Formal models for safety analysis of a Data Center system

Mokhtar Walid Bennaceur

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES
université PARIS-SACLAY

Formal Models for Safety Analysis of a Data Center System

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'université de Versailles Saint Quentin-en-Yvelines

École doctorale n°580 Sciences et technologies de l'information et de la communication
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Versailles, le 21/11/2019, par

Walid Mokhtar Bennaceur

Composition du Jury :

Amar Ramdane-Cherif Professeur, Université de Versailles Saint Quentin-en-Yvelines	**Président du Jury**
Jean-Yves Choley Professeur, Institut Supérieur de Mécanique de Paris (Supméca)	**Rapporteur**
Karama Kanoun Directrice de recherche, LAAS-CNRS, Toulouse	**Rapporteur**
Frank Ortmeier Professeur, Otto-von-Guericke Universität Magdeburg, Allemagne	**Examinateur**
Leïla Kloul Maître de conférences (HDR), Université de Versailles Saint Quentin-en-Yvelines	**Directrice de thèse**
Arnaud De Moissac Ingénieur, DCBrain	**Invité**

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Modèles formels pour l'analyse de sûreté de fonctionnement d'un Data Center

Mots clés : Sûreté de fonctionnement, Data Center, Arbres de production, Fiabilité, Disponibilité.

Résumé : Les Data Centers (DCs) ont considérablement évolué pour répondre aux exigences des nouvelles technologies telles que le cloud computing, le e-commerce et les réseaux sociaux. Un DC est un système complexe composé de 3 sous-systèmes : électrique, thermique et réseau. C'est un système de production avec différent types de flux qui circulent, entrainant des dépendances fonctionnelles entre les sous-systèmes. Compte tenu des enjeux majeurs des DCs et de leur complexité, l'analyse de leur sûreté de fonctionnement devient de plus en plus cruciale. Dans cette thèse, nous avons développé une méthodologie d'analyse des DCs. En effet, à notre connaissance, il n'existe aucune étude d'analyse, prenant en compte l'ensemble des sous-systèmes du DC. La méthodologie développée est basée sur la technique de modélisation des Arbres de Production (AP), et permet d'évaluer différentes mesures de performance et de sûreté. En raison des interactions entre les sous-systèmes du DC, nous avons étendu cette technique en introduisant un nouveau mécanisme permettant la modélisation des dépendances entre les flux de différents types. Nous avons également proposé une méthode de résolution des APs afin d'estimer la fiabilité et la disponibilité du système. Enfin, nous avons développé un outil qui met en œuvre la méthodologie d'analyse que nous avons proposée pour calculer les indicateurs de sûreté de fonctionnement et de performance du système du DC.

Title : Formal models for safety analysis of a Data Center system

Keywords : Safety, Data Center, Production Trees, Reliability, Availability.

Abstract : Data Centers (DCs) have evolved dramatically to meet the new technologies demands such as cloud computing, ecommerce and social networks. A DC system is a complex system composed of 3 sub-systems: electrical, thermal and network. It is a production system with different flows circulating, leading to functional dependencies between the sub-systems. Given the major challenges of the DCs, and with their increase of complexity, their safety analysis become crucial. In this thesis, we have proposed a methodology to analyze DCs. Indeed, in our knowledge, no analysis study, taking into account all the DC, Exists. The developed methodology is based on Production Trees (PT) modeling technique and allows assessing different performances and safety indicators. Because of the interactions between the DC sub-systems, we have extended this technique by introducing a new mechanism which allows modeling dependencies between the different types of flows. We proposed also a solution method to assess PT models and estimate system reliability and availability. Finally we developed a graphical tool for our methodology in order to estimate the DC safety and performance indicators.

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

Contents

List of Figures

Acronyms

AccR	**Acc**ess **R**outer
AggS	**Agg**regation **S**witch
BAT	**Bat**tery
BDMP	**B**oolean **D**riven **M**arkov **P**rocess
BDP	**B**ack low-voltage **D**istribution **P**anel
CFD	**C**omputational **F**luid **D**ynamics
CH	**Ch**iller
CONV	**Conv**erter
CRAC	**C**omputer **R**oom **A**ir **C**onditioning
CT	**C**ooling **T**ower
CTMC	**C**ontinuous **T**ime **M**arkov **C**hain
DBA	**D**ysfunctional **B**ehavior **A**nalysis
DC	**D**ata **C**enter
DFT	**D**ynamic **F**ault **T**ree
DRBD	**D**ynamic **R**eliability **B**loc **D**iagram
DSM	**D**omain **S**pecific **M**odeling
DT	**D**istribution **T**able
DTMC	**D**iscrete **T**ime **M**arkov **C**hain
EFM	**E**nergy **F**low **M**odel
EMF	**E**clipse **M**odeling **F**ramework
ET	**E**vent **T**ree
FDP	**F**ront low-voltage **D**istribution **P**anel
FMECA	**F**ailure **M**ode, **E**ffect, and **C**riticality **A**nalysis
FT	**F**ault **T**ree

FTA	**F**ault **T**ree **A**nalysis
GEF	**G**raphical **E**ditor **F**ramework
GMF	**G**raphical **M**odeling **F**ramework
GSPN	**G**eneralized **S**tochastic **P**etri **N**et
GTS	**G**uarded **T**ransition **S**ystem
IT	**I**nformation **T**echnology
LTM	**L**oad **T**ransfer **M**odule
MBSA	**M**odel-**B**ased **S**ystems **A**nalysis
MBSE	**M**odel-**B**ased **S**ystems **E**ngineering
MC	**M**arkov **C**hain
MDE	**M**odel-**D**riven **E**ngineering
MTTF	**M**ean **T**ime **T**o **F**ailure
MTTR	**M**ean **T**ime **T**o **R**epair
OCL	**O**bject **C**onstraint **L**anguage
OM	**O**perational **M**ode
PCWS	**P**rocess **C**hilled **W**ater **S**upply
PDC	**P**robability **D**istribution of **C**apacity
PDU	**P**ower **D**istribution **U**nit
PG	**P**ower **G**enerator
PRA	**P**reliminary **R**isk **A**nalysis
PS	**P**ower **S**ource
PT	**P**roduction **T**ree
QN	**Q**ueueing **N**etwork
QPN	**Q**ueueing **P**etri **N**et
RAW	**R**isk **A**chievement **W**orth
RBD	**R**eliability **B**lock **D**iagram
REC	**Rec**tifier

RR	**R**isk **R**eduction
SAML	**S**afety **A**nalysis **M**odeling **L**anguage
SDEP	**S**tate **D**ependency
SDL	**S**ystem **D**efinition **L**anguage
SE	**S**ystems **E**ngineering
Ser	**Ser**ver
SPN	**S**tochastic **P**etri **N**et
ToR	**T**op **o**f **R**ack
TU	**T**ransfer **U**nit
UPS	**U**ninterruptible **P**ower **S**ystem
VDC	**V**irtual **D**ata **C**enter

Abstract

A Data Center DC is a building whose purpose is to host IT devices to provide different internet services. These devices have three essential needs: a physical space, industrial power energy and a constant supply of cold air. So a DC can be seen as a complex system with 3 different sub-systems: electrical, thermal and network. Physical space is a place where different IT devices are located. Their interconnections form an important network. To ensure constant operation of these devices, energy is provided by the electrical system, and to keep them at a constant temperature, a cooling system is necessary. Each of these needs must be ensured continuously, because the consequence of breakdown of one of them leads to an unavailability of the whole DC system, and this can be fatal for a company. For example, an electrical break of 10 seconds can cause service outage of 10h, and 1 minute of interruption can cost more than 7000 euros. DCs are therefore built to meet strong constraints of continuity of service. These constraints can represent 50% of the DC cost, that is, several billion euros.

In our Knowledge, there exists no safety and performance studies, taking into account the whole DC system with the different interactions between its sub-systems, in the literature. The existing analysis studies are partial and focus only on one sub-system, sometimes two. The main objective of this thesis is to contribute to the safety analysis of a DC system. To achieve this purpose, we study, first, each DC sub-system (electrical, thermal and network) separately, in order to define their characteristics. This step is very important to find the appropriate technique for assessing the different safety indicators (reliability and safety). Each DC sub-system is a production system and consists of combinations of components that transform entrance supplies (energy for the electrical system, air flow for the thermal one, and packets for the network one) into exits, which can be internet services. Currently the existing safety analysis methods for these kinds of systems are inadequate, because the safety analysis must take into account not only the internal state of each component, but also the different production flows circulating between components. For

example, the use of static Fault Trees (FT) is not suitable for these systems, because they look only at the internal state (working or failed) of DC systems components. In this thesis, we consider a new modeling methodology called Production Trees (PT) which allows modeling the relationship between the components of a system with a particular attention to the flows circulating between these components.

The PT modeling technique allows dealing with one kind of flow at once. Thus its application on the electrical sub-system is suitable, because there is only one kind of flows (the electric current). However, when there are dependencies between sub-systems, as in thermal and network sub-systems, different kinds of flows need to be taken into account, making the application of the PT modeling technique inadequate. Therefore, we extend this technique to deal with dependencies between the different kinds of flows in the DC. Accordingly it is easy to assess the different safety indicators of the global DC system, taking into account the interactions between its sub-systems. Moreover we make some performance statistics. We validate the results of our approach by comparing them to those obtained by a simulation tool that we have implemented based on Queuing Network theory.

So far, Production Trees models are not tool supported. Therefore we propose a solution method based on the Probability Distribution of Capacity (PDC) of flows circulating in the DC system. This approach calculates both availability and reliability of the system by using a set of predefined formulas. It is more restricted and provides more accuracy than simulation methods. We implement also the PT model using the AltaRica 3.0 modeling language, and use its dedicated stochastic simulator to estimate the reliability indices of the system. This is very important to compare and validate the obtained results with our assessment method. In parallel, we develop a tool which implements the PT solution algorithm. This is an EMF-based (Eclipse Modeling Framework) with an interactive graphical interface, which allows creating, editing and analyzing PT models. The tool allows also displaying the results, and generates an AltaRica code, which can be subsequently analyzed using the stochastic simulator of AltaRica 3.0 tool.

Chapter 1

Introduction

Systems engineering is a subject that has attracted a lot of attention in the last decades and it is often seen as an emerging discipline [82]. It is an interdisciplinary field of engineering that focuses on how to design and manage complex systems over their life cycles. It begins by identifying the needs of the users and the functions that meet these needs, allocating those functions to the system entities (components), and finally confirming that the system performs as designed and satisfies the needs of the users [16]. In order to confirm that the designed system satisfies the current industrial standards and environmental pressure requirements, the systems designers have to conduct risk analysis studies, also called systems safety. It is the most important specialty within systems engineering which supports program risk management.

Systems safety is the application of engineering and management principles, criteria and techniques to optimize safety by the identification of safety related risks, and eliminating or controlling them by design and procedures [38]. This is very important because it allows determining how to enhance the system reliability, and anticipate possible failures leading to interruptions of services.

Nowadays most of modern systems become more and more complex [40]. A complex system is a system with a large number of components, interconnections and interactions. In the industrial field, several types of complex systems can be found. For example hybrid systems, in which the system behavior is defined by a set of discrete modes. In each of these modes, a different set of continuous dynamics

governs the system behavior [15]. Another type of complex systems is systems of systems. These systems consist of numerous sub-systems which are themselves complex and interact between them.

Considering the emergence of complex systems, safety analysis of these systems is becoming a very important topic. However these systems are usually difficult to describe, understand, predict, manage and design, because different design teams collaborate using different modeling languages, at different abstraction levels, which creates a gap between these modeling languages and the specifications of the system under study. Therefore in order to manage this complexity, it is necessary to use a common modeling language by developing a formalized application of modeling called Model-Based Safety Assessment (MBSA) [69]. MBSA is a process which provides the framework to allow the systems engineering teams to unify activities related to system of systems architecture development. The goal of MBSA is to optimize safety by the identification of safety related risks, eliminating or controlling them by design or procedures, based on acceptable system safety precedence. Several attributes related to systems safety can be quantified such as reliability, availability, maintainability and security.

Our research works deal mainly with safety assessment of a special complex industrial system, namely the Data Center system. This kind of system is constantly growing in the field of Information Technology (IT), because of the rapid increase of computing and communication capabilities offered by companies which host Data Centers, such as social networking, e-commerce and cloud computing [77]. Thus, ensuring a continuous service by avoiding downtime is becoming a competitive factor among these companies.

A Data Center (DC) is a building whose purpose is to host IT devices to provide different internet services [89]. It is the physical materialization of the internet and the cloud. The devices within the DC have three essential needs: a physical space, industrial power energy and a constant supply of cold air. So a DC is a system of systems which includes 3 different sub-systems: electrical, thermal and network. Physical space is a place where different IT devices are located. Their interconnections form an important telecommunication network. To ensure constant

operation of these devices, energy is provided by the electrical system, and to keep them at a constant temperature, a cooling system is necessary. Each of these needs must be ensured continuously, because the consequence of breakdown of one of them leads to the unavailability of the whole DC system, and this can be fatal for a company. For example, an electrical breakdown of 10 seconds can cause service outage of 10h, and 1 minute of interruption can cost more than $7000 [22]. DCs are therefore built to meet strong constraints of continuity of service. These constraints can represent 50% of the DC cost, that is, several billion euros [22].

There are many reasons for a DC downtime. The power interruption and hardware failures are the major causes. Power interruption is caused by the electrical power system inadequacy to provide sufficient energy to the telecommunication devices due to the failure of its components. The hardware failures come in many forms and can be attributed to several causes such as component quality issues, human intervention or temperature variation within the DC room. Usually, operating at temperatures higher than typical working conditions can have a negative impact on the reliability of electronics components. Indeed when the temperature rises above the recommended 20 to 25 °Celsius range, the hardware inside IT components may fail more frequently [11]. Therefore the cooling system of a DC may also impact the availability of the services provided by the network system.

Given the major challenges of the DCs, from an economic and societal point of views, and with their increase of size and complexity, their safety analysis become more and more crucial. Thus in this thesis works, we propose a tool-supported model-based methodology to analyze both safety and performances of the DC system. In our Knowledge, there exists no methodology which combines safety and performance studies, taking into account the whole DC system with the different interactions between its sub-systems. This methodology, which is based on Production Trees modeling technique (proposed in [66]), allows assessing different performances and safety indicators.

Production trees formalism allows modeling the relationship between production system components with a particular attention to the production levels of the components located upstream and downstream a production line. It is a modeling

technique for availability analysis of production systems in general [66]. Production trees are similar to Fault Trees (FTs) with their basic components and gates. However, unlike the gates of FTs, the gates of PT are not logical. They allow dealing with production flows upstream and downstream a production line, according to the type of these flows. They serve to permit, inhibit or modify the passage of flows.

A DC system is a production system because it consists of combinations of components that transform entrance supplies (energy for the electrical system, air flow for the thermal one, and packets for the network one) into exits, which can be internet services. Moreover, it is necessary to generate sufficient power energy and cooled air, and transport it to the load points (IT components), taking into account the maximum capacity of each component in the system (production capacity problem). Furthermore, it is necessary to offer a sufficient network capacity (bandwidth) taking into account the maximum capacity of each component in the network. Thus safety analysis of such a system has to deal with a production capacity problem, and traditional techniques in the literature do not address such a problem. For example, Fault Tree Analysis (FTA) and Reliability Block Diagrams (RBD) [25] have convenient graphical representations which is important for industrial models, but as well as they do not deal with the production capacity problem, they take into account neither the functional, nor the temporal dependencies between events occurrences. Consequently, it is not possible to take into account the order in which events occur and events can occur any time, no matter the current state of the system. This problem is partially solved using Dynamic Fault Trees (DFT) [51]. However, the semantics of this approach is not always clear [108] [88]. Moreover, currently there are no effective resolution techniques. The usual techniques for solving FTA being ineffective on DFT, in general, they are automatically converted to Markov Chain (MC) [50] before being solved using standard resolution techniques. Another approach is to map the DFT into a high level language (I/O Interactive Markov chain, PEPA model, ...) [63]. In all these cases, we find ourselves having to address the underlying MC. Other examples of techniques proposed in the literature are Markov chains and Generalized Stochastic Petri Nets (GSPN) [65]. These techniques are very used to represent complex models with

production capacity. They have a convenient graphical representation but this representation becomes unreadable for large scale models and it is difficult to represent propagation of flows. Another major obstacle that these techniques face is the state space explosion problem.

Unlike these techniques, Production Trees are very suitable to describe the behavior of systems with the production capacities' problem.

In addition to the capacity problem, a DC system has a particular behavior, because of the different dependencies between its sub-systems. Each component's failure in a DC sub-system (electrical, thermal and network) can affect the sub-system's itself with a possible effect on the whole DC system. These dependencies mean the presence of several kind of flows in the DC system (air flow, energy flow, and packets flow). However, so far PT modeling technique allows dealing with only one kind of flow at once. Therefore in this thesis works, we introduce a new modeling mechanism in PT formalism allowing to deal with the different types of flows in a DC system, which is currently not possible. This mechanism allows us also to study the DC system as a global entity (system of systems).

Temperature variations within the DC room is also an important factor affecting the DC system safety. Therefore, we enrich our methodology by including an Arrhenius model [4] which relates the lifetime of the DC electronic component to the operating temperature. Thus the PT assessment approach estimates the reliability and availability of the DC system taking into account the temperature within the DC room.

Moreover, as currently PTs models are not tool supported, we propose a new assessment algorithm based on the Probability Distribution of Capacity (PDC) of flows circulating in a DC system. This approach calculates both availability and reliability of the system using a set of predefined formulas. The basic idea of this assessment algorithm of the PTs is inspired from [10]. Instead of identifying all basic sub-systems and combine them after, each gate of the production tree model combines all its entries (children) by applying rules. The defined rules depend on the semantics of the gates and their policies.

Our methodology combines both safety and performance analyses. First we

assess different safety indicators of the whole Data Center system. Then we analyze the performances of the most important part of DC system responsible for providing services, that is, the network sub-system. The main performances metrics are the total network throughput, the mean end-to-end delay and packet loss probabilities. Generally Queueing Network (QN) theory is used to analyze the performances of such systems [18]. However, these techniques do not take into account the system components failure. In these thesis works, the performances are estimated knowing that each component of the network sub-system can fail. For that, we enrich the obtained PT model by introducing performance measures on flows circulating in it.

Finally, our methodology is supported by a graphical tool we have developed. This tool is an EMF-based (Eclipse Modeling Framework) with an interactive graphical interface, which allows creating, editing and analyzing PT models. The tool allows also displaying the results, and comparing these results with those obtained using the high level modeling language AltaRica 3.0. Indeed, in order to validate the results of our approach, we model the DC system using the AltaRica 3.0 modeling language [79] and use its dedicated stochastic simulator. Moreover the performance results of our approach are validated by comparing them to simulation results obtained with a simulation tool we have implemented for an open finite QN where each queue is a $M/M/1/K$ [101].

This thesis is structured as follows:

- Chapter 2 summarizes the different works carried out in the literature to perform the safety analysis of Data Center systems. Many of these research works studied each DC sub-system separately (electrical, thermal, network) in terms of reliability and availability, while little studied the whole DC system with the interactions between its sub-systems.

- Chapter 3 demonstrates the challenging aspects of designing complex systems and provides an overview of safety analysis and its relationship with systems engineering. Then, the main safety analysis techniques, including classical approaches and model-based approaches, are presented. Finally, some related high level modeling languages are discussed.

- Chapter 4 presents an overview of the Data Center global system, its different sub-systems and their different interactions. First we show that the proper functioning of a DC system is based on the continuity of the services provided by the equipments of the network sub-system. Then these equipments must be provided with sufficient power energy by the electrical sub-system, and kept in acceptable temperature by the thermal one.

- Chapter 5 deals with the Production Trees modeling technique. We give first a description of this new modeling technique. Then we propose an extension of the technique, to deal with the dependencies between different kinds of flows. Finally we propose a solution method to assess a PT modeling a system to estimate reliability and availability measures.

- Chapter 6 presents our model-based tool to analyze safety and performances of DC systems. This tool allows editing and visualizing graphically PT models, then in order to assess them, we present the different algorithms implemented to estimate safety indicators.

- Chapter 7 illustrates a case study of a real DC system. First we describe how to model each DC sub-system using production trees formalism. Then we show how our proposed PT extension allows modeling interactions between these sub-systems.

- Chapter 8 investigates the safety and performances of the DC system of our case study. We demonstrate how this technique helps analyzing both reliability and performance of the system. The comparison with the simulation results shows a promising effectiveness of this integrated method.

- Chapter 9 presents conclusions drawn from this work, as well as perspectives for the continuation of these research works.

Chapter 2

Safety Engineering

2.1 Introduction

The objective of this Chapter is to provide an overview of the safety engineering and its relationship with Systems Engineering (SE). Generally the notion of engineering applies known principles to practical ends [38]. Systems engineering is concerned with the design, building, and use of systems composed of concrete entities such as engines, machines, and structures [82]. Without systems engineering we would not have had several systems like space shuttles and aircraft that became part of our daily life. These systems have certainly been well planned or in other words, they have been systems engineered.

Traditionally, systems were designed and validated using manual, documents and techniques [16]. Nowadays, with the explosive growth of technologies, systems are getting more and more complex, and the existing design techniques are not suitable for these kinds of systems [82]. The main drawback of document-based approaches is that a major part of the effort is given to documents management rather than engineering [16]. It is laborious and time-consuming to classify, update and find the needed documents. Thus the application of these techniques are difficult to handle and may increase the risk of encountering unpredictable outcomes which may impact the project cost and schedule. Models are then needed to help managing the system complexity. This led to what we call Model-Based Systems Engineering (MBSE), which is a process for providing the formalized application of

modeling which allows the systems engineering team to be effective and consistent from the start of any project [21]. Basically the MBSE process allows improving the traditional document-based approach and helps engineers of different disciplines (thermal, electrical, software design, cost management, ...) to obtain the design information necessary for the creation of their models.

The most important specialty within systems engineering which supports program risk management is systems safety. It is the application of engineering and management principles, criteria and techniques to optimize safety by identifying safety related risks, and eliminating or controlling them by design and procedures [40]. Several safety analysis techniques and methods are developed for this purpose and widely used in industry [15]. However, these techniques have a major drawback. They rely on too low level modeling formalisms. As a consequence, there is always a gap between the specifications of the system under study and the associated safety models [19]. Thus, considering the added value of models in the field of SE, models are also used in safety analysis, and this led to what we call Model-Based Systems Analysis (MBSA).

This Chapter describes the basic concepts about the main topic of this thesis: safety analysis. Its aim is to provide the basic knowledge necessary to understand the remaining parts of these thesis works. It is organized as follows. Section 2.2 is dedicated to an overview of systems engineering and systems safety. Then, a summary of safety analysis is given in Section 2.3. Section 2.4 provides an overview of the most used safety analysis techniques including classical approaches and model-based approaches. Finally, Section 5 concludes this Chapter.

2.2 System Engineering

According to INCOSE [21], systems engineering is defined as follows:

> Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and

system validation while considering the complete problem.

In the American military standard Mil-Std-499A, 1974 [82], systems engineering is defined as follows:

> The application of scientific and engineering efforts to:
> • Transform an operational need into a description of system performance parameters and a system configuration through the use of an iterative process of definition, synthesis, analysis, design, test and evaluation.
> • Integrate related technical parameters and assure compatibility of all physical, functional and program interfaces in a manner that optimizes the total system definition and design.
> • Integrate reliability, maintainability, safety, survivability, human and other such factors into the total technical engineering effort to meet cost, schedule and technical performance objectives.

Thus systems engineering begins by identifying the system users' needs and the stakeholders to assure that the right problem is being addressed by the system. It allows defining the system according to the needs, identifying the functions that meet these needs, allocating these functions to the different system entities (components) and finally confirming that the system performs as designed and satisfies the users' needs.

In the SE process there are a technical and a management process [40]. The technical process addresses the design and the implementation efforts necessary to transform the operational needs into a system. Along the way, it produces the documentation necessary to implement, operate, and maintain the system. The management process supports the technical process by planning, assessing risks, integrating the various engineering specialties and design groups, maintaining configuration control, and continuously auditing the effort to ensure that cost, schedule, and technical performance objectives are satisfied. Together, the management and technical processes create the systems that will meet the customers needs. To be

effective in all these areas, systems engineering must, therefore, provide an organized, repeatable, iterative, and convergent approach to develop systems [15]. The approach must be *organized*, because without an organized approach the details of the system under development will be misunderstood. The approach must be *iterative* and *convergent*, which means the engineering processes are repeated at each level of system design and ensure the convergence of the development process to a solution.

However with the industrial growth, systems become more and more complex. The documentation produced during the SE process will be difficult to handle, and this may increase the risk of encountering unpredictable outcomes which may impact the project cost and schedule. Moreover, increasing the system complexity produces a large amounts of information from different design teams, which will be difficult to manage and share between system designers. Thus the main characteristic of complex systems is that their behavior cannot be thoroughly planned, understood and anticipated [15]. This makes the use of models necessary during the design. Models are more expressive to describe systems and more likely to be understood in an unambiguous way than document-based descriptions. For this reason, systems' engineers prefer the use of models [16].

2.2.1 System Modeling

Models are common to human experience in order to understand the way the world works. Childrens toys are simple models of the world around them. They help children to link imagination (an abstract representation) to a real object. Thus a model is a physical representation of an abstract idea [15].

There are four elements of such a model: language, structure, argumentation, and presentation [71].

- **Language:** the model is seen in terms of language. The System Definition Language (SDL) [44] expresses and represents the model clearly. This is critical to successful system design. The system definition language must be clear and unambiguous in order to depict the model accurately.

- **Structure:** the model must have a structure allowing to capture the system behavior by describing the relationships between its entities.
- **Argumentation:** the purpose of the model is to represent the system's design allowing the design's teams to demonstrate that the system accomplishes the purposes for which it is designed.
- **Presentation:** the system must not only be capable of making arguments, but also include mechanisms showing the arguments in order to be seen and understood.

In the world of systems engineering, these elements (language, structure, argumentation, and presentation), which form a model, present a clear and coherent design which helps to develop, test and deploy the system. Therefore including models in systems engineering has led to Model-Based Systems Engineering.

2.2.2 Model-Based Systems Engineering

Model-Based Systems Engineering (MBSE) is fundamentally a thought process [102]. It provides the framework to allow the systems engineering team to be effective and consistent from the start of any project. The INCOSE Systems Engineering Vision 2025 [31] defines MBSE as:

> the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.

MBSE enhances the ability to capture, analyze, share and manage the information associated with the specifications of a product. The system model is a primary artifact of the SE process. MBSE formalizes the application of SE through the use of models.

MBSE is applied in order to manage systems' complexity, because today systems are becoming more and more complex, and it is important to deal with this

complexity. MBSE with its graphical models improves collaboration and communication between the teams. For example, software engineers collaborate with hardware engineers or system engineers focusing on requirements of finance. It helps also to preserve historical information during the SE's process. Thus the existing models can be reused which is precious for an enterprise. Moreover, it is very easy to hide some issues in design in textual description. For example, in a state machine with a deadlock state, the problem is easily identified.

2.3 System Safety

As noted previously in this Chapter, systems engineering is an integral part of the management of a system development project. At every step of this development, unpredictable outcomes can be encountered that pose risks of unacceptable consequences that may require a program change which impacts project cost and schedule. One of the greatest challenges in SE is risk management in order to satisfy performance and safety requirements. Thus systems safety is a very important specialty within SE that supports risk management.

Systems safety emerged as a necessity during the XXth century particularly with the industrial revolution [82]. This term appeared in an advertisement on Dodge Brothers engines in the 1930s [102]. Systems safety is a field of activity that offers ways to increase the reliability of systems in a timely and cost-effective manner. It is often called science of failures [15]. It includes failures' knowledge, evaluation, forecasting, measurement and control. This is a transverse domain that requires a global knowledge of the system such as its conditions of use, the external risks, the functional and material architectures, the structure and the aging of the materials. The goal of systems safety is to achieve the Grail of system design: no accidents, no shutdowns, no failures (and even no maintenance), using quantified attributes called systems safety parameters.

2.3.1 System safety parameters

The systems safety's objective is to estimate a quantified attributes of the system under study such as reliability, availability, maintainability and security. These attributes allow defining the expected objectives of the system and evaluate the quality of the service delivered by this system, in order to target the critical points to be improved. They are defined as follows:

- **Reliability:** noted $R(t)$, it is the probability that a component or system will perform its intended function with no failures for a given period of time $[0, t[$ (mission time) when used under specific operating conditions (test environment or operating environment). It is also defined as:

  $R(t) = P(\text{system is working in } [0, t[)$

- **Availability:** noted $A(t)$, it is the probability that a repairable system will perform its intended function at a given time under specific operating environment. It is also defined as:

  $A(t) = P(\text{system is working at instant } t)$

- **Maintainability:** noted $M(t)$, it is the probability that a failed item will be restored or repaired within a specified period of time $[0, t[$. It is also defined as:

  $M(t) = P(\text{system is repaired at instant } t)$

- **Security:** noted $S(t)$, it is the ability of a system to avoid critical or catastrophic events over a period of time $[0, t[$. It is also defined as:

  $S(t) = P(\text{system without catastrophic events in } [0, t[)$

The parameter defining systems reliability, availability and maintainability is the failure rate (λ) [20]. It is a value which represents the characteristic of breakdown occurrence frequency. There are some common basic categories of failure rates:

- **Mean Time To Failure (MTTF):** it is one of basic measures of reliability for non-repairable systems. This statistical value is defined as the average time

expected until the first failure of a component. MTTF can be calculated as the failure rate inverse $1/\lambda$. For repairable systems, MTTF is the anticipated time period from repair to the first or next breakdown.

- **Mean Time To Repair (MTTR):** it is the total time spent to perform all corrective or preventive maintenance repairs divided by the total number of repairs. It is the anticipated time period from a failure to a repair or maintenance. It is used only for repairable systems.

These four parameters (Reliability, Availability, Maintainability and Security) are dependent on each others as shown in Figure 2.1. Firstly, decreasing the reliability of the system leads to a decrease of the availability due to the presence of many failures, and has an impact on the system security level (a failure that can lead the system to a dangerous state). Moreover, inadequate maintainability (in the case of repairable systems) can affect the system availability due to the increased number of failures, and also the system safety. Finally, increasing the security level of the system by adding multiple security elements, reduces its availability because the system stops unexpectedly at the first failure. While increasing the level of availability is achieved at the expense of the security level.

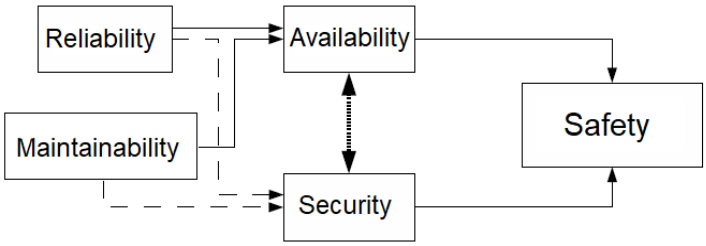

Figure 2.1: Dependencies between Safety parameters

One common way to improve the system safety is redundancy. The principle is to "over-size" the system in order to tolerate certain failures that are known to be frequent or dangerous. There are two types of redundancy: the hardware redundancy where several components can perform the same function and therefore provide the same service, and the functional redundancy where several technical solutions can ensure the same service. These two types of redundancy can be implemented in two

manners: hot redundancy where all solutions operate permanently in parallel, and cold redundancy where the system activates redundant solutions and reconfigures its state after a hardware or a functional failure.

2.3.2 System's Safety Analysis

In order to assess the safety of a system, it is important to carry out analyzes in order to characterize the system dysfunctional behavior. This process can be divided into two steps: Preliminary Risk Analysis (PRA) [14] and Dysfunctional Behavior Analysis (DBA) [17].

The objective of the PRA is to formalize the knowledge about the failure of the system components. The most commonly used method is FMECA (Failure Mode, Effects, and Criticality Analysis). This method allows identifying the different failure modes of each component, as well as their impact on the component and its environment. In parallel with this analysis, it is necessary to quantify the probabilities of occurrences of events responsible for system failure. This is generally estimated through experience feedback.

DBA allows exploring the knowledge gathered by PRA, in order to understand the dysfunctional behavior of the system under study. From a *qualitative* point of view, it is a question of identifying the different scenarios of events leading to the system failure. From a *quantitative* point of view, it is the estimation of safety parameters (reliability, availability, ...).

- **Qualitative Analysis:** it consists in extracting from the model the shortest sequence of events leading to the failure of the system [102].

  Boolean Algebra is widely used in this context. It consists in mapping the system behavior into a Boolean expression by determining the logical relationships between the combinations of basic events leading to the top event. This formulation can then be refined into a minimal canonical form, allowing to extract all the sets containing events which cause the system failure. A set is minimal if the system fails when removing any event from this set.

  This approach is very strong mathematically, but unfortunately not very op-

erational. Indeed, the size of the algebraic expressions increases significantly with the system size, and the extraction of the minimal sets becomes difficult.

- **Quantitative Analysis:** it consists in giving the structure of the system as well as the failure probability of basic events, in order to evaluate the failure probability of the complete system. A system can fail if it is observed over a period of time. It is assumed that the system starts at time $t = 0$ and has only one failure mode. The component operates over a random period of time and then it fails. In the case of repairable system, the component stays in a failure mode for a random repair time.

2.4 Safety Analysis Techniques

Safety analysis techniques and methods have the objective to assess the system safety during the design phase and ensure that the designed systems have a satisfactory safety level. The safety analysis proceeds in three steps as illustrated in Figure 2.2.

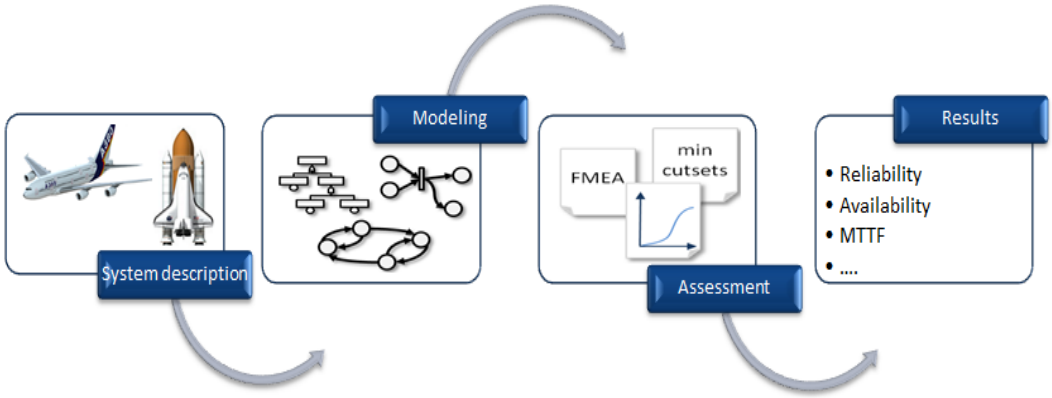

Figure 2.2: Safety Assessment

- The first step is modeling. It consists in creating an appropriate safety model of the system under study. The choice of the formalism depends on the system whether it is static or dynamic, as it will be detailed in the following sections.

- The second step consists in solving the created model in order to estimate the failure scenarios and probabilistic indicators.

- The third step consists in studying the obtained results by analyzing if the system satisfies the given safety requirements.

The modeling step is indispensable for safety analysis. The modeling approaches can be classified into two categories: classical and model-based approaches, as shown in Figure 2.3

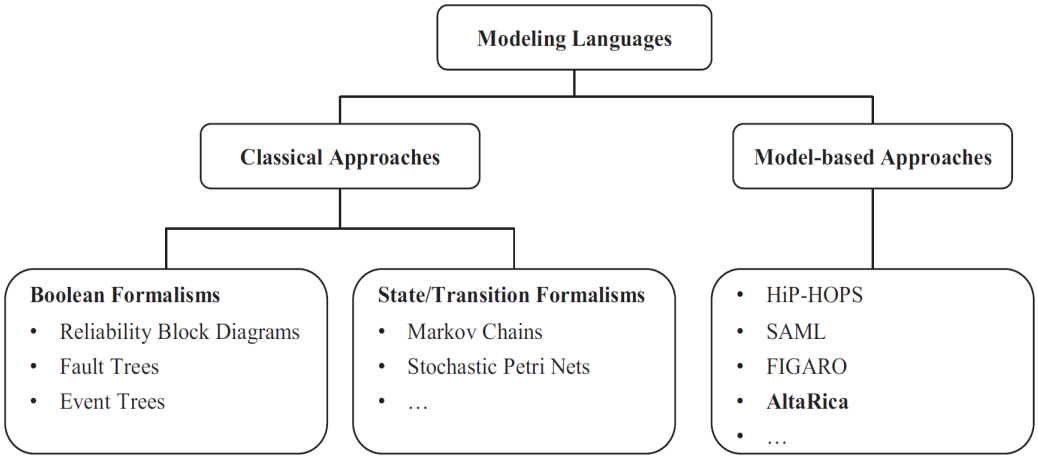

Figure 2.3: Categories of modeling formalisms [112]

2.4.1 Classical approaches

Classical approaches are graphical and event-based. They have a convenient graphical representation which is important for industrial models. They are classified in two categories: Boolean formalisms and States/Transitions formalisms [25].

2.4.1.1 Boolean formalisms

Boolean formalisms look at the system components, critical events, and system characteristics. They describe static links between the elementary failures and the system failure. This class of formalisms include Event Trees (ET) [105], Fault Trees (FT) [80], and Reliability Block Diagrams (RBD) [80]. The mostly-used in safety studies of industrial systems are FTs and RBDs.

2.4.1.1.1 Fault Trees

Fault Trees (FTs) are the safety models the most used in the literature. Their graphical representation describes the relationships between the failure events of the modeled system. The root of the FT, noted *top event*, represents the global system failure (the undesirable event). The leaves of the FT, noted *basic events*, represent the failures of the individual components. FTs are constructed as a logical illustration of how lower level events (basic events) combine together to result in the upper level (top event) through logical gates. The FTs formalisms are widely used because of their simplicity. However, their logic is purely combinatorial (AND, OR, and K / N Boolean gates), which does not allow representing dependencies between basic events. An extension has been proposed in the literature, in order to model the dependency aspects problem, called Dynamic Fault Trees (DFTs) [51].

The three gates most commonly used in DFTs are the *PAND* (Priority AND), *SPARE* and *FDEP* (Functional DEPendency) gates. The *PAND* gate was introduced in [52] to add an order of occurrence constraint to the inputs of an *AND* gate. The authors in [76] define the *SPARE* gate to model a redundancy between system components. Finally, the *FDEP* gate is used to model the correlation between several events [52]. However, whatever the number of dynamic gates added in the formalism, the behavior model remains limited because the semantics of these gates is not always clear.

Let's consider the example of the processing system depicted in Figure 2.4. The system consists of two processors, *Processor 1* and *Processor 2*. P1 and P2 are their input components, respectively. Initially P1 and P2 are working, and both have one spare part, S1 and S2, respectively. The global system fails if *Processor 1* breaks down first and then *Processor 2* fails. The system is not repairable, and we note λ_{Pi} and λ_{Si}, $i = 1, 2$, the failure rate of component P_i and S_i, respectively.

In classical FT such a behavior is modeled using an AND gate. However in this example, the failure order of *Processor 1* and *Processor 2* is important and classical FTs are inadequate. So this failure order of processors is modeled using Dynamic FTs with a *PAND* gate. The failure of *Processor 1* and *Processor 2* depends on their

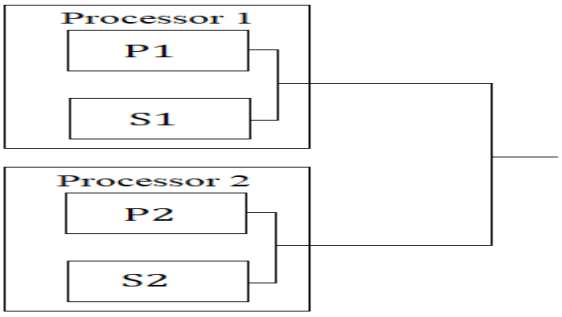

Figure 2.4: Example of a processing system

respective input components (P1 and P2). If P1 or P2 fails, S1 or S2, respectively, will take over. This is modeled using two *SPARE* gates, one between P1 and S1, and one between P2 and S2. Figure 2.5 shows the DFT modeling the processing system.

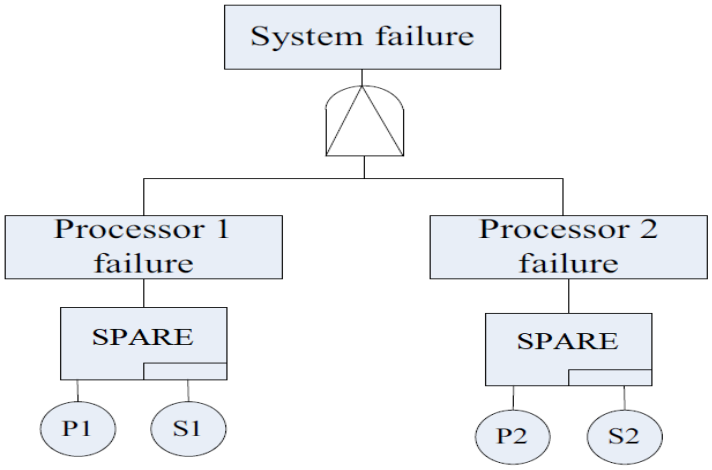

Figure 2.5: DFT modeling the processing system

2.4.1.1.2 Reliability Block Diagrams

Reliability Block Diagrams (RBD) are the most basic and intuitive safety models [80]. RBD have been widely used due to their simplicity and are one of the most practical reliability modeling tools. An RBD model consists of blocks behaving like switches connected in series or in parallel to connect an input to an output. Each block represents a component of the system. When a component fails, the corresponding block is removed from the diagram. The whole system is operational if

there is at least one path between the input and the output. As for FT models, these models are suitable to represent static systems. In particular, they have a convenient graphical representation which allows visualizing the different sections of the system, that is, the component failure combinations sufficient to cause the system failure. However, they are not adapted to take into account dynamic aspects, in their original version. This problem is partially solved by Dynamic Reliability Bloc Diagrams (DRBDs) [92]. DRBDs are obtained by extending RBD with new constructs that allow the modeling of dynamic behaviors and dependencies between the components of the system. As an example, the State Dependency (SDEP) bloc allows modeling the dependencies between system components [92]. Thus, some dynamic aspects can be modeled. However, the use of this formalism in the literature is rare, because the semantics of the additional blocs is not always clear. Figure 2.6 shows the DRBD modeling the processing system depicted in Figure 2.4. The parallel configuration between the two DRBD blocks (*Processor 1* and *Processor 2*) corresponds to the PAND gate of DFT. The redundancy is modeled using the *Spare* block between redundant components.

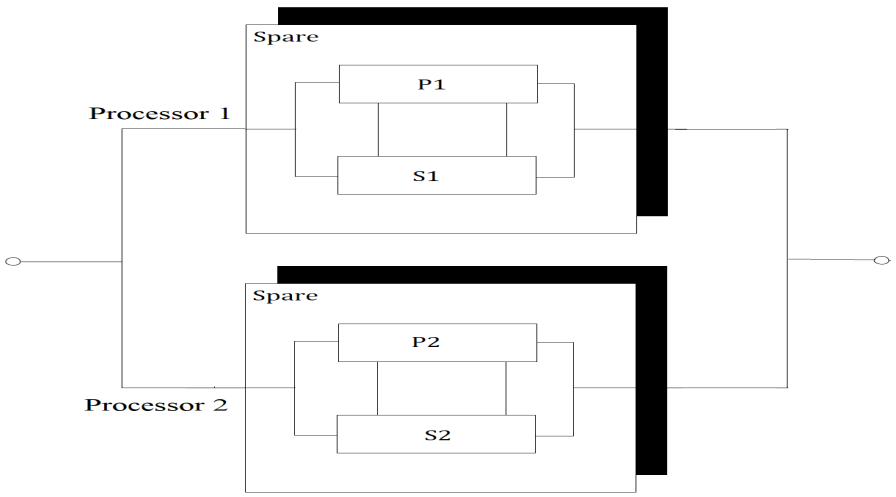

Figure 2.6: DRBD modeling the processing system

2.4.1.1.3 Boolean formalisms assessment tools

Usually Reliability Block diagrams can be easily transformed into Fault Trees [80]. Both qualitative and quantitative analysis can be performed on a Fault Tree. The qualitative analysis is assessed by the calculation of minimal cut sets, and the quantitative one is assessed by associating probabilities with each basic event in the tree. The probability of the system failure is the probability of the top event according to the Boolean equations representing the model. Several importance factors (Risk Reduction (RR), Risk Achievement Worth (RAW), Birnbaum Index) can be estimated using fault trees [12].

A lot of commercial RAMS (Reliability, Availability, Maintainability, Safety) workbenches based on either FTs or RBDs are available, for example Aralia Fault Tree Analyzer (Dassault Systemes) [8], FaultTree+ (Isograph) [2], BlockSim (ReliaSoft Corporation) [5], Item Toolkit (ITEM Software) [6], CAFTA (Electric Power Research Institute) [1]. In general, these workbenches include a graphical user interface and assessment tools to calculate minimal cut sets and probabilities of events.

2.4.1.2 State/transition formalisms

The second category of classical formalisms used in safety analysis are state/transition formalisms. They allow modeling dependencies between components redundancies. Many techniques have been proposed in the literature such as Boolean logic Driven Markov Processes [19], Finite State Machines [87] and I/O Markov Chains [11]. However in the following section, we will present only the most commonly used in industry, that is Markov chains and Stochastic Petri Nets (GSPN) [65].

2.4.1.2.1 Markov Chains (MC)

The use of Markov chains to model the behavior of systems is widely used in the literature [98]. For this type of modeling, the system behavior is seen as a stochastic process verifying the Markov property: system evolution depends only on the current state of the system. Markov chains are classified into Continuous Time Markov Chains (CTMC) and Discrete Time Markov Chains (DTMC). The state of a DTMC

is allowed to change only at discrete instants while that of CTMC can change at any time. They have a convenient graphical representation and provide two types of representations: circles which represent system states, and arrows, labeled by rates. In a DTMC these arrows are marked with the corresponding transition probability, while in a CTMC, they are marked with the transition rate corresponding to the failure rate λ or to the repair rate μ of system components. Some states are considered as operational for the system under study, others are considered as failure states. Figure 2.7 illustrates the CTMC representing the processing system of Figure 2.4.

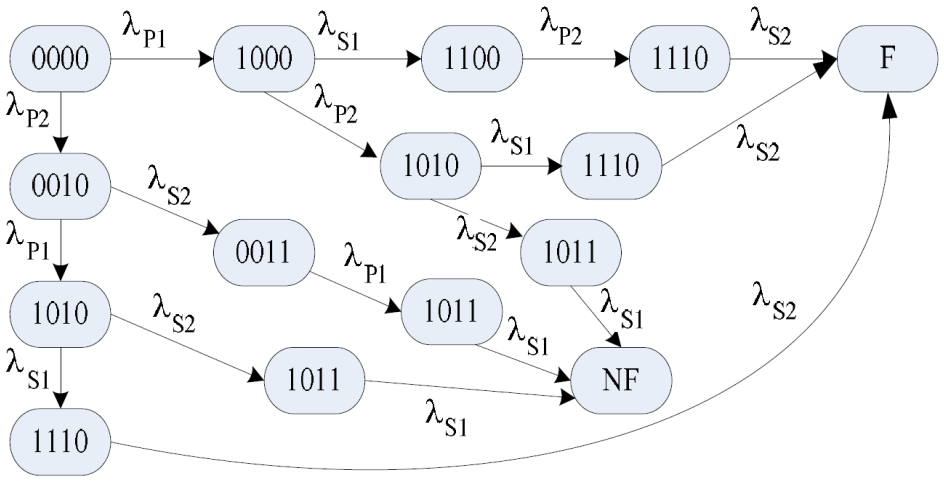

Figure 2.7: Example of a Markov Chain of an unrepairable system

The processing system contains 4 components P1, S1, P2 and S2. Each state of the CTMC is noted $x_1x_2x_3x_4$, where $x_i \in 0, 1$, $i = 1, \ldots, 4$, is the state a component of the system. "0" means the component is working, and "1" failed. State 0000 is the initial state of the system. The CTMC represents all states leading to the failure state F or those leading to no-failure state NF. Note that only components failure rates are associated with transitions because the system is considered unrepairable.

Markov chains remain a modeling method with a strong precision compared to the other formalisms. However, the major limitation of this formalism is the state space explosion when dealing with large systems. As a result, Markov chains are either used to model a specific part of a system, or generated automatically from another model built using a high level modeling language, such as GSPN, in order to facilitate quantitative analysis.

2.4.1.2.2 Generalized Stochastic Petri Nets (GSPN)

Petri Nets can be seen as an abstraction of Markov chains [13]. Places (circles) can be interpreted as system states and transitions are often associated with events. A stochastic Petri net is a Petri net for which the occurrence of transitions is associated with a stochastic delay. Moreover, transitions may be immediate or timed. When several immediate transitions are fireable at time t, the choice is done according to the probability associated with each fireable transition. Figure 2.8 illustrates a Petri net representing the processing system example of Figure 2.4.

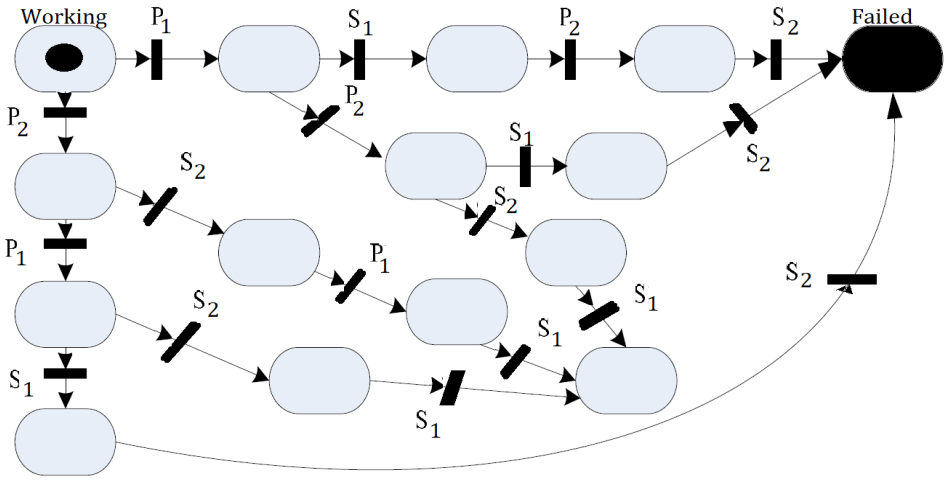

Figure 2.8: Example of a GSPN of an unrepairable system

Like Markov chains, SPNs allow modeling the behavior of a system taking into account the dynamic aspects and the dependencies between the system components. However, the models become difficult to build, when the system is complex, and their analysis requires a high computational cost due to the state space explosion.

2.4.1.2.3 State/Transitions formalisms assessment tools

For these formalisms, the quantitative analysis is assessed by converting the SPN into a Markov chain and solving directly the obtained Markov chain, or using stochastic simulation. Regarding the qualitative analysis, it is not easy to define the sequence of events leading to a critical state. An approximation of most probable sequences can be estimated by simulation.

Total has developed GRIF [3], a system analysis software platform used to calculate system reliability, availability, performance and safety indicators. It includes several graphical modeling modules such as Markov chains and Generalized Stochastic Petri Nets, and a Monte-Carlo simulation engine to assess these models. ITEM [6] developed a software based on Markov chains to estimate systems availability, reliability and maintainability.

2.4.2 Model-Based approaches

As explained in Section 2.2, systems engineers prefer the use of models to design systems, and this led to Model-Based Systems Engineering (MBSE). Thus, to facilitate the integration of safety activities into the design processes, it is important to integrate them to the model-based engineering process. Moreover, the classical approaches presented (Boolean and state/transition formalisms) have reached their limits. They are very close to mathematical equations, and thus there is a distance between system specifications and the models representing the system behavior. Therefore in order to reduce the distance between systems specifications and the associated safety models, Model-Based Safety Assessment (MBSA) is used [91]. It is an approach in which system engineers and safety engineers share models built according to a process of common development.

The basic idea is to write models using high level modeling formalisms so as to keep them close to the functional and physical architecture of the system [91]. The high level model can be assessed directly or by compiling it into a low level model, such as Fault Tree or Markov chain.

Writing models in high level modeling language allows obtaining models which are structurally close to models designed by other system engineering disciplines. This is very important in order to integrate safety analysis with system design processes and thus to join MBSA to MBSE. In the following section, we discuss different properties that a high level modeling language should have for safety analysis.

2.4.2.1 MBSA properties

The modeling activity needs to be supported by concepts, methods and tools. Thus to design a model, a modeling language should have some properties. Firstly, a high level modeling language must be formal and well defined. This helps the compilation step, when needed, to have a low level formalism, such as Fault Tree or Markov chain. Secondly, it must combine the advantages of both Boolean and States/Transitions formalisms, since it is assumed to be able to represent dynamic models, and describe a system as a hierarchy of its components. Finally, a high level modeling language must be capitalized in order to preserve information about the system under study, which can help to reuse these information in the future if needed.

In the following section, we present different high level modeling formalisms that allow performing Model-Based Safety Assessment.

2.4.2.2 Hip HOP

A Hip HOP model [81] (Hierarchically Performed Hazard Origin and Propagation Studies) can be seen like a RBD with a set of interconnected blocks. The main difference is that links between blocks are carriers of flows (packets, liquid, electric, ...) which allow propagating specific effects of certain failure modes. Thus, this approach is well suited to perform system performance analyzes. It helps generating automatically Fault Trees and FMECA tables. In addition, models can be imported from different modeling tools: Matlab/SIMULINK, Eclipse-based UML tools or SimulationX [81].

2.4.2.3 FIGARO

Developed by EDF, Figaro [39] is a graphical modeling language dedicated to safety assessment of complex systems. It is used as a description language to create knowledge bases such as libraries of reusable components for KB3 [39], a workbench developed by EDF to automatically perform systems safety assessment. Unlike Hip HOP models, KB3 includes Monte-Carlo simulation, Markov chain generation and quantification and generation of critical sequences.

2.4.2.4 SAML

SAML (Safety Analysis Modeling Language) [59], is a formal language expressed in terms of finite stochastic state automata. Automata are all executed in discrete time steps with parallel composition. The semantics of a SAML model is defined as Markov decision process. S3E [59] is a design and verification environment focused on SAML models. It provides a model editor and model analysis tools: a stepwise simulator and translator to the input languages of the probabilistic model checker PRISM and the symbolic model checker NuSMV.

2.4.2.5 AltaRica 3.0

AltaRica [79] is a high level modeling language which allows designing the model of the system under study with a structure that is close to the functional and the physical architecture of the system. AltaRica 3.0 implements the prototype-oriented paradigm [14]. This paradigm fits well with the level of abstraction reliability and safety analysis standards. As for mathematical foundations, AltaRica 3.0 is based on Guarded Transition Systems (GTS) [90]. A GTS is an automaton where states are represented by variables and state changes are represented by transitions triggered by events. Each event is associated with a cumulative probability distribution of its delay. Variables are separated into two groups: states variables whose values are modified only in the actions of transitions and flow variables that represent flows circulating through the system. It is also possible to synchronize events in order to describe remote interactions between components of the system under study. The semantics of GTS is similar to the one of GSPN [90]. Basic components are represented by means of classes. Classes are GTSs that contain variables, events, transitions, and everything necessary to describe their behavior.

2.5 Conclusion

In this Chapter, we presented the main concepts that will be tackled in this work. First, we illustrated the need of systems engineering, and more precisely MBSE,

to design and validate complex systems. These systems complexity makes the assessment of their safety very important. Thus we gave the basic notions of safety and reliability studies. An overview of classical formalisms used to perform Safety Analysis is also highlighted in this Chapter. Finally, we discussed the model-based approach for safety assessment and presented some related high level modeling languages.

In the next Chapter, we will present the notion of Data Center before analyzing their safety. Such systems are complex because they are systems of sub-systems with different interactions between their sub-systems.

Chapter 3

Data Centers

3.1 Introduction

In recent years, the new technologies, such as social networking, e-commerce and cloud computing, are constantly growing. The companies offering these online services become competitive and this has led to a rapid increase of computing and communication capabilities provided by Data Centers [96].

A Data Center (DC) is a large cluster of computers (telecommunication devices) that is owned and operated by an organization. These devices are responsible for providing various internet and cloud services, with the goal of optimizing both costs and performances (see an example in Figure 3.1).

There are many different types of DCs built for a lot of different applications. These DCs are categorized according to their sizes [23]. Small size DCs employ a hundred racks (set of servers), and are normally used for smaller businesses, like experimentation facilities. Mid size DCs employ between a hundred to a thousand racks, and are typically used for medium size businesses such as banks and companies. Finally, there are large size DCs which host more than a thousand of racks, and are used by huge corporations like Microsoft, Google and Facebook. An example of Facebooks DC [53] located in Clonee (Ireland) is illustrated in Figure 3.2.

The infrastructure within a DC can be structured into three main parts, namely:

- The network or Information Technology (IT) system which consists of racks

Figure 3.1: Example of a DC

Figure 3.2: Aerial view of the Facebook DC [53]

(containing servers, switches and routers) placed in the main DC room.

- The cooling or thermal system which is a room containing equipments responsible for producing cooled air in order to keep the IT room in a constant temperature. This room is connected to a kind of fan, called CRAC (Computer

Room Air Conditioning) unit, used for air blowing within the IT room.

- The electrical power system which consists of different electric materials (Uninterrupted Power Systems (UPS), switches, power generators, Power Distribution Units (PDU)) used for supplying the IT and cooling systems with power.

A DC contains also a control room used for monitoring the functioning of the DC, for example the power distribution, the energy consumption and the temperature inside the IT room. If the fire fighting system detects a fire, it uses gas to suppress the fire without damaging electronic devices. Figure 3.3 gives a brief description of the main parts of a DC.

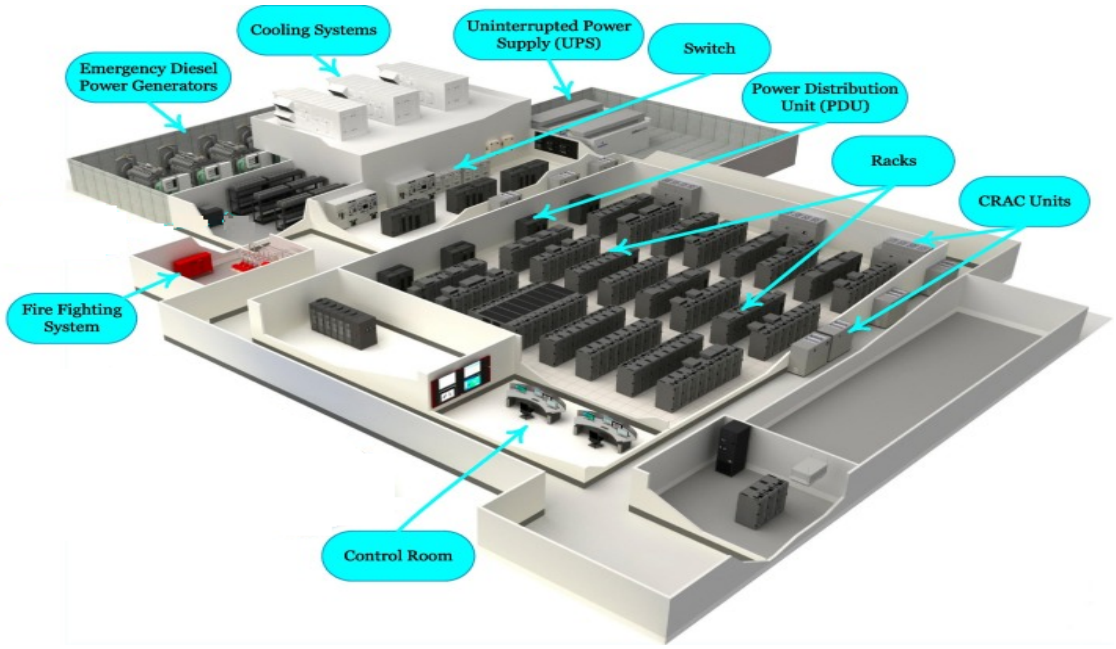

Figure 3.3: The main parts of a DC

The proper functioning of a DC is based on the continuity of the services provided by the equipments of the network sub-system. In order to ensure a constant service, these equipments must be provided with a sufficient and continuous power energy by the electrical sub-system, and kept in a constant and acceptable temperature by the thermal one. The electrical sub-system provides energy to both the network and the cooling sub-systems. Thus the network sub-system depends on

both the electrical sub-system and the cooling sub-system, which itself depends on the electrical sub-system to operate properly (see Figure 3.4).

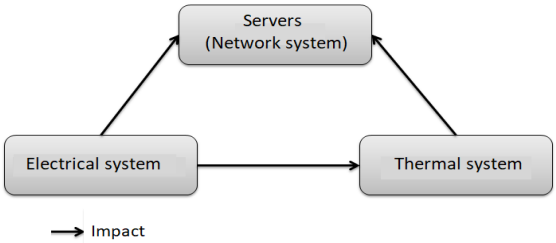

Figure 3.4: Dependencies between sub-systems of a DC system

This Chapter is organized as follows. First, an overview of the main DC entity, which is the network sub-system, responsible for delivering online services, is described in Section 3.2. Then, the electrical one is illustrated in Section 3.3. Section 3.4 provides an overview of the thermal sub-system. Finally, the Chapter is concluded in Section 3.5.

3.2 Data Center's Network System

The physical implementation of a Data Center relies on a stable architectural organization, to guarantee its constant functioning. This architecture is based on the network connectivity and its structure, where connectivity could be at the physical level or at the logical level (link-level). At the physical level a network topology could be hierarchical, non-hierarchical, wireless, wired or hybrid. At the link level, a network topology can be built either on top of a hierarchical interconnection structure or an arbitrary structure. As a result, the DC's network architecture is classified in 14 different topologies as illustrated in Figure 3.5 [23].

Hierarchical network topologies adopt the *Clos* and *Hypercube* topologies [60] to build a hierarchical structure for the network. The Clos topology can deliver high bandwidth using Ethernet commodity switches and routers. These topologies can efficiently lower the cost of building networks. However, they suffer from performance bottlenecks due to oversubscription at the higher layers of the hierarchy, which means allowing many incoming flows to share the same output ports band-

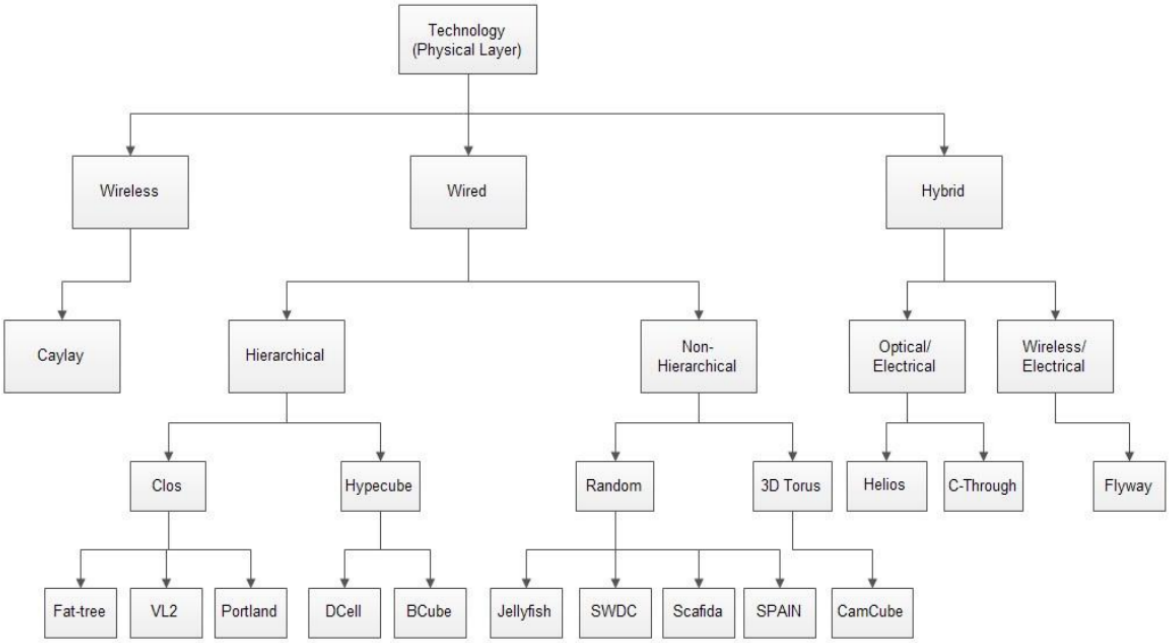

Figure 3.5: Data Center Network System Classification

width resulting in higher latency [60]. Using more expensive devices at the upper layers to provide more bandwidth might solve this problem. The *Fat-tree* is an example of hierarchical topology shown in Figure 3.6.

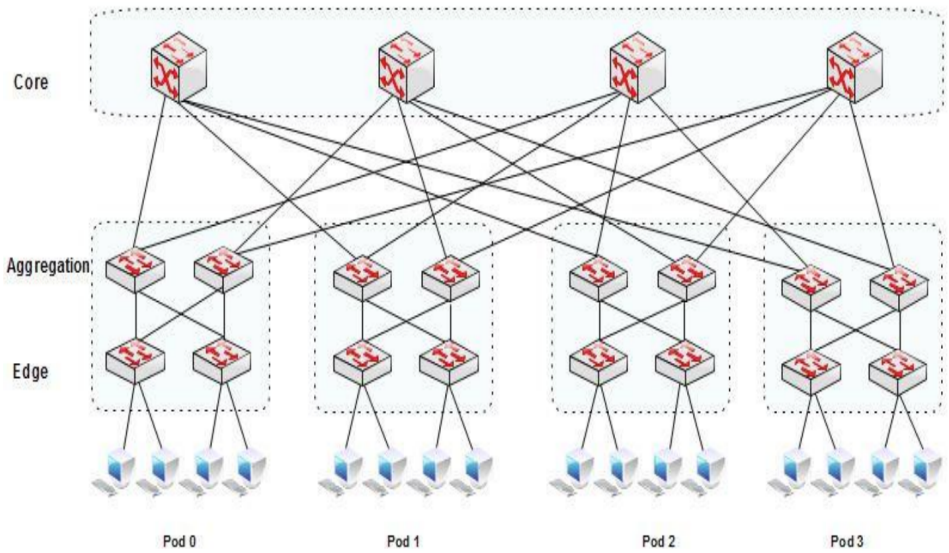

Figure 3.6: Data Center fat-tree topology

The Hypercube topology reduces path length and improves bandwidths. The

interconnection networks in this class depend on numerous commodity switches connected using complex network patterns [61]. They allow expansion to certain degrees, but require servers with free ports especially reserved for future expansions. However the hypercube topology requires complex cabling which is hard to maintain.

Due to limitations in the hierarchical topologies, researchers moved to other ways (non-hierarchical) to interconnect networks more effectively. In *Random structures*, hierarchical switches are avoided, and the network is wired randomly to connect nodes. These structures are inspired by other networks like *small world* and *scale-free* networks [23]. The developers of these topologies have adopted solutions from these networks to overcome incremental expansion problems in DC's networks. Many proposals of architectures were proposed in the literature such as, *Jellyfish* [58], *Small World Data Centers* [96], *Scafida* [104] and *SPAIN* [109].

Finally *Hybrid* topologies combine the advantages of wireless and optical networks. For example, an optical circuit can hold a very large bandwidth over the packet switching technology. Many proposals of architectures have been introduced such as *C-Through* [54] and *Helios* [60].

Let's consider an example of a fat-tree topology illustrated in Figure 3.7. The network consists of M racks, $Rack_i$, $i = 1, \ldots, M$, containing a certain number of interconnected servers through M Top of Rack ToR_i, $i = 1, \ldots, M$, which in turn are connected to two Aggregation switches $AggS_A$ and $AggS_B$, for redundancy. Each redundant pair of $AggS$ aggregates traffic from $ToRs$, which is then forwarded to two routers R_A and R_B. These routers route the traffic to the external network and internet.

In order, for each component in the network, to ensure its function and route the traffic to other components, it has to be powered by the electrical system. The following section details the Data Center electrical sub-system.

3.3 Data Center's Electrical System

An electrical power system is designed to deliver power to customer loads [62]. It is a complex system consisting of components such as Power Distribution Units (PDU),

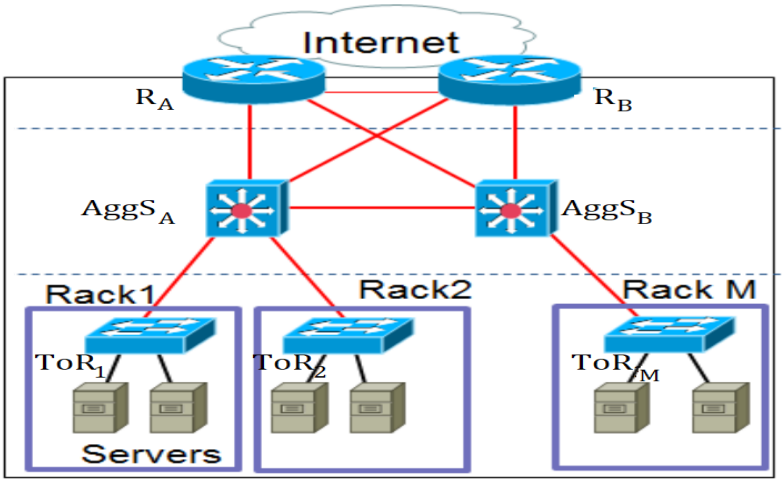

Figure 3.7: An exmaple of a fat-tree Data Center network topology

switches, backup batteries, generators and transformers. To ensure a maximum availability, a large number of power system components are doubled to ensure continuity of service.

Figure 3.8 shows the components of a typical DC power system. Power enters first at a utility substation (transformer) which transforms high voltage (typically 110 kV and above) to medium voltage (typically less than 50 kV) [62]. Medium voltage is used for the distribution to the distribution panels. From here, the power enters the building with the low-voltage lines going to the uninterrupted power supply (UPS) systems. The UPS switchgear will also take a second feed (at the same voltage) from a set of diesel generators that will cut in when utility power fails. Therefore, the output lines from the UPS system are finally routed to the DC floor where they are connected to Power Distribution Units (PDUs).

Important supply systems are also needed to compensate for possible power interruption. Thus several systems take turns in case of cutoff [62], in this order:

- Batteries also called UPS (Uninterruptible Power Supply) are used to smooth the current. They provide power to other components in the system during the first few seconds of a breakdown.

- After a few seconds, generators take over. They are fueled by oil tanks that provide several hours or even days of autonomy. In the case of a prolonged shutdown, the tanks can be refueled by truck to ensure a power of 10 MW for

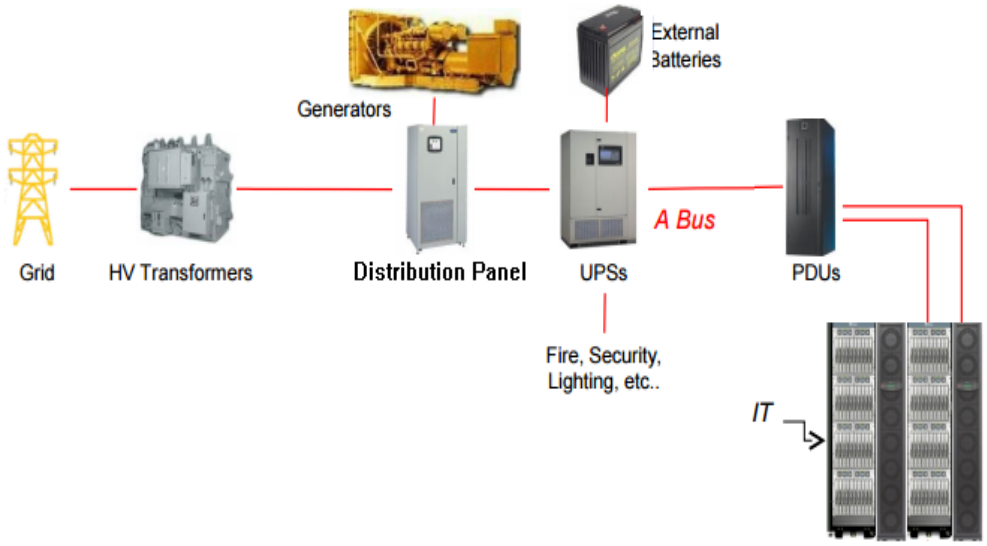

Figure 3.8: The main components of a typical DC power system

48 hours with a generator set with a 45% efficiency. About 100000 liters of fuel oil are needed to power the generators, that is, 9 to 10 19-tonne tankers with a capacity of 15m3 [62].

- If the generators fail, an emergency power line is used to power the DC. The use of this line is much more expensive for the operator, hence it is used as a last resort [62].

- The output of the UPS is then routed to the Power Distribution Units (PDUs) that are attached to the Data Center floor. PDUs look like breakers in homes. They take a very high voltage and divide it into many circuits of 110 or 220 V to supply the servers. Figure 3.9 illustrates the different electrical components that are interconnected to provide power energy.

The Uptime Institute [100] has created a Data Center electrical system classification standard to systematically evaluate various installations, in terms of performance and availability. This classification is in tiers, and each tier corresponds to different levels of equipments and availability. The design of a Data Center is often classified as belonging to Tier I-IV [100].

- Tier I Data Centers have a single path for power distribution, UPS, and cooling

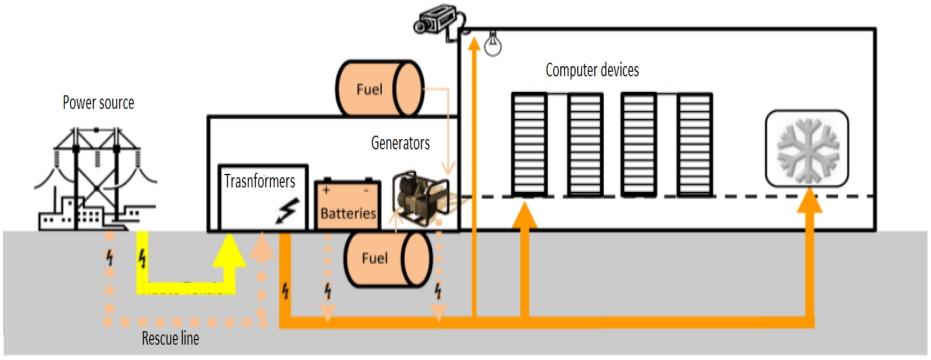

Figure 3.9: Relay system in a DC

distribution, without redundant components.

- Tier II adds redundant components to this design (N + 1), improving availability
- Tier III Data Centers have one active and one alternate distribution path for utilities. Each path has redundant components and are concurrently maintainable, that is, they provide redundancy even during maintenance.
- Tier IV Data Centers have two simultaneously active power, redundant components in each path, and are supposed to tolerate any single equipment failure without impacting the load.

To ensure a high system availability, component redundancy is a possible solution (Tier II). However, Tier III and Tier IV are more demanding, in terms of materials and energy, because of redundancy in both components and paths. The redundant path of Tier III is not active (in standby and is activated in case of failure) while the one of Tier IV is active all the time (see Figure 3.10), offering a 99.995% availability corresponding to an interruption of 0.4 hour per year [100].

Let's consider an example of Tier IV classification depicted in Figure 7.1. This topology consists of two flow paths from the electric power sources to the load points, namely the servers. In a normal operating mode, the servers are powered by both paths.

Each path is supplied by two different power sources PS_1 and PS_2. However, if one of these power supplies fails, the power is supplied by a backup power generator

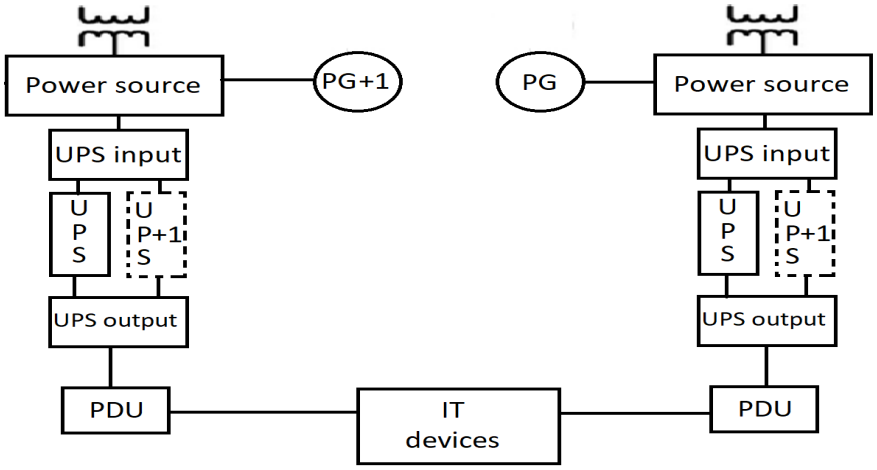

Figure 3.10: Example of Tier IV architecture

(PG). Thus, initially, the generator is on standby and is only brought online after PS_1 or PS_2 becomes unavailable. Power sources provide a medium voltage, typically less than 50 kV. This voltage is used for distribution to two transformers Tr_1 and Tr_2, one on each flow path. Transformers are used to decrease the voltage of electricity. Then, the power enters the building with low-voltage lines going to FDP_1 and FDP_2, the Front low-voltage master Distribution Panels, to supply four uninterrupted power supply systems noted UPS_i, $i = 1, \ldots, 4$, two per path.

Typically, an UPS combines three functions in one system. First, it contains a transfer switch that chooses the active power input (either power source or generator power). After a power source failure, the transfer switch senses when the generator has started and is ready to provide power. Typically, a generator takes 10 to 15 seconds to start and complete the full rated load [62]. Second, the UPS contains some form of energy storage (battery) to bridge the time between the utility failure and the availability of power generator. Third, the UPS conditions the incoming power feed, removing voltage spikes in the alternating current. This conditioning is accomplished via the two components included in the UPS system (inverter and converter).

The output lines from the two UPS systems on each flow path are finally routed to a Back low-voltage master Distribution Panel (BDP) installed in the Data Center floor. We note BDP_i the panel on i^{th} flow path, $i = 1, 2$. Finally, each BDP_i is

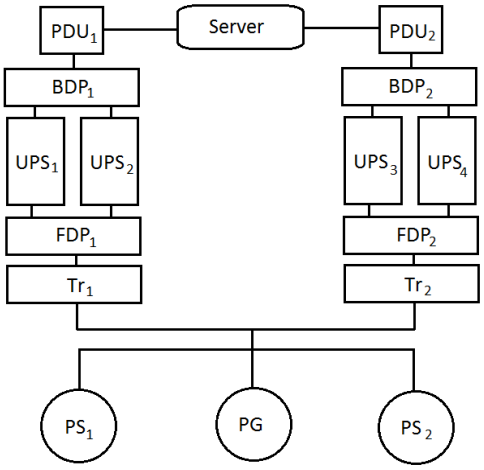

Figure 3.11: A typical Tier IV DC electrical power system

connected to a Power Distribution Unit, noted PDU_i.

The power distribution units are like the breaker panels in residential houses but can also incorporate transformers for final voltage adjustments. They take a larger input feed and break it up into many smaller circuits that distribute power to the servers. A typical PDU handles 75 to 225 kW of load [62]. PDUs are the last layer in the distribution architecture to route the power to the servers or the load points.

As explained previously in this section, the network sub-system components (telecommunication devices) must be powered to ensure their functions. Therefore, these components produce heat (known in the literature as *Joule heating*), which must be removed from the IT room to prevent the equipments temperature from rising to an unacceptable level. Thus an important part of the Data Center system which is responsible for the extraction of heat is the thermal sub-system. This one will be detailed in the next section.

3.4 Data Center's Thermal System

The power energy consumed by computer equipments is almost entirely transformed into heat by joule effect. To keep the equipments at a constant temperature, a cooling system (thermal sub-system) is necessary. On a personal computer, this role is held by one or two fans. In a room containing several hundreds of computers,

the installation of air conditioning is necessary. The energy consumed by cooling systems represents at least 50% of DC consumption in 2008 [83] (see Figure 3.12). New DCs often state that they use natural cold sources, usually air, in addition to air conditioning [62].

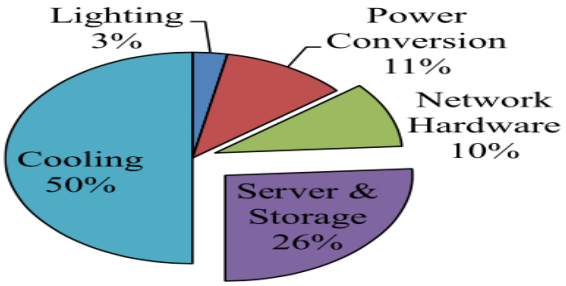

Figure 3.12: Data Center's Energy Consumption

The American Society of Heating and Air-Conditioning (ASHRAE) issued its first thermal guidelines for DCs in 2004 [28] to avoid hotspots in racks and inside the IT room. The original ASHRAE air temperature recommended value range is $20-25°C$ $(68-77°F)$. If the temperature is too high, the DC devices get damaged or switched off automatically.

The cooling system is a bit simpler than the power system. The floor is not only used for cabling, but also for the passage of fresh air to the IT devices. A well-cooled Data Center is a Data Center where the air circulates properly. There must be no mixing between the air consumed and the air blown. The mixture causes the cooling system to operate less efficiently.

Cooling systems evacuate the heat generated by all equipment. To evacuate heat, a cooling system must utilize some hierarchy of loop systems, each bringing in a cold water that warms up via some form of heat exchange and is somehow cooled back again. An open loop system replaces the outgoing warm water with a cool supply from the outside, so that each cycle through the loop uses new material. A closed-loop system recirculates the same water again and again, transferring heat to an adjacent upper loop in a heat exchanger, and eventually the environment. All systems must contain a loop to the outside environment for ultimate heat rejection.

The simplest closed loop systems contain two loops. The first loop is the air

circuit shown in Figure 3.13 (fresh air in blue color and hot air in red color), and the second loop (that is the liquid supply to the Computers Room Air Conditioning (CRAC)) leads directly from the CRAC to external heat exchangers (typically placed on the building roof) that discharge the heat to the environment.

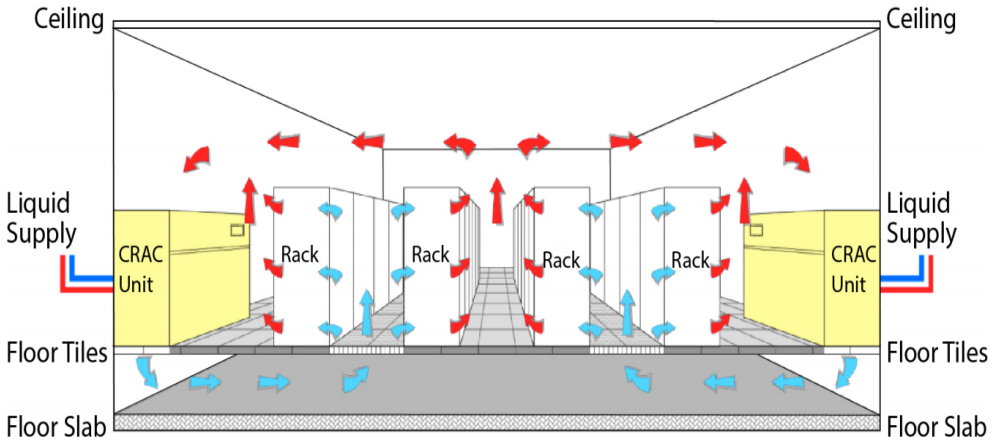

Figure 3.13: Data Center's hot/cold aisle architecture

A three-loop system is shown in Figure 3.14. The CRACs receive chilled water, called Process Chilled Water Supply (PCWS), from an intermediate circuit containing a chiller. The chiller exchanges the heat into a condenser water loop that is open to the atmosphere through cooling towers. The condenser water loop rejects the heat coming from the condenser side of the chiller.

Each topology presents tradeoffs in complexity, efficiency, and cost. For example, fresh air cooling can be very efficient but does not work in all climates, does not protect from airborne particulates, and can introduce complex control problems. Two loop systems are easy to implement, are relatively inexpensive to construct, and offer protection from external contamination, but typically have lower operational efficiency. Three-loop systems are the most expensive to construct and have moderately-complex controls [84], but offer a good efficiency when employing economizers.

Let's consider the example of a DC's thermal system depicted in Figure 3.15. The cooling tower pumps water to the chiller in order to be cooled. Then cooled water is delivered to the CRAC unit inside the IT room. The CRAC unit extracts

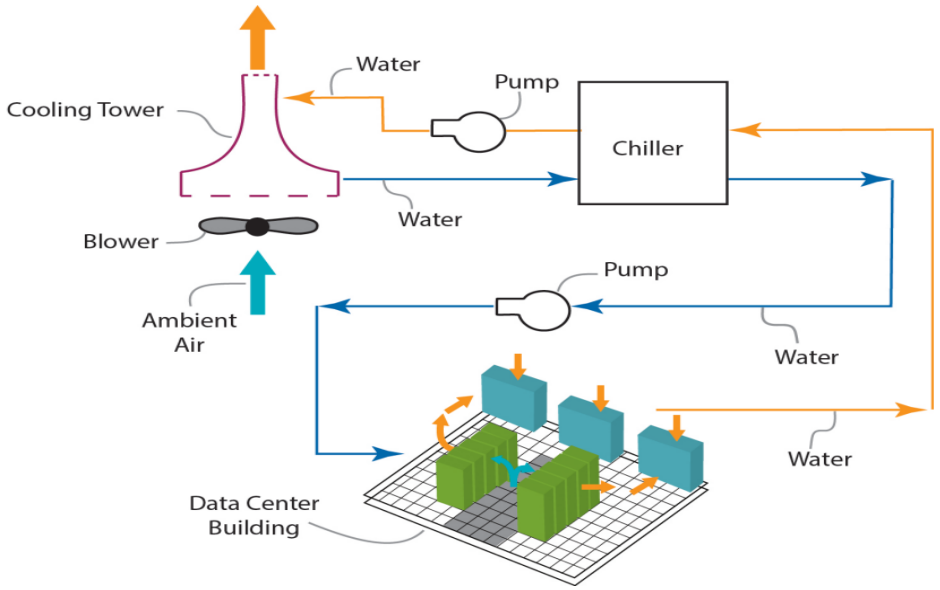

Figure 3.14: A three-loop thermal system

the air from the cooled water and provides the cooled air to the servers.

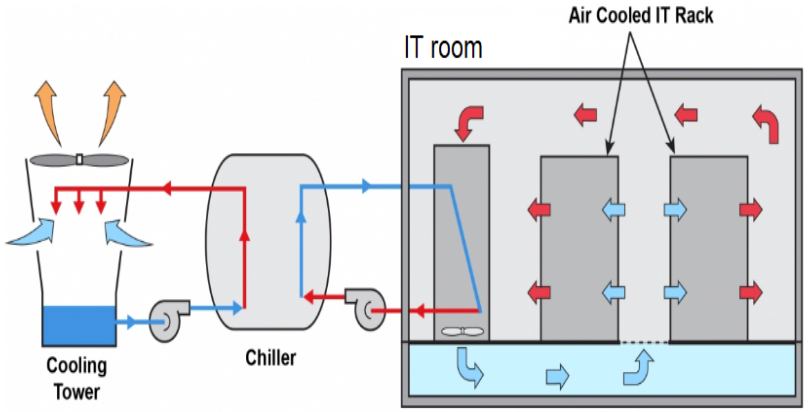

Figure 3.15: Example of a DC's thermal system

3.5 Conclusion

In this Chapter, we presented the notion of Data Center system in general and its different sub-systems. First we highlighted that the proper functioning of a DC system is based on the continuity of the services provided by the equipments of the network sub-system. Then, in order to ensure a constant service, these equipments must be provided with a sufficient and continuous power energy, and kept

in a constant and acceptable temperature. The electrical sub-system is responsible for providing energy, and the thermal sub-system is responsible for keeping the network sub-system in constant temperature by providing fresh air (or extracting the produced heat).

In the next chapter we present the different works carried out in the literature to perform safety analysis of Data Center systems.

Chapter 4

State of Art

In the last few years, Data Centers became an important topic related to safety analysis. The main objective of a Data Center system is to provide a constant and continuous service. This service is provided by the equipment of the network sub-system. Thus the DC network sub-system plays a significant role in determining the level of reliability, availability, communication bandwidth and latency. The DC network sub-system's equipment must also be powered by the electrical power sub-system, and cooled by the thermal sub-system. Accordingly many researches have studied each DC sub-system separately (electrical, thermal, network) in terms of reliability and availability, while other researches have studied correlations and dependencies between these sub-systems.

4.1 Network System

When looking at the DC's network system, the main objective is to maintain a continuous service of the IT equipment with a certain quality of service. Thus network performance and reliability are key design goals for any DC's network system.

Some approaches model the network system without providing support for reliability analysis. In this context, Queueing Network (QN) theory has been widely used to model DC's network sub-system for performance analysis. Yang et al. [111] consider a finite capacity M/M/R queue with second optional channel. Using the matrix - geometric method, they obtain the steady-state probability distribution

and various system performance measures such as packet loss probability and the mean delay. Sharma and Virtamo [101] consider a queue with finite buffer. They propose algorithms to compute the stationary density of the workload process, the waiting times and the packet loss probability. In [78], the authors investigate the blocking probability of QN with finite buffers and a Markovian arrival process. Other approaches were used to model the network system as Queueing Petri Nets (QPN). They [94] [87] combine the modeling power of Stochastic Petri Nets (SPN) and QN to model specific network components with a high level of details. In [67] [68] black-box statistical models are used to analyze the network performance. The study focuses only on selected parts of the network system. Such models are highly specific and are difficult to use for a different system configuration than the one for which they have been designed.

The above-mentioned contributions propose approaches to predict performance indicators (mean delay and packets loss) of DC' network systems based on QN approaches. However the definition of the required performance models needs a non-negligible effort and the analysis of such models requires a high computational cost due to the state space explosion problem. Another major limitation is that they do not take into account the reliability of the DC's network system components.

Others studies examine the DC's network reliability [103]. They usually analyze the failure of network devices based on the network error logs collected from thousands of network devices [57]. The study of *syslog* messages are often used to identify a failure. However, *syslog* messages may be misleading. Indeed a network device may send a *Link down* message even if a link is operational. In [95], the network reliability for Torus and Benes networks is assessed using Reliability Block Diagrams (RBDs). However, this method does not take into account neither the functional dependencies, nor temporal dependencies between events occurrences. In [78] authors introduce several computation measures using a Markov model. However, the model becomes unreadable for large scale network systems (state space explosion).

In summary, most of these studies focus on the failure characteristics of network devices and links, without studying their impact on network traffic (packets

circulating in the DC's network). Network incidents often manifest as failures or anomalies in packets production (traffic).

4.2 Electrical Power System

When looking at the electrical system, most studies have been carried out to improve the energy efficiency of this system. However the energy efficiency cannot be improved without maintaining the operation of the system's equipment. Thus, some works have also considered the reliability and the availability measures of the system.

In [110], authors propose a heuristic algorithm for the combinations of events, which lead to failures. The goal is to identify sequences of events (alarms) which lead to a critical failure based on a dependency graph. However, these sequences do not cover all parallel event sequences, and flows circulating in the system are not represented. This problem is partially solved in [49] by introducing in the model the different flows circulating in the system. The basic disadvantage of this approach is that it generates a large number of flow combinations, that is, all possible flows that satisfy simultaneously the demands for all specified components. The method thus becomes extravagant even for small sized electrical systems.

A model-based approach to calculate the power system reliability using SPN is presented in [75]. The authors compare the different electrical topologies in terms of reliability and availability. In [70], fuzzy reasoning with SPN is used to detect a failure in an electrical power system. For every section or part of the system, there is a model based on the expert knowledge with possible failure causes. Backward reasoning with probability values is used to identify causes of a failure. Continuous-time Markov chain (CTMC) models are also adopted to model the availability of DC's electrical topology in [26]. However, SPN and CTMC models have a limitation with large complex system which is the state space explosion.

To overcome this disadvantage, SPNs are usually combined with other modeling formalisms, where SPNs are used to model the system behavior, and the other formalisms for modeling system components separately. For example in [97], the au-

thors propose a methodology which combines the advantages of both SPNs and RBD to assess dependability of a DC's power system taking into account the interactions between its components. Nevertheless, despite the strong mathematical properties of the approach, it does not take into account the maximum power capacity to evaluate adequacy. This is why in [42], a tooled approach to estimate reliability and availability of a DC's power system, called Mercury, is proposed. This tool supports RBD, SPN and Energy Flow Model (EFM). The EFM verifies the energy flow model on the electrical power system, taking into account the power capacity that each component can provide. Another research work in [106] employs Failure Modes, effects, and Critically analysis (FMECA) with RBDs, considering them as strong mathematical modeling techniques, to evaluate the reliability of DC's electrical power system and provide high system availability. But it is difficult to use this technique since the failure rates are particularly difficult to estimate when human performance is involved.

The main advantage of the application of RBDs in industrial systems, is that they involve only a combination of series or parallel configurations, that is, they do not take into account the redundancy configurations for example. This is why this approach is extended and Dynamic Reliability Block Diagram (DRBD) model is proposed in [93]. This technique supports the reliability analysis of a DC's electrical system. The additional blocks for modeling dependencies made the DRBD model complex. The DRBD model is automatically converted to a Petri net model in order to perform behavior properties analysis, which may certify the correctness of the model. However the major limitation of this technique is that it is impossible to represent the propagation of electrical flows in the system.

Finally, in [43] Bouissou proposes a new modeling formalism, called Boolean logic Driven Markov Processes (BDMPs), to solve all the modeling difficulties of complex systems with dynamic reconfigurations. This technique, based on Markov Chain models, allows analyzing the reliability of DC's electrical system with standby redundancies. However this technique is not suitable for production and repairable systems.

4.3 Thermal System

When looking at the thermal system, the most important is to keep the IT equipments (servers) in an acceptable temperature by evacuating heat (generated by equipments due to joule effect) from the Data Center room. A well-cooled Data Center is a Data Center where the air circulates correctly. Therefore some works have been carried out to study the heat dissipation in Data Centers.

A network of temperature sensors is usually deployed to monitor thermal dynamics of Data Centers. In [64] the thermal system is modeled with Computational Fluid Dynamic (CFD) technique. This technique simulates the thermodynamic processes inside the servers room and estimates the temperature inside the DC room. The same CFD technique is used in [85]. In these works, not only the temperature is estimated but also the propagation of the air flow and heat transfer within the DC. In [45], a new approach, which combines the accuracy of CFD with real-time data-driven prediction algorithms, is applied to improve measurement of the temperature variation. Another detailed CFD analysis of various air distribution systems and their cooling efficiency is described in [47]. However the CFD-based solutions in the literature are effort-intensive for model preparation and time consuming for gaining good results, which makes them inadequate for complex systems.

In [46] a data-driven approach is used to detect the cooling problem. A workload cooling profile for each server is build using monitoring data available in most DCs, such as environmental temperature and hardware status. Then, with these profiles, failures are detected by comparing the observed temperature with those obtained by model prediction of cooling profiles among different servers. The approach is applied on servers only, the other network devices are not considered. The limitation of these research works is that the profile data are related to a specific DC's cooling system, with specific environmental and operational characteristics. Since these research works are data-driven, the results are limited to the information that could be obtained from the recorded data.

4.4 Global DC System

Some works were carried out in different perspectives such as cost-effective and low-latency [99], energy-aware issues [67] and structural robustness [74]. But, very few works characterized operational failure and recovery behaviors in a detailed manner. They quantified only reliability and availability of servers in DC network. Wang [107] studies the impact of DC's electrical components failures on network components, captured through the use of fault regions, which is the case of a set of connected components failing together. The study considers different metrics of interest including throughput and routing failure rate. However, reliability and availability are not considered. Alshahrani [24] presents a detailed analytical modeling methodology based on queuing theory. However, only performance indices (throughput and delay) of a typical fat-tree network are evaluated, and only the reliability impact of the electrical sub-system is taken into account. [27] considers a large amount of data generated by means of CFD simulations, and defines a method based on neural networks for predicting the temperature of the air inside a servers room. The servers energy consumption is also considered. However, this research work focuses only on energy consumption impact on the temperature within the DC room. No reliability and availability metrics were estimated.

In [73], Patterson evaluates the impact of the temperature on energy efficiency and suggests the correct temperature for DC operation. However, the author was not concerned with the availability within the DC environment. In [86], authors present an approach to calculate the reliability of different DC's topologies and compare them using SPN. However, the authors do not focus on the dependencies between thermal and electrical systems. Wei [30] combines the advantages of both RBD and SPN for quantifying availability of Virtual Data Center (VDC). DC cooling architectures are not the focus of this work and the proposed models are specific for modeling VDC. In [56], a comprehensive analysis on how cooling infrastructures impact DCs sustainability, cost and dependability is provided. The authors present five real-world DC cooling architectures and data to explore the environmental impact and dependability metrics. But this work does not take into

account the network system. In [32], an analysis is carried out to ensure availability by providing adequate cooling resources to match the heat load. This work did not cover the impact of cooling component failures on the availability of the IT room.

In [29] researchers at Google implement neural networks to model the DC thermal topology. Using a statistical model, they study the influence of one or more controllable parameters on the power efficiency, reliability and cooling cost. This study does not focus on the network topology, and does not consider the impact of the energy production on servers. Couto [48] presentes a preliminary study on reliability of network topologies in DCs. The study is a graph-based and takes into account the failures of the main network elements (servers, switches, and links) in relation to power energy consumption. However the study does not consider repair behaviors and other related failure causes such as temperature variations.

Finally in [113] authors present a comprehensive availability analysis for a commercial and high-availability server system with multiple physical components, namely IBM BladeCenter ®, consisting of 14 separate blade servers impacted by the power and the cooling systems. The study identifies availability for different configurations, compares different designs, and demonstrates that the system designs can deliver a high availability to meet customer requirements. However, the methodology applied in this work is based on fault trees, and the major limit of such a formalism is that it takes into account neither the order of events occurrences, nor the relationships between the system components, in terms of flows circulating between these components.

There has been little research works in the literature that studied the whole DC's system with the different interactions between its sub-systems. The existing studies are partial and focus only on one sub-system, sometimes two. In our knowledge, the approaches in the literature do not allow the analysis of the interactions between all the DC sub-systems. Moreover, none of these approaches allows both reliability and performance analysis of the whole DC's system. In these thesis works, we propose to use Production Trees modeling technique [66] to analyze the complete DC's system, taking into consideration the interactions between its sub-systems.

The next Chapter is dedicated to the description of the Production Tree mod-

eling technique, the formalism on which our methodology is based. We will introduce also an extension of this technique, to deal with the dependencies between the DCs sub-systems. Moreover we will propose an algorithm to solve a production tree modeling a system in order to estimate the different reliability, availability and performances metrics of this system.

Chapter 5

Production Trees

5.1 Introduction

Production Trees (PTs) are a very recent modeling methodology developed for production availability analysis [66]. This formalism allows modeling the relationships between a system components with a particular attention to the flows circulating between these components. PTs look like Fault Trees (FT) with nodes, which represent components, and gates which model the behaviors. A capacity flow moving from a source to a target component is also represented to provide a sound semantics to classical FT.

The PT technique is suitable for DC's systems reliability and availability analysis, as it allows modeling the DC's system behavior taking into account the flows circulating between its components according to their maximum capacity of production. For example, in order to satisfy load demands in the DC's system, it is necessary to generate sufficient power energy and transport it to the load points (IT components), taking into account the maximum capacity of each component in the DC's system.

However, the interactions between the sub-systems of the DC's system involve different types of flows (energy, air and packets flows). Currently the PT modeling technique allows dealing with only one kind of flow at once. Therefore, in order to deal with dependencies between different types of DC's flows, we introduce a new modeling mechanism to this modeling technique.

Moreover, as PTs modeling technique is not tool supported, simulation is used as a solution for reliability and availability analysis of the system. However, the simulation produces only approximate responses because it relies mainly on the use of random number generators to provide the input of the model. Therefore to analyze a Production Tree model, we propose a PT assessment algorithm based on flows circulating in a production system to estimate the reliability and the availability of the modeled system. The basic idea of this assessment algorithm is inspired from [10]. Instead of identifying all basic subsystems and combine them after, each gate of the production tree combines all its entries (children) by applying rules. The rules depend on the semantics of the gates and their policies.

This Chapter is structured as follows. In Section 5.2, we present an overview of the Production Trees modeling technique as introduced in [66]. Section 5.3 is dedicated to the PT extension to model particular system behaviors (interactions between different types of flows). Section 5.4 is dedicated to the assessment algorithm developed to analyze a Production Tree model. Finally, Section 5.5 concludes this Chapter.

5.2 Production Trees

Production Trees [66] provide two types of components to model a production system: basic components and gates. Basic components represent the production or treatment units of the system whereas the gates model the interactions between these units and thus the behavior of the whole system. Basic components are similar to basic events in a Fault Tree (FT). However, unlike the gates of FT, the gates of PT are not logical. They allow dealing with production flows upstream and downstream a production line, according to the type of these flows. Three types of flows circulate in a PT:

- *Capacity flow* moving forward from a source to target units.
- *Demand flow* moving backward from a target to source units.
- *Production flow* moving forward from a source to target units.

The production depends on the demand which itself depends on the capacity.

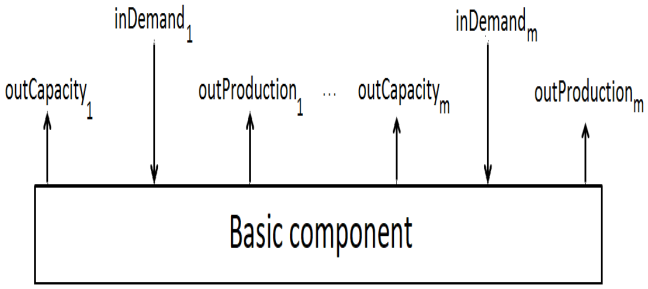

Figure 5.1: Flows circulating in/out a component

First, each component exports its actual production capacity, noted *outCapacity*. This capacity is null if the component is failed and equal to its intrinsic capacity (*intraCapacity*) otherwise. Then, the component receives a demand, noted *inDemand*, which, in stabilized situations, should not exceed the component capacity. Finally, the component exports a production (*outProduction*), which is the minimum of its actual capacity and the input demand. If the demand is null, the component is considered in standby mode. Figure 5.1 shows the flows circulating in and out a component having m parents and n children.

In PT, the gates cannot fail. They only serve to permit, inhibit or modify the passage of flows. In [66], three types of gates are defined: the *PLUS-gate*, the *MIN-gate* and the *SPLITTER-gate*.

1) The *MIN-gate*: It has one parent and two or more children. Its output capacity is the minimum of the output capacities of its children and of its intrinsic capacity (Equation 5.1). The input demand of the gate (coming from its parent) is propagated unchanged to its children. Finally, the output production of the gate is the minimum of the output production of its children (Equation 5.2). Figure 5.2 shows the graphical representation of the *MIN-gate* with two children (n=2).

$$outCapacity = min(inCapacity_1, \ldots, inCapacity_n, intraCapacity) \tag{5.1}$$

$$outProduction = min(inProduction_1, \ldots, inProduction_n) \tag{5.2}$$

2) The *PLUS-gate*: It has one parent and several children. Its output capacity is the minimum of its intrinsic capacity and the sum of the output capacities of its

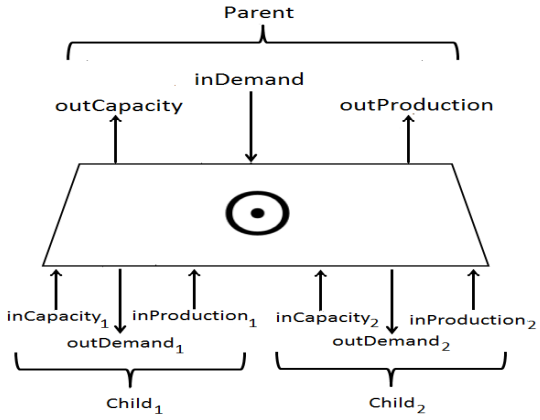

Figure 5.2: Graphical representation of a MIN-gate for n=2

children as specified in Equation 5.3. The input demand of the gate is propagated unchanged to its children. Finally, the output production of the gate is the sum of the output productions of its children (Equation 5.4). In the case where the output capacity of the gate is not equal to the output capacity of its children, the input demand of the gate is propagated to its children according to an allocation strategy. Several allocation strategies can be considered. One of these strategies is pro-rata strategy, in which the demand is allocated according to a pro-rata of their capacities. Another strategy, which is priority, consists to allocate the maximum production to the first child, the maximum of the rest to the second child, etc. Figure 5.3 shows the graphical representation of the *PLUS-gate* with two children (n=2).

$$outCapacity = min(\sum_{i=1}^{n} inCapacity_i, intraCapacity) \tag{5.3}$$

$$outProduction = min(\sum_{i=1}^{n} inProduction_i) \tag{5.4}$$

Note that for both *MIN-gate* and *PLUS-gate*, $inCapacity_i$ is equal to $outCapacity_i$ of child i, $i = 1, \ldots, n$. Similarly, $inProduction_i = outProduction_i$.

3) The *SPLITTER-gate*: unlike the other gates, this gate has only one child and several parents. The output capacity of the *SPLITTER-gate* is the minimum of its intrinsic capacity and the output capacity of its unique child. It is transmitted unchanged to its parents. The output demand of the gate is the sum of its parents demands. Finally, the output production of the gate is split among its parents

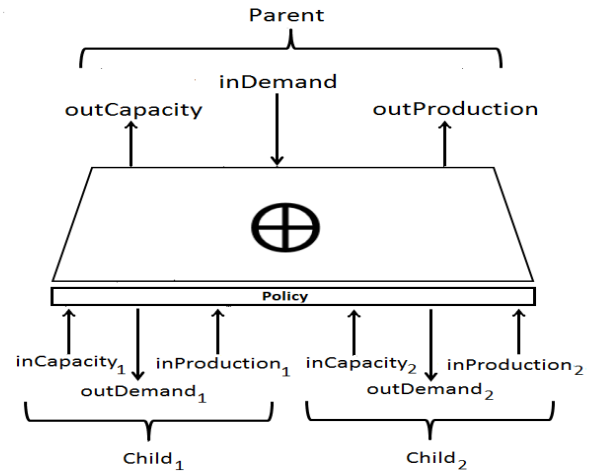

Figure 5.3: Graphical representation of a PLUS-gate for n=2

following an allocation strategy (priority, pro-rata, ...), as for *PLUS-gate*. Figure 5.4 shows the graphical representation of the *SPLITTER-gate* with two parents (m=2).

Note that, $inCapacity$ and $inProduction$ are equal to $outCapacity$ and $outProduction$, respectively, of the unique child of the gate.

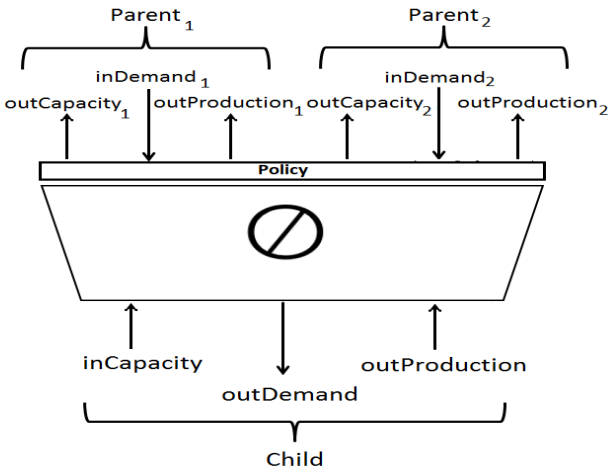

Figure 5.4: Graphical representation of a SPLITTER-gate for m=2

Let's consider the example of the chilling system depicted in Figure 5.5.

The system consists of two chillers Ch_1 and Ch_2 responsible for chilling the coming water. First, the water is routed to both chillers in two redundant paths. In the PT, this is modeled using a *SPLITTER-gate*. Each output of this gate becomes then one of the two inputs of a *MIN-gate*, the other input being the intrinsic capacity

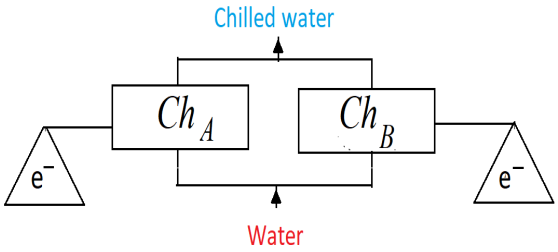

Figure 5.5: Example of a chilling system

of a chiller. Finally, as both chillers have then to route the chilled water, their production capacities are combined using a *PLUS-gate*. The obtained PT model is illustrated in Figure 5.6.

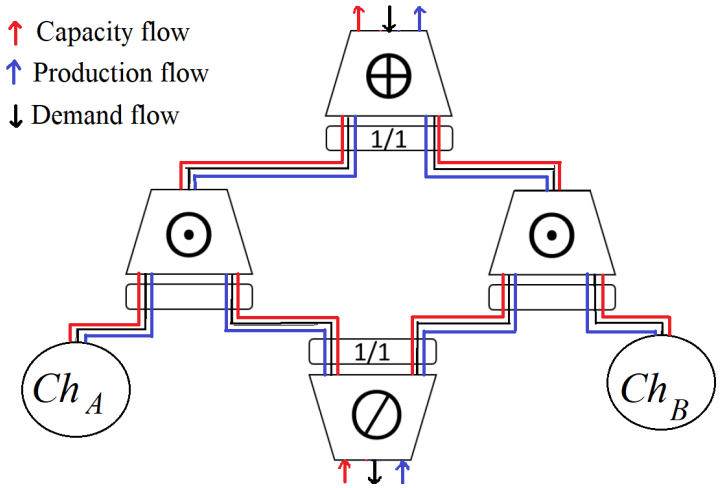

Figure 5.6: The PT modeling the chilling system

.

5.3 PT model Extension

Currently, the gates defined in the PT modeling technique allow dealing with one kind of production flow at once. However some behaviors in production systems, such as Data Centers, involve different types of flows. In order to be able to deal with these behaviors, we extend the PT modeling technique by introducing a new gate, namely the *COND-gate* [37].

The *COND-gate* has one parent and two children or more. Each child repre-

sents a specific kind of flow. Let K_1 and K_2 be the types of two flows at the input of the gate. The output flow of the gate is a flow of type K_1 and its output capacity $outCapacity_{K_1}$ depends, on the one hand, the gate intrinsic capacity $intraCapacity$, and, on the other hand, the input capacity of type K_2 flow, according of a predefined function $f(inCapacity_{K_1}, inCapacity_{K_2}) : N \times N \rightarrow N$, where $inCapacity_{K_1}$ and $inCapacity_{K_2}$ are the input capacities of flow types K_1 and K_2, respectively.

$$outCapacity_{K_1} = min(intraCapacity, f(inCapacity_{K_1}, inCapacity_{K_2})) \quad (5.5)$$

It follows that the input demand of the gate is of K_1 type flow ($inDemand_{K_1}$). Since the gate has two children, this demand is forwarded unchanged to the gate children, according to their type, namely $outDemand_{K_1}$ and $outDemand_{K_2}$. These demands depend on both $inDemand_{K_1}$ and $outCapacity_{K_1}$, according to a predefined function, for example, the *min* function.

Finally, the output production of the gate is a K_1 flow type and its value is according to function $f(inProduction_{K_1}, inProduction_{K_2}) : N \times N \rightarrow N$ where $inProduction_{K_1}$ and $inProduction_{K_2}$ are the input productions of flow types K_1 and K_2, respectively.

Figure 5.7 shows the graphical representation of the *COND-gate* with two children (n=2).

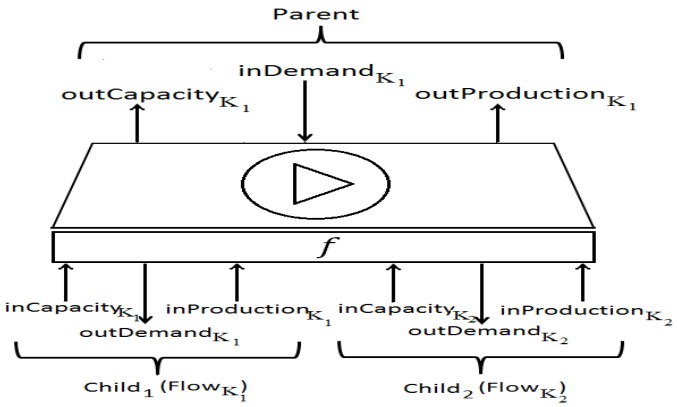

Figure 5.7: Graphical representation of a COND-gate for n=2

Let's consider the example of the chilling system depicted in Figure 5.5. The role of the chillers is to chill the water as long as they are provided with power energy (e^-). Therefore we use a *COND-gate* to model this dependency as illustrated

in Figure 5.8. When a water demand is routed through a *COND-gate*, the required power demand is sent to the electrical sub-system. This one sends a response (electrical flow) which corresponds to the received demand, unless this demand exceeds its production capacity.

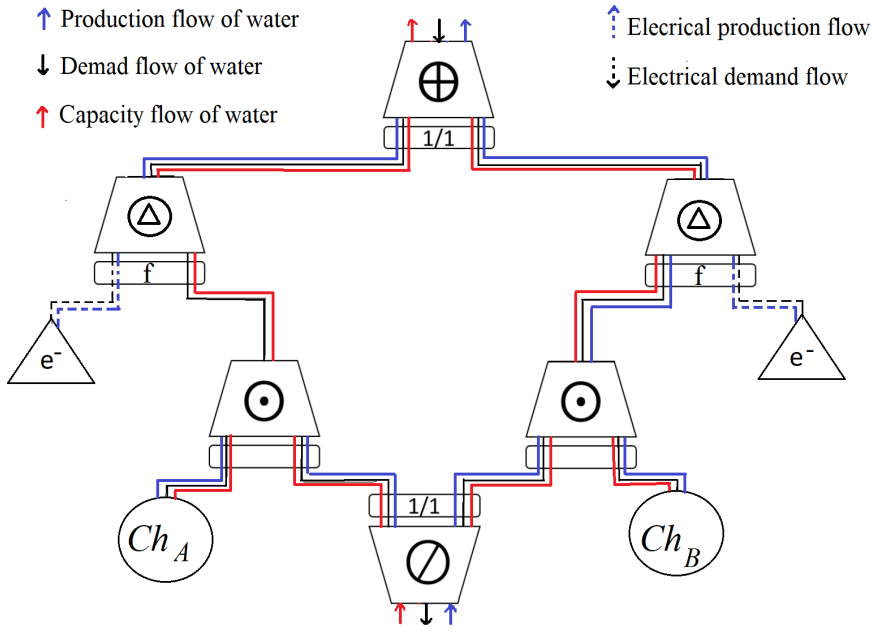

Figure 5.8: The PT modeling the chilling system with a *Cond-gate*

5.4 Production Trees Assessment

The objective of the production tree model analysis is to compute the probability distribution of the production capacity of the modeled system. Our assessment algorithm is based on the Probability Distribution of Capacity (PDC) of flows circulating in the system.

The simplest case is when two components are connected directly in series configuration (without a gate). If c_1 and c_2 are components in such a configuration, the probability of production (or not) in component c_1 depends on its own probability of working (or failing) and the probability of receiving a demand from c_2. However, c_2 sends a demand only if it is working. Thus, the production probability of c_1 depends also on the probability of being working of c_2.

The basic idea of our approach is to consider each gate in the production tree individually and combine all its entries (children) by applying rules which depend on the semantics of the gate and its policies.

For that, we define formally a production tree as a directed graph $G = (V, E_\alpha, s, t, \lambda, \mu, w)$ where:

- V=$V_C \cup V_G$ is a set of nodes where the disjoint subsets are defined as follows:

  - V_C: a set of nodes representing the components.

  - V_G: a set of nodes representing the gates.

- E_α=$E_{Cap_\alpha} \cup E_{Dem_\alpha} \cup E_{Pro_\alpha}$ is a set of edges where disjoint subsets are defined as follows:

  - E_{Cap_α}: a set of edges representing capacity flows of type α.

  - E_{Dem_α}: a set of edges representing demand flows of type α.

  - E_{Pro_α}: a set of edges representing production flows of type α.

- s: $E_\alpha \rightarrow V$, is a function which assigns a source node to each edge $e \in E$.
- t: $E_\alpha \rightarrow V$, is a function which assigns a target node to each edge $e \in E$.
- w: $E_\alpha \rightarrow R^+$, is a function which assigns a value in R^+ to each edge $e_f \in E_f$.
- λ, μ: $V \rightarrow R^+$, are two functions which assign a value in R^+ to each node $v \in V$.

A node $v \in V$ can be a component or a gate. An edge $e_\alpha \in E_\alpha$ may represent either a capacity flow *outCapacity*, a demand flow *inDemand* or a production flow *outProduction* of type α (electric current flow, packet flow, air flow, ...), that circulates between a source node $s(e_\alpha)$ and a target node $t(e_\alpha)$. The edge weight is defined by function $w(e_\alpha)$.

Each node $v \in V_C$ has a Probability Distribution of Capacity (PDC) table, which consists of two attributes. The first one is the value of the capacity flow $w(e_\alpha)$ where $e_\alpha \in E_{Cap_\alpha}$ and $s(e_\alpha) = v$. The second attribute represents the probability of having this capacity.

A component in the production tree model can have two states: *working* or *failed*. In the graph, with each node $v \in V_C$ are associated two values using functions $\lambda(v)$ and $\mu(v)$. These values are, respectively, the failure and the repair rates of component v in the PT. Considering a Markov model, we can calculate the probability of being in each of these states as follows:

$$Pr(failed) = \frac{\lambda(v)}{\lambda(v) + \mu(v)}. \tag{5.6}$$

$$Pr(working) = \frac{\mu(v)}{\lambda(v) + \mu(v)}. \tag{5.7}$$

So, two values are associated with each node $v \in V_C$: when the corresponding component is functional, $w(e_\alpha) = outCapacity$ such that $e_\alpha \in E_{Cap_\alpha}$, and $s(e_\alpha) = v$ with a probability $Pr(working)$. Similarly, if the corresponding component is failed, $w(e_\alpha) = outCapacity = 0$ with a probability $Pr(failed)$.

Using a bottom-up approach, graph G is processed as follows:

- If node v is a component (it never sends a flow demand), that is $\forall e \in E_{Dem_\alpha}$ $\nexists\, v \in V_C$ such that $s(e_\alpha) = v$, then do nothing.
- If node v is not a leaf ($\forall e_\alpha \in E_{Dem_\alpha}$, $\exists v$ such that $s(e_\alpha) = v$), and is a component ($v \in V_C$) then *update* the node's PDC according to its predecessor(s) in the graph.
- If node v is a gate, then *combine* the PDCs of its children according to the gate type (*PLUS-gate*, *MIN-gate*, *COND-gate* or *SPLITTER-gate*).
- If node v is the top node of the graph and thus it does not send a flow capacity, that is $\forall e_\alpha \in E_{Cap_\alpha}, \nexists\, v \in V$ such that $s(e_\alpha) = v$, then *combine* its PDC according to its predecessor(s).

The assessment of a PT model depends mainly on two treatments : *update* the PDC table or *combine* two PDC tables. These treatments are applied on the PT gates, and depend on the gate types and the allocation strategy used, if any. In this thesis work, we have considered two strategies: priority and pro-rata, because they are the most used in DC's systems. In the case where a pro-rata strategy is used in

a PT gate, we use a pro-rata index, noted I_p. Then the PDC table of each child of the gate is updated by multiplying its PDC table by the pro-rata index I_p. When the priority strategy is used, the PDC tables of children gate are not updated, and will be combined directly according to their priority order.

5.4.1 The *PLUS-gate*

This gate is characterized by an output production which is the sum of the output production of its children. Thus computing the total PDC table of the gate consists in summing the PDCs of its children. The principle of computing the sum of two PDC's tables is the following: consider two nodes c_1 and $c_2 \in V_C$ and let X and Y be two random variables with a discrete distribution representing the capacity contents of the two PDCs tables of nodes c_1 and c_2, respectively. The new distribution Z is given by:

$$Pr(Z = z) = Pr(x + y = z) = \sum_{x,y,x+y=z} Pr(X = x) * Pr(Y = y)$$

When a pro-rata strategy is considered, before summing the PDCs of children, we update them first according to the pro-rata index I_p.

5.4.2 The *MIN-gate*

This gate is characterized by the fact that the input demand is propagated unchanged to its children. Moreover, the output production of the gate is the minimum of the output production of its children. Thus, the distribution of the gate is computed as the minimum of PDCs of its children. Let X and Y be two random variables with a discrete distribution representing the capacity contents of the PDCs of nodes c_1 and $c_2 \in V_C$, respectively, the new distribution Z is defined as follows:

$$Pr(Z=z) = \begin{cases} \sum\limits_{x,y,min(x,y)=z} Pr(X=x) * Pr(Y=y) \\ \qquad\qquad \text{if } x \neq 0, y \neq 0 \\ \\ \sum\limits_{x,y,x+y=z} Pr(X=x) * Pr(Y=y) \\ \qquad\qquad \text{if } x = 0, y = 0 \end{cases}$$

5.4.3 The *SPLITTER-gate*

This gate is characterized by a unique child and one or more parents. However, the treatment is similar to the one used for the *MIN-gate* when the pro-rata strategy is adopted, except that the minimum of two PDCs is applied between the unique child's PDC and the PDCs of parents (taking one by one). Supposing X and Y are random variables with a discrete distribution representing the capacity contents of the PDCs of nodes c_1 and $c_2 \in V_C$, respectively, the new distribution Z is defined as follows:

$$Pr(Z=z) = Pr(X=z) * Pr(Y >= z) + Pr(Y=z) * Pr(X >= z)$$

When priority strategy allocation is considered, the minimum between the child's PDC and the first parent's PDC is calculated. Then the child's PDC is updated by the result of subtraction between the PDC of the first parent and the PDC of the child. Let X and Y be random variables with a discrete distribution representing the capacity contents of the PDCs of two nodes v_1 and $v_2 \in V_C$, respectively. The new distribution Z is defined as follows:

$$Pr(Z=z) = \begin{cases} \sum\limits_{x,y,z=0} Pr(X=x) * Pr(Y=y) & \text{if } \; x < y \\ \sum\limits_{x,y,(x-y)=z} Pr(X=x) * Pr(Y=y) & \text{if } \; x \geq y \end{cases}$$

Then, the same treatment is applied to each parents until PDC of the unique child is equal to 0.

5.4.4 The *COND-gate*

This gate is similar to the *MIN-gate*. But instead of dealing with only one specific type of flow, it deals with 2 types of flows corresponding to its two children K_1 and K_2 with a particular attention to the associated function f. In our case, the function considered is *min*, because each DC's component has a capacity of production $outProduction$, and needs the same quantity of energy E to accomplish its function (as explained in Chapter 3). Therefore the function f is defined as: $f(outProduction, E) = min(outProduction, E)$, and the distribution of the gate is computed by calculating the minimum between PDCs of its two children as for the *MIN-gate*. Let X and Y be two random variables with a discrete distribution representing PDCs of nodes c_1 and $c_2 \in V_C$ (corresponding to flow types K_1 and K_2), respectively, the new distribution Z is defined as follows:

$$Pr(Z = z) = \sum_{x,y,f(x,y)=z} Pr(X = x) * Pr(Y = y)$$

5.5 Conclusion

In this Chapter, we gave a description of the modeling technique called Production Trees. We proposed an extension of the technique to deal with the dependencies between flows, by adding a new gate. Finally we proposed a solution method to assess a PT modeling a system which allows estimating reliability and availability of this system. In the next Chapter, we will present our graphical modeling tool and how the proposed PT assessment method is implemented.

Chapter 6

Production Trees Assessment

6.1 Introduction

A significant factor behind the difficulty of developing complex softwares is the wide conceptual gap between the problem and the implementation domains of discourse, which can be reduced using Model-Driven Engineering (MDE) [112]. MDE is a software development methodology whose goal is creating and exploiting domain models. In MDE, meta-modeling allows providing an abstract notation able to describe problem domains in terms of Domain Specific Modeling (DSMs) [113]. Therefore, models are often based on a graphical representation and supported by graphical design tools. A set of DSM tools, such as *Eclipse Modeling Framework (EMF)* [102], enables the user to create models relied on meta models by generating automatically a certain part of codes.

In this Chapter, we present a new EMF-based graphical tool for modeling complex systems, using Production trees, and allowing safety and performance indicators estimation. Firstly, the graphical interface of our tool allows realizing all the steps for building PT models graphically. The model entry is conducted through windows to guide the user, allowing him visualizing and editing PT models within Eclipse using *Sirius framework* [114]. Moreover, as explained in Section 5.4, the PT model is stored in the form of tree. Using the dedicated algorithm for each of its gates, the PT gate is assessed and different safety indicators are estimated and graphically represented. Finally an AltaRica code modeling the system can be

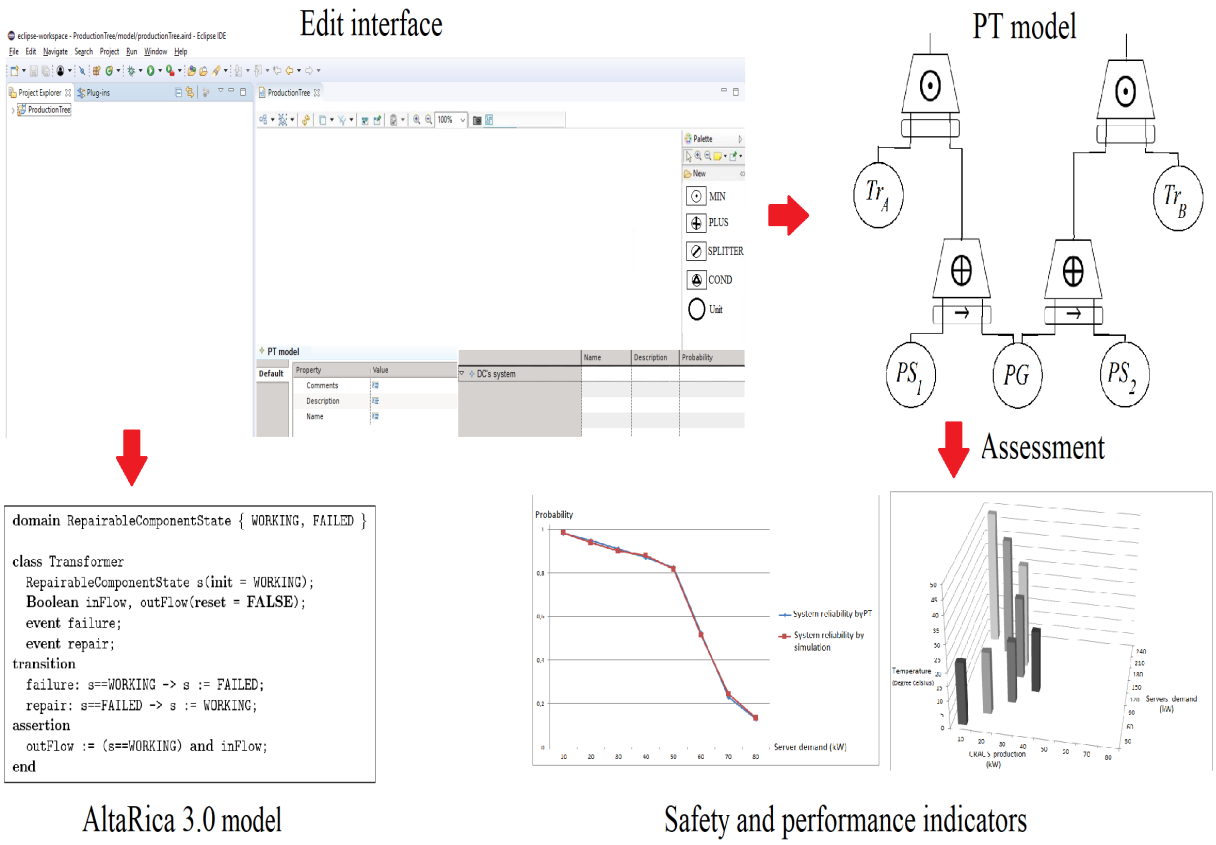

Figure 6.1: An overview of the proposed PT assessment tool

generated in order to validate the obtained results. These results are obtained after assessing the AltaRica code using the different analysis tools of the AltaRica 3.0 tool (see Figure 6.1).

This Chapter is structured as follows. Section 6.2 presents the graphical model-driven engineering tool Sirius Framework, and an overview of our graphical Editor based on Sirius framework, for building PT models. Section 6.3 is dedicated to the algorithms for PT models assessment. Finally Section 6.4 concludes this Chapter.

6.2 Sirius Framework

Our software tool is based on *Eclipse Modeling Framework (EMF)*, which provides an object graph for representing models, as well as capabilities for serializing models in a number of formats, checking constraints, and generating various types of tree editors for use in Eclipse. The *Graphical Editor Framework (GEF)* [124] and *Draw2D* [125]

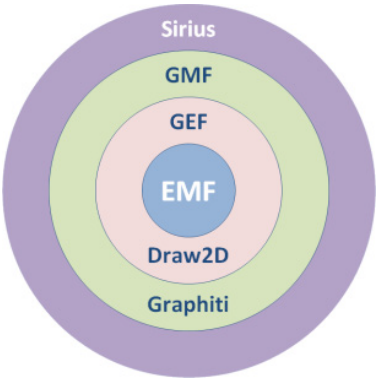

Figure 6.2: Hierarchy of Graphical Model-Driven Engineering tools

provide the foundations for building graphical views for EMF and other model types. The *Graphical Modeling Framework (GMF)*, by encapsulating GEF and *Draw2D* (see Figure 6.2), provides a tool for creating graphical editor with a high degree of flexibility. Creation of editor in GMF is often complex and highly depends on Java, XML and Eclipse plug-in knowledge. By using *Graphiti* framework [91], that hides GEF's complexities from the developer and bridges EMF and GEF to speed up the development of graphical editors, it is possible to design homogeneous graphical editors that visualize an underlying model based on a tool-defined graphical notation [9]. These frameworks (GMF, GEF, Graphiti) are a high level of required knowledge in domain of Java object oriented language, EMF and Eclipse plug-in development. However, *Sirius framework* offers a solution for rapid development of Graphical tool for DSM, without need for understanding any of back-end processes.

Within EMF, the definition of a DSL syntax is usually given using meta-languages such as ECore, used to specify meta models, and OCL (Object Constraint Language) to handle static semantics. Figure 6.3 gives the basic usage flow for developing a graphical editor using GMF.

The starting point is the definition of an ECore meta-model. From this meta-model, GMF provides wizards to create additional models related to the graphical concrete syntax. The graphical model specifies the shape of the PT editor (components and gates). The tooling model states the available tools. The mapping model binds the information from the domain model, graphical model and tooling model. The generator model is used as input for the GUI code generator. As there is no standard meta-model for Production Trees defined in advance, we built an ECore

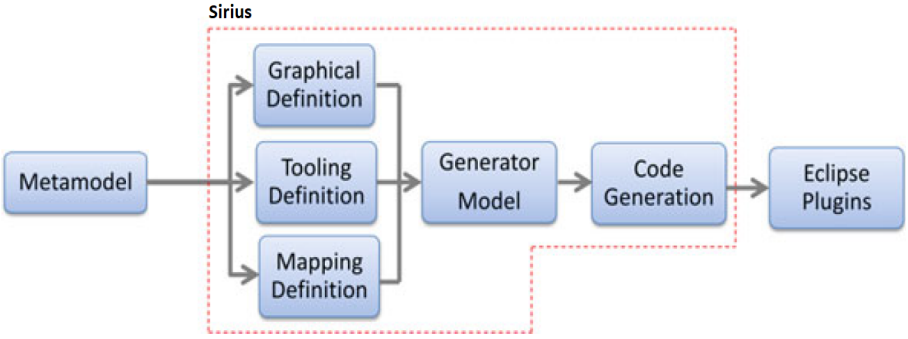

Figure 6.3: Overview of GMF Development Flow

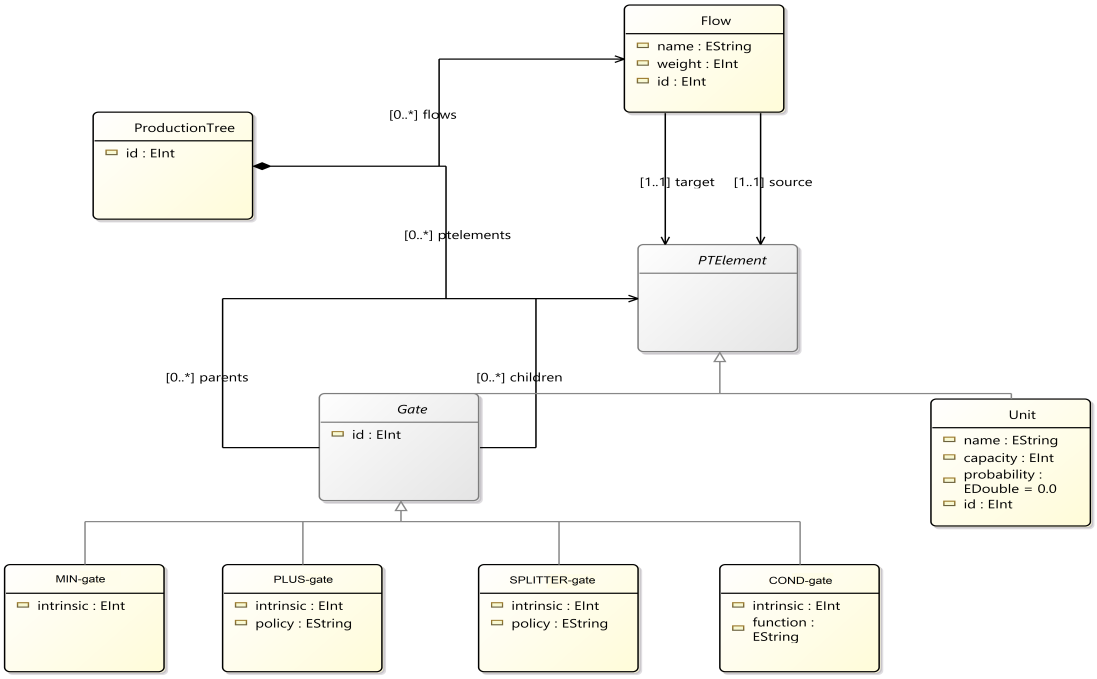

Figure 6.4: Production Trees ECore Meta-model

meta-model for our transformation (see Figure 6.4).

The PT model (ProductionTree class in Figure 6.4) contains components (units) and flows circulating in this model. Each unit has an *id*, a *name*, an intrinsic capacity, noted *capacity*, and a Probability Distribution entry set (explained in Chapter 5), noted *probability*. Each flow has also an *id*, a *name* (air flow demand, air flow production, electric current capacity,...), and a capacity, noted *weight*.

As explained in Chapter 5, the analysis of the production tree model is based on the PDC (*probability*) of each unit (component) according to PT gates. Each gate has an intrinsic capacity, noted *intrinsic*, and a *policy* (for PLUS-gate and SPLITTER-gate) or a *function* (for COND-gate only). PT gates have a specific

treatment according to their types, and this will be illustrated in the following Section.

6.3 PT Assessment Algorithm

As explained in Chapter 5, the treatment of a PT gate depends on its type (semantics) and the allocation strategy used, if any. However, because they rely on the PDC calculations, the algorithms developed for dealing with the gates share some common procedures. Thus before presenting these algorithms, we introduce first these procedures.

• Function $Distrib(x)$ returns the PDC (*probabiliy* entry set) of component x.

• Functions $successor(x)$ and $parent(X)$ return the list of successors and parents of component x, respectively.

• Function $update\_PDC$ is used to update the distribution by multiplying it by I_p, a pro-rata index, in the case where a pro-rata strategy is used in the gate.

• Function sum_PDC is used to sum two distributions and relies on the treatment described in Subsection 5.4.1.

• Function $optim\_PDC$ is used to calculate the minimum between two distributions and relies on the treatment described in Subsection 5.4.2.

• Function sub_PDC is used to subtract two distributions and relies on the treatment described in Subsection 5.4.3.

• Function $cond\_PDC$ is a conditional sum of two distributions and relies on the treatment described in Subsection 5.4.4.

6.3.1 The *PLUS-gate*

The function implementing the *PLUS-gate* treatment is $PlusGate(x)$. It takes as input component x which is its first child. The main objective is to calculate the PDC of this gate by combining PDCs of its children. For this we link first an empty PDC to the gate ($result$), then we sum the PDC of the first child ($Distrib(x)$), and its successors ($successor(x)$), which is the second child of the gate, using function sum_PDC.

If the gate policy is *Priority*, then we sum the DPCs of the children one by one until the *intrinsic* capacity of the gate is reached. Otherwise (pro-rata strategy) we update first the children PDC using *update_PDC* function, then we sum them using function *sum_PDC*. This is detailed in **Algorithm** 1.

Algorithm 1 Distribution of a *PLUS-gate*

1: **function** PLUSGATE(x)
2: $max \leftarrow 0$ ▷ variable used to indicate if the demand is satisfied
3: **struct** Var {float Cap; float Pr} ▷ PDC entry set;
4: $capacity \leftarrow 0$
5: $demand \leftarrow w(e)$ where $e \in E_{Dem}$ and $t(e) == x$;
6: $suc \leftarrow successor(x)$;
7: $size \leftarrow |suc|$;
8: $i \leftarrow 0$;
9: create a new set of Var $d \leftarrow null$;
10: create a new set of Var $result \leftarrow null$;
11: **if** $Policy ==$ "$Priority$" **then**
12: **while** $i < size$ or $max < demand$ **do**
13: $d \leftarrow distrib(suc[i])$;
14: $capacity \leftarrow w(e)$ where $e \in E_{Cap}$ and $s(e) == s[i]$;
15: $max \leftarrow max + capacity$;
16: **if** $max > demand$ **then**
17: $sum\_PDC(result, update\_PDC(d, demand/max))$;
18: **else**
19: $sum\_PDC(result, d)$;
20: **end if**
21: $i \leftarrow i + 1$;
22: **end while**

23: **else if** $Policy == "Prorata"$ **then**
24: **while** $i < size - 1$ **do**
25: $d \leftarrow distrib(suc[i])$;
26: $sum\_PDC(result, update\_PDC(d, pro[i]))$;
27: $i \leftarrow i + 1$;
28: **end while**
29: **end if**
30: **return** $result$
31: **end function**

6.3.2 The *MIN-gate*

The function implementing the *MIN-gate* treatment is $MinGate(x)$. It takes as input component x which is its first child. As for the *PLUS-gate*, we link first an empty PDC to the gate ($result$), then since the gate has no policy, we calculate directly the minimum between the PDC of the first child ($Distrib(x)$), and its successors ($successor(x)$) using function min_PDC. The algorithm dedicated to the *MIN-gate* is provided in **Algorithm** 2.

Algorithm 2 Distribution of a *MIN-gate*

1: **function** MinGate(x)
2: **struct** Var {float Cap; float Pr} ▷ PDC entry set;
3: $suc \leftarrow successor(x)$;
4: $size \leftarrow |s|$;
5: $i \leftarrow 0$;
6: create a new result set of Var $d \leftarrow null$;
7: create a new result set of Var $result \leftarrow null$;
8: $result \leftarrow distrib(suc[0])$;
9: $d \leftarrow distrib(suc[i + 1])$;

10: **while** $i < size$ **do**

11: $result \leftarrow optim\_PDC(result, d)$;

12: $i \leftarrow i + 1$;

13: **end while**

14: **return** $result$

15: **end function**

6.3.3 The *SPLITTER-gate*

The function implementing the *SPLITTER-gate* treatment is $SplitterGate(x)$. As for the other gates, it takes as input component x which is its unique child. The PDC of this unique child will be distributed among its parents ($parent(x)$), that is, the PDC of the child will be combined with the first parent using function min_PDC, then the second one until the last, always using min_PDC (several iterations according to the number of parents). If a priority strategy is adopted, we combine, at each iteration, the PDC of a parent with the PDC of the unique child, then the result of this operation (combining two PDCs) will be subtracted from the PDC of the unique child, until the PDC of this child is equal to 0. Otherwise if the pro-rata strategy is adopted, we update first the parents' PDC using $update\_PDC$ function, then we apply the same treatment as in the priority strategy. The algorithm dedicated to the *SPLITTER-gate* is provided in **Algorithm** 3.

Algorithm 3 Distribution of a *SPLITTER-gate*

1: **function** SplitterGate(x)

2: **struct** Var {float Cap; float Pr} ▷ PDC entry set;

3: $max \leftarrow 0$ ▷ variable used to indicate if the demand is satisfied

4: $demand \leftarrow w(e)$ where $e \in E_{Pro}$ and $t(e) == x$;

5: $par \leftarrow parent(x)$;

6: $suc \leftarrow successor(x)$;

7: $size \leftarrow |s|$;

8: $i \leftarrow 0$;

```
9:    create a new set result of Var result ← null;
10:   create a new set d of Var d ← null;
11:   result ← distrib(suc[0]);
12:   if Policy == "Priority" then
13:       while i < size or max < demand do
14:           d ← distrib(par[i]);
15:           capacity ← w(e) where e ∈ E_Dem and s(e) == p[i];
16:           max ← max + capacity;
17:           if max > demand then
18:               distrib(par[i]) ← min_PDC(result, update_PDC(d, demand/max));
19:               result ← sub_PDC(result, d);
20:           else
21:               distrib(par[i]) ← min_PDC(result, d);
22:               result ← sub_PDC(result, d);
23:           end if
24:           i ← i + 1;
25:       end while
26:   else if Policy == "Prorata" then
27:       while i < size do
28:           d ← distrib(par[i]);
29:           distrib(par[i]) ← min_PDC(result, update_PDC(d, pro[i]));
30:           result ← sub_PDC(result, d);
31:           i ← i + 1;
32:       end while
33:   end if
34:   return result
35: end function
```

6.3.4 The *COND-gate*

The function implementing the *COND-gate* treatment is $CondGate(x)$. The algorithm is similar to Algorithm 2 for the *MIN-gate*. The unique difference is that we combine the PDC of the first child ($Distrib(x)$) with its successors ($successor(x)$) according to function f. In our case this function is min, so we calculate directly the minimum between the PDC of the first child ($Distrib(x)$), and its successors ($successor(x)$) using function min_PDC without any strategy. The algorithm dedicated to the *COND-gate* is provided in **Algorithm** 4.

Algorithm 4 Distribution of a *COND-gate*

```
1:  function CondGate(x)
2:      struct Var {float Cap; float Pr}                ▷ PDC entry set;
3:      suc ← successor(x);
4:      size ← |s|;
5:      i ← 0;
6:      create a new set result of Var d ← null;
7:      create a new set result of Var result ← null;
8:      result ← distrib(suc[0]);
9:      while i < size do
10:         d ← distrib(suc[i + 1]);
11:         result ← cond_PDC(result, d);
12:         i ← i + 1;
13:     end while
14:     return result
15: end function
```

The PT assessment algorithm is an analytical method with a short calculation time. It has linear complexity in the best case, while it has exponential complexity in the worst case.

Finally, in order to validate the obtained results, the user has the option to create an AltaRica code modeling the system he wants to analyze, based on the defined ECore model, and use the high level modeling language AltRica 3.0 with its

tools to assess the obtained model.

6.4 Conclusion

In this Chapter, we proposed a graphical interface for building production trees models using *Sirius* framework. Then in order to estimate availability and reliability of the modeled system, we proposed an algorithm to assess the obtained PT model. This assessment algorithm is mainly based on the PT flows and the type of each of its gates. In the next Chapter, we will show the applicability of this modeling technique on a real DC system, and how our proposed PT extension allows modeling dependencies between DC's sub-systems.

Chapter 7

Modeling a Data Center System using Production Trees

7.1 Introduction

In this Chapter, the production tree modeling technique is applied on a case study. We model first each DC's sub-system (electrical, thermal, and network) separately. Then, by means of the extended PT version we have proposed, we model the interactions between the DC's sub-systems, in terms of flows circulating in the complete system of the DC.

This Chapter is structured as follows. In Section 7.2, we present the DC's system we are interested in. Section 7.3 is dedicated to modeling the DC's system using Production Tree technique. Finally, Section 7.4 concludes this Chapter.

7.2 Case Study

The proper functioning of a DC is based on the continuity of the services provided by the equipments of the network sub-system. In order to ensure a constant service, these equipments must be provided with a sufficient and continuous power energy, and kept in a constant and acceptable temperature. The electrical sub-system provides energy to both the network and the cooling sub-systems. Thus the network sub-system depends on both the electrical sub-system and the cooling sub-system,

which itself depends on the electrical sub-system to operate properly (see Figure 3.4).

7.2.1 The Electrical sub-system

We consider the topology illustrated in Figure 7.1 which combines an electrical system and a thermal system. The topology of the electrical sub-system we consider consists of four layers: *production layer*, *transformation layer*, *storage layer* and *distribution layer*. These are real thermal and electrical systems of a DC [36]. Note that the network sub-system is represented by only twelve (12) servers to deliver service to users. The whole network sub-system will be detailed in the dedicated section (subsection 7.2.3).

In a normal operating mode, the servers are powered by two paths A and B. Each path is supplied by two different power sources PS_1 and PS_2. However, if one of these power supplies fails, the power is supplied by a backup power generator (PG). Thus, initially, the generator is on standby and is only brought online after PS_1 or PS_2 becomes unavailable. Power sources provide a medium voltage, typically less than 50 kV and they represent the *production layer*. This voltage is used for distribution to two transformers Tr_A and Tr_B, one on each flow path [62]. Transformers are used to decrease the voltage of electricity (*transformation layer*). Then, from the *transformation layer*, the power enters the building (*storage layer*) with low-voltage lines going to FDP_A and FDP_B, the front low-voltage master distribution panels, to supply two Uninterruptible Power Supply (UPS) systems per path noted UPS_{iA} and UPS_{iB}, $i = 1, 2$.

An UPS combines three functions. First, it contains a transfer switch or converter ($Conv$) which chooses the active power input (either power source or power generator). Second, the UPS contains a battery (Bat) to bridge the time between the utility failure and the availability of power generator. Third, the UPS contains a rectifier (Rec) to remove voltage spikes in the alternating current.

The output flow from each UPS system is finally routed to the *distribution layer* which contains a back low-voltage master distribution panel installed in the data center floor. We note BDP_{iX} the i^{th} panel on flow path $X = A, B$ and $i = 1, 2$. Then, both $BDPs$ on a path $X = A$ (respectively $X = B$) are connected to four

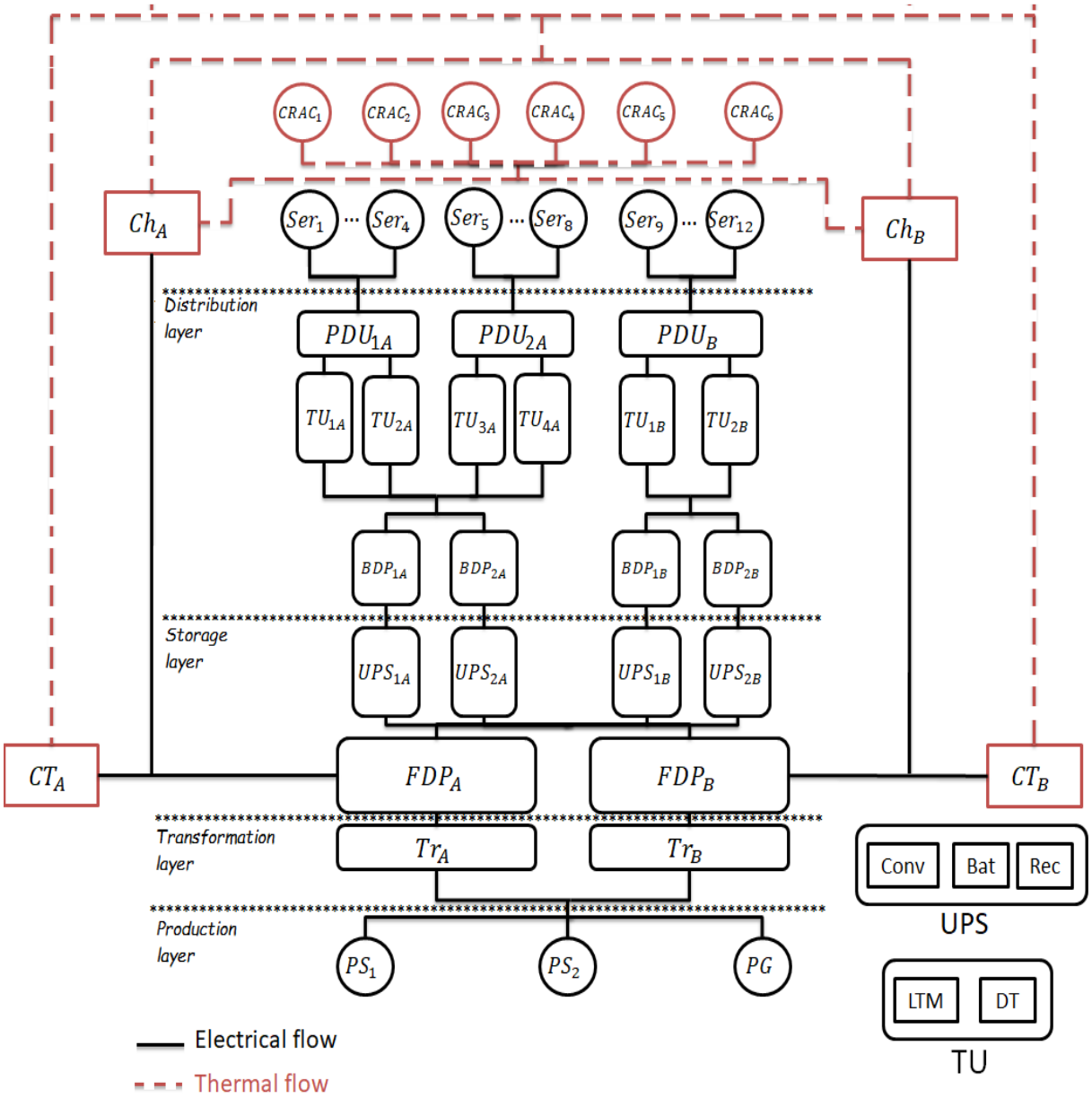

Figure 7.1: The thermal and power sub-systems of a data center

(respectively two) Transfer Units (TU). These units are responsible of transferring the load through a Load Transfer Module (LTM) to a Distribution Table (DT). Finally, each two transfer units are connected to a Power Distribution Unit (PDU). PDUs are the last components in the *distribution layer* to route the power to the servers or the load points. Each PDU provides the electrical flow to 4 servers, Ser_j, $j = 1, \ldots, 12$, grouped in 3 racks (4 servers per rack).

7.2.2 The Thermal sub-system

The thermal system considered in this case study consists of ten components: six CRAC units, two chillers and two CTs. These components are distributed in three layers: *production layer*, *cooling layer* and *extraction layer*. The two redundant cooling towers CT_A and CT_B in the *production layer* drive water from a source S. Each one contains pumps and needs to be powered to get the water from source S. The power energy is provided by FDP_A and FDP_B, respectively (see Figure 7.1). The pumped water is routed to the *cooling layer* containing two chillers Ch_A and Ch_B. The main role of a chiller is to cool the water as long as it is powered by the electrical sub-system. Chillers Ch_A and Ch_B are powered by PDU_A and PDU_B, respectively. Once the water chilled, it is delivered to the *extraction layer* which contains 6 CRAC units. Each CRAC unit extracts the air from the chilled water on condition that it is powered by at least one BDP. Finally, the CRAC units provide the cooled air to the servers. In this scenario each CRAC unit provides air to a rack containing four servers and the cooling system is considered to be operational if at least one of the two CRAC units $CRAC_i$, $i = 1, \ldots, 6$ associated with each rack is working, the other one being in a standby mode (see Figure 7.2).

7.2.3 The Network sub-system

In the description of the DC topology illustrated in Figure 7.1, we have considered the servers as the whole network sub-system [34]. In this section, the network sub-system components are detailed. We consider the fat-tree network [72] illustrated in Figure 7.3. The network has four layers: the lowest layer ($layer4$) contains 80

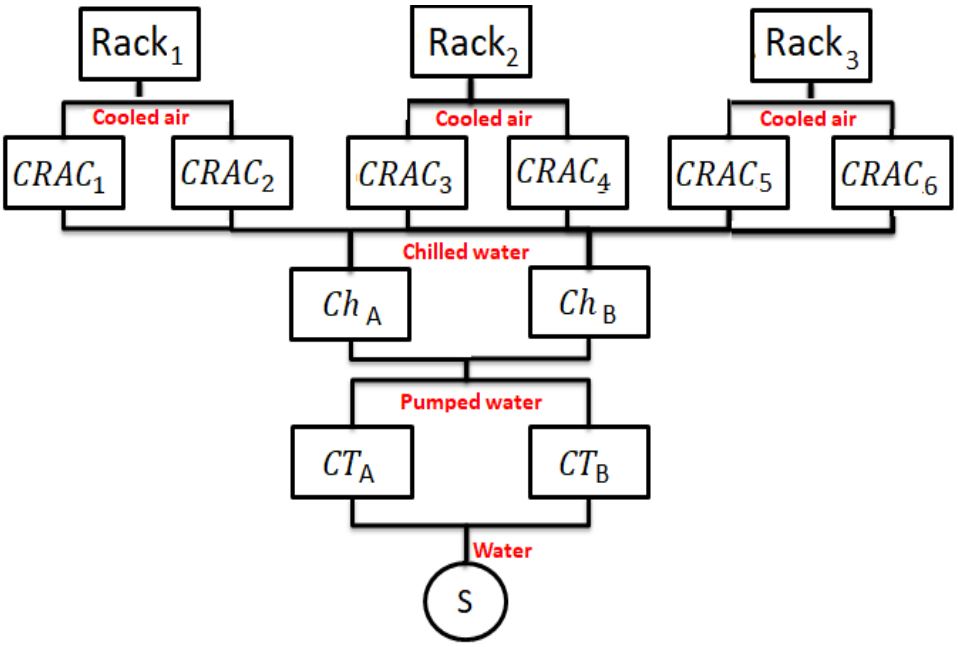

Figure 7.2: The thermal sub-system of a DC

servers Ser_j, $1 \leq j \leq 80$, instead of 12 servers, distributed in four Racks $Rack_i$, $1 \leq i \leq 4$ (each rack contains 20 servers). The layer above ($layer3$) contains 4 switches ToR_i, $1 \leq i \leq 4$, each one is connected to a rack $Rack_i$ in $layer4$. The switches are connected to two aggregation switches $AggS_A$ and $AggS_B$ ($layer2$), for redundancy. The aggregated traffic is then forwarded to the top layer ($layer1$) which contains two access routers $AccR_A$ and $AccR_B$. These route the traffic to Core routers $Core_A$ and $Core_B$ which are connected to the external network (internet).

The function of each component is to route the traffic and thus has a certain communication capacity. we consider that the component treatment capacity, known as the throughput, is considered as the component capacity in both upload and download links.

The servers and ToR switches are powered by the electrical component PDU installed in each rack, while the other switches and routers are powered by at least one BDP installed in the DC room.

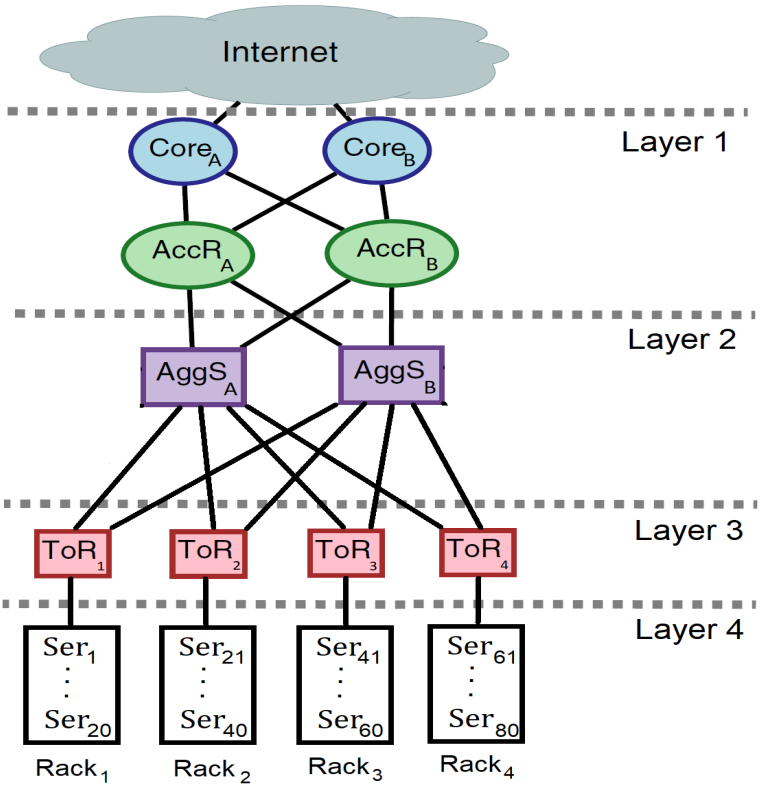

Figure 7.3: the DC network sub-system

7.3 Modeling the system using PT

The electrical sub-system is responsible for providing power to both cooling and network sub-system. In order to model the global system, we model first the electrical sub-system, because it does not depend on any other sub-system of the DC. Then we model the thermal and network sub-systems individually, taking into account the dependency links each of them has with the electrical sub-system [37].

7.3.1 Modeling the electrical sub-system

In general, building the PT model goes through 3 steps: the transmission of the capacity (Step 1), the transmission of the demand (Step 2) and the transmission of the production (Step 3). However, the electrical sub-system has a particular behavior. The electrical production components do not export their capacities. The load points (the servers in our case) export directly their demand, in terms of energy, to the other components of the sub-system. Then the power sources produce the

energy taking into account their maximum capacities (intrinsic capacities). This energy is transmitted to the servers, through the other components of the sub-system. If the demand exceeds the intrinsic capacity of a component on the path, the circuit breaker of this component cuts-off the electrical flow, and the system will not be able to supply the load point. Therefore building the PT model goes through only Step 2 and Step 3, that is, the exportation of the demand by the load points (servers) (step 2) and the transmission of the energy production according to the demand received by the energy sources (step 3).

Step 2: it starts with the servers. They send their power demand $outDemand_{ser_i}$, $i = 1, \ldots, 12$, to the $PDUs$ of the *distribution layer*. This is modeled using the *SPLITTER-gate* in order to combine the total demand coming from servers.

From each PDU, the demand is propagated to a pair of TUs (block of LTM and DT). This is modeled using a *PLUS-gate* with a pro-rata strategy according to the capacity each one can treat. Since a TU block contains components in series, we use a *MIN-gate* to combine them.

The demand continues its traversal in the *distribution layer* to BDP_{iX}, $i = 1, 2$ and $X = A$. Each BDP_{iA} sends its demand to the *storage layer* which contains UPS_{iA}. A *MIN-gate* is used to model it. We model similarly the other path ($X = B$) of the system.

As each UPS_{iX}, $i = 1, 2$, $X = A, B$, has to send its demand to FDP_A and FDP_B in the same layer, a *SPLITTER-gate* is used to collect the sum of demands $outDemand_{UPS_{iX}}$. Then a *PLUS-gate* is used to propagate the total demand between two redundant paths (FDP_A and FDP_B).

Since FDP_A and FDP_B are is series with transformers Tr_A and Tr_B in the *transformation layer*, respectively, they are combined using a *MIN-gate*. Finally the demand is transmitted to the *production layer*. The demand $outDemand_{Tr_A}$ from Tr_A is sent to power source PS_1 and power generator PG with a priority to PS_1. This is modeled using a *PLUS-gate* with priority strategy. We model similarly the second path through Tr_B; a demand $outDemand_{Tr_B}$ is sent to power source PS_2 and PG with a priority to PS_2, the PG being initially in standby mode.

Step 3: according to the received demand, the *production layer* provides the

required energy to supply two paths. The first path is supplied by PS_1 and PG through the *PLUS-gates* with a priority to PS_1, and the second one with PS_2 and PG through the other *PLUS-gates* with a priority to PS_2. Then, in each path, the minimum between the energy coming from the *production layer* and the transformer's production is provided through the *MIN-gate* (one in each path). The energy production goes through the *MIN-gate* to the *storage layer* to get the minimum with FDP production. This energy production is then gathered from the two paths through the *PLUS-gate*, and splitted among $UPSs$ through the *SPLITTER-gates* using the pro-rata strategy according to their maximum capacity. Since UPS contains 3 components ($Conv$, Bat and Rec), the total energy production goes through *MIN-gates* to get the minimum of production between these components (one gate per component). Once the energy production reaches $BDPs$ in the *distribution layer*, their total energy production is gathered through the *PLUS-gates*, then divided between TUs using the pro-rata strategy. The energy production continues its traversal through gates of the *distribution layer*, until it reaches $PDUs$. At this point, each PDU supplies 4 servers, then the $PDUs$ energy production is divided between servers through the *SPLITTER-gates* using the pro-rata strategy according to their maximum capacity.

The complete model is presented in Figure 7.4. Note that as the electrical sub-system does not depend on any other sub-system of the DC, there is only one kind of flow in the system (electrical flow), so no *COND-gate* is required in the PT model. Furthermore, to simplify the graphical representation, only production flows are represented.

7.3.2 Modeling the thermal sub-system

The production tree modeling the thermal system has to catch the different interactions between the production and the treatment units. Moreover, it has to take into account dependencies between this sub-system and the electrical one, as thermal system components become operational only if they are powered by the electrical system. Thus the PT modeling the thermal sub-system has to take into account two different kinds of flows: air flow and electrical flow. For that, we use several

COND-gates to model the dependency behaviors like the output air flow dependency on, not only the capacity of the input air flow, but also the capacity of the input electrical flow.

Unlike the electrical sub-system, the thermal system has no particular behavior, and building the PT model goes through the 3 steps: the exportation of the capacity (bottom-up from the production sources), the exportation of the demand (top-down) and the exportation of the production (bottom-up).

Step 1: the first step is the transmission of production capacity of the system in terms of cooled air to the racks (servers). Assume that the source of water S will never fail and produces an infinite quantity of water. This source provides water to both cooling towers CT_A and CT_B which will export their production capacities $outCapacity_{CT_A}$ and $outCapacity_{CT_B}$, respectively, under the condition that they are powered by the electrical system. This is modeled using two *COND-gate*, one for each cooling tower. The first child of each gate provides the real production capacity of the cooling tower while the second one provides the electrical production coming from the PT modeling the electrical sub-system.

Since the water production of the CTs is the minimum between K_1 (their intrinsic capacity of production) and K_2 (their power consumption), the associated function f is defined as $min(inCapacity_{K_1}, inCapacity_{K_2})$. It follows that:

$$outCapacity_{K_1} = min(inCapacity_{K_1}, inCapacity_{K_2}) \tag{7.1}$$

The power to CT_A and CT_B is transmitted by FDP_A and FDP_B, respectively (see Figure 7.3). In order to simplify the graphical representation and prevent duplicating sub-branches at multiple tree locations, transfer functions represented by triangles 1 and 2 are used in Figure 7.5 to refer to the power path from FDP_A and FDP_B, respectively.

Then, since the water production of the cooling system is the sum of the water production of CT_A and CT_B, the outputs of both *COND-gate* are combined using a *PLUS-gate* with two input flows $outCapacity_{CT_A}$ and $outCapacity_{CT_B}$.

Since the chillers are on two redundant paths, the output capacity of the *PLUS-gate* is propagated unchanged to them. In the PT, this is modeled using a *SPLITTER-gate*. Each output of this gate becomes then one of the two inputs of a

MIN-gate, the other input being the intrinsic capacity of a chiller.

Moreover, in order to chill the water coming from the cooling towers, chillers Ch_A and Ch_B need to be powered by the electrical system. This is done through electrical components PDU_{1A} and PDU_B, respectively. In the PT model, this is captured by two *COND-gates*, one for each chiller. The power paths from PDU_{1A} and PDU_B are represented using transfer functions 3 and 4, respectively.

As both chillers have then to route the chilled water to CRAC units (servers) in the system, we first combine the production capacities $outCapacity_{Ch_A}$ and $outCapacity_{Ch_B}$ coming from Ch_A and Ch_B, respectively, using a *PLUS-gate*. A *SPLITTER-gate* is then used. By definition this gate will propagate the production capacity unchanged to the CRAC units.

Moreover, as a CRAC unit requires 1 $Watt$ to extract the air from 1 *liter* of chilled water, we use a *COND-gate* which output capacity is given by Equation 7.1. The other input to this gate is the output of a *MIN-gate* between the CRAC unit (intrinsic capacity) and the *SPLITTER-gate* output used to propagate the production capacity from chillers.

Step 2: once a server Ser_i, $i = 1, \ldots, 80$ has been informed about the production capacity of cooled air, it sends its demand $outDemand_{ser_i}$. This demand is propagated unchanged to the cooling towers through corresponding two $CRAC$ units and chillers. The server demand continues its traversal through the same gates as the production capacity until it reaches the thermal production source, that is the cooling towers. Note that, when a demand is routed through a *COND-gate*, the required power demand $outDemand_{K_2}$ is sent to the electrical system through its corresponding PT model.

Step 3: the third and final step is the transmission of the cooled air to the servers. According to the demand received by each cooling tower, and in order to satisfy this demand, the quantity of water is divided between the chillers (prorata strategy). Then the chilled water produced by the chillers is sent to the CRAC units. At this step, the first *SPLITTER-gate* will divide the production between the CRAC units according to their demands. Similarly the CRAC units will provide the cooled air to the servers (racks). The last *SPLITTER-gate* will divide the production

between the servers according to their demands.

The complete model is presented in Figure 7.5. Note that, for all gates of type *COND-gate*, since the thermal components production output (K_1) is equal to their power consumption (K_2), the associated function f is defined as:

$$outProduction_{K_1} = f(inProduction_{K_1}, inProduction_{K_2})$$
$$= min(inProduction_{K_1}, inProduction_{K_2})$$

To simplify the graphical representation, only production flows are represented.

7.3.3 Modeling the network sub-system

As the users of the DC'system send directly their demands to the network components, only the last 2 steps are required, like in the electrical sub-system.

In the following, only Step 3 is detailed. The demand transmission (Step 2) follows a similar reasoning but in the opposite direction (top-down).

The production transmission (Step 3) represents the service provided by the racks according to the received demand. All racks export their responses (packets production in PT) $outProduction_{Rack_i}$, $1 \leq i \leq 4$, to the corresponding ToR switches to which they are directly connected, under the condition that they are powered by the electrical system. This is modeled using eight *COND-gates*, one for each rack and switch (see Figure 7.6, *Layer* 3 and *Layer* 4). The associated function f with each *COND-gate* is defined as $min(inProduction_{K_1}, inProduction_{K_2})$, K_1 and K_2 being the treatment capacity of the component (rack or switch) and the electrical flow, respectively. Then, since each rack is connected to a ToR switch, each output of the *COND-gate* becomes one of the two inputs of the *MIN-gate* to get the minimum packets production.

The total packets production of *Layer* 3 and *Layer* 4, which is the sum of the productions of ToR switches, is exported. Thus, the output of each *MIN-gate* is combined with *PLUS-gate* with four input flows $outProduction_{ToR_i}$, $1 \leq i \leq 4$. Then, since $AggS$ switches are on two redundant paths, the output production of *Layer* 3 and *Layer* 4 is divided between them through the *SPLITTER-gate* with a pro-rata strategy (see Figure 7.6, *Layer* 2). Each output of this gate becomes then one of the two inputs of the *MIN-gate*, the other input being the intrinsic

capacity of $AggS_A$ and $AggS_B$, respectively. Moreover, in order to route the traffic from ToR switches, $AggS_A$ and $AggS_B$ need to be powered by the electrical system. This is captured by two *COND-gates*, one for each. The total packets production from *Layer* 2, which is the sum of the $AggS$ switches productions, is transmitted. Thus, the output of each *MIN-gate* is combined with a *PLUS-gate* with 2 input flows $outProduction_{AggS_A}$ and $outProduction_{AggS_B}$.

Then, the packets production from *Layer* 2 continues its traversal through the corresponding gates until it reaches *SPLITTER-gate* of *Layer* 1 which divides the flow between $AccR$ routers. Each output of this gate becomes then one of the two inputs of the *MIN-gate*, the other input being the intrinsic capacity of $AccR_A$ and $AccR_B$, respectively. The total packets production is then captured by *PLUS-gate* combining the output of each *MIN-gate*. Once again, as for $AccR$ routers, the total production is divided between $Core$ routers through the *SPLITTER-gate* with a pro-rata strategy (see Figure 7.6, *Layer* 1). Each output of this gate becomes then one of the two inputs of the *MIN-gate*, the other input being the intrinsic capacity of $Core_A$ and $Core_B$, respectively.

Finally, the output flow of each *MIN-gate* is the flow received by the user which corresponds to the packets production.

The PT model is presented in Figure 7.6, where, due to a high number of servers, these are not represented. The racks (containing 20 servers each) are represented. Moreover, to simplify the graphical representation, only production flows are represented.

7.4 Conclusion

In this Chapter, we presented a case study of a real DC's system. We described first the specific architecture of each DC's sub-system. Then we illustrated how to model the system using the production trees modeling technique. We showed also how our PT extended version allows modeling dependencies between flows circulating in the system. In the next Chapter, we will assess the obtained PT model in order to estimate different safety and performance indicators of the system under study.

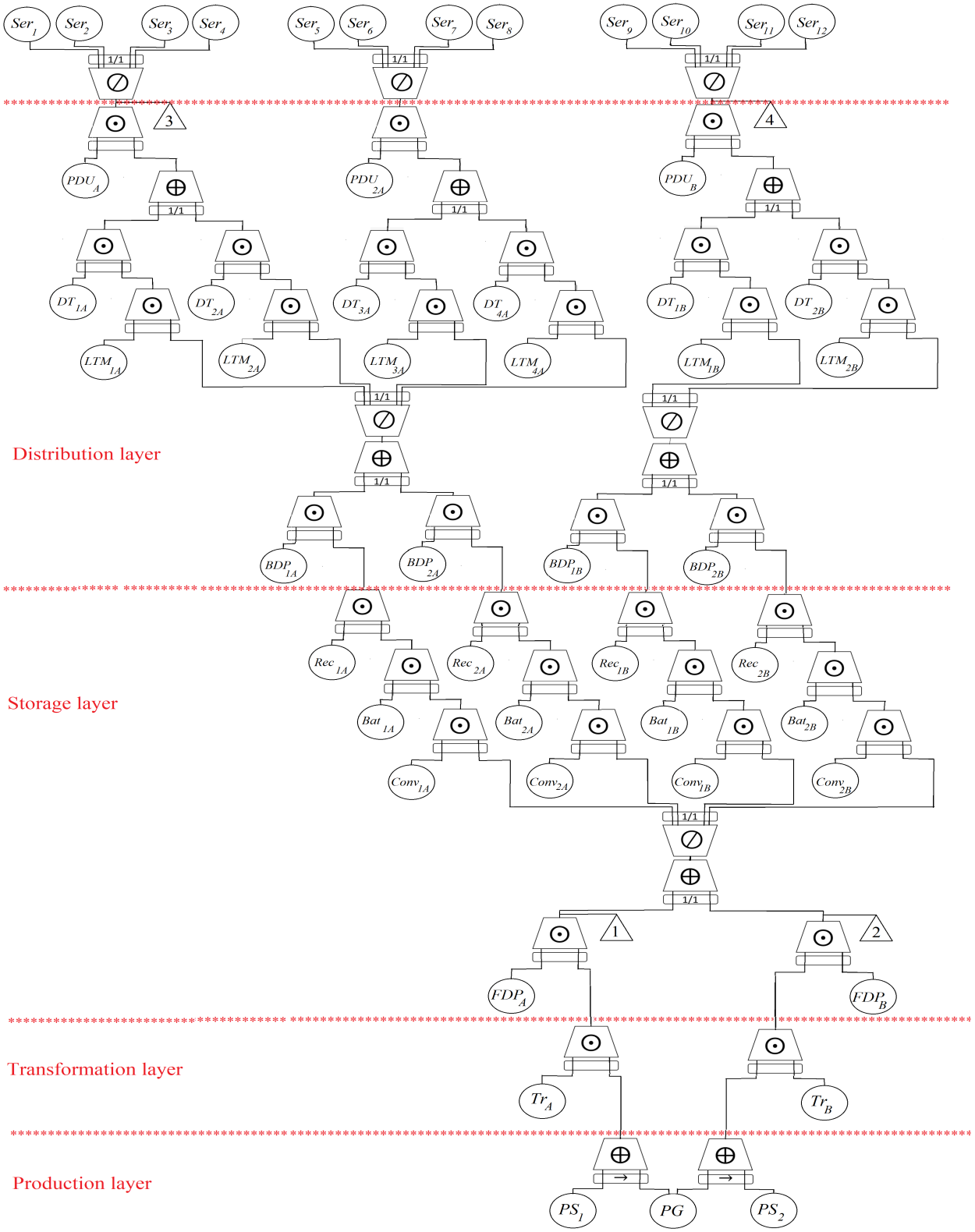

Figure 7.4: PT of the Electrical sub-system

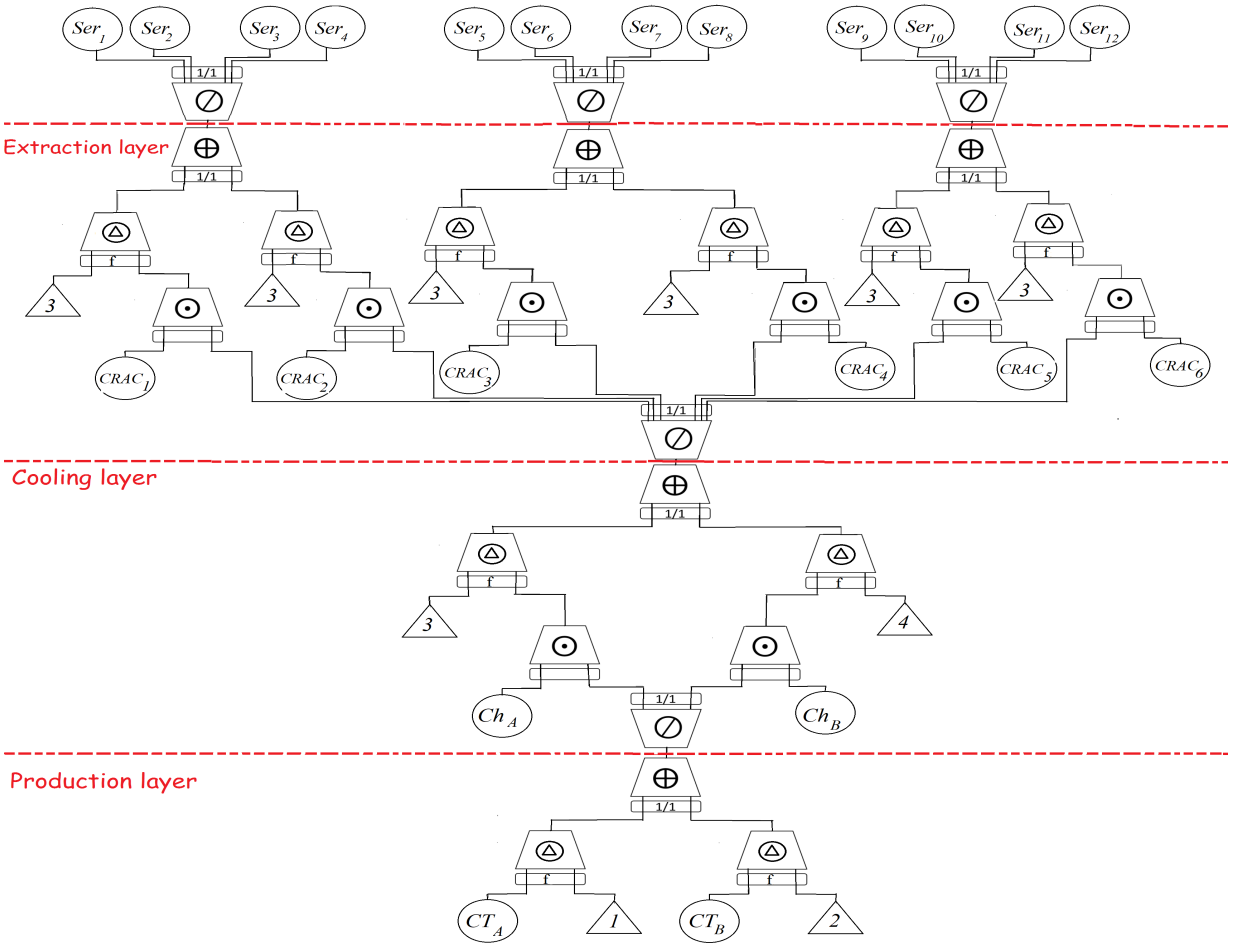

Figure 7.5: PT of the thermal sub-system

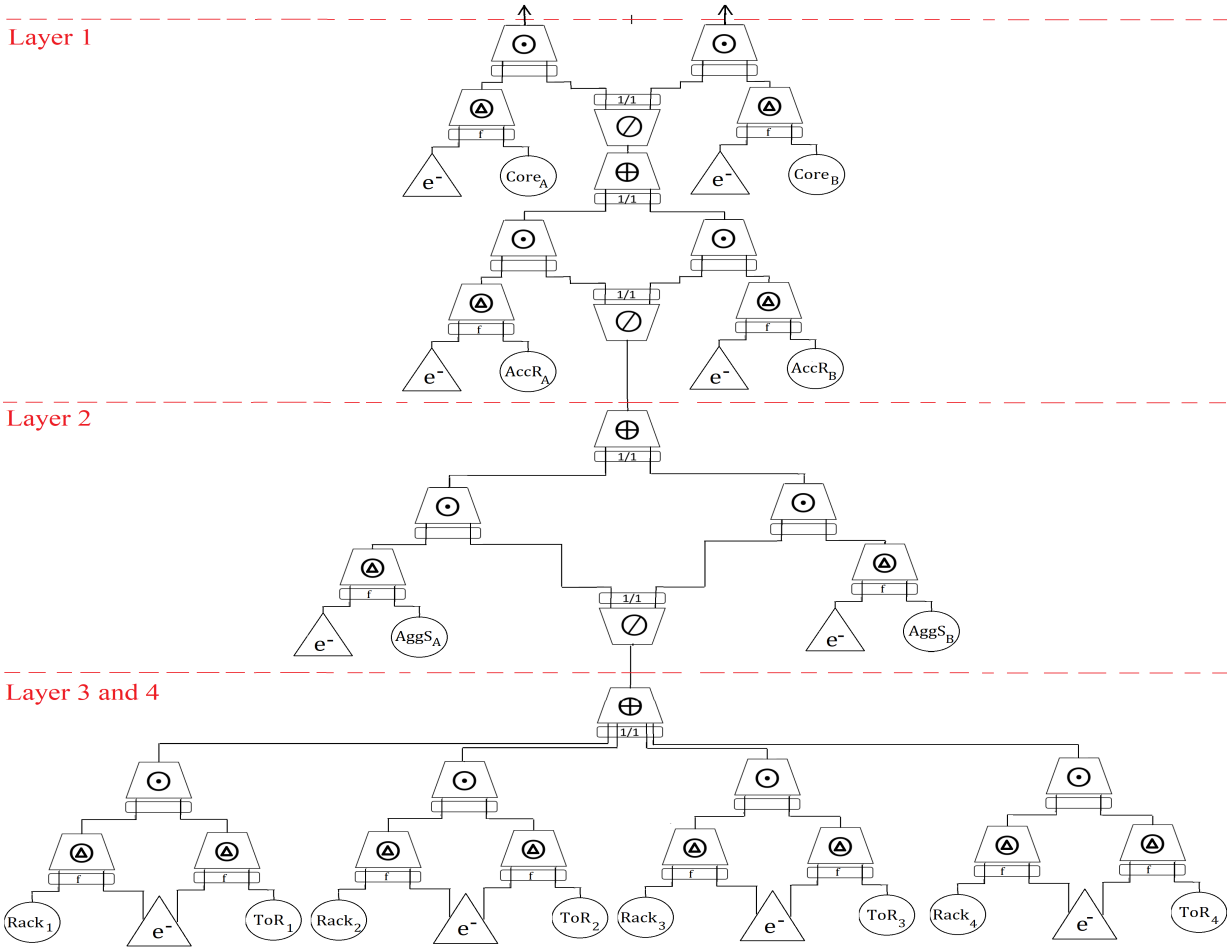

Figure 7.6: PT of the network sub-system

Chapter 8

Safety and Performance Assessment

8.1 Introduction

In this Chapter, we propose a methodology which allows analyzing both safety and performances of the DC's system described in Chapter 7. For safety analysis, it is important to solve the obtained PT modeling this system. In order to estimate safety indicators of the DC's system, we apply the assessment algorithm proposed in Chapter 6 on the PT modeling the network sub-system (see Figure 7.6). Indeed this model involves the PT modeling the electrical sub-system through the *COND-gates* used. For the thermal sub-system, it is invoked when analyzing the impact of temperature variations on network sub-system's components by applying the *Arrhenius model*. This model allows to know whether the total heat within the IT room is extracted or not. The obtained results are compared with those obtained with the AltaRica 3.0 model.

For performances analysis, some indicators are estimated by making statistics on packets flows circulating between the DC's network sub-system components. The obtained results are compared with those obtained with the simulation tool we have implemented for queueing networks.

This chapter is structured as follows. In Section 8.2, we present data used for the different safety and performance indicators we want to estimate. Section 8.3

summarizes the obtained results. Finally, Section 8.4 concludes this Chapter.

8.2 The model analysis methodology

Our methodology combines an analysis of both safety and performances of the DC's system [35]. The complete process is illustrated in Figure 8.1.

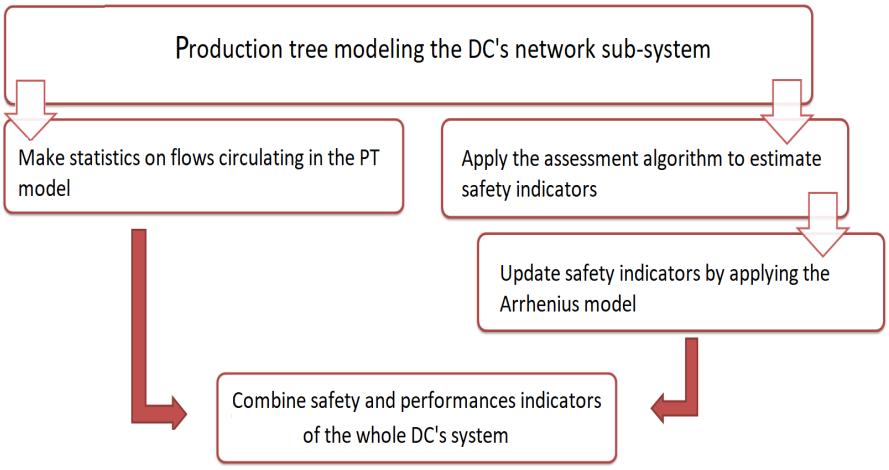

Figure 8.1: The DC's system safety and performance analysis methodology

a) Safety: we apply our assessment algorithm (see Chapter 6) to solve the PT modeling the network sub-system of the DC. Because of the dependencies between the DC's sub-systems, the application of the assessment algorithm automatically involves the resolution of the PT modeling the electrical sub-system through the *COND-gates*. The obtained safety indicators are updated by analyzing the impact of the temperature variations on the DC's network sub-system. To analyze this impact, we consider the Arrhenius model.

Arrhenius model relates the lifetime of an electronic component to the operating temperature [4]. The following equation estimates the relationship between this temperature and the Mean Time To Failure (MTTF) of the device:

$$r = A * e^{(\frac{-E_a}{K*TP})}. \tag{8.1}$$

Where:

- r is the reaction rate.
- TP is the temperature (in degrees Kelvin) at which components breakdown.
- K is the Boltzmann constant.

- A is a pre-exponential constant.
- E_a is the activation energy usually within the range $0.3eV - 0.7eV$ [7].

In this thesis works, we consider the activation energy $E_a = 0.642eV$ which is generally used for cooling components [9]. We consider also the initial temperature TP_0 of the IT room, such that $TP > TP_0$. This leads to the following MTTF expression:

$$MTTF_{TP} = MTTF_{TP_0} * e^{(\frac{E_a}{k} * (\frac{1}{TP} - \frac{1}{TP_0}))} \tag{8.2}$$

Equation 8.2 allows us to compute a new value of MTTF at elevated temperature [55]. The revised MTTF provided by the Arrhenius model is inserted into the PT model [33].

Moreover, we take into account not only the thermal sub-system impact but also the electrical one in terms of power energy demand. Therefore, the system is analyzed by considering the dynamical impact of the thermal sub-system on the electrical one. So when the thermal sub-system demand exceeds the electrical sub-system capacity, the former may not fail, since the latter may adapt its production capacity by producing more energy to satisfy the demand. When the demand is less than the production capacity, this production can be reduced in order to optimize the energy consumption by the thermal sub-system.

In order to validate the results of our approach, we implement the system under study using the AltaRica 3.0 modeling language [79] and use its dedicated stochastic simulator. The AltaRica 3.0 assessment tool estimates the reliability indicators by simulating the actual behavior of the system in order to create a realistic life time scenario of the system. A set of 1000 histories and a time limit of 36000 *hours* were performed.

b) Performance: once the system reliability is estimated, we analyze the performance of the most important part of DC system responsible for providing services which is the network sub-system (see Figure 7.3), taking into account its dependencies with the other sub-systems (electrical and thermal). Generally Queueing Network (QN) theory is used to analyze performance of such systems. However, this technique does not take into account the components failure. In our case, we

estimate the performance of the network sub-system, knowing that each of its components can fail. For that, we enrich the PT modeling the network sub-system depicted in Figure 7.6 by introducing performance measures on flows circulating in the model.

PT models, with their basic components and gates, are sufficient to deal with both DC reliability and performance issues. Instead of having a deterministic flow propagation like in a QN model, the propagation is dynamic in a PT model, according to the state of each network sub-system component (working or failed) or according to its treatment capacity. Moreover, thanks to *COND-gates*, the functional dependencies between the DC sub-systems can be modeled.

The most important performance statistics estimated are the total network throughput, the mean end-to-end delay and packet loss probabilities. At each basic component i in the PT model, packets arrive at rate λ_i and leave with rate μ_i which corresponds to the precessing rate P_{r_i}. Two cases are considered:

1) Case 1: when the processing rate P_{r_i} at component i is greater than the arrival rate λ_i, the queueing delay at any component of the network is null. The processing delay is constant for all packets $D = D_i = 1/P_{r_i}$ (assuming that all packets have the same treatment time).

2) Case 2: when the arrival rate λ_i is greater than the processing rate P_{r_i}, packets experience queueing delays. And since the buffer size is bounded, packets may be lost. We note:

- the sending interval $S_i = 1/\lambda_i$,
- the processing time $T_i = 1/P_{r_i}$,
- the number of packets $N_i(t)$ in the buffer at instant t,
- the size of the buffer K_i,
- the packet loss probability $P(t)$ at instant t.

A component i sends M packets every S_i time interval. The first of these packets reaches another component at instant t_d where t_d is the transmission delay. Then the instant of arrival at component i for packet j is $A_{ij} = t_d + j * S_i$ where $0 \leq j \leq M - 1$.

The instant of processing of a packet j at component i is $P_{ij} = max(A_{ij}, j*T_i)$. Thus for $j = 0$, $P_{ij} = A_{ij}$.

At instant t_K the number of packets $N_i(t_K)$ in the buffer i reaches the size limit K_i, and the packets arriving after this instant are lost. Therefore, the packet loss probability is $P(A_{ij} \geq t_k)$.

Queueing delay for packet j at component i is $Q_{ij} = P_{ij} - A_{ij}$ and the total delay is $D_{ij} = Q_{ij} + T_i$. The average delay D:

$$\begin{aligned} D &= (P_0 - A_0) + \sum_{j=0}^{M-1} (j * (T_i - S_{ij}) - t_d + T_i)/M \\ &= (P_0 - A_0) + ((M-1) * M/2) * (T_i - S_{ij}) - t_d + T_i)/M \end{aligned}$$

And since $P_0 = A_0$:

$$D = ((M-1) * M/2) * (T_i - S_{ij}) - t_d + T_i)/M$$

Finally, in order to validate the performance results of our approach, we compare them to simulation results. We have implemented a simulation tool for an open finite QN where each queue is a $M/M/1/K$ [101]. We use a confidence interval for the admission decision of 95%.

8.3 Numerical Results

We consider the components reliability data in [7]. The servers in each rack are connected to $ToRs$ via a $1Gbps$ link. The processing rates (treatment capacity) of the network components are real data (provided in Table 8.1). For confidentiality reasons, the reference cannot be provided. In the PT model, we consider the mechanism of Skipping the Unavailable Nodes (SUN) [41]. Flows are not allowed to enter a failed component and jump to the next one according to the routing table (redundancy case).

Component i	P_{r_i} **(Mbps)**
$Core$ and $AccR$	450
$AggS$ and ToR	400
$Server$	716

Table 8.1: Components treatment capacities

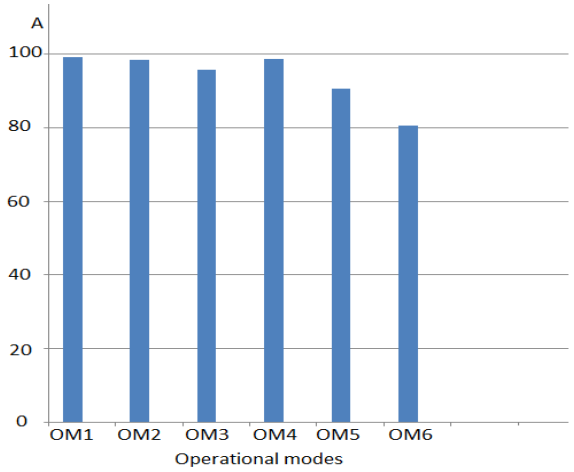

Figure 8.2: System availability according to the operational modes

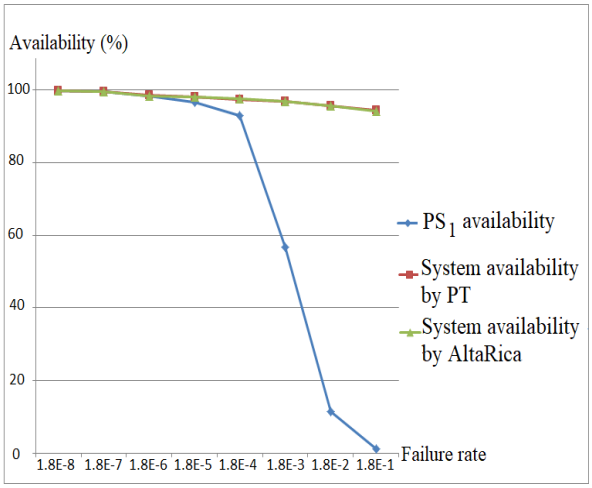

Figure 8.3: PS1 and system availability in OM_5

In this thesis work, to compute the availability A, we consider the six following operational modes.

• OM_1: in this mode, the availability is computed assuming that the system is operational if at least one server in a rack is working, the others servers and racks are in standby mode (knowing that the server demands $10kW$ of power energy).

• OM_2: the system is considered to be working if at least one rack is working, that is, all servers in this rack are working ($10kW$ for each server in the rack).

• OM_3: the system is considered to be working if all racks (including their servers) are working. Note that if one of the servers within a rack fails, the rack is considered as failed.

• OM_4: the system is operational if at least one server in a rack is working with $30kW$ of energy demand.

• OM_5: the system is considered to be working if at least one rack is working ($30kW$ of demand in each server).

• OM_6: the system is considered to be working if all racks are working with the same energy demand from each server.

Figure 8.2 is a summary of the results of system availability evaluated separately in each operational mode. This figure shows that the system in modes OM_1, OM_2 and OM_4 has the highest availability. In these cases the system generates a

sufficient power allowing the servers to operate properly, and there is redundancy between racks. The system in OM_3 and OM_5 has a lower availability. In OM_3, this is due to no redundancy between components, because if one rack fails, no other rack will take over. However, in OM_5, this is due to the increase of power consumption (from $10kW$ to $30kW$), and the power sub-system reaches the maximum of its production. The system in OM_6 has the lowest availability (80%), because of the increase of the power demand and no redundancy. The system produces more power by activating the PG (Power Generator), initially in standby mode, and if one rack fails, there is no rack that will take over.

Figures 8.4 (a) and (b) show the probability distribution of the total electrical sub-system production and the probability distribution of air production by the CRAC units, respectively, when OM_5 is considered. According to Figure 8.4 (a), the electrical sub-system produces $60kW$ of energy with a high probability. This is sufficient to satisfy the servers demand. However, according to Figure 8.4 (b), the CRAC units are able to extract $50kW$ with a high probability and $60kW$ with a null probability. Therefore, the servers' power demand is satisfied but the total of the heat is not extracted from the data center room. This is consistent with the results obtained in Figure 8.2 and explains why the availability of the system, when OM_5 is considered, decreases.

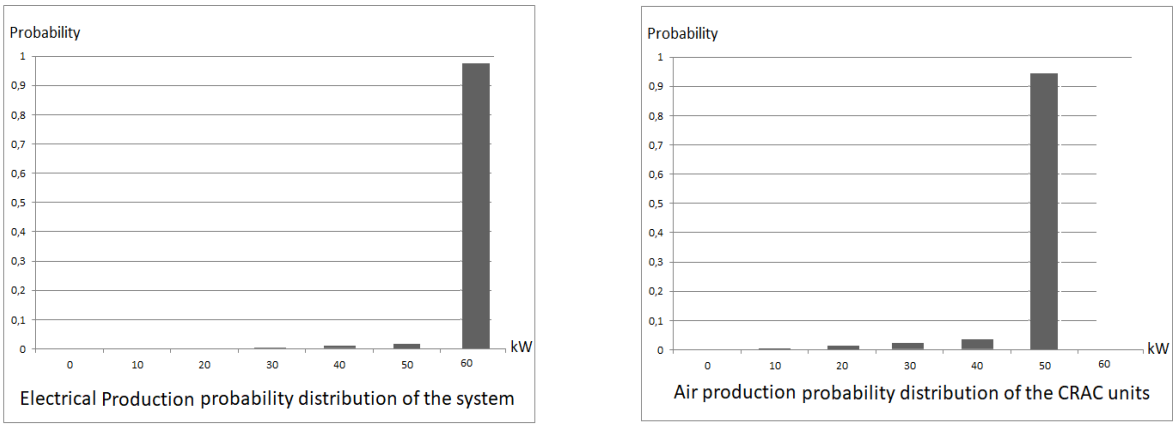

Figure 8.4: Probability distribution of the production capacity in OM_5

In order to identify the different dependencies between components and sub-

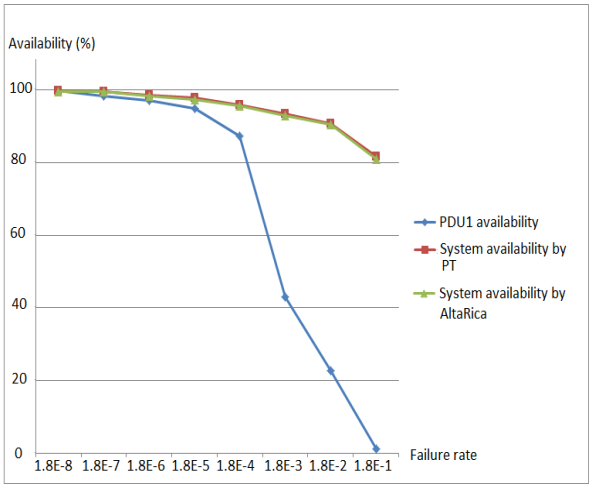

Figure 8.5: PDU and system production availability in OM_5

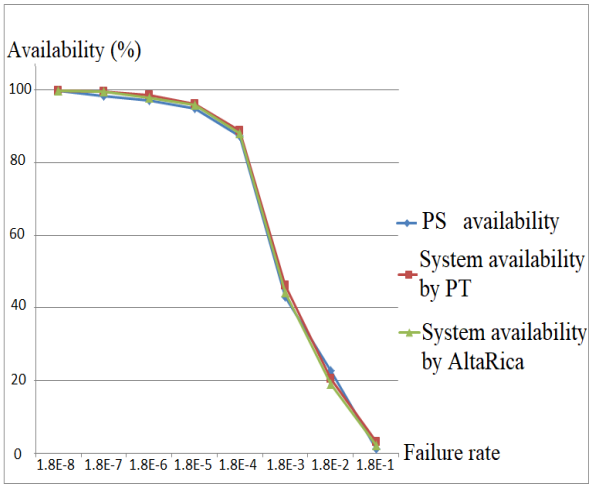

Figure 8.6: Server and system production availability in OM_5

systems, we study the variation of component's failure rates and its impact on the global availability of the system in OM_5. Let's start with the power sources PS_1 and PS_2. Figure 8.3 provides the variation of failure rates of the power source PS_1 (same for PS_2) and its impact on the system availability. When the failure rate varies between $1.8 * 10^{-8}$ failures/h and $1.8 * 10^{-4}$ failures/h, the system's availability is not impacted. Indeed although the availability of the power source decreases when the failure rate increases, the power production is ensured by both PS_2 and PG (PS_1 and PG if PS_2 is considered). However from $1.8 * 10^{-3}$ failures/h the system's availability is impacted, because PS_1 fails more frequently (approximately every 10 hours), and the power sub-system is at its maximum of production (system in OM_5). So even if the power production is ensured by both PS_2 and PG, it's not enough when PS_1 (same for PS_2) fails frequently. The simulation results of the AltaRica model match those obtained using PT model.

The $PDUs$ represent the distribution points of the power flow to the servers. This is why it is important to analyze their failure impact on the system in OM_5. According to the results presented in Figure 8.5, the variation of failure rates of a PDU unit does not affect too much the system availability due to redundancy. Once again, the simulation results of the AltaRica model match those obtained using PT model.

Finally, if we analyze the impact of the servers failure rates variation on the system in OM_5 (Figure 8.6), it is clear that the system availability depends on the servers availability, because the proper functioning of a DCs system is mainly based on servers availability which provide services. The simulation results of the AltaRica model match those obtained using PT model.

It is clear from the previous results that failure rates variation of the power sources (PS_1 and PS_2) impact the system availability. Indeed when PS_1 fails more frequently (from $1.8 * 10^{-3}$ failures/h to $1.8 * 10^{-1}$ failures/h), the system availability decreases strongly. Considering this value range of failure rates, we study their impact on both the electrical sub-system average production (Figure 8.7 (a)) and the thermal sub-system average production (Figure 8.7 (b)), when OM_5 is considered. The electrical sub-system in a normal operational mode produces $50kW$ with a very high probability (Figure 8.4 (a)). However when PS_1 (or PS_2) fails frequently, the system is unable to produce this energy (a null probability). The electrical production varies between $0kW$ and $40kW$, which is not sufficient to satisfy the servers' demand. As a consequence, the thermal sub-system (CRAC units) demand is also not satisfied, which impacts its cooled air production. Thus the thermal sub-system is not able to extract heat from the DC room which leads to a temperature rise.

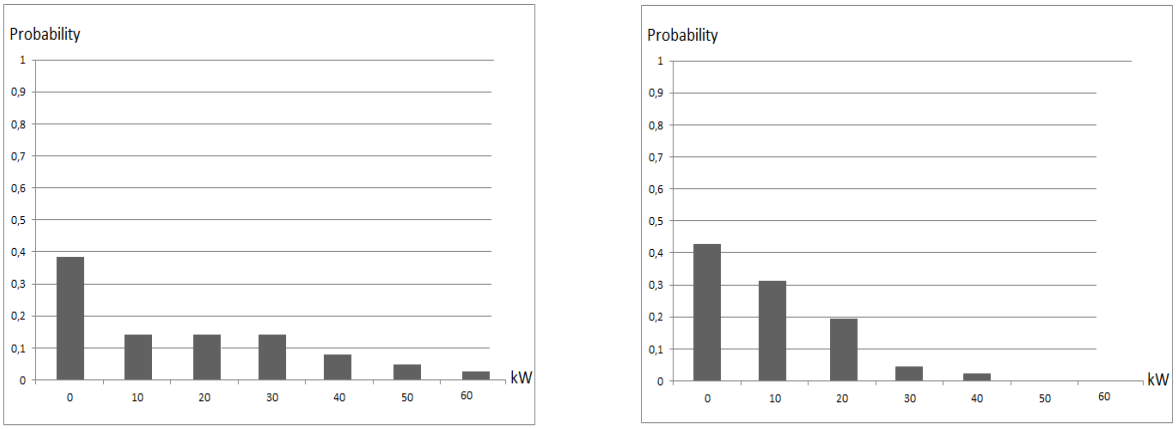

Figure 8.7: The impact of PS_1 failure rate on probability distribution of the production capacity in OM_5

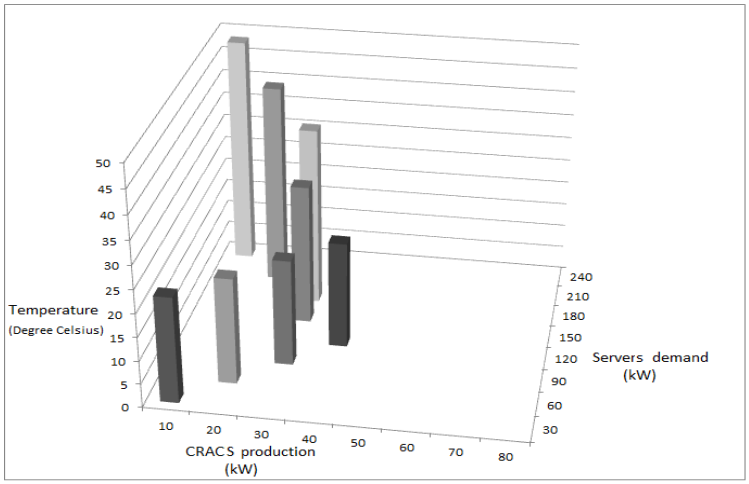

Figure 8.8: Electrical sub-system impact on thermal sub-system

Figure 8.8 shows the impact of the servers demand variation on the CRAC unit's production and thus the temperature in the IT room. The servers demand varies between $30kW$ and $240kW$. When the demand is less than $120kW$, the temperature remains unchanged because the CRAC units are working well and extract the total heat from the IT room. When the demand exceeds $120kW$, the CRAC units production starts decreasing progressively because the electrical sub-system is not able to produce a sufficient energy to satisfy the CRAC (thermal) demands, but can produce sufficient energy for the servers. In this case, the servers are fully powered, but the CRAC units' demand is not satisfied. For example, the servers need $180kW$ of power, and the electrical sub-system produces $200kW$. The servers consume $180kW$ then generate $180kW$ of heat and the CRAC units will need $180kW$ of power to extract this heat. However, only $20kW$ (the rest of energy produced) is transmitted to the CRAC units. Therefore, only $20kW$ on $160kW$ of heat are extracted. This leads to an excess of the normal DC temperature (25 °C) in the IT room. Clearly, the increase of temperature affects the servers and the system availability.

Figure 8.9 shows the system reliability according to the demand arrival rate λ_i. For confidentiality constraint, the failure rates data we use for the DC's system components cannot be provided.

According to Figure 8.9, the probability that the system will perform its func-

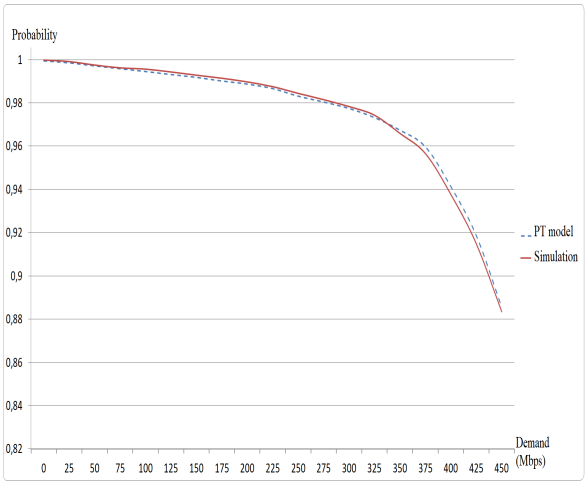

Figure 8.9: System reliability

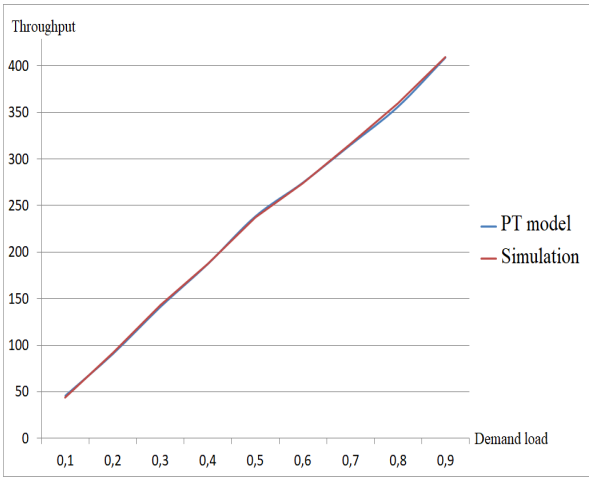

Figure 8.10: The total system throughput

tion over $1year$, which corresponds to produce a sufficient throughput to treat demands from users, is high due to the redundancy. Then the probability starts decreasing slightly until 0.96 approximately, which corresponds to $400Mbps$ of demand. This is due to the electrical sub-system impact. As shown above, a component is initially *idle* and becomes *active* when it receives a request. Therefore, when the demand increases, the power consumption increases too, and this leads to the decrease of the system reliability. Indeed, a switch can handle a number of packets according to its maximum capacity ($400Mbps$). When this capacity is reached, the second switch, initially in standby mode, is activated and starts receiving packets. Thus, the failure probability increases because no other switch will take over in case of failure (no redundancy). This explains why the reliability starts decreasing strongly from $400Mbps$ of demand. The results obtained using PT model match very well those obtained using simulation (QN model).

Figure 8.10 shows the total throughput of the network sub-system. The throughput increases according to the demand load, and reaches its maximum ($450Mbps$) which corresponds to the maximum capacity of both routers $Core_A$ and $Core_B$. Once again, the results obtained using PT model match very well those obtained using simulation.

Figure 8.11 and Figure 8.12 show the packet loss probability and the mean

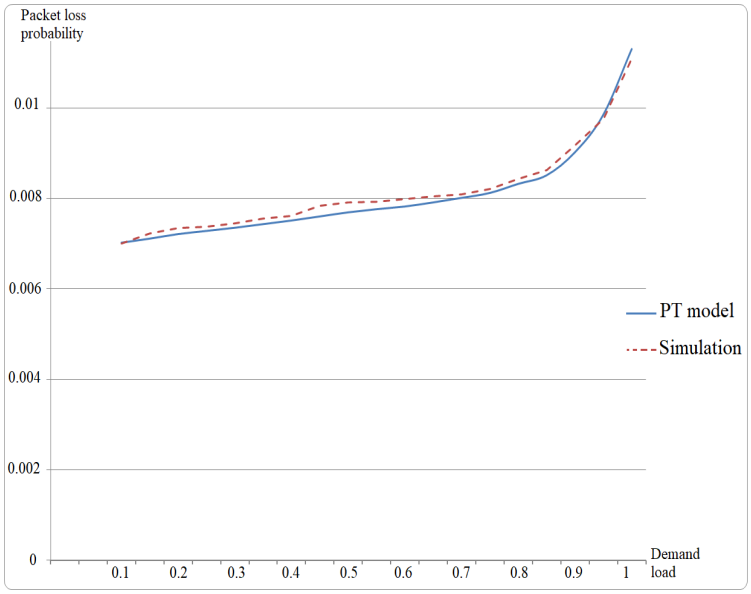

Figure 8.11: System packet loss probability

delay, according to demand load (fractions of 450 Mbps), respectively. Clearly, the mean delay as well as the packet loss probability increase when the demand increases. From a load of 0.86, which corresponds to approximately $400Mbps$, the curves have the same behavior change as both start increasing strongly. This is the point where the network reliability decreases strongly in Figure 8.9. This shows that the switches are responsible for the packets losses. And since there is a retransmission of lost packets, the mean delay increases too. Once again, the results obtained using our approach match very well those obtained using simulation.

Clearly, the obtained results show that the switches impact the reliability as well as the mean delay of the system. This conclusion is confirmed by the results in Figure 8.13 and Figure 8.14 which provide the impact of the switches reliability on the global system reliability and the mean delay, respectively. The impact is important and the network sub-system depends essentially on the switches reliability.

As a DC system has to ensure a continuous service with high performances (in terms of throughput and delay), the obtained results are promising in order to identify the components (or sub-systems) impacting this objective. The DC components which impact more the whole system can be improved as well as the whole system can be redesigned in order to meet the required demand in terms of

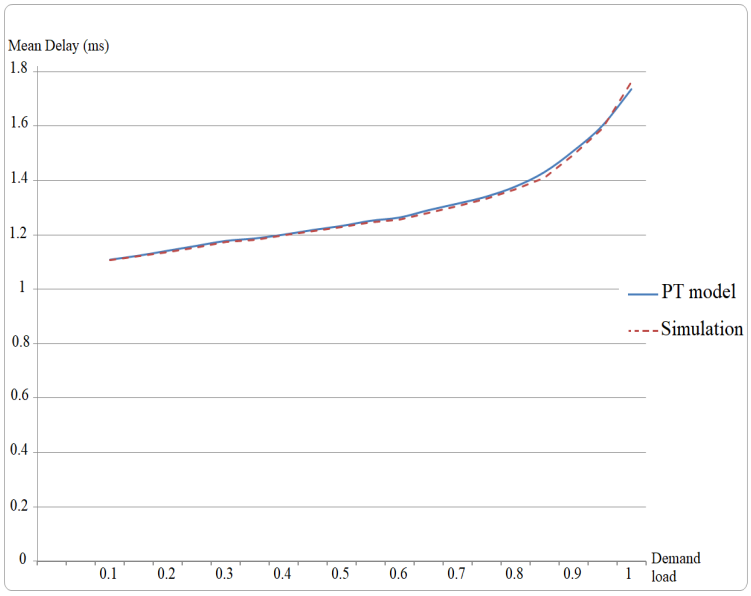

Figure 8.12: The Mean delay in the system

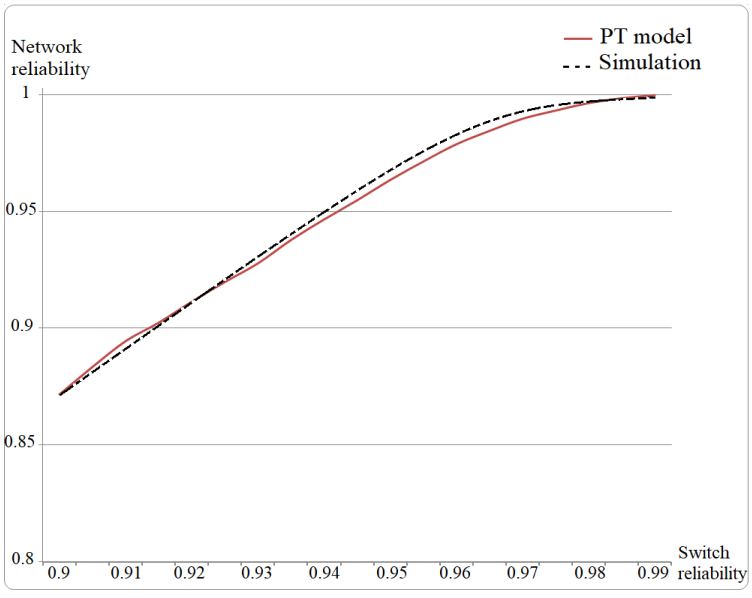

Figure 8.13: Switch reliability impact on the system reliability

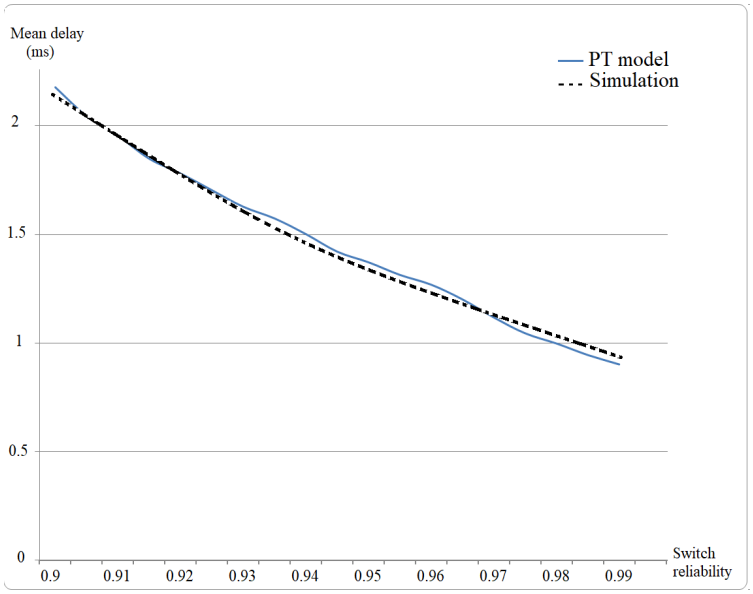

Figure 8.14: Switch reliability impact on delay

throughput and delay.

8.4 Conclusion

In this chapter, we investigated the reliability and availability of a DC system using Production Trees. We showed how easily this modeling technique allows taking into account, not only the different flows circulating in a DC, but also the dependencies between its sub-systems (electrical, thermal and network). We took into account the dynamic aspect of these dependencies as the DC's sub-systems have an impact on each others dynamically. We have also showed how this technique helps analyzing both reliability and performance of the system. The comparison of our results with those obtained using, on the one hand, the AltaRica stochastic simulator (electrical and thermal sub-systems) and, on the other hand, the QN-based simulation tool (network sub-system), shows a promising effectiveness of this integrated method.

Chapter 9

Conclusion

In this thesis, we proposed a methodology to analyze both safety and performances of a Data Center's system.

First we gave a description of the Data Center System. It is complex system with three sub-systems (system of systems). The sub-systems (electrical, thermal, network) are themselves complex and interact between them. Indeed the proper functioning of a DC system is based on the continuity of the services provided by the equipments of the network sub-system. And in order to ensure a constant service, these equipments must be provided with a sufficient and continuous power energy (the electrical sub-system), and kept in a constant and acceptable temperature (the thermal sub-system). A failure or a breakdown of one of the three DC's sub-systems can lead to the unavailability of the whole DC system, and this can be fatal for a company.

Considering the economics stakes of the DCs, and their increase of complexity, safety analysis of these systems become more and more crucial. In this thesis works, we have proposed a tool-supported model-based methodology to analyze both safety and performances of the DC system. Indeed, in our knowledge, no safety and performance study, taking into account the whole Data Center's system, exists. The developed methodology is based on Production Trees modeling technique and allows assessing different performances and safety indicators.

Production trees technique allows modeling the relationships between a system components with a particular attention to the flows circulating between these com-

ponents. Because of the interactions between a DC sub-systems in terms of flows, and the current PT modeling technique deals with only one kind of flow at once, we have extended this technique by introducing a new modeling mechanism. This one allows dealing with dependencies between the different types of flows circulating in the DC's system. We proposed a solution method to assess the PT modeling a system which allows estimating reliability and availability.

Moreover we showed the applicability of the PT modeling technique on a real DC system, and how this technique helps analyzing both reliability and performance of the system. The proposed solution for PT assessment is more restricted and provides more accuracy in terms of system availability and reliability values.

Finally, we developed a graphical tool for our methodology. This is an interactive interface which allows creating, editing and analyzing PT models.

There are several short and long time perspectives that will be interesting to deepen in the scope of this research work:

- Integrate thermal indices to Production Trees models to detect and identify random hot temperatures inside the Data Center called Hotspots.
- Introduce other factors that can impact the system availability such as the servers virtualization. Virtualized servers consume more energy than physical ones, and this can affect the electrical sub-system power energy production, which in turn can impact the global system availability.
- Optimize the resolution algorithm for Production Trees models in terms of time execution for very large systems.
- Improve our tool by integrating another graphical interface to design systems directly. The Production Tree modeling the system will be automatically generated.

Appendix A

Résumé

Résumé

Un Data Center DC est un bâtiment dont le but est d'héberger des appareils informatiques pour fournir différents services. Ces appareils ont trois besoins essentiels : un espace physique, une énergie électrique industrielle et une production constante en air froid. Ainsi, un DC peut être considéré comme un système complexe avec 3 sous-systèmes différents : électrique, thermique et réseau. L'espace physique est un endroit où se trouvent différents appareils informatiques. Leurs interconnexions forment un réseau important. Pour assurer un fonctionnement constant de ces appareils, l'énergie est fournie par le système électrique, et pour les maintenir à une température constante, un système de refroidissement est nécessaire. Chacun de ces besoins doit être assuré en permanence, car la conséquence d'une panne de l'un d'eux entraîne une indisponibilité de l'ensemble du système, ce qui peut être fatal pour une entreprise. Par exemple, une coupure de courant de 10 secondes peut entraîner une interruption de service de 10h, et une minute d'interruption peut coûter plus de 7000 euros. Les DCs sont donc construits pour répondre à de fortes contraintes de continuité de service. Ces contraintes peuvent représenter 50 % du coût des DCs, soit plusieurs milliards d'euros.

A notre connaissance, dans la littérature, il n'existe pas d'études de sûreté et de performance, prenant en compte l'ensemble du système du DC avec les différentes interactions entre ses sous-systèmes. Les études d'analyse existantes sont partielles et se concentrent sur un seul sous-système, parfois deux. L'objectif principal de cette thèse est de contribuer à l'analyse de sûreté de fonctionnement d'un DC. Pour ce faire, nous étudions, dans un premier temps, chaque sous-système du DC (électrique, thermique et réseau) séparément, afin d'en définir les caractéristiques. Cette étape est très importante pour trouver la technique appropriée pour évaluer les différents paramètres de sûreté (fiabilité et sécurité).

Chaque sous-système du DC est un système de production et se compose de combinaisons de composants qui transforment des entrées (énergie pour le système

électrique, flux d'air pour le système thermique, et paquets pour le réseau) en sorties, qui peuvent être des services Internet. Actuellement, les méthodes d'analyse de sûreté existantes pour ce type de systèmes sont inadéquates, car l'analyse de sûreté doit prendre en compte non seulement l'état interne de chaque composant, mais aussi les différents flux de production circulants entre les composants. Par exemple, l'utilisation des Arbres de Fautes Statiques (AFS) n'est pas adaptée à ces systèmes, car ils ne prennent en compte que l'état interne des composants du DC.

Dans cette thèse, nous considérons une nouvelle méthodologie de modélisation appelée Arbres de Production (AP) qui permet de modéliser les dépendances entre les composants d'un système avec une attention particulière aux flux circulants entre ces composants. La technique de modélisation dAP permet de traiter un seul type de flux à la fois. Son application sur le sous-système électrique est donc appropriée, car il n'y a qu'un seul type de flux (le courant électrique). Toutefois, lorsqu'il existe des dépendances entre les sous-systèmes, comme dans le sous-système thermique et le sous-système réseau, il faut tenir compte de différents types de flux, ce qui rend l'application de la technique de modélisation dAP inadéquate. C'est pourquoi nous étendons cette technique pour traiter les dépendances entre les différents types de flux qui circulent dans le DC. Il est donc facile d'évaluer les différents indicateurs de sûreté du système global (DC), en tenant compte des interactions entre ses sous-systèmes. De plus, nous faisons quelques statistiques de performance. Nous validons les résultats de notre approche en les comparants à ceux obtenus par un outil de simulation que nous avons développé basé sur la théorie des réseaux de file dattente.

Jusqu'à présent, il nexiste pas doutils de résolution pour les modèles d'arbres de production. C'est pourquoi nous proposons une méthode de résolution basée sur la Distribution de Probabilité de Capacité (Probability Distribution of Capacity - PDC) des flux circulants dans le DC. Cette approche calcule à la fois la disponibilité et la fiabilité du système en utilisant un ensemble de formules prédéfinies. Elle est plus restreinte et plus précise que les méthodes de simulation. Nous implémentons également le modèle d'AP en utilisant le langage de modélisation AltaRica 3.0, et utilisons son simulateur stochastique pour estimer les indices de fiabilité du système. Ceci est très important pour comparer et valider les résultats obtenus avec notre

méthode de résolution.

En parallèle, nous développons un outil qui implémente l'algorithme de résolution des APs. Il s'agit d'un framework de modélisation EMF (Eclipse Modeling Framework) avec une interface graphique interactive qui permet de créer, éditer et analyser des modèles AP. L'outil permet également d'afficher les résultats et génère un code AltaRica, qui peut être analysé par la suite en utilisant le simulateur stochastique d'AltaRica 3.0.

Bibliography

[1] Computer aided fault tree analysis system. https://www.epri.com/#/pages/product/000000003002000020/?lang=en.

[2] Fault tree analysis software. https://www.isograph.com/software/reliability-workbench/fault-tree-analysis-software/.

[3] Graphiques interactifs pour la fiabilité. http://grif-workshop.fr/.

[4] Jedec global standards for the microelectronics industry. arrhenius equation for reliability. http://www.jedec.org/standards-documents/dictionary/terms/arrhenius-equation-reliability.

[5] Le logiciel blocksim de reliasoft. https://www.reliasoft.com/fr/produits/analyse-de-fiabilite/blocksim.

[6] Markov analysis software. https://www.itemsoft.com/markov.html.

[7] On semiconductor quality and reliability handbook. http://www.onsemi.com/publink/Collateral/HBD851-D.PDF.

[8] Safety tools development. https://www.satodev.com/en/.

[9] Vishay semiconductor reliability. http://www.vishay.com/docs80116/80116.PDF.

[10] *User's guide for the UNIRAM version 4.1 for Windowsavailability assessment methodology, Availability Systems* . 1996.

[11] Ashrae, thermal guidleines for data processing environment. *Ashrae 1st edition*, 2000.

[12] *Fault Tree Analysis*, chapter 11, pages 183–221. John Wiley and Sons, Ltd, 2005.

[13] *Petri Net Analysis (PNA)*, chapter 17, pages 307–316. John Wiley and Sons, Ltd, 2005.

[14] *Preliminary Hazard Analysis*, chapter 5, pages 73–93. John Wiley and Sons, Ltd, 2005.

[15] *Systems Engineering and the World of Modem Systems*, chapter 1, pages 3–30. John Wiley and Sons, Ltd, 2005.

[16] *Systems Engineering The Real Deal*, chapter 15, pages 349–393. John Wiley and Sons, Ltd, 2007.

[17] *Dysfunctional Behavior*, pages 73–74. DUV, Wiesbaden, 2008.

[18] *Open Queueing Networks*, chapter 6, pages 169–195. John Wiley and Sons, Ltd, 2008.

[19] *RISK AND SAFETY OF ENGINEERED SYSTEMS*, chapter 1, pages 1–14. John Wiley and Sons, Ltd, 2011.

[20] *Introduction to Risk Assessment*, chapter 1, pages 1–12. John Wiley and Sons, Ltd, 2012.

[21] *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Wiley, 2015.

[22] *Principles and Concepts of Cloud Computing*, chapter 1, pages 1–32. John Wiley and Sons, Ltd, 2016.

[23] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[24] Reem Alshahrani and Hassan Peyravi. Modeling and simulation of data center networks. *SIGSIM-PADS 2014 - Proceedings of the 2014 ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*, 05 2014.

[25] JD Andrews and TR Moss. Reliability and risk assessment. *John Wiley and Sons*, 1993.

[26] Chee-Wei Ang and Chen-Khong Tham. Analysis and optimization of service availability in a high availability cluster with load-dependent machine availability. *IEEE Transactions on Parallel and Distributed Systems*, 18:1307–1319, 2007.

[27] Marcel Antal, Tudor Cioara, Ionut Anghel, Radoslaw Gorzenski, Radoslaw Januszewski, Ariel Oleksiak, Wojciech Piatek, Claudia Pop, Ioan Salomie, and Wojciech Szeliga. Reuse of Data Center Waste Heat in Nearby Neighborhoods: A Neural Networks-Based Prediction Model. *Energies*, 12(5):814, 2019.

[28] ASHRAE. Ashrae handbook., 2018. `https://www.ashrae.org/technical-resources/ashrae-handbook`.

[29] Jayati Athavale, Minami Yoda, and Yogendra Joshi. Comparison of data driven modeling approaches for temperature prediction in data centers. *International Journal of Heat and Mass Transfer*, 135:1039 – 1052, 2019.

[30] Wei B, Lin C, and Kong X. Dependability modeling and analysis for the virtual data center of cloud computing. *the IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, 2011.

[31] Bruce Beihoff, Christopher Oster, Sanford Friedenthal, Christiaan Paredis, Duncan Kemp, Heinz Stoewer, David Nichols, and Jon Wade. *A World in Motion Systems Engineering Vision 2025*. 01 2014.

[32] AH Beitelmal and CD Patel. Thermo-fluids provisioning of a high performance high density. *Springer Science*, 2006.

[33] W. M. Bennaceur and L. Kloul. Electrical and thermal system impact on the availability of a data center's system. In *2018 3rd International Conference on System Reliability and Safety (ICSRS)*, pages 142–148, Nov 2018.

[34] W. M. Bennaceur and L. Kloul. Reliability and performance analysis of a data center's network architecture. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Nov 2018.

[35] Walid M. Bennaceur and Leila Kloul. Safety analysis of a Data Center system. In *Congrès Lambda Mu 21, " Maîtrise des risques et transformation numérique : opportunités et menaces "*, Reims, France, October 2018.

[36] Walid Mokhtar Bennaceur, Leïla Kloul, and Antoine Rauzy. Safety analysis of a data center's electrical system using production trees. In Marco Bozzano and Yiannis Papadopoulos, editors, *Model-Based Safety and Assessment*, pages 82–96, Cham, 2017. Springer International Publishing.

[37] Walid Mokhtar Bennaceur and Leïla Kloul. Formal models for safety and performance analysis of a data center system. *Reliability Engineering System Safety*, 193:106643, 2020.

[38] Steven M. Biemer and Andrew P. Sage. *Systems Engineering: Basic Concepts and Life Cycle*, chapter 5, pages 145–171. John Wiley and Sons, Ltd, 2010.

[39] M. Bouissou, H. Bouhadana, M. Bannelier, and N. Villatte. Knowledge modelling and reliability processing: Presentation of the figaro language and associated tools. *IFAC Proceedings Volumes*, 24(13):69 – 75, 1991. IFAC Symposium on Safety of Computer Control Systems 1991 (SAFECOMP'91), Trondheim, Norway, 30 October-1 November 1991.

[40] Roger C. Burk. *Systems Engineering in Professional Practice*, chapter 7, pages 197–226. John Wiley and Sons, Ltd, 2010.

[41] Sauer C and Daduna H. Availability formulas and performance measures for separable degradable networks. *Stochastics and Quality Control*, 18:165–194, 2003.

[42] G. Callou, J Ferreira, P Maciel, D Tutsch, and R Souza. An integrated modeling approach to evaluate and optimize data center sustainability, dependability and cost. *Energies 2014*, 7(1):238–277, January 2014.

[43] P Carer, J Bellvis, M Bouissou, Jean Domergue, and J Pestourie. A new method for reliability assessment of electrical power supplies with standby redundancies. *Proceedings of the 7th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, (January), 2002.

[44] Walter Cazzola. Domain-specific languages in few steps. In Thomas Gschwind, Flavio De Paoli, Volker Gruhn, and Matthias Book, editors, *Software Composition*, pages 162–177, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[45] Charley Chen, Guosai Wang, Jiao Sun, and Wei Xu. Detecting data center cooling problems using a data-driven approach. pages 1–8, 08 2018.

[46] Charley Chen, Guosai Wang, Jiao Sun, and Wei Xu. Detecting Data Center Cooling Problems Using a Data-driven Approach. pages 1–8, 2018.

[47] Jinkyun Cho, Joonyoung Yang, and Woopyoung Park. Evaluation of air distribution system's airflow performance for cooling energy savings in high-density data centers. *Energy and Buildings*, 68:270 – 279, 2014.

[48] Rodrigo De Souza Couto, Stefano Secci, Miguel Elias Mitre Campista, and Luís Henrique Maciel Kosmalski Costa. Reliability and survivability analysis of data center network topologies. *CoRR*, abs/1510.02735, 2015.

[49] C. Dabrowski and F. Hunt. Using markov chain and graph theory concepts to analyze behavior in complex distributed systems. *The 23rd European Modeling and Simulation Symposium*, 2011.

[50] J. B. Dugan. Automated analysis of phased-mission reliability. *IEEE Transactions on Reliability*, 40(1):45–52, April 1991.

[51] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, Sep. 1992.

[52] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, Sep. 1992.

[53] Facebook. Engineering facebook's notes., 2017. `http://www.facebook.com/notes.php?id=9445547199`.

[54] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. volume 41, pages 339–350, 10 2010.

[55] K Caldwel G Arrheni and S Wood. A tribute to the memory of svante arrhenius . 2008.

[56] P Maciel D Tutsch G Callou and J Araujo. Models for dependability and sustainability analysis of a data center cooling architectures. *Dependable systems and networks workshops*, pages 1–6, 2012.

[57] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, August 2011.

[58] Albert Greenberg, James Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. *Communications of the ACM*, 54:95–104, 10 2009.

[59] M. Gudemann and F. Ortmeier. A framework for qualitative and quantitative formal model-based safety analysis. In *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*, pages 132–141, Nov 2010.

[60] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, Songwu Lu, and Guohan Lv. Bcube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*. Association for Computing Machinery, Inc., August 2009.

[61] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: A scalable and fault-tolerant network structure for data centers. volume 38, pages 75–86, 10 2008.

[62] MD Hill. *The Datacenter as a Computer*. Morgan and Calypool, 2009.

[63] J. Hillston and N. Thomas. Product form solution for a class of pepa models. In *Proceedings. IEEE International Computer Performance and Dependability Symposium. IPDS'98 (Cat. No.98TB100248)*, pages 152–161, Sep. 1998.

[64] Nanyan Jiang and Manish Parashar. Enabling autonomic power-aware management of instrumented data centers. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[65] C. Kehren. Motifs formels d'architectures de systemes pour la surete de fonctionnement. *These de Doctorat, Ecole Nationale Superieure de l'Aeronotique et de l'Espace (SUPAERO)*, 2005.

[66] L Kloul and A Rauzy. Production trees: a new modeling methodology for production availability analyses. *Reliability Engineering and System Safety*, 167:561–571, 2017.

[67] S. Kounev, K. Sachs, J. Bacon, and A. Buchmann. A methodology for performance modeling of distributed event-based systems. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 13–22, May 2008.

[68] Samuel Kounev, Konstantin Bender, Fabian Brosig, Nikolaus Huber, and Russell Okamoto. Automated simulation-based capacity planning for enterprise data fabrics. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, pages 27–36, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[69] Nicolas Larrieu and Antoine Varet. *Developing Model-Based Design Methods in Software Engineering*, chapter 1, pages 1–22. John Wiley and Sons, Ltd, 2014.

[70] Hu-Chen Liu, Jian-Xin You, ZhiWu Li, and Guangdong Tian. Fuzzy petri nets for knowledge representation and reasoning. *Eng. Appl. Artif. Intell.*, 60(C):45–56, April 2017.

[71] D. Long and Z. Scott. *A Primer for Model-Based Systems Engineering*. Vitech Company, 2011.

[72] Al-Fares M, Loukissas A, and Vahdat A. A scalable, commodity data center network architecture. *Computer Communication ACMSIGCOM*, 38:63–74, 2008.

[73] Patterson M. The effect of data center temperature on energy efficiency. *In Proceedings of the ITHERM08, 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2008.

[74] Daniel S. Marcon, Rodrigo R. Oliveira, Luciano P. Gaspary, and Marinho P. Barcellos. *Datacenter Networks and Relevant Standards*, chapter 4, pages 73–104. John Wiley and Sons, Ltd, 2015.

[75] M Marwah, P Maciel, A Shah, R Sharma, and T Christian. Quantifying the sustainability impact of data center availability. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):64–68, March 2010.

[76] Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage, and Andrea Bobbio. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. *Reliability, IEEE Transactions on*, 59:250 – 261, 04 2010.

[77] Microsoft. Creating a greaner data center., 2013. `http://www.microsoft.com/presspass/features/2009/apr09/04-02Greendatacenters.mspx`.

[78] Montgomery and Warren A. 545 Technology Square Permission to copy without fee all or part of this material is granted provided that the copies are

not made or distributed for direct commercial advantage , the ACM copyright notice and the title of the publication and its date appe. *Technology*, pages 143–149, 1979.

[79] J Noble, A Taivalsaari, and I Moore. Prototype-based programming: Concepts, languages and applications. *Springer-Verlag, Berlin and Heidelberg*, 1999.

[80] P. D. T. O'Connor. Reliability and risk assessment, j.d. andrews and t.r. moss, longman scientific and technical, 1993 (john wiley in u.s.a.), number of pages: 368. *Quality and Reliability Engineering International*, 11(1):74–74, 1995.

[81] Yiannis Papadopoulos, Martin Walker, David Parker, Erich Rüde, Rainer Hamann, Andreas Uhlig, Uwe Grätz, and Rune Lien. Engineering failure analysis and design optimisation with hip-hops. *Engineering Failure Analysis*, 18(2):590 – 608, 2011. The Fourth International Conference on Engineering Failure Analysis Part 1.

[82] Gregory S. Parnell. *Introduction to Systems Engineering*, chapter 6, pages 183–195. John Wiley and Sons, Ltd, 2010.

[83] Chandrakant Patel, Cullen Bash, Ratnesh Sharma, Monem Beitelmal, and Rich Friedrich. Smart cooling of data centers. 01 2003.

[84] Chandrakant Patel, Cullen Bash, Ratnesh Sharma, Monem Beitelmal, and Rich Friedrich. Smart cooling of data centers. 01 2003.

[85] Massoud Pedram and Shahin Nazarian. Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. *Proceedings of the IEEE*, 94:1487 – 1501, 09 2006.

[86] K Camboin J Ferreira P Maciel. R Souza, G Callou. The effects of temperature variation on data center it systems. *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference.*, 2013.

[87] Tomasz Rak. Performance modeling using queueing petri nets. In Piotr Gaj, Andrzej Kwiecień, and Michał Sawicki, editors, *Computer Networks*, pages 321–335, Cham, 2017. Springer International Publishing.

[88] K. Durga Rao, V. Gopika, V.V.S. Sanyasi Rao, H.S. Kushwaha, A.K. Verma, and A. Srividya. Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment. *Reliability Engineering & System Safety*, 94(4):872 – 883, 2009.

[89] Neil Rasmussen.l. Calculating total tooling requirements for data centers. *Schneider Electric White Pape 25r*, 2015.

[90] A Rauzy. Guarded transition systems: a new states/events formalism for reliability studies. *Journal of Risk and Reliability*, 222:495–505, 2008.

[91] Antoine B. Rauzy and Cecilia Haskins. Foundations for model-based systems engineering and model-based safety assessment. *Systems Engineering*, 22(2):146–155, 2019.

[92] R Robidoux. Automated modeling of dynamic reliability block diagrams using colored petri nets. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(2):337–351, March 2010.

[93] R Robidoux, H Xu, M Zhou, S Member, L Xing, and Member. Automated modeling of dynamic reliability block diagrams using colored petri net. *IEEE Trans Syst Man Cybern Part A Syst*, 40:337–351, June 2010.

[94] P. Rygielski and S. Kounev. Data center network throughput analysis using queueing petri nets. In *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 100–105, June 2014.

[95] Sharma, Abhilasha, and RG Sangeetha. Reliability Analysis of Data Center Network. 99:99.

[96] Ji-Yong Shin, Bernard Wong, and Emin Gün Sirer. Small-world datacenters. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, pages 2:1–2:13, New York, NY, USA, 2011. ACM.

[97] S Silvaa, B Silvaa, P Romero, M Maciela, and A Zimmermannb. Dependability evaluation of data center power infrastructures considering substation switching operations. *Probabilistic Safety Assessment and Management conference*, June 2014.

[98] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[99] Wang T. Towards cost-effective and low latency data center network architecture. *Computer Communication*, 82:1–12, 2016.

[100] W Pitt Turner, JH Seader, and KG Brill. Tier classifications define site infrastructure performance. *Uptime Institute White Paper*, 2013.

[101] Sharma V and Virtamo JT. A finite buffer queue with priorities. *Performance Evaluation*, 47:1–21, 2002.

[102] Prabhakar V. Varde and Michael G. Pecht. *System Reliability Modeling*, pages 71–113. Springer Singapore, Singapore, 2018.

[103] G. Wang, L. Zhang, and W. Xu. What can we learn from four years of data center hardware failures? In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 25–36, June 2017.

[104] Guohui Wang, David G. Andersen, Michael Kaminsky, Konstantina Papagiannaki, T.S. Eugene Ng, Michael Kozuch, and Michael Ryan. c-through: part-time optics in data centers. *SIGCOMM Comput. Commun. Rev.*, 41(4):–, August 2010.

[105] J.X. Wang and M.L. Roush. *What Every Engineer Should Know About Risk Engineering and Management*. What Every Engineer Should Know. Taylor & Francis, 2000.

[106] M Wiboonrat. An empirical study on data center system failure diagnosis. *Internet Monitoring and Protection*, July 2008.

[107] Wang X, Yao Y, Wang X, Lu K, and Cao Q. Carpo: Correlation-aware power optimization in data center networks. *In Proceedings of the 2012 IEEE INFOCOM*, 2012.

[108] J. Xiang, F. Machida, K. Tadano, K. Yanoo, W. Sun, and Y. Maeno. A static analysis of dynamic fault trees with priority-and gates. In *2013 Sixth Latin-American Symposium on Dependable Computing*, pages 58–67, April 2013.

[109] Junjie Xie, Yuhui Deng, and Ke Zhou. Totoro: A scalable and fault-tolerant data center network by using backup port. In Ching-Hsien Hsu, Xiaoming Li, Xuanhua Shi, and Ran Zheng, editors, *Network and Parallel Computing*, pages 94–105, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[110] Jian Xu, Liang Tang, and Tao Li. System situation ticket identification using svms ensemble. *Expert Syst. Appl.*, 60(C):130–140, October 2016.

[111] DY Yang, Wang KH, and Kuo YT. Economic application in a finite capacity multi-channel queue with second optional channel. *Applied Mathematics and Computation*, 217:7412–7419, 2011.

[112] Juntao Zhang, Hyungju Kim, Yiliu Liu, and Mary Ann Lundteigen. Combining system-theoretic process analysis and availability assessment: A subsea case study. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 233(4):520–536, 2019.

[113] Chenglong Zhou, Chunxin Yang, Chao Wang, and Xingjuan Zhang. Numerical simulation on a thermal management system for a small data center. *International Journal of Heat and Mass Transfer*, 124:677 – 692, 2018.