FEDERAL UNIVERSITY OF PERNAMBUCO

CENTER OF INFORMATICS

PROGRAM OF POST-GRADUATION IN COMPUTER SCIENCE

"An automated approach for systems performance and dependability improvement through sensitivity analysis of Markov chains"

By

Rubens de Souza Matos Júnior

Master's Dissertation

Recife, February 2011

FEDERAL UNIVERSITY OF PERNAMBUCO

CENTER OF INFORMATICS

PROGRAM OF POST-GRADUATION IN COMPUTER SCIENCE

Master's Dissertation

"An automated approach for systems performance and dependability improvement through sensitivity analysis of Markov chains"

Dissertation presented to the Program of Post-Graduation in Computer Science, Center of Informatics, Federal University of Pernam-. buco, as a partial requirement for the achievement of Master's degree in Computer Science.

Student: Rubens de Souza Matos Júnior

Advisor: Prof. Dr. Paulo Romero Martins Maciel

Recife, February 2011

Dedicatory

To my family: father, mother, sisters and fiancee. These two years of hard work would not be successful without their support and comprehension.

Acknowledgement

Thanks for all colleagues in UFPE, my advisor prof. Paulo Maciel, and for prof. Trivedi and his students at Duke University, for the valuable help and guidance.

Abstract

Computer systems are in constant evolution to satisfy increases in the demand, or new user exigences. The administration of these systems requires decisions that are able to provide the highest level of performance and dependability metrics with minimal changes to the existing configuration. It is common to carry out performance, reliability, availability and performability analysis of systems via analytical models, and Markov chains represent one of the most used modeling formalisms, allowing to estimate some measures of interest, given a set of input parameters. Although, sensitivity analysis, when done, is executed simply by varying the set of parameters over their ranges and repeatedly solving the chosen model. Differential sensitivity analysis allows the modeler to find bottlenecks in a more systematic and an efficient manner. This work presents an automated approach for sensitivity analysis that aims to guide the improvement of computer systems. The proposed approach is able to speed up the process of decision making, regarding software and hardware tuning, besides acquisition and replacement of components. Such methodology uses Markov chains as formal modeling technique, and differential sensitivity analysis of these models, fulfilling some gaps found in the sensitivity analysis literature. Lastly, the sensitivity analysis of some chosen distributed systems, that is conducted in this work, shall highlight bottlenecks in these systems and provide examples on the accuracy of the proposed methodology, as well as illustrate its applicability.

Keywords: Sensitivity analysis, Markov chains, Systems optimization, Dependability and Performance Modeling

Contents

List of Figures

List of Tables

Listings

Glossary

COA Capacity Oriented Availability. 75–77

CTMC Continuous-Time Markov Chain. 8, 32, 35, 46, 53

DEDS Discrete Event Dynamic System. 25

DSPN Deterministic and Stochastic Petri Net. 11, 13, 33

DTMC Discrete-Time Markov Chain. 8, 35

GSPN Generalized Stochastic Petri Net. 5, 13, 32

MRM Markov Reward Model. 12, 28, 35, 46

MTTA Mean Time to Activate. 64

PA Perturbation Analysis. 25

pmf Probability mass function. 9

S.A. Sensitivity Analysis. 16, 17, 23, 32, 46

SAN Storage Area Network. 72

SPN Stochastic Petri Net. 11, 13

SRN Stochastic Reward Net. 5, 13

Chapter 1

Introduction

The process of systems modeling commonly deals with many different input parameters. Modern computer systems are complex, due to the amount of hardware and software components, and the various interactions between them. Distributed and parallel systems are typical examples of such complexity. In order to describe such systems, and manage them properly, the models that predict their behavior often need to include a large number of variables or parameters. Given that each parameter has a distinct impact on performance, availability and reliability measures, the required knowledge of the influence order of model parameters is critical for deciding on the appropriate attention that should be paid to each individually.

Parametric sensitivity analysis is an efficient method for determining the order of influence of parameters on model results. According to [Hamby 1994], models are prone to two kinds of influence from their parameters. The first kind is related to the variability, or uncertainty, of an input parameter, that may cause a high variability in the model's output. The second kind is the actual correlation between an input parameter and model results, so that small changes in the input value may result in significant changes in the output. There are different types of analysis to deal with each kind of parametric sensitivity.

The relative importance of each parameter, as stated in the second kind of sensitivity, is mainly used to elaborate a list of the input parameters sorted by the amount of contribution each one has on the model output. This is a usual case when parameters have little uncertainty to propagate to the results, or when the variability of results is assumed to be a minor concern.

When dealing with analytic models, parametric sensitivity analysis is a particularly impor-

tant technique used to find performance and reliability bottlenecks in the system, thus guiding the optimization process [Blake et al. 1988, Abdallah and Hamza 2002]. It can also guide the exclusion of parameters without significant effect to the results. Large models, with dozens of rates, may be drastically reduced by using this approach.

There are many ways of conducting parametric sensitivity analysis. Some of them can be properly used in analytical models, whereas other approaches are better suited to experimental measurement based analysis. The simplest method, in a conceptual view, is to repeatedly vary one parameter at a time, while keeping the others fixed. When applying this method, the sensitivity ranking is obtained by noting the corresponding changes in the model output. This method is commonly used in conjunction with plots of input versus output, and it is likely the predominant in performance and dependability analyses found in the literature.

Although, varying one parameter at a time is less useful in some opportunities. When the amount of parameters is large, the analysis of scatter plots becomes harder, mainly due to the proximity of curves. The difference in magnitude orders is another possible complicating factor, since all parameters cannot be visualized in the same plot, forbidding accurate interpretations about the differences among parameters influence. Due to such cases, methods that are based on numerical sensitivity indexes should have preference in spite of "one parameter at a time" approach.

Differential sensitivity analysis, also referred to as the direct method, is the backbone of many other sensitivity analysis techniques [Hamby 1994]. It may be performed in an efficient computational manner on analytic models commonly used in performance and dependability analyses. This method provides a single sensitivity coefficient, that denotes the amount of change in the output that is produced by an incremental change of a given input. It is performed by computing the partial derivatives of the measure of interest with respect to each input parameter. In [Frank 1978], these partial derivatives are referred to as sensitivity functions, since there may be other parameters involved, that are not part of the differentiation (e.g., a time parameter).

The main performance and dependability modeling techniques for computer systems can undergo a differential sensitivity analysis. Sensitivity functions may be computed for Markov chains, queueing networks, stochastic automata networks, and stochastic Petri nets, for instance.

The derivative of equations that solve these models, as well as the adaptations in corresponding algorithms has been published in the literature. Although, the use of differential sensitivity analysis is neither widespread among the research community of performance and dependability modeling nor in the computer systems industry.

A likely reason for this lack of usage of formal sensitivity analysis techniques is the scarcity of such features in modeling software tools. The process of differential sensitivity analysis, and even other related techniques, may be hard without the proper computational support. Despite the existence of a number of software packages that allow the fast creation and solution of analytical models, few tools include features to perform sensitivity analysis in these models. Specifically, no Markov chain tool is known to support differential sensitivity analysis.

There are some systems to which closed-form equations can be written to compute their performance or dependability metrics, demanding minimal computer support. But even for such systems, the analyses found in literature rarely include sensitivity indices due to the absence of a well-defined methodology for sensitivity analysis in stochastic modeling.

This work proposes an automated sensitivity analysis methodology, aiming to provide a efficient, objective and unambiguous way to perform system optimization and model enhancement. Such methodology uses Markov chains as formal modeling technique, and fulfills some gaps found in the sensitivity analysis literature.

1.1 Motivation

The majority of performance and dependability analyses do not include the computation of quantitative sensitivity indices. Sensitivity analysis, when done, is executed simply by varying the set of parameters over their ranges and repeatedly solving the chosen model. Formal or differential sensitivity analysis is not commonly used, what makes the determination of bottlenecks more difficult. Subsequently, system optimization is usually performed in a inefficient and manual process.

This work was motivated by the absence of an automated methodology for sensitivity analysis focused on performance and dependability improvement of computer systems. Some existing techniques may be gathered and organized in order to provide a sequence of well defined

and automated steps for systems optimization through analytic modeling. The well founded background about computation of sensitivity functions in Markov chains is another motivating factor. It was seen an opportunity to contribute in speeding up the decision about improvements in computer infrastructures, by easing the process of sensitivity analysis. Such process would be more efficient by means of specific features in a complete modeling tool, able to provide proper ways to model specification and a full range of solving methods.

Commonly, the computation of expenses in system improvement and estimates of budget gains is left to the last analysis phase, using this information in a fragmented manner. This kind of evaluation process is not proper for scenarios where optimization actions have very different costs, or when there is doubts about the trade-off between improvement of a given metric and the efforts spent on such optimization. There is a need to integrate cost information to the sensitivity analysis, in order to find the most cost effective point of optimization, considering all model parameters and a reasonable trade-off between measure enhancement and investments in system modifications.

Moreover, there is a gap in the sensitivity analysis approaches, related to the scaling (normalization) of the sensitivity index. No guidance is found about which cases require the use of scaled sensitivities, nor what are the proper normalization methods for the performance/dependability metrics that are usual in computer systems analysis. The present work aims to solve this problem, which will help in more accurate sensitivity analyses.

1.2 Objectives

The main objective of this work is to propose an automated methodology for systems optimization using sensitivity analysis of Markov chains. This approach is focused on performance and dependability aspects of computer systems. Additional, and more specific, objectives are listed below:

- Present clear guidelines to the decision about the use of scaled or unscaled sensitivity functions
- Allow the correct interpretation of each parameter's importance even when there are different orders of magnitude involved

- Propose an approach to perform cost-based constrained optimization using sensitivity analysis
- Develop a software to support the activity of sensitivity analysis of performance and dependability models
- Propose optimizations in selected distributed systems by using the proposed approach

1.3 Justification

Definition of importance order in analytical models is often not trivial, due to the existence of many parameters, or differences in orders of magnitude between one parameter and another. Especially, in performance and dependability models, some rates may be defined in seconds, while other ones are measured in hours, for example, what makes difficult the correct interpretation of graphical plots and similar analysis means.

In the optimization of complex systems, it is important to follow a methodology that accurately defines which parameters are the bottlenecks, and which do not deserve attention during improvements of a specific performance or dependability aspect. The usual approach of repeatedly varying one parameter at a time and compare the respective model outputs is not feasible or efficient in many situations. A methodology based on differential sensitivity analysis fits better the requirements of analytic models commonly used in the evaluation of systems behavior. This methodology is based on quantitative sensitivity indices, and can be adapted to include constrained optimization with little effort and avoiding subjective interpretation of the results.

Although tools already exist for the differential sensitivity analysis of GSPNs (Generalized Stochastic Petri Nets) and SRNs (Stochastic Reward Nets), there is a need for implementing such features for Markov chains, since there are many systems described in a simple but effective manner by this model type. Besides, the existing tools for other models present some restrictions, such as amount of parameters, and symbolic definition of rates as functions of parameters. The aimed implementation shall allow the analysis of a wide range of models in a flexible way, without those restrictions.

1.4 Aimed contributions

The proposed approach will benefit researchers and systems administrators, by showing a precise methodology to be followed in the enhancement, acquisition and replacement of components for computational infrastructures. Performance tuning through adjusts in software and hardware parameters should also be undergone in a agile and efficient manner, using the methods and tools presented here.

This work also intends to fulfill a gap in the sensitivity analysis literature, providing guidelines about the use of scaled or unscaled sensitivity functions in scenarios of performance and dependability modeling. These guidelines will help in the accurate identification of the importance of each parameter to system behavior.

The possibility to include costs definition and constrained optimization during the sensitivity analysis is also expected. The proposed methodology shall allow optimizations based not only on direct financial costs, but energy consumption, working time, and similar expenses. General constraint-oriented improvements of computer systems may find in this work a useful guide.

The sensitivity analysis of some chosen distributed systems, that is carried out here, shall highlight bottlenecks in these systems and provide examples on the accuracy of the proposed methodology, as well as illustrate its possible uses.

1.5 Structure of the dissertation

The remaining of this dissertation is structured as follows. Chapter 2 presents the main theoretical aspects that are the foundation for this work, regarding Markov chains and sensitivity analysis. Chapter 3 describes briefly some related works that have been found in the literature about sensitivity analysis of analytic models. Chapter 4 presents the main contributions of the present master's dissertation, explaining the proposed methodology for systems optimization. Details about the methodology automation are found in Chapter 5, that described the computer support developed for the sensitivity analysis of Markov chains. Chapter 6 is composed by the description of three case studies regarding sensitivity analysis of different systems, providing examples on the accuracy of the proposed methodology, and illustrating its possible uses in systems improvement. Chapter 7 states the final considerations of this work, pointing out the

reached contributions as well as some future works.

Chapter 2

Foundations

This chapter presents the theoretical basics over which the present Master's dissertation was developed. First, Markov chains are introduced, from its mathematical concepts to the application in performance and dependability modeling. Afterward, sensitivity analysis definitions and methods are disclosed, completing the necessary knowledge to understand the methodology proposed in Chapter 4.

2.1 Markov chains

Markov models are the fundamental building blocks upon which most of the quantitative analytical performance techniques are built [Kolmogorov 1931, Trivedi 2001]. Such models may be used to represent the interactions between various system components, for both descriptive and predictive purposes [Menascé et al. 2004]. Markov models have been in use intensively in performance and dependability modeling since around the fifties [Maciel et al. 2011]. Besides computer science, the range of applications for Markov models is very extensive. Economics, meteorology, physics, chemistry and telecommunications are some examples of fields which found in this kind of stochastic [1] modeling a good approach to address various problems. In this section, the formalism of Markov models, and more specifically Markov chains, is presented with focus on the performance and dependability analysis that can be executed using such analytical approach.

A Markov model can be described as a state-space diagram associated to a Markov pro-

[1] Stochastic refers to something which involves or contains a random variable or variables

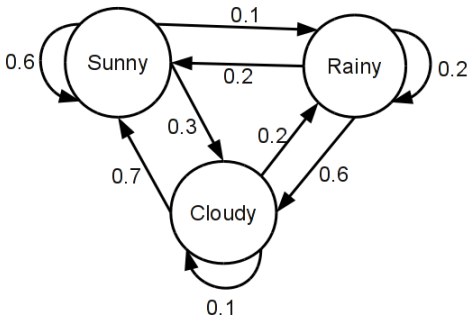

Figure 2.1: Simple Markov model

cess, which constitutes a subclass of stochastic processes. A definition of stochastic process is presented:

> A stochastic process is defined as a family of random variables $\{X_t : t \in T\}$ where each random variable X_t is indexed by parameter $t \in T$, which is usually called the time parameter if $T \subset R_+ = [0, \infty)$, i.e., T is in the set of non-negative real numbers. The set of all possible values of X_t (for each $t \in T$) is known as the state space S of the stochastic process [Bolch et al. 2001].

Let $Pr\{k\}$ be the probability of a given event k occurs. A Markov process is a stochastic process in which $Pr\{X_{t_{n+1}} \leq s_{i+1}\}$ depends only on the last previous value X_{t_n}, for all $t_{n+1} > t_n > t_{n-1} > ... > t_0 = 0$, and all $s_i \in S$. This is the so-called Markov property [Haverkort 2002], which, in plain words, means that the future evolution of the Markov process is totally described by the current state, and is independent of past states [Haverkort 2002]. Figure 2.1 shows a simple Markov model, that represents the daily behavior of weather in a city. There are only three states, so the state space is defined as $S = \{Sunny, Cloudy, Rainy\}$. According to the Markov property, given that in a moment t_n the weather is found in Sunny state, the probability of transitioning to Rainy state in moment t_{n+1} does not depend on which states have been visited before. The same is true for all other states and transitions. Also, the time spent in current state does not influence the transition probability.

In this work, there is only interest on discrete (countable) state space Markov models, also known as Markov chains, which are distinguished in two classes: Discrete-Time Markov Chains (DTMC) and Continuous-Time Markov Chains (CTMC) [Kleinrock 1975]. In DTMCs, the transitions between states can only take place at known intervals, that is, step-by-step. Systems

where transitions only occur in a daily basis, as the example of Figure 2.1, or following a strict discrete clock are well represented by DTMCs. If state transitions may occur at arbitrary (continuous) instants of time, the Markov chain is a CTMC. The Markov property implies that the time of transitions is driven by a memoryless distribution [Bolch et al. 2001]. In the case of DTMC, the geometric distribution is the only discrete time distribution that presents the memoryless property. In the case of CTMC, the exponential distribution is used.

As shown in [Bolch et al. 2001], when dealing with DTMC, the process's one-step transition from state i to state j at time n, $p_{ij}(n)$, has a particular conditional probability mass function (pmf) that can be written with the following shorthand notation:

$$p_{ij}(n) = P(X_{n+1} = s_{n+1} = j | X_n = i). \tag{2.1}$$

Given that for each state $i \in S$, $\sum_j p_{ij} = 1$ and $0 \leq p_{ij} \leq 1$, a stochastic transition matrix P is used to summarize all transition probabilities of a DTMC. For the DTMC of Figure 2.1, considering a equivalent state-space $S = \{0, 1, 2\}$, the matrix P is:

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} 0.6 & 0.3 & 0.1 \\ 0.7 & 0.1 & 0.2 \\ 0.2 & 0.6 & 0.2 \end{pmatrix}$$

The transition matrix has a key role in the computation of the state probability vector $\underline{\pi}$. This vector yields the information about the probability of the system being in a given state, either in a specific number n of steps (see Equation 2.2) or in steady-state, when $n \to \infty$ (see Equation 2.3). It is important to highlight the need for the previous knowledge of initial state probabilities, $\underline{\pi}(0)$, in case of time-dependent probabilities, whereas steady-state probabilities do not depend on the system's initial condition. From the state probability vector, nearly all other main metrics can be derived, depending on the system that is represented.

$$\underline{\pi}(n) = \underline{\pi}(0) P^n \tag{2.2}$$

$$\underline{\pi} = \underline{\pi} P \tag{2.3}$$

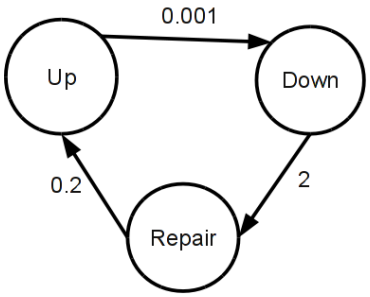

Figure 2.2: Simple CTMC

When dealing with CTMCs, such as the availability model of Figure 2.2, the transition matrix is referenced as infinitesimal generator matrix, since in this case transitions occur with a rate, instead of a probability, due to the continuous nature of this kind of model. Considering the CTMC availability model, the rates are measured in failures per second, repairs per second, and detections per second. The generator matrix Q is composed by components q_{ii} and q_{ij}, where $i \neq j$ and $\sum q_{ij} = -q_{ii}$. Using the availability model that was just mentioned, considering a state-space $S = \{Up, Down, Repair\} = \{0, 1, 2\}$ the Q matrix is:

$$Q = \begin{pmatrix} q_{00} & q_{01} & q_{02} \\ q_{10} & q_{11} & q_{12} \\ q_{20} & q_{21} & q_{22} \end{pmatrix} = \begin{pmatrix} -0.001 & 0.001 & 0 \\ 0 & -2 & 2 \\ 0.2 & 0 & -0.2 \end{pmatrix}$$

Equation 2.4 and the system of Equations 2.5 describe, respectively, the computation of transient (time-dependent) and steady-state (stationary) probability vector.

$$\underline{\pi}'(t) = \underline{\pi}(t)Q, \qquad \text{given } \underline{\pi}(0). \tag{2.4}$$

$$\underline{\pi}Q = 0, \sum_{i \in S} \pi_i = 1 \tag{2.5}$$

Detailed explanations about how to obtain these equations may be found in [Haverkort and Meeuwissen 1995, Bolch et al. 2001].

Other important measure for which there exist well-known computation methods is the cumulative transient probability, which may be used to measure cumulative accomplishments during a given interval of time $[0, t)$, and is described as:

$$\underline{L}(t) = \int_0^t \underline{\pi}(u)du. \tag{2.6}$$

Vector $\underline{L}(t)$ encompasses the total expected time spent in each state of the CTMC during that interval of time. Equation 2.7, resulting from integration on both sides of Equation 2.4 is commonly used in the solution of cumulative measures.

$$\frac{d\underline{L}(t)}{dt} = \underline{L}(t)Q + \underline{\pi}(0), \underline{L}(0) = 0 \tag{2.7}$$

In the model of Figure 2.2, $\underline{L}(8640)$ may be used to compute the time in which the system was found in Down, Up, or Repair state, after 1 year (8640 hours) of operation. The vector components $\underline{L}_{Down}(8640)$, $\underline{L}_{Up}(8640)$, and $\underline{L}_{Repair}(8640)$ provides this information for each corresponding state of the Markov chain. Other useful metric for this model may be the throughput of repairs, i.e., the mean amount of repair actions executed per hour, or per month. This information is useful in dimensioning the repair team and related resources. Therefore, throughput of repairs may be defined as $R = \pi_{repair}\mu_r$, where π_{repair} is the steady-state probability of finding the system in state Repair, and μ_r is the repair rate, the inverse of mean time to change from Repair to Up state.

Markov chains as a whole (DTMCs and CTMCs) may also be classified according to how their states are reached over time. In an **irreducible** chain, every state can be reached from every other state, so all states are recurrent[2] in this kind of model. If, once the system leaves any state, that state can not be visited anymore, the Markov chain is **acyclic**. If the chain is neither acyclic nor irreducible, it is classified as **phase-type** [Sahner et al. 1996]. It will eventually reach an absorbing[3] state, but while this does not happen, the chain passes by one or more transient[4] states. Figure 2.3 show examples of the three types of chains. For further details on this classification, see [Sahner et al. 1996].

DTMCs and CTMCs enable the analysis of many kinds of systems, with proper accuracy and modeling power. Other modeling formalisms, such as queueing networks [Kleinrock 1975, Chandy and Martin 1983], stochastic Petri nets (SPNs) [Molloy 1982,

[2]A state is called recurrent if the system returns to this state infinitely often [Sahner et al. 1996].

[3]A state is absorbing if once it is reached, the chain stays there forever [Sahner et al. 1996].

[4]A state is transient if, during an infinitely long time period, the system visits this state only finitely often [Sahner et al. 1996].

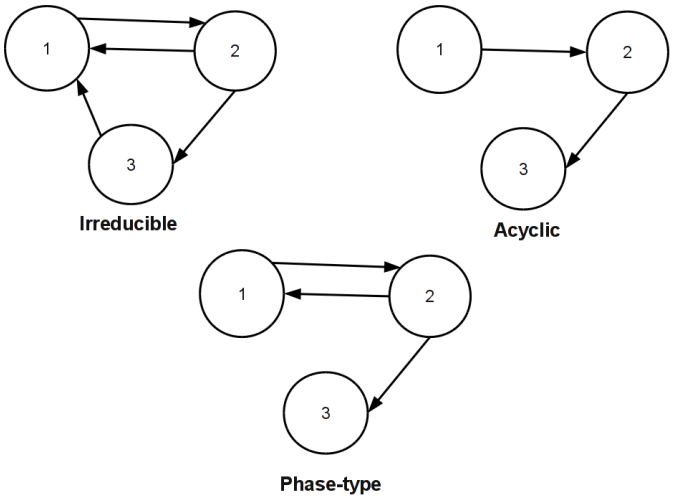

Figure 2.3: Types of Markov chains

Ajmone Marsan et al. 1984, Marsan et al. 1987], and deterministic stochastic Petri nets (DSPNs) [Marsan and Chiola 1987, German and Mitzlaff 1995], may be considered as high-level specification techniques for generation of Markov chains [Bolch et al. 2001]. Due to broad usage in computer systems performance and dependability modeling, CTMCs are the main focus of this work, together with Markov Reward Models (MRMs) [Smith et al. 1987], which are an extension to the fundamental models just explained in this section.

2.1.1 Markov Reward Models

Markov reward models (MRM) are commonly used to obtain combined measures, providing for instance performance and reliability integrated analysis. In order to build a MRM, a constant reward rate r_i is attached to each state i of a CTMC. It is also possible to associate reward rates with the transitions of the CTMC [Trivedi et al. 1994], and in such case they are known as impulse rewards. In general, the reward associated with a state denotes the performance[5] level given by the system while it is in that state [Sahner et al. 1996, Abdallah and Hamza 2002]. Using again the model depicted in Figure 2.2, if the reward rates are defined as the vector $\underline{r} = (r_1, r_2, r_3) = (5, 0, -2)$, representing the revenue obtained (or lost) in each state, the expected instantaneous reward rate of system at time t is:

$$E[X(t)] = \sum_{i \in S} r_i \pi_i(t) = 5\pi_{up}(t) + 0\pi_{down}(t) + -2\pi_{repair}(t). \tag{2.8}$$

[5]The word "performance" here is used in a broad sense, meaning any measurable aspect of system's behavior

$E[X(t)]$ may be considered as the expected revenue rate obtained by the system operation at time t. Other types of rewards may be used in Markov reward models, such as computational power, system availability status and job completions per time unit. Regardless of which reward rate, r_i, is adopted, the expected reward rate in steady-state, and the accumulated expected reward in the interval $[0, t)$ may be computed using Equations 2.9 and 2.10, respectively.

$$E[X] = \sum_{i \in S} r_i \pi_i \tag{2.9}$$

$$E[Y(t)] = \sum_{i \in S} r_i L_i(t) \tag{2.10}$$

For the previous example model, $E[X]$ and $E[Y(t)]$ provide particularly useful information, since they show, respectively, the mean revenue rate obtained when the system reaches a stationary state, and the total revenue obtained in a given period of time. Due to the possibility of computing such kinds of measures, Markov reward models have been widely adopted in performability studies, which combine performance and dependability aspects in a unified framework [Trivedi et al. 1992].

As Markov chains may be generated from SPN models, Markov reward models may also be generated and analyzed through an extension of SPNs, such as Generalized and Stochastic Petri nets - GSPNs [Ajmone Marsan et al. 1984], Deterministic and Stochastic Petri nets - DSPNs [Marsan and Chiola 1987], and Stochastic Reward Nets - SRNs [Ciardo et al. 1993]. In a SRN, reward rates are associated to the markings, that constitute each state of the underlying Markov model. In its turn, reward impulses are associated to the transitions between markings. For more details about the GSPN, DSPN, and SRN formalisms, see [Ajmone Marsan et al. 1984], [Marsan and Chiola 1987], and [Ciardo et al. 1993], respectively.

2.1.2 Performance and Dependability Modeling using Markov Chains

Performance and dependability modeling of computer systems enable one to represent the behavior of a system and compute measures which describe, in a quantitative way, how the service is provided and how much confidence can be put on the system operation. The measures of interest and the purposes of the performance evaluation may influence the choice of modeling

technique to be employed. The two major types of scientific models (simulation and analytic models) encompass tens of techniques which have advantages and drawbacks.

According to [Barcellos et al. 2006], simulation allows a variable degree of abstraction, from simple to detailed, depending on the model built and of the corresponding source code. Although, models that are based in mathematical formalisms (analytic models) such as Markov chains, are less expensive to construct and, in general, tend to be computationally more efficient to run than simulation models [Menascé et al. 2004]. A drawback of analytic models is the difficulty in representing certain complex systems, what it is determined by the modeling power of each specific formalism, and the characteristics of interactions between system components.

In performance evaluation studies, some metrics which usually deserve interest are: response time, job completion rate (throughput) and level of resource utilization, because these metrics are directly related to the user perception of system performance and they may also highlight the need for changes. Among other models, stochastic Petri nets, queueing networks and Markov chains have been used, in both academy and industry, in order to compute metrics such as those just mentioned. These models provide good abstraction of real world problems, and have well established solving methods and tools, that ease the analysis of a huge variety of systems.

Since stochastic Petri nets (SPN) and queueing networks (QN) may be converted to Markov chains, the latter one has a distinguished importance for the field of stochastic modeling. By employing the proper abstraction level, Markov chains may provide a large set of measures for the system under analysis. For example, the CTMC of Figure 2.4 is a classical representation of a queueing system, in which there is a single server, with capacity for 3 jobs, at maximum. That model type is well-known as a birth-death process, since transition can only occur to neighbor states. The arrival of a new job (or client) is a birth, and has a exponentially distributed rate λ. The departure of a job (due to service completion) is a death, which happens with rate μ, that is also exponential. Birth-death models just like that of Figure 2.4 are equivalent to M/M/1 queueing systems [Bolch et al. 2001], as that one depicted in Figure 2.5, and may be used in their solution. It is important to state that, for the queueing system being stable, $\lambda < \mu$.

In Figure 2.4, state 0 represents the idle condition, in which there is no job in the system. After a job arrival, the system is found in state 1, when there is one job being processed. If

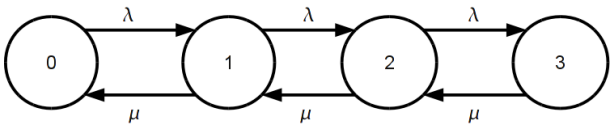

Figure 2.4: Birth-death CTMC

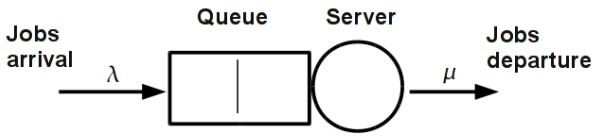

Figure 2.5: Single server queueing system

the service is completed before a new job arrives, the system goes to state 0 again, else it goes to state 2, that indicates one job is being processed and other job is waiting in the queue. In a similar way, state 3 denotes the presence of 2 jobs in the queue. No job arrival is allowed when the queue is full (i.e., in state 3). This Markov chain may be used to compute the utilization level of the server, by means of Equation 2.11, which computes the probability of having at least one job in the system, when it enters in a stationary state.

$$U = \pi_1 + \pi_2 + \pi_3 = 1 - \pi_0 \tag{2.11}$$

Other interesting measure that could be computed for the CTMC of Figure 2.4 is the mean number of jobs (J) in the system, that includes that ones in waiting queue and the job being processed. Equation 2.12 describes the computation of this metric for the steady-state case.

$$J = 1 \cdot \pi_1 + 2 \cdot \pi_2 + 3 \cdot \pi_3 \tag{2.12}$$

In fact, U and J could also be computed as the steady-state expected reward of a corresponding Markov reward model. In order to do this, a reward vector $r = (0, 1, 1, 1)$ should be assigned to the CTMC of Figure 2.4, for computing the utilization, whereas for the average number of jobs, the reward vector should be $r = (0, 1, 2, 3)$.

Besides performance, dependability aspects deserve great attention for the assurance of the quality of the service provided by a system. Dependability studies look for determining reliability, availability, security and safety metrics for the infrastructure under analysis [Malhotra and Trivedi 1994]. Reliability block diagrams [Shooman 1970], fault trees [Watson 1961] and Petri nets are, as well as Markov chains, widely used to capture the sys-

tem behavior and allow the description and prediction of dependability metrics.

The CTMC of Figure 2.2 represents the most basic availability aspects of a system, which may be applied to computational and non-computational environments. Failure and repair events bring the system to different configurations and operational modes, and these changes are expressed by the transitions from one state to another in the Markov chain. The steady-state availability is a common measure extracted from this kind of model and, in this case, it can be computed as the steady-state probability of being in state Up: π_{Up}. The downtime D in a given period T can also be obtained in a straightforward manner: $D = (1 - \pi_{Up}) \cdot T$.

The combined analysis of performance and dependability aspects, so-called performability analysis, is other frequent necessity when dealing with computer systems, since many of them may continue working after partial failures. Such gracefully degrading systems [Beaudry 1978] require specific methods in order to achieve an accurate evaluation of their metrics. As mentioned in previous section, Markov reward models constitute an important framework for performability analysis. In this context, the hierarchical modeling approach is also an useful alternative, in which top-level MRMs may be used to model the dependability relationships of the system, and the performance results from lower-level CTMCs provided the rewards for each state in the top-level Markov reward model. An example of such approach is found in [Ma et al. 2001]

For all kinds of analysis using Markov chains, an important aspect must be kept in mind: the exponential distribution of transition rates. The behavior of events in many computer systems may be fit better by other probability distributions, but in some of these situations the exponential distribution is considered an acceptable approximation, enabling the use of Markov models. It is also possible to adapt transition in Markov chains to represent other distributions by means of phase approximation, as shown in [Trivedi 2001]. The use of such technique allows the modeling of events described by distributions such as Weibull, hypoexponential, hyperexponential, Erlang and Cox.

2.2 Sensitivity Analysis

Sensitivity analysis (S.A.) is a method of determining the most influential factors in a system [Frank 1978, Hamby 1994]. The effect of changes in data distribution or the impact caused by changes in parameters are examples of study subjects for sensitivity analysis. According to [Hamby 1994], many authors consider that models are sensitive to input parameters in two distinct ways: (1) the variability, or uncertainty, of a sensitive input parameter has a large contribution to the overall output variability, and (2) there may be a high correlation between an input parameter and model results, so that small changes in the input value result in significant changes in the output. These two types of parametric sensitivity are handled using different types of analysis.

Figure 2.6 shows the context of sensitivity analysis for a given model used for prediction of the system behavior. Through this picture, it is possible to distinguish the focus of each type of parametric sensitivity analysis. A comprehensive set of approaches for the first case (related to uncertainty analysis) is found in [Saltelli et al. 2004], where sensitivity analysis is defined as "the study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input". Similar approaches can be found in [Galdino et al. 2007] and [Galdino and Maciel 2008]. Figure 2.6 shows that errors and uncertainty in the modeling may be due either to simplifications of real world made during the model construction or to errors and inaccuracy during the measurement of input parameter values. In contrast to this uncertainty-oriented analysis, there are scenarios in which parameters are already known with reasonable accuracy and precision, thereby having little uncertainty to add to the output, matching the second form of sensitivity cited previously, focused on the relative importance of each parameter. For such cases, which are the focus of this work, S.A. is mainly used to elaborate a list of the input parameters sorted by the amount of contribution each one has on the model output.

When dealing with analytic models such as Markov chains, parametric sensitivity analysis is a particularly important technique for computing the effect on the measures of interest caused by changes in the transition rates. This approach may be used to find performance or reliability bottlenecks in the system, thus guiding the optimization process [Blake et al. 1988, Abdallah and Hamza 2002]. Another benefit of sensitivity analysis is the

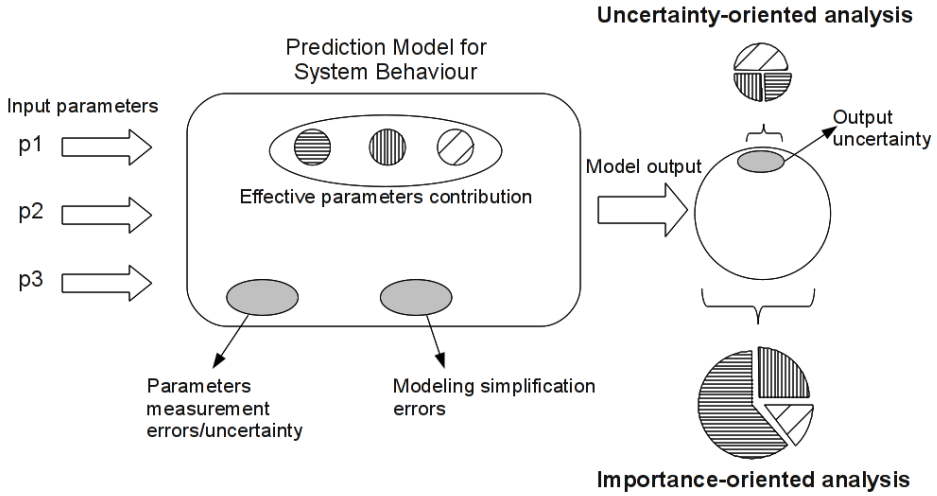

Figure 2.6: Types of parametric sensitivity analysis

identification of parameters which can be removed without significant effect to the results. Large models, with dozens of rates, may be drastically reduced by using this approach.

2.2.1 "One parameter at a time" approach

There are many ways of conducting parametric sensitivity analysis. Some of them can be properly used in the analytical models like Markov chains, whereas other approaches are better suited to experimental measurement based analysis. The simplest method, in a conceptual view, is to repeatedly vary one parameter at a time, while keeping the others fixed [Hamby 1994]. When applying this method, the sensitivity ranking is obtained by noting the corresponding changes in the model output. This method is commonly used in conjunction with plots of input versus output. Such plots enable graphic detection of non-linearities, non-monotonicities, and correlations between model inputs and outputs [Marino et al. 2008]. Unexpected relationships between input and output variables may also be revealed with this approach, triggering the need for further investigations, based on different approaches [Hamby 1994]. A given percentage of the parameter's mean value may be used as the increment for the cited approach. Each parameter may also be increased by a factor of its standard deviation, in case this information is known [Downing et al. 1985].

Figure 2.7 shows a simple example of plot, in which a hypothetical measure Y is plotted against its input parameters: α, β and γ. In this case, the impact caused by each parameter variation is clearly distinguished among them. The result from sensitivity analysis using this

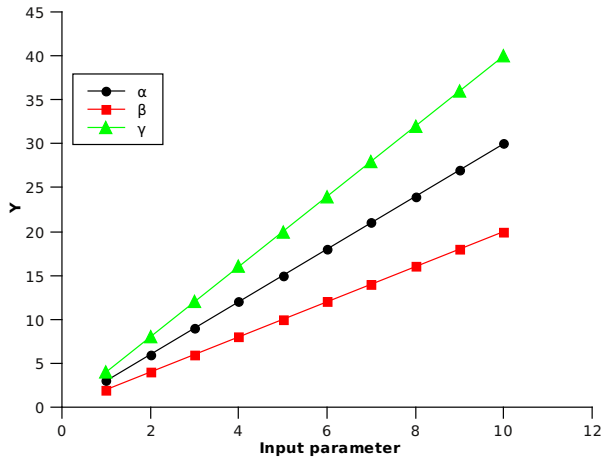

Figure 2.7: Example of plot for one parameter at a time S.A.

approach should be a sensitivity ranking with the following order: $\{1^{st}$: γ, 2^{nd}: α, 3^{rd}: $\beta\}$. Therefore, γ is considered to be the input parameter that cause the major influence on the measure Y.

Although, varying one parameter at a time is less useful in some opportunities. When the amount of parameters is large, the analysis of scatter plots becomes harder, mainly due to the proximity of curves. The difference in magnitude orders is another possible complicating factor, since all parameters cannot be visualized in the same plot, forbidding accurate interpretations about the differences among parameters influence. Due to such cases, methods that are based on numerical sensitivity indexes should have preference in spite of "one parameter at a time" approach.

2.2.2 Correlation and Regression Analysis

Correlation analysis and regression analysis are two important approaches to find how much a given variable has its variability associated to a single parameter. Pearson's correlation coefficient, also known as Pearson's r, is one of the most used indexes to measure correlation. According to [Ross 2010], considering data pairs (x_i, y_i), $i = 1, ..., n$, sample correlation coefficient is defined by Equation 2.13, where $\bar{x}$ and $\bar{y}$ denote the sample mean of the x values and the sample mean of the y values, respectively.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.13}$$

Notice that a paired set of observed values of x and y is needed to compute the correlation coefficient. When $r > 0$ the sample data pairs are said to be positively correlated, and when $r < 0$, they are negatively (or inversely) correlated. r is always between -1 and 1, and the larger is its value, stronger is the correlation between x and y.

The use of correlation coefficients in sensitivity analysis requires the assumption of linear relationships between the measure of interest and model's input parameters, what may not represent properly the behavior of the system under analysis. A possible correlation between input parameters would also difficult accurate interpretations of Pearson's r for sensitivity analysis.

According to [Hamby 1994], regression methods are often used to replace a highly complex model with a simplified "response surface", i.e., a regression equation that approximates model output using only the most sensitive model input parameters. A simple linear regression relationship is given by $E[Y] = a + bX$, that supposes a straight-line relationship between the mean value of the response Y and the value of the input variable X [Ross 2010]. Equation 2.14 presents a multiple linear regression equation, in which n input variables are considered.

$$E[Y] = a + \sum_{i}^{n} b_i X_i \tag{2.14}$$

The coefficients a and b must be estimated from experimental data. Coefficient a is the "Y-intercept" and it can be interpreted as the value predicted for Y if $X_i = 0, \forall i \in \{1, ..., n\}$. b_i is the regression coefficient, which represents the difference in the predicted value of Y for each one-unit difference in the input variable X_i. Therefore, a sensitivity ranking can be determined based on the relative magnitude of the regression coefficients. Standardization methods may be used to remove the influence of units and place all parameters on an equal level [Saltelli et al. 2004]. Non-linear regressions (e.g., quadratic, logarithmic) are also possible, and they may be transformed into linear regressions using $E[Y] = a+bZ$, where Z is any non-linear function $f(x)$.

2.2.3 Factorial Experimental Design

Another well-known technique is the factorial experimental design [Jain 1990]. In a full factorial design, every possible combination of configuration and workload is examined. In a given system, that have its performance affected by k factors (parameters), and each factor have

n possible levels (values), the number of experiments would be n^k. This method enables finding the effect of every factor, including the interactions among them. Although, this comprehensive study usually takes too much time and spend substantial resources.

The number of levels for each factor can be reduced as a strategy to deal with a large amount of factors. [Jain 1990] considers the 2^k factorial design as a popular approach, in which only two levels are evaluated for each factor. Since very often the effect of a factor is unidirectional, a good initial analysis may be done by experimenting at the minimum and the maximum level of the factor. This helps to decide if the difference in performance is significant enough to justify detailed examination. Also, after finding out which factors are relevant to the measure of interest, the list of factors may be reduced substantially and it becomes feasible to try more levels per factor.

A fractional factorial design is another alternative to the full factorial design. It avoids running all the experiments, by selecting a subset that contains a fair combination of main levels for the factors. Despite how much levels or factors are tested, factorial experimental design assumes that the system may be changed to perform some tests, what it is not always possible.

2.2.4 Differential Sensitivity Analysis

Differential analysis, also referred to as the direct method, is the backbone of many other sensitivity analysis techniques [Hamby 1994]. It may be performed in an efficient computational manner on analytic models commonly used in performance and dependability analysis. This method provides a single sensitivity coefficient that denotes the amount of change in the output that is produced by an incremental change of a given input.

Differential sensitivity analysis is based on the existence of an algebraic equation that describes the relationship between the measure of interest and the input parameters. Differential sensitivity analysis is performed by computing the partial derivatives of the measure of interest with respect to each input parameter. In [Frank 1978], this derivatives are referred to as **sensitivity functions**. Subsequently, the sensitivity function of a given measure Y, which depends on a parameter λ, is computed as in Equation 2.15, or 2.16 for a scaled sensitivity.

$$S_\lambda(Y) = \frac{\partial Y}{\partial \lambda} \tag{2.15}$$

$$SS_\lambda(Y) = \frac{\partial Y}{\partial \lambda}\left(\frac{\lambda}{Y}\right) \tag{2.16}$$

In Equation 2.16, other scaling methods can be used, depending on the nature of input parameters, the measure of interest and the need for removing the effects of units. $S_\lambda(Y)$ and $SS_\lambda(Y)$ are also referred to as sensitivity coefficients [Hamby 1994], whose ordered values produce the ranking that is used to compare the degree of influence among all parameters.

Independence of input parameters in relation to each other is often expected, but [Frank 1978] shows some cases and conditions for which dependences among the parameters can be considered. Among these cases are:

1. The parameters, $\lambda_1, \lambda_2, ..., \lambda_n$, are functions of a single variable, say a. Then Y may be considered as a function of a: $Y = f(a)$. The total derivative of f with respect to a may be obtained.

2. The parameters, $\lambda_1, \lambda_2, ..., \lambda_n$, are functions of another set of independent variables, say $a_1, a_2, ..., a_m$. Then Y may be considered as a function of the a_j's: $Y = f(a_1, a_2, ..., a_m)$. The partial derivatives of f with respect to $a_1, a_2, ..., a_m$ may be obtained.

3. The parameters, $\lambda_1, \lambda_2, ..., \lambda_{n-1}$, are functions of λ_n. Then Y may be considered as a function of λ_n only: $Y = f(\lambda_n)$. The total derivative of f with respect to λ_n may be obtained.

Frank [Frank 1978] also demonstrates that the product rule and quotient rule hold for any two functions, Y and Z, since their derivatives exist. This relations are expressed in Equations 2.17, and 2.18, respectively.

$$S_\lambda(Y.Z) = Y.S_\lambda(Z) + Z.S_\lambda(Y) \tag{2.17}$$

$$S_\lambda(Y/Z) = \frac{Z.S_\lambda(Y) - Y.S_\lambda(Z)}{Z^2} \tag{2.18}$$

If $Y = f(\beta)$ and $\beta = g(\lambda)$, then the chain rule holds, as shown in Equation 2.19.

$$S_\lambda(Y) = S_\beta(Y).S_\lambda(\beta) \tag{2.19}$$

For scaled sensitivities, also referred to as relative sensitivities [Frank 1978], the same relations are described by Equations 2.20, 2.21, and 2.22.

$$SS_\lambda(Y.Z) = SS_\lambda(Y) + SS_\lambda(Z) \tag{2.20}$$

$$SS_\lambda(Y/Z) = SS_\lambda(Y) - SS_\lambda(Z) \tag{2.21}$$

$$SS_\lambda(Y) = SS_\beta(Y).SS_\lambda(\beta) \tag{2.22}$$

Therefore, if there exists some kind of relationship among the parameters, and the relations cited in [Frank 1978] are not considered, the sensitivity analysis results may lead to inaccurate interpretations, invalidating the importance assessment. Such fact is not a problem for most of Markov models created to analyze computational performance and dependability aspects, because rate transitions are often defined as functions of independent parameters, and the sensitivity analysis is performed with respect to each of those parameters. When this does not happen, the existing relationships are included in those cases previously cited.

Another important consideration is that the function describing the measure of interest must be differentiable in the analyzed point. It is important to highlight that, if the measure Y behaves like a linear (first-order polynomial) function of λ when all the other parameters are fixed, $S_\lambda(Y)$ will be a constant, valid for all values of λ. This is the case of measure Y, expressed by Equation 2.23. Figure 2.7 shows the changes in Y, according to changes in each one of the three input parameters, while keeping the others fixed. Equations 2.24, 2.25 and 2.26 show the sensitivities of Y with respect to α, β and γ, respectively. The parameter γ has the biggest effect on the measure Y, followed by α and β, and due to the linear nature of the measure Y with respect to all parameters, the sensitivity ranking remains unchanged for all input values.

$$Y = 3\alpha + 2\beta + 4\gamma \tag{2.23}$$

$$S_\alpha(Y) = \frac{\partial Y}{\partial \alpha} = 3 \tag{2.24}$$

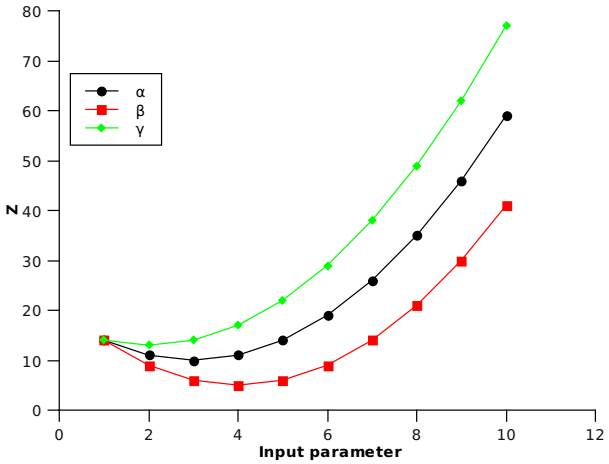

Figure 2.8: Plot for non-linear and non-monotonic function

$$S_\beta(Y) = \frac{\partial Y}{\partial \beta} = 2 \tag{2.25}$$

$$S_\gamma(Y) = \frac{\partial Y}{\partial \gamma} = 4 \tag{2.26}$$

Differential S.A. is closely related to the approach in which one parameter at a time is changed and plotted against the result in the measure Y. The sensitivity coefficient may be understood as the slope of the corresponding line for a specific point in the plot. From this view, it is possible to notice that interpretation of analysis results must be more careful if the parameters can be far removed from the base values, mainly if the function is not linear or not monotonic. An example of a non-linear and non-monotonic function is $Z = (\alpha - 3)^2 + (\beta - 4)^2 + (\gamma - 2)^2$, depicted in Figure 2.8, in which the slopes of curves vary in each point of the analysis, so the sensitivity of Z with respect of each parameter quantify the impact of changes just in regions close to that analyzed point.

When carrying computer performance and dependability analyses, it is common looking for incremental improvements in system configuration, so the localized range of sensitivity results is well fitted in this context.

Besides computer performance evaluation, sensitivity functions based on partial derivatives are widely used in areas such as signal processing, economics, chemistry and physics [Saltelli et al. 2004, Hamby 1994]. In such fields, the improvement of model accuracy is often the major goal, through additional measurements of parameters with the highest sensitivity

coefficients.

2.2.5 Sensitivity Analysis Methods for Performance and Dependability Evaluation

The main performance and dependability modeling techniques for computer systems can undergo a differential sensitivity analysis. This is due to the existence of algebraic equations that provide exact solutions, or at least approximate estimates to many of the metrics commonly found in such studies. The derivative of equations, as well as the adaptations in corresponding algorithms, has been published by some researchers, that used various types of systems to validate their accomplishments.

There are further approaches, similar to the differential sensitivity analysis, that have been proposed in the field of performance and dependability evaluation. Perturbation Analysis (PA) is one of the related methods. It calculates the parametric sensitivity of discrete event dynamic systems (DEDS). It can be used to determine the perturbed performance value (i.e. the performance after a change in system's input parameters) without the need of actually carrying out the experiment in which the parameter values are changed (perturbed experiment) [Ho 1985]. Perturbation Analysis recognizes the timing of events as the most basic element in the description of a DEDS. Therefore, considering a dynamic system, the calculation of sensitivities for a performance measure, hereafter called PM, is given by Equation 2.27, that assumes the existence of two kinds of events: events i, which have their timing directly affected by the change of a system parameter θ, and events k, which have their timing affected by the change in the timing of events i. So, $\frac{\partial t_i}{\partial \theta}$ is the change in timing of events i caused by a change in the value of a system parameter, θ, $\frac{\partial t_k}{\partial t_i}$ is the change in the timing of events k caused by a change in the timing of a event i, and $\frac{\partial PM}{\partial t_k}$ is the change in system performance caused by a change in the timing of a event k.

$$\frac{\partial PM}{\partial \theta} = \sum_i \frac{\partial PM}{\partial t_k} \frac{\partial t_k}{\partial t_i} \frac{\partial t_i}{\partial \theta} \tag{2.27}$$

The method of Perturbation Analysis is also employed in simulation studies which search for estimates of measures derivatives. In the context of simulation of DEDS, the likelihood ratio

[Reiman and Weiss 1989] is another method that was proposed. Highly dependable systems, as well as generic queueing systems, are some application fields for which this approach was already validated.

Other sensitivity analysis method that may be used for DEDS is based on combining a original and a perturbed Markov chain, to construct an *augmented* chain, and further a *reduced augmented* chain [Cassandras and Strickland 1989]. Given a parameter change, the state probability sensitivities for the original system may be expressed as simple linear combinations of the reduced augmented chain state probabilities. This approach was shown to be useful in finding performance measures sensitivities for various queueing systems.

In the context of reliability modeling, one approach to sensitivity analysis is to use upper and lower bounds on each parameter in the model to compute optimistic and conservative bounds on system reliability. [Smotherman et al. 1986] shows that conservative and optimistic reliability models can be derived from more complex models, reducing the state space and the amount of transitions. Subsequently, this pair of models may have their parameters changed one at a time to determine those ones that bring the biggest variation to the reliability computed from the original model.

Reliability importance indices, such as Birnbaum's Component Importance [Birnbaum 1969] and Component Criticality Importance [Wang et al. 2004], are in fact differential sensitivity analysis techniques. Equation 2.28 defines Birnbaum's Component Importance, where $I_k^B(t)$ is the reliability importance of the k^{th} component, $R_S(t)$ is the system reliability at time t and $R_k(t)$ is the reliability of component k at time t.

$$I_k^B(t) = \frac{\partial R_S(t)}{\partial R_k(t)} \tag{2.28}$$

According to [Wang et al. 2004], whereas the Birnbaum importance provided the probability that a given component would be responsible for the failure at time t, Component Criticality Importance can be used to determine the probability that the given component was responsible for system failure before time t. This measure is given by Equation 2.29 for a system S, where $F_S(t)$ is the system unreliability and $F_k(t)$ is the unreliability of component k at time t.

$$I_k^C(t) = \frac{\partial R_S(t)}{\partial R_k(t)} \cdot \frac{F_k(t)}{F_S(t)} = I_k^B(t) \cdot \frac{F_k(t)}{F_S(t)}. \tag{2.29}$$

Structural importance is another measure for ranking components based on their influence on overall system reliability. Such metric does not use partial derivatives, but it uses the differences between system state when the chosen component is up and when that component has failed. Details about the calculation of structural importance are found in [Kuo and Zuo 2003], which also states that the structural importance actually measures the importance of the position of the component. It is independent of the reliability value of the component under consideration. Structural importance and Birnbaum's component importance are among the most used reliability importance indices [Fricks and Trivedi 2003].

Chapter 3

Related work

In the fields of performance and dependability evaluation, it is possible to find a number of researchers that have already demonstrated how to perform parametric sensitivity analysis in some analytic models. One of the most outstanding papers in this topic is found in [Blake et al. 1988], which presents the foundations for transient sensitivity analysis in continuous time Markov chains (CTMC) and Markov reward models, and shows how the sensitivity functions can guide system optimization, model refinement and the detection of reliability/performability bottlenecks.

The development of differential S.A. methods followed similar ways for all modeling formalisms, mainly when only state-space models are considered. The correctness and utility of such approach in each of the studied models is well founded, but little guidance is found about whether scaled or unscaled sensitivities should be used. In [Blake et al. 1988], scaled sensitivities are used only for model refinement. By contrast, [Sato and Trivedi 2007] and [Muppala and Trivedi 1990] use scaled sensitivities for system optimization and bottleneck detection purposes, whereas [Xing and Dugan 2002] and [Bondavalli et al. 1999] do not employ any scaling factor to the sensitivity functions they use to enhance reliability and dependability metrics of the systems under analysis. An important consideration to be done is that, in optimization studies such as those just cited, the benefits of using scaled or unscaled sensitivities are not justified, as far as was seen in the literature.

The following sections present some related works based on analytic models, focusing on Markov chains, their derived MRMs, and other similar formalisms.

3.1 Differential S.A. on Queueing Systems

Queueing system is one example of analytic model whose sensitivity analysis has been described in literature. Yin [Yin et al. 2007] give sensitivity formulas for the performance of $M/G/1$[1] queueing systems, which are described by semi-Markov processes. They show that the embedded Markov chain of a $M/G/1$ model may be used to provide the desired steady-state sensitivity measures. This is possible because the semi-Markov process has always the same steady-state probabilities as the embedded Markov chain. In [Yin et al. 2007], the sensitivity analysis of $M/M/1$ and $M/C_2/1$[2] queueing systems is also discussed, since they can be considered as specialized versions of the $M/G/1$ case.

Opdahl [Opdahl 1995] presents sensitivity functions for the performance of open queue networks[3], in a combined analysis of hardware and software performance. The software performance models are directly derived from design specifications annotated with performance parameters to minimize additional modeling effort. A bank example is used to demonstrate the approach, and sensitivity measures are validated by means of plots of the predicted residence time for different parameter values.

An approach for sensitivity estimation in closed queueing networks is found in [Cao 1996]. It is based on Perturbation Analysis [Ho 1985] and Likelihood Ratio [Reiman and Weiss 1986] methods. Instead of actual differentiation of the performance measure of interest, the algorithm proposed in his work uses a sample path of the model to estimate the derivative of the steady-state probability vector. Networks with general service time distributions may be analyzed using those sensitivity estimates.

No scaling or normalization methods are used in that paper. This is a characteristic of most works it was possible to read in the literature about sensitivity of queueing systems, except by [Liu and Nain 1991], that propose general formulas to quantify the effects of changing the model parameters in open, closed, and mixed product-form queueing networks. These formulas encompass the derivative of the expectation of known functions of the state of the network with respect to any model parameter (i.e., arrival rate, mean service demand, service rate, visit ratio,

[1] A queue $M/G/1$ has a service time that follows an arbitrary (general) distribution, in contrast to the exponential nature of service time in a queue $M/M/1$ [Kleinrock 1975].

[2] A queue $M/C_2/1$ has a service time that follows a two-stage Coxian distribution [Yin et al. 2007]

[3] Queueing networks whose operations all have transaction workload intensities are open, while other queueing networks are closed or mixed [Opdahl 1995].

traffic intensity). The sensitivity functions for the throughput and queue length are presented in that paper, which also demonstrates an example of cost-based optimization.

3.2 Differential S.A. on Markov Chains

Stages of the differential sensitivity analysis of Markov chains include computing the derivative of the rate generator matrix and the differentiation of equations used in the solution methods (presented in Section 2.1), or even the development of new solution methods. Marie [Marie et al. 1987] presents a sensitivity analysis method regarding transient and cumulative measures in acyclic Markov chains. The ACE (**A**cyclic Markov **C**hain **E**valuator) algorithm is used to find the state probabilities of an acyclic CTMC as a symbolic function of t, and it is adapted to the compute the respective sensitivity functions.

[Blake et al. 1988] show how to compute the same measures of the [Marie et al. 1987] work, but using the uniformization technique [Heidelberger and Goyal 1987], which allows the analysis of more general models, with cycles. The sensitivity functions are applied in a reliability/performability study, which also introduces the sensitivity of expected reward rate and a specific sensitivity function for the mean time to failure (MTTF) of a system.

The analysis in [Blake et al. 1988] uses a cost-scaled sensitivity to find performability bottlenecks in three models of multiprocessor systems, constructed from processors, shared memories and a interconnection network. The cost function is based on three parts: the amount of copies of a given component, its failure rate, and the unitary cost of that component. The transient sensitivity analysis of the unreliability shows that there are specific moments when the importance of a component becomes critical. It also reveals the difference in the order of components importance, according to the architecture under analysis. Parameter-scaled sensitivity is only employed to model refinement, finding the most effective way to improve the accuracy of model outputs.

Another related study is shown in [Ou and Dugan 2003], that developed an approximate approach for the computation of sensitivity analysis in acyclic Markov reliability models, reducing the computation time for large models. That approach is used for solving a dynamic fault tree and hence assessing the importance of each component according to its failure probability.

The method is compared to Birnbaum's reliability importance index [Birnbaum 1969], but the algorithm presented in [Ou and Dugan 2003] considers situations when components have individual failure behavior, and when there is some dependency between the operational state of a component and the failure of other components, that is the case of spare parts. That work also presents the computation of sensitivities for modules of some components, that can be combined to produce the system level sensitivities. A chain-rule approach is used to calculate sensitivity measures for the separate modules, and to combine them hierarchically for higher-level results.

A different method to deal with the sensitivity computation of the expected accumulated reward is found in [Abdallah and Hamza 2002]. The work proposes an approach based on the uniformized power (UP) technique presented in [Abdallah 1997], as an alternative to the standard uniformization (SU) method. This method aims to save computation time if the state space size is moderate and the mission time is long. Other objective of this approach is to reduce the computation time for stiff models, that is the case of life critical systems for which the failure rates are much smaller than the repair rates.

Continuous time Markov reward processes are also studied in [Grassi and Donatiello 1992]. They evaluate a closed-form expression for the derivative of the cumulative reward probability distribution over a finite time interval, thus enabling a sensitivity analysis in the framework of performability evaluation of fault-tolerant systems. Both transition rates and reward rates are assumed to be function of a given system parameter, reflecting how a design change can have an impact on system reliability (transition rates) as well as on system performance (reward rates). The computational complexity of the analytical expression is shown to be polynomial in the number of processes states and reward rates. Hence it is a good substitute to other approaches, considering the scope of fault-tolerant systems.

[Haverkort and Meeuwissen 1992] also presents a study related to Markov reward models, but in that paper sensitivity analysis is used to deal with uncertain parameters and is compared to a Monte Carlo based uncertainty propagation. They analyze the uncertainty in coverage factors of failures in a specific computer system. Since the performability metric was defined as a function of failure rates, they uses the chain rule to obtain the sensitivity with respect to the coverage parameter. Stochastic Petri nets were used to analyze indirectly the underlying Markov reward model, by means of the SPNP [Hirel et al. 2010] package. This analysis allowed to

detect that an increase in the bus coverage yields a higher system dependability but, surprisingly, a lower system performability, due to certain reward configurations.

In [Sato and Trivedi 2007], Markov chains were created for a travel agent system. They performed a sensitivity analysis of response time and reliability metrics with respect to each parameter. Despite using Markov chains for computing the metrics of interest, closed-form equations are found for both measures, and the differential sensitivity analysis is carried out using these equations. The authors highlight that closed-form equations can not be found for all systems, so a model-based sensitivity analysis would be helpful for a broader range of situations.

3.3 Differential S. A. on Petri Nets

Among some models derived from Petri nets, generalized stochastic Petri net (GSPN) is one of the most flexible approaches, providing useful modeling mechanisms to represent concurrency, parallelism and synchronization in complex systems. They are an extension of the stochastic Petri nets formalism [Molloy 1982], and allows both immediate and timed (exponentially distributed) transitions.

[Muppala and Trivedi 1990] introduce a process to compute sensitivity functions of GSPNs. Since the reduced reachability graph of a GSPN is a continuous-time Markov chain, it is possible to translate the process of S.A. in CTMCs to a GSPN-based sensitivity analysis. In order to accomplish this task, sets of vanishing markings (V) and tangible markings (T) are identified. Therefore, the probabilities and rates of transition between markings are used to obtain the generator matrix of the underlying CTMC.

[Muppala and Trivedi 1990] demonstrate the derivative of equations for steady-state, transient and cumulative measures in their work, which also includes the implementation of sensitivity analysis features in the SPNP package [Hirel et al. 2010] and the analysis of a processor cluster system, to show the applicability of the approach. They compute normalized (scaled) sensitivities for the probability of rejection of jobs with respect to each parameter, therefore detecting the performance bottleneck of that system.

In [Ciardo et al. 1993], the definition of Stochastic Reward Nets is complemented by the demonstration of sensitivity formulas for that model. It follows a process that is similar to that

for GSPN models, in which tangible and vanishing markings shall be identified first, as well as the transitions that may occur in these sets of markings.

[Choi et al. 1993] constitute the first work to elaborate a method for parametric sensitivity analysis of deterministic and stochastic Petri nets (DSPNs) [German and Mitzlaff 1995], which are an extension to GSPN models. DSPNs allow the association of a timed transition either with a deterministic or an exponentially distributed firing delay. Some characteristics of the solution for GSPNs, found in [Muppala and Trivedi 1990], are used in that work, but the analysis of a DSPN requires additional steps, since other stochastic processes (semi-Markov process and non-Markov DSPN process) are involved in this type of model. The usefulness of the sensitivity functions for DSPNs is shown in the optimization of a polling system with vacation, by finding a point where the derivative of the measure of interest is zero, i.e., that value of a given parameter maximizes the system performance.

Chapter 4

A methodology for systems optimization using differential sensitivity analysis

Sensitivity analysis is a common demand for anyone who creates and uses analytic models. The majority of performance and dependability analyses do not include the computation of quantitative sensitivity indices. Sensitivity analysis, when done, is executed simply by varying the set of parameters over their ranges and repeatedly solving the chosen model. Formal or differential sensitivity analysis is not commonly used, what makes the determination of bottlenecks more difficult. Hence, system optimization is usually performed in a inefficient and manual process.

It is worth observing that such scenario had no modifications in spite of important developments on differential sensitivity functions, mentioned in the last chapter. A likely reason for this lack of usage of formal sensitivity analysis techniques is the absence of a well-defined methodology for S.A. in computer systems stochastic modeling. Moreover, there is a gap in the S.A. approaches, related to the scaling (normalization) of the sensitivity index. No guidance is found about the proper normalization methods for the performance/dependability metrics that are usual in computer systems analysis. In few cases, the use of scaled, or unscaled, sensitivities is based on a formal approach.

This chapter presents the main contributions of this dissertation. The first one is an iterative methodology for S.A., that is supported by a modeling tool, aiming to provide a efficient and unambiguous way to perform system optimization and model enhancement. Such methodology

uses Markov chains as formal modeling technique, and enables cost-constrained optimization of computer systems. The second contribution of the current work is to present directions about the decision of scaling sensitivity indices. The formal basis for the proposed decision criteria is also presented here.

4.1 Methodology description

Based on the study of many research papers and reports about performance and dependability analysis, some common steps were found regarding the improvement of a given system, despite numerical sensitivity indices are rarely used. Besides, as stated before, the choice between using scaled or unscaled sensitivity indices is always made in a arbitrary manner. Figure 4.1 depicts the flowchart of the proposed methodology for systems optimization through sensitivity analysis. It uses some of the common steps of analytic modeling solutions and fulfills the mentioned deficits. The focus of that approach is to find bottlenecks and guide optimization of specific measures, defined by the system administrator. Such optimization shall be conducted by incremental changes in the parameters. If structural changes are executed, the process returns to its beginning, since the model should be redefined.

The dashed rectangles represent activities exclusively related to constrained optimization. Subsequently, they would be executed only if cost information is available and is considered as relevant for the current analysis. All activities that compose this methodology are described as follows.

Creation of model: The definition and creation of the model for the system under analysis is required before any other step. Since Markov chains are the mathematical formalism chosen here, global states must be identified and enumerated, as well as the transitions which may occur between certain states. The system characteristics and subsequent modeling decisions will determine if CTMCs, DTMCs or MRMs is used.

Definition of parameters: Given the created model, transition and/or reward rates must be defined as expressions using independent parameters. It is important to employ here all parameters, including those ones which are thought to have minimal influence on results, with no formal argument supporting such belief.

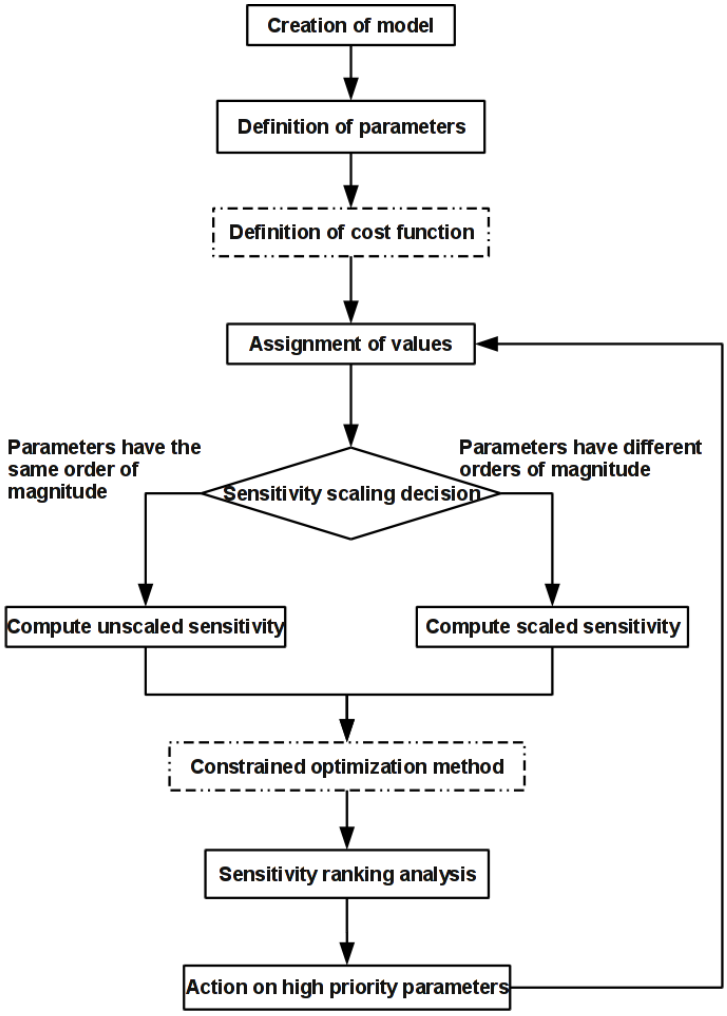

Figure 4.1: Flowchart for the proposed systems optimization methodology

Definition of cost function: The analyst may define a cost function, based on both Markov model parameters and other cost-specific parameters, which constitutes a constraint for the system optimization.

Assignment of values: Estimated or accurately known values are assigned to the model parameters. In this methodology, all parameters have their values defined with acceptable accuracy, so that S.A. works as an strict system optimization technique, where uncertainty propagation is not the focus.

Sensitivity scaling decision: Based on the parameters values, assigned in the previous step, the analyst decides between using unscaled or scaled sensitivity.

Compute unscaled sensitivity: Parameters with the close orders of magnitude can be compared without sensitivity index scaling or normalization. Strict performance and pure

reliability analyses are often found in such situation.

Compute scaled sensitivity: Parameters with the different orders of magnitude should be compared using scaled sensitivity indices. This is a common case for availability and performability analysis, which are known by the merge of fast and slow events, reflecting in parameter values.

Constrained optimization method: A mathematical optimization method may be employed here, using the defined cost function and the sensitivity indices computed in the previous step. This optimization method produces a cost-aware sensitivity ranking.

Sensitivity ranking analysis: The sensitivity ranking must be analyzed in a direct way, looking to the parameters on the top and on the bottom. Parameters having similar sensitivity indices should be considered with equivalent effect on the measure of interest, and other criterion may be employed to distinguish them in the ranking.

Action on high priority parameters: System administrators must concentrate in the enhancement of parameters with high sensitivity indices, either to achieve system optimization, or to perform model refinement. The last parameters in sensitivity ranking may be removed from the model, if model simplification is a goal.

After the necessary actions on high priority parameters, the result of such changes on the measure of interest is analyzed and, if required, a new sensitivity analysis iteration is executed, using the changed values of parameters. Such analysis may result in a sensitivity ranking that is different from the previous one, because the parameters which were improved have lost importance, i.e., their respective sensitivity indices were reduced, while other ones became the current bottlenecks for the optimization of the measure of interest. It is also possible that no changes occur in the sensitivity ranking from one iteration to another.

An example of the sensitivity reduction situation just described may be found in an availability analysis, when the repair rate (inverse of mean time to repair) of a given component A is increased until it reaches an optimal condition, so changes in the repair of other components cause higher impact on overall reliability than an additional change in A. Figure 4.2 illustrates this behavior in a plot of component repair rates versus system availability, characterized by the

reduced slope near to the current value of component A repair rate, while changes in the current repair rate of component B are able to increase more effectively the system availability. In such case, a repair rate may be closely related to the number of people allocated to the repair of that component, so human resources planning is a possible achievement from this analysis.

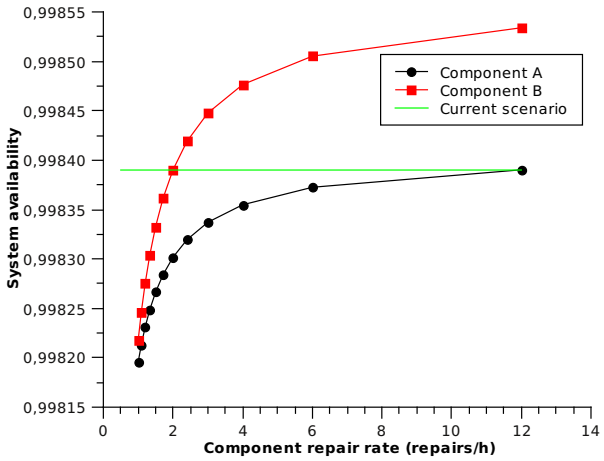

Figure 4.2: Example of sensitivity reduction in an availability analysis

4.2 Methodology steps for constrained optimization

It is common to ignore financial costs when a performance or dependability model is built. So, the computation of expenses in system enhancement and estimates of budget gains is left to the last analysis phase, using these pieces of information in a fragmented manner. This kind of evaluation process is justified for some scenarios, in which all possible optimization actions have similar costs, or when the improvement of a metric of interest always bring benefits that are bigger than the efforts spent on such optimization.

However, if cost information is available in early phases of performance and dependability analysis, the steps related to cost definitions and constrained optimization should be followed, in order to find the most cost effective point of optimization, considering all model parameters and a mathematically justified trade-off between measure enhancement and investments in system modifications.

The cost function will likely depend on some of the parameters for that model, but other specific parameters may appear in this step. One can notice that this methodology is not restricted to direct financial costs, but energy consumption, working time, and similar expenses

may be the main constraint to be considered when defining the cost function.

Some constrained optimization methods may be employed in the proposed approach, what constitute another flexible point, adjustable to the system characteristics as well as to the analyst knowledge or preferences. In [Avriel 2003], there is a comprehensive list of methods, just to mention some of the most known: penalty function, reduced gradient, quadratic programming, and Lagrange multipliers.

A point that is always present, whatever be the chosen method, is the formulation of the optimization problem. A common way to describe this problem is:

$$\begin{aligned} &\text{Maximize:} && Y \\ &\text{Subject to:} && C \leq Max\_cost\ , \end{aligned}$$

where Y is the measure of interest, C is the cost function, and Max_cost is the maximum acceptable cost defined for the system under analysis.

In this work, the classical method of Lagrange multipliers is suggested, due to its well-known optimization capabilities, ease of use and proximity to differential sensitivity analysis. Basically, this approach leads one to assume that, considering the measure Y and parameters θ_i, optimal values of θ_i satisfy the Equation 4.1, where k is a constant, and C is the system cost for the given value of θ_i.

$$\frac{\partial Y}{\partial \theta_i} = k \frac{\partial C}{\partial \theta_i} \tag{4.1}$$

The same equation can also be described as:

$$\left(\frac{\partial C}{\partial \theta_i}\right)^{-1} \cdot S_{\theta_i}(Y) = k. \tag{4.2}$$

So, the system bottleneck, regarding measure Y, may be stated as the parameter θ^*, that satisfies Equation 4.3, based on unscaled sensitivity of measure Y, and unscaled sensitivity of the cost function C. θ^* is the parameter θ_i that maximizes the relation $\left|\frac{S_{\theta_i}(Y)}{S_{\theta_i}(C)}\right|$. Changes in θ^* have the best trade-off between improvements on measure Y, and impact on costs. Besides this way of finding the optimal parameter, this work proposes an adaption that takes into account

the scaling decision step of the methodology presented in this chapter. If the use of scaled sensitivities is suggested in the methodology, due to the measure and parameters characteristics, so the Equation 4.4 should be considered, and $\theta^{\star}$ should be the parameter deserving higher priority in its improvement.

$$\theta^{*} = argmax_{\theta_i} \left| \frac{1}{S_{\theta_i}(C)} \cdot S_{\theta_i}(Y) \right|. \tag{4.3}$$

$$\theta^{\star} = argmax_{\theta_i} \left| \frac{1}{SS_{\theta_i}(C)} \cdot SS_{\theta_i}(Y) \right|. \tag{4.4}$$

Following the proposed methodology, a sensitivity ranking shall be generated, by listing the results of $\left|\frac{S_{\theta_i}(Y)}{S_{\theta_i}(C)}\right|$, or $\left|\frac{SS_{\theta_i}(Y)}{SS_{\theta_i}(C)}\right|$, for all parameters θ_i under analysis. The subsequent actions on high priority (and cost-effective) parameters will lead to other iteration, in which the system cost was already changed, as well as the chosen parameters. As previously mentioned, this iterative process stops when no enhancement is considered to be necessary for the system, but other possible reason for the stop is when cost restrictions can not be satisfied anymore. In such a situation, structural changes in the system may be necessary, leading to changes in the corresponding model, what would restart the process that is proposed here.

4.3 Decision support for the use of scaled sensitivity functions

The decision about employing scaled or unscaled sensitivity indices deserves special attention in this methodology. A scaled (also called relative) sensitivity function should be employed to remove the influence of big differences in absolute values of the parameters. The use of scaled sensitivities when all parameters have no big differences in their values may generate incorrect sensitivity rankings. Also, for models which comprehend distinct orders of magnitude, by scaling the sensitivity indices, one removes unfair influences of very large or very small units.

This property may be explained by the indirect relation between scaled sensitivity functions and logarithms, mentioned in [Frank 1978] and shown in Equation 4.5.

$$\frac{\partial\, ln\, Y}{\partial\, ln\, \lambda} = \frac{\frac{\partial\, Y}{Y}}{\frac{\partial\, \lambda}{\lambda}} = S_\lambda(Y)\frac{\lambda}{Y} = SS_\lambda(Y) \tag{4.5}$$

There may be a case in which the application of logarithm in the measure Y is not proper, because it is already a logarithmic measure, for example. For such situation, the logarithm may be applied only in the parameters, so the scaled sensitivity will be described by Equation 4.6. In this kind of scaling, the sensitivity is multiplied only by the parameter, instead of the ratio between parameter and measure of interest. $\widetilde{SS_\lambda}(Y)$ is also referred to as a semi-relative sensitivity function.

$$\widetilde{SS_\lambda}(Y) = \frac{\partial\, Y}{\partial\, ln\lambda} = \frac{\partial\, Y}{\partial\, \lambda/\lambda} = S_\lambda(Y)\lambda \tag{4.6}$$

Despite the mentioned relations between scaled sensitivities and natural logarithm, it was not possible to find in the literature similar uses for logarithms of other bases. The property of change of base can be applied in this context, so for the usual base 10: $log_{10}Y = \frac{ln\, Y}{ln\, 10}$. Therefore, the relation between scaled sensitivities and base 10 logarithms can be stated as in Equation 4.7, which shows that scaled sensitivities also represent the derivative of $\frac{\partial\, log_{10}\, Y}{\partial\, log_{10}\, \lambda}$.

$$\frac{\partial\, log_{10}Y}{\partial\, log_{10}\lambda} = \frac{\partial\left(\frac{ln\, Y}{ln\, 10}\right)}{\partial\left(\frac{ln\, \lambda}{ln\, 10}\right)} = \frac{\frac{1}{ln\, 10}\frac{\partial\, Y}{Y}}{\frac{1}{ln\, 10}\frac{\partial\, \lambda}{\lambda}} = \frac{\lambda\cdot(ln\, 10)\cdot\partial\, Y}{Y\cdot(ln\, 10)\cdot\partial\, \lambda} = \frac{\lambda\,\partial\, Y}{Y\,\partial\, \lambda} = S_\lambda(Y)\frac{\lambda}{Y} = SS_\lambda(Y) \tag{4.7}$$

Due to the property of change of base, the same relation holds for any logarithmic base. Such characteristic enables the use of scaled sensitivities for a range of scenarios that is broader than those seen in the literature by the moment. Therefore, this kind of analysis is proper for scenarios where the application of logarithm in the measure and its parameters is an acceptable method to counterbalance parameter values disparity. This depends on which measure is under analysis and what is the nature of input parameters.

Considering the usual scenario of computer systems availability analysis, there are recovery-related parameters, defined in a range of minutes or a few hours at most, and failure parameters, which commonly have thousands of hours as a minimum value. In an analytical model, all

values must be defined in the same unit, so in case of having all parameter in hours, this kind of analysis may handle a ratio $r = 10^3/10^{-1} = 10^4$ between parameter values, justifying the need for use logarithmic scale to compare their influences to the measure of interest.

It is also important to highlight that availability in real computer systems may also be measured with logarithmic metrics, such as **number of nines** [Marwah et al. 2010, Yu et al. 2006], what demonstrates that applying logarithm in availability analysis scenarios is a valid approach. So, a scaled sensitivity of availability will indirectly measure the impact that changes in a parameter have on the number of nines. But if the metric under analysis is already the number of nines, semi-relative sensitivity functions should be used, in order to avoid applying a logarithm on other logarithm.

For performability analysis, availability - or reliability - metrics are combined with performance measures. This combination may be direct, in specific formulas, or indirect, by means of composite and hierarchical models, as in [Ma et al. 2001]. Despite some differences in those approaches, again there is a need for repair and failure rates, and moreover, there are job completion rates - or similar parameters - which result in even bigger differences in the orders of magnitude. So, a relative sensitivity function, based on logarithms, as in Equations 4.5 and 4.7, fit the impact of each parameter on the performability metric.

In its turn, strict performance or pure reliability models often have parameters with similar ranges of values. For instance, in the usual case of performance studies, the maximum difference between parameters is from seconds to minutes, or hundreds of milliseconds to seconds. Therefore, in performance and reliability analysis, unscaled sensitivities fit the needs of comparison between the parameters. It is important to remember that this is just an indirect relation, and the main point in the decision is the actual scenario of parameter values.

The formal basis just presented and the analysis of some case studies suggest that a difference of three orders is a sufficient threshold for the scaling decision. Some of these case studies are presented in Chapter 6, and perform a cross-validation using graphical plots to reinforce the correctness of the decision support provided here.

4.4 A comparison with existing approaches

The methodology proposed in this work gathers existing modeling and evaluation techniques for computer systems performance and dependability analysis. The methodology introduces a structured vision of the entire process of systems improvement through differential sensitivity analysis, and it addresses some open topics in this area.

Since the sensitivity analysis is performed directly on Markov chains, there is no need to find closed-form equations that describe the system's behavior, as it was required in [Sato and Trivedi 2007]. This is an advantage because closed-form equations can not be found for all systems. The focus on analytic solution of the Markov chains also enables a more efficient computation of results, in comparison to simulation-based approaches, which is the case of Perturbation Analysis and Likelihood Ratio [Cao 2003], [Nakayama et al. 1994], and [Ho 1985].

While [Sato and Trivedi 2007], [Xing and Dugan 2002], [Bondavalli et al. 1999], [Opdahl 1995], and [Muppala and Trivedi 1990] use unscaled or scaled sensitivities without presenting any criterion, a clear point of decision is introduced here. Such decision is mainly based on the range of parameter values, and the characteristics of the measure of interest. This work also indicates relations between the type of analysis (availability, reliability, performance, or performability) and the most proper kind of sensitivity function. Such kind of relation is not found in any of the approaches already proposed.

The methodology also presents the adaption of Lagrange Multipliers method for using scaled sensitivities. This is an extension to the approach seen in [Blake et al. 1988], where only unscaled sensitivities are employed in the cost-aware optimization of a system. In this way, when the system is analyzed using scaled sensitivities, the same values can be used in the constrained optimization, in order to find the most cost-effective point to improve the system.

Chapter 5

Computer support for the proposed methodology

The automation of the process presented in Chapter 4 is very important to speed up the system evaluation and improvement through the sensitivity analysis. There is a number of software packages that allow the fast creation and solution of Markov chains, making the initial steps of the proposed methodology easier, so the modeler is able to concentrate in the observation of "real world" system, and in the correct translation of its characteristics to the model. Although, few tools include features to perform sensitivity analysis in the models. Specifically, no Markov chain tool is known to support differential sensitivity analysis. Therefore, the development of differential S.A. features in a Markov chain tool has arisen as a valuable step to the success of the proposed methodology.

Figure 5.1 shows an overview of the sensitivity computation process in a Markov chain modeling tool, planned and executed as part of this master's work. The upper dashed rectangle represents the user view and the lower dashed rectangle denotes the software backend view. A modeler who intends to perform a sensitivity analysis must build a Markov chain and define the transition rates of this chain in the form of symbolic parameters, such as $p, q, r, s,$ and t, or mathematical expressions using these parameters. After the model creation and parameters definition, parameters values have to be assigned and a measure of interest must be chosen, in order to solve the model and compute the sensitivities of this measure with respect to each input parameter. It may be observed that this user interaction with the modeling software is closely

related to the proposed methodology, depicted in Figure 4.1. These steps are also very similar to an usual interaction with modeling tools, without sensitivity analysis features. This similarity shows how much transparent is the integration of S.A. activity in the routine of systems modeling.

The backend steps, not visible to the user, begin with the creation of internal data structures for the Markov chain representation. The Q matrix is stored in a symbolic form, allowing its differentiation with respect to one of model parameters (e.g., q in Figure 5.1). The result of differentiation is the V matrix, corresponding to $\frac{\partial Q}{\partial q}$, essential to computation of transient and steady-state probability sensitivity. This process is the basis for determining sensitivity of all other measures. In Figure 5.1 the user requests only the sensitivity of $\pi(3)$, the probability of being in state 3, but it is important to highlight that the complete sensitivity vector, $S(\pi)$ is computed at once, despite only the information for the specific requested state is shown.

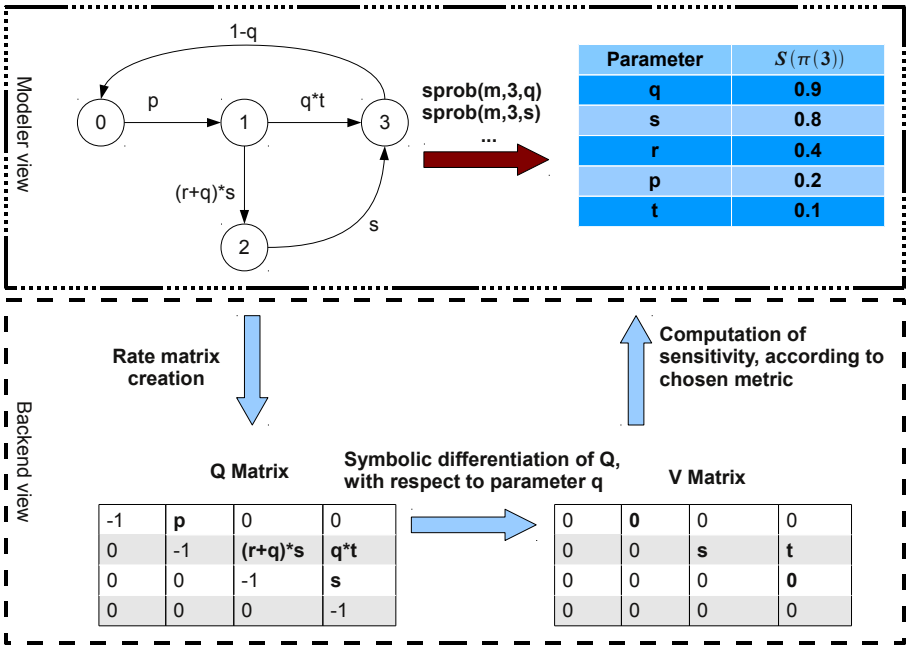

Figure 5.1: Overview of automated sensitivity computation

5.1 Development of S.A. features in the SHARPE tool

SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) [Trivedi and Sahner 2009] is a tool for specifying and analyzing performance, reliability and

performability models. It provides a specification language and solution methods for most of the commonly used model types for performance, reliability and performability modeling. Model types include combinatorial and state-space ones. Steady-state, transient and interval measures can be computed. A partnership with SHARPE developers allowed us obtaining its source code and implementing the desired S.A. features.

Note that SPNP [Hirel et al. 2010] package, a modeling tool for Stochastic Reward Nets, can implement sensitivity, but for them the derivatives of the underlying generator matrix may only be computed if the user supplies the rates in a very restricted form, and only the simple multiplication of parameters is permitted. Additions, subtractions and more complex functions cannot be used for the rates in SPNP. In its turn, SHARPE stores the generator matrix entries in a symbolic form, whereas the majority of tools store the entries in a numerical form. SHARPE therefore affords the opportunity to automate the whole process, without the limitations of other tools.

A set of representative metrics was chosen in order to implement their respective sensitivity functions in SHARPE. An important aspect of this decision was selecting both transient and steady-state metrics, as well as focusing on functions which would be helpful for CTMC and MRM evaluation. These criteria allowed the scope reduction whereas fulfilled a fair range of performance, dependability and performability modeling possibilities. The selected metrics were:

- Steady-state probability of the system being in state i;
- Expected steady-state reward rate of the system;
- Transient probability of the system being in state i at time t;
- Expected transient reward rate of the system at time t;
- Cumulative state probability over the interval $(0, t)$;
- Expected cumulative reward over the interval $(0, t)$.

These metrics are widely adopted to capture useful information about a variety of systems, and also serve as basis to compute other specific measures. As mentioned in Chapter 3, some

authors had already showed how to compute the sensitivity of each one of these metrics. The following subsections describe such procedures to find the chosen sensitivity functions, besides the programming-level aspects of the software development.

5.1.1 Sensitivity of steady-state probability

In order to compute the sensitivity of steady-state probability, the derivative of the following set of equations is required:

$$\pi Q = 0 \tag{5.1}$$

$$\sum_i \pi_i = 1 \tag{5.2}$$

In Equations 5.1 and 5.2, Q denotes the CTMC generator matrix and π represents the steady-state probability vector. After differentiation with respect to a parameter θ, the corresponding set of equations is obtained, as seen in Equations 5.3 and 5.4.

$$\frac{\partial \pi}{\partial \theta} Q = -\pi \frac{\partial Q}{\partial \theta} \tag{5.3}$$

$$\sum_i \frac{\partial \pi_i}{\partial \theta} = 0 \tag{5.4}$$

Details related to the solution of these equations using the Successive Over Relaxation (SOR) method are given in [Ciardo et al. 1993] and [Stewart 1994]. This solution was implemented in the SHARPE package in order to support the methodology proposed in this work. The developed features enable the computation of sensitivity of steady-state probability with the same computational complexity as the steady-state probability vector is already computed in SHARPE.

5.1.2 Sensitivity of expected steady-state reward rate

In a similar manner, in order to compute the sensitivity of $E[X]$, the expected steady-state reward rate, the derivative of Equation 5.5 is required. If the reward rates, r_i, associated with the model states are functions of the parameter θ, the sensitivity is expressed by Equation 5.6. If reward rates do not depend on this parameter, then the respective sensitivity is computed by

Equation 5.7. Both cases were considered in the function implemented in SHARPE.

$$E[X] = \sum_i r_i \pi_i \tag{5.5}$$

$$\frac{\partial E[X]}{\partial \theta} = \sum_i \frac{\partial r_i}{\partial \theta}\pi_i + \sum_i r_i \frac{\partial \pi_i}{\partial \theta} \tag{5.6}$$

$$\frac{\partial E[X]}{\partial \theta} = \sum_i r_i \frac{\partial \pi_i}{\partial \theta} \tag{5.7}$$

5.1.3 Sensitivity of transient state probability

For the transient (time-dependent) probability, the sensitivity computation is based on the uniformization method [Heidelberger and Goyal 1987]. The probability vector is found in an iterative process defined by Equation 5.8, where t is the time of interest, $\underline{P}(t)$ is the transient-state probability vector, q is selected so that $q > max_i|q_{ii}|$, and $Q^* = Q/q + I$. The differentiation of Equation 5.8 with respect to θ is expressed by Equation 5.9 where $\underline{\Pi}(i) = \underline{\Pi}(i-1)Q^*$ and $\underline{\Pi}(0) = \underline{P}(0)$.

$$\underline{P}(t) = \sum_{i=0}^{\infty} \underline{\Pi}(i) e^{-qt} \frac{(qt)^i}{i!} \tag{5.8}$$

$$\underline{S}(t) = \frac{\partial}{\partial \theta} \sum_{i=0}^{\infty} \underline{\Pi}(i) e^{-qt} \frac{(qt)^i}{i!} = \sum_{i=0}^{\infty} \underline{\Pi}(i)' e^{-qt} \frac{(qt)^i}{i!} \tag{5.9}$$

5.1.4 Sensitivity of expected transient reward rate

The reward rate of a system for a specific time t and its sensitivity are other useful metrics for performance, dependability and performability analysis. In order to compute the sensitivity of $E[X](t)$, the expected reward rate at time t, the derivative of Equation 5.10 is required. Similarly to the correspondent steady-state measure, if the reward rates, r_i, associated with the model states are functions of the parameter θ, the sensitivity is expressed by Equation 5.11. If reward rates do not depend on this parameter, then the respective sensitivity is computed by Equation 5.12. Both cases were considered in the function implemented in SHARPE.

$$E[X](t) = \sum_i r_i \pi_i(t) \tag{5.10}$$

$$\frac{\partial E[X](t)}{\partial \theta} = \sum_i \frac{\partial r_i}{\partial \theta} \pi_i(t) + \sum_i r_i \frac{\partial \pi_i(t)}{\partial \theta} \tag{5.11}$$

$$\frac{\partial E[X](t)}{\partial \theta} = \sum_i r_i \frac{\partial \pi_i(t)}{\partial \theta} \tag{5.12}$$

5.1.5 Sensitivity of cumulative state probability

As mentioned in Chapter 2, the cumulative state probabilities of a CTMC are represented by vector $\underline{L}(t)$ that satisfies the system of Equations 5.13.

$$\frac{d\underline{L}(t)}{dt} = \underline{L}(t)Q + \underline{\pi}(0),\ \underline{L}(0) = 0 \tag{5.13}$$

As demonstrated in [Reibman and Trivedi 1989], the uniformization method, used in the computation of transient probabilities, can also be used to calculate the vector $\underline{L}(t)$, as shown in Equation 5.14, where $\Pi(i)$ is the same vector of Equation 5.8.

$$\underline{L}(t) = \frac{1}{q} \sum_{i=0}^{\infty} \underline{\Pi}(i) \cdot \sum_{j=i+1}^{\infty} e^{-qt} \frac{(qt)^j}{j!} \tag{5.14}$$

Equation 5.15 is an extension to the uniformization method, which allows computing $\frac{\partial \underline{L}(t)}{\partial \theta}$ without solving the ordinary differential equation that results from the partial derivative of Equation 5.13 with respect to θ.

$$\frac{\partial \underline{L}(t)}{\partial \theta} = \frac{1}{q} \sum_{i=0}^{\infty} \frac{\partial \underline{\Pi}(i)}{\partial \theta} \cdot \sum_{j=i+1}^{\infty} e^{-qt} \frac{(qt)^j}{j!} \tag{5.15}$$

5.1.6 Sensitivity of expected cumulative reward

The last measure to be presented is the sensitivity of expected cumulative reward. Since Equation 5.16 describes how to compute the expected cumulative reward, its derivative with respect to parameter λ is shown in Equation 5.17. As seen in these equations, $L(t)$, the cumulative

probability vector and its sensitivity are required to compute this measure.

$$E[Y(t)] = \underline{r} \cdot \underline{L}(t) \tag{5.16}$$

$$E[Y(t)] = \frac{\partial \underline{r}}{\partial \theta} \cdot \underline{L}(t) + \underline{r} \cdot \frac{\partial \underline{L}(t)}{\partial \theta} \tag{5.17}$$

5.2 Programming aspects of sensitivity functions development

To implement the sensitivity functions previously described, an essential step is the symbolic differentiation of the generator rate matrix entries, as well as reward rate entries. These symbolic computations were possible due to the use of GiNaC framework [GiNAC 2010]. It is a set of C++ libraries for manipulating symbolic mathematical expressions, that has support for linear algebra, including symbolic matrices and vectors.

The efficient integration of C++ functions, developed using GiNaC, with the C code of SHARPE was an important reason for choosing GiNaC among a number of mathematical packages and computer algebra systems available. Another positive aspect was that include the GiNaC libraries within SHARPE's code did not lead to a significant increase in the size of the final executable file.

The main features of GiNaC framework used for the development of sensitivity functions were the symbolic differentiation and symbolic to numerical evaluation. The sample code listed in Listing 5.1 shows how these features are employed in a function (named *symbolic_diff*, line 20) that calculates the partial derivative of a given vector or matrix with respect to one parameter, *param* argument in the function. The generator rate matrix is defined as the q matrix, which have symbolic entries, i.e., defined as character strings. A vector containing the names of all model parameters, also defined as character strings, is other argument for *symbolic_diff*, as well as other vector with their values. These two vectors are converted to GiNaC lists between lines 45 and 48 of the code. Between lines 53 and 63 the derivative of matrix q is computed, all symbols are assigned to their values and the resulting expressions are evaluated, in order to obtain a

Listing 5.1: The developed code of symbolic differentiation using GiNaC libraries

```
#include <iostream>
#include <stdio.h>
#include <cstdio>
#include <string.h>
#include <stdexcept>
#include <ginac/ginac.h>
#include <ginac/container.h>
using namespace std;
using namespace GiNaC;

extern "C" {void symbolic_diff(int array_size, char q[][80], double *v, char *param, int nsymbols,
 char names[][80], double *values);}

/*************************************************************************//**
 * @brief This function performs symbolic differentiation of an array.
 * @details The input array is q parameter. This function returns its derivative
 *  array, for a supplied value.
 *************************************************************************/

void symbolic_diff(int array_size, char q[][80], double *v, char *param, int nsymbols, char names[][80],
 double *values)
{

parser reader;
ex e[array_size];

try {
int i;
/* Store each rate as a GiNaC expression */
for (i=0;i<array_size;i++){
  e[i] = reader(q[i]);
}

/* Get all symbols (parameters) in the rates */
symtab table = reader.get_syms();

/* Look for the parameter and convert it to a GiNaC symbol*/
symbol x = table.find(param) != table.end() ? ex_to<symbol>(table[param]) : symbol(param);
int j;
ex temp;
double d;
lst l_names, l_values;

/* Create one list for the symbol names, and another to the symbol values */
for (i=0;i<nsymbols; i++){
    l_names.append(reader(names[i]));
    l_values.append(values[i]);
}

/* For each expression (rate in the matrix), differentiate with respect to
 * the parameter (x), and then substitute the symbol names for their values
 */
for (j=0;j<array_size;j++){
  temp = evalf(e[j].diff(x).subs(l_names,l_values));

  // ex_to<numeric> is an unsafe cast, so check the type first
  if (is_a<numeric>(temp)) {
    d = ex_to<numeric>(temp).to_double();
    v[j]=d;
  } else {
    cout << "Error in conversion symbolic -> numeric" << endl;
  }
}
}
catch (exception &p) {
  cerr << p.what() << endl;
}
}
```

numeric derived matrix, that is the v matrix. The v matrix will be used in SHARPE for the computation of the sensitivity functions previously described.

Since SHARPE already allowed the user to enter symbolic expressions to the transition rates of Markov chains, only little adaptations were needed during the integration of differentiation functions and the corresponding specific sensitivity function of each chosen measure. One of these adaptations was the inclusion of data structures for storing the reward rates in the symbolic form. Before, reward rates defined in a symbolic form were immediately evaluated and converted to their numerical form, so those symbolical expressions were not stored. That change was necessary in order to implement the sensitivity function described in Equation 5.7.

Another existing behavior of SHARPE package that was also modified is related to successive analyses of the same model. Before this work, SHARPE only analyzed the system if any of the following conditions hold:

1. The values of system parameters are different from last time;
2. The system has never been analyzed;
3. Variable bindings have changed since the last analysis;
4. The chain under analysis has absorbing states and either the new analysis is for rewards and the last was not or vice versa;
5. The requested analysis is a transient analysis and either the system has never been analyzed or the time is different from last time.

Preceding inclusion of sensitivity functions, described in this work, those conditions were used in SHARPE to avoid unnecessary computations. From now on, there is another possible situation in which the system must be analyzed: a sensitivity function is called and the input parameter, or time, for such function is not the same from the last analysis. This new condition appears because a distinct derivative matrix is computed depending on the parameter that is passed as an argument for the sensitivity function. Only the last computed derivative matrix is stored in order to save memory space, so to obtain the sensitivity with respect to a different parameter a new computation process is necessary.

Table 5.1: Sensitivity functions developed for SHARPE

Sensitivity function	**Description**	**Related function**
sprob	Sensitivity of steady-state probability	prob
stvalue	Sensitivity of transient probability	tvalue
sexrss	Sensitivity of expected steady-state reward rate	exrss
sexrt	Sensitivity of expected reward rate at time t	exrt
sctvalue	Sensitivity of cumulative state probability	-
scexrt	Sensitivity of expected cumulative reward at time t	cexrt

Finally, new elements were added to the syntax accepted in SHARPE input scripts. The keywords associated to the sensitivity functions are similar to the related steady-state and transient analysis functions that already existed in SHARPE. Table 5.1 shows these keywords, their description and corresponding related functions. Only the cumulative state probability is not available for computation in the SHARPE package. Such measure may be indirectly calculated by means of the cumulative reward at time t, if all reward rates were defined as equal to 1.

The complete syntax of the developed functions is found in Appendix A, where the arguments required by each function are explained in detail.

5.3 Example of SHARPE utilization for sensitivity analaysis

An example of how the sensitivity functions can be used is presented in Listing 5.2, that shows a script for analysis of a simple Markov model using SHARPE. This model, named *queue*, is shown in Figure 5.2, and it is based on the birth-death CTMC model of Figure 2.4. Reward rates were included, turning it into a Markov reward model for the single server queuing system, previously mentioned in Section 2.1.2. There are four states (Q0, Q1, Q2, and Q3), representing the amount of jobs in the system (including jobs being serviced and jobs waiting in the queue) and its transition rates depend on two parameters, namely λ, and μ. Reward rates are assigned to states in which there is at least one job waiting in the queue, as a penalty associated to the wait.

Between lines 24 and 34 of Listing 5.2, the following measures, and their sensitivities, are assigned to variables, for posterior computation: steady-state probability that the queue is full; probability that the queue is full after 2 minutes from system startup; and steady-state reward

Listing 5.2: A script for sensitivity analysis of a simple model using SHARPE and its new functions

```
*Markov chain definition
markov queue
  Q0 Q1 lambda
  Q1 Q2 lambda
  Q2 Q3 lambda
  Q3 Q2 mu
  Q2 Q1 mu
  Q1 Q0 mu
reward
  Q0 0
  Q1 0
  Q2 10
  Q3 20
end
  Q0 1.0
end

*Binding of values (in jobs/min) to model parameters
bind
  lambda 3
  mu 5
end

*Measures selected for analysis
var probFull prob(queue,Q3)
var sensFullLambda sprob(queue,Q3,lambda)
var sensFullMu sprob(queue,Q3,mu)
var probFull2min tvalue(2;queue,Q3)
var sensFull2minLambda stvalue(2;queue,Q3,lambda)
var sensFull2minMu stvalue(2;queue,Q3,mu)
var costQueue exrss(queue)
var sensCostLambda sexrss(queue,lambda)
var sensCostMu sexrss(queue,mu)


*Printing results for all measures:

*Probability of queue is full in steady-state
expr probFull

*Sensitivity of probFull with respect to lambda
expr sensFullLambda

*Sensitivity of probFull with respect to mu
expr sensFullMu

*Probability of queue is full 1 hour after startup
expr probFull2min

*Sensitivity of probFullHour with respect to lambda
expr sensFull2minLambda

*Sensitivity of probFullHour with respect to mu
expr sensFull2minMu

*Expected steady-state cost due to jobs waiting in the queue
expr costQueue

*Sensitivity of costQueue with respect to lambda
expr sensCostLambda

*Sensitivity of costQueue with respect to mu
expr sensCostMu

end
```

rate of the system (i.e., the expected cost due to queue penalties). The sensitivities are computed with respect to both parameters, λ and μ. For example, *sensFullLambda* is the sensitivity of steady-state probability of system being is state Q3 (queue is full) with respect to changes in λ, and it is computed through *sprob(queue,Q3,lambda)* - see line 26. The correspondent sensitivity with respect to changes in μ is associated to the variable *sensFullMu* in the code, and it is computed through *sprob(queue,Q3,mu)* - see line 27. The sensitivity of probability of queue is full at 2 minutes with respect to changes in λ is computed through *stvalue(2;queue,Q3,lambda)* - see line 29. The expected steady-state reward rate is associated to a variable named *costQueue*,

and its sensitivity with respect to changes in λ is computed through *sexrss(queue,lambda)* - see line 32. Between lines 39 and 63 of the code, all the defined metrics are evaluated and printed though the keyword *expr*. This evaluation considers the last values that were bound for the parameters λ and μ - see lines 20 and 21.

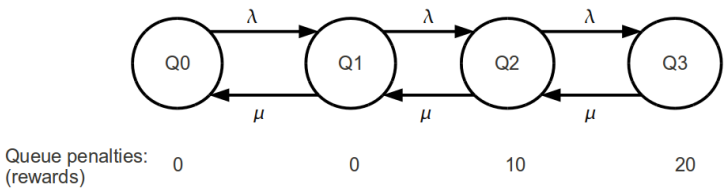

Figure 5.2: Markov reward model for a single server queueing system

The sensitivity analysis results, shown in Table 5.2, indicate that for the parameters configuration shown in Listing 5.2 ($\lambda = 3, \mu = 5$), incremental changes in job arrival rate, λ, will affect all measures in a higher extent than changes in service rate, μ, do. This analysis is valid for values near to the current ones. For a different configuration (see Conf. 2 in Table 5.2), the order of importance of these parameters may be swapped. It is important to highlight that the signal of the sensitivity index only means that there is an inverse relation between parameter and metric of interest: when one grows, the other is reduced, and vice-versa.

Table 5.2: Sensitivity results from example of Listing 5.2

Configuration	**Param.**	**sprob** [$\frac{\partial \pi_{Q3}}{\partial param.}$]	**stvalue** [$\frac{\partial \pi_{Q3}(2)}{\partial param.}$]	**sexrss** [$\frac{\partial E[X]}{\partial param.}$]
Conf. 1: $\lambda = 3$, $\mu = 5$	λ	0.069	0.068	1.991
	μ	-0.042	-0.040	-1.195
Conf. 2: $\lambda = 6$, $\mu = 5$	λ	0.068	0.069	1.490
	μ	-0.082	-0.081	-1.788

Note that, by analyzing metrics results, in Table 5.3, together with sensitivity results, in Table 5.2, important decisions may be made. When the system is found in Configuration 1, a system administrator may not need to perform any improvement action, because the $\pi_{Q3} = 9.9\%$, so the queue will hardly be full. By contrast, an action for system's performance enhancement is likely required in Configuration 2, since it presents 32.2% for the same metric. In this case, as parameter μ has the biggest impact on the probability of the queue is full, the administrator should try to improve the service rate (reducing the service time), instead of trying to reduce the arrival rate of new jobs, for example.

Table 5.3: Metrics results from example of Listing 5.2

Configuration	**prob [π_{Q3}]**	**tvalue [$\pi_{Q3}(2)$]**	**exrss [$E(X)$]**
Conf. 1	0.099	0.098	3.640
Conf. 2	0.322	0.321	9.121

In the example provided here, unscaled sensitivities are used, since the parameter values are close to each other. The functions implemented in the SHARPE package return unscaled sensitivities. When the user needs to scale the sensitivity, he can use the normal arithmetic expressions supported by SHARPE. For a complete reference on SHARPE's syntax (without sensitivity functions), see [Sahner et al. 1996].

Chapter 6

Case Studies

This chapter presents three case studies regarding parametric sensitivity analysis of different systems. The analyses presented here follow the methodology that was proposed and discussed in Chapter 4. The case studies encompass performance, reliability and availability modeling of systems. They are carried out to highlight bottlenecks in the respective systems, as well as provide examples on the accuracy of the proposed methodology, and illustrate its possible uses in systems improvement.

In Sections 6.1 and 6.2, the proposed methodology is used without the steps of constrained-optimization, since it is assumed that no cost function is available for those systems. Section 6.3 shows the analysis of the same system with and without cost-constrained optimization, enabling the comparison of the approaches. All case studies are conducted for only one iteration of the sensitivity analysis methodology, since the possible next iterations are straightforward, and would not add significant value to the understanding of the proposed approach.

6.1 Composite Web Services: Performance and Reliability Analysis

The first system analyzed was a travel agent process presented in [Sato and Trivedi 2007]. Such a system is composed of multiple Web services requiring individual steps to complete a travel reservation. In Figure 6.1, adapted from [Sato and Trivedi 2007], a UML activity diagram of this process is presented.

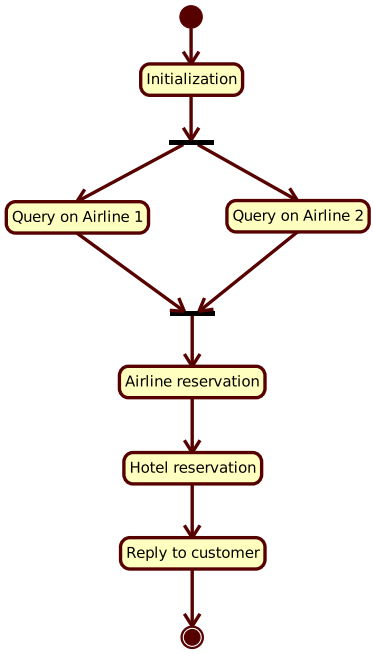

Figure 6.1: Travel agent process

The process of travel reservation, first, simultaneously searches for vacancy on two different airlines. When they respond, one is chosen based on some criterion such as fare or schedule. If one of the airlines fails to respond, the other is selected, and in the case of both failing to respond, the travel agent gives up and aborts. Other steps include effective airline invocation for reservation (after the selection), hotel reservation and success notification to the consumer. Any individual Web service may fail to respond, from which the system attempts to recover by means of a restart, except by the concurrent airlines searches.

6.1.1 Creation of models and definition of parameters for composite web services

A CTMC for this system is shown in Figure 6.2. This model, originally shown in [Sato and Trivedi 2007], was developed for computing performance and reliability of Web services and for detecting bottlenecks, by a formal sensitivity analysis based on closed-form equations.

The translation from the UML model activities to the Markov chain states is almost direct, but some states were added. RIni, RAresv, RHt, and RRep represent the states of Web service restart after a failure. For example, when the Initialization service fails, the model goes to RIni

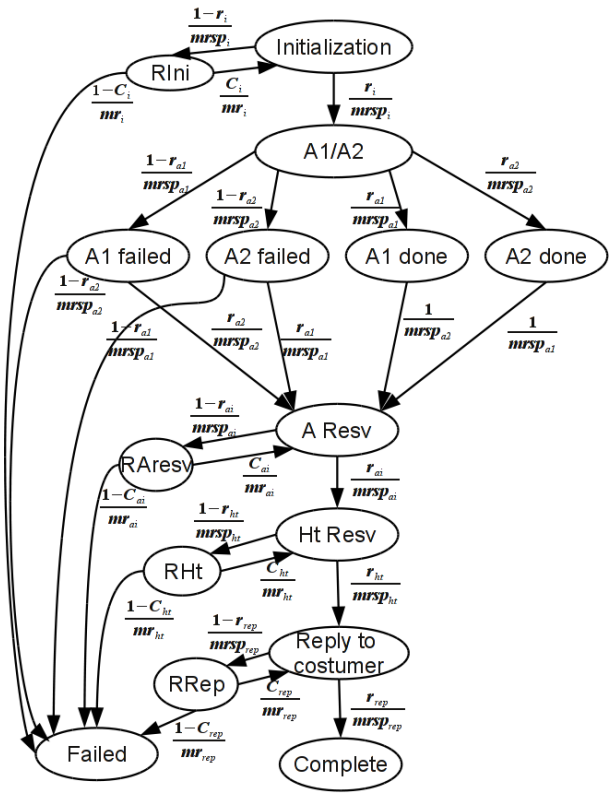

Figure 6.2: CTMC for the composite Web service with restarts

state, from where there is 2 possible next states: a) Initialization, if there was a covered failure, that could be solved with the restart; and b) Failed, if the failure was not covered by the restart, so the overall system fails. The other restart states have similar interpretation.

The parameters $mrsp_i$, $mrsp_{a1}$, $mrsp_{a2}$, $mrsp_{ai}$, $mrsp_{ht}$ and $mrsp_{rep}$ are the mean response time of the following web services, respectively: initialization, query on airline 1, query on airline 2, airline reservation (invocation), hotel reservation and reply to customer. The other parameters follow a similar notation. Notice that some transition rates are equal to the inverse of the mean response time of the Web service ($mrsp_x$), weighted by the probability of that transition occurs, i.e, the reliability of the respective Web service (r_x). In case of any web service x fails, the transition is equal to $\frac{1-r_x}{mrsp_x}$. The transition rates outgoing from restart states are equal to the inverse of mean restart time (mr_x), weighted by the coverage factor of failures for that Web service (C_x) in case of successful restart. If the restart of that web service is unsuccessful, the model transitions to the **Failed** state, with a rate $\frac{1-C_x}{mr_x}$.

Another model was also developed to represent the process without restarts. In the CTMC of Figure 6.3 any failure causes the process to abort. It is a pure reliability model, since repair actions are not represented, nor the response times of each Web service. The notation for

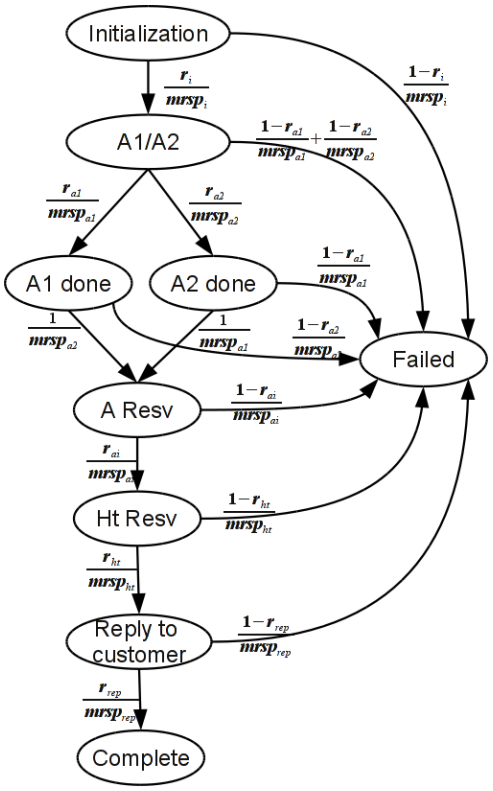

Figure 6.3: CTMC for the composite Web service without restarts

parameters in the transition rates is similar to the previous model.

6.1.2 Assignment of values and sensitivity scaling decision

The base values defined for all parameters of both models are shown in Table 6.1. They are the same ones defined in [Sato and Trivedi 2007], that in its turn obtained the values from experimental measurements. For the first model, one can see a mix of time values, defined as few seconds, reliability and coverage values, defined as probabilities in a $[0..1]$ range. In spite of differences in units, the orders of magnitude are close on to each other (the difference is less than 3 orders), so according to the proposed methodology, this justifies the use of unscaled sensitivities. For the second model, the situation is similar, just removing the parameters related to coverage and restart times, so unscaled sensitivities are also appropriate.

6.1.3 Results from sensitivity ranking analysis

The sensitivity analysis carried out was based on two measures, one for performance and the other for reliability. The performance measure was the probability of response time to be less

Table 6.1: Parameter values for Travel Agent models

Component	Reliab. (r)	Resp. time (mrsp)	Coverage (C)	Rest. time (mr)
Initialization ($init$)	1.0	1 s	1.0	0.15 s
Airline 1 ($a1$)	0.9	2 s	-	-
Airline 2 ($a2$)	0.9	2 s	-	-
Airline invocation (ai)	0.9	1 s	1.0	0.15 s
Hotel (ht)	0.9	2 s	0.0	0.15 s
Reply to customer (rep)	1.0	1 s	-	0.15 s

than or equal to time t: $Pr[Resp \leq t]$. The reliability measure was the probability of eventually reaching the state "**Complete**": $R = \pi_{Complete}$, i.e. the probability of success in a travel agent invocation. For the performance sensitivity analysis, the first model, with restarts, was used. For the reliability sensitivity analysis, the second model, without restarts, was used.

The model in Figure 6.2 was used to compute the probability of system being in state "**Complete**" by a given time t ($\pi_{Complete}(t)$). The value of $\pi_{Complete}(t)$ is equivalent to the probability of response time to be less than or equal to time t: $Pr[Resp \leq t]$, whose sensitivity is wanted. The time $t = 8.36s$, that is the mean response time for this system, was computed with SHARPE, and for this time, $\pi_{Complete}(t) = 0.558$.

Table 6.2 presents results for the sensitivity of $\pi_{Complete}(t)$ with respect to some parameters. It is assumed that in this analysis only strict performance parameters deserve attention, so reliability and coverage parameters are not included now. Notice that negative sensitivity values indicate an inverse relation between changes in the parameter and corresponding changes in the metric of interest, so when the response time of a given Web service decreases, $\pi_{Complete}(t)$ increases, since it is more likely the process has been completed by the time $t = 8.36s$.

The results show that $\pi_{Complete}(t)$ is more influenced by the response time of airline invocation than it is by other web services. This sensitivity ranking matches the results found in [Sato and Trivedi 2007], where scaled sensitivities were computed directly by the derivatives of closed-form solutions and validated by real experiments. In that paper, it was shown that hotel reservation (Ht) has a bigger impact on the overall response time than the airline 1 query ($A1$) and the airline 2 query do. Only those three parameters were considered in the comparison in [Sato and Trivedi 2007].

Since the sensitivity with respect to mr_{Init} and mr_{Rep} is null, these parameters can be removed from the model without affecting the accuracy of results. Although, if $\pi_{Complete}(t)$ is not

Table 6.2: Ranking of sensitivities for $\pi_{Complete}(t)$

Parameter θ	$\boldsymbol{S_\theta(\pi_{Complete}(t))}$
$mrsp_{Ai}$	-0.12546
$mrsp_{Rep}$	-0.11360
$mrsp_{Init}$	-0.11360
$mrsp_{Ht}$	-0.11204
$mrsp_{A1}$	-0.08035
$mrsp_{A2}$	-0.08035
mr_{Ainv}	-0.01262
mr_{Ht}	-0.01104
mr_{Init}	0.00000
mr_{Rep}	0.00000

the unique metric of interest for this model, the cited parameters should be kept.

It is possible to compare the differential sensitivity results with the approach of varying one parameter at a time, while holding the other ones fixed. Figure 6.4 shows a plot with the overall response time computed for a range of values of the response time parameters. The response time of each web service was changed from 1.0 second to 1.9 seconds. The lines corresponding to airline 1 and airline 2 are fully overlapped, as it was expected by looking in the respective sensitivities in Table 6.2. The same occurs for the Web services initialization and reply to customer. One can also notice that the slopes of lines corresponding to the response time of airline queries are the smallest ones among all the lines, confirming that changes on them have less contribution to changes in overall response time than other web services do.

If the values in Table 6.2 were scaled, or normalized, their new order would not match the order seen in the plot. For example, the impact of Ai would be considered lower than the impact of $A1$, what it is not true. Therefore, the plot is also useful to confirm the criterion of the proposed methodology, about using unscaled sensitivities in this case.

In spite of some similar conclusions taken from the plot and from the sensitivity ranking, the benefits of using the sensitivity functions are highlighted by looking at the lines of parameters Ht, Ai, $Init$ and $Reply$. It is not easy to distinguish between the impact of changes in hotel reservation and in airline invocation, for example. The use of numerical indices, as shown in the sensitivity ranking, is a more reliable way to identify what parameter deserve priority in its improvement, in order to have the highest improvement in the overall response time.

The sensitivity analysis of the reliability metric $R = \pi_{Complete}$ has produced another rank-

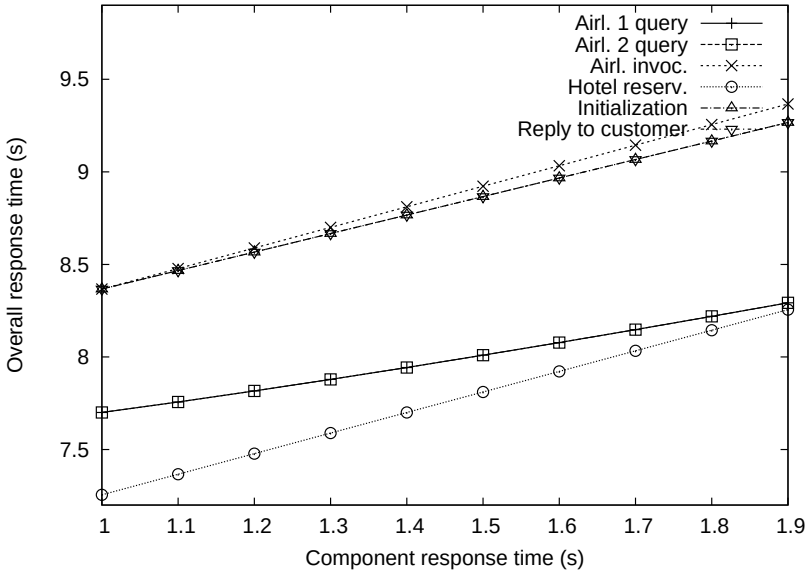

Figure 6.4: Plot of overall response time according to each component response time

ing, presented in Table 6.3. Is important to highlight that, since the model is acyclic, $\pi_{Complete}$ represent the probability of eventually reaching the state "**Complete**", in steady-state. The results show that the reliabilities of airline invocation and hotel reservation are the ones which have the biggest impact on R, while the sensitivities with respect to airline 1 and airline 2 queries (r_{A1}, r_{A2}) are the lowest ones. Such behavior is consistent with the redundancy provided by the concurrent queries to two airlines, that reduces the impact of a failure in one of the airline Web services. This also matches results in the cited paper [Sato and Trivedi 2007], which uses unscaled sensitivities too. Table 6.3 also shows that, for the adopted metric, the reliabilities of airline invocation and hotel reservation are equally important, what does not happen for the response times in the previous analysis.

Table 6.3: Ranking of sensitivities for R

Parameter θ	$\boldsymbol{S_\theta(R)}$
r_{Ai}	0.8099
r_{Ht}	0.8099
r_{Reply}	0.7290
r_{Init}	0.7290
r_{A1}	0.4049
r_{A2}	0.4049

In the same way as done for the performance analysis, the parameters values were also changed one at a time and plotted against R. This plot, in Figure 6.5 it is possible to confirm

that changes in r_{Ai} have the same impact as changes in r_{Ht}, since theirs lines are fully overlapped, and subsequently have equal slopes. The plot also confirms that $S_{r_{Reply}}(R) = S_{r_{Init}}(R)$, and $S_{r_{A1}}(R) = S_{r_{A2}}(R)$. But, again, the numerical ranking generated by the differential sensitivity analysis allows a faster decision about which parameters are able to provide the biggest improvement in the measure of interest.

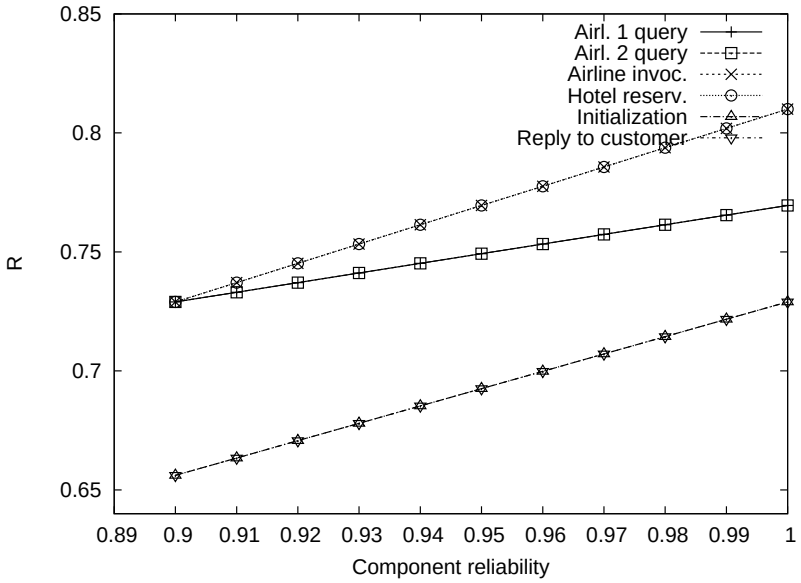

Figure 6.5: Plot of R vs. each component reliability

6.2 Redundant networks: Availability Analysis

This case study presents the availability analysis of a network, including the use of redundant components. Failure, recovery and reconfiguration events are considered, for three architectures with different levels of redundancy.

The first architecture represents a system without any redundancy, and it is illustrated by Figure 6.6. It is composed of two machines, a switch and two routers connected by a single link. The second architecture, shown in Figure 6.7, has aspects of fault-tolerance based on link redundancy. It is composed of two machines, a switch and two routers that are connected by redundant links (L0 and L1). When the main link (L0) fails, the spare link (L1) assumes the role of the main one. After main link restoration, the system returns to the initial condition.

The third scenario has router-level redundancy. It is composed of two machines, a switch, three routers, and two links, as seen in Figure 6.8. One link connects router R0 to router R2,

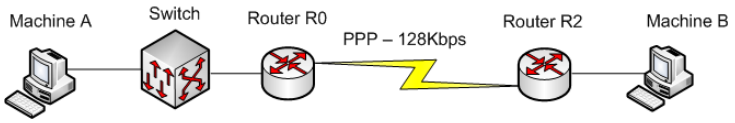

Figure 6.6: Network without redundancy

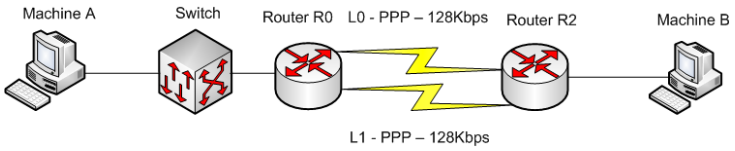

Figure 6.7: Network with link redundancy

whereas the other link connects routers R1 and R2. The system uses fault-tolerance based on warm-standby redundancy. When one of the primary components (R0 or L0) fails, the spare components (R1 and L1) assume the role of the primary components. This switchover process takes a time for the spare components to start operation, named Mean Time to Activate (MTTA). After restoration of the primary components, the system returns to the initial condition.

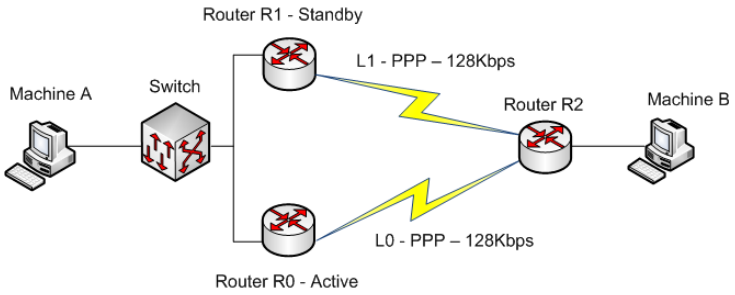

Figure 6.8: Network with router redundancy

6.2.1 Creation of models and definition of parameters for the network architectures

Three CTMC (Continuous Time Markov Chain) availability models were developed to analyze the architectures described. In Figure 6.9, the Markov chain represents the first scenario, which is the simplest one, with no redundancy. There is only one link, named L0, connecting router R0 and router R1. In this model, the normal operation of a component is denoted by the label U (up), and a failed component is represented by label D (down). A state in the Markov chain is defined by a sequence of labels, representing router R0, router R1 and link L0, respectively. The failure and repair time of each component are assumed to be exponentially

distributed. λ_{R0}, λ_{R1} and λ_{L0} are the respective failure rates of R0, R1 and L0. In a similar notation, μ_{L0}, μ_{R0} and μ_{R1} are the respective repair rates of each system component. Once any component (R0, R1, or L0) has failed, the overall system is in a down state and subsequently no additional failures occur until the component is repaired. In Table 6.4, a description of each state is given. For this model, the system is up and running only in the state UUU. All the other states are shaded gray in Figure 6.9, representing the system down states.

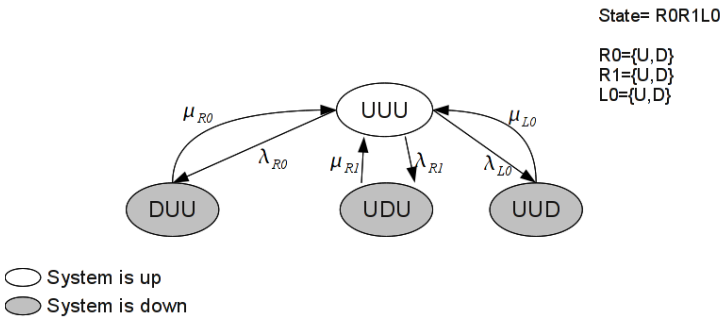

Figure 6.9: Markov chain for the availability of non-redundant network

Table 6.4: States of CTMC Model with link redundancy

State	Description
UUU	The System is UP
DUU	Down state, Router R0 failed
UDU	Down state, Router R1 failed
UUD	Down state, Link L0 failed

Figure 6.10 shows the Markov chain for the system with redundancy only at the link level. The notation is similar to the previous model. A state in the Markov chain is also defined by a sequence of labels, representing router R0, router R1, link L0, and link L1, respectively. The ideal condition for this system is denoted by state UUUU, in which all components are in non-failed condition. In states shaded gray, the system has failed, due to a failure in one of the routers, or a failure in both links. In those states, it is assumed that no additional failures can occur, since the remaining components are in an idle condition. Another assumption for this model is that there is a repair policy, that prioritizes the repair of link L0 over link L1, when both are failed. There is no priority in the repair of routers because it is not possible in this model to have both routers down. In Table 6.5, a description for each state is given.

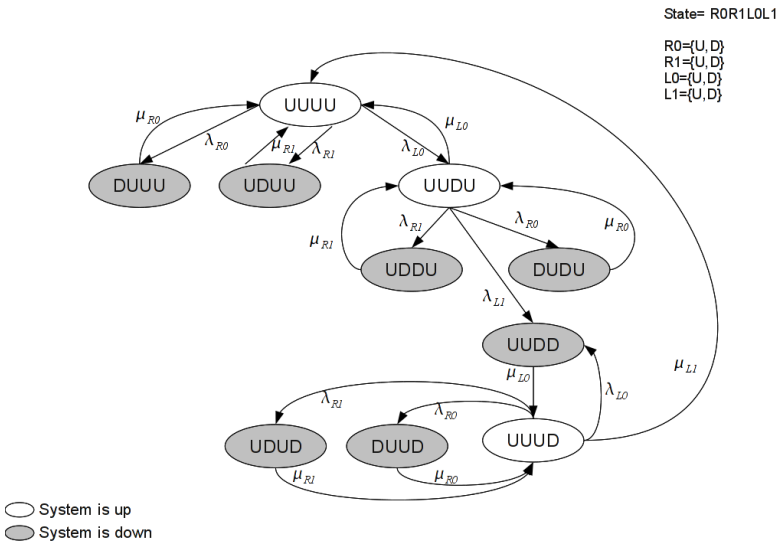

Figure 6.10: Markov chain for the availability of link-redundant network

Table 6.5: States of CTMC Model with link redundancy

State	Description
UUUU	The System is UP
DUUU	Down state, Router R0 failed
UDUU	Down state, Router R1 failed
UUDU	The System is UP, Link L0 failed
UDDU	Down state, R1 and L0 failed
DUDU	Down state, R0 and L0 failed
UUDD	Down state, L0 and L1 failed
UUUD	The System is UP, L1 failed
DUUD	Down state, R0 and L1 failed
UDUD	Down state, R1 and L1 failed

In Figure 6.11, a Markov chain represents the system illustrated in Figure 6.8. The failure and repair rates of each component are represented by λ_X and μ_X, where $X \in \{R0, R1, R2, L0, L1\}$. Rates α_R and α_L are the inverse of mean time to activate the spare router and the spare link, respectively.

Some simplifications were also made here, but they do not significantly affect the results obtained from the analysis. One of the assumptions is that there is a priority in the repair of components. Router R2 has the higher priority, followed by router R0, link L0, router R1, and link L1, in descending order. Consider also that no failure is possible when a component is in waiting condition.

As in the other two models, the nomenclature of states is based on the current condition of

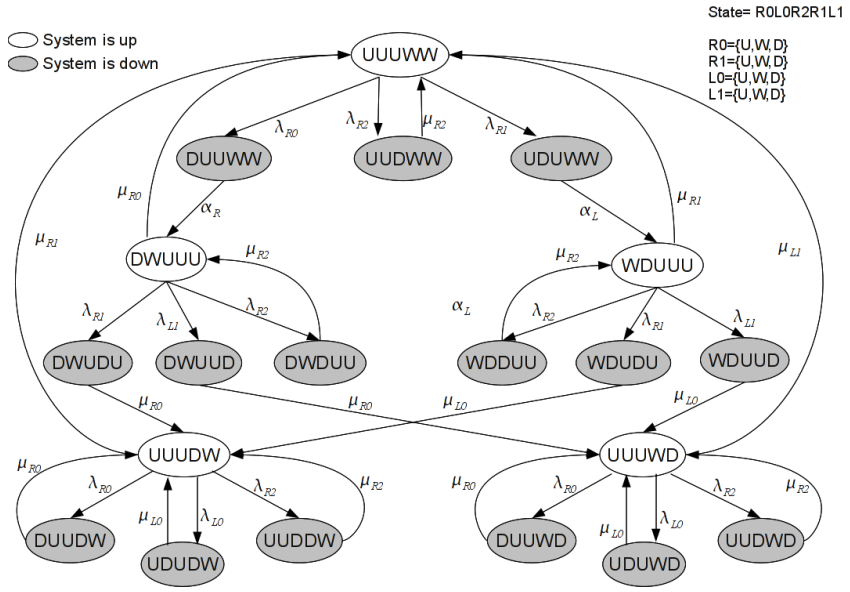

Figure 6.11: Markov chain for the availability of router-redundant network

each system component, in the following order: router R0, link L0, router R2, router R1, link L1. A letter U indicates the up condition, when component is active. Letter D denotes a down condition for that component, meaning that it has failed, and that a repair is needed. Letter W represents a waiting condition, in which the component is not being used, but is ready to enter in active mode, as soon as it is needed. Therefore, state UUUWW denotes router R0, link L0, and router R2 are active, router R1 and link L1 are on waiting condition.

For an active state, the system must present one of the following combinations: UUUWW, DWUUU, WDUUU, UUUDW or UUUWD (see Figure 6.11). Particularly, in the DWUUU state, router R2 with spare router and link (R1 and L1) are active, while R0 is in down state and L0 is waiting, since it only works together with R0. A similar situation happens in WDUUU state, where R2, R1 and L1 are active, but L0 is down, leaving R0 in a waiting condition. Table 6.6 lists the system availability condition for each state of this Markov chain.

6.2.2 Assignment of values and sensitivity scaling decision

The MTTFs of components used in the three architectures are respectively: 131,000 hours for routers and 11,988 hours for links. The mean time to repair (MTTR) is equal to 12 hours, for all components. The mean time to activate the spare components (MTTA) is equal to 10 seconds, or 0.0027 hours. Those values shall be considered as the base case throughout this section, unless another value is specified in each specific analysis. Notice that all λ_i in the

Table 6.6: States of CTMC Model with router redundancy

State	Description
UUUWW	The System is UP
DUUWW	Down state, Switchover Started
UDUWW	Down state, Switchover Started
UUDWW	The System is Down
DWUUU	The System is UP
WDUUU	The System is UP
DWUDU	The System is Down
DWUUD	The System is Down
DWDUU	The System is Down
WDDUU	The System is Down
WDUDU	The System is Down
WDUUD	The System is Down
UUUDW	The System is UP
UUUWD	The System is UP
DUUDW	The System is Down
UDUDW	The System is Down
UUDDW	The System is Down
DUUWD	The System is Down
UDUWD	The System is Down
UUDWD	The System is Down

models are equal to $1/MTTF_i$, all μ_i are equal to $1/MTTR_i$, and all α_i are equal to $1/MTTA_i$. In fact, the model was parameterized in function of the time parameters, instead of the rates. The rates are shown in the figures only due to their compact form, that ease the understanding of the model.

The values assigned to the parameters have big differences in their magnitudes. The ratio between router MTTF and router MTTR is about 10,000, for instance. There is a similar ratio between a MTTR parameter and a MTTA parameter. So, according to the proposed methodology, scaled sensitivities should be used to elaborate the sensitivity ranking in this case.

6.2.3 Results from sensitivity ranking analysis

First, the results of parametric sensitivity analysis for the architecture modeled by Figure 6.11 will be shown. Sensitivity analysis of steady-state availability is carried out here by computing $SS_{MTTF_i}(A)$ as the scaled sensitivity of availability with respect to $MTTF_i$, and $SS_{MTTR_i}(A)$ as the corresponding measure with respect to $MTTR_i$. In the base case, using the values previously mentioned, the steady-state availability is 0.999906968.

Table 6.7: Sensitivity of Availability for scenario with router redundancy

Parameter k	$\boldsymbol{SS_k(A)}$
$MTTF_{R2}$	9.159×10^{-5}
$MTTR_{R2}$	-9.159×10^{-5}
$MTTR_{L0}$	-2.182×10^{-6}
$MTTF_{L0}$	1.317×10^{-6}
$MTTF_{L1}$	1.091×10^{-6}
$MTTR_{R0}$	-1.997×10^{-7}
$MTTF_{R0}$	1.205×10^{-7}
$MTTF_{R1}$	9.985×10^{-8}
$MTTR_{L1}$	-1.190×10^{-9}
$MTTR_{R1}$	-1.089×10^{-10}

The sensitivity analysis results are shown in Table 6.7. Parameters $MTTF_{R2}$ and $MTTR_{R2}$ assume the greatest importances in system steady-state availability, since they have the highest sensitivity values. Any change in these parameters will have a major impact on system availability, but in opposite directions. Sensitivity with respect to $MTTF_{R2}$ is positive, since the availability increases when this parameter increases. In contrast, $SS_{MTTR_{R2}}(A)$ is negative, because a smaller repair time of R2 implies on an increased availability. In Table 6.7, one can also notice that time to repair spare components ($MTTR_{R1}$ and $MTTR_{L1}$) have the smallest impact on the system availability. This result matches the established repair policy, since failed spare components are repaired only after main components have returned to normal operation.

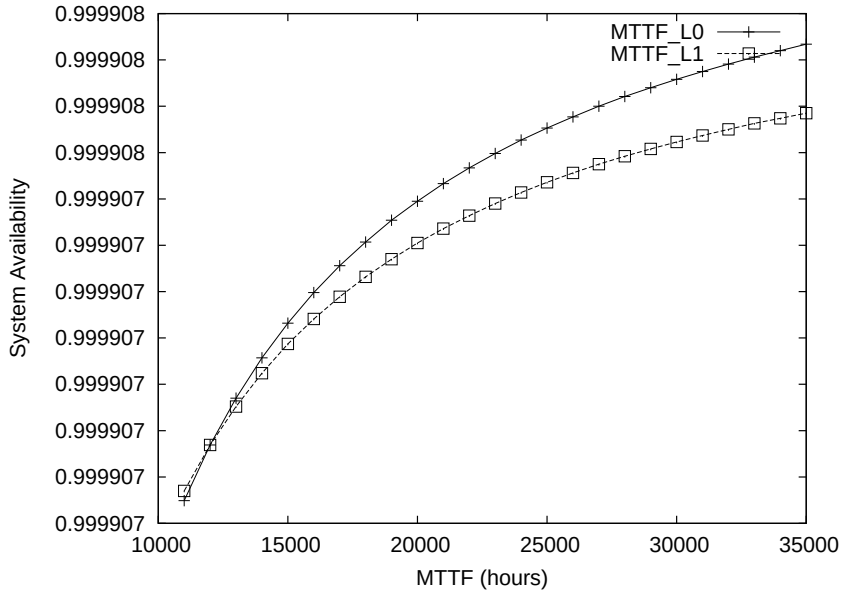

Figure 6.12: Effect of each link MTTF on system availability (3rd scenario)

Figure 6.12 confirms that efforts in expanding the time to failure of link L0 have more impact on system availability than increases in $MTTF_{L1}$ do. Figure 6.13 also validates the results from sensitivity ranking. If $MTTF_{R2}$ is increased, the benefits will be much higher than those resulting from enhancements on either R0 or R1 MTTFs. Although, the difference between the impact of changes in $MTTF_{R0}$ and $MTTF_{R1}$ is not visible in the plot, due to the large range of changes in the availability caused by $MTTF_{R2}$. Another plot with only those two parameters would be necessary to notice this difference, while the sensitivity ranking already allows to state clearly the order of importance between them.

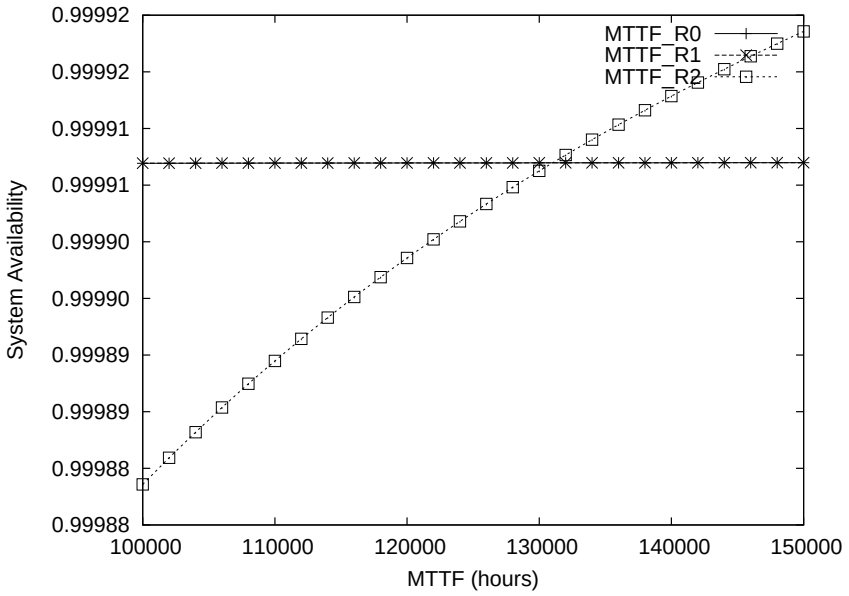

Figure 6.13: Effect of each router MTTF on system availability (3rd scenario)

Figure 6.14 shows the system availability as a function of each component MTTR. Router R2 is the component whose time to repair causes the biggest effect on the steady-state availability, followed by link L0. This information is the same as one obtained comparing the corresponding parameters in Table 6.7, which also highlights other differences in parameter importance not seen in the plot. Another benefit of differential sensitivity analysis here is the ability to compare repair and failure parameters, what can not be made only with plots.

The next step in this analysis is to measure the impact of R0 and L0 MTTFs on the system availability in different redundancy schemes (i.e., different architectures). These parameters were selected because they are the only ones that exist in all three architectures.

Comparing the results presented in Tables 6.9 and 6.8, an expected behavior is noticed: the link redundancy mechanism makes the system availability less sensitive to failures of primary

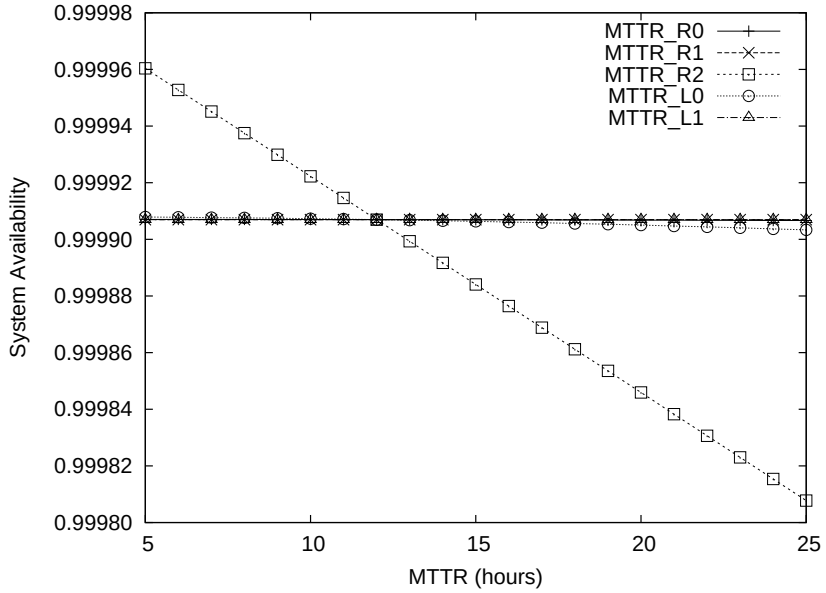

Figure 6.14: Effect of each component MTTR on system steady-state availability (3rd scenario)

link (L0), while the sensitivity of availability with respect to $MTTF_{R0}$ is almost the same in both scenarios (with no redundancy and with link redundancy). Comparing the results presented in Tables 6.8 and 6.7, one sees that availability in the third scenario is less affected by increases in $MTTF_{L0}$ and in $MTTF_{R0}$ than it is affected in the second scenario. This kind of comparison can be done even if there is a larger number of parameters in the model, or a large number of models to be compared. This constitutes another advantage in using the proposed methodology.

Table 6.8: Sensitivity of Availability for scenario with link redundancy

Parameter k	$\boldsymbol{SS_k(A)}$
$MTTF_{R0}$	9.158×10^{-5}
$MTTF_{R1}$	9.158×10^{-5}
$MTTR_{R0}$	-9.158×10^{-5}
$MTTR_{R1}$	-9.158×10^{-5}
$MTTF_{L0}$	1.000×10^{-6}
$MTTR_{L0}$	-1.999×10^{-6}
$MTTF_{L1}$	9.998×10^{-7}
$MTTR_{L1}$	-9.998×10^{-10}

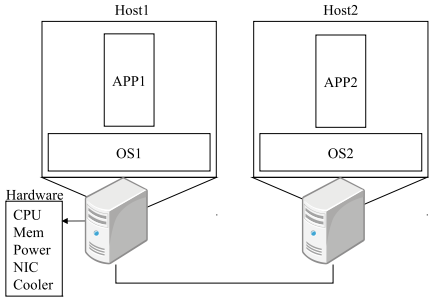

Figure 6.15: Architectures of non-virtualized two hosts system

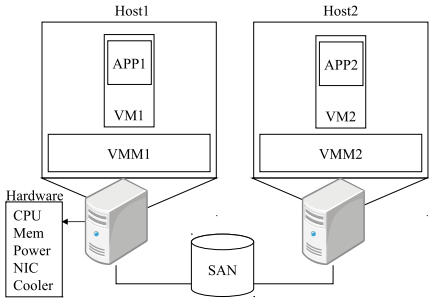

Figure 6.16: Architecture of virtualized two hosts system

6.3 Virtualized Servers: Availability Analysis

As a third case study, it is presented a virtualized system for an application hosting provider. It is composed of two hosts, in which each host has one virtual machine (VM) running on a virtual machine monitor (VMM). This system was first analyzed in [Kim et al. 2009], but in that paper no formal sensitivity analysis is carried out. Figures 6.15 and 6.16 show all hardware and software parts of the system, with and without virtualization, respectively. Figure 6.15 illustrates a non-virtualized system, in which the operating system of each server directly uses the underlying hardware. In Figure 6.16 the virtualized version is represented, in which one VMM is running in each host. The VMMs are responsible for providing the access to hardware

Table 6.9: Sensitivity of Availability for scenario with no redundancy

Parameter k	$\boldsymbol{SS_k(A)}$
$MTTF_{L0}$	9.998×10^{-4}
$MTTR_{L0}$	-9.998×10^{-4}
$MTTF_{R0}$	9.149×10^{-5}
$MTTR_{R0}$	-9.149×10^{-5}
$MTTF_{R1}$	9.149×10^{-5}
$MTTR_{R1}$	-9.149×10^{-5}

resources for the guest operating systems that may be running on top of the VMMs. The following hardware components were modeled in each host: CPU, memory, power, network card and cooling subsystem. Both hosts share a common SAN (storage area network), that helps to support VM live migration. Using this configuration, in case of some failures at VM or host level, the VM may be migrated to the other host, keeping the access to the data that it was handling before the failure. This mechanism reduces the downtime and helps in keeping data consistency. The same kind of application is running in Host1 and Host2. The application App1 is the instance originally running in Host1/VM1, and similarly, App2 is the application instance originally running in Host2/VM2. This is called an active/active configuration in a virtualized system [Loveland et al. 2008].

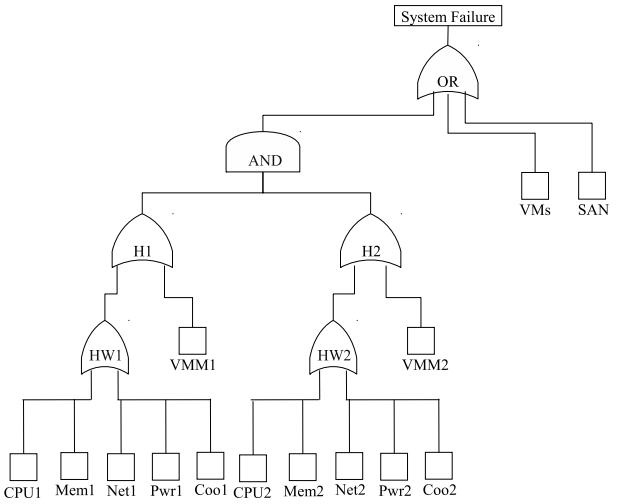

Figure 6.17: Virtualized system availability model

Figure 6.17 is a top level fault tree model for the system illustrated by Figure 6.16. At the leaf nodes of this fault tree are the components whose availabilities are computed using underlying Markov chains. All the Markov sub-models are described in more detail in [Kim et al. 2009]. For further information about dependability analysis using fault tree, please see [Sahner et al. 1996].

6.3.1 Creation of model for VMs subsystem availability

This case study performs a parametric sensitivity analysis of VMs subsystem availability, represented by the Markov model shown in Figure 6.18. The other models for the fault tree's leaf nodes are not considered here, allowing a more focused analysis. The nomenclature for

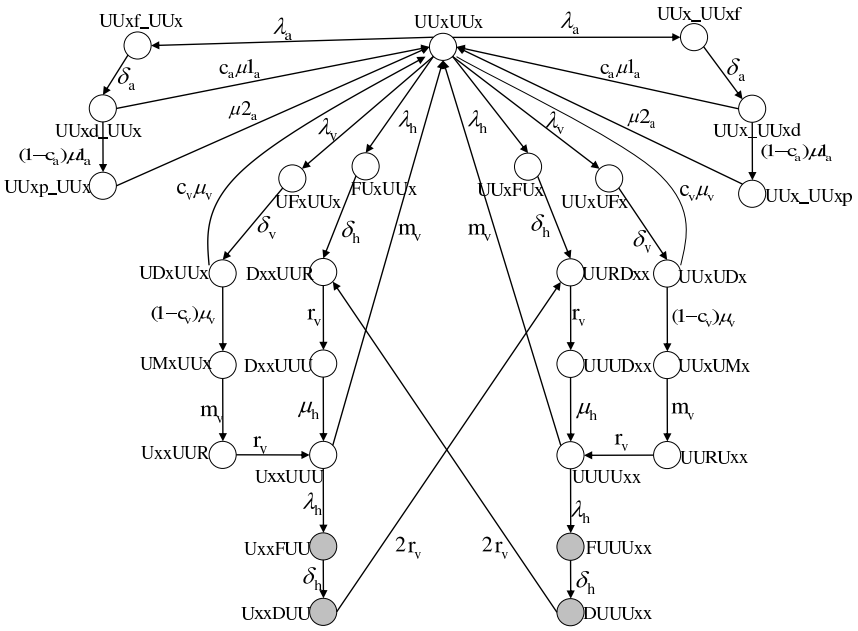

Figure 6.18: VMs Availability Model

states was based on the current condition of each component: VM, Host and Application. For example, state UUxUUx represents, from the left to right, host1 is up, VM1 is up, host1 has capacity to run another VM (in short, x), host2 is up, VM2 is up, host2 has capacity to run another VM.

The meaning of some of the states is explained in Table 6.10. The other states are easily understood, as they have similar meanings of those in Table 6.10, but with the host number changed, e.g., state FUUUxx is the opposite of state UxxFUU, so in state FUUUxx the host H2 is up, H1 failed when VM1 and VM2 were running on it. In the shaded states of Figure 6.18, the system is down because both applications are not working properly.

6.3.2 Definition of parameters and assignment of values

After model creation, the next step in the methodology proposes that the parameters must be defined and their values shall be assigned. Table 6.11 shows the input parameters, with their descriptions and values. Some of these values were obtained from experimental studies and other ones were “estimated guesses”, since some parameters values are not available, or are confidential. Note that in Table 6.11, rate parameters, such as λ_h, are presented as mean time parameters, i.e. the inverse of the rates, because their values may be better shown and interpreted in this way.

Table 6.10: Nomenclature of states

State	Description
UUxUUx	VM1 running on H1, VM2 running on H2
UUxf_UUx	App1 failed, both VMs and Hosts are up
UUxd_UUx	App1 failure is detected
UUxp_UUx	App1 failure is not covered. Additional recovery step is started.
UFxUUx	H1 up, VM1 failed, VM2 running on H2
UDxUUx	VM1 failure is detected
UMxUUx	VM1 to be moved to Host2
UxxUUR	VM1 is restarting on H2
UxxUUU	VM1 and VM2 are running on H2
FUxUUx	H1 failed, VM2 running on H2
DxxUUR	H1 failure is detected, VM1 is restarted on H2
DxxUUU	H1 is down, VM1 and VM2 running on H2
UxxFUU	H1 up, H2 failed when VM1 and VM2 were running on it
UxxDUU	H2 failure is detected and two VMs are on H2

6.3.3 Sensitivity scaling decision

The sensitivity analysis of the system represented in Figure 6.18 was based on the following measures: Equivalent Mean Time to Failure of VMs subsystem ($MTTF_{VMs}$), and Capacity Oriented Availability [Heimman et al. 1990] of VMs subsystem (COA_{VMs}). The first measure ($MTTF_{VMs}$) is computed by the Equation 6.1, where $T_{failure}$ represents the throughput of failure transitions, computed by Equation 6.2, and UP denotes the set of up states in the VM availability model.

$$MTTF_{VMs} = \frac{\sum_{i \in UP} \pi_i}{T_{failure}} \tag{6.1}$$

$$T_{failure} = (\pi_{UxxUUU} \cdot \lambda_h) + (\pi_{UUUUxx} \cdot \lambda_h) \tag{6.2}$$

In order to find the sensitivity of $MTTF_{VMs}$ to parameter λ_h, one needs to compute the derivative of Equation 6.1 with respect to λ_h. This partial derivative is expressed by Equations 6.3 and 6.4.

Table 6.11: Input parameters for VMs model

Parameter	Description	Value
$1/\lambda_h$	Mean time to Host failure	2654 hr
$1/\lambda_v$	Mean time to VM failure	2880 hr
$1/\lambda_a$	Mean time to Application failure	336 hr
$1/\delta_h$	Mean time to Host failure detection	30 sec
$1/\delta_v$	Mean time to VM failure detection	30 sec
$1/\delta_a$	Mean time to App failure detection	30 sec
$1/m_v$	Mean time to migrate a VM	5 min
$1/r_v$	Mean time to restart a VM	5 min
$1/\mu_v$	Mean time to repair a VM	30 min
$1/\mu 1_a$	Mean time to App first repair (covered case)	20 min
$1/\mu 2_a$	Mean time to App second repair (not covered case)	1 hr
c_v	Coverage factor for VM repair	0.95
c_a	Coverage factor for Application repair	0.9

$$S_{\lambda_h}(MTTF_{VMs}) = \frac{\sum_{i \in UP} \frac{\partial \pi_i}{\partial \lambda_h} * T_{failure} - \sum_{i \in UP} \pi_i * \frac{\partial T_{failure}}{\partial \lambda_h}}{T_{failure}^2} \tag{6.3}$$

$$\frac{\partial T_{failure}}{\partial \lambda_h} = \left(\frac{\partial \pi_{UxxUUU}}{\partial \lambda_h} \cdot \lambda_h \right) + \pi_{UxxUUU} + \left(\frac{\partial \pi_{UUUUxx}}{\partial \lambda_h} \cdot \lambda_h \right) + \pi_{UUUUxx} \tag{6.4}$$

The sensitivities of this measure to other parameters were also computed in a similar way, so their respective equations are not shown here.

The measure COA_{VMs} denotes how much service the VMs subsystem actually delivers, assuming that the available amount of resources depends on the number of VM(s) on a host server. This metric is computed by assigning reward rates, r_i, to each state i of the VMs availability model, so it becomes as Markov reward model. Reward rate 1 is assigned to states where one VM is running on each host (e.g., UUxUUx); reward rate 0.75 is assigned to states where two VMs are running on a single host (e.g., UUUUxx, UUUDxx); reward rate 0.5 is assigned to states where only one VM is running (e.g., UUxFxx, UUxDxx); zero reward rate is assigned to all the other states. Thus, COA_{VMs} is the expected steady-state reward rate of the VMs Markov chain, and it is described in Equation 6.5. Since in this case the reward rates do not depend on

any system parameter, $S_{\lambda_h}(COA_{VMs})$ was computed as shown in Equation 6.6. The sensitivity of COA_{VMs} with respect to other parameters was computed in a similar way.

$$COA_{VMs} = \sum_i r_i \pi_i \tag{6.5}$$

$$S_{\lambda_h}(COA_{VMs}) = \frac{\partial COA_{VMs}}{\partial \lambda_h} = \sum_i r_i \frac{\partial \pi_i}{\partial \lambda_h} \tag{6.6}$$

Using the values listed in Table 6.11, $MTTF_{VMs}$ and COA_{VMs} were computed, as well as their scaled sensitivity indices. Scaled sensitivities were used because the ratio between parameter values has more than 4 orders of magnitude (e.g., the mean time to a failure of a host is measured in thousands of hours, whereas a failure detection takes a few seconds). Besides, both measures $MTTF_{VMs}$ and COA_{VMs} have characteristics that justify the use of scaled sensitivities, according to the concepts described in Section 4.3.

6.3.4 Sensitivity analysis results

The scaled sensitivities of the mean time to failure of VMs subsystem, $SS_\theta(MTTF_{VMs})$, with respect to six parameters are shown in Table 6.12. The remaining model parameters were excluded from Table 6.12 due to the sensitivity with respect to them is very small, and in order to simplify the presentation of results. It is important to highlight that parameters are ordered according to absolute values of scaled sensitivities. A negative sensitivity indicates only that if the parameter value increases, the measure of interest decreases. A positive sensitivity indicates that an increase in the parameter value causes an increase in the measure of interest.

Table 6.12: Ranking of sensitivities for $MTTF_{VMs}$

Parameter θ	$\boldsymbol{SS_\theta(MTTF_{VMs})}$
λ_h	-1.95624
m_v	0.99993
λ_v	-0.04370
λ_a	0.00262
μ_v	-0.00035
r_v	-0.00007

This decreasing ranking of sensitivities shows that the host failure rate, λ_h, is the most

important among all parameters, when $MTTF_{VMs}$ is the measure of interest. Table 6.12 also shows that the migration rate, m_v, is the most important among the "recovery parameters" (m_v, μ_v, r_v). Both conclusions can be compared to the observation of plots in Figures 6.19 and 6.20, in spite of they present the mean times associated to each rate of Table 6.12. Failure and recovery parameters were separated due to scale differences, since the unit for failure events is hours, while that for recovery actions is in minutes. In both plots, all parameters were fixed to their base value, expressed in Table 6.11, and only one parameter was changed at a time, checking the corresponding change in $MTTF_{VMs}$.

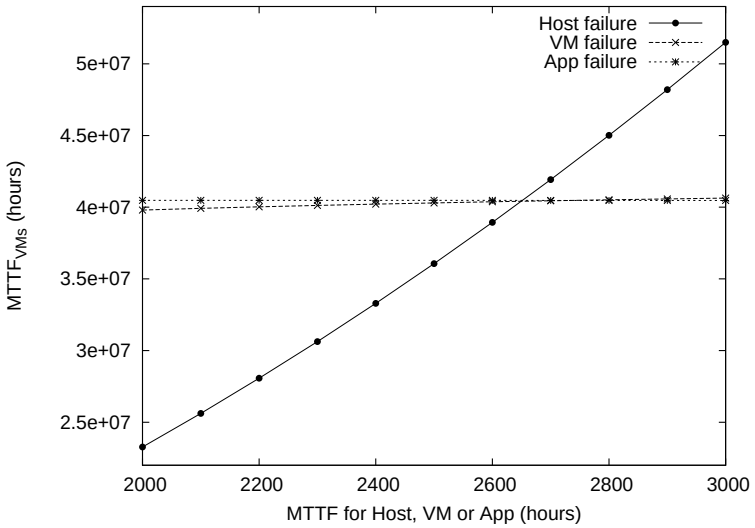

Figure 6.19: Effect of each failure parameter on $MTTF_{VMs}$

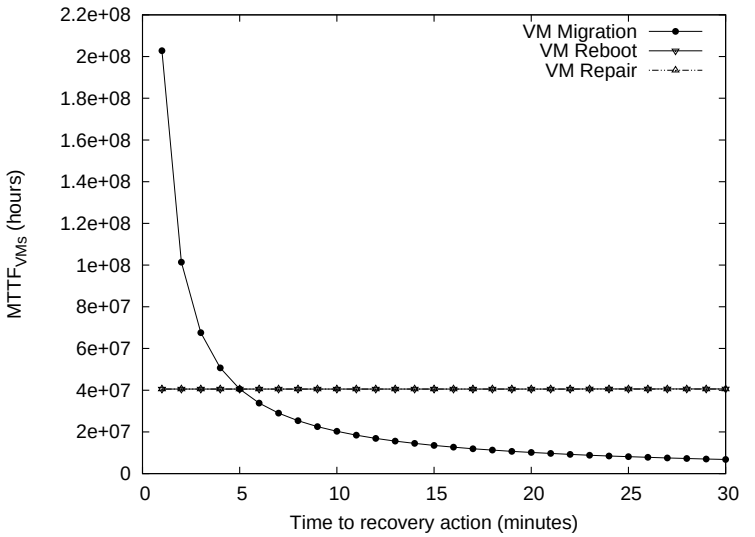

Figure 6.20: Effect of each recovery parameter on $MTTF_{VMs}$

Notice that the slopes of lines in Figure 6.19 match the order of parameters in Table 6.12,

Table 6.13: Ranking of sensitivities for COA_{VMs}

Parameter θ	$\boldsymbol{S_\theta(COA_{VMs})}$
λ_a	-0.001308
λ_v	-0.000179
μ_v	0.000173
λ_h	-0.000120
r_v	0.000033
m_v	0.000018

confirming the ranking of sensitivities for $MTTF_{VMs}$. The slope of the line corresponding to host failure is higher than the slope of VM failure. In its turn, the line of VM failure is a bit more inclined than the line of Application failure. This order is the same seen in Table 6.12. Similarly, the slopes of lines in Figure 6.20 match the order of recovery parameters presented in Table 6.12.

For capacity oriented availability (COA_{VMs}), the ranking of sensitivities is presented in Table 6.13. This ranking is very different from that one in Table 6.12. Notice that for COA, the most important parameter is the failure rate of the application. This is an important conclusion since it shows that software failure is a major concern for the COA whereas host failure rate (λ_h) is most important for steady-state availability ($MTTF_{VMs}$).

Figures 6.21 and 6.22 confirm these results. The COA decreases faster when application failure rate, λ_a, increases than when host or VM failure grows. A line representing the "Current COA" was added to Figure 6.22, in order to ease the comparison of their results with the sensitivity ranking. The points where this line intersects the parameter lines correspond to the values of each parameter that was used in the differential sensitivity analysis. Therefore, by looking at the points ($\mu_v = 2, m_v = 12, r_v = 12$), Figure 6.22 shows that the increase in VM repair rate, μ_v, has more influence on COA_{VMs} than VM migration or VM reboot rates have, in the current scenario. Such a kind of graphical interpretation becomes difficult, and even unfeasible if the amount of parameter is large. It is important to highlight that this is other example where the differential sensitivity analysis provides a faster way to identify the parameter which deserves priority in its improvement.

Section 4.1 mentions the possibility of sensitivity reduction after iterations and subsequent improvements on some parameters. Table 6.14 shows how much the sensitivity of COA may

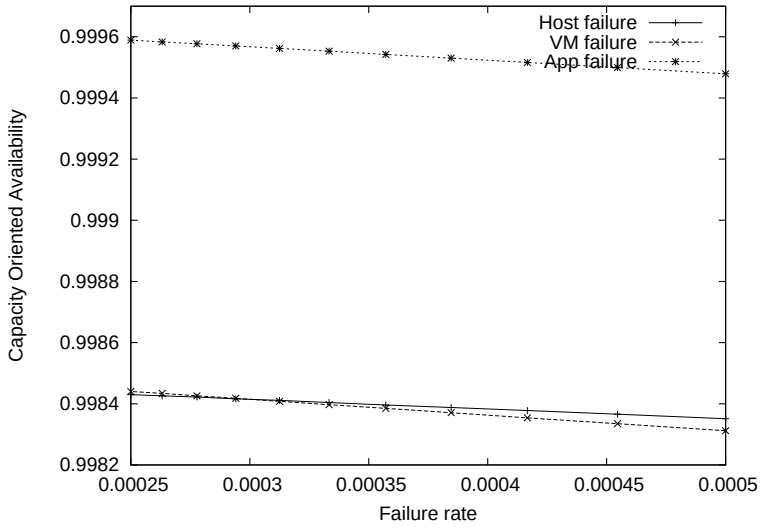

Figure 6.21: Effect of each failure parameter on COA_{VMs}

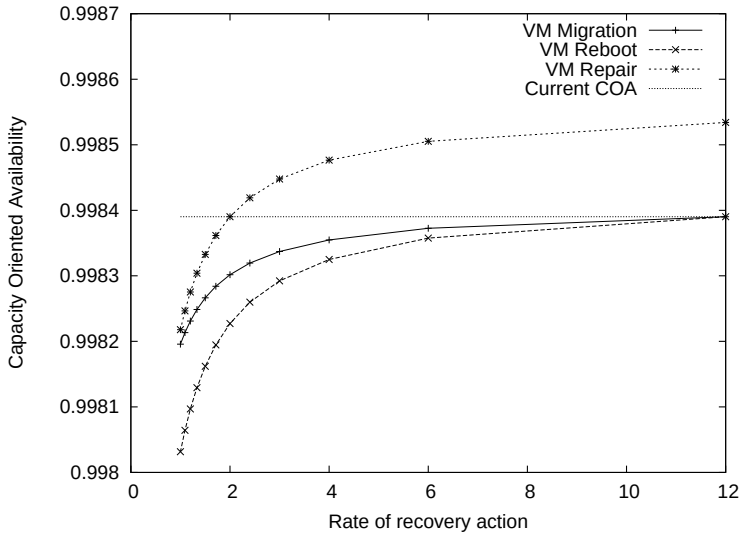

Figure 6.22: Effect of each recovery parameter on COA_{VMs}

change depending on the host failure level. In order to analyze this, the sensitivity of COA was computed using 6 different MTTFs for host, from 1440 hours (2 months) until 8640 hours (12 months), with intervals of 2 months for each time. The sensitivity with respect to λ_a, λ_v, and μ_v does not have significant changes, with the increase of host MTTF. Although, for migration (m_v) and reboot (r_v) rates, the COA's sensitivity is bigger when the time to host failure is small. As host (hardware and VMM) becomes more reliable, COA becomes less dependent on the time for migration or reboot.

The order of parameters is the same for all host MTTFs. This table shows that, in this system, changes in hardware reliability do not affect the importance order of those parameters

Table 6.14: Ranking of scaled sensitivities for COA_{VMs}, varying $MTTF_h = 1/\lambda_h$

$\boldsymbol{MTTF_h}$	$\lvert\boldsymbol{SS_{\lambda_a}}\rvert$	$\lvert\boldsymbol{SS_{\lambda_v}}\rvert$	$\lvert\boldsymbol{SS_{\mu_v}}\rvert$	$\lvert\boldsymbol{SS_{r_v}}\rvert$	$\lvert\boldsymbol{SS_{m_v}}\rvert$
1440 hrs	-1.307×10^{-3}	-1.791×10^{-4}	1.726×10^{-4}	5.900×10^{-5}	3.085×10^{-5}
2880 hrs	-1.307×10^{-3}	-1.792×10^{-4}	1.727×10^{-4}	3.023×10^{-5}	1.651×10^{-5}
4320 hrs	-1.308×10^{-3}	-1.792×10^{-4}	1.727×10^{-4}	2.063×10^{-5}	1.172×10^{-5}
5760 hrs	-1.308×10^{-3}	-1.792×10^{-4}	1.727×10^{-4}	1.584×10^{-5}	9.336×10^{-6}
7200 hrs	-1.308×10^{-3}	-1.792×10^{-4}	1.728×10^{-4}	1.296×10^{-5}	7.900×10^{-6}
8640 hrs	-1.308×10^{-3}	-1.792×10^{-4}	1.728×10^{-4}	1.104×10^{-5}	6.943×10^{-6}

to capacity oriented availability.

6.3.5 Sensitivity analysis with cost information

Here is presented an example of how to use the adapted version of the proposed methodology for cost-constrained optimization, considering the virtualized system that was previously described and analyzed. Only the steps related to "Definition of cost function", "Constrained optimization method", and "Sensitivity ranking analysis" will be presented, because there is no changes in the other steps, already shown in the previous analysis.

The improvement of parameters involved in the virtualized system availability have financial costs, which may be estimated by step-wise or continuous functions. The derivative of continuous functions is computed in a easier manner, so they should be preferred when using optimization methods such as Lagrange multipliers [Avriel 2003].

The cost of improvement of a failure rate may be thought as inversely proportional to that rate, so to get a lower failure rate, a bigger financial investment is needed. For repair, reboot and migration rates, the relation is directly proportional, since to obtain higher repair rates (lower time to repair) involve higher costs.

The costs of improvements in failure, repair, reboot and migration rates are summarized by the Equation 6.7. C_{λ_h}, C_{λ_v} and C_{λ_a} are the costs of improvements in host, VM, and application failure rates, respectively. C_{μ_h} ,C_{μ_v}, $C_{\mu 1_a}$ and $C_{\mu 2_a}$ are the costs of improvements in host, VM, application 1 and application 2 repair rates, respectively. C_{m_v} is the cost of improvements in the VM migration rate, while C_{r_v} is the cost related to the improvement of VM reboot rate. The approximated cost functions for each component of Equation 6.7 are detailed further.

$$C_{improve} = C_{\lambda_h} + C_{\lambda_v} + C_{\lambda_a} + C_{\mu_h} + C_{\mu_v} + C_{\mu 1_a} + C_{\mu 2_a} + C_{m_v} + C_{r_v} \tag{6.7}$$

Host failure rate The failure rate of a host server depends on the reliability of the devices and their configurations in the host server. The reliability of the host server can be improved by installing more reliable devices and dedicated tuning. The cost related to procure more reliable devices and perform a tuning is related to the failure rate of host server. It is assumed that the following cost function approximates the relationship between the required costs and the failure rate of a host server:

$$C_{\lambda_h} = \frac{K_{\lambda_h}}{\lambda_h^{\alpha_{\lambda_h}}} + K_{\lambda_{h0}} \tag{6.8}$$

where K_{λ_h}, $K_{\lambda_{h0}}$ and α_{λ_h} are constants. $K_{\lambda_{h0}}$ is the fixed part of cost, independent of the rate λ_h. In other words, $K_{\lambda_{h0}}$ is the minimum cost of improvements in the failure rate of a host. α_{λ_h} is an exponent of λ_h, and K_{λ_h} represents a proportional cost to $1/\lambda_h^{\alpha_{\lambda_h}}$, i.e., the unitary cost per decrement in the host failure rate.

VM and Application failure rate The reliability of software component can be improved by conducting software test and debug. However, such test and debug process requires manual operations and related laboring costs. The amount of costs invested in software test directly affects the failure rate of software component. The costs related to the failure rates of VM and application are approximated with the following cost functions:

$$C_{\lambda_v} = \frac{K_{\lambda_v}}{\lambda_v^{\alpha_{\lambda_v}}} + K_{\lambda_{v0}} \tag{6.9}$$

$$C_{\lambda_a} = \frac{K_{\lambda_a}}{\lambda_a^{\alpha_{\lambda_a}}} + K_{\lambda_{a0}} \tag{6.10}$$

where K_{λ_v}, $K_{\lambda_{v0}}$, α_{λ_v}, K_{λ_a}, $K_{\lambda_{a0}}$ and α_{λ_a} are constants. $K_{\lambda_{v0}}$ and $K_{\lambda_{a0}}$ are the minimum cost of improvements in the VM and application failure rates, respectively. α_{λ_v} and α_{λ_a} are exponents for λ_v and λ_a, respectively. K_{λ_v} and K_{λ_a} represent the proportional costs to $1/\lambda_v^{\alpha_{\lambda_v}}$ and $1/\lambda_a^{\alpha_{\lambda_a}}$, i.e., the unitary cost per decrement in VM and application failure rates.

Host, VM and Application repair rates Similarly, the recovery rates for host servers and software components can be improved by investing in organizing system administrations. The more investment in administration group enriches their administrative operations and hence it improves the failure recovery rates. The cost related to the recovery rates of host, VM and application is approximated with the following cost functions:

$$C_{\mu_h} = K_{\mu_h}\mu_h^{\alpha_{\mu_h}} + K_{\mu_{h0}} \tag{6.11}$$

$$C_{\mu_v} = K_{\mu_v}\mu_v^{\alpha_{\mu_v}} + K_{\mu_{v0}} \tag{6.12}$$

$$C_{\mu 1_a} = K_{\mu 1_a}\mu 1_a^{\alpha_{\mu 1_a}} + K_{\mu 1_{a0}} \tag{6.13}$$

$$C_{\mu 2_a} = K_{\mu 2_a}\mu 1_a^{\alpha_{\mu 2_a}} + K_{\mu 2_{a0}} \tag{6.14}$$

where K_{μ_h}, $K_{\mu_{h0}}$, α_{μ_h}, K_{μ_v}, $K_{\mu_{v0}}$, α_{μ_v}, $K_{\mu 1_a}$, $K_{\mu 1_{a0}}$, $\alpha_{\mu 1_a}$, $K_{\mu 2_a}$, $K_{\mu 2_{a0}}$ and $\alpha_{\mu 2_a}$ are constants. $K_{\mu_h 0}$ is the minimum cost of improvements in the host repair rate. α_{μ_h} is an exponent of μ_h and K_{μ_h} represents a proportional cost to $\mu_h^{\alpha_{\mu_h}}$, i.e., the unitary cost per increment in the host repair rate. Similar meanings are assigned to the other constants in C_{μ_v}, $C_{\mu 1_a}$ and $C_{\mu 2_a}$.

VM reboot and migration rates The reboot rate (inverse of mean time to reboot) may be reduced by means of software tunings. Similar adjustments, besides network investments, may be performed in order to reduce the migration time (improving the migration rate). The cost function for migration and reboot rate are approximated as follows:

$$C_{m_v} = K_{m_v} m_v^{\alpha_{m_v}} + K_{m_{v0}} \tag{6.15}$$

$$C_{r_v} = K_{r_v} r_v^{\alpha_{r_v}} + K_{r_{v0}} \tag{6.16}$$

where K_{m_v}, $K_{m_{v0}}$, α_{m_v}, K_{r_v}, $K_{r_{v0}}$ and α_{r_v} are constants. $K_{m_{v0}}$ and K_{rv0} are the minimum costs of improvements in the migration and reboot rates, respectively. α_{m_v} and α_{r_v} are exponents of m_v and r_v. K_{m_v} and K_{r_v} represent proportional costs to $m_v^{\alpha_{m_v}}$ and $r_v^{\alpha_{r_v}}$, i.e., the unitary costs per increment the migration and reboot rates.

The values used for all new parameters, related to the cost functions, were based on experts experience, and are presented in Appendix B.

After defining the cost function, is important to highlight that the objective of this analysis is to find the most cost-effective point to make an incremental investment, aiming the optimization of $MTTF_{VMs}$ and COA_{VMs} measures. The scaled sensitivities previously computed can be used as an input for the constrained optimization method of the methodology. The method of Lagrange multipliers, presented in Section 4.2, will be used here, and the system bottleneck from the $MTTF_{VMs}$ point of view may be stated as the parameter $\theta^{\star}$, that satisfies the following relation:

$$\theta^{\star} = argmax_{\theta} \left| \frac{SS_{\theta}(MTTF_{VMs})}{SS_{\theta}(C_{improve})} \cdot \right|. \tag{6.17}$$

A similar reasoning can be given to find the bottleneck from the COA_{VMs} point of view. Therefore, Table 6.15 shows the ranking for $MTTF_{VMs}$, considering costs. The corresponding ranking for COA_{VMs} is shown in Table 6.16.

Table 6.15: Ranking for $MTTF_{VMs}$ sensitivity considering costs

Parameter θ	$\left\lvert \frac{\boldsymbol{SS_{\theta}(MTTF_{VMs})}}{\boldsymbol{SS_{\theta}(C_{improve})}} \right\rvert$
m_v	3.4×10^{2}
λ_h	3.4×10^{1}
λ_v	7.6×10^{-1}
λ_a	7.9×10^{-2}
r_v	2.2×10^{-2}
μ_v	1.4×10^{-2}

In Table 6.15, one may notice that $\theta^{\star} = m_v$, so the migration rate of VMs is the parameter which has the better balance between improvements on $MTTF_VMs$ and impact on the cost function. m_v and λ_h are, respectively, the first and the second parameters when considering $MTTF_{VMs}$. Therefore, according to this approach, those parameters are the optimization points to be considered in the virtualized server environment of this case study. In the previous analysis, these two parameters were also the most important for $MTTF_{VMs}$, but in the inverse order. Table 6.16 shows that λ_a and r_v are the first and second of ranking regarding COA_{VMs}. Note that the VM reboot rate was one of the last parameters in the ranking for COA_{VMs}, when the cost function was not considered. The low cost of improving such parameter has put it as a good cost-effective optimization point.

Table 6.16: Ranking for COA_{VMs} sensitivity considering costs

Parameter θ	$\frac{\boldsymbol{SS_\theta(COA_{VMs})}}{\boldsymbol{SS_\theta(C_{improve})}}$
λ_a	3.4×10^{-4}
r_v	9.8×10^{-5}
μ_v	6.2×10^{-5}
m_v	5.3×10^{-5}
λ_v	2.7×10^{-5}
λ_h	1.8×10^{-5}

Another interesting aspect to highlight in this analysis is that, for both measures, there are significant differences between the importance order obtained from the first approach (without cost-based optimization) and those achieved from this second one (based on improvement cost). The optimization using a cost function helps in distinguishing parameters which have similar impact on the performance/dependability measure. In this case, a higher priority will be assigned to the parameter that causes the smallest changes to the cost function. This second approach may also put a parameter down in the ranking, if the improvement of such parameter implies in an increase to costs that is not compensated by the enhancement of systems performance/dependability.

The results presented here show that, if a cost function is available, the methodology steps related to constrained optimization shall be used, in order to obtain a improvement guide that is coherent with the existing restrictions. The approach without cost-based optimization, seen in this case study and in the previous ones in Sections 6.1 and 6.2, remains valid for the cases when the values of all parameters can be changed with similar costs, when the cost function is difficult to be determined, or even if there is no kind of restriction to the system tuning. Finally, all analyses presented in this chapter helped to highlight bottlenecks in their respective systems. They were also useful to provide examples on the proposed methodology, illustrating its possible uses in systems improvement.

Chapter 7

Conclusion

Computer systems are usually evolving to satisfy increases in the demand or new user exigences. The administration of these systems requires decisions that are able to provide the highest level of performance and dependability metrics with minimal changes to the existing configuration.

It is common to carry out performance, reliability, availability and performability analysis of systems via analytical models, and Markov chains constitute one of the most used modeling formalisms, allowing to estimate some measures of interest, given a set of input parameters. Although, the most used method of sensitivity analysis is the simple variation of the set of parameters over their ranges, in order to repeatedly solve the chosen model. Formal or parametric sensitivity analysis allows the modeler to find bottlenecks in a manner that is more systematic and efficient than that variation of one parameter at a time.

This work presented an automated methodology for sensitivity analysis that aims to guide the improvement of computer systems. The proposed approach is able to speed up the process of decision making, regarding enhancements in computer infrastructures, such as software and hardware tuning, besides acquisition and replacement of components. This methodology is based on analytical modeling through Markov chains, and differential sensitivity analysis of these models. Systems which need a constrained optimization can be handled by following specific steps of the proposed methodology.

The methodology was validated by means of some case studies, that focused on bottleneck detection and system tuning. Three case studies were presented, encompassing performance, re-

liability and availability modeling of distributed systems and computer networks. Both steady-state and transient measures were considered, as well as many types of Markov chains were used to avoid biased conclusions.

The first case study considered the performance and reliability evaluation of composite web services. The analysis results showed that the airline invocation web service is the component with major impact on the overall response time. By contrast, some parameters were identified as almost irrelevant, so they could be removed from the model without problems to the accuracy of results. For the same case study, but considering the reliability, the airline invocation is also a bottleneck, but it has the same impact as the hotel reservation has on system reliability. The results obtained match the results of closed-form equations presented in the literature.

The second case study analyzed a fault-tolerant computer network. The methodology pointed out the router without redundancy as the component deserving highest priority in its improvement, considering both MTTF and MTTR parameters. By contrast, the time to repair of the spare components was shown to be the least important for the availability of the system. This case study also demonstrated how to use the sensitivity ranking to compare the importance of a given parameter in different system architectures. Redundancy mechanisms was shown to make the system availability less sensitive to failures of the primary components.

The third, and last, case study presented the analysis of two availability-related measures of a virtualized servers system. The proposed approach was important to show how the measures may have quite different sensitivities: software failure is a major concern for the capacity-oriented availability, whereas host failure is most important for system availability. The analysis using the methodology steps for constrained optimization showed sensitivity rankings with significant differences from the approach without cost function. Parameters that have small contribution to the improvement of the availability showed to be interesting optimization points due to their small costs. This demonstrated the benefit of using the steps related to constrained optimization, if cost functions are available.

7.1 Contributions

This work contributes to the field of systems optimization by presenting a precise methodology that enables a formal model-based way to perform improvements by means of changes in components of a computational infrastructure. It can benefit researchers and systems administrators, with an automated and efficient manner to make decisions in tuning performance and dependability metrics through adjusts in software and hardware parameters. It can also be used to compare models for different versions of a system, with structural changes, even if there is a large number of parameters in the model or a large number of models to be compared.

The integration of a constrained optimization technique to the sensitivity analysis makes the proposed methodology helpful in analyzing performance and dependability aspects while considers also financial cost or energy consumption, for instance. Such characteristic allows finding a reasonable trade-off between measure enhancement and increases in costs due to system tuning.

This work also proposes guidelines about the use of scaled and unscaled sensitivity functions. The difference between orders of magnitude in the parameters under analysis is the main point to decide which kind of sensitivity function will be used. A difference bigger than 3 orders raises the need for a scaling (normalization) of the computed indices. The characteristics of most performability and availability models are better represented by scaled sensitivities, while unscaled sensitivities are proper, in general, for pure performance and reliability models. The indication of the relationship between model type and sensitivity function type is other contribution of this work.

The implementation of sensitivity analysis features in the SHARPE package is another useful contribution. The use of such tool can make the process of model-driven engineering more efficient, enabling the optimization of many systems which are described by Markov chains, even if the rates and functions are defined through complex relations of input parameters. The automated sensitivity analysis of models with many parameters, such as the virtualized system model, is especially important since it may take a long time to check one parameter at a time in order to calculate the extent that changes influence the model's results. The features developed for SHARPE were essential for the analysis of the case studies presented here.

The bottlenecks identified through the case studies shall be useful in the improvement of

those systems, as well as in stimulate the adoption of the proposed methodology for the analysis and optimization of similar kinds of systems.

7.2 Future work

An investigation on the sensitivity analysis of hierarchical models is intended. Properties of sensitivity functions presented in [Frank 1978], such as chain rule and product rule, may be used in a future definition of S.A. techniques for models that are composed of sub-models of the same formalism, or even when there are different kinds of models involved in the same analysis.

The integration between differential sensitivity analysis and other techniques that generate numerical indices, such as structural importance, correlation and regression analysis is another subject to be handled in future works. This would ease the analysis of composite and hierarchical models, as well as allow the mix of uncertainty-oriented and importance-oriented sensitivities.

The functions that were developed for the SHARPE package can provide a basis for future instantiations of the methodology using stochastic Petri nets or queueing networks, since both models are solved by means of underlying Markov chains. The development of similar functions for those models would enable the analysis of systems for which it is difficult to create Markov chains directly.

Appendix A

Syntax reference of SHARPE sensitivity functions

A.1 The function stvalue

The function stvalue provides transient sensitivity result with respect to a system parameter, for a single value of t. It uses the iterative algorithm known as standard uniformization. It is the sensitivity correspondent to the **tvalue** function.

The function stvalue is valid only with Markov chains. The syntax is:

stvalue (t; system_name, state_eword, param_name {;arg_list})

A.2 The function sctvalue

The function sctvalue provides cumulative sensitivity result over (0,t), with respect to a system parameter. It uses the iterative algorithm known as standard uniformization.

The function sctvalue is valid only with Markov chains. The syntax is:

stvalue (t; system_name, state_eword, param_name {;arg_list})

A.3 The function sprob

The function **sprob** provides sensitivity of probabililty for a given state, with respect to a system parameter. It is the sensitivity correspondent to the **prob** function.

The function **sprob** is valid only with Markov chains. The syntax is:

prob (system_name, state_eword, param_name {; arg_list})

For an acyclic or phase-type Markov chain, sprob gives the sensitivity of probability that the given state was ever visited. For an irreducible Markov chain, sprob gives the sensitivity of steady-state probability for the given state.

A.4 The function sexrss

The function **sexrss** provides sensitivity of the expected steady-state reward rate with respect to a system parameter, for Markov reward models. It is the sensitivity correspondent to the **exrss** function. The syntax is:

sexrss (system_name, param_name {;arg_list})

A.5 The function sexrt

The function **sexrt** provides sensitivity of the expected reward rate at time t with respect to a system parameter, for Markov reward models. It is the sensitivity correspondent to the **exrt** function. The syntax is:

sexrt (t; system_name, param_name {;arg_list})

A.6 The function scexrt

The function **scexrt** provides sensitivity of the cumulative expected reward over (0,t), with respect to a system parameter, for Markov reward models. It is the sensitivity correspondent to the **cexrt** function. The syntax is:

scexrt (t; system_name, param_name {;arg_list})

Appendix B

Parameter values for cost functions

Parameter name	Value
α_{λ_h}	0.3
α_{λ_v}	0.3
α_{λ_a}	0.3
α_{μ_h}	0.5
α_{μ_v}	0.5
$\alpha_{\mu 1_a}$	0.5
$\alpha_{\mu 2_a}$	0.5
α_{m_v}	0.2
α_{r_v}	0.2
K_{λ_h}	2 \$/$h^{0.3}$
K_{λ_v}	2 \$/$h^{0.3}$
K_{λ_a}	2 \$/$h^{0.3}$
K_{μ_h}	4 \$ · $h^{0.5}$
K_{μ_v}	4 \$ · $h^{0.5}$
$K_{\mu 1_a}$	4 \$ · $h^{0.5}$
$K_{\mu 2_a}$	4 \$ · $h^{0.5}$
K_{m_v}	1 \$ · $h^{0.2}$
K_{r_v}	1 \$ · $h^{0.2}$
$K_{\lambda_{h0}}$	10 \$
$K_{\lambda_{v0}}$	5 \$
$K_{\lambda_{a0}}$	5 \$
$K_{\mu_{h0}}$	3 \$
$K_{\mu_{v0}}$	3 \$
$K_{\mu 1_{a0}}$	3 \$
$K_{\mu 2_{a0}}$	3 \$
$K_{m_{v0}}$	1 \$
$K_{r_{v0}}$	1 \$

Bibliography

[Abdallah 1997] Abdallah, H. (1997). Sensitivity computation of reliability markov models using the uniformized power method. *Reliability Engineering & System Safety*, 56(1):53 – 59.

[Abdallah and Hamza 2002] Abdallah, H. and Hamza, M. (2002). On the sensitivity analysis of the expected accumulated reward. *Performance Evaluation*, 47(2):163–179.

[Ajmone Marsan et al. 1984] Ajmone Marsan, M., Conte, G., and Balbo, G. (1984). A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2:93–122.

[Avriel 2003] Avriel, M. (2003). *Nonlinear Programming: Analysis and Methods*. Dover Publishing, Mineola, NY.

[Barcellos et al. 2006] Barcellos, M. P., Facchini, G., and Muhammad, H. H. (2006). Bridging the gap between simulation and experimental evaluation in computer networks. In *Proceedings of the 39th annual Symposium on Simulation*, ANSS '06, pages 286–293, Washington, DC, USA. IEEE Computer Society.

[Beaudry 1978] Beaudry, M. (1978). Performance-related reliability measures for computing systems. *Computers, IEEE Transactions on*, C-27(6):540 –547.

[Birnbaum 1969] Birnbaum, Z. W. (1969). On the importance of different components in a multicomponent system. *Multivariate Analysis - II*, pages 581–592. Academic Press.

[Blake et al. 1988] Blake, J. T., Reibman, A. L., and Trivedi, K. S. (1988). Sensitivity analysis of reliability and performability measures for multiprocessor systems. In *SIGMETRICS '88:*

Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems, pages 177–186, New York, NY, USA. ACM.

[Bolch et al. 2001] Bolch, G., Greiner, S., de Meer, H., and Trivedi, K. S. (2001). *Queuing Networks and Markov Chains: modeling and performance evaluation with computer science applications*. John Wiley and Sons, 2 edition.

[Bondavalli et al. 1999] Bondavalli, A., Mura, I., and Trivedi, K. S. (1999). Dependability modelling and sensitivity analysis of scheduled maintenance systems. In *Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, EDCC-3, pages 7–23, London, UK. Springer-Verlag.

[Cao 1996] Cao, X.-R. (1996). Uniformization and performance sensitivity estimation in closed queueing networks. *Mathematical and Computer Modelling*, 23(11-12):77 – 92. Elsevier.

[Cao 2003] Cao, X.-R. (2003). Constructing performance sensitivities of markov systems with potentials as building blocks. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, pages 4405 – 4409 Vol.5.

[Cassandras and Strickland 1989] Cassandras, C. and Strickland, S. (1989). On-line sensitivity analysis of markov chains. *Automatic Control, IEEE Transactions on*, 34(1):76 –86.

[Chandy and Martin 1983] Chandy, K. M. and Martin, A. J. (1983). A characterization of product-form queuing networks. *J. ACM*, 30:286–299.

[Choi et al. 1993] Choi, H., Mainkar, V., and Trivedi, K. S. (1993). Sensitivity analysis of deterministic and stochastic petri nets. In *Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, MASCOTS '93, pages 271–276, San Diego, CA, USA. Society for Computer Simulation International.

[Ciardo et al. 1993] Ciardo, G., Blakemore, A., Chimento, P. F., Muppala, J. K., and Trivedi, K. S. (1993). Automated generation and analysis of Markov reward models using stochastic reward nets. 48:145–191.

[Downing et al. 1985] Downing, D., Gardner, R., and Hoffman, F. (1985). An examination of response-surface methodologies for uncertainty analysis in assessment models. *Technometrics*, 27(2):151–163.

[Frank 1978] Frank, P. M. (1978). *Introduction to System Sensitivity Theory*. Academic Press Inc.

[Fricks and Trivedi 2003] Fricks, R. M. and Trivedi, K. S. (2003). Importance analysis with markov chains. In *Reliability and Maintainability Symposium, 2003. Annual*, pages 89 – 95.

[Galdino and Maciel 2008] Galdino, S. M. L. and Maciel, P. R. M. (2008). Outer estimates of interval system of linear equations: Ispn models in dependability evaluation. In *Proceedings of 2008 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2008)*.

[Galdino et al. 2007] Galdino, S. M. L., Maciel, P. R. M., and Rosa, N. (2007). Interval markovian models in dependability evaluation. *International Journal of Pure and Applied Mathematics*, 41:151–176.

[German and Mitzlaff 1995] German, R. and Mitzlaff, J. (1995). Transient analysis of deterministic and stochastic petri nets with timenet. In *Lecture Notes in Computer Science Vol. 977: Quantitative Evaluation of Computing and Communication Systems*, pages 209–223.

[GiNAC 2010] GiNAC (2010). GiNaC is Not a CAS.

[Grassi and Donatiello 1992] Grassi, V. and Donatiello, L. (1992). Sensitivity analysis of performability. *Performance Evaluation*, 14(3-4):227 – 237. Performability Modelling of Computer and Communication Systems.

[Hamby 1994] Hamby, D. M. (1994). A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, pages 135–154.

[Haverkort and Meeuwissen 1995] Haverkort, B. and Meeuwissen, A. (1995). Sensitivity and uncertainty analysis of Markov-reward models. *IEEE Transactions on Reliability*, 44(1):147–154.

[Haverkort 2002] Haverkort, B. R. (2002). *Markovian models for performance and dependability evaluation*, pages 38–83. Springer-Verlag New York, Inc., New York, NY, USA.

[Haverkort and Meeuwissen 1992] Haverkort, B. R. and Meeuwissen, A. M. (1992). Sensitivity and uncertainty analyses in performability modelling. In *Proceedings of the 11th Symposium on Reliable Distributed Systems, 1992*, pages 93–102. IEEE.

[Heidelberger and Goyal 1987] Heidelberger, P. and Goyal, A. (1987). Sensitivity analysis of continuous time markov chains using uniformization. In *Proceedings of the 2nd International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, pages 93–104.

[Heimman et al. 1990] Heimman, D. I., Mittal, N., and Trivedi, K. S. (1990). *Advances in Computers*, volume 31, chapter Availability and Reliability Modeling for Computer Systems, pages 175–229. Academic Press.

[Hirel et al. 2010] Hirel, C., Tu, B., and Trivedi, K. S. (2010). SPNP: Stochastic Petri Nets. version 6.0.

[Ho 1985] Ho, Y. (1985). A survey of the perturbation analysis of discrete event dynamic systems. *Annals of Operations Research*, 3(8).

[Jain 1990] Jain, R. (1990). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*.

[Kim et al. 2009] Kim, D. S., Machida, F., and Trivedi, K. S. (2009). Availability modeling and analysis of a virtualized system. In *Proceedings of the IEEE International Symposium Pacific Rim Dependable Computing (PRDC'09)*. IEEE.

[Kleinrock 1975] Kleinrock, L. (1975). *Queueing Systems*, volume 1. Wiley, New York.

[Kolmogorov 1931] Kolmogorov, A. (1931). Über die analytischen methoden in der wahrscheinlichkeitsrechnung (in german). *Mathematische Annalen*. Springer-Verlag.

[Kuo and Zuo 2003] Kuo, W. and Zuo, M. (2003). *Optimal reliability modeling: principles and applications*. John Wiley & Sons.

[Liu and Nain 1991] Liu, Z. and Nain, P. (1991). Sensitivity results in open, closed and mixed product form queueing networks. *Performance Evaluation*, 13(4):237 – 251.

[Loveland et al. 2008] Loveland, S. et al. (2008). Leveraging virtualization to optimize high-availability system configurations. *IBM Systems J.*, 47(4).

[Ma et al. 2001] Ma, Y., Han, J., and Trivedi, K. (2001). Composite performance and availability analysis of wireless communication networks. *IEEE Transactions on Vehicular Technology*, 50(5):1216–1223.

[Maciel et al. 2011] Maciel, P. R. M., Trivedi, K., Matias JR., R., and Kim, D. (2011). *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, chapter Dependability Modeling. IGI Global, Hershey, Pennsylvania.

[Malhotra and Trivedi 1994] Malhotra, M. and Trivedi, K. S. (1994). Power-hierarchy of dependability-model types. *IEEE Trans. Reliability*, 43(3).

[Marie et al. 1987] Marie, R. A., Reibman, A. L., and Trivedi, K. S. (1987). Transient analysis of acyclic markov chains. *Performance Evaluation*, 7(3):175 – 194.

[Marino et al. 2008] Marino, S., Hogue, I. B., Ray, C. J., and Kirschner, D. E. (2008). A methodology for performing global uncertainty and sensitivity analysis in systems biology. *J Theor Biol*.

[Marsan et al. 1987] Marsan, M. A., Balbo, G., Chiola, G., and Conte, G. (1987). Generalized stochastic petri nets revisitied: Random switches and priorities. In *The Proceedings of the Second International Workshop on Petri Nets and Performance Models*, pages 44–53, Washington, DC, USA. IEEE Computer Society.

[Marsan and Chiola 1987] Marsan, M. A. and Chiola, G. (1987). On petri nets with deterministic and exponentially distributed firing times. In *Advances in Petri Nets 1987, covers the 7th European Workshop on Applications and Theory of Petri Nets*, pages 132–145, London, UK. Springer-Verlag.

[Marwah et al. 2010] Marwah, M., Maciel, P., Shah, A., Sharma, R., Christian, T., Almeida, V., Araújo, C., Souza, E., Callou, G., Silva, B., Galdino, S., and Pires, J. (2010). Quantifying the sustainability impact of data center availability. *SIGMETRICS Perform. Eval. Rev.*, 37:64–68.

[Menascé et al. 2004] Menascé, D. A., Almeida, V. A., and Dowdy, L. W. (2004). *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall PTR.

[Molloy 1982] Molloy, M. K. (1982). Performance analysis using stochastic petri nets. *IEEE Trans. Comput.*, 31:913–917.

[Muppala and Trivedi 1990] Muppala, J. K. and Trivedi, K. S. (1990). GSPN models: sensitivity analysis and applications. In *ACM-SE 28: Proceedings of the 28th annual Southeast regional conference*, pages 25–33, New York, NY, USA. ACM.

[Nakayama et al. 1994] Nakayama, M. K., Goyal, A., and Glynn, P. W. (1994). Likelihood ratio sensitivity analysis for markovian models of highly dependable systems.

[Opdahl 1995] Opdahl, A. L. (1995). Sensitivity analysis of combined software and hardware performance models: open queueing networks. *Performance Evaluation*, 22(1):75 – 92. 6th International Conference on Modelling Techniques and Tools for Computer Performance Evalution.

[Ou and Dugan 2003] Ou, Y. and Dugan, J. B. (2003). Approximate sensitivity analysis for acyclic markov reliability models. *IEEE Transactions on Reliability*, (2).

[Reibman and Trivedi 1989] Reibman, A. and Trivedi, K. (1989). Transient analysis of cumulative measures of Markov model behavior. *Communications in Statistics - Stochastic Models*, (4):683–710.

[Reiman and Weiss 1986] Reiman, M. I. and Weiss, A. (1986). Sensitivity analysis via likelihood ratios. In *Proceedings of the 18th conference on Winter simulation*, WSC '86, pages 285–289, New York, NY, USA. ACM.

[Reiman and Weiss 1989] Reiman, M. I. and Weiss, A. (1989). Sensitivity Analysis for Simulations via Likelihood Ratios. *OPERATIONS RESEARCH*, 37(5):830–844.

[Ross 2010] Ross, S. (2010). *Introductory Statistics*. Elsevier Science.

[Sahner et al. 1996] Sahner, R. A., Trivedi, K. S., and Puliafito, A. (1996). *Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package*. Kluwer Academic Publishers, Norwell, MA, USA.

[Saltelli et al. 2004] Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. Halsted Press, New York, NY, USA.

[Sato and Trivedi 2007] Sato, N. and Trivedi, K. S. (2007). Stochastic modeling of composite web services for closed-form analysis of their performance and reliability bottlenecks. In *ICSOC*, pages 107–118.

[Shooman 1970] Shooman, M. (1970). The equivalence of reliability diagram and fault-tree analysis. *IEEE Trans. Reliability*, 19:74–75.

[Smith et al. 1987] Smith, M. R., Trivedi, S. K., and Nicola, F. V. (1987). The analysis of computer systems using markov reward processes. Technical report, Durham, NC, USA.

[Smotherman et al. 1986] Smotherman, M., Geist, R. M., and Trivedi, K. S. (1986). Provably conservative approximations to complex reliability models. *IEEE Trans. Comput.*, 35:333–338.

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

[Trivedi 2001] Trivedi, K. S. (2001). *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley and Sons, 2 edition.

[Trivedi et al. 1994] Trivedi, K. S., Malhotra, M., and Fricks, R. M. (1994). Markov reward approach to performability and reliability analysis. In *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, MASCOTS '94, pages 7–11, Washington, DC, USA. IEEE Computer Society.

[Trivedi et al. 1992] Trivedi, K. S., Muppala, J. K., Woolet, S. P., and Haverkort, B. R. (1992). Composite performance and dependability analysis. *Performance Evaluation*, 14(2-3):197–215.

[Trivedi and Sahner 2009] Trivedi, K. S. and Sahner, R. (2009). SHARPE at the age of twenty two. *SIGMETRICS Perform. Eval. Rev.*, 36(4):52–57.

[Wang et al. 2004] Wang, W., Loman, J., and Vassiliou, P. (2004). Reliability importance of components in a complex system. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pages 6 – 11.

[Watson 1961] Watson, H. A. (1961). Launch control safety study. Technical report, Bell Labs, NJ. Section VII, Vol. 1.

[Xing and Dugan 2002] Xing, L. and Dugan, J. (2002). Analysis of generalized phased-mission system reliability, performance, and sensitivity. *Reliability, IEEE Transactions on*, 51(2):199 –211.

[Yin et al. 2007] Yin, B., Dai, G., Li, Y., and Xi, H. (2007). Sensitivity analysis and estimates of the performance for M/G/1 queueing systems. *Perform. Eval.*, 64(4):347–356.

[Yu et al. 2006] Yu, H., Gibbons, P. B., and Nath, S. (2006). Availability of multi-object operations. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 16–16, Berkeley, CA, USA. USENIX Association.