Connector to fully meet the requirement of coverage, as Figure 26 shows with the double enhancement.

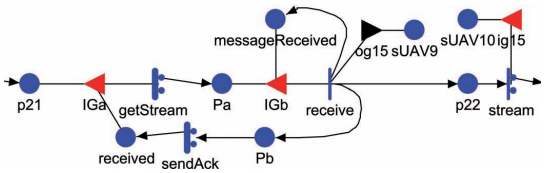


Figure 27.: Portion of the Connector model, replaced by the Retry mechanism

For illustrative purposes, Figure 27 shows the portion of the CONNECTOR model where the *getStream* activity has been replaced by the Retry Mechanism between the places *p21* and *p22* of Figure 25.

*Run-time Analysis*

Once dependability and performance analysis provides positive feedback about the CONNECTOR, it is deployed and begins to operate. At this stage the Updater module is in charge to update the model on the basis of the data related to real executions (Section 4.6) of the CONNECTed system as gathered through the monitor enabler, and a new analysis is performed at run-time.
Figure 28 shows analysis results on latency at varying the transmission rate of the communication channel between 0.1 and 1 (on the $x$ axis). The value of the transmission rate used in the model at pre-deployemnt time is 0.8 and the analysis results show that the latency requirement, set to the value 7 *time units*, is satisfied. Nevertheless, during the take off phase, latency is very close to the threshold, therefore the transmission rate is the parameter to be monitored at run-time. After collecting a sample of observations, we observe that the real parameter is 0.62, which is different with respect the one used a pre-deployment. The Updater module updates the model parameter and a new analysis on the refined model is performed highlighting that the latency requirement is no longer satisfied

In the case of latency requirement not satisfied, Probing and the Parallel Retry mechanisms are the candidate mechanisms to improve the CONNECTed system with respect to timing constraints. In the presence of limited availability of

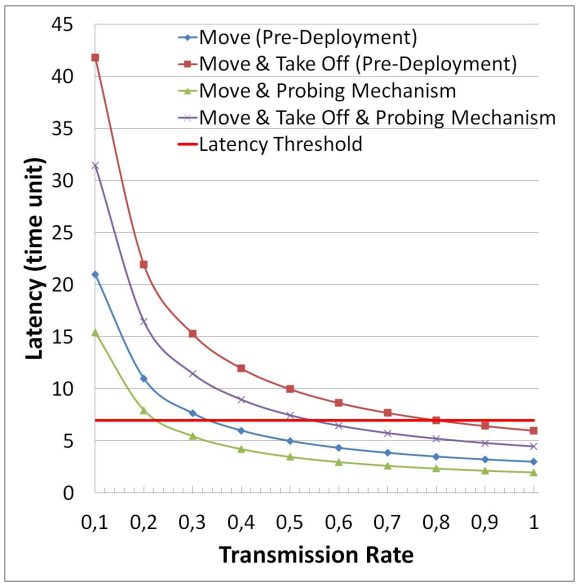


Figure 28.: Latency assessment at varying of the rate of the transmission

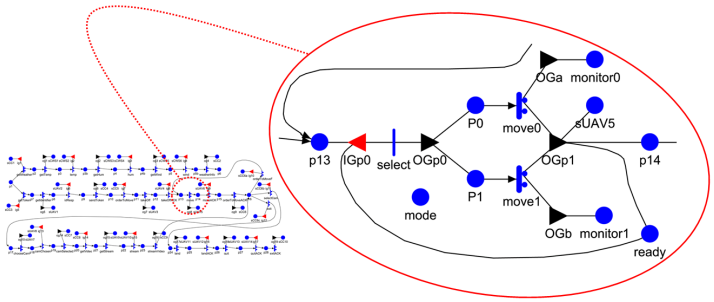


Figure 29.: SAN Connector model highlighting the portion where to integrate the Probing mechanism (represented inside the big ellipse)

channels in exclusive usage by the Connected system under analysis, preference goes to the Probing mechanism. Then, the second phase is to select the possible model element(s) to enhance with Probing. Following the steps delineated in Section 4.5.4, the element for which the mechanism brings the highest benefit in decreasing the latency metric is the `move` activity.

The results obtained when employing the mechanism are also shown in Figure 28. We note that, for values of transmission rate greater than 0.62, the Connected system is able to satisfy the requirement also during the take off phase. Figure 29 shows the SAN model of the Connector with the zoom where *Probing* needs to be employed in order to enhance the Connector.

## 4.8 CONCLUSIONS

This chapter tackled the challenge of dependability and performance analysis in dynamic and evolving systems, whose peculiarities make traditional methods largely inadequate. The idea to cope with the issues raised in the addressed context resorts to integrate pre-deployment stochastic model-based analysis with run-time monitoring, to achieve adaptive dependability assessment through re-calibration and enhancement of the dependability and performance prediction along time.

The aim of this two-phase analysis framework is twofold. On one side, the stochastic model-based analysis performed at pre-deployment time provides a primary support to the realization of the "rightest" system, given the partial knowledge about the involved subsystems and environment conditions. During this phase it is proposed an approach to automate the selection of a suitable dependability and performance mechanism for the enhancement of the system to be deployed. Specifically, the approach targets generic mechanisms based on the synthesis of mediating software bridges (Connectors) that allow interoperability among heterogeneous devices. In addition, we have investigated a generic method for identifying elements in the Connector that must be reinforced with

the selected dependability mechanism in order to improve performance– and dependability–related metrics.
However, the resulting unavoidable potential inaccuracy on the prediction so obtained could lead to more or less severe consequences if awareness about it is never acquired. This is the point where, run-time analysis provides a concrete contribution, by updating the model with concrete and real parameters collected during the operation of the system, thus allowing to enhance the previously performed analysis

This pre-deployment and run-time integrated framework is proposed as a general, automated approach to fulfill the dependability and performance assessment needs in dynamic and evolving contexts. Therefore, although not novel in its basic principles, the work done so far constitutes an important step towards the definition of an automated and adaptive process to provide dependability and performance analysis accounting for modern and future application needs.

However, additional effort is needed to improve the approach, for instance, in the automatic generation of the dependability and performance model from the specification of the system, techniques could be sought able to optimise the model on the basis of the specific metrics that needs to be assessment.

# 5

# A MODEL-BASED METHODOLOGY FOR INSIDER THREAT ASSESSMENT

Security is a major challenge for today's companies, especially ICT ones which manage large scale cyber-critical systems. Amongst the multitude of attacks and threats to which a system is potentially exposed, there are insider attackers i.e., users with legitimate access which abuse or misuse of their power, thus leading to unexpected security violations (e.g., acquire and disseminate sensitive information). These attacks are very difficult to detect and mitigate due to the nature of the attackers, which often are company's employees motivated by socio-economical reasons, and to the fact that attackers operate within their granted privileges. It is a consequence that insider attackers constitute an actual threat for ICT organizations. This part of the thesis aims at presenting a methodology for insider threats assessment and mitigation. We perform this work in the context of the Italian research Project SECURE! [2]. The project allows us to apply the methodology to the project case study, which includes different kinds of users and consequently is potentially exposed to a large set of insider threats.

## 5.1 THE NEED OF A METHODOLOGY FOR THE INSIDER THREAT ASSESSMENT

Today's ICT organizations are constantly facing the challenge of ensuring high degrees of security and privacy. Security measures are attentively selected and maintained, mainly with the intent of protecting the organization from external threats. Several tools and solutions are available for this purpose, for example firewalls. A lesser amount of solutions is instead available for mitigating threats coming from within the company, that is, from its own employees; these threats, that we refer to as insider threats, are most often mitigated almost exclusively through regulations and policies [65]. For example, insiders to an organization such as former, or newly fired employees or system administrators might abuse their privileges to conduct masquerading, data harvesting, or simply sabotage attacks. Although some intrusion detection systems offer insider threats capability, it is still very difficult to characterize all the threats, transform them into rules (or, in case of anomaly-based intrusion detection, instruct the detector to identify them as anomalies), and effectively detect intruders.
The problem of insider threats has been, and currently is, largely discussed in literature, because it is particularly challenging to identify insiders and mitigate the possible threats they pose on a system. In fact, it should be considered that an insider may have socio-economical roots, and the detection of false

positive in insider attacks may have severe consequences on an organization (e.g., due to the impact of false accusations of insider threats on both the individual and the organization [74]). Mitigation may be composed of prevention including deterrents as strict regulatory aspects, surveillance, legal implications, or detection methods and procedures that may help protecting the system. It appears evident that protecting from insider threats requires to study the socio-economical profiles of the users, the assets they use, their actions, and the impact of the actions on the assets, systems and organization. This calls for a tailored insider threats assessment activity, which takes into account socio-economical aspects while identifying the attacks, their impact on the system and organization, and possible countermeasures. We aim at tackling this problem by proposing a methodology for insider threats assessment and mitigation. The methodology presents the following features: *i)* it is tailored for the challenges posed by insider threats; *ii)* although it benefits of the support of attack libraries and tools for system and attack modelling, it does not impose restrictions on the characteristic of the libraries and tools to use; *iii)* it takes into account socio-economical aspects, including a description of the profile of the attacker; *iv)* relies on model-based formalism to represent the system and the attack paths and to analyze threats and evaluate countermeasures. The methodology first defines the system requirements and the attackers profiles, then identifies the threats, the attack paths and the potential countermeasures. The methodology also includes a maintenance phase during which reconfiguration facilities of the system are supported to update the assessment.

## 5.2 DEFINITION OF INSIDER THREAT

Talking of insider is often confusing. In literature there are different definitions of insiders, each of which, in general, has a negative meaning of the concept of insider. Although the question *"who is an insider?"* seems simple, a well established definition is still missing. Definition in [28], [29] suggests that an insider must be defined with respect to a set of rules that is part of a security policy:

> *"A trusted entity that is given the power to violate one or more rules in a given security policy... the insider threat occurs when a trusted entity abuses that power."*

The CERT Program goes further by providing a definition of malicious insider [127]:

> *"A malicious insider is defined as a current or former employee, contractor, or business partner who meets the following criteria: i) has or had authorized access to an organization's network, system, or data; ii) has intentionally exceeded or intentionally used that access in a manner that negatively*

*affected the confidentiality, integrity, or availability of the organization's information or information systems."*

In literature several uses of the term insider can be found [133]. In general, we can simply define an insider as an entity that has been given the privileges to act within a specific environment. What is of interest is the use of privileges (being it an abuse or misuse of privilege, or simply a mistake) in such a way that it constitutes an insider threat (being it malicious or accidental [74]).

## 5.3 STATE OF THE ART AND ADVANCEMENTS

A wide literature exists on the issue of insider threats, although most of it is devoted to the description of techniques and methods useful to prevent insider attacks and to protect the system or the infrastructure. Examples are initiatives such as to restrict remote access and system administrator access, or to inhibit the use of removable drives. Also works have been done to predict insiders activities, for example [18] presents a prediction model based on graph theory approaches, where alarms are raised when it is detected an increasing risk that users actions might lead to compromise systems resources. Recent years have witnessed an increasing number of works devoted to research solutions for (early) detection of insider attacks [74, 127]. However, to the best of our knowledge, most of the proposed works have been devised for specific and well defined case studies, sometimes also requiring to introduce simplifying assumptions, and without offering good portability to different systems or environments. In [121], the authors aim at detecting masquerade attacks, where a user impersonates another user, by profiling user behaviors. To evaluate these attacks, UNIX command data were collected from a certain number of users and then the data were contaminated with masqueraders. The experiment was performed and compared with six masquerade detection techniques: Bayes one-step Markov, Hybrid multi-step Markov, IPAM, Uniqueness, Sequence-Match and Compression. In [132] an approach is pursued to predict financial fraud from insiders by considering: *i)* the audit data, gathered during the operation of the system, and *ii)* the human factor, as a qualitative aspect to integrate with the classic quantitative analysis of financial transactions. The works [82] and [94] focus on the analysis of anomalous commands executed on data bases. In [82] a user profile has been generated according to a syntax-centric approach, that represents the structure of the SQL queries submitted by the users, in order to detect anomalous queries. Another approach which considers a relational database management system as case study is in [94], where the approach is data-centric unlike the previous one. In this approach the anomalies are identified looking at the data that are retrieved following the user's request. In [79] it is proposed a graph-based model of the basic network connectivity and access control mechanisms of the system, to identify system vulnerabilities which

can be exploited by a malicious insider. In [65], a study within an enterprise was conducted, where analysts were monitored while performing analytical operations. Using Grounded Theory research method (Grounded Theory is defined as the discovery of theory from data systematically obtained from social research), the objectives are: *i)* to understand how security event analysis works, and *ii)* to create an event-based model to analyze insider threats. This model shows how alerts and events created can be analyzed to determine if malicious insider behaviour is present. This approach defines a process and a model to identify and characterize insider threats. While great efforts have been devoted for the research of solutions to the thorny problem of the insider threats, few efforts have been made to identify and delineate a methodology for the insider threats assessment. Threats assessment processes, not specific for insider threats, exist; for example we mention the guide elaborated by NIST [7], as we believe it is a representative and comprehensive example of a threats assessment process. Although generic and largely applicable, it does not propose methodologies and libraries to support the assessment process, and most important it does not differentiate or specialize for insider threats. In fact, in our view the assessment process for insider threats requires to shift the focus from the architectural view of the system (being it a logical or physical view, depending on which one is available at the time threats assessment is performed) to a user-centric view, where socio-economical aspects, motivations and permitted actions of the users are at the very foundation of the analysis and are bounded to functional requirements. The attack goals and the vulnerabilities exploitable to achieve them come from the definition of the user and of his permitted actions described in the functional requirements, with little knowledge on the system architecture. Although compliant to the steps of the NIST process, our approach focuses on insider threats assessment, defining a methodology supported by libraries, techniques and tools. Summarizing, we propose to advance the state of the art by:

- showing a methodology for the insider threats assessment, valuable both during system design, and maintenance activities;
- presenting an example of threat libraries, specifically built for the adopted case study but easily reusable in different contexts;
- showing the possible usage of templates, libraries, and tools as well as formal or semi-formal languages to structure the methodology application and favour its reuse and update through the system lifecycle; and
- showing, through model-based approaches, how it is possible to provide a proper contribution to the insider threat assessment of systems.

## 5.4 THE METHODOLOGY

The objective we aim at is to provide an incisive, clear and supported methodology to handle systems, and related risk assessment processes, in which the insider threat issue could be relevant. Especially the methodology we propose is independent from the type of system. In the following sections we present the methodology, pointing out the description of six key iterative phases composing the methodology itself [103, 104].

### 5.4.1 *System Under Analysis*

The first step of the methodology relates to the description of the system under analysis. A system is characterized by a number of resources (e.g., services, computers, removable drives, etc.), one or more communication networks, and users, which can use the system or in general interact with it. In addition, new features of the system can be integrated over time, due to the evolution of technologies, and the update of system specifications. When characterizing a system, we can identify macro-components which constitute the overall architecture. A system can thus be seen as a set of $n$ macro-components. A description of the possible actions of the users in the system should be provided and used here, being it a description of requirements in natural language or supported by semi-formal or formal languages. Although not mandatory, a semi-formal or formal description is very important to guide the analysis and also to support possible future updates of the analysis in case of system modifications. Thus, given that precise requirements and a model of the system is preferable (especially as input for the successive phases of the methodology), the methodology does not provide restrictions on the description method and level of detail. For example, we are aware that in practise a threats assessment may incur while the system requirements definition is still ongoing and thus only description in natural language is available (note that this may call for additional iterations of the procedure once a semi-formal description is available). Anyway, providing a formal description of the overall system, in particular if complex, may be somewhat expensive in terms of time. For this reason we consider convenient and sufficient to have a semi-formal description limited to the aspects of interest of the system (e.g., services used) and the interactions that potential insiders may have with it. Through a semi-formal notation, it is possible to immediately understand the description of the system using graphical notations along with natural language descriptions, thus reducing ambiguous interpretations. There exists several notations that can be adequate: Data-Flow Diagrams, Finite State Machines, Activity Diagrams, etc. In Section 5.5 we propose the usage of Use Case Diagrams, which allow the description of the system functionalities and use case scenarios, from the point of view of the users/insiders.

In the case study (Section 5.5), we propose templates to help the system characterization, where use case diagrams show the interaction between the offered functionalities and the actors involved. Through the utilisation of tables, containing the relevant information, we provide the description of each diagram.

### 5.4.2 *Insiders*

The second phase of the methodology consists in profiling potential insiders, determining their dangerousness to the system and their key attributes. We structure this phase in two steps. In the first step, all possible users involved in the system under analysis are identified. In the second step their potential attributes as insiders are defined. To support this task, we define a library of insiders. We refer to the attributes presented in [34], where a detailed Threat Agent Library (TAL) has been provided, which constitutes a consistent reference library describing the human agents involved in IT systems and that could pose threats to such kind of systems, although not limited to insider threats. The idea is not to represent specific individuals, but instead the library is intended to create a taxonomy of specific attributes useful to identify a category of users/insiders. Specifically, according to the TAL [34], we consider eight attributes and we describe them in the following, with a specialisation for insider threats:

INTENT. Whether the insider intends to cause harm. Insiders fall into two categories based on their intent: *Hostile* and *Non-Hostile*.

ACCESS. Defines the extent of the insider's access to the company's assets. There are two options: *Internal* or *External*.

OUTCOME. Defines the insider's primary goal, that is, what the insider tries to accomplish with a typical attack. Possible outcomes are: *i) Acquisition/Theft* of essential assets or sensitive data; *ii) Business Advantage* to acquire business processes or assets; *iii) Damage* to injury physical or digital assets, or intellectual and industrial property; *iv) Embarrassment* to generate loss of credibility, influence, and competitiveness; *v) Technical Advantage* to acquire production processes or assets rather than a business process.

LIMITS. Legal and ethical limits that may constrain the insider, e.g., the extent to which the insider may be prepared to break the law. Options are: *i) Code of Conduct*, when the insiders follow both the applicable laws and an additional code of conduct accepted within a profession or an exchange of goods or services; *ii) Legal*, where the insiders act within the limits of applicable laws; *iii) Extra-legal minor*, when the insiders may break the law in relatively minor, non-violent ways, such as minor vandalism or trespass, e.g., activist; *iv) Extra-legal major*, when insiders take no account of the law

and may engage in felonious behaviour resulting in significant financial impact or extreme violence, e.g., members of organized crime.

RESOURCE. Defines the organizational level at which an insider typically works, which in turn determines the resources available to that insider for use in an attack. Options are: *i) Individual*: insider acts independently; *ii) Club*: members interact on a social and volunteer basis; *iii) Contest*: participants interact together for a very short period of time and perhaps in anonymous way with the objective to achieve a single goal; *iv) Team*: a well organized group with a leader, typically motivated by a specific goal and organized around that goal; *v) Organization*: a larger and better resourced than a Team; *vi) Government*: controls public assets and functions within a jurisdiction, it is very resourced and persists long term.

SKILL LEVEL. The special training or expertise an insider typically possesses. Options are: *i) None*, when the attacker has no expertise in the methods useful to attack the system, but has the ability to perform random acts of disruption or destruction; *ii) Minimal*, when the insider has the ability to copy or use existing attack methods; *iii) Operational*, if the insider knows the system and the underlying technology, and he is able to create new attacks within the system domain; *iv) Adept*, the insider is an expert in technologies and attack methods and can apply existing attacks or create new ones getting excellent results.

OBJECTIVE. The action that the insider intends to take in order to achieve a desired outcome. Options are: *i) Copy* (make a replica of the asset); *ii) Destroy* (destroy the resource in order to make it useless); *iii) Injure* (compromise the asset in order to limit its functionality or value); *iv) Take possession* (get exclusive possession of the asset); *v) Don't care* (the insider does not have a clear plan and/or may decide only at the time of the attack).

VISIBILITY. The extent to which the insider intends to conceal or reveal his identity. Options are: *i) Overt*, when the insider intentionally carries out an attack to the system and his identity is known at the time of the attack; *ii) Covert*, when the attack is revealed at the time of occurrence or soon after, but the identity of the attacker remain unknown; *iii) Clandestine*, when the insider aims at keeping secret his identity and the attack; *iv) Don't care*, if the insider does not place importance on secrecy.

### 5.4.3 *Insider Threats*

The third phase is the identification and description of possible threats to which the system could be vulnerable. This activity is of critical relevance because it

allows the identification of the damage that can be done on the system and the potential consequences. Through this phase it is possible to define the threats that need to be addressed with higher priority. Goals and threats may also be ranked on the basis of the level of dangerousness for the system assets, as often performed in risk analysis (see for example [7]). It is important to note that for the mere identification of the threats we are not interested on the motivations that lead an insider to put into practice an attack.
The potential threats, which a generic system may be subject, are typically: installation of improper software/apps (e.g., viruses, Trojans, spyware, backdoors, key loggers, logic bombs, etc.); improper data operations (e.g., deleting, exporting, producing fake data, etc.); managing of user profiles (e.g., creation of fake users). The number of threats to be analyzed can be considerable, especially if the target system is complex. Usually a good compromise is to identify a subset of potential threats with regard to specific assets.
The approach we propose consists in identifying all the possible threats of interest and associate them to the insiders identified in the previous step of the methodology. As a result we obtain a table that lists the threats and the corresponding matching with the insiders e.g., "YES" if the insider may realize the threat in a legitimate way, "NO" otherwise. Further details will be given in Section 5.5, which describe the application of the methodology to a case study.

### 5.4.4 *Attack paths*

This phase has the objective to identify the path(s) exploitable by the insider to realize the threat(s) and achieve the goal(s).
Several techniques exist and can result very useful for determining which threats exist in a system and how to deal with them, e.g., attack trees [120], attack graphs [125], privilege graphs [51], or adversary views [88]. The latter extends the concept of attack graph by considering different attack goals, attack preferences of specific attackers, and creating customizable models to produce quantitative analyses based on specific metrics of interest.
Regardless of the adopted approach, the capability to build an attack path is of paramount importance, as it allows for example to get information on the probability of occurrence of an attack, the success rate, the weaknesses in the system, etc. There exist modelling approaches for quantitative evaluation of attacks and attack paths [88], [96]. Such approaches are useful to quantify the probability of successful attacks, given the "success probability" of single steps, exploitable to realize the goal. For example, to perform quantitative evaluation, the ADVISE (ADversary VIew Security Evaluation) modelling formalism [88] could be used. ADVISE is a formalism for security evaluation that extends attack graphs, by including the concepts of time, costs, and success probability of the attack steps, and takes into account the attack behaviour and proficiencies of dif-

ferent attack profiles. However quantitative security analysis may be unfeasible for insider threats because the steps of the attack paths are "authorized steps", thus "doable", and consequently it is difficult if not meaningless to define a success probability of a path.
The identification of the overall set of attack paths is a critical step, especially if we think of unknown paths. Attackers may be able to explore unexpected attack paths, that are unknown w.r.t. the set of attack paths considered during the threats assessment [97]. Threat analysis is limited by the ability to describe the use cases and the system paths. System evolutions may bring the emergence of new paths (thus iterations of the threats analysis can be performed to identify them), and the activity of discovering unknown paths is strongly linked to an analysis of the system requirements and functionalities.

### 5.4.5 *Countermeasures selection*

This phase consists in the selection of the proper countermeasure(s), in order to avoid or mitigate the identified threat(s), by preventing the execution of the attack paths, or detect or discourage its execution. The countermeasures identification and selection can be supported by *i)* libraries which list the countermeasures for determined attacks, and *ii)* techniques from the state of the art. In general we propose to classify countermeasures in *preventive*, *deterrent*, and *detection*. Preventive countermeasures prevent the execution of an attack path (e.g., making a user unable to perform specific operations without supervision). Deterrent countermeasures do not allow to block an attack, but aim at discouraging attackers (e.g., forensic data logging for a-posteriori analysis [48]). Detection countermeasures aim at timely identifying the attack (e.g., the early detection solutions in [74, 127]).
Introduction of such countermeasures may require to re-assess the system, improved with the countermeasures, to verify that each threat is mitigated and give evidence of the security improvement. In case a model of the system and of the countermeasure is available, these can be integrated with the attack paths.

### 5.4.6 *Iteration and Update*

It is well known that current large scale systems and infrastructures are subject to changes and evolutions; this can require to iterate the phases of the methodology in order to align the assessment outcomes to the most recent status of the system. This require to: *i)* collect feedbacks about system status, *ii)* use those feedback to understand the evolution of the system, its users, and possible changes of system requirements, and *iii)* update the insider threats library and the attack paths. This will ultimately lead to define new countermeasures to be applied.
This is especially relevant in systems characterized by evolutionary and dynamic

behavior, as for example Internet of Services, large-scale architectures, or cyber-critical infrastructures, in which traditional, pre-deployment verification and validation is inefficient and thus continuous verification steps [36, 37, 115] to cope with the changes in the system and its environment are required.

## 5.5 APPLICATION OF THE METHODOLOGY

In this section we show an application of the methodology to a compact, but concrete case study, which is based on the utilisation of the Secure! framework designed in the context of the Secure! project [2]. We point out that we do not consider the sixth phase, because the Secure! framework is still in design phase, thus the application of the methodology will cover only the phases 1 to 5.

### 5.5.1 *System Under Analysis*

Secure! [122] is a service-oriented system whose objectives are: *i)* prevent crisis as terrorist acts, vandalism, sabotage, and *ii)* support crisis management (e.g., in case of environmental catastrophes, human sabotages, and authorized or not-authorized demonstrations). The Secure! system is able to integrate several heterogeneous information (audio, video, images, text) originated from different sources, including social networks (e.g., Facebook, Twitter, Flickr), emergency numbers, surveillance camera, telecommunication network, and data provided by the users on field through a Secure! application for mobile devices (anyone can be a user, after downloading the application). The Secure! system includes instruments to semi-automatically correlate, query and analyze the data, supporting both the intervention of crisis management teams, and a-posteriori forensic analysis.
Secure! is subject to severe security and privacy requirements. In fact, in order to have users trust the Secure! system and to make it acceptable to the community, it is mandatory to provide guarantees that users authorized to work with the data collected do not compromise, counterfeit, steal or even unnecessarily query them, or do not abuse of the data correlation and data search capacity behind what is strictly necessary for their work.
In the next section we present the Use case diagrams for the Secure! insiders.

### 5.5.2 *Insiders*

According to the system's specification provided by the project [122], we propose in the following a taxonomy of users physically or logically involved within the system organized in six plausible groups of users, investigating their roles as potential insiders.

OPERATOR. A Secure! user who works in the security field, e.g., trained firefighter, or rescue personnel. He is a potential insider because he can access the system to acquire stored information, or can transmit false information.

HUMAN SENSOR. Any citizen voluntarily registered to the Secure! platform in order to cooperate using the Secure! application. He is a potential insider because he could send false information on accidents.

DOMAIN EXPERT. Represents an expert in the homeland security field, e.g., a police officer or a component of national intelligence. He is a potential insider because he can access private data stored in Secure!, and manipulate the data.

UNKNOWN USER. A citizen who is not registered to the Secure! platform. He may unwittingly interact with the system, e.g., by posting relevant information on social media, blog, etc. This user is in general very little expert of Secure!, and he is not acknowledged as a potential insider.

SYSTEM EXPERT (SE). A special user which can access the system in order to use, install and configure the various services of Secure!. He is a potential insider because he has access to the internals of the Secure! system.

SYSTEM ADMINISTRATOR (SA). A special user which has remote and local access to Secure! in order to perform maintenance tasks, removal, exports and drop of data, managing user profiles, etc. Moreover, SA is in charge of managing emergency situations in which the system is undergoing maintenance, or under cyber-attacks, etc. He is a potential insider because he has wide access to the internals of the Secure! system.

Given the dimension of the Secure! system [123] and the amount of different interactions that groups of users can have with it, in the following we explore only the System Administrator (SA) and the System Expert (SE) groups of users. Figures 30 and 31 show the use case diagrams of the system related respectively to SA and SE. These diagrams represent the fundamental operations that SA and SE may have with the system.

Table 6 describes the Use Case Diagram of the SA category, where for each use case, we report: the actor(s) involved, the pre-conditions and post-conditions, and the general description of the use case.

In Table 7 we provide a detailed description of the Use Case Diagram of the SE category shown in Figure 31.

According to the eight attributes defined in Section 5.4.2, in this phase we report in Table 8 a preliminary matching between Attributes and Values, which will be refined during the next *Insider threats* phase. In fact, the values assigned to the attributes could vary based on the threat to be considered. Table 8 refers to attributes and values identified for SA. The corresponding table of SE differs

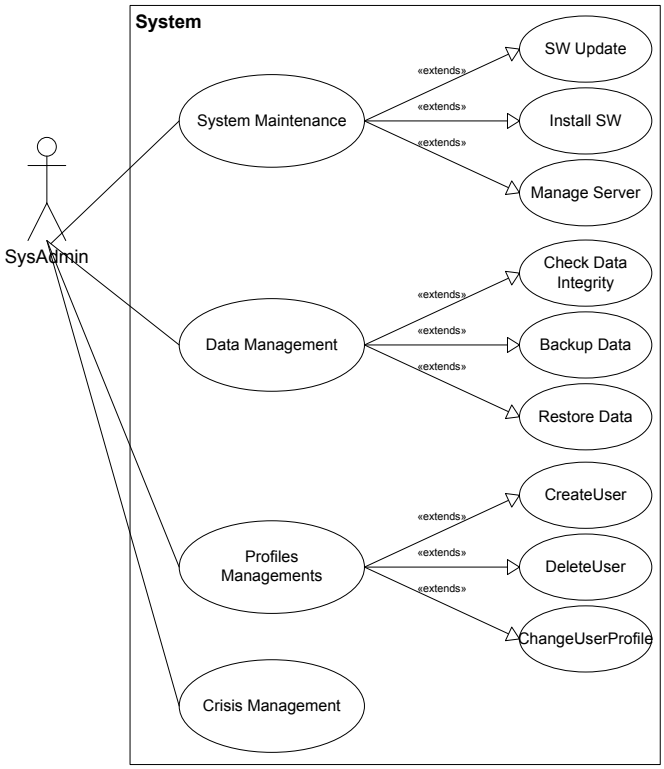


Figure 30.: UML Use Case Diagram of System Administrator

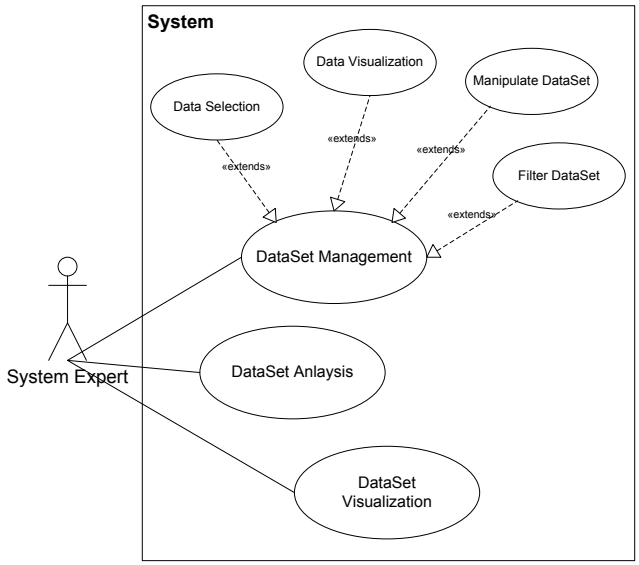


Figure 31.: UML Use Case Diagram of System Expert

only in the values assigned to the Minimum Skills attribute (in this case that value for SE is set to Operational), thus it is not shown.

| System Maintenance Use Case | |
|---|---|
| *Actor/s:* | SA. |
| *Pre-condition:* | SA must be authenticated. |
| *Post-condition:* | SA has full access to the system. |
| *Description:* | Apply OS patches and upgrades on a regular basis the system, the administrative tools and utilities. Configure/add new services as necessary. Upgrade and configure system software or Asset Management applications. Maintain operational, configuration, or other procedures. Perform periodic performance reporting. Perform ongoing performance tuning, hardware upgrades, and resource optimization. |
| **Data Management Use Case** | |
| *Actor/s:* | SA. |
| *Pre-condition:* | SA must be authenticated. |
| *Post-condition:* | SA has full access to the data. |
| *Description:* | Perform daily backup operations, ensuring the integrity and availability of data. |
| **Profile Management Use Case** | |
| *Actor/s:* | SA. |
| *Pre-condition:* | SA must be authenticated. |
| *Post-condition:* | SA has full access to the system data. |
| *Description:* | Create, change, and delete user accounts. |
| **Crisis Management Use Case** | |
| *Actor/s:* | SA. |
| *Pre-condition:* | SA must be authenticated. |
| *Post-condition:* | SA has full access to the system data. |
| *Description:* | Repair and recover from hardware or software failures or from cyber attacks. Coordinate and communicate any recovery actions. |

Table 6.: Description of UML Use Case Diagram - System Administrator

| Dataset management Use Case | |
|---|---|
| *Actor/s:* | SE. |
| *Pre-condition:* | SE must be authenticated.<br>Availability of a system dataset. |
| *Post-condition:* | Availability of the list of events analyzed (aggregates and related). |
| *Description:* | This use case allows SE to access the data collected by the system, selecting them according to criteria established by the same (data selection), and displaying the attributes, e.g., references to time and / or geographical (data visualization). The data set can also be filtered (filter dataset) for the removal of data with low informative content. It may also be necessary for SE to carry out a data integration (manipulate datasets) from the data sources. |

Table 7.: Description of UML Use Case Diagram - System Expert

### 5.5.3 *Insider threats*

In the context of the Secure! project, we can identify a number of threats of different type of severity, which are related to the actions performed by the insiders. This phase of the methodology aims at identifying all threats

| Attribute | Value |
|---|---|
| Intent | Hostile/Non Hostile |
| Access | Internal/External |
| Outcome/Goal | Acquisition/Theft, Embarrassment, Damage |
| Limits | Code of Conduct, Legal, Extra-legal |
| Resources | Individual, Organization, Team |
| Minimum Skills | Operational, Adept |
| Objective | Copy, Destroy, Take, Injure |
| Visibility | Covert, Clandestine |

Table 8.: Preliminary matching attributes-values

whose impact, in case of successful attack, is intolerable in terms of economic losses, image, damages to the infrastructure and/or to the environment, etc. In Secure!, for example, human sensors could use their own Secure! apps in order to intentionally provide fake information to the system, causing delay on supporting operations on field. System Experts and System Administrators could install and improperly configure components on the Secure! system. SA and SE can potentially realize a consistent number of threats with respect to the other insiders. Specifically, thanks to their privileges, they could install malicious software/code, create backdoors, disable system logs and anti-virus, create new users or change users privileges, install remote network tools, plant malware as logic bombs [49], perform operation on data base in order to have Secure! creating erroneous reports, modify, delete, and copy data.
The approach is to list all the possible threats of interest and try to associate them to the previously identified insiders, as we report in Figure 32, which shows the potential threats achievable individually by SA and SE, indicated in the rows of the table. We use the numbers in the second row to identify the threats; such values are used in the logical formulae in Section 5.5.4, in order to identify the attack paths followed to achieve the attack goals. The threats reported in Figure 32 are: *i)* disabling of system logs; *ii)* corruption of data; *iii)* visualization of confidential data; *iv)* addition of new services e.g., to collect information; *v)* setting improper systems components configurations (e.g., server, firewall, antivirus, intrusion detection, etc.); *vi)* abusing of system management privileges; *vii)* modification of privileges to target users, *viii)* installation of unsecured and vulnerable software; *ix)* installation of unsecured and vulnerable services; *x)* installation and utilisation of defective hardware; *xi)* transferring of confidential data; *xii)* disclosing of access keys of users (this is typically mitigated by systems setting, at least in default configurations); *xiii)* addition of Trojan horses in the system; *xiv)* disabling of components as for example STC (Secure-Two-party Computation) and PEP (Policy Enforcement Point) [48]; *xv)* modify log files and audit trails.

| Attribute | Value - SA | Value - SE |
|---|---|---|
| Intent | Hostile | Hostile |
| Access | Internal/External | Internal |
| Outcome/Goal | Damage, Acquisition/Theft | Damage, Acquisition/Theft |
| Limits | Code of Conduct, Legal, Extra-legal | Legal |
| Resources | Individual | Individual, Organization |
| Minimum Skills | Adept | Operational |
| Objective | Copy, Destroy, Take | Injure, Copy, Destroy |
| Visibility | Clandestine | Covert, Clandestine |

Table 9.: Final matching attributes-values for SA and SE

| | Threats | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **Insider** | Disable system logs | Corrupt data | View confidential data | Add not required services | Improper configuration | Improper user management | Elevate users privileges | Install vulnerable supporting SW | Install vulnerable Secure! services | Use of defective HW | Transfer confidential files | Access to crypto keys | Putting Trojan horses | Disabling protection of components | Altering audit trails and logs |
| **SA** | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES | YES |
| **SE** | NO | YES | NO | NO | NO | NO | NO | NO | YES | NO | YES | NO | YES | YES | NO |

Figure 32.: Mapping of Insiders to Threats

### 5.5.4 *Attack paths*

To achieve his attack goals, reported in Table 9, that are damage (e.g., performance degradation of the system or the delay/sabotage of the Secure! operations as for example emergency management operations), and acquisition/theft (e.g., theft of sensitive data), the SA and the SE perform the steps described in the following.

*Performance degradation*

SA may be willing to achieve the degradation of system performance. To achieve this goal, SA may succeed in realizing one or more threats among those listed in Figure 32, in conjunction or separately, thus leading to the identification of different exploitable attack paths. The logical formula 5.1 states the threats that directly make him succeed in the goal.

$$[4 \vee 10 \vee [(8 \vee 9 \vee (13 \wedge 14) \wedge 5]] \vee [1 \wedge [4 \vee 10 \vee [(8 \vee 9 \vee (13 \wedge 14) \wedge 5]]] \quad (5.1)$$

In order to assemble defective hardware (threat 10 in Figure 32), SA needs: *i)* physical access to the system and *ii)* to assemble the defective part, consequently causing a worsening of system performance.
To realize the other threats listed in the formula 5.1, SA needs to perform various actions, that constitute his attack paths. First of all, SA has to log into the system, from local or from remote (the login is obviously not considered a threat). Then, SA can add not required services (threat 4), or can perform improper configuration and system management (threat 5), as installing vulnerable supporting software (threat 8), installing vulnerable Secure! services (threat 9), and putting

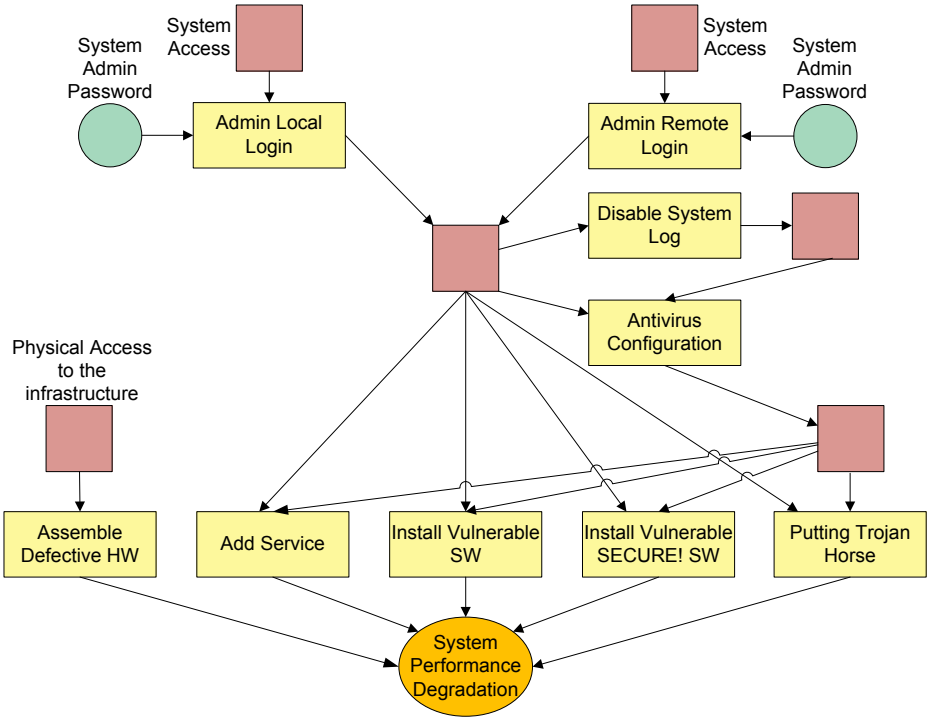


Figure 33.: ADVISE attack execution graph for performance degradation

Trojan horses (threat 13). Before realizing these threats, SA may also disable system log (threat 1 in Figure 32), to hide the attack. At the same time, to infect the system with a Trojan horse (threat 13), after authentication SA may have to re-configure the antivirus or protection software (threat 14) e.g., by adding some exceptions, and again he can optionally disable the system log to leave no trace of the configuration.
Figure 33 shows the ADVISE attack execution graph representing the above-mentioned paths exploitable by SA to realize his attack goal. To allow the reader a clear understanding of the ADVISE diagram, we briefly describe the meaning of the graphical notation: the rectangular boxes represent the attack steps the attacker has to perform in order to achieve the attack goal; the squares are the access domain; the circles are the knowledge items and finally the ovals represent the attack goal. More details on the formalism and graphical notation can be found in the logical formula [88].
As regard to SE, the logical expression of reference is shown in 5.2, where we can notice that the identified threats are a subset compared to those of SA.

$$9 \vee 13 \tag{5.2}$$

The differences are mainly due to the access modes to the system; in fact SE can access the system solely through the Secure! Graphical User Interface (GUI), with limited privileges, and without the right to change the system configuration.

We do not report the corresponding attack graph but it can be easily derived from Figure 33.

*Theft of sensitive data*

Another relevant goal that SA may be interested to achieve is the theft of sensitive data. To accomplish this goal, SA may succeed in realizing one or more threats among those listed in Figure 32. The logical formula 5.3 states the threats that directly make him succeed in the goal:

$$3 \vee 11 \vee (5 \wedge 13) \tag{5.3}$$

The ADVISE diagram representing the attack paths to reach the goal is shown in Figure 34, where we can notice that the first attack step is the login into the system, either locally, or remotely, or via the Secure! GUI. SA may directly access the data base and execute queries, or he may execute reporting tools through the Secure! GUI, in order to embezzle sensitive data. Moreover, to keep his actions hidden, SA may accomplish some intermediate steps e.g., disable system logs, configure the antivirus, etc.
Regarding SE, the corresponding logical expression is shown in 5.4, where the considerations of the previous attack goal with respect to the exploitable paths are still valid.

$$9 \vee 11 \vee 13 \tag{5.4}$$

*Delay of the Secure! rescue operations*

Another means to achieve delay/sabotage is by delaying the Secure! operations for crisis management. This approach is very relevant to the Secure! functionalities, that are *i)* supporting rescue teams, and *ii)* managing crisis. The logical expression that represents the correlation between the identified threats that allow achieving the goal is shown in the logical formula 5.5.

$$2 \vee 4 \vee 8 \vee 9 \vee 10 \vee 14 \vee (5 \wedge 13) \tag{5.5}$$

Figure 35 shows the ADVISE execution graph of the attacks, where the individual attack steps are already described in the previous sections.
The logical formula 5.6 relates to SE, where once again we observe that the considered threats are a subset of the SA case.

$$2 \vee 9 \vee 13 \vee 14 \tag{5.6}$$

The ADVISE diagram in this case, as in the previous one, should be slightly simpler because of the limited privileges owned by the System Expert.

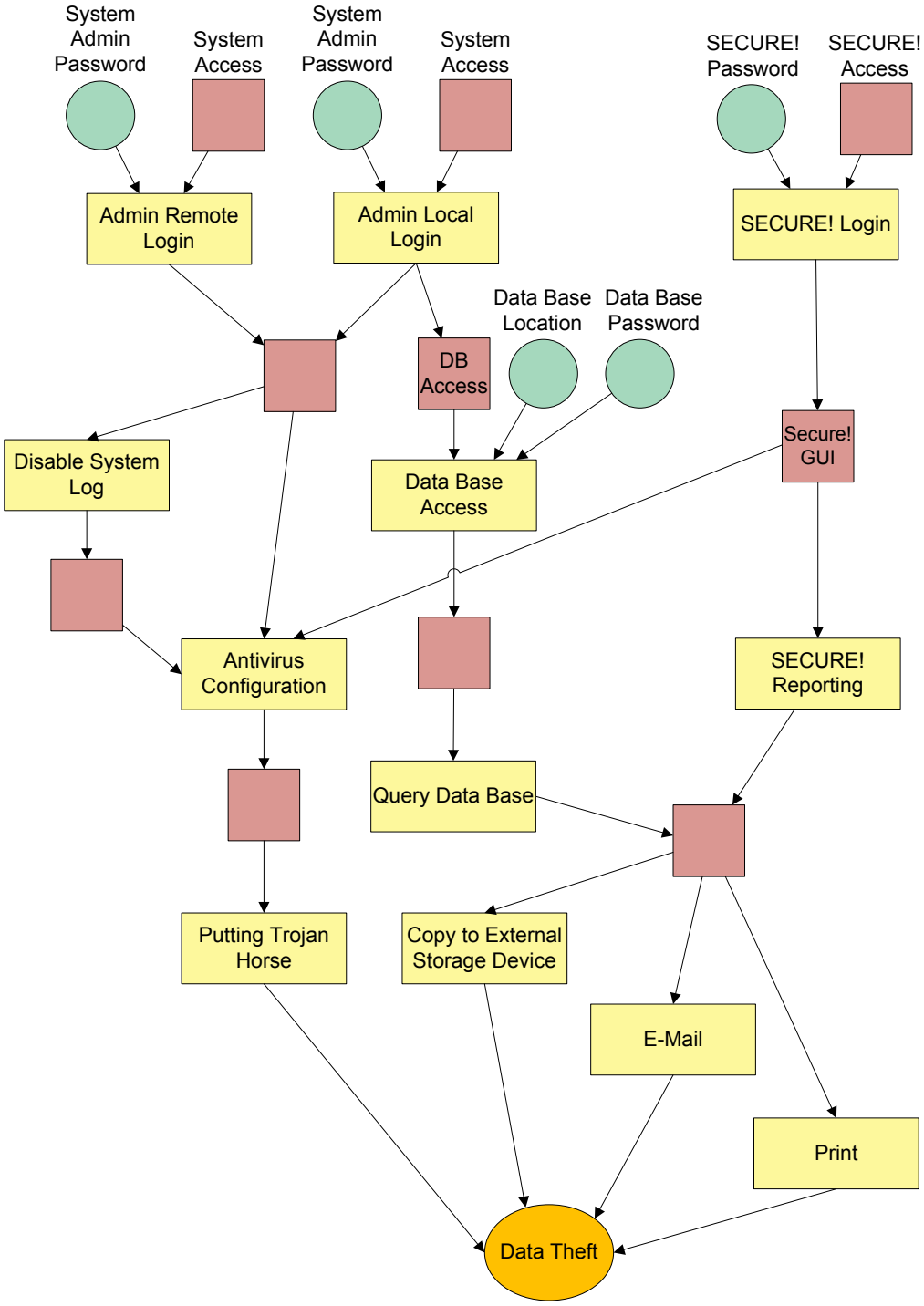


Figure 34.: ADVISE attack execution graph for Data Theft

### 5.5.5 *Countermeasures selection*

The individual actions which constitute the graphs in Figure 33, Figure 34, and Figure 35 are legitimate. Countermeasures, being them oriented to prevention,

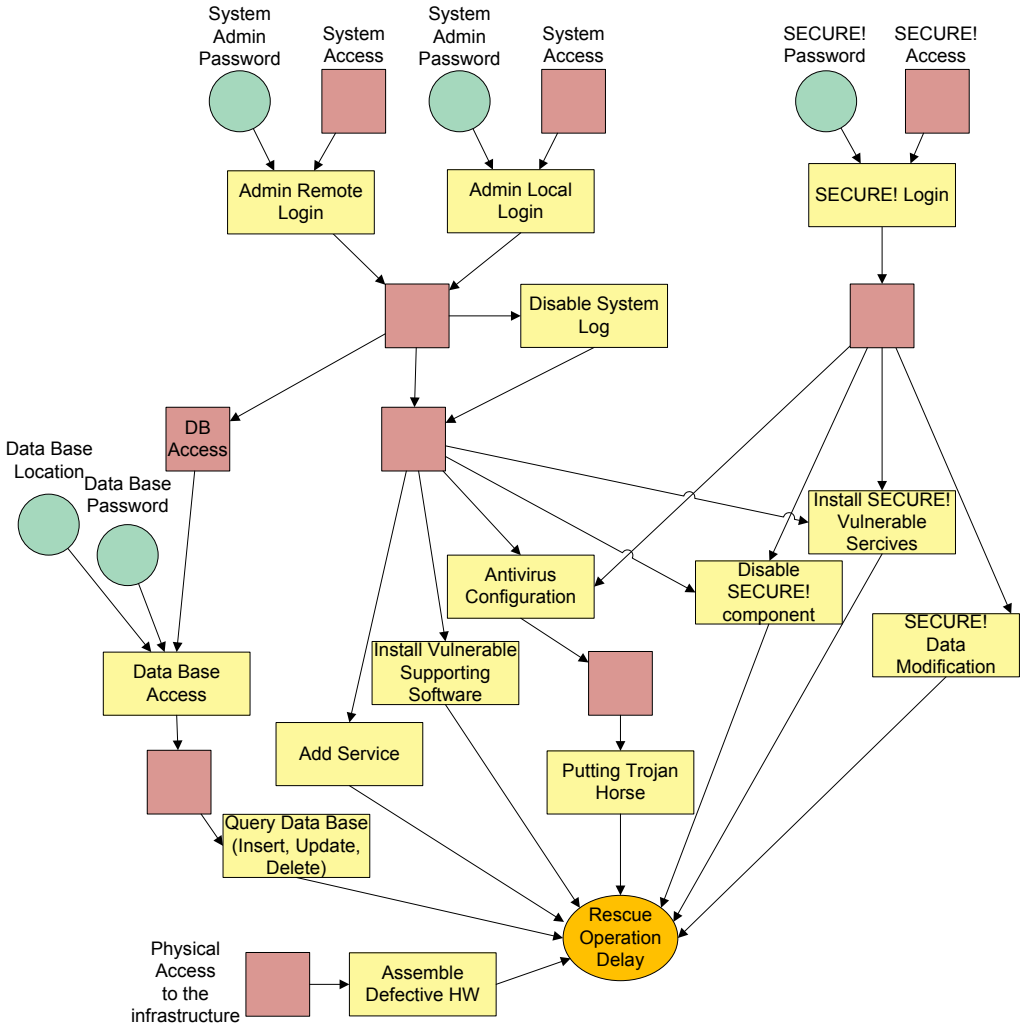


Figure 35.: ADVISE attack execution graph for Delay of Secure! rescue operations

deterrence, or detection [64, 106], must be imposed. This subsection provides the countermeasures we propose to the Secure! project in order to consider their implementation within the Secure! framework. Considering the attack goals previously described, possible countermeasures valid for both SA and SE, unless otherwise specified, are listed in the following.

**Preventive countermeasures**

Prevention aims at avoid attacks, usually are based on strict security policies to be applied on the system infrastructure and in the work environment.

- Keep the technical room locked, allowing access only in presence of staff of the same grade or higher, and under authorization from a higher-ranking person (SA only).

- Avoid to log into the system during holiday days or outside the office hours, except under authorization from a higher-ranking person.

**Deterrent countermeasures**
Deterrence averts the initiation and continuation of an attack attempt by increasing the necessary effort for an attack to succeed, increasing the risk associated with the attack, and /or devaluing the perceived gain that would come with success.

- Install monitoring cameras (SA only).
- Allow forensic log of users access [13], keeping track of the username, date and time of the event (timestamp), the event description (computer system, devices, utilized software, software installation, error condition, etc.) and any other relevant metric for a-posteriori analysis [92].
- Introduce a biometric continuous authentication system [35], which every predetermined time (minutes), performs an identity check thus validating and logging user identity.
- Discourage users actions by keeping forensic logs and capability log analysis.

**Detection countermeasures**
Detection discriminates attack attempts and preparation from normal activity and allows to alert the authorities.

- Identify the sensitive data and set up a detection system that prevents all queries on such data, except under specific authorization from a person having a higher rank, and at the same time keeping track of the activity.
- Allow printing reports only in specific printers, which are physically located in a specific protocol office in charge of release of documents under authorization.
- Implement an e-mail system with an automatic cc forwarding to a higher-ranking person.

Selecting one or more among the proposed countermeasures and implementing them properly in the original model, allows to re-evaluate the security of the Secure! system with respect to the considered attack.

### 5.5.6 *Security Assessment*

This section provides the results of the security assessment we perform on the SECURE! framework, with respect to the data theft attack. The attack execution graph of Figure 34 is valid both for SA and SE. Indeed, the ADVISE formalism

allows to define different adversary profiles, which are defined through a set of specific characteristic.

The adversary profile defines a set of access domains, knowledges, and attack skills, owned by the attacker before the attack begins. The attacker assigns a specific payoff value to the attack goal, which is obtained by achieving that goal. Moreover, each attackers defines three attack preference weights: maximizing the payoff ($Weight_{payoff}$), e.g., achieving the attack goal; minimizing costs ($Weight_{cost}$); and minimizing the probability of being detected ($Weight_{detection}$). Each of the attack step has a specific duration of time, cost, success probability, and detection probability, which are specific for each adversary profile, based on the ability to carry out the attack. Table 10 shows the set of parameters we use to define both the SA and SE profiles. We want to point out that the values are arbitrary and they are used for illustrative purpose only.

For both the profiles, the insider has little interest in minimizing the costs, as the cost of attack are irrelevant; while he mainly cares about detection and then about the potential payoff. Due to the different ability and privileges of the insiders, Table 10 shows different values of the success probability for each attack step, reasonably the probability to successfully get through the attack steps is higher for the system administrator profile with respect to System Expert. Indeed SA has the chance to follow more attack paths with respect to SE.
The measure of interest we consider for the analysis is the success probability of the insider attack, without being detected. The analysis we perform aims at assessing the system in its original design; then we perform the same analysis on the system with the *detection countermeasures* implemented, which allows to decrease the success probability of the attack steps. Figure 36 shows both the analysis results, as expected the success probability to achieve the attack

| | | SA | SE |
|---|---|---|---|
| Adversary Preferences | $Weight_{payoff}$ | 0.4 | 0.4 |
| | $Weight_{cost}$ | 0.1 | 0.1 |
| | $Weight_{detection}$ | 0.5 | 0.5 |
| Success Probability Attack Steps | *Admin Remote Login* | 1 | / |
| | *Admin Local Login* | 1 | / |
| | *SECURE! Login* | 1 | 1 |
| | *Disable System Log* | 0.9 | / |
| | *Antivirus Configuration* | 0.9 | 0.75 |
| | *Putting Trojan Horse* | 0.9 | 0.6 |
| | *Data Base Access* | 0.99 | / |
| | *Copy to External Storage Device* | 0.99 | 0.9 |
| | *Send E-Mail* | 0.99 | 0.9 |
| | *Print Report* | 0.9 | 0.8 |
| | *SECURE! Reporting* | 0.9 | 0.9 |

Table 10.: Profiles of System Administrator and System Expert

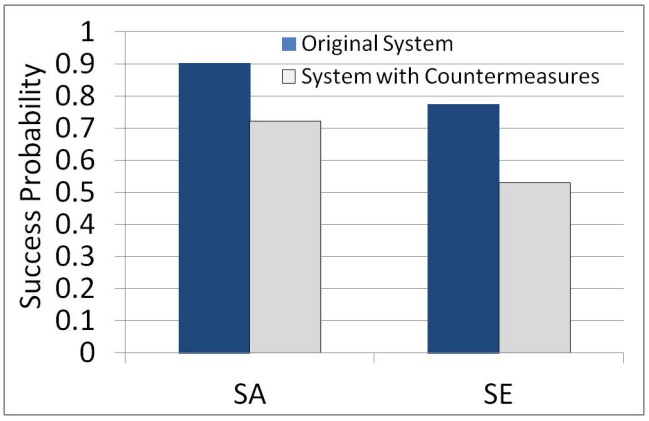


Figure 36.: Success probability to achieve the attack goal for System Administrator (SA) and System Expert (SE)

goal is high, especially for SA. Furthermore, we observe that by implementing the detection countermeasures decreases the success probability. A more interesting analysis could also take into account the costs of implementing the countermeasures, which at this stage are considered costless.

## 5.6 CONCLUSIONS

Increasing attention is being paid to insider threats and attacks. Several techniques exist to avoid or detect the risk that a legitimate user abuses of his authority in the usage of the system. However, we identified a lack in the definition of a methodology and related supports for the systematic investigation and quantitative assessment of insider threats. Investigation of insider threats and mitigation is a well-known, recent topic which highlights the growing need of solutions as systems became more open, dynamics and with non-fixed boundaries, and profiles of potential users multiply.
This part of the thesis presented a detailed methodology for the assessment of insider threats and an extensive case study. Moreover, the application of the methodology to the Secure! case study was carried out through a deeper investigation of the potential threats and a new insider (System Expert) was considered, along with the System Administrator. These improvements allowed us to formalize the methodology and to re-apply it within the Secure! project, leading us to produce new considerations on the system vulnerabilities related to the insider threats and suggest to the designer new and appropriate countermeasures.

# 6 CONCLUSIONS

Model-based evaluation can be used to estimate the degree to which a given design provides the required non functional properties, thus allowing system designer to understand and learn about meaningful characteristic of the system, to detect possible weak points or bottlenecks, to perform early validation of dependability, performance, and security requirements, or to suggest solutions for future releases or modifications.
The number of complex systems and their interdependencies are continuously increasing and affecting the way we live. Thus, analysis of such systems, such as critical infrastructures, is a very difficult task especially because of interdependent structure and behaviour of their composing parts. Complexity, heterogeneity, interdependency and, especially, evolution of system/services specifications, related operating environments and user needs, are more and more highly relevant characteristics of modern and future software applications. Approaches to dependability, performance and security are challenged when systems are made up of networks of heterogeneous applications/devices, especially when operating in unpredictable open-world settings. The research community is tackling this problem and exploring means for enabling interoperability at the application level.

The thesis addressed the study and the analysis of aspects of dependability, performance and security of complex systems, through model-based approaches. The dissertation was carried out in three main parts. The first part aimed at showing how the use of stochastic model-based approaches is crucial for the assessment of critical infrastructures (e.g., electrical power systems), thus allowing the realization and the improvement of such system with respect to dependability aspects. Then we dealt with heterogeneous and interoperable systems by providing a model-based approach for the assessment at pre-deployment time, which is able to suggest, if necessary, possible enhancement of the system, allowing to satisfy the dependability and performance requirements. After the deployment of the system, the proposed approach allows a refinement of the model parameters by collecting real values at run-time. Finally, we dealt with security aspects of the systems, specifically related to the insider threats, and we propose a specific methodology for the assessment of the system through model-based approaches.

In particular, the first part of the dissertation is related to the analysis of resilience regarding critical infrastructure, in particular in the context of elec-

trical power systems. This study investigates the possibility of extending the EPS modelling framework already developed, in order to be able to deal with scenarios characterized by heterogeneity of the loads criticality and the repair time of failed power lines, in presence of one, or a cluster of, failed power lines. By extending the set of the considered heterogeneity aspects, we represented more faithfully in our analyses the variety of processes that are in place in existing system configurations. The main novelty we have introduced, with respect to basic framework, is in the criterion applied to select the grid components for heterogeneity and failure. In fact, while random choices have been made in the past, here we directed the choice on specific power grid elements, to investigate on power loss in presence of well identified circumstances. This kind of analysis is useful, e.g., to electric operators to understand the robustness of their grid when affected by malfunctions in specific areas of the topology, or to set-up appropriate contractual policies with users requiring specific service conditions. Also refinements on the measures to evaluate have been performed, by defining indicators better representative of the blackout size from both a user and operator point of view.

The performed analyses, although limited to the considered scenarios, are successful in showing: *i)* the ability of the framework to account for heterogeneous characteristics and failure phenomena affecting specific grid elements, and *ii)* the importance of accounting for these aspects to get accurate results in specific EPS conditions, e.g., when compared with more simplistic cases where homogeneous load criticality is assumed.

Further explorations would be interesting, e.g., in terms of additional or more complex inter-related heterogeneity aspects to address. Also, enriching the indicators of quality of service perceived by users would be another direction to explore, e.g., coupling the mean values considered in this dissertation for the undelivered load indicator with information on the distribution function and the trend over time of the indicator itself, useful to power grid operators to plan proper reaction in order to limit the damage caused to user by the power loss, in relation with user's usage of the requested load.

The second part of the thesis dealt with the pre-deployment enhancement of dynamic and heterogeneous interoperable systems through stochastic model-based analysis, together with an on-line refinement of model parameters by collecting actual values during system operation. This has been discussed in the context of networks of interoperable systems, as targeted by the EU project CONNECT, which aims at the synthesis of mediating software bridges (CONNECTors) that allow interoperability among heterogeneous devices. In this context we have investigated an automated approach to select a proper dependability and performance mechanism and how to apply it on the model of the system in order to meet given dependability and performance requirements. In addition, we have investigated a generic method for identifying the elements in the CONNECTor that must be reinforced with the selected dependability mechanism in

order to improve performance and dependability related metrics. Moreover, we proposed an approach aiming at integrating the pre-deployment and run-time analysis to fulfill the dependability and performance assessment needs in dynamic and evolving contexts. Indeed, due to the uncertainty about the interoperable systems and the dynamic context, the pre-deployment analysis may lead to incorrect results, caused by the inaccuracy of initial model parameters. An on-line monitoring of the system, during its operation, is then required in order to gather relevant data useful to refine the model parameters and to perform a more refined analysis.
The approach has been demonstrated through a case study based on the GMES European Programme. Specifically, a scenario describing the management of forest-fire at the border between two different countries has been considered, where several resources, provided by both countries, have the same functionalities, but use different protocols and therefore a CONNECTOR is necessary to allow interoperation.
For future work, it would be interesting to investigate methods to predict, with a satisfactory level of accuracy, the minimum set of model elements for the enhancement that would be necessary for requirement satisfaction, thus optimizing the system enhancement at pre-deployment time. Moreover, in order to reduce the time of the analysis at run-time, it could be interesting to extend the analysis at pre-deployment time in order to evaluate a range for the parameters values via sensitivity analysis, thus avoiding repetitions of the analysis triggered after the deployment of the system.

The last part of the thesis aimed at dealing with security issue, specifically related to the insider threats assessment, through a model-based methodology. Security is one of the main problems that companies have to face with, especially ICT ones. Together with the multitude of external attacks to which a system is potentially exposed, there are insider attacks i.e., when users, with legitimate access to the system, abuse or misuse of their power, thus leading to unexpected security violations (e.g., disclosure of sensitive information, sabotage of the system, and so on). Due to their characteristics to be carried out from the inside, these attacks are very difficult to detect and mitigate. It is a consequence that insider attacks constitute a real threat for ICT organizations.
The thesis addressed this issue by providing a specific methodology for the insider threats assessment, based on a model-based approach. Indeed, we have found the lack of a specific methodology, while the existing techniques are mainly designed for specific cases.
The proposed methodology aims at analysing both the system and the interactions that (category of) users can have with it. Specifically, the users are profiled, and specific attributes are assigned to each profile. Moreover, the actions/attacks the insiders can carry on the system are modelled and analysed according to specific security measures of interest, thus allowing us to obtain a quantitative

assessment of the system security with respect to the insider threats.
We have carried out this part of the thesis in the context of the Italian research Project SECURE!, which provided us the opportunity to apply the proposed methodology on the SECURE! framework developed within the project.

Overall, the thesis provides an advancement in the study of complex systems by exploiting model-based approaches to support the assessment activities of such systems with respect to dependability, performance, and security properties. We contributed to assessment activities of critical and complex systems by developing modeling approaches that aim at improving and refining the analyzed system, applicable both as design-time enhancement techniques, and as run-time refinement methods when properly supported by a system monitoring facility. Moreover, the thesis offers a new methodology for security assessment of the insider threats, based on modeling approaches, which allows to quantitatively evaluate security properties with respect to potential insider attacks.

Although the contribution of the thesis is an important step in the assessment of complex systems, dependability, performance and security are more and more required to be considered in strict relationship. Therefore, taking final decisions on the system under development based on analysis results obtained through separate evaluation models only, would limit the system quality. Methods to evolve from this essential, but basic step, towards capturing consequences of failures and attacks on non-functional properties (dependability, performance and security) in combination, are undoubtedly relevant for a more accurate system analysis. This is a research challenge that certainly deserves to be further investigated in order to identify approaches that allow comprehensive analysis of today's complex systems.

# BIBLIOGRAPHY


[1] Iso 27001. http://www.27000.org/iso-27001.htm, accessed December 2014.

[2] Secure! Regional Project POR-CREO 2007-2013, http://secure.eng.it/, accessed December 2014.

[3] Security of Energy Systems (SoES). http://www.soes-project.eu, accessed December 2014.

[4] Trustworthy Cyber Infrastructure for the Power Grid (TCIPG). http://tcipg.org/, accessed December 2014.

[5] Ieee std 610.12-1990 (n.d.). ieee standard glossary of software engineering terminology, 1990. http://ieeexplore.ieee.org/.

[6] *Common Criteria for Information Technology Security Evaluation (CC)*. Version 3.1. September 2012.

[7] *Guide for conducting risk assessment*. NIST SP 800-30. National Institute of Standards and Technology, September 2012.

[8] *Security and Privacy Controls for Federal Information Systems and Organizations*. NIST SP 800-53r4. National Institute of Standards and Technology, April 2013.

[9] Emerging trends in {ICT} security. In Babak Akhgar and Hamid R. Arabnia, editors, *Emerging Trends in {ICT} Security*, pages i – iii. Morgan Kaufmann, Boston, 1st edition, 2014.

[10] Power blackout risks: Risk management options, November 2011. http://www.thecroforum.org/cro-forum-positioning-on-power-blackout-risks/, accessed December 2014.

[11] *Rapporto CLUSIT 2014. Sulla Sicurezza ICT in Italia*. September, 2014.

[12] AFTER. European Project AFTER - A Framework for electrical power sysTems vulnerability identification, dEfense and Restoration. http://www.after-project.eu/, accessed December 2014.

[13] Muhammad Afzaal, Cesario Di Sarno, Luigi Coppolino, Salvatore D'Antonio, and Luigi Romano. A resilient architecture for forensic storage

of events in critical infrastructures. In *High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on*, pages 48–55, Oct 2012.

[14] National Transportation Safety Board Independent Federal Agency. Railroad Train Derailment - Preliminary Report DCA14MR004, January 13, 2014. `http://www.ntsb.gov/investigations/2013/casselton_nd/casselton_nd.html`, accessed December 2014.

[15] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, May 1984.

[16] Akamai Technologies, Inc. Akamai sureroute for failover and performance, 2003.

[17] Arachi Alessandra. Guasto a centrale, blackout elettronico a roma, January 3, 2004. `http://archiviostorico.corriere.it/2004/gennaio/03/Guasto_centrale_blackout_elettronico_Roma_co_9_040103051.shtml`, accessed December 2014.

[18] Qutaibah Althebyan. *Design and Analysis of Knowledge-base Centric Insider Threat Models*. PhD thesis, Fayetteville, AR, USA, 2008. AAI3329146.

[19] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35:131–145, October 2001.

[20] Andrea Bondavalli and Silvano Chiaradonna and Felicita Di Giandomenico. Model-Based Evaluation as a Support to the Design of Dependable Systems. In *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, pages 57–86. Wiley, 2005.

[21] BBC News Asia. Missing malaysia plane mh370: What we know. `http://www.bbc.com/news/world-asia-26503141`, year = September 8, 2014.

[22] Algirdas Avizienis and John P. J. Kelly. Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80, August 1984.

[23] Algirdas Avizienis, Jean-Claude Laprie, Brian Randel, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[24] Gianfranco Balbo. Introduction to Stochastic Petri Nets. In Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 84–155. Springer, 2001.

[25] Yaneer Bar-Yam. General features of complex systems. *Encyclopedia of Life Support Systems (EOLSS), UNESCO, EOLSS Publishers, Oxford, UK*, 2002.

[26] Luciano Baresi, Elisabetta Di Nitto, and Carlo Ghezzi. Toward open-world software: Issue and challenges. *Computer*, 39(10):36–43, Oct 2006.

[27] Marco Beccuti, Silvano Chiaradonna, Felicita Di Giandomenico, Susanna Donatelli, Giovanna Dondossola, and Giuliana Franceschinis. Quantification of dependencies between electrical and information infrastructures. *International Journal of Critical Infrastructure Protection*, 5(1):14 – 27, 2012.

[28] Matt Bishop. Position: "Insider" is Relative. In *Proceedings of the 2005 Workshop on New Security Paradigms*, NSPW '05, pages 77–78, New York, NY, USA, 2005. ACM.

[29] Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen, and Carrie Gates. We have met the enemy and he is us. In *Proceedings of the 2008 Workshop on New Security Paradigms*, NSPW '08, pages 1–12, New York, NY, USA, 2008. ACM.

[30] Andrea Bobbio and Miklós Telek. Non-Exponential Stochastic Petri Nets: an Overview of Methods and Techniques. In *Computer Systems Science & Engineering*, pages 339–351, 1998.

[31] Gunter Bolch, Stefan Greiner, Hermann De Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley & Sons, 2006.

[32] Borzoo Bonakdarpour and Sandeep S. Kulkarni. SYCRAFT: A Tool for Synthesizing Distributed Fault-Tolerant Programs. In *CONCUR 2008: International Conference on Concurrency Theory*, pages 167–171. Springer Verlag, 2008.

[33] R. T. Braden. RFC 1122: Requirements for Internet hosts — communication layers, oct 1989. `https://tools.ietf.org/html/rfc1122`, accessed December 2014.

[34] Timothy Casey. Threat Agent Library Helps Identify Information Security Risks, September 2007. `http://communities.intel.com/docs/DOC-1151`, accessed December 2014.

[35] Andrea Ceccarelli, Andrea Bondavalli, Francesco Brancati, and Ernesto La Mattina. Improving security of internet services through continuous and transparent user identity verification. In *Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems*, SRDS '12, pages 201–206, Washington, DC, USA, 2012. IEEE Computer Society.

[36] Andrea Ceccarelli, Marco Vieira, and Andrea Bondavalli. A Service Discovery Approach for Testing Dynamic SOAs. In *Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, ISORCW '11, pages 133–142, Washington, DC, USA, 2011. IEEE Computer Society.

[37] Andrea Ceccarelli, Marco Vieira, and Andrea Bondavalli. A Testing Service for Lifelong Validation of Dynamic SOA. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pages 1–8, Nov 2011.

[38] Silvano Chiaradonna, Felicita Di Giandomenico, and Paolo Lollini. Evaluation of critical infrastructures: Challenges and viable approaches. In Rogerio de Lemos, Felicita Di Giandomenico, Cristina Gacek, Henry Muccini, and Marlon Vieira, editors, *Architecting Dependable Systems V*, volume 5135 of *Lecture Notes in Computer Science*, pages 52–77. Springer Berlin Heidelberg, 2008.

[39] Silvano Chiaradonna, Felicita Di Giandomenico, and Nicola Nostro. Analysis of Electric Power Systems accounting for interdependencies in heterogeneous scenarios. In *Ninth European Dependable Computing Conference (EDCC 2012)*, May 2012.

[40] Silvano Chiaradonna, Felicita Di Giandomenico, and Nicola Nostro. Model-based assessment of multi-region electric power systems showing heterogeneous characteristics. In Frank Ortmeier and Peter Daniel, editors, *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, pages 328–339. Springer Berlin Heidelberg, 2012.

[41] Silvano Chiaradonna, Felicita Di Giandomenico, and Nicola Nostro. Stochastic Assessment of Power Systems in Presence of Heterogeneity. *International Journal of Critical Computer-Based Systems*, 4(4):326–348, February 2013.

[42] Hoon Choi, Vidyadhar G. Kulkarni, and Kishor S. Trivedi. Markov regenerative stochastic Petri nets. *Performance Evaluation*, 20(1-3):337–357, May 1994.

[43] Gianfranco Ciardo, Alex Blakemore, Philip F. Jr. Chimento, Jogesh K. Muppala, and Kishor S. Trivedi. Automated generation and analysis of markov reward models using stochastic reward nets. In Carl D. Meyer and Robert J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, volume 48 of *The IMA Volumes in Mathematics and its Applications*, pages 145–191. Springer, 1993.

[44] Allan Clark, Stephen Gilmore, Jane Hillston, and Mirco Tribastone. Stochastic process algebras. In Marco Bernardo and Jane Hillston, editors, *Formal Methods for Performance Evaluation*, volume 4486 of *LNCS*, pages 132–179. Springer, 2007.

[45] Graham Cluley. Dragonfly hackers target 1000 western energy firms, industrial control systems, July 1, 2014. `http://grahamcluley.com/2014/07/dragonfly-hackers-target-1000-western-energy-firms-industrial-control-systems/`, accessed December 2014.

[46] CONNECT. Emergent Connectors for Eternal Software Intensive Networked Systems (EU FP7 Project Connect (FP7 231167)). `https://www.connect-forever.eu/`, accessed December 2014, 2009-2013.

[47] CONNECT Consortium. Deliverable 6.3 – Experiment scenarios, prototypes and report - Iteration 2, 2012.

[48] Secure! Consortium. Deliverable 2.2 - architettura della infrastruttura di gestione dell'affidabilità, sicurezza, fiducia e privacy. Technical report, July 2013.

[49] Tara Conway, Susan Keverline, Michelle Keeney, Eileen Kowalski, Megan Williams, Dawn Cappelli, Andrew P. Moore, Stephanie Rogers, and Timothy J. Shimeall. Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. May 2005. `http://www.cert.org/archive/pdf/insidercross051105.pdf`, accessed December 2014.

[50] CRUTIAL. European Project CRUTIAL - CRitical Utility InfrastructurAL resilience (Project IST-FP6-027513). `http://crutial.rse-web.it/`, accessed December 2014, 2006.

[51] Marc Dacier and Yves Deswarte. Privilege graph: An extension to the typed access matrix model. In Dieter Gollmann, editor, *Computer Security - ESORICS 94*, volume 875 of *Lecture Notes in Computer Science*, pages 319–334. Springer Berlin Heidelberg, 1994.

[52] Marc Dacier, Yves Deswarte, and Mohamed Kaâniche. Information systems security. chapter Models and Tools for Quantitative Assessment of Operational Security, pages 177–186. Chapman & Hall, Ltd., London, UK, 1996.

[53] David Daly, Daniel D. Deavours, Jay M. Doyle, Aaron J. Stillman, and Patrick G. Webster. Möbius: An extensible tool for performance and dependability modeling. In B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *11th International Conference TOOLS 2000*, volume 1786 of *LNCS*, pages 332–336. Springer Verlag, 2000.

[54] Felicita Di Giandomenico, Antonia Bertolino, Antonello Calabrò, and Nicola Nostro. An approach to adaptive dependability assessment in dynamic and evolving connected systems. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 4(1):1–25, 2013.

[55] Salvatore Distefano and Antonio Puliafito. Dependability evaluation with dynamic reliability block diagrams and dynamic fault trees. *IEEE Transactions on Dependable and Secure Computing*, 99(2), 2008.

[56] Alejandro D. Dominguez-Garcia, John G. Kassakian, Joel E. Schindall, and Jeffrey J. Zinchuk. An integrated methodology for the dynamic performance and reliability evaluation of fault-tolerant systems. *Reliability Engineering & System Safety*, 93(11):1628 – 1649, 2008.

[57] Joanne Bechta Dugan. Software reliability analysis using fault trees. In M. Lyu, editor, *Handbook of Software Reliability Engineering*, pages 615–660. IEEE Computer Society Press and McGrawHill, 1996.

[58] Ali Ebnenasir. Automatic synthesis of fault-tolerance, 2005. PhD Dissertation, Michigan State University.

[59] Engineering and Physical Sciences Research Council (EPSRC) and - National Natural Science Foundation of China (NSFC). Developing Scalable Smart Grid Infrastructure to Enable Secure Transmission System Control. `http://www.abdn.ac.uk/engineering/research/epsrcnsfc-project-339.php`, accessed December 2014, 2013-2016.

[60] Nicolas Falliere, Liam O. Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 2011.

[61] Andrea Fiaschetti, Francesco Lavorato, Vincenzo Suraci, Andi Palo, Andrea Taglialatela, Andrea Morgagni, Renato Baldelli, and Francesco Flammini. On the use of semantic technologies to model and control security, privacy and dependability in complex systems. In *SAFECOMP*, pages 467–479, 2011.

[62] Francesco Flammini, Stefano Marrone, Nicola Mazzocca, and Valeria Vittorini. A new modeling approach to the safety evaluation of n-modular redundant computer systems in presence of imperfect maintenance. *Reliability Engineering & System Safety*, 94(9):1422 – 1432, 2009.

[63] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 37–54, Washington, DC, USA, 2007. IEEE Computer Society.

[64] Steven M. Furnell, Sokratis Katsikas, Javier Lopez, and Ahmed Patel. *Securing Information and Communications Systems: Principles, Technologies, and Applications*. Artech House, Inc., Norwood, MA, USA, 1st edition, 2008.

[65] Gary Doss and Guvirender Tejay. Developing insider attack detection model: A grounded approach. In *Intelligence and Security Informatics, 2009. ISI '09. IEEE International Conference on*, pages 107–112, June 2009.

[66] Rahul Ghosh, DongSeong Kim, and Kishor S. Trivedi. System resiliency quantification using non-state-space and state-space analytic models. *Reliability Engineering & System Safety*, 116(0):109 – 125, 2013.

[67] Felicita Di Giandomenico, Massimiliano L. Itria, Paolo Masci, and Nicola Nostro. Automated synthesis of dependable mediators for heterogeneous interoperable systems. *Reliability Engineering & System Safety*, 132(0):220 – 232, 2014.

[68] Gianfranco Ciardo and Reinhard German and Christoph Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. In *Petri Nets and Performance Models, 1993. Proceedings., 5th International Workshop on*, pages 170–179, October 1993.

[69] Thomas R. Gruber and Gregory R. Olsen. An ontology for engineering mathematics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR*, pages 258–269. Morgan Kaufmann, 1994.

[70] Peter G. Harrison and Ben Strulo. SPADES - a Process Algebra for Discrete Event Simulation. *Journal of Logic and Computation*, 10(1):3–42, 1 2000.

[71] Boudewijn R. Haverkort. Markovian models for performance and dependability evaluation. In Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors, *Lectures on Formal Methods and PerformanceAnalysis*, volume 2090 of *Lecture Notes in Computer Science*, pages 38–83. Springer Berlin Heidelberg, 2001.

[72] Jane Hillston. *A Compositional Approach to Performance Modeling*. PhD thesis, Cambridge University Press, 1995.

[73] Markus C. Huebscher and Julie A. McCann. A survey of autonomic computing: degrees, models, and applications. *ACM Computing Surveys*, 40(3):237 – 254, 2008.

[74] Jeffrey Hunker and Christian W. Probst. Insiders and insider threats - an overview of definitions and mitigation techniques. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 2(1):4–27, 3 2011.

[75] IEEE RTS Task Force of the APM Subcommittee. IEEE reliability test system. *IEEE Trans. Power App. Syst.*, PAS-98(6):2047–2054, Nov 1979.

[76] IEEE RTS Task Force of the APM Subcommittee. The IEEE reliability test system - 1996. *IEEE Trans. Power Syst.*, 14(3):1010–1020, 1999.

[77] International Electrotechnical Commission. IEC 60125: "Fault tree analysis (FTA)", December 2006. Second Edition.

[78] IRRIIS. European Project IRRIIS - Integrated risk reduction of information-based infrastructure systems. `http://irriis.org`, accessed December 2014.

[79] Anusha Iyer and Hung Q. Ngo. Towards a theory of insider threat assessment. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, DSN '05, pages 108–117, Washington, DC, USA, 2005. IEEE Computer Society.

[80] John Lowry. An initial foray into understanding adversary planning and courses of action. In *DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings*, volume 1, pages 123–133 vol.1, 2001.

[81] Allen M. Johnson, Jr. and Miroslaw Malek. Survey of software tools for evaluating reliability, availability, and serviceability. *ACM Comput. Surv.*, 20(4):227–269, December 1988.

[82] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. Detecting anomalous access patterns in relational databases. *The VLDB Journal*, 17(5):1063–1077, August 2008.

[83] Marc-Olivier Killijian, Ludovic Courtes, and David Powell. A survey of cooperative backup mechanisms, October 2006. LAAS Technical Report 06472.

[84] Rüdiger Klein, Erich Rome, Césaire Beyel, Ralf Linnemann, Wolf Reinhardt, and Andrij Usov. Information modelling and simulation in large interdependent critical infrastructures in irriis. In Roberto Setola and Stefan Geretshuber, editors, *Critical Information Infrastructure Security*, volume 5508 of *Lecture Notes in Computer Science*, pages 36–47. Springer Berlin Heidelberg, 2009.

[85] Brian Krebs. Internet worm hits airline, banks, January 26, 2003. `http://www.washingtonpost.com/wp-dyn/articles/A46928-2003Jan26.html`, accessed December 2014.

[86] Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S. Choi. A taxonomy of computer program security flaws. *ACM Comput. Surv.*, 26(3):211–254, September 1994.

[87] Jean-Claude Laprie. From dependability to resilience. In *38th IEEE/IFIP International Conference on Dependable Systems and Networks Supplemental Volume, Anchorage, Alaska, June 2008.*, pages G8–G9. IEEE, 2008.

[88] Elizabeth LeMay, Michael D. Ford, Ken Keefe, William H. Sanders, and Carol Muehrcke. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In *Proceedings of the 2011 Eighth International Conference on Quantitative Evaluation of SysTems*, QEST '11, pages 191–200, Washington, DC, USA, 2011. IEEE Computer Society.

[89] Phillip Y. Lipscy, Kenji E. Kushida, and Trevor Incerti. The fukushima disaster and japan's nuclear plant vulnerability in comparative perspective. pages 6082–6088, May 29, 2013.

[90] Michael R. Lyu. *Software Fault Tolerance*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[91] Manish Malhotra and Kishor S. Trivedi. Power-hierarchy of dependability-model types. *IEEE Transactions on Reliability*, 43(3):493–502, 1994.

[92] Marcello Cinque and Domenico Cotroneo and Roberto Natella and Antonio Pecchia. Assessing and improving the effectiveness of logs for the analysis of software faults. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 457–466, June 2010.

[93] Marco Ajmone Marsan and Giovanni Chiola. On Petri nets with deterministic and exponentially distributed firing times. In Rozenberg, Grzegorz, editor, *Advances in Petri Nets 1987*, volume 266 of *LNCS*, pages 132–145. Springer Berlin Heidelberg, 1987.

[94] Sunu Mathew, Michalis Petropoulos, HungQ. Ngo, and Shambhu Upadhyaya. A data-centric approach to insider attack detection in database systems. In Somesh Jha, Robin Sommer, and Christian Kreibich, editors, *Recent Advances in Intrusion Detection*, volume 6307 of *Lecture Notes in Computer Science*, pages 382–401. Springer Berlin Heidelberg, 2010.

[95] Michael K. Molloy. Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers*, 31(9):913–917, 9 1982.

[96] Leonardo Montecchi, Paolo Lollini, Andrea Bondavalli, and Ernesto La Mattina. Quantitative security evaluation of a multi-biometric authentication system. In Frank Ortmeier and Peter Daniel, editors, *Computer Safety, Reliability, and Security*, volume 7613 of *LNCS*, pages 209–221. Springer, 2012.

[97] Andrew P. Moore, Dawn M. Cappelli, and Randall F. Trzeciak. The "Big Picture" of Insider IT Sabotage Across U.S. Critical Infrastructures. In Salvatore J. Stolfo, Steven M. Bellovin, Angelos D. Keromytis, Shlomo Hershkop, Sean W. Smith, and Sara Sinclair, editors, *Insider Attack and Cyber Security*, volume 39 of *Advances in Information Security*, pages 17–52. Springer US, 2008.

[98] Michael Mrissa, Stefan Dietze, Philippe Thiran, Chirine Ghedira, Djamal Benslimane, and Zakaria Maamar. Context-based Semantic Mediation in Web Service Communities. In Irwin King and Ricardo Baeza-Yates, editors, *Weaving Services and People on the World Wide Web*, pages 49–66. Springer Berlin Heidelberg, 2009.

[99] Jogesh K. Muppala, Ricardo M. Fricks, and Kishor S. Trivedi. Techniques for system dependability evaluation. In Winfried K. Grassmann, editor, *Computational Probability*, volume 24 of *International Series in Operations Research & Management Science*, pages 445–479. Springer US, 2000.

[100] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[101] Meenakshi Nagarajan, K. Verma, A.P. Sheth, J. Miller, and J. Lathem. Semantic Interoperability of Web Services - Challenges and Experiences. In *Web Services, 2006. ICWS '06. International Conference on*, pages 373–382, 2006.

[102] David M. Nicol, William H. Sanders, and Kishor S. Trivedi. Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48–65, 2004.

[103] Nicola Nostro, Andrea Ceccarelli, Andrea Bondavalli, and Francesco Brancati. A methodology and supporting techniques for the quantitative assessment of insider threats. In *Proceedings of the 2Nd International Workshop on Dependability Issues in Cloud Computing*, DISCCO '13, pages 3:1–3:6, New York, NY, USA, 2013. ACM.

[104] Nicola Nostro, Andrea Ceccarelli, Andrea Bondavalli, and Francesco Brancati. Insider threat assessment: A model-based methodology. *SIGOPS Oper. Syst. Rev.*, 48(2):3–12, 2014.

[105] Transportation Safety Board of Canada. Runaway and main-track derailment, montreal, maine & atlantic railway freight train MMA-002, mile 0.24, sherbrooke subdivision, lac-mégantic, quebec, 06 july 2013., 2014. `http://epe.lac-bac.gc.ca/100/201/301/weekly_checklist/2014/internet/w14-34-U-E.html/collections/collection_2014/bst-tsb/TU3-6-13-0054-eng.pdf`, accessed December 2014.

[106] Jon Olnes. Development of security policies. *Comput. Secur.*, 13(9):628–636, October 1994.

[107] Rodolphe Ortalo, Yves Deswarte, and Mohamed Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *Software Engineering, IEEE Transactions on*, 25(5):633–650, 1999.

[108] James Lyle Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[109] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[110] B. Plateau and K. Atif. Stochastic automata network for modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 10 1991.

[111] ReSIST Consortium. EU project ReSIST: Resilience for Survivability in IST. Deliverable D11: Support for Resilience-Explicit Computing - first edition. Technical report, 2007. http://www.resist-noe.org/.

[112] Steven M. Rinaldi, James P. Peerenboom, and Terrence K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems Magazine*, 21(6):11–25, 12 2001.

[113] Ronald A. Howard. *Dynamic Probabilistic Systems: Markov Models*, volume 1. John Wiley and Sons, New York, 1971.

[114] Juan Carlo Ruiz, Marc-Olivier Killijian, Jean-Charles Fabre, and Pascale Thvenod-Fosse. Reflective fault-tolerant systems: from experience to challenges. *IEEE Transactions on Computers*, 52(2):237 – 254, 2003.

[115] John Rushby. Runtime verification. chapter Runtime Certification, pages 21–35. Springer-Verlag, Berlin, Heidelberg, 2008.

[116] Sandeep S. Kulkarni and Anish Arora. Automating the Addition of Fault-Tolerance. In *6th Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 82–93, 2000.

[117] William H. Sanders and John F. Meyer. Stochastic Activity Networks: Formal Definitions and Concepts. In Brinksma, E. and Hermanns, H. and Katoen, J. P., editor, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 315–343. Springer Verlag, 2001.

[118] Phillip F. Schewe. *The Grid: A Journey Through the Heart of Our Electrified World Hardcover*. Joseph Henry Press, Washington D.C., December 1, 2006.

[119] Walter Schmitz. Simulation experiments: the emerging instruments for CIP. *Int. Journal of Critical Infrastructures (IJCIS)*, 5(1/2):5–23, 2009.

[120] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000.

[121] Matthias Schonlau, William DuMouchel, Wen-Hua Ju, Alan F. Karr, Martin Theus, and Yehuda Vardi. Computer intrusion: Detecting masquerades. *Statistical Science*, 16(1):pp. 58–74, 2001.

[122] Secure! project. Deliverable 1.1 - Requirements specification, 2013.

[123] Secure! project. Deliverable 4.1 - Modelli di gestione dei contenuti e delle decisioni, 2013.

[124] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 273–, Washington, DC, USA, 2002. IEEE Computer Society.

[125] Oleg Mikhail Sheyner. *Scenario Graphs and Attack Graphs*. PhD thesis, Pittsburgh, PA, USA, 2004. AAI3126929.

[126] Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems (3rd Ed.): Design and Evaluation*. A. K. Peters, Ltd., Natick, MA, USA, 1998.

[127] Silowash George and Cappelli Dawn and Moore Andrew and Trzeciak Randall and Shimeall Timothy and Flynn Lori. Common Sense Guide to Mitigating Insider Threats. Technical report, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=34017 accessed December 2014.

[128] Silvano Chiaradonna and Felicita Di Giandomenico and Nicola Nostro. Modeling and analysis of the impact of failures in Electric Power Systems organized in interconnected regions. In *Dependable Systems Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*, pages 442–453. June 2011.

[129] Silvano Chiaradonna and Felicita Di Giandomenico and Paolo Lollini. Definition, Implementation and Application of a Model-based Framework for the Analysis of Interdependencies in Electric Power Systems. *International Journal of Critical Infrastructure Protection (IJCIP), Elsevier*, 4(1):24–40, 2011.

[130] Silvano Chiaradonna and Paolo Lollini and Felicita Di Giandomenico. On a Modeling Framework for the Analysis of Interdependencies in Electric Power Systems. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 185–195. IEEE, 2007.

[131] Michael Stamatelatos, William Vesely, Joanne Dugan, Joseph Fragola, Joseph Minarick, and Jan Railsback. Fault Tree Handbook with Aerospace Applications, 8 2002.

[132] Stefan Hoyer and Halyna Zakhariya and Thorben Sandner and Michael H. Breitner. Fraud Prediction and the Human Factor: An Approach to Include Human Behavior in an Automated Fraud Audit. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 2382–2391, Jan 2012.

[133] Salvatore Stolfo, Steven M. Bellovin, Angelos D. Keromytis, Sara Sinclair, Sean W. Smith, and Shlomo Hershkop. *Insider Attack and Cyber Security: Beyond the Hacker (Advances in Information Security)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 1 edition, 2008.

[134] Chee-Wooi Ten, Chen-Ching Liu, and M. Govindarasu. Vulnerability assessment of cybersecurity for scada systems using attack trees. In *Power Engineering Society General Meeting*, pages 1–8. IEEE, 24-28 2007.

[135] Michael Thildebrandt and Michael Harrison. The temporal dimension of dynamic function allocation. In *Proceeding of 11th European Conference on Cognitive Ergonomics (ECCE)*, pages 283–291, 2002.

[136] Thomas C. Reed. *At the Abyss: An Insider's History of the Cold War Paperback*. Presidio Press, March 1, 2005.

[137] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons Ltd., Chichester, UK, 2nd edition edition, 2002.

[138] Andrij Usov and Cesaire Beyel. Simulating interdependent Critical Infrastructures with SimCIP. *European CIIP Newsletter*, 4(3):6–8, 2008.

[139] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Toward measuring network security using attack graphs. In *Proceedings of the 2007 ACM Workshop on Quality of Protection*, QoP '07, pages 49–54, New York, NY, USA, 2007. ACM.

[140] Andreas Wenger, Isabelle Abele-Wigert, and Myriam Dunn. *International CIIP Handbook 2006*. Center for Security Studies, ETH Zurich, 2006.

[141] Wenjun Zeng and Yingnan Zhu and Haibin Lu and Xinhua Zhuang. Path-Diversity P2P Overlay Retransmission for Reliable IP-Multicast. *IEEE Transactions on Multimedia*, 11(5):960–971, Aug. 2009.

# APPENDICES

# A

## ACRONYMS

| | |
|---|---|
| AC | Alternating Current |
| ADVISE | ADversary VIew Security Evaluation |
| AEG | Attack Execution Graph |
| AGs | Attack graphs |
| APs | Attack Paths |
| AT | Attack Tree |
| BNSF | Burlington Northern and Santa Fe |
| CI | Critical Infrastructure |
| CIP | Critical Infrastructure Protection |
| COTS | Commercial Off-The-Shelf |
| DC | Direct Current |
| EI | Electrical Infrastructure |
| EPS | Electrical Power System |
| FI | Future Internet |
| FEC | Forward Error Correction |
| FT | Fault Tree |
| FTA | Fault Tree Analysis |
| GSPNs | Generalized Stochastic Petri Nets |
| GUI | Graphical User Interface |
| ICT | Information and Communications Technology |
| IT | Information Technology |
| ITCS | Information Technology based Control System |
| LCS | Local Control System |
| LP | Linear Programming |
| LTS | Labeled Transition System |
| MW | Mega Watt |
| NSs | Networked Systems |
| NTSB | National Transportation Safety Board |
| QoS | Quality of Service |

| | |
|---|---|
| RBD | Reliability Block Diagram |
| RG | Reliability Graphs |
| RTS | Regional Telecontrol System |
| SA | System Administrator |
| SAN | Stochastic Activity Network |
| SE | System Expert |
| TAL | Threat Agent Library |
| UAV | Unmanned Aerial Vehicle |
| UGV | Unmanned Ground Vehicles |

# B

# LIST OF GRAPHICS

## FIGURES

TABLES