Pós-Graduação em Ciência da Computação

Amanda Ferraz de Albuquerque

Modeling and Performance Evaluation of Software Maintenance Process

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Amanda Ferraz de Albuquerque

Modeling and Performance Evaluation of Software Maintenance Process

Master dissertation presented to the Postgraduate Program in Computer Science of the Informatics Center of the Federal University of Pernambuco, as a partial requirement to obtain the title of Master in Computer Science.

Concentration area: performance evaluation
Advisor: Paulo Romero Martins Maciel
Coadvisor: Teresa Maria Medeiros Maciel

Recife
2019

FICHA

Amanda Ferraz de Albuquerque

"**Modeling and Performance Evaluation of Software Maintenance Process**"

Master dissertation presented to the Postgraduate Program in Computer Science of the Informatics Center of the Federal University of Pernambuco, as a partial requirement to obtain the title of Master in Computer Science.

Approved on: 08/26/2019.

Advisor: Paulo Romero Martins Maciel

Coadvisor: Teresa Maria Medeiros Maciel

EXAMINATION BOARD

Prof. Dr. Hermano Perrelli de Moura
Informatics Center / UFPE

Prof. Dr. Renata Cristine de Sá Pedrosa Dantas
Federal Institute of Education, Science and Technology of Pernambuco / IFPE

To my family and friends.

ACKNOWLEDGEMENTS

To my advisor, Professor Paulo Maciel, who from the beginning accepted me as his student, even though he knew I wouldn't be a full time student. He has a steady pulse and guided me the way I needed, always bringing reality when needed. Also to my co-advisor, Teresa Maciel, who gave me all the support and started me on this journey. Thank you so much for your empathy, I know I took some work, but we went ahead and got a great result.

I thank my mother, Socorro, my greatest supporter, without her, I would not have started this journey. Thank you for being this battling woman who teaches me so much. And to my family, friends, and boyfriend, who gave me all the support and strength to follow through.

Thanks to all colleagues and friends that I met in UFPE and to those from the MoDCS group, especially Paulo, Renata, and Jamilson, who tried to help me and answered my questions whenever I needed without hesitation.

To Pitang, the company I worked for, which supported me with its master's support benefit, allowing me to work with a shorter workload so that I could continue my studies.

And finally, thanks to the members of the board for the opportunity to enrich this work and the willingness to participate.

"Spend your most valuable time on your most valuable activities and you'll change the trajectory of your life."(TRACY, 2017).

ABSTRACT

Teams always seek to improve their performance, but they need to understand which points in the process need to be improved. Some metrics can be raised and analyzed to assist with this performance improvement process before implementing actual process changes without knowing where their bottleneck is. A performance evaluation of a software maintenance process is performed to provide information on utilization, throughput and response time as metrics to be analyzed, assisting in the decision making process to reduce delivery time and improve the use of human resources. In a scope of software maintenance processes, this work provides a Stochastic Petri Net (SPN) model as the basis for a study of data and continuous improvement of the process under analysis. The validation of this model is performed using the experience of the IT sector of a federal public education agency, as well as a sensitivity analysis and simulation with the best scenario are presented as case studies. The results obtained with this work show that it is possible to analyze various possibilities and find, based on the proposed SPN model, a better work configuration that better leverages the human resources of the team without the cost of implementing these changes.

Key-words: Stochastic Petri Net. Performance Evaluation. Modeling. Sensitivity Analysis.

RESUMO

As equipes buscam sempre melhorar seu desempenho, mas para isso precisam entender quais pontos do processo precisam ser melhorados. Algumas métricas podem ser levantadas e analisadas para ajudar nesse processo de melhoria de desempenho, antes de implantar alterações reais no processo sem saber onde é o seu gargalo. A avaliação de desempenho de um processo de manutenção de software é realizada com o objetivo de fornecer informações sobre o uso, vazão e tempo de resposta como métricas a serem analisadas, auxiliando no processo de tomada de decisão para reduzir o tempo de entrega e melhorar o uso dos recursos humanos. Em um escopo de processos de manutenção de software, este trabalho fornece um modelo de Rede de Petri Estocástica (SPN) como base para um estudo dos dados e melhoria contínua do processo em análise. A validação desse modelo é realizada utilizando a experiência do setor de TI de uma agência federal pública de educação, assim como uma análise de sensibilidade e uma simulação com o melhor cenário são apresentadas como estudo de casos. Os resultados obtidos com este trabalho mostram que é possível analisar várias possibilidades e encontrar, com base no modelo SPN proposto, uma melhor configuração de trabalho, que aproveite melhor os recursos humanos da equipe sem o custo de implementar essas alterações.

Palavras-chaves: Redes de Petri Estocásticas. Avaliação de desempenho. Modelagem. Análise de Sensibilidade.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

λ	Lambda
θ	Theta

CONTENTS

1 INTRODUCTION

This chapter provides a brief introduction to software maintenance, its needs and demands, and presents a way to evaluate the quality of the process through its performance, focusing on capacity, speed, and productivity, as well as highlighting the importance of having this analysis for the project planning. Next, the motivations and objectives of this dissertation are presented, and finally, the general structure.

1.1 CONTEXT AND MOTIVATION

Each day the Information Technology (IT) sector of public organizations are more focused on responding to their customers' demands quickly and with quality, providing support to reach corporate strategic goals and better experiences for users of their IT services.

Software development has many phases. These phases include Requirements Engineering, Architecting, Design, Implementation, Testing, Software Deployment, and Maintenance. Maintenance is the last stage of the software life cycle. After the product has been released, the maintenance phase keeps the software up to date with environmental changes and changing user requirements (SOMMERVILLE, 2011). All this happens through a software process, that is, through the execution of a set of activities and associated results that generate a software product.

To achieve the continuous improvement in the software development, several methods and best practices can be applied to improve the process as well as the quality of IT service delivery and business alignments, such as Agile Methodologies (BECK, 2001) and Lean (BELL; ORZEN, 2012). However, regardless of the method, if there is no measurement and evaluation of the performance of the service in the value chain, there will be insufficient visibility and data to support decision making.

According to Sommerville (SOMMERVILLE, 2011) it is impossible to produce software of any size that does not need to be modified. When the software is put into use, it is possible to identify new needs, as soon as these new requirements arise, the existing requirements need to be modified. Just as errors may have been identified in the software that needs correction, or the need to improve its performance and other non-functional characteristics. This means that after the software is delivered to its customers it always evolves in response to changing demands.

Maintenance consists of four parts. Corrective maintenance deals with fixing bugs in the code. Adaptive maintenance deals with adapting the software to new environments. Perfective maintenance deals with updating the software according to changes in user requirements. Finally, preventive maintenance deals with updating documentation and making the software more maintainable. All changes to the system can be characterized

by these four types of maintenance. Corrective maintenance is 'traditional maintenance' while the other types are considered as 'software evolution' (SOMMERVILLE, 2011).

Larger and complex software projects require significant management control. According to Erdil et Al (ERDIL K.; YOON, 2003), they also introduce challenges to management as complex software systems are a crucial part of the organization. Also, the maintenance of large software systems requires a large number of employees. Therefore, management needs to find ways to increase productivity and ensure job satisfaction, which can be achieved by employing the right people, as well as motivating and training employees. Another factor that affects maintenance is selecting an appropriate way to organize maintenance tasks. This will increase productivity, control cost and deliver a quality system to the customer.

As with the case in other engineering disciplines, measurement is very significant in software engineering. Software measurement aims to find out if the requirements for software quality have been met throughout the software development life cycle. Software measurement is an approach to comprehending, controlling and managing software processes, and also to monitoring and improving their performance. This approach is about the application of a software criterion to a specific software product, and the improvement of software quality through software metrics.

There has been great interest in the measurement of software development processes in the last two decades. According to Kurtel (KURTEL, 2013), Software measurement studies are complemented by software engineering concepts, both of which aim to encourage the efforts for both software and quality improvement.

While working together with a team of a federal Organization, we felt the need to improve the response time of its services. For that, we needed to start having visibility of its work process so that we could analyze the possibilities of process changes that will result in the improvement of the services provided.

Defining adequate processes that guarantee product quality and productivity has been a challenge for the software industry. In this context, the structure of flexible processes composed of independent assets, to be easily adapted to each specific project, has been an approach widely adopted by the organizations. To increase the quality, notations such as the Unified Modeling Language - UML (LARMAN, 2006) (BOOCH, 1998) specified by the OMG provide considerable advances in the modularization and maintenance of software processes. However, these models alone do not provide support for evaluating the performance of process specifications, so it is necessary to map these semi-formal models to formal models.

Thus, environments that provide the evaluation of the processes performance and that make possible the estimation of the use of resources are mechanisms that help improve the indexes of quality and productivity of the organizations. Process execution models focused on performance estimation that takes into account combinations of diverse and active

scenarios can bring substantial gains in productivity both in the process customization and in the process effectiveness defined for the project.

With Stochastic Petri Nets being applied in the most diverse areas, ranging from the computer science to the areas of business administration and because it makes possible to specify competing, asynchronous, distributed, parallel, non-deterministic and stochastic systems (MARSAN M.A.; FRANCESCHINIS, 1995).

Methodologies and methods for process evaluation have been widely adopted in various systems engineering contexts. Performance Engineering is the term used to refer to the set of policies, activities, practices, and tools applied to the life cycle phases of the processes aiming at ensuring that the implemented solutions meet the defined non-functional requirements. It involves, therefore, the monitoring of the existing systems, considering loads appropriate to the representation of scenarios to allow the reproduction of the temporal behavior. In Performance Engineering, particular emphasis is given to stochastic modeling in the various stages of development (SMITH, 1993).

Typical representations of performance models are based on stochastic simulation mechanisms, queue networks (KLEINROCK, 1975), timed Petri nets (most notably Stochastic Petri Nets), and stochastic process algebras. Some of these means of representation enable both evaluations through simulation and via numerical analysis.

Stochastics Petri Nets allows the evaluation of systems from metrics, for example, the probability of using a computational resource (processor, disk, memory, among others), or the availability andor reliability of these resources. These metrics can be computed for a given time interval (transient analyzes), or when the system comes into equilibrium (stationary analyzes) (GERMAN, 2000). Also, the metrics results can be obtained from analytical form, as well as by model simulation (MURATA, 1989). With these information, this formalism was chosen to evaluate the Organization processes performance that will be analyzed in this study.

Being defined use Stochastic Petri Net models to map processes, statistical data can be obtained and will help organizations better analyze its processes performance and applicability. With this in mind, this work is concerned with demonstrating a way to evaluate process performance by evaluating the proposed model and analyzing its metrics and was designed to follow the objectives described in the following session.

1.2 OBJECTIVES

The main objective of this study is to define a methodology to evaluate software maintenance processes and validate it using real cases, verifying the applicability of this proposal in common scenarios in IT service organizations.

This methodology presents a series of steps ranging from the study of the maintenance process and modeling UML activity diagrams to the generation and analysis of Stochastic Petri net (SPN) models. This methodology supports the mapping of different processes

to SPN models, using a proposed resource SPN model as basis to facilitate this mapping, and is the basis for the analysis that supports the continuous improvement of the analyzed process. This can be accomplished without the need for real implementation, making the process more agile and cheap.

More specifically, this work has the following objectives:

- Propose a methodology that allows the evaluation of software maintenance processes by converting mapped processes in activity diagrams to SPN models;
- Propose a stochastic model performance model that will be used to evaluate the software maintenance process;
- Validate the proposed methodology and SPN model in different processes and recommend improvements to these processes through the sensitivity analysis results.

1.3 METHODOLOGY

The methodology used in the production of this dissertation aimed to evaluate the applicability of the proposed evaluation methodology from the perspective of studies applied in real scenarios, considering the response time, the throughput, and the utilization as metrics to be evaluated through the performance evaluation from different teams process, consisting of the generation of analytical models, scenario evaluation and presentation of alternatives for improvements in the evaluated maintenance process. In the description of the proposed methodology, the steps of how it is possible to perform an evaluation of maintenance processes are detailed, needing to define only the stakeholder interest objective to be achieved with this study. These studies were made following the guidelines of the evaluation methodology phase *Scenarios Evaluation*.

1.4 STRUCTURE OF DISSERTATION

This work is organized into eight chapters. Besides this first chapter, Introduction, Chapter 2 presents the theoretical basis of the work, introducing the fundamental concepts about software engineering, software process, software change process, the fundamental concepts about Petri nets and metrics. Chapter 3 presents the relevant works to the research, found in the literature, that demonstrate the differentiation of this dissertation concerning the other studies, commenting on their goals and what they propose, contributing to the accomplishment of this work. Chapter 4, presents the methodology used in this work from the information gathering to the analysis of the results. Chapter 5 presents a basic SPN model that represents a minimal team structure that can be used as a basis for mapping other processes. Chapter 6 present the SPN model for each process that will be evaluated and its metrics. Chapter 7, addresses studies for the application of the proposed

model. Chapter 8 sets out the conclusion and future work. The Appendix A presents the results of the searches from the digital libraries for the systematic review, the Appendix B presents the list of articles that were not returned in the searches but were included in Related Works, and finally, the Appendix C with complementary information that was used for the development of this work.

2 BACKGROUND

This chapter presents the main concepts used in this dissertation. First, concepts about Software Maintenance are introduced. Next, the concepts of software process and process modeling are addressed through the Unified Modeling Language (UML) activity diagram. After that, an overview of Stochastic Petri nets (SPN) is presented. And finally, concepts about Sensitivity Analysis.

2.1 SOFTWARE MAINTENANCE

Software maintenance is a result of changes in the business environment of the organization where the software operates, and because it is performed when the system is in its full operating phase, it can not be left in the background, an undisputed necessity and the one responsible for consuming the largest share of the financial and human resources of a software company. As software systems age, it becomes increasingly difficult to keep them up and running without maintenance.

The types of maintenance of existing software is not a consensus, for Sommerville (SOMMERVILLE, 2011), software maintenance encompasses only three activities: corrective maintenance (software defect repair), adaptive maintenance (an adaptation of software to a different operating environment), and evolutionary maintenance (to add functionality to the software or modify it). For Pressman (PRESSMAN, 2009), there is a fourth type of maintenance: preventive maintenance or re-engineering, which would be changed in the software to improve reliability or provide a better structure for future maintenance.

These four types of maintenance can be structured as shown in Figure 1, which exemplifies this structure from a modification request, followed by a generic evaluation of the types of maintenance that would be the correction or improvement processes in the software.

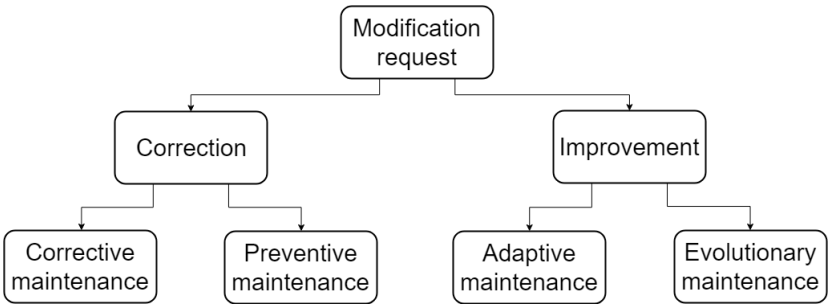

Figure 1 – Types of software maintenance.

Corrective maintenance deals with the repair of faults or defects found. A defect can result from design errors, logic errors and coding errors (TAKANG A.A.AND GRUBB, 2003).

Design errors occur when, for example, changes made to the software are incorrect, incomplete, wrongly communicated or the change request is misunderstood. Logic errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow or incomplete test of data. Coding errors are caused by the incorrect implementation of detailed logic design and incorrect use of the source code logic. Defects are also caused by data processing errors and system performance errors. All these errors, sometimes called *residual errors* or *bugs*, prevent the software from conforming to its agreed specification. The need for corrective maintenance is usually initiated by bug reports drawn up by the end-users (COENEN; BENCH-CAPON, 1993). Examples of corrective maintenance include correcting a failure to test for all possible conditions or a failure to process the last record in a file (MARTIN; MCCLURE, 1983).

Preventive maintenance concerns activities aimed at increasing the system's maintainability, such as updating documentation, adding comments, and improving the modular structure of the system (VLIET, 2000). The long-term effect of corrective, adaptive and perfective changes increases the system's complexity (TAKANG A.A.AND GRUBB, 2003). As a large program is continuously changed, its complexity, which reflects the deteriorating structure, increases unless work is done to maintain or reduce it. This work is known as preventive change. The change is usually initiated from within the maintenance organization to make programs easier to understand and hence facilitating future maintenance work (TAKANG A.A.AND GRUBB, 2003). Examples of preventive change include restructuring and optimizing code and updating documentation.

Adaptive maintenance consists of adapting software to changes in the environment, such as the hardware or the operating system. The term environment in this context refers to the totality of all conditions and influences which act from outside upon the system, for example, business rule, government policies, work patterns, software and hardware operating platforms (TAKANG A.A.AND GRUBB, 2003). The need for adaptive maintenance can only be recognized by monitoring the environment (COENEN; BENCH-CAPON, 1993). An example of a government policy that can affect a software system is the proposal to have a 'single European currency', the ECU. Acceptance of this change will require that banks in the various member states, for example, make significant changes to their software systems to accommodate this currency (TAKANG A.A.AND GRUBB, 2003). Other examples are an implementation of a database management system for an existing application system and an adjustment of two programs to make them use the same record structures (MARTIN; MCCLURE, 1983).

Evolutionary maintenance mainly deals with accommodating to new or changed user requirements. Perfective maintenance concerns functional enhancements to the system and activities to increase the system's performance or to enhance its user interface (VLIET, 2000). A successful piece of software tends to be subjected to a succession of changes, increasing the number of requirements. This is based on the premise that as the software

becomes useful, the users tend to experiment with new cases beyond the scope for which it was initially developed (TAKANG A.A.AND GRUBB, 2003). Examples of perfective maintenance include modifying the payroll program to incorporate a new union settlement, adding a new report in the sales analysis system, improving a terminal dialogue to make it more user-friendly, and adding an online HELP command (MARTIN; MCCLURE, 1983).

Among these four types of maintenance, only corrective maintenance is 'traditional' maintenance. The other types can be considered software 'evolution'. The term evolution has been used since the early 1960s to characterize the growth dynamics of software (CHAPIN N.; TAN, 2001). Software evolution is now widely used in the software maintenance community.

One of the key metrics used at the managerial level is the cost of development activities. According to surveys (SOMMERVILLE, 2011), the bulk of the IT budget for companies is targeted at software maintenance activity (approximately two-thirds of the maintenance budget, versus one-third for development).

The purpose of this study is not to present an analysis of change costs, but rather to provide a way to analyze metrics related to process performance from a proposed model and then to help decision making in project management to better take advantage of available resources, minimizing the costs of maintaining software.

Niessink and van Vliet (NIESSINK; VLIET, 2000) proposed software maintenance be seen as providing a service, whereas software development is concerned with the development of products. However, this is not yet widely recognized. Within the software maintenance domain, the focus is still on product aspects. The final phases of software development supposedly concern the delivery of an operations manual, installing the software, handling change requests and fixing bugs (VLIET, 2000).

A service is defined as an essentially intangible set of benefits or activities that are sold by one party to another (NIESSINK; VLIET, 2000). The main differences between products and services are: services are intangible; services tend to be more heterogeneous than products; services are produced and consumed simultaneously, whereas production and consumption of products can be separated and services are perishable, products are not (VLIET, 2000).

The difference between products and services is not clear-cut. For example, babysitting is a 'relatively' pure service, while packaged food is a 'relatively' pure product. There is a product-service continuum for software development and maintenance. For example, adaptive maintenance can be seen as a hybrid of product and service, whereas corrective maintenance is a product-intensive service, and software operation is a relatively pure service. A custom software development is a service-intensive product (NIESSINK; VLIET, 2000).

According to Niessink and van Vliet (NIESSINK; VLIET, 2000), customers judge the quality of software maintenance differently from how they judge the quality of software

development. This implies a need to carry out software maintenance through different processes from those used by the average software development organization.

There are a few aspects of software maintenance that set it apart from the other phases. Software maintenance cost comprises more than half of the total software development cost. Also, without software maintenance, it is impossible to change the problems within the product after its release, and many disasters can happen because of immature software.

Some characteristics of software that affect software maintenance are system size, age, and structure. Understanding the characteristics of the software will facilitate maintaining the software more efficiently. It is also important to look at how software maintenance fits into the relationship between products and services. Software maintenance, including software operation, has relatively more aspects of a service than a product, whereas software development yields a product rather than a service.

Maintenance processes vary depending on the type of software being maintained, the development processes used in an organization, and the personnel involved in the process. In some organizations, maintenance can be an informal process. Most maintenance requests arise from conversations between users and system developers. In other organizations, this is a formalized process, with structured documentation produced at each stage of the process (SOMMERVILLE, 2011)).

In general, the use of metrics is considered as important for the understanding of the generated software product itself. Even with the difficulty of comparing different products, the measurement activity assists in the analysis of quality and productivity and, through experience in past projects, in the elaboration of the planning and estimates. Despite this, few companies have made a long-term commitment to collecting data about their software (SOMMERVILLE, 2011), so it is difficult to see where the possible causes of low-quality software are.

2.2 PROCESS MODELING

A process is defined as a sequential set of steps to be followed consisting of activities, methods, practices, and transformations, used to achieve a goal, which is usually associated with one or more concrete final results, which are the products of the execution of the process. A process is characterized by containing detailed documentation of what is done (product), when (steps), by whom (agents), what uses (input) and what produces (result) (JACOBSON I.; RUMBAUGH, 1999).

In the most frequent conception, the process is any activity or set of activities that take an input, adds value to it and provides output to a specific customer. Processes use the organization's resources to deliver objective results to its customers (PEREIRA, 2011).

The purpose of process modeling is to ensure understanding of the structure and dynamics of the organization, aiming at an understanding among all involved, as well as: customers, end-users, and developers so that everyone has a common vision. The

professionals who are responsible for the modeling of business processes can count on several techniques, tools, processes, and methods used for the modeling of the processes.

Modeling makes it possible to study the system since models allow us to understand how systems work or will work, as well as allowing us to relate and 'experiment' with alternatives to how it can get better. Modeling allows our assumptions to become visible and therefore available for review and correction. The modeling enables the discussion of corrections, modifications, and validation with the client and the team at a low cost. This is possible because the discussion occurs using a simulation environment, using 'virtual' agents and inputs and times smaller than those needed for real-world experimentation. Modeling facilitates communication between members of the analysis and design teams and between them and customers and users (PEREIRA, 2011).

Models are constructed using languages that allow you to completely specify, without ambiguity, business rules, structural aspects (these include business concepts, the relationships they maintain among each other, sequences of operations, and other aspects necessary for the accomplishment of the purposes of the system. In this way, the team's understanding of all aspects of the process is the same. The modeling also allows us to document systems, recording all their characteristics, the decisions taken throughout the project and the other aspects necessary for the complete understanding and correct operation of the system.

The UML is a language with a semiformal semantic specification, which includes abstract syntax, well-defined rules, and dynamic semantics. Visual modeling is the use of semantically rich graphic and textual design notations to capture software designs. A notation, such as UML, allows the level of abstraction to be increased while maintaining strict syntax and semantics.

The UML Activity Diagram is used to describe programming logic, business processes, and workflows. For a process, this diagram determines the essential sequence rules to follow for execution, showing the flow from one activity to another (JACOBSON I.; RUMBAUGH, 1999). An activity diagram consists of activities and transitions, showing the flow of control from activity to activity. They can be used to model both sequential and concurrent activities. Also, an activity diagram can be viewed as a graph with nodes representing activities and edges labeled with transitions (SAPNA; MOHANTY, 2008).

The activity diagram is formed by some components such as the initial state and final state, that determine the beginning and the end of the control flow of the diagram. There must always be an initial state and there may be several final states. The decision component is represented by a diamond and indicates the possibility of choosing between the available flows. For the definition of conditions, you can use free text or pseudo-code and the condition is expressed in the guard condition. The merge component, also known as merge, is like the decision, represented by a diamond. Action is a task to be performed within an activity. It can be of type *entry* - executed immediately upon entering the

activity; *exit* - executed immediately before exiting the activity; *do* - performed during the stay in the activity; or *event* - performed when a specific event happens. The UML components are exemplified in more detail in Chapter 5.

An activity diagram contains activity states, which represent the execution of a statement in a procedure or the performance of an activity in a workflow. Instead of waiting for an event, as in a normal wait state, an activity state waits for the completion of its computation. When the activity completes, the execution proceeds to the next activity state within the diagram. A completion transition in an activity diagram fires when the preceding activities are complete. An activity diagram may contain branches, as well as forking of control into concurrent threads. Concurrent threads represent activities that can be performed concurrently by different objects or persons in an organization (CHEN M.; LI, 2007).In this dissertation, we consider an activity diagram as a design specification, which describes the workflow of a process. Each activity state in the activity diagram is interpreted as the execution of a step of the process.

2.3 PERFORMANCE EVALUATION

To evaluate the performance of systems, a set of techniques must be considered, which can be classified into two components: measurement-based techniques and modeling techniques (LILJA, 2005).

Measurements are only possible if something similar to the proposed system already exists, such as in designing an improved version of a product. Measurement-based techniques require the construction of a real environment and involve monitoring the system while under the action of a workload (LILJA, 2005). Techniques based on modeling can be solved both analytically and by simulation. The analytical models use closed formulas or a set of system equations to describe the behavior of a system. The metrics of interest can be provided through the solution of closed formulas or the exact or approximate solution of a set of system equations provided by numerical mathematical algorithms (BOLCH G.; TRIVEDI, 2006).

Measurement-based techniques require the construction of a real environment and involve monitoring the system while under the action of a workload. Before the workload is applied to the system, a primary study of the load to be applied must be carried out. The choice of workload is just as important as the definition of which measurement strategy should be followed since it is from this that you must choose tools and measurement strategies.

Tools that aid the performance evaluation of systems modify the behavior of what is being measured. The greater the amount of information and resolution that the measuring tool can provide, the greater the disturbance introduced by this tool. This disturbance introduced by the measurement tool makes the data collected by its less reliable (LILJA, 2005). In this way, event-driven measurement tools are, in a way, more reliable because

they offer less disturbance to the measured data, occasioned only when the occurrence of events, on the other hand, the frequency of the events is that will determine the disturbance caused by the events. Another type of measuring tool is sampled, this type of tool causes disturbances regardless of the number of times the event occurs (LILJA, 2005) (MENASCE D.A.; ALMEIDA, 2004).

Techniques based on modeling can be solved both analytically and by simulation. The analytical models use closed formulas or a set of system equations to describe the behavior of a system. The metrics of interest can be provided through the solution of closed formulas or the exact or approximate solution of a set of system equations provided by numerical mathematical algorithms (BOLCH G.; TRIVEDI, 2006).

Simulation models can be used both in the evaluation of systems performance and in the validation of analytical models. Unlike measurements, simulations are based on abstract models of the system, so they do not require the system to be fully deployed for them to be applied. Thus, the models used during the simulation are elaborated through the abstraction of essential characteristics of the system, and the complexity and the degree of abstraction can vary from one system to another. During the simulation, the values assumed by system parameters (LILJA, 2005) (MENASCE D.A.; ALMEIDA, 2004) are more efficiently controlled.

Simulation is a useful technique for computer systems performance analysis. A simulation model provides an easy way to predict the performance or compare several alternatives. Further, even if a system is available for measurement, a simulation model may be preferred over measurements because it allows the alternatives to be compared under a wider variety of workloads and environments (JAIN, 1991).

Performance evaluation helps determine system performance and the necessary improvements to these systems, however, given the number of existing systems, there is no established performance standard, not even a standard environment or technique for evaluating performance. The three performance assessment techniques are analytical modeling, simulation, and measurement. The key consideration in deciding the evaluation technique is the phase of the life cycle in which the system is (JAIN, 1991). Thus, the first step in evaluating performance is to select the most appropriate measures, environments, and techniques. Thus, the objectives will be determined seeking the best performance for a given cost, which should be done by the analyst who will indicate the requirements and compare the performance alternatives that will best meet their need.

According to Jain (JAIN, 1991), for each performance study, a set of performance criteria or metrics must be chosen. One way to prepare this set is to list the services offered by the system. For each service request made to the system, there are several possible outcomes. Generally, these outcomes can be classified into three categories: the system may perform the service correctly, incorrectly, or refuse to perform the service. For example, a gateway in a computer network offers the service of forwarding packets to

the specified destinations on heterogeneous networks. When presented with a packet, it may forward the packet correctly, it may forward it to the wrong destination, or it may be down, in which case it will not forward it at all.

If the system performs the service correctly, its performance is measured by the time taken to perform the service, the rate at which the service is performed, and the resources consumed while performing the service. These three metrics related to time-rate-resource for successful performance are also called responsiveness, productivity, and utilization metrics, respectively. For example, the responsiveness of a network gateway is measured by its *response time* - the time interval between the arrival of a packet and its successful delivery. The gateway's productivity is measured by its *throughput* - the number of packets forwarded per unit of time. The *utilization* indicates the percentage of time the resources of the gateway are busy for the given load level. The resource with the highest utilization is called the bottleneck. Performance optimizations at this resource offer the highest payoff. Finding the utilization of various resources inside the system is thus an important part of performance evaluation (JAIN, 1991).

Response time is defined as the interval between a user's request and the system response. This definition, however, is simplistic since the requests, as well as the responses, are not instantaneous (JAIN, 1991). The users spend time typing the request and the system takes time outputting the response. There are two possible definitions of the response time in this case. It can be defined as either the interval between the end of a request submission and the beginning of the corresponding response from the system or as the interval between the end of a request submission and the end of the corresponding response from the system. Both definitions are acceptable as long as they are specified. The second definition is preferable if the time between the beginning and the end of the response is long. Following this definition, the response time for interactive users in a timesharing system would be the interval between striking the last return (or enter) key and the receipt of the last character of the system's response. A reduction in the order cycle time leads to a reduction in the supply chain response time. This is an important measure as well as a major source of competitive advantage (BOWER; HOUT, 1988) (CHRISTOPHER, 1992). According to Towill (MASON-JONES; TOWILL, 1997), it directly influences the customer satisfaction level.

Throughput is defined as the rate (requests per unit of time) at which the requests can be serviced by the system (JAIN, 1991). For batch streams, the throughput is measured in jobs per second. For interactive systems, the throughput is measured in requests per second. For CPUs, the throughput is measured in Millions of Instructions Per Second (MIPS), or Millions of Floating-Point Operations Per Second (MFLOPS). For networks, the throughput is measured in packets per second (pps) or bits per second (bps). For transaction processing systems, the throughput is measured in Transactions Per Second (TPS). The throughput of a system generally increases as the load on the system initially

increases. After a certain load, the throughput stops increasing; in most cases, it may even start decreasing. The maximum achievable throughput under ideal workload conditions is called the nominal capacity of the system.

The *utilization* of a resource is measured as the fraction of time the resource is busy servicing requests (JAIN, 1991). Thus this is the ratio of a busy time and total elapsed time over a given period. The period during which a resource is not being used is called the idle time. The device with the highest total service demand has the highest utilization and is called the bottleneck device. This device is the key limiting factor in achieving higher throughput. Improving this device will provide the highest payoff in terms of system throughput. Improving other devices will have little effect on system performance. Therefore, identifying the bottleneck device should be the first step in any performance improvement project. According to Slack et al. (SLACK N.; JOHNSTON, 1995), capacity utilization directly affects the speed of response to customers' demand. Hence, by measuring capacity, gains in flexibility, lead-time and deliverability will be achieved.

Performance metrics can be evaluated by adopting measurement approaches and modeling techniques. The most suitable models to evaluate performance metrics are: Temporal Logics, Networks of Queues and Markov Chain based models (e.g., SPN) (MACIEL, 2011).

2.4 STOCHASTIC PETRI NETS

Petri nets (PNs) (MURATA, 1989) are a family of formalisms very well suited for modeling several system types, since concurrency, synchronization, communication mechanisms as well as deterministic and probabilistic delays are naturally represented. This work adopts a particular extension, namely, Stochastic Petri Nets (MARSAN M.A.; FRANCESCHINIS, 1995), which allows the association of stochastic delays to timed transitions.

The applicability of Petri nets as a tool for systems studies is important because it allows for mathematical representation, analysis of models and also for providing information about structure and behavior modeling systems. Petri net applications can occur in many areas (manufacturing systems, software development, administrative systems, among others).

Petri nets (PN) allow the modeling and analysis of discrete event systems that are too complex to be described by automata or row models (REISIG, 1992). Thus, Petri nets allow for mathematical representation and analysis through graphical models, providing useful information on the structure and behavior of the systems.

In general, Petri nets are a bipartite directed graph, in which places (represented by circles - Figure 2 (a)) denote local states, and transitions (depicted as rectangles - Figure 2 (b)) represent actions. Arcs (directed edges - Figure 2 (c)) connect places to transitions, and vice versa and tokens (marks - Figure 2 (d)) represent the current state of the system. This elements are shown in the Figure 2.

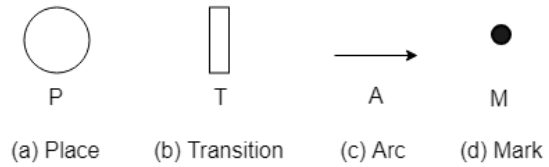

Figure 2 – Structural elements of Petri net.

To perform a certain action in a Petri net, it must be associated with some precondition, that is, there is a relation between the places and the transitions that allow or not the accomplishment of a certain action. After performing a certain action, some places will have their information changed and may create a postcondition. The arcs represent the flow of the marks through the network and the marks represent the state in which the system is at a given moment.

The original Petri Net does not have the notion of time for analyzing performance and dependability; the introduction of event durations results in a timed Petri Net. Stochastic Petri nets (SPN) are a special type of timed Petri Net, which allows the association of probabilistic delays with the transition, by using exponential distribution. Stochastic Petri nets are very well suited for modeling several system types. This is because concurrency, synchronization, communication mechanisms, and deterministic and probabilistic delays, are naturally represented.

The timed transitions model activities through the associated times, so that the timing transition enablement period corresponds to the execution period of the activity, and the trigger of the timed transition corresponds to the term of the activity. Different levels of priority can be attributed to the transitions. The tripping priority of the immediate transitions is greater than that of the timed transitions. Priorities can solve confounding situations (MARSAN M.A.; FRANCESCHINIS, 1995). The firing probabilities associated with the immediate transitions can solve conflict situations (BALBO, 2000).

It is a high-level model that allows automatically to generate and evaluate Continuous Time Markov Chain (CTMC) (TRIVEDI, 2001). This characteristic is particularly useful when the system's state space is large and/or system components interactions are complex. Besides, SPN may also be evaluated through simulation. Simulation may be the alternative when a non-phase-type distribution is required and/or the system state space is infinity.

Figure 3 shows an example of a system through a simple model in Stochastic Petri Nets. A server is shown answering a call. Place *P3* containing a token represents a state in which the server services will be available, similarly to not having a token, place *P3* represents a state in which the system is unavailable.

In this example, the directed arc of the place *P0* to the transition *T0* indicates the arrival of a call in the queue to be answered, for it, is necessary that there is a token in the place *P3*, indicating the availability of a server to attend to such a call. One having a

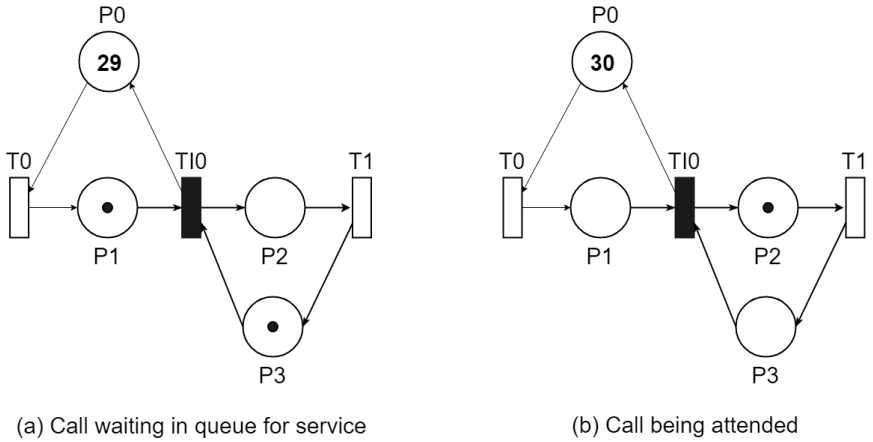

Figure 3 – Petri net system model example.

token in place *P3* the directed arc from place *P1* to the transition *TI0* indicates that the call is being resolved by the server. Likewise, the directed arc from place *P2* to transition *T1* indicates that the server has been released and has become available again, the token returns to the server location. The location of the token on the Petri net will indicate whether the call is waiting for service (Figure 3 (a)) or is being attended (Figure 3 (b)).

Timed transitions can be characterized by different firing semantics known as *single server*, *multiple server* and *infinite server* (MARSAN M.A.; FRANCESCHINIS, 1995).

In the *single server* semantics, the markings are serially processed. After the first trip of the timed transition, the timer was restarted as if the timed transition had been enabled again. This type of semantic is used in the availability models, considering that there is only one maintenance team, when several components of the system enter a fault condition.

In the semantic *multiple server*, the markings are processed with a maximum degree K of parallelism. If the degree of enabling is greater than K, no new timer will be created to process the time for the new trip until the degree of enablement has decreased of K. This type of semantic is used in the availability models considering that there is a number of maintenance teams smaller than the number of components in the fault condition. Excess components will be in a row.

In the semantic *infinite server*, the value of K is infinite, all markings are processed in parallel, and the associated timings are decremented to zero in parallel. This type of semantic is used in the availability models, considering that there are as many maintenance teams as there are the failed components. For each component, there is an exclusive and independent maintenance team. In this type of semantic, all markings are processed in parallel.

The SPN models are used for performance analysis of systems since they allow the description of the activities of systems through reachability graphs. These graphs can be converted into Markovian models, which are used for the quantitative evaluation of the

analyzed system.

2.5 SENSITIVITY ANALYSIS

Sensitivity Analysis aims at identifying the factors for which the smallest variation implies the highest impact in the model's output measure (HAMBY, 1994). The main aim of Sensitivity Analysis is to predict the effect on outputs (measures) concerning variations in inputs (parameters), helping to find performance or reliability bottlenecks, and guiding an optimization process (BLAKE J. T.; TRIVEDI, 1988). Another benefit of Sensitivity Analysis is the identification of parameters that can be removed without a significant effect on the results.

Sensitivity Analysis is an efficient method for determining the order of influence of parameters on model results. According to Hamby et al. (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, which 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.

When dealing with analytic models, Sensitivity Analysis is a particularly important technique used to find performance and reliability bottlenecks in the system, thus guiding the optimization process (BLAKE J. T.; TRIVEDI, 1988; ABDALLAH; HAMZA, 2002). It can also guide the exclusion of parameters without a significant effect on the results. Large models, with dozens of rates, may be drastically reduced by using this approach.

There are many ways of performing Sensitivity Analysis. Factorial experimental design, correlation analysis, and regression analysis are some well-known techniques (JAIN, 1991). The simplest method is to repeatedly vary one parameter at a time while keeping the others constant. When applying this method, a sensitivity ranking is obtained by noting the changes to the model output.

Without a Sensitivity Analysis, analysts often place too much emphasis on the results obtained in their analysis, presenting them as truth. However, it has no evidence of how it achieved the results (JAIN, 1991). In the absence of adequate SA, the results obtained are not certain, so some questions arise, such as: are the conclusions correct? Could the conclusions be different, or what would happen if the analysis had been performed in a slightly different scenario? Moreover, not considering the SA, it is difficult to identify which are the relevant parameters, especially if the system has many parameters (JAIN, 1991). Thus, SA brings the necessary security and can drive the results from the perspective pre-established by system administrators.

Several methods of Sensitivity Analysis are available in the literature such as: correlation analysis, regression analysis and perturbation analysis (PA), parametric differential

analysis, one by one variation, Monte Carlo simulation, Pearson correlation, Spearman correlation, analysis of variance (ANOVA), fourier amplitude sensitivity test (FAST), design of experiments (DoE), percentage difference, importance for reliability and availability and the Sobol method.

One of the methods for determining the Sensitivity Analysis parameter, and the one used in this dissertation, is to calculate the percentage output difference when varying an input parameter from its minimum value to its maximum value (HAMBY, 1994). Hoffman and Gardner (HOFFMAN F.; GARDNER, 1983), advocate the use of the full range of each possible value parameter to evaluate the sensitivity of parameters.

The sensitivity index is calculated using Equation 2.1. This equation shows the expression for this approach, where $max\{Y(\theta)\}$ and $min\{Y(\theta)\}$ are the maximum and minimum output values, respectively, calculated by varying the parameter θ over a range of n possible values of interest. If $Y(\theta)$ is known to vary monotonically, so that only the extreme values of θ (i.e., θ1 and θn) can be used to calculate $max\{Y(\theta)\}$ and $min\{Y(\theta)\}$ and hence $S\theta\{Y\}$ (MATOS R., 2015).

$$S\theta\{Y\} = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}} \tag{2.1}$$

Where,

$$max\{Y(\theta)\} = max\{Y(\theta 1), Y(\theta 2), \ldots, Y(\theta n)\} \tag{2.2}$$

and

$$min\{Y(\theta)\} = min\{Y(\theta 1), Y(\theta 2), \ldots, Y(\theta n)\} \tag{2.3}$$

Choosing which method to use to perform Sensitivity Analysis is a difficult step. This choice must be following the issues to be addressed, the available computational resources and the characteristics of the problems addressed (CAMPOLONGO F.; TARANTOLA, 1999);(PIANOSI F., 2016).

Sensitivity Analysis strategies are a key piece that can help keep these environments running most of the time throughout the year. This allowance happens as these strategies indicate the critical components with security and reliability, so that system administrators can implement corrective improvement actions on these components to improve system availability.

2.6 FINAL CONSIDERATIONS

The topics covered in this chapter such as software maintenance, process modeling, performance evaluation, Stochastic Petri Nets, and Sensitivity Analysis have guided the development of this work and made possible the establishment of connections of these concepts

with those applied in this research. The next chapter will present the related works and a systematic review of the literature.

3 RELATED WORKS

This chapter aims to present some of the work related to this dissertation, highlighting some of the main contributions related to the context of this document, as well as the need to propose improvements and the importance of evaluating the performance of the maintenance processes through the formal models.

A systematic review of the literature is a form of secondary study that uses a well-defined methodology to identify, analyze and interpret all available materials from various authors related to a specific research question impartially (KITCHENHAM; CHARTERS, 2007). It covers research to answer a key question by making a critical study of the literature. It starts with a question that guides the main objective to make a review project. Then, a literature search is done to find similar studies and finally, methodological criteria are applied to make an analysis.

The systematic review is a complex study that would take a long time, but to make more structured research of the related works, a simplification of the process recommended by Kitchenham (KITCHENHAM; CHARTERS, 2007) was made and in this chapter, the planning and results of this activity are presented. This review was carried out for six months at the beginning of this study, in 2017.

3.1 PLANNING

The first step in undertaking a systematic review is to define the research question. The research question that guided this review was:

Which studies deal with the operational analysis focused on service management?

Three digital libraries were selected to be searched to identify relevant published articles:

- Portal CAPES (http://www.periodicos.capes.gov.br/)
- Google Scholar (http://scholar.google.com.br)
- IEEE Xplore (http://ieeexplore.ieee.org/)

Keywords were defined according to related categories and are described in Table 1.

For an article to be included in the analysis, it should be available online and be clearly describing operational analysis focused on service management. The classification of the results found followed two steps: initially, when reading title and abstract, the articles were separated into two groups:

Table 1 – Keywords used in the systematic review of the literature.

Categories	Keywords
Operational Analysis	Operational
	Analysis
	Performance
Metrics	Metrics
	Statistical Analysis
Agile Methodology	Agile
Management	Management
	Stochastic modeling
	Process
modeling	Petri Net

- *[Incl]* indicating the candidate articles related to the operational analysis;
- *[Excl]* indicating articles not related to the operational analysis.

All articles in the *[Excl]* group were excluded, while the articles in the *[Incl]* group were analyzed in more detail from the reading of some sections of the article, namely: introduction, parts related to the main contribution of the article and conclusion. From this (second step), a subset of articles in the group *[Incl]* was selected, leaving only those who had information about the operational analysis.

3.2 SEARCH

For each digital library, search strings were constructed according to the characteristic of each search tool in each library and some refinements were used.

1. Google Scholar:

   a) String:

   ((Operational) AND (Analysis) AND (Performance) AND (Agile) AND (Metrics) AND (Management) AND (Stochastic modeling) AND (Statistical analysis) AND (Process) AND (Petri Net))

   b) Refinement:

   - Since 2017
   - Not include Patents and Citations

2. IEEE Xplore:

   a) String:

((((((((((Operational) AND Analysis) AND performance) AND Agile) AND metrics) AND Management) AND Stochastic modeling) AND Statistical analysis) AND Process) AND Petri Net)

b) Refinement:

- Year: 2017-2018

3. Portal CAPES:

a) String:

operational analysis performance agile metrics management stochastic modeling statistical analysis process Petri net

b) Refinement:

- 2016 – 2017

3.3 REFINEMENT

Until reaching the final result, the search went through three steps, the first was the selection by reading only the title and abstract; the second was from the materials selected in the previous step, perform the reading of the introduction, contributions, and completion sessions. Each step is detailed below.

It is possible to observe that several returned articles were excluded from the analysis since they have related keywords, however, the research area is not related to the work, the great majority was related to electrical engineering.

For an article to be included in the analysis, it should be available online, be written in English or Portuguese, and be clearly describing operational analysis in service management.

3.3.1 First step

The first step consists in reading the title and the Abstract and then classify the works from the information obtained, among the following categories:

- *[Incl]*: included articles in the results.
- *[Excl]*: excluded articles from the results.

When searching the digital libraries, a total of 65 results were returned. The list of all articles returned and the classification of each can be found in Appendix A. After the initial step, seven articles were selected for the second classification step, as shown in Table 2.

Table 2 – Initial search results.

Digital library	Number of Articles	[Incl]	[Excl]
Google Scholar	57	7	50
IEEE Xplore	4	0	4
Portal CAPES	4	0	4
Total	65	7	58

3.3.2 Second step

The seven articles of the group *[Incl]* were analyzed in more detail from the reading of the sessions: Introduction, Contribution, and Conclusion.

Table 3 – Final search results.

Digital library	Number of Articles	Repeated	Not relevant	Selected
Google Scholar	7	0	4	3
IEEE Xplore	0	-	-	0
Portal CAPES	0	-	-	0
Total	7	0	4	3

After the second step, only two related works were selected from the query string and none were repeated, as shown in Table 3.

The work (SILVA, 2017) proposed the structuring of a decision support model, seeking to highlight the scientific contribution of the Performance Evaluation applied to the public procurement planning process in Brazil. This article was excluded from the list because it focused much more on public management and was not relevant to the study in question.

The article (SETH D.; DHARIWAL, 2017) has almost the same objectives, which would improve cycle time, but uses Value Stream Mapping (VSM) with graphical resources as the basis for identifying value activities and non-value in ETO environments (Engineer to Order) and HMLV (High-Mix Low-Volume) in waste reduction. (VASAVA, 2017) also works with VSM to streamline process production and try to identify non-value-added activities to make the work process efficient. These studies were also excluded from the analysis because they were not related to performance evaluation.

In (BI J.; LI, 2017), the focus is on dynamic resource allocation to minimize service provider energy costs and maximize revenue in complex cloud environments. Also being classified as not relevant because it is not the focus of the study.

In (LIMA-JUNIOR; CARPINETTI, 2017) a literature review of 84 studies proposing quantitative models that support the performance evaluation of Supply Chain is also carried out. It is also not included in the study and is excluded from the analysis because it suggests a framework for literary analysis.

Other works that did not return in the queries of the digital libraries, but meet the inclusion criterion defined were included in the related works. These related articles are listed in Appendix B and discussed in the following section along with those selected from the systematic review.

3.4 RELATED WORKS

In the article (SILVA F.A.; MACIEL, 2017), a modeling strategy in SPN (Stochastic Petri Nets) is proposed to represent the execution of a method call of mobile cloud systems. This approach allows a designer to plan and optimize Mobile Cloud Computing (MCC) environments where SPN's represent the behavior of the system and estimate the runtime of parallelizable applications. SPNs are used in this study to estimate the performance metrics of each downloadable task and the entire application. This work has as contributions the design and implementation of an SPN modeling approach that allows predicting the behavior of the system in terms of execution time by calculating three statistics: (i) estimated execution time of the application based on the number of remote server instances; (ii) the number of method tickets per unit of time; and (iii) the probability of terminating application execution at a specific time; with this same SPN model, a representation of the MCC and the application was made; an SPN modeling approach that represents the distribution of tasks that allow predicting the number of necessary target resources, and finally, the MCC-Adviser was created, a graphical tool that generates and solves SPN's based on the proposed model to plan and design an MCC environment.

In (APRIL A.; DUMKE, 2005), a maturity model for daily software maintenance activities is proposed to evaluate and improve the software maintenance process based on CMMi integration (Capability Maturity Model integration) and is designed to be used as a complement to this model. An inventory of software engineering maturity models is also presented, identifying those that include software maintenance topics and discussing the limitations of the CMMi model concerning unique software maintenance activities. The authors present an overview of a proposed Software Maintenance Maturity Model (SMMM) and its architecture. To illustrate the details of this model, the objectives and detailed practices of a Key Process Area (KPA) are presented. An initial validation consisted of case studies in an industrial setting. Empirical studies on the use of the SMMM as a tool for continuous improvements in maintenance management could contribute to the development of a better understanding of the problems of the software maintenance function.

In (AN Y.; CHEN, 2017), the problem of developing a traffic-priority traffic control model using colored hierarchical Petri nets (HCPN) is considered. This paper focuses on the use of Petri nets (PN) to formalize the traffic signal priority control model (TSP), adopting state-space analysis to verify the correctness and reliability of the control model before the implementation phase. The resulting system is based on the on-board unit and

road unit (RSU) network for TSP applications that provide traffic vehicles with safety and convenience. This work for the first time uses HCPN to design a traffic control system for a four-phase intersection dealing with priority to provide public transit passing through the intersection. This helps to enhance the state of the art in traffic real-time detection and traffic management of an intersection. Future work is to design a traffic control system that is complex and close to the actual application. For example, the impact of pedestrians on the transit priority should be considered.

In (JIN Y.; GE, 2017), the authors propose an approach to find backlogged activities in software processes and present a case study that demonstrates the high-level efficiency of the approach to concretely illustrate this solution. Besides, some reasonable reviews and modifications are developed at the end. In this paper, is adopted two research methods (quantitative analysis method and validation method). On one hand, it is designed an algorithm to perform quantitative analysis for calculating probabilities of the places with tokens. On the other hand, the research is validated through a case study. First, a transaction flow diagram (TFD) of the software process was constructed and then we transferred it to the SPN model, then the isomorphic Markov chain (MC) and the achievable marking chart of the SPN were drawn to calculate the equations in the MC probability transfer matrix. Thus, it was possible to locate the sites that contain tokens with the highest probability values, which correspond to the activities delayed in software processes. Finally, a practical example was presented to show the correctness and rationality of this algorithm.

Another work (RACHDI A.; DAHCHOUR, 2016) related to Petri nets and process evaluation is based on the presentation of a method for the verification of BPMN models, defining the formal semantics of BPMN in terms of mapping for timed Petri nets (TPN). Given that BPMN has not been provided with a formal semantics, which limits the analysis of BPMN models to using solely informal techniques such as simulation, in order to address this limitation and use formal verification, this work defines a certain "mapping" between BPMN and a formal language, that means, a method for the verification of BPMN models by defining formal semantics of BPMN in terms of a mapping to Time Petri Nets (TPN), which are equipped with very efficient analytical techniques. After the mapping, a verification is done to ensure that some functional properties are satisfied by the model under investigation, namely liveness and reachability properties. The main advantage of this approach over existing ones is that it takes into account the time components in modeling Business process models.

The study (DIAS, 2010) proposes a model in the SPN to evaluate the impact of costs in executing the software change process by studying its difficulties in the execution, deadlines, effort and cost in the CCB's involvement in the impact analysis. It focuses on possible process failures that can generate rework, whether there is planning and what roles are allocated for the analysis of the change request. Also, a performance evaluation methodology is proposed, to assist the modeling and evaluation processes, using metrics

and analyzing them as an instrument to better understand the difficulties and to assemble strategies by facilitating the use of the process. Finally, case studies were presented showing the applicability of the work. This study had as a contribution a developed methodology that can be applied to support the planning to any other type of software process, wherewith the application of the proposed methodology, several problems related to the modeling of the analyzed process can be solved.

The authors, (RAMAMOORTHY; HO, 1980), discussed a systematic method to evaluate and verify the performance of concurrent systems. The system to be studied is first modeled by a Petri net. Based on the Petri net model, the system is classified. Procedures for predicting and verifying the system performance of all three types are presented. It is discussed as a systematic method to evaluate and verify the performance of concurrent systems. The system to be studied is first modeled by a Petri net and then is classified into either: a consistent system or an inconsistent system. A consistent system is further subclassified into a decision free system; a safe persistent system; or a general system. The performance of decision-free systems and safe persistent systems can be computed quite efficiently. With that, an approach for computing the upper and lower bounds of the performance of a conservative general system is proposed. However, further research is needed.

The paper (CAR; MIKAC, 2002) proposes a method for software process modeling and evaluation of telecommunication software maintenance process performances to improve the software maintenance process. The method is based on queuing networks and it applies discrete simulation to determine process performances. The queuing network model focuses on such as utilization and service quality expressed as the time the user modification request spends in the maintenance process. This work was divided into seven activities: project definition; process modeling in the form of queuing network; collection and data analysis; queuing and model simulation; process performances analysis; modeling and analysis of alternative process designs; and finally, comparison of alternative design performances. The suggested method can be applied to different telecommunication software maintenance processes.

In (FEBBRARO A.; SACCO, 2016) a deterministic and Stochastic Petri Net based microscopic model for urban traffic networks, whose objective is the heuristic optimization of the green duration of each phase has been proposed. The model can represent both light and saturated traffic conditions, as well as deadlock situations which may affect a traffic system minimization of queue lengths in the traffic network through the optimization of phase durations. This paper describes the adopted model for an urban traffic network and proposes the Petri net models for representing traffic dynamics within intersections, roads, and traffic networks, together with a discussion about their possible uses and applications. The control system aimed at minimizing the traffic congestion of the network is also presented, and a case study and some numerical results are reported to illustrate

the proposed approach. The control strategy has the objective of minimizing the sum of queue lengths through the optimal setting of phase durations every 5 minutes, following the incoming traffic flows.

The work (FADAHUNSI; SATHIYANARAYANAN, 2016) proposes an approach to using a mathematical formula to revise business processes modeled combined with Petri nets formalism, to reduce the cost and cycle time taken to complete the business process by reducing the number of tasks needed to complete the process. Based on the findings of this research, it has been confirmed that it is possible to exploit the graphical and quantitative features of Petri nets, combined with combination mathematical theory, in revising a current business process to generate future process design options based on the objective of process attribute optimization via a reduction in the number of tasks or events required in an end-to-end business process.

The present work offers a different approach from those described above and from others in the literature by proposing a modeling strategy through SPN to evaluate the software maintenance process and estimate performance metrics to support backlog reduction and better use of resources.

3.5 COMPARISON OF MAIN RELATED WORKS

Table 4 summarizes the main related works mentioned in this Chapter, establishing a comparison between them and this thesis based on these four subjects: operational analysis, performance metrics, modeling, and agile methodology.

The papers (FEBBRARO A.; SACCO, 2016) and (AN Y.; CHEN, 2017) both deal with modeling the traffic lights system, one using colored Petri Net and the other using Stochastic Petri Net modeling. (SILVA F.A.; MACIEL, 2017) uses Stochastic models to evaluate mobile cloud performance. In (RACHDI A.; DAHCHOUR, 2016) it is possible to find an analysis of BPMN models based on Time Petri Nets. In (RAMAMOORTHY; HO, 1980) it is made a performance evaluation of asynchronous concurrent systems Using Petri Nets. By comparison, all are different from what is proposed in this thesis, which would be an evaluation of the software maintenance process through its SPN model.

The paper (APRIL A.; DUMKE, 2005) proposes a maturity model for daily software maintenance activities, for this maturity model construct it was made these processes mapping. In (JIN Y.; GE, 2017) showed an approach to locating delayed activities in Software Processes. Dias (DIAS, 2010) mapped the change software process using SPN. Car (CAR; MIKAC, 2002) made the maintenance process modeling using a queue system. And finally, in (FADAHUNSI; SATHIYANARAYANAN, 2016) was made a business process mapping in SPN. None of them used sensitivity analysis. This thesis presented here covers those four subjects that had not been previously combined in the literature reviewed so far.

Table 4 – Comparison table of related works

Work	Process Mapping	Performance metrics	PN Modeling	Sensitivity Analysis
This thesis	*Yes*	*Yes*	*Yes*	*Yes*
Dias, M.D.S.(DIAS, 2010)	*Yes*	*Yes*	*Yes*	No
Silva, F.A. et. al. (SILVA F.A.; MACIEL, 2017)	No	*Yes*	*Yes*	No
Jin, Y. et. al. (JIN Y.; GE, 2017)	*Yes*	No	*Yes*	No
Rachdi, A. et. al. (RACHDI A.; DAHCHOUR, 2016)	*Yes*	No	*Yes*	No
Fadahunsi, O. and Sathiyanarayanan, M. (FADAHUNSI; SATHIYANARAYANAN, 2016)	*Yes*	No	*Yes*	No
Ramamoorthy, C.V. and Ho, G.S. (RAMAMOORTHY; HO, 1980)	No	No	*Yes*	No
Car, Z. and Mikac, B. (CAR; MIKAC, 2002)	*Yes*	No	No	No
An, Y. et. al. (AN Y.; CHEN, 2017)	No	No	*Yes*	No
April, A. et. al. (APRIL A.; DUMKE, 2005)	*Yes*	No	No	No
Di Febbraro, A. et. al. (FEBBRARO A.; SACCO, 2016)	No	No	*Yes*	No

3.6 FINAL CONSIDERATIONS

This chapter presented a systematic review of the literature based on the process recommended by Kitchenham and was presented some of the works related to this dissertation. The next chapter will present the proposed support methodology used to guide this study.

4 EVALUATION METHODOLOGY

This chapter presents the support methodology used during the research to understand the maintenance model, to support create an SPN model to be analyzed and simulated, and the results of these evaluations will support decision making for process improvements.

4.1 OVERVIEW

Although organizations have an interest in improving their processes, they generally do not evaluate the performance of the process that guides the activities of the team. To provide a better understanding of the software maintenance model and the performance evaluation of the process, it is possible to identify process bottlenecks, this section presents a methodology to support organizations in decision making, proposing process modeling of software maintenance in performance models based on Stochastic Petri Nets (SPN).

The support methodology used in this study is divided into three phases: Process Mapping, Model Evaluation, and Scenarios Evaluation. Process Mapping involves two steps: Understand the maintenance model and define the activity diagram, and Generate SPN model with parameters and metrics. The Model Evaluation is divided into the steps: Validate model, Evaluate model and metrics, and Adjust model and/or metrics. In Scenarios Evaluation we have the steps: Setting Scenario and change parameters, Execute the model simulation, Execute the Sensitivity Analysis, Analyze the results, and finally Present results and recommendations. The flowchart of Figure 4 represents visually the methodology adopted in this work.

The rectangles represent each step of the support methodology that was followed obeying the order of execution pointed out by the arrows. Only when one step is completed does the evaluator move on to the next step. The diamond represents a decision and we have two different paths, depending on the result obtained. In this case we have two decisions: (1) the analysis can proceed to the next step if the models and the results are satisfactory, that is, they do not present errors in the model or return unusual results, or go back to the previous step for adjustments if they are not; (2) when the analysis of another scenario is necessary, go back to set the new scenario and change the parameters, or finalize the study and present results and recommendations.

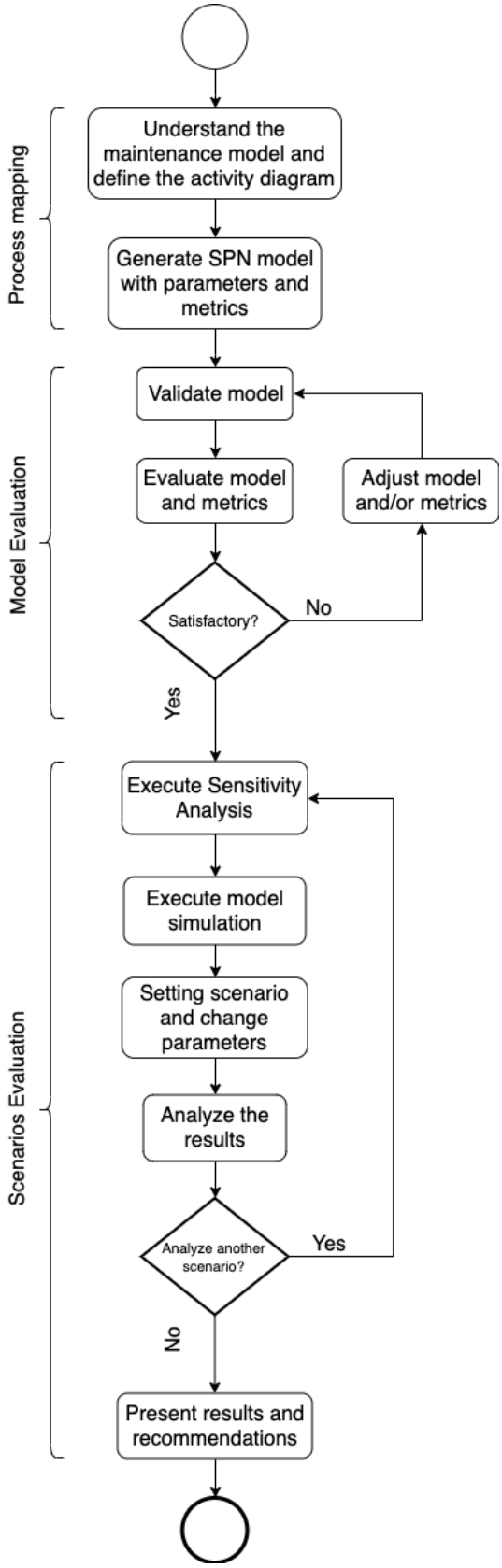

Figure 4 – Methodology.

4.2 PROCESS MAPPING

This section presents the three steps that constitute the Process Mapping phase: Understanding the maintenance model, and generation of the process model with parameters and metrics. The objective is to present the necessary inputs for the implementation of the next phase.

Understand the maintenance model and define the activity diagram: The step of understanding the maintenance model was carried out following the maintenance team and by understanding the way the team works and the steps that the ticket goes through until it is resolved. In this step was constructed the UML model of the process to be evaluated, which served as the basis for the construction of the SPN model. This is the main point because it will serve as a basis for the process modeling, so the understanding of the process requires great attention and special care by the evaluator to avoid errors of interpretation and commitment of the other stages of the methodology. It is part of this step as well as raise the parameters to use in the model evaluation, this can be done by searching for historical data in the tool used by the team to manage the tickets. The result obtained from this step is further detailed in Chapter 6.

Generate process model in SPN with parameters and metrics: At this stage, the models and metrics expressions will be effectively constructed, considering the understanding of the maintenance model that was raised in the previous step and the SPN base model described in Chapter 5. For the SPN modeling, we used the Mercury tool (SILVA B.; MACIEL, 2015), where we can do the evaluation and allows the computation of the metrics. The model, the metrics expressions and the parameters used to validate the model are detailed in Chapter 6.

4.3 MODEL EVALUATION

The Model Evaluation phase is divided into three steps: Validate model, Evaluate model and metrics and Adjust model and/or metrics. It aims to use all the knowledge acquired and material developed in the previous steps to evaluate the SPN model.

Validate model: With the model built, it is made a validation through the Token Game from Mercury tool (SILVA B.; MACIEL, 2015) to ensure that the model is analyzable and simulable. Token Game corresponds to the functionality of the tool in which users can simulate SPN models' behavior. In order words, users are able, for instance, to debug the model that is under analysis. Thus, model construction mistakes can be easily discovered as well as their solution. This step is detailed in Chapter 6.

Evaluate model and metrics: This is the step where the evaluation is executed and then the output values are compared with a predefined reference value, in this case, it would be the parameters obtained from historical data raised in the *Understand the maintenance model and define the activity diagram* step. When the computed value is satisfactory,

the process proceeds to the step *Setting scenario and change parameters*. However, if at least one metric of interest has not reached a satisfactory level, the process must go to the *Adjust model and/or metrics* step so that necessary changes are made and then the model is reevaluated. This step is detailed in Chapter 7, Section 7.1.

Adjust model and/or metrics: In case the results from the evaluation are not satisfactory we must change the model and/or metrics until we reach a satisfactory comparison with the historical data.

4.4 SCENARIOS EVALUATION

The final phase of the methodology is the Scenarios Evaluation and it is composed of the following steps. The objective is to use the results that were obtained in the previous evaluation step and use them to construct scenarios, giving possibilities of configurations to analyze possibilities of improvements.

Execute Sensitivity Analysis: Sensitivity Analysis is useful for testing different scenarios to answer "what if" questions and identifying which number of uncertain input values have the greatest impact on a specific evaluated metric. This analysis can also be done using the Mercury tool. A range is defined where the metric to be analyzed varies and then the list of metrics that most impact the result of the analyzed metric is returned. This data can be used to help generate the scenarios to be evaluated.

Setting scenario and change parameters: This step of the methodology is the construction of the scenario for the evaluation. Based on the *Evaluate model and metrics* step, which uses parameters with historic team data, it is possible to identify bottlenecks in the process. Possible improvements can be a scenario to be evaluated. For this, it is necessary to insert the new parameters in the model to be evaluated. This technique does not guarantee the best answer, but it is possible to analyze various situations. This step is detailed in Chapter 7.

Execute model simulation: This is the step where the transient simulation is executed using the Mercury tool and the results obtained from this simulation will be compared with the results of other possible scenarios and the outcome of the *Evaluate model and metrics* step.

Analyze the results: This step verifies the results achieved based on scenarios, with a good estimate that this same behavior can be presented by a real system. In this way, it is possible to identify the influence of a given parameter on the analyzed metrics. With the sequence of parameter changes and model evaluations/simulations, scenarios are generated. The choice of the best parameter combination gives us the best scenario based on the characteristics adopted. The description of scenario generation and search for better solutions are explained in detail in Chapter 7.

Present results and recommendations: The final step of this support methodology is to present the results and recommendations. That is, with all the data obtained in the

simulations and the analysis made from it, it is necessary to present this information to the interested parties. Start giving context about what goals this evaluation was trying to achieve; Present the data in charts and their analysis comparatively with the parameters obtained from the historical data raised in the step *Understand the maintenance model and define the activity diagram*; and finally show the recommendations and explain how implementing each suggestion will benefit the process.

4.5 FINAL CONSIDERATIONS

This chapter presented the proposed support methodology that will be used as a guide to this study to understand the process to be evaluated, to support create an SPN model for this process and the results of the evaluation be used to support decision making for process improvements. The next chapter will present a basic SPN model to be used as a basis for the mapping process and will explain the translation of activity diagrams into SPN models.

]

5 MODELING: TRANSLATING ACTIVITY DIAGRAMS TO SPN MODELS

This chapter presents a basic SPN model that represents a minimal team structure that can be used as a basis for mapping other processes. The translation of activity diagrams into an SPN model with examples is presented and performance metrics are described.

5.1 PROCESS MAPPING

The most widely used way to specify software processes is through semiformal languages, such as UML (Unified Modeling Language), mainly due to its friendly and intuitive notation. The UML was developed to specify, visualize and model software systems, as well as for business modeling and software processes. The UML represents a notation that has been successfully approved in system modeling (SIEGEL, 2019). However, semiformal models generated by these languages alone do not provide support for evaluating the performance of system specifications, so it is necessary to map these semiformal models to formal models. Formal models are backed by solid mathematical fundamentals that support their precise semantics, stimulate performance evaluation, and provide support for qualitative property checks and analyzes.

Activity diagrams are used to describe programming logic, business processes, and workflows. Also, this diagram has many features that allow you to represent complex structures, such as parallel and conditional processes, exceptions, events, among others. Activity diagrams represent pre and postcondition actions and their results and provide a graphical representation of the flow of interactions. An activity diagram uses rounded rectangles to describe a specific function of the system, arrows to represent precedence relations between the system components and diamonds to represent decisions (SIEGEL, 2019) (PENDER, 2003).

In the next sections, the elements of the UML activity diagrams are mapped to SPN to adopt them for the construction of the process model. Also, is proposed a stochastic Petri net model and its expressions to calculate the utilization, throughput and response time metrics for the maintenance process that is studied. The proposed model for performance evaluation will address the problem of bottlenecks. The validation of the proposed model is carried out in Chapter 7, section 7.1.

5.2 MAPPING ACTIVITY DIAGRAMS IN SPN

Stochastic Petri Nets have been chosen because it provides refinement and abstraction mechanisms that are of great importance for complex systems designs, there is a wide variety of tools available for modeling, analysis, and verification and has several extensions

for the representation of characteristics of competition study and analysis of practical problems of organizations (GIRAULT; VALK, 2003).

The activities represent the execution of a process, involving one or more actions. An action consists of processing that results in a change of state in the system.

Figure 5 shows a simple flow of activities represented through the activity diagram. This diagram describes a simplified example of the ticket opening process. The activity of requesting the opening of a ticket starts with the ticket to the Service Central. Once the problem is understood, the activity of opening the ticket begins in the monitoring system. Finally, the last activity consists of the screening for the Central responsible for solving the problem.

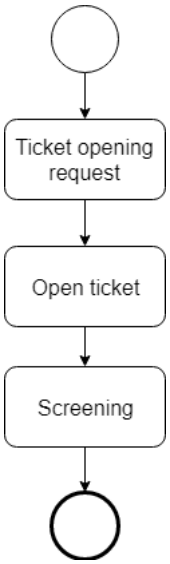

Figure 5 – Example of an Activity Diagram.

Activities are represented by rectangles with rounded corners and depict an invocation of an operation that may be physical or electronic and action represents a step within a single activity.

In the SPN Model generated by the mapping process, activities are represented by places and transitions. The places *in_A* and *out_A* represent, respectively, the preconditions and postconditions of activity *A*. The transition *t_A* can represent the duration of the activities or a set of conditions for assigning a certain time, such as the variation of the time of the activity according to the number of people involved. Figure 6 illustrates the mapping of an activity to the SPN model.

According to [Gue08], a transition represents an event that causes a change in the state of the entity, generating a new activity. Transitions are graphically represented by an arrow pointing to the target activity. Besides, a transition can have several origins (in this case, it represents a junction of several simultaneous activities) and targets (in this case, it represents a fork for various activities simultaneous). The transition represents

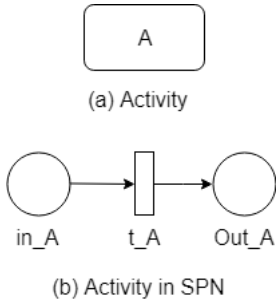

Figure 6 – Mapping activities to SPN model.

the relationship between two or more activities and is called a non-activated transition because it does not represent a space of time.

In Figure 5, a transition related to the ticket opening process is presented. In this example, there are two activities, the first, the user acts performing the opening request. After this action is performed, it causes a transition to the open ticket activity. The transition is represented by an arrow connecting the two activities.

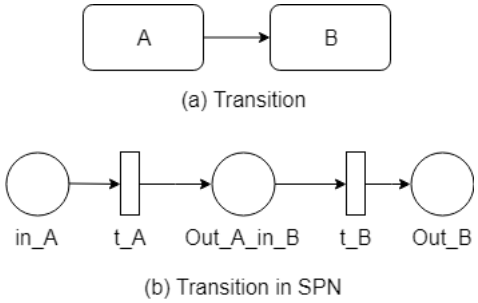

Figure 7 – Mapping transitions to SPN model.

Figure 7 (a) represents the transition from activity A to activity B in the UML activity diagram. In the SPN model generated by the mapping process; the transitions are represented by a place. The place ($out\_A\_in\_B$) represents the transition from activity A to activity B (see Figure 7 (b)).

The initial state is a pseudo-state, whose function is to indicate (point to) the starting point of the activity diagram, this pseudo-state is represented by a circle (see Figure 8 (a)). This initial state is represented by a place composed of a mark in the SPN model, Figure 8 (b) represents the initial state, where the place $iniSta\_A$ contains the mark. The transition t_in_A represents the transition between the initial state and the beginning of an activity A. Figure 8 shows the mapping.

The final state is a pseudo-state, whose function is to indicate the completion of the flow in the activity diagram, this pseudo-state is represented by a bold circle, as can be seen in Figure 9 (a).

The final state is mapped in one place ($endSta\_A$) in the SPN model, in which the presence of a mark represents the end of the activity diagram, as can be seen in Figure 9

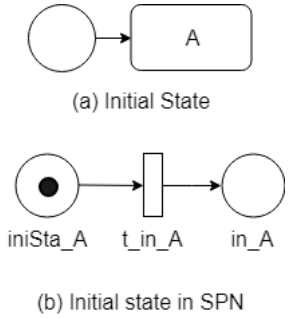

Figure 8 – Mapping initial state to SPN model.

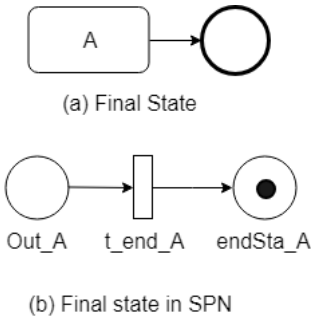

Figure 9 – Mapping final state to SPN model.

(b). Also, the transition *t_end_A* is used to represent the transition between the terminus of activity *A* and the final state. Figure 9 shows the mapping.

The decision represents a point in the flow of control where a decision must be made. It is usually represented by a diamond with one entrance and several exits. The outputs have guard conditions that control which transitions (from a set of alternative transitions) succeed the activity that will be completed. Figure 10 shows an example in which, at the end of the sorting activity, two possibilities are verified: perform maintenance or adjustment in the database. If you choose to perform maintenance, the flow will go to the maintenance activity, if you choose adjustment in the database it is forwarded to the database activity.

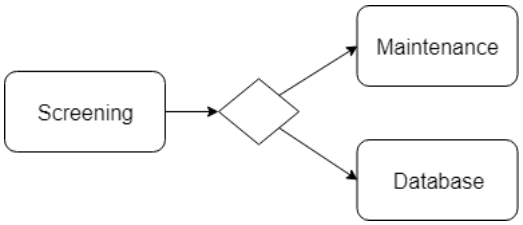

Figure 10 – Example of an activity diagram decision.

In Figure 11 (a), the result of activity *A* will be verified by the decision element through the expressions of conditions and thus chosen one of activities *B* or *C*. In the mapping to SPN, the decision is defined by two transitions *t_decB* or *t_decC*, see Figure 11 (b).

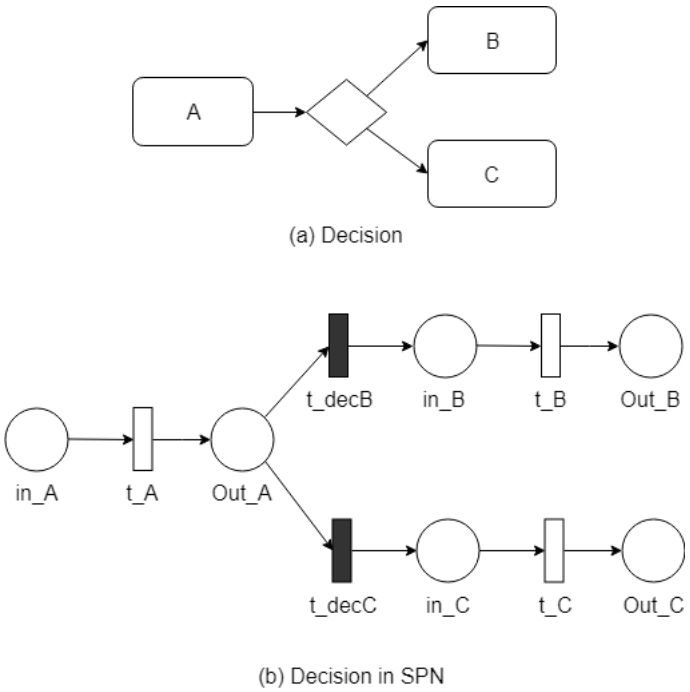

Figure 11 – Mapping decisions to SPN model.

5.3 SPN PROPOSED MODEL

Based on these transformations of activities, transitions, initial and final states and decisions in SPN, the service model of a team role is proposed. The model is shown in Figure 13 represents the arrival of a ticket and its resolution in a certain area, based on the queuing and server model described in the above session.

In this work is being used, for means of modeling and simulation, the Mercury tool, that is a software for supporting performance, dependability, and energy flow modeling easily and powerfully. The tool provides graphical interfaces for creating and evaluating Stochastic Petri Nets (SPN), Reliability Block Diagrams (RBD), Energy Flow Models (EFM), Continuous-Time Markov Chains (CTMC) and Discrete-Time Markov Chain (DTMC) (GROUP,).

To help understand the SPN model, it is possible to compare with the queue representation showed in Figure 12. The queue part is equivalent to place *P0*, and the server is represented by *P3*.

The transition trigger *T0* is generated at an arrival rate $\lambda T0$. The letter K in the place *P0* represents the definition of the buffer size and the letter N in the place *P3*, represents the quantity definition of persons working in the area. The transition *TI0* is immediately triggered upon reaching a ticket in the queue, represented by the place *P1*, as long as there is an available resource, that means, there is at least one token in place *P3*. After a rate $\lambda T1$ the ticket is resolved by the area in question and leaves the place *P2* for the next area queue if it exists, otherwise, the token is consumed and represents

the ticket resolution.

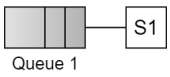

Figure 12 – Queue notation representing a queue and its server.

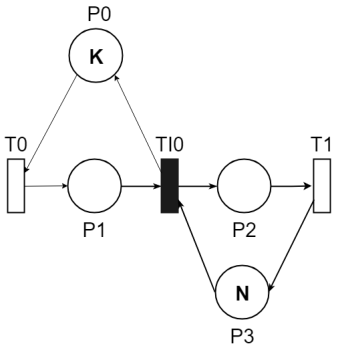

Figure 13 – SPN resource representation.

The idea of this base model is that it can represent any area of the process, that is, a piece of the puzzle that is joined to others would form the process to be studied. In the study case in question, this base model is used to represent the areas of analysis, database, development, testing, and configuration engineering.

5.4 METRICS

After mapping the SPN models is the time to generate their metrics equations. Performance metrics were used to evaluate the currently used software maintenance process, with the main objective of having a sense of the use of the available resources and being able to evaluate the ticket delivery capacity. The metrics selected for analysis in this study are described below using the notation adopted by the Mercury tool, for the SPN resource representation (Figure 13):

Area Utilization (AU)[1]: The expression below represents the percentage of the occupation level of at least one person in the analyzed area is working in one ticket. This metric is obtained by computing the probability of existing tokens in place *P2*.

$$AU = P\{\#P2 > 0\}$$

Individual Utilization (IU)[2]: The expression below represents the mean of the occupation level of the people in the analyzed area to carry out the ticket. This metric is obtained by

[1] Operator P{#*Pn* <condition>}, is the probability of the number of tokens in place *Pn* reaches the specified condition.

[2] Operator E{#*Pn*}, is the expected number of tokens in place *Pn*

computing the expected number of tokens in place *P2*, divided by the number of people in the area.

$$IU = \frac{E\{\#P2\}}{N}$$

Throughput (TP): The following expression represents how many tickets per hour the area can deliver.

$$TP = \frac{1}{\lambda T1} \times (1 - ((P\{\#P1 = K\}) + (P\{\#P2 = N\})))$$

Response Time (RT): The expression below represents how long the team can resolve one ticket. This metric is obtained by computing the expected value of tokens at the places *P1* and *P2*, divided by the throughput (TP).

$$RT = ((E\{\#P1\}) + (E\{\#P2\}))/TP$$

The most widely used resource can be considered the bottleneck of the system. Also, if this resource is considered the bottleneck of the system, it should have greater attention on the part of the appraiser to optimize it.

5.5 FINAL CONSIDERATIONS

This chapter presented the translation of activity diagrams into the SPN model using examples and presented a basic SPN model as a proposal to be the basis of process mapping. The next chapter will present the process mapping for two different teams in SPN using this basic SPN model and the expressions for each metric.

6 PROCESSES MAPPING

This chapter aims to present the SPN model of each process that will be analyzed in Chapter 7 and the expressions of their metrics following the guideline of the proposed methodology initial steps: *Understand the maintenance model and define the activity diagram*, and *Generate process model in SPN with parameters and metrics*. The processes of two different IT teams from a federal public education agency (Federal University of Pernambuco - UFPE) that will be used as input for scenario evaluation are presented.

6.1 TEAM ONE PROCESS MAPPING

The team one is a maintenance team responsible for the SIG@ system and is part of the Operations Central, the UFPE IT sector. The SIG@ system is used by the academic community of UFPE to support teaching, research, human resources, administrative processes, institutional planning, patrimonial management, the election process, and university restaurant management.

With the information obtained from the step *Understand the maintenance process and define the activity diagram*, it was created a UML model represented in Figure 14. The user, having a problem in the system to be reported, contacts the service center, which is responsible for opening the ticket. With the ticket opened, the person in charge of the analysis does the initial screening to classify the ticket and direct it to the corresponding sector. For this study, the interest is in the Operations central. Once assorted, the ticket passes through a second screening to be designated between database and maintenance.

Inside maintenance, the ticket will be resolved by the developer available in the team. Once resolved, the tester will perform the validation of the ticket resolution. And finally, the configuration engineer is responsible for deploying the most up-to-date version of the system to production. And the activities classified as being a problem related to the database, will be resolved by the database analyst and closed, not going through all the steps that the ticket classified as maintenance problem needs to pass. The validation of the ticket resolution will be done by the service center, which will contact the user to validate the correction of the problem. For this study, was taken into account only the analysis of the Operations Central (N2) area.

With that, a graphic queue notation was created, represented in Figure 15, based on the steps of the process to be modeled in Petri net, for the sake of visualizing the conversion of the UML model to SPN, it is not a required step. The ticket arrives initially in the server one (*S1*) queue and after *S1* does its service, that is, it analyzes the ticket, the ticket is destined for server two (*S2*) queue, that represents the database area queue; or server three (*S3*) queue, that is the start of maintenance queue, which goes to the

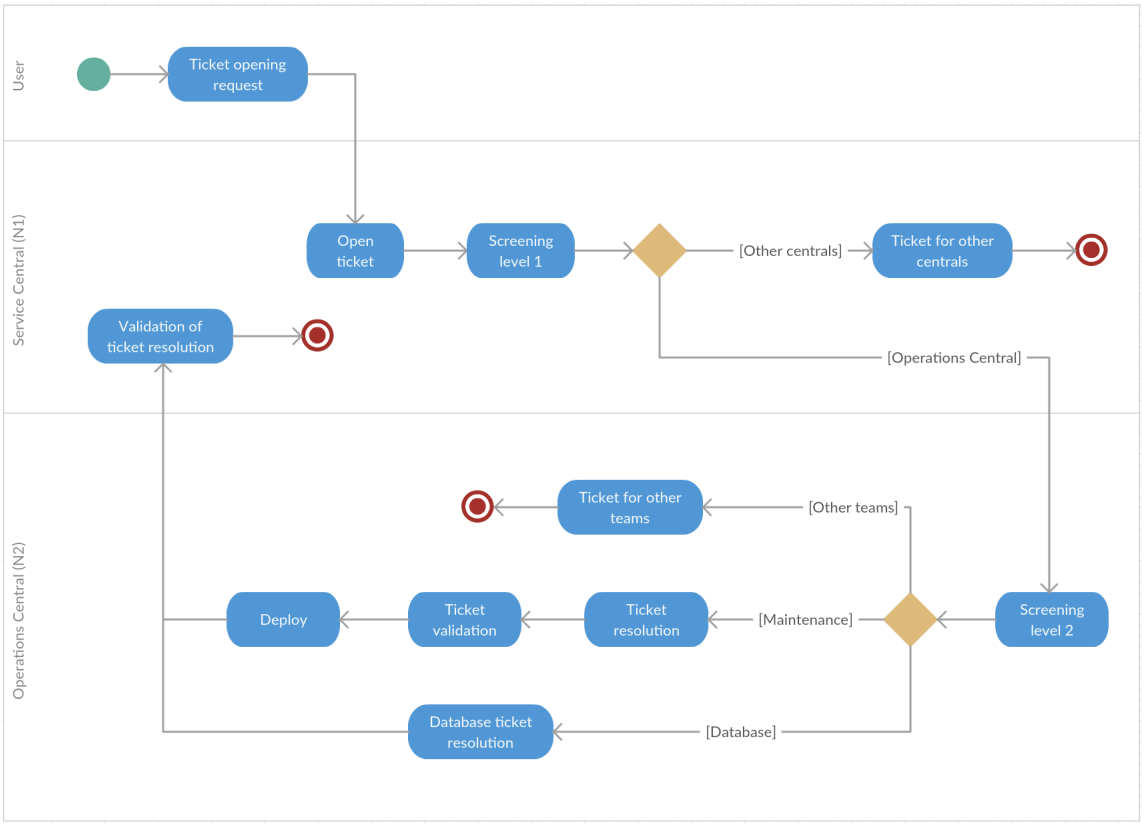

Figure 14 – High level team one process UML model.

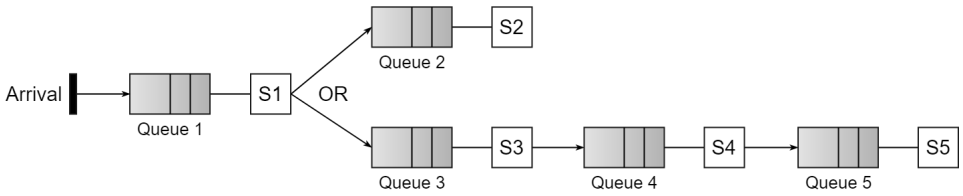

Figure 15 – Queue and server representation.

queues of servers four (*S4*) and five (*S5*) after the respective servers does its service.

As part of the *Generate process model in SPN with parameters and metrics* step, it was constructed the complete SPN model, as shown in the Figure 16, from the base model defined and explained in Chapter 5 and from the UML defined for the process. It was necessary to add transitions that represent the bifurcation between the activities of database and maintenance and ensure that the model is analyzable and simulable, allowing its reuse in other contexts. This work only addresses the simulation by computational limitations.

Explaining the behavior of the defined model, once having at least one token in the places that represents the queue (*Q_Anl*, *Q_DB*, *Q_Dev*, *Q_Tst* and *Q_Prod*) and respective servers available (there are tokens in places *P2*, *P7*, *P10*, *P13* and *P16*), means that the respective immediate transitions (*TI_Start_Anl*, *TI_Start_DB*, *TI_Start_Dev*, *TI_Start_Tst* and *TI_Start_Prod*) are able to start the work.

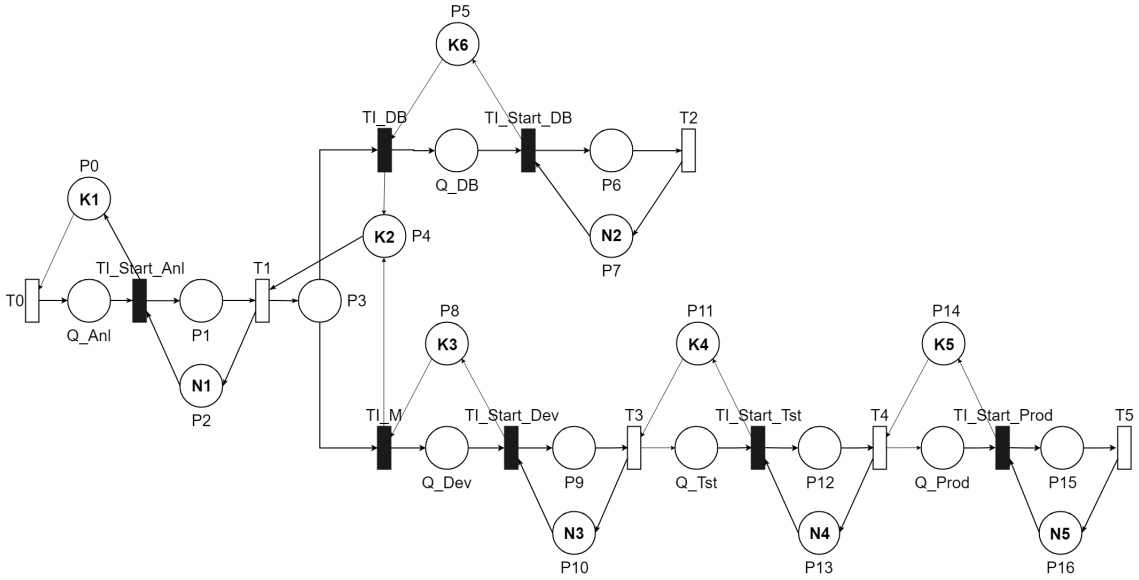

Figure 16 – SPN Process model for team one.

The SPN model has two immediate transitions that have probability associate, the *TI_DB* has *Pdb* associate as a value and the *TI_M* transition has *Pm* associate, representing the probability the token have to go to each queue, that is, the database queue and the maintenance queue respectively.

The parameters *Kn* in places *P0, P4, P5, P8, P11* and *P14* exists to limit the queue and allow the model to be also analyzable, as previously explained, but in the study scenario, the queue is infinite. The value of *Kn* was randomly set to a high value that would not influence the results. The places *P2, P7, P10, P13*, and *P16*, has the number of tokens that represent the number of persons working in each area.

To validate if the model is also analyzable, it is necessary to limit the queues and ensure that the tokens are not lost, that is, that they are attended and return to the queue from which they originated, for this it was used the token game, as defined by the step *Validate model*. It corresponds to the functionality of the Mercury tool in which users can simulate SPN models' behavior. In other words, users are able, for instance, to debug the model that is under analysis. Thus, model construction mistakes can be easily discovered as well as their solution, also, users can simulate failures as well as those correspondents consequences on the system availability.

Token Game starts by highlighting the active transitions, in this case, the *T0* transition, meaning that there is only this transition ready to fire. Each time we click on the highlighted transition the Mercury tool highlights the next active transitions and the tokens change accordingly. Then, by clicking on the *T0* transition, a token of the place *P0* is transferred to the place *Q_Anl*, the variable *K1* has 1 taken from its value and the immediate transition *TI_Start_Anl* is enabled. This means that a ticket has arrived in the analysis queue and is ready for the Analyst to accept this ticket for the screening.

By clicking on the *TI_Start_Anl* enabled transition, the tool enables the next transitions *T0* and *T1*, the token of the place *Q_Anl* is transferred to the place *P1* and the token of the place *P2* meets the place *P1*; therefore, the *N1* variable has 1 taken from its value. This means the Analyst is screening the ticket and at the same time, the queue is open to receive other tickets.

Now, the tool gave two options: if we want to put another ticket on the queue, we can click on the transition *T0*, but if we want to continue with the process flow, we click on the transition *T1*. Clicking on the transition *T1*, immediate transitions *TI_DB* and *TI_M* are enabled, the token that was in place *P1* is returns to place *P2* and the other one follow the flow and is transferred to place *P3*, the token from place *P4* meets the one in place *P3* and the variable *K2* has 1 removed from its value. This means the screening has been done and will now define whether the token will go to the database or maintenance queue and that the Analyst is free to begin work on the next incoming ticket.

Following the flow to the database queue, we need to click on the transition *TI_DB*. After clicking this transition, the token that was in place *P3* returns to the place *P4* adding 1 to the value of the variable *K2*, the same token that was in place *P3* follows the flow to the place *Q_DB* and the token from the place *P5* also goes to the place *Q_DB* removing 1 from the value of the variable *K6* and enables the immediate transition *TI_Start_DB*. This means that the ticket is now in the database Analyst queue and ready to be resolved.

Now the only active transition is the *TI_Start_DB*, so by clicking on this transition, the token in place *Q_DB* will be transferred to place *P6*, the token in place *P7* will meet it in place *P6*, so the value of the *N2* variable will decrease by 1 and the transition *T2* will be enabled. This means that the database Analyst is resolving the ticket. By clicking on the enabled transition *T2*, the token that was put on place *P6* from place *P7* returns to place *P7*, and the token that was following the flow is resolved.

Going back to the step where we decided to follow the database flow, but now following the maintenance flow, by clicking on the immediate transition *TI_M*, the token that was in place *P3* is back to the place *P4* adding 1 to the value of the variable *K2*, the same token that was in place *P3* follows the flow to the place *Q_Dev* and the token from the place P8 also goes to the place *Q_Dev* removing 1 from the value of the *K3* variable and enables the immediate transition *TI_Start_Dev*. This means that the ticket is now in the development queue and ready for the Developers to resolve it.

Now the only active transition is the *TI_Start_Dev*, by clicking this transition, the token in place *Q_Dev* will be transferred to place *P9* adding 1 to the value of the *K3* variable, the token in place *P10* will meet it in place *P9*, so the *N3* variable will decrease 1 from its value and the transition *T3* will be enabled. This means that one Developer is resolving the ticket.

Clicking on the enabled transition *T3*, the token that went to place *P9* from place *P10* returned to place *P10*, and the token that is following the flow goes to the place *Q_Tst*,

the token in place *P11* meets the token in place Q_Tst and the *K4* variable decreases 1 from its value, and the immediate transition *TI_Start_Tst* is enabled. This means that the development is done and the ticket is ready to be tested.

Now the only active transition is the *TI_Start_Tst*, by clicking this transition, the token in place *Q_Tst* will be transferred to place *P12* adding 1 to the value of the *K4* variable, the token in place *P13* will meet it on place *P12*, so the *N4* variable will decrease 1 from its value, and the transition *T4* will be enabled. This means that the Tester is testing the ticket.

By clicking the enabled transition *T4*, the token that went to place *P12* from place *P13* returned to place *P13*, and the token that is following the flow goes to the place *Q_Prod*, the token in place *P14* meets the token in place Q_Prod and the *K5* variable decreases 1 from its value, and the immediate transition *TI_Start_Prod* is enabled. This means that the test is done and the ticket is ready to be deployed.

Now the only active transition is the *TI_Start_Prod*, by clicking on this transition the token in place *Q_Prod* will be transferred to place *P15*, the token in place *P16* will meet it in place *P15*, so the *N5* variable will decrease 1 from its value, and the transition *T5* will be enabled. This means that the Configuration Engineer is deploying the version to production. Clicking on the enabled transition *T5*, the token that went to place *P15* from place *P16* is back to place *P16*, and the token that was following the flow is resolved.

With the entire model validated with Token Game, as part of the *Generate process model in SPN with parameters and metrics* step, with the SPN model developed, we can assemble the expressions for each metric by following the concepts explained in Chapter 5. The metrics were made by area and are described below using the notation adopted by the Mercury tool:

Analysis Area Utilization (AUanl): This expression represents the percentage of the analysis area occupancy level to screen a ticket.

$$AUanl = P\{\#P1 > 0\}$$

Analyst Utilization (IUanl): This expression represents the percentage of an Analyst's individual occupancy level to screen the ticket.

$$IUanl = \frac{E\{\#P1\}}{N1}$$

Database Analysis Area Utilization (AUdb): This expression represents the percentage of the database analysis area occupancy level to resolve a ticket related to database problems.

$$AUdb = P\{\#P6 > 0\}$$

Database Analyst Utilization (IUdb): This expression represents the percentage of a database analyst's individual occupancy level to resolve a ticket related to database problems.

$$IUdb = \frac{E\{\#P6\}}{N2}$$

Development Area Utilization (AUdev): This expression represents the percentage of the development area occupancy level to resolve a ticket.

$$AUdev = P\{\#P9 > 0\}$$

Developer Utilization (IUdev): This expression represents the percentage of a developer's individual occupancy level to resolve a ticket.

$$IUdev = \frac{E\{\#P9\}}{N3}$$

Testing Area Utilization (AUtst): This expression represents the percentage of the testing area occupancy level to validate the resolution of a ticket.

$$AUtst = P\{\#P12 > 0\}$$

Tester Utilization (IUtst): This expression represents the percentage of tester's individual occupancy level to validate the resolution of a ticket.

$$IUtst = \frac{E\{\#P12\}}{N4}$$

Production Area Utilization (AUprod): This expression represents the percentage of the production area occupancy level to deploy the updated version to production.

$$AUprod = P\{\#P15 > 0\}$$

Configuration engineer Utilization (IUprod): This expression represents the percentage of configuration engineer's individual occupancy level to deploy the updated version to production.

$$IUprod = \frac{E\{\#P15\}}{N5}$$

Throughput - Database (TPdb): This expression represents how many tickets per hour the database analyst can resolve.

$$TPdb = \frac{1}{\lambda T1} \times (1 - ((P\{\#Q\_Anl = K1\}) + (P\{\#P1 = N1\}))) \times Pdb$$

Throughput - Maintenance (TPm): This expression represents how many tickets per hour the maintenance team can resolve.

$$TPm = \frac{1}{\lambda T1} \times (1 - ((P\{\#Q\_Anl = K1\}) + (P\{\#P1 = N1\}))) \times Pm$$

Response Time - Database (RTdb): This expression represents how long the database analyst can resolve one ticket.

$$RTdb = \frac{((E\{\#Q\_Anl\}) + (E\{\#P1\}) + (E\{\#P3\}) + (E\{\#Q\_DB\}) + (E\{\#P6\}))}{TPdb}$$

Response Time - Maintenance (RTm): This expression represents how long the maintenance team can resolve one ticket.

$$RTm = ((E\{\#Q\_Anl\}) + (E\{\#P1\}) + (E\{\#P3\}) + (E\{\#Q\_Dev\}) + (E\{\#P9\}) + (E\{\#Q\_Tst\}) + (E\{\#P12\}) + (E\{\#Q\_Prod\}) + (E\{\#P15\}))/TPm$$

With all the steps for the *Process mapping* phase completed, and with the validation made as defined in phase *Model evaluation*, we want to evaluate the model and metrics. The parameters to be used in the SPN model for this step were extracted from the OTRS system that is used by the team to manage the tickets. The data were collected during three months and these, after analysis, were inserted into the model to evaluate the software maintenance process. The step *Evaluate model and metrics* is detailed in Chapter 7, Section 7.1. Model and metrics adjustments were made during this process, as defined by the methodology, but the model presented in this chapter was the final one, with all the adjustments made in step *Adjust model and/or metrics*.

6.2 TEAM TWO PROCESS MAPPING

To validate the application of the model in another context, was take into account a different team process. This is a team that is also part of the Operations Central but is responsible for maintaining another internal system. With the information obtained from the step *Understand the maintenance process and define the activity diagram*, it was created a UML model represented in Figure 17.

The user, having a problem in the system to be reported, contacts the service center, the center is responsible for opening the ticket. With the ticket opened, the person in charge of the analysis does the initial screening to classify the ticket and direct it to the corresponding sector. The interest of this study remains to be in the Operations central (N2). Once assorted, the ticket passes through a second screening to be designated between the internal team and external team queues.

Inside the internal team, the ticket will be resolved by the developer. Once resolved, the configuration engineer is responsible for deploying the most up-to-date version of the system to production. Inside the external team, the ticket is resolved and then the ticket is returned to the internal team to be validated, that is, the internal team tester performs the ticket validation resolved by the external team and then releases to the configuration engineer to deploy the version of the system to production. The validation of the ticket resolution will be done by the service center, which will contact the user to validate the correction of the problem. For this study, was taken into account only the analysis of the Operations Central (N2) area. The external team process was not taken into account in this study.

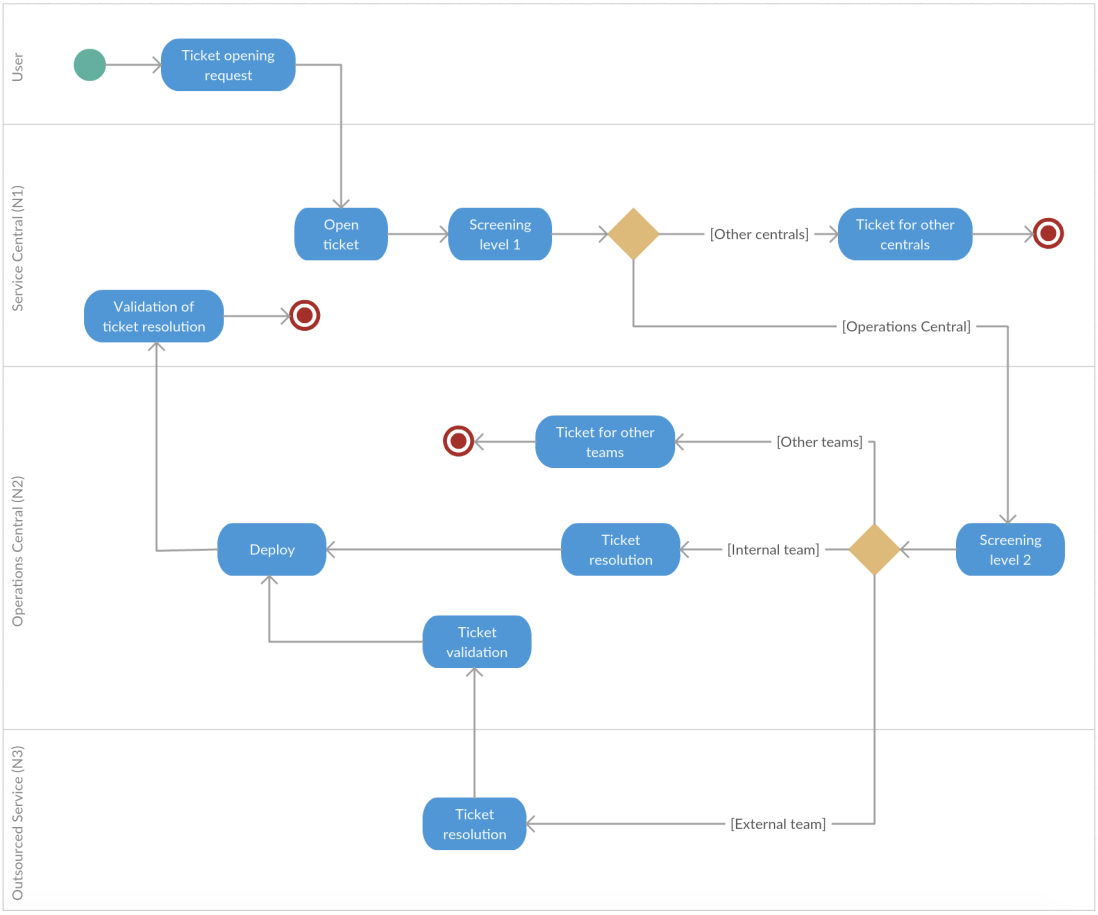

Figure 17 – High level team two process UML model.

As part of the *Generate process model in SPN with parameters and metrics* step, it was constructed the complete SPN model, as shown in the Figure 18, from the base model defined and explained in Chapter 5 and from the UML defined for the process. It was necessary to add transitions that represent the bifurcation between the activities of database and maintenance and ensure that the model is analyzable and simulable, allowing its reuse in other contexts. This work only addresses the simulation by computational limitations.

Explaining the behavior of the defined model for the internal team, once having at least one token in the places that represents the queue (*Q_Anl*, *Q_Exe*, *Q_Val* and *Q_Prod*) and respective servers available (there are tokens in places *P2*, *P7*, *P10* and *P14*), means that the respective immediate transitions (*TI_Start_Anl*, *TI_Start_Exe*, *TI_Start_Val* and *TI_Prod*) are able to start the work.

The SPN model also has two immediate transitions that have probability associate, the *TI_Exe* has *Pexe* associate as a value and the *TI_Val* transition has *Pval* associate, representing the probability the token have to go to each queue, that is, the execution queue and the validation queue respectively.

The parameters *Kn* in places *P0*, *P5*, *P8* and *P11* exists to limit the queue and allow

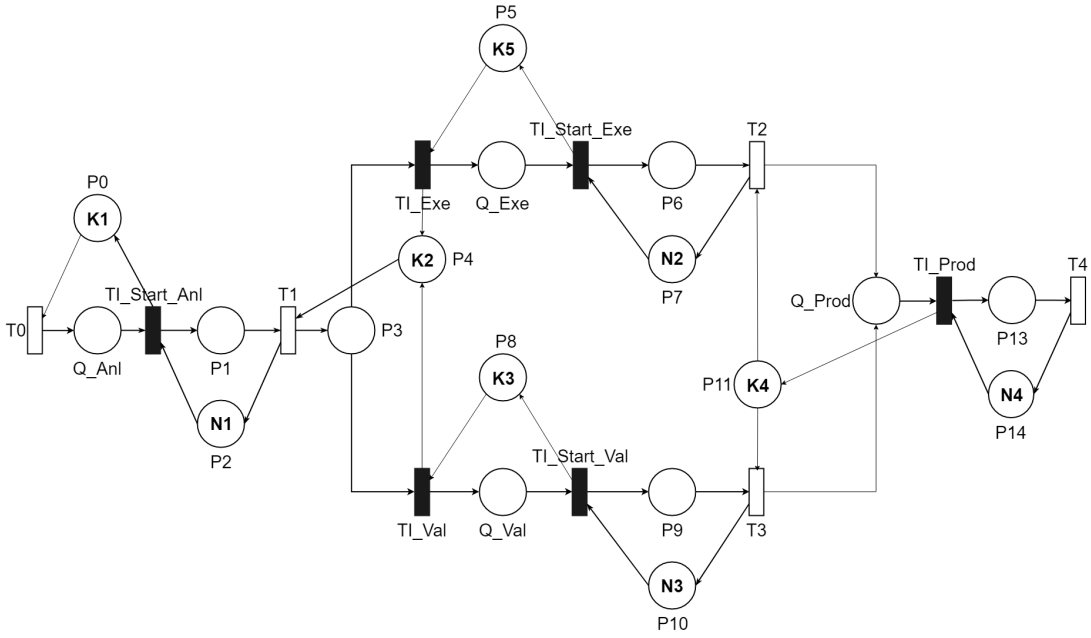

Figure 18 – PN model.

the model to be also analyzable. The value of *Kn* was randomly set to a high value that would not influence in the results. The places *P2*, *P7*, *P10* and *P14*, has the number of tokens *Nn* that represent the quantity of persons working in each area.

To validate if the model is also analyzable, it is necessary to limit the queues and ensure that the tokens are not lost, that is, that they are attended and return to the queue from which they originated, for this it was used the token game, as defined by the step *Validate model*.

Token Game starts by highlighting the active transition *T0*. Then, by clicking the *T0* transition, a token of the place *P0* is transferred to the place *Q_Anl*, the variable *K1* has 1 taken from its value and the immediate transition *TI_Start_Anl* is enabled. This means that a ticket has arrived in the analysis queue and is ready for the Analyst to accept this ticket for the screening.

By clicking the *TI_Start_Anl* enabled transition, the tool enables the next transitions *T0* and *T1*, the token of the place *Q_Anl* is transferred to the place *P1* and the token of the place *P2* meets the place *P1*; therefore, the *N1* variable has 1 taken from its value. This means the Analyst is screening the ticket and at the same time, the queue is open to receive other tickets.

Now, the tool gave two options: if we want to put another ticket on the queue, we can click on the transition *T0*, but if we want to continue with the process flow, we click on the transition *T1*. Clicking on the transition T1, immediate transitions *TI_Exe* and *TI_Val* are enabled, the token that was in place *P1* returns to place *P2* and the variable *N1* has its value back, the other one follow the flow and is transferred to place *P3*, the token from place *P4* meets the one in place *P3* and the variable *K2* has 1 removed from

its value. This means the screening has been done and will now define whether the token will go to the execution or validation queue and that the Analyst is free to begin work on the next incoming ticket.

Following the flow to the execution queue, we need to click on the transition *TI_Exe*. After clicking this transition, the token that was in place *P3* returns to the place *P4* adding 1 to the value of the variable *K2*, the same token that was in place *P3* follows the flow to the place *Q_Exe* and the token from the place *P5* also goes to the place *Q_Exe* removing 1 from the value of the variable *K5* and enables the immediate transition *TI_Start_Exe*. This means that the ticket is now in the execution queue and ready to be resolved.

Now the only active transition is the *TI_Start_Exe*, so by clicking on this transition, the token in place *Q_Exe* will be transferred to place *P6*, the token in place *P7* will meet it in place *P6*, so the value of the *N2* variable will decrease by 1 and the transition *T2* will be enabled. This means that the Developer is resolving the ticket.

By clicking on the enabled transition *T2*, the token that was put on place *P6* from place *P7* returns to place *P7* adding 1 back to the value of the variable *N2*, the token that was following the flow goes to the place *Q_Prod*, the token in the place *P11* meets the on in place *Q_Prod*, the value of the variable *K4* decreases in 1, and the *TI_Prod*transition is enabled. This means that the execution is done and the ticket is ready to be deployed.

Clicking the *TI_Prod* immediate transition, the token in place *Q_Prod* goes to place *P13*, the token from place *P14* meets the one in place *P13*, the value of the variable *N4* decreases in 1, the token that was in place *Q_Prod* from place *P11* goes back to the place *P11* and is added 1 to the value of the variable *K4*. This means that the Configuration Engineer is deploying the version to production. Clicking on the enabled transition *T4*, the token that went to place *P13* from place *P14* is back to place *P14* adding 1 back to the value of the variable *N4*, and the token that was following the flow is resolved.

Going back to the step where we decided to follow the execution flow, but now following the validation flow, by clicking on the immediate transition *TI_Val*, the token that was in place *P3* is back to the place *P4* adding 1 to the value of the variable *K2*, the same token that was in place *P3* follows the flow to the place *Q_Val* and the token from the place P8 also goes to the place *Q_Val* removing 1 from the value of the *K3* variable and enables the immediate transition *TI_Start_Val*. This means that the ticket is now in the validation queue and ready for the Testers to resolve it.

Now the only active transition is the *TI_Start_Val*, by clicking this transition, the token in place *Q_Val* will be transferred to place *P9* adding 1 to the value of the *K3* variable, the token in place *P10* will meet it in place *P9*, so the *N3* variable will decrease 1 from its value and the transition *T3* will be enabled. This means that the Tester is resolving the ticket.

Clicking on the enabled transition *T3*, the token that went to place *P9* from place *P10* returned to place *P10* adding 1 back to the value of the *N3* variable, and the token

that is following the flow goes to the place *Q_Prod*, the token in place *P11* meets the token in place Q_Prod and the *K4* variable decreases 1 from its value, and the immediate transition *TI_Prod* is enabled. This means that the validation is done and the ticket is ready to be deployed.

Clicking the *TI_Prod* immediate transition, the token in place *Q_Prod* goes to place *P13*, the token from place *P14* meets the one in place *P13*, the value of the variable *N4* decreases in 1, the token that was in place *Q_Prod* from place *P11* goes back to the place *P11* and is added 1 to the value of the variable *K4*. This means that the Configuration Engineer is deploying the version to production. Clicking on the enabled transition *T4*, the token that went to place *P13* from place *P14* is back to place *P14* adding 1 back to the value of the variable *N4*, and the token that was following the flow is resolved.

With the entire model validated with Token Game, so as part of the *Generate process model in SPN with parameters and metrics* step, with the SPN model developed, we can assemble the expressions for each metric following the concepts explained in Chapter 5. The metrics were made by area and they are described below using the notation adopted by the Mercury tool:

Analysis Area Utilization (AUanl): This expression represents the percentage of the analysis area occupancy level to screen a ticket.

$$AUanl = P\{\#P1 > 0\}$$

Analyst Utilization (IUanl): This expression represents the percentage of an Analyst's individual occupancy level to screen a ticket.

$$IUanl = \frac{E\{\#P1\}}{N1}$$

Development Area Utilization (AUexe): This expression represents the percentage of the development area occupancy level to resolve a ticket.

$$AUexe = P\{\#P6 > 0\}$$

Developer Utilization (IUexe): This expression represents the percentage of a developer's individual occupancy level to resolve a ticket.

$$IUexe = \frac{E\{\#P6\}}{N2}$$

Testing Area Utilization (AUval): This expression represents the percentage of the testing area occupancy level to validate the resolution of a ticket.

$$AUval = P\{\#P9 > 0\}$$

Tester Utilization (IUval): This expression represents the percentage of tester's individual occupancy level to validate the resolution of a ticket.

$$IUval = \frac{E\{\#P9\}}{N3}$$

Production Area Utilization (AUprod): This expression represents the percentage of the production area occupancy level to deploy the updated version to production.

$$AUprod = P\{\#P13 > 0\}$$

Configuration engineer Utilization (IUprod): This expression represents the percentage of configuration engineer's individual occupancy level to deploy the updated version to production.

$$IUprod = \frac{E\{\#P13\}}{N4}$$

Throughput (TP): This expression represents how many tickets per hour the team can resolve.

$$TP = \frac{1}{\lambda T1} \times (1 - ((P\{\#Q\_Anl = K1\}) + (P\{\#P1 = N1\})))$$

Response Time (RT): This expression represents how long the team can resolve a ticket.

$$\begin{aligned} RT = ((E\{\#Q\_Anl\}) + (E\{\#P1\}) + (E\{\#P3\}) \\ +(E\{\#Q\_Exe\}) + (E\{\#P6\}) \\ +(E\{\#Q\_Val\}) + (E\{\#P9\}) \\ +(E\{\#Q\_Prod\}) + (E\{\#P13\}))/TP \end{aligned}$$

With all the steps for the *Process mapping* phase completed, and with the validation made as defined in phase *Model evaluation*, we want to evaluate the model and metrics. The parameters to be used in the SPN model for this step were extracted from the OTRS system that is used by the team to manage the tickets. The data were collected during three months and these, after analysis, were inserted into the model to evaluate the software maintenance process. The step *Evaluate model and metrics* is detailed in Chapter 7, Section 7.1. Model and metrics adjustments were made during this process, as defined by the methodology, but the model presented in this chapter was the final one, with all the adjustments made in step *Adjust model and/or metrics*.

6.3 FINAL CONSIDERATIONS

This chapter presented the details on the steps related with the *Process mapping* and *Model evaluation* phases for the two process that will be use as input for the *Scenarios evaluation* phase detailed in the next chapter, Chapter 7.

7 SCENARIOS EVALUATION

In this chapter, three studies are presented to evaluate situations of practical interest. Initially, it is made the evaluation using the two processes presented in Chapter 6. After this, a Sensitivity Analysis is made and, finally, the last study shows the results of the simulation using the Sensitivity Analysis data and comparing these results with the results of the first study that used real data.

7.1 REAL CASES PRESENTATION

As part of the step *Evaluate model and metrics*, this section has as objective to validate the SPN model in two different maintenance processes. The first one is showed in the next Subsection 7.1.1 and the second one in the Subsection 7.1.2.

7.1.1 Team One Process

In this stage of the adopted methodology, the data related to system maintenance was collected. The collection consists of measuring events of interest in the system, through historical data and in obtaining information from the team.

Table 5 shows the arrival rate calculated from the data obtained from the OTRS tool for the SIG@ system in three months (January to March).

Table 5 – Ticket per hour in the first quarter of 2017.

Month	Database	Maintenance	Total
Jan	0.125	0.3667	0.4917
Feb	0.1083	0.3583	0.4667
Mar	0.1667	0.4667	0.6333
Average	0.1333	0.3972	0.5305

The first evaluation is based on the context currently experienced by Team One, which is made of ten people and is distributed as follows: an analyst, five developers, a tester, two configuration engineers and a database analyst. The number of people described working in each area will represent the token quantity in the designated place in the SPN model (Figure 14) presented in Chapter 6.

Based on the scenario currently used by the Operations Central, the stationary simulation was made using the average data as parameters described on Tables 5, the number of people described on Table 6, and the delays for each timed transition described on Table 7. The confidence level was set to 95% and the maximum relative error was set

Table 6 – Number of people working in each area of team one.

Place	Variable	Quantity
P2	N1	1
P7	N2	1
P10	N3	5
P13	N4	1
P16	N5	2

Table 7 – Timed transitions parameters.

Transition	Delay	Type
T0	3.44 h	Single Server
T1	0.5 h	Infinite Server
T2	6 h	Infinite Server
T3	8 h	Infinite Server
T4	4 h	Infinite Server
T5	2 h	Infinite Server

to 10%. The probabilities associated with the immediate transitions *TI_DB* and *TI_M* were 0.25 and 0.75, respectively. The results of the simulation are described in Table 8.

Table 8 – Team One simulation results.

Metric	Result	Confidence Interval (95%)
TPdb	0.07267442 ticket/h	[0.07267441860467339, 0.0726744186046778]
TPm	0.21802325 ticket/h	[0.21802325581437196, 0.21802325581438306]
RTdb	12.9086989 h	[12.79764059863279, 13.019757388672756]
RTm	42.2280976 h	[41.83142343547035, 42.62477188754499]
AUan	14.5321688%	[0.14504350176043118, 0.14559987455409884]
AUdb	43.582423%	[0.43339386567724225, 0.43825461275539757]
AUdev	82.7807516%	[0.826945159108653, 0.8286698730381001]
AUtst	87.1441065%	[0.8701775697938032, 0.872704561026328]
AUprod	35.7572249%	[0.35705578513688113, 0.3580887138259479]
IUan	14.5321688%	[0.14504350176043118, 0.14559987455409884]
IUdb	43.5824239%	[0.43339386567724225, 0.43825461275539757]
IUdev	35.8623960%	[0.3577285324166937, 0.35951938783721876]
IUtst	87.1441065%	[0.8701775697938032, 0.872704561026328]
IUprod	21.8428328%	[0.21799654903075885, 0.21886010842601442]

It is possible to see from these results that the use of the configuration engineer is low and that the use of the tester is very high, as well as that of the developers, thus

identifying the process bottleneck, with almost 100% of utilization of the resources, these areas need changes. The database and the analysis area have very low utilization, the possible solution to better take advantage of these resources, since it only has one person per area, would be allocating them also in other activities.

7.1.2 Team Two Process

The second validation is based on the context currently experienced by another team from NTI, where five people are distributed as follows: one analyst, one developer, two testers and one configuration engineer. The number of people described working in each area will represent the token quantity in the designated place in the SPN model (Figure 17) presented in Chapter 6.

For the stationary simulation, the confidence level was set to 95% and the maximum relative error was set to 10%. The probability *Pexe* of getting a ticket to the execution queue is around 33%, so the probability *Pval* of the ticket going to the validation queue is 67%. The other immediate transitions have no associated probability. The other parameters used in this model are detailed in Tables 9 and 10.

Table 9 – Number of people working in each area of team two.

Place	Variable	Quantity
P2	N1	1
P7	N2	1
P10	N3	2
P14	N4	1

Table 10 – Timed transitions parameters.

Transition	Delay	Type
T0	2.2 h	Single Server
T1	10.4 h	Single Server
T2	28 h	Single Server
T3	32 h	Infinite Server
T4	2 h	Single Server

Table 11 shows the results of the metrics obtained by the simulation in the Mercury tool. It is possible to see from these results that the use of the configuration engineer is very low and the use of the analyst and the testers is very high, showing the possible process bottlenecks. In order to validate it is possible to compare the Response Time calculated with the real data reported by the team, that on average every 58 hours deploys a version to production and the calculated was 62 hours, we have a result with a difference of 6.5%,

Table 11 – Team Two process simulation results.

Metric	Result	Confidence Interval (95%)
TP	0.0177 ticket/h	[0.017121, 0.019149]
RT	62.2119 h	[62.192328, 62.627803]
AUan	99.99 %	[0.999964, 1.000007]
AUexe	92.83 %	[0.923290, 0.933378]
AUval	99.51 %	[0.992370, 0.997477]
AUprod	18.89 %	[0.187193, 0.194578]
IUan	99.99 %	[0.999964, 1.000007]
IUexe	92.83 %	[0.923290, 0.933378]
IUval	98.33 %	[0.982370, 0.987477]
IUprod	18.89 %	[0.187193, 0.194578]

thus being considered a valid model for representing the process in question, since the result is lower than the relative error.

7.2 SENSITIVITY ANALYSIS

To evaluate the impact of the parameters in the system and find the ideal quantity of human resources per area, the Sensitivity Analysis was made for each metric. The process used in this analysis was the Team One process, validated in 7.1.1.

It is important to highlight that parameters are ordered according to absolute values of scaled sensitivities. A negative sensitivity indicates that if the parameter value increases, the measure of interest decreases, which means have an inverse impact. A positive sensitivity indicates that an increase in the parameter value causes an increase in the measure of interest.

The scaled Sensitivity Analysis of the database services Throughput, concerning five parameters is shown in Table 12.

Table 12 – Sensitivity indices for database services Throughput metric.

Parameters	Result
*Testers N*4	1.2085
*Analysts N*1	1.1840
*Database Analysts N*2	-0.2796
*Developers N*3	-0.3806
*Configuration Engineers N*5	-0.5194

This decreasing sensitivity ranking shows that the number of testers has the greatest impact among all parameters when the database service Throughput is the measure of

interest, but it's a small impact. This parameter indicates that when the number of testers changes, the database throughput may change more than if the other quantities change. This impact is followed by the number of Analysts, which also shows to have a major impact on the database Throughput metric when compared to the other parameters that have an inverse impact on the result.

The scaled Sensitivity Analysis of the maintenance services Throughput, concerning five parameters is shown in Table 13.

Table 13 – Sensitivity indices for maintenance services Throughput metric.

Roles	Result
Testers $N4$	1.2087
Database Analysts $N2$	1.1985
Analysts $N1$	-0.2427
Developers $N3$	-0.2744
Configuration Engineers $N5$	-0.3705

This decreasing sensitivity ranking shows that the number of testers also has the greatest impact among all parameters when the maintenance service Throughput is the measure of interest, but it's a small impact. This parameter indicates that when the number of testers changes, the maintenance throughput may change more than if the other quantities change. This impact is followed by the number of database Analysts, which also shows to have a major impact on the maintenance Throughput metric when compared to the other parameters that have an inverse impact on the result.

The scaled Sensitivity Analysis of the maintenance services Response Time, concerning five parameters is shown in Table 14.

Table 14 – Sensitivity indices for response time of the maintenance services metric.

Roles	Result
Testers $N4$	98.9158
Database Analysts $N2$	14.3621
Configuration Engineers $N5$	-0.2315
Analysts $N1$	-0.2594
Developers $N3$	-0.2938

This decreasing sensitivity ranking shows that the number of testers also has the greatest impact among all parameters when the maintenance services Response Time is the measure of interest, and it's the greatest impact compared with the other parameters. This parameter indicates that when the number of testers changes, the maintenance Response Time change more than if the other quantities change. This impact is followed

by the number of database Analysts, which also shows to have a major impact on the maintenance Response Time metric when compared to the other parameters that have an inverse impact on the result, but still a small impact compared with the number of testers impact.

The scaled Sensitivity Analysis of the database services Response Time, concerning five parameters is shown in Table 15.

Table 15 – Sensitivity indices for response time of the database services metric.

Roles	Result
Testers $N4$	149.3573
Database analysts $N2$	105.8583
Developers $N3$	-0.0965
Analysts $N1$	-0.3876
Configuration engineers $N5$	-0.4288

This decreasing sensitivity ranking shows that the number of testers also has the greatest impact among all parameters when the database services Response Time is the measure of interest, and it's the greatest impact compared with the other parameters. This parameter indicates that when the number of testers changes, the database Response Time change more than if the other quantities change. This impact is followed by the number of database Analysts, which also shows to have a great impact on the database Response Time metric when compared to the other parameters that have an inverse impact on the result.

The rankings presented showed that the number of testers is the most important among all parameters when the Throughput or Response Time is the measure of interest.

The results of the Sensitivity Analysis related with the Utilization metrics are detailed in the Figures 20, 21, 22 and 23 for the number of Analysts, database Analysts, Developers, Testers and Configuration Engineers respectively.

It is important to point out that these results showed in Figures 19 to 29 are point estimates, that is, mean values and not a confidence interval, and the stopping criteria used were the confidence level set to 95% and the maximum relative error set to 10%.

It is important to stress these parameters to see which one has the highest influence on the performance metrics. Our Sensitivity Analysis helped to see that the Testers have the highest influence, and the second one for the most metrics are the database Analysts.

The Sensibility Analysis was made in addition to the number of persons in each area, also for the parameters $\lambda T0$ varying from 0.5 to 4 hours and transitions TI_DB and TI_M weight as well, varying from 0.1 to 0.9 each.

The variation of the arrival rate for the metrics TPdb, RTdb, TPm and RTm are shown in the Figures 24, 26, 25 and 27 respectively.

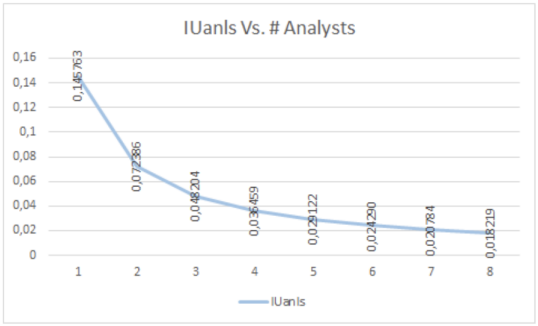

Figure 19 – Individual Analyst Utilization vs. Number of Analysts chart.

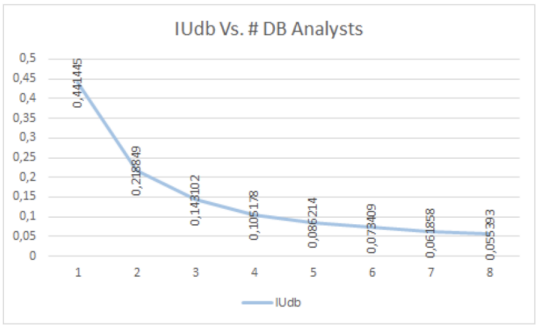

Figure 20 – Database Analyst Utilization vs. Number of database Analysts chart.

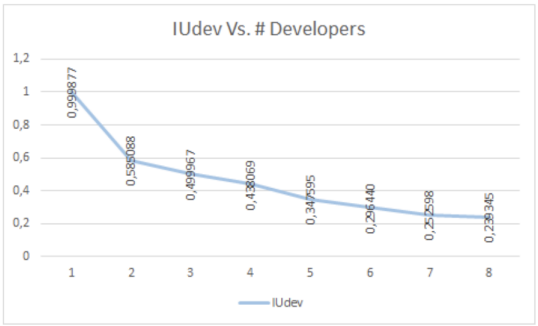

Figure 21 – Developer Utilization vs. Number of Developers chart.

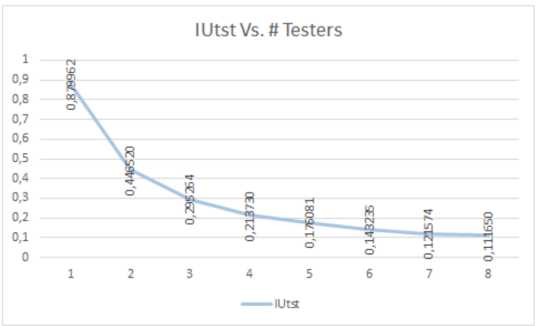

Figure 22 – Tester Utilization vs. Number of Testers chart.

With an arrival delay of 0.5 hours, we get the highest result for the TPdb and TPm metrics. From the charts, you can see that with 1 hour delay or more, these metrics start

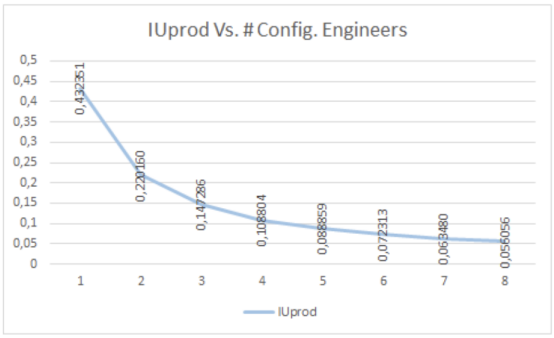

Figure 23 – Configuration Engineer Utilization vs. Number of Configuration Engineers chart.

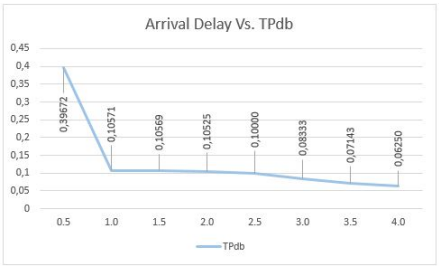

Figure 24 – Throughput - Database vs. Arrival delay chart.

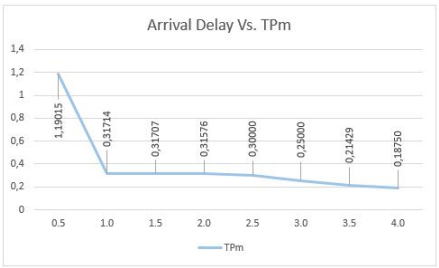

Figure 25 – Throughput - Maintenance vs. Arrival delay chart.

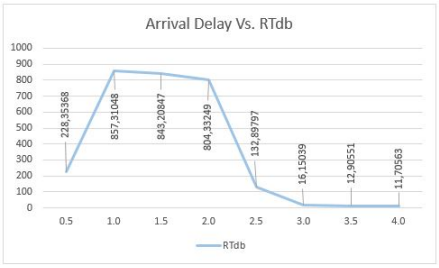

Figure 26 – Response Time - Database vs. Arrival delay chart.

to get lower results, which means the team will take longer to deliver tickets. For RTdb and RTm metrics, you can get the highest result with a task arrival delay between 1 hour and 2 hours.

For the TPdb, TPm, RTdb, and RTm with the transitions *TI_DB* and *TI_M* weight from 0.1 to 0.9, the results do not have many variations. For illustration, following on the

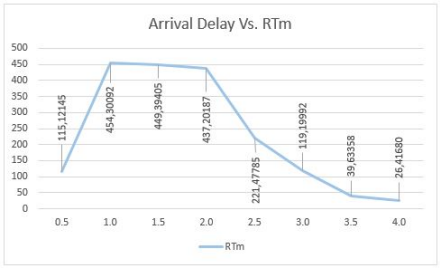

Figure 27 – Response Time - Maintenance vs. Arrival delay chart.

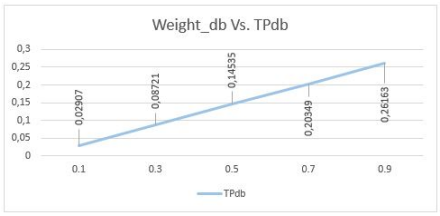

Figure 28 – Throughput - Database vs. Weight of transition - Database chart.

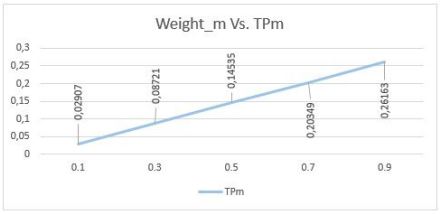

Figure 29 – Throughput - Maintenance vs. Weight of transition - Database chart.

Figures 28 and 29, are the charts for TPdb and TPm results, respectively.

Working with integers numbers, since we are talking about the number of people, the number of persons working in each area of the process can be as follows: one Analyst, two database Analysts, four Developers, two Testers, and one Configuration Engineers. With this configuration in the team, we may improve each aspect of the performance of the Team One process of the Operations Central of Federal University of Pernambuco (UFPE), and the next study detailed in Section 7.3 will evaluate the model with these parameters to validate this assumption.

As the importance of this study, numerical factors were presented that impacted on process metrics and, consequently, impacted on decision making. It is important to emphasize that the presented study does not compose an in-depth study, so this study aimed to highlight only the superficial relationship, which future works may deepen since the pure look of the metric does not necessarily represent the best solution for the process improvement. The main results of the Sensitivity Analysis had the goal to serve as input for the next study.

7.3 SIMULATION OF THE SENSITIVITY ANALYSIS SCENARIO

As the last study, the data set obtained in the Sensitivity Analysis study was used to replace the number of people parameters in each area of the Team One process. The step *Execute model simulation* was done with these new parameters to validate the scenario and a comparison was made between these results, and the results from the first study, using the real data.

According to the Sensitivity Analysis, one possible scenario would be to have a team with 1 Analyst, 2 database Analysts, 4 Developers, 2 Testers, and 2 Configuration Engineers. These data were used to replace respectively the number of tokens *Nn* in the places *P2, P7, P10, P13* and *P16* of the model represented in the Figure 16 for the simulation. These parameters are shown in Table 16. The confidence level was set to 95% and the maximum relative error was set to 10%.

Table 16 – Number of people working in each area of the process.

Place	Variable	Quantity
P2	N1	1
P7	N2	2
P10	N3	4
P13	N4	2
P16	N5	1

The results obtained in this simulation are detailed in Table 17. As shown in the table, it was observed that the maintenance Response Time had the greatest improvement compared to the simulation with real data, changing from 42.2 hours to 16 hours.

This proved to corroborate with the Sensitivity Analysis result that for the maintenance Response Time, the number of Testers had a much greater impact on the metric than compared to the other parameters, followed by the number of database Analysts, which was also changed.

Comparing the data obtained in the simulation using the real data of the Team One with the data obtained in this scenario, it is possible to see in Figure 30 that we had great improvements especially in the Response Time metrics. The database Response Time decreased from 12.9 hours to 8.4 hours, and the maintenance Response Time decreased from 42.2 hours to 16 hours.

We have achieved many improvements in the metrics evaluated. These improvements are shown in detail in Table 18. Throughput metrics were the least impacted, with the biggest improvement of 0.01 %. In Response Time metrics, we can see big improvements, decreasing by 61.99 % and 34.7 % for maintenance and database, respectively. Analyzing utilization metrics is complicated because what we are evaluating is human resources and not computational systems, so we don't want team members to be overwhelmed and we

Table 17 – Simulation result with data obtained with Sensitivity Analysis.

Metric	Result	Confidence Interval (95%)
TPdb	0.07267441 ticket/h	[0.07267441860467157, 0.0726744186046863]
TPm	0.21802325 h	[0.21802325581433146, 0.21802325581440574]
RTdb	8.428818 h	[8.197279482401399, 8.660357523408436]
RTm	16.048574 h	[15.92003631580553, 16.177112393313426]
AUan	14.715205%	[0.14658249026154863, 0.14772161034051712]
AUdb	34.690879%	[0.3386973072770197, 0.3551202825815622]
AUdev	82.242852%	[0.8218860897280867, 0.8229709525234329]
AUtst	61.437301 %	[0.6083283022843549, 0.6204177321509128]
AUprod	35.860994 %	[0.3532588042272103, 0.3639610786758422]
IUan	14.715205%	[0.14658249026154863, 0.14772161034051712]
IUdb	21.108947%	[0.20452786138326637, 0.21765108083330423]
IUdev	43.43442%	[0.42853122063003174, 0.44015728906725077]
IUtst	44.449616 %	[0.4424599497125482, 0.44653238705065346]
IUprod	43.773480 %	[0.43202754417194483, 0.44344207461932045]

Figure 30 – Comparative charts of simulation results using real data and Sensitivity Analysis data.

don't want also them to be underused, so it's up to the evaluator to determine if the results are good or not. But for the purposes of this study, we can consider that the results obtained, even if the individual Utilization is less than 50%, are good results, since we obtained improvements in the Throughput and Response Time metrics, which directly impact the users.

In this scenario, we only made the simulation with the data obtained in the Sensitivity

Table 18 – Comparison of simulation results using real data and Sensitivity Analysis data.

Metric	Real data Simulation	Sensitivity Analysis data Simulation	Percentage
TPdb	0.07267442 ticket/h	0.07267471 ticket/h	0.00039904%
TPm	0.21802325 ticket/h	0.21804597 ticket/h	0.0104209%
RTdb	12.9086989 h	8.428818 h	-34.70436%
RTm	42.2280976 h	16.048574 h	-61.9955%
AUan	14.53217 %	14.71521 %	1.25955%
AUdb	43.58242 %	34.69088 %	-20.40167%
AUdev	82.78075 %	82.24285 %	-0.649789%
AUtst	87.14411 %	61.43730 %	-29.4992%
AUprod	35.75722 %	35.86099 %	0.290207%
IUan	14.5321688 %	14.715205 %	1.259524%
IUdb	43.5824239 %	21.10894 %	-51.5655%
IUdev	35.862396 %	43.43442 %	21.1141%
IUtst	87.1441065 %	44.449616 %	-48.993%
IUprod	21.842833 %	43.77348 %	100.402%

Analysis, but it is possible to test several scenarios, changing the parameters as desired with no considerable costs for the company. It is worth pointing out that, if this solution is implemented by the team, they can improve their service and balance the load as saw in the results, and consequently, will also be able to improve the quality of the team's work environment, since team members are not overwhelmed with activities.

The solution provided by Sensitivity Analysis may not be the best option for the company because the cost of putting more people on the project may not be feasible. This is why simulating different scenarios is important. Sensitivity Analysis should be done to obtain guidelines for the choice of parameters. Several simulations can be done so that the balance between cost and benefit can be found. This step is detailed in the proposed method step *Setting scenario and change parameters*.

The studies presented in this chapter used in their execution the guidelines described in the methodology proposed in Chapter 4, thus revealing the applicability of this methodology to obtain comparative simulation results, with the predictability of maximizing metrics results if the changes tested in the simulations are applied to the process.

7.4 FINAL CONSIDERATIONS

This chapter presented three studies that showed the usefulness and efficacy of the methodology proposed. The first study evaluated the models from two different teams. These evaluations were made by comparing the simulation result with the data collected with the team. The second study was an investigation, carried out through the Sensitiv-

ity Analysis, to find the ideal number of people in each area where the best results of the analyzed metrics would be obtained. The last study showed a simulation with the numbers obtained with the Sensitivity Analysis, and its results showed improvements in Throughput, Response time and Utilization metrics.

8 CONCLUSIONS

In a software maintenance context, the response time factor for request delivery time is very important. Users of the system in maintenance are directly impacted by the problems encountered in the production environment and the shorter the response time the better for them. It is needed to calculate some metrics to analyze the maintenance process and then improve this service for the users.

Analytical and simulation models are useful for planning and predicting systems behaviors. This dissertation showed that simulation models are important and effective for the analysis of processes. Although the creation and analysis of performance models for maintenance processes is a challenge that requires the combination of some skills for reaching significant results, the proposed methodology brings in a simplified way to make this analysis and to create these models.

This dissertation addressed the fundamental factors for a team's process evaluation, providing guidance on how to perform these measures that relate to the process performance that also impacts users. As one of the contributions, the model presented allows mathematical expressions to calculate the probable results of the metrics. The studies presented showed that the evaluation of throughput, response time and utilization in different scenarios is possible through simulations and their results provide the equivalent response to process improvement if these simulation changes are implemented in the real context of the team.

This research achieved many results in the areas that it explored, being the main contribution the proposal of a methodology that seeks to guide the stakeholders to detect improvement points or bottlenecks in the processes of their teams based on the factors that are most important to them through the analysis of the response time, the throughput and the utilization of software maintenance teams and individuals metrics, and at the same time, it is possible to improve the waiting time of the users. The methodology can also be applied in an integrated manner with the sensitivity analysis to help create different scenarios for evaluation. We found that by using simulation and sensitivity analysis together to find opportunities for process improvements we were able to achieve good improvements in metric results.

The proposed methodology was tested during the process evaluation of different teams. In this dissertation, three studies focused on two maintenance teams were presented. The first study presented validations of the SPN models of the process of each team. These validations were made by comparing the simulation result with the historical data. The second study showed a sensitivity analysis where it was possible to obtain the parameters that most impact the metrics results and use them to generate the scenarios to be evaluated. The result of the second study was used as a parameter for the last study, where

simulation results were presented with a different team distribution than the existing one. The contributions of this study are summarized below.

8.1 CONTRIBUTIONS

This study presented a methodology to evaluate software maintenance processes, as a way of trying to validate assumptions of process improvements to be evaluated through simulations. As seen in Chapter 3, none of the studies found while researching related works brought this approach. This methodology consists of a series of steps that are divided into three phases: process mapping, model evaluation, and scenarios evaluation. Ten different steps are presented as the basis of the proposal: understand the maintenance model and define the activity diagram, generate SPN model with parameters and metrics, validate model, evaluate model and metrics, adjust model and/or metrics, setting scenario and change parameters, execute model simulation, execute Sensitivity Analysis, analyze the results, and finally present results and recommendations. This study showed that with the application of the proposed methodology, it is possible to evaluate the impact of changes in the process without the cost of implementing each possibility of improvement, to evaluate its performance, to carry out simulations to obtain more accurate estimates and to assist in team planning.

One of the steps of the proposed methodology is *Generate SPN model with parameters and metrics*. As part of this step, another contribution presented was the SPN model of the service of a team area was proposed and represents the arrival of a ticket and its resolution by this area to be used as a basis to model processes. Also, the translation of activities diagrams to Stochastic Petri Nets was detailed. How to map activities, transitions, initial and final states, and decisions to an SPN model were shown. From this basic knowledge provided by this study, it is possible to create the model of a software maintenance process, and then, with the model built it is possible to use the parameters collected in the step *Understand the maintenance model and define activity diagram* to make the simulation and evaluate the process.

In summary, the main contributions of this dissertation are:

- The proposition of a methodology to support the performance evaluation of software maintenance processes;
- The mapping of UML activity diagrams that represent the process to SPN models, was through these models it is possible to measure performance metrics;
- Proposition of a resource base SPN model to represent an area of the process, that ease the transition from UMLs to SPN models.

8.2 DIFFICULTIES AND LIMITATIONS

The difficulties encountered in the development of this project were as follows:

- The first difficulty found was to find works related to performance evaluation of software maintenance processes;
- Another difficulty was to obtaining the historical data of the team process, since they were in process of change in the way of registering this data, because in the real projects the teams do not record the information at the exact moment and the time of the ticket in each queue and the main response time had to be complemented through the feeling of the team members;
- The last one was the effort of the validation phase, several adjustments were necessary for the model to make it also analyzable and not only simulable, as well as adjustments so that the results obtained in the simulations were close to the real and so we could consider that the model is acceptable.

This dissertation has some limitations:

- The main limitation is that process models in the most distinct scenarios tend to be large. In this way, the model is not evaluated analytically and begins to be solved only by simulation, due to the explosion of states;
- Another limitation is the models and techniques used in this dissertation. The user must have the basic knowledge for the development of the model, the parameterization of its variables and its metrics.

8.3 FUTURE WORKS

Although this dissertation has achieved results and covered some points related to the improvement of the software maintenance process, there are some possibilities to extend the current work. As said before, the knowledge about formal models for performance evaluation is not widespread among IT organizations, so the implementation of a tool that automatically converts UML, or similar modeling languages that are well known by software engineers, to SPN models would be helpful and will allow a complete abstraction regarding the use of SPN. With this future work, the limitation about the need of prior knowledge on the part of those involved for the development of the model is remedied.

Another study that could be integrated into the proposed methodology in future works is to perform case studies that involve other software processes besides maintenance and validate the application of the proposed methodology in other contexts. In the same line of thought, extended this work to cover other metrics of process performance, in addition

to utilization, response time and throughput. With these add-ons, the work would cover more contexts and help more teams to follow continuous improvement.

One thing that this study does not consider is the times when someone on the team is unavailable to complete the service, whether for vacation, illness, etc. An interesting future work would be to include in the analysis the calculation of the Mean Time Of Unavailability (MTOU) to provide even more accurate results.

REFERENCES

ABDALLAH, H.; HAMZA, M. *On the sensitivity analysis of the expected accumulated reward*. [S.l.]: Performance Evaluation, 47(2):163–179, 2002.

AN Y., W. N. Z. X. L. X.; CHEN, P. Hierarchical colored petri nets for modeling and analysis of transit signal priority control systems. 2017.

APRIL A., H. J. A. A.; DUMKE, R. Software Maintenance Maturity Model (SMmm): the software maintenance process model. *Journal of Software Maintenance and Evolution: Research and Practice*, 2005.

BALBO, G. *Lectures on Formal Methods and Performance Analysis: Introduction to Stochastic Petri Nets*. 1. ed. [S.l.]: Springer, 2000. 84-155 p.

BECK, K. e. a. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://agilemanifesto.org/>.

BELL, S.; ORZEN, M. *Lean IT: Enabling and sustaining your lean transformation*. 1. ed. [S.l.]: CRC Press, 2012.

BI J., Y. H. T. W. Z. M. F. Y. Z. J.; LI, J. Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center. 2017.

BLAKE J. T., R. A. L.; TRIVEDI, K. S. *Sensitivity analysis of reliability and performability measures for multiprocessor systems*. [S.l.]: Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems, pages 177–186, New York, NY, USA. ACM, 1988.

BOLCH G., G. S. d. M. H.; TRIVEDI, K. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. 2. ed. [S.l.]: John Wiley & Sons, 2006. ISBN 978-0-471-56525-3.

BOOCH, J. R. e. I. J. G. *UML User Guide*. [S.l.]: Addison-Wesley Longman, 1998.

BOWER, J.; HOUT, T. *Fast cycle capability for competitive powers*. [S.l.]: Harvard Business Review, 1988. 110-118 p.

CAMPOLONGO F.; TARANTOLA, S. S. A. Tackling quantitatively large dimensionality problems. *Computer physics communications*, v. 117, p. 75–85, 1999.

CAR, Z.; MIKAC, B. A method for modeling and evaluating software maintenance process performances. *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, 2002.

CHAPIN N., H. J. K. K. M. R. J.; TAN, W. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, v. 13, p. 3–30, 2001.

CHEN M., Q. X. X. W. W. L. Z. J.; LI, X. Uml activity diagram-based automatic test case generation for java programs. *The Computer Journal Advance Access*, 2007.

CHRISTOPHER, M. *Logistics and Supply Chain Management*. [S.l.]: FT Press, 1992. ISBN 0273731122.

COENEN, F.; BENCH-CAPON, T. *Maintenance of Knowledge-Based Systems: Theory, Techniques and Tools*. [S.l.]: Hartnolls Ltd, 1993.

DIAS, M. Um modelo de avaliação de desempenho para suporte ao planejamento do processo de mudança de software. 2010. Disponível em: <https://repositorio.ufpe.br/handle/123456789/2319>.

ERDIL K., F. E. K. K. M. J. P. S.; YOON, D. Software maintenance as part of the software life cycle. 2003. Disponível em: <http://hepguru.com/maintenance/Final_121603_v6.pdf>.

FADAHUNSI, O.; SATHIYANARAYANAN, M. Visualizing and analyzing dynamic business process using Petri Nets. *2nd International Conference on Contemporary Computing and Informatics (ic3i)*, 2016.

FEBBRARO A., G. D. D.; SACCO, N. A Deterministic and Stochastic Petri Net Model for Traffic-Responsive Signaling Control in Urban Areas. *IEEE Transactions on Intelligent Transportation Systems*, v. 17, n. 2, 2016.

GERMAN, R. *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. [S.l.]: John Wiley & Sons, 2000. ISBN 0471492582.

GIRAULT, C.; VALK, R. *Petri nets for systems engineering: a guide to modeling, verification, and applications*. [S.l.]: Springer, 2003. ISBN 9783662053249.

GROUP, M. R. *Mercury Tool Manual v4.6.5*. Disponível em: <http://www.modcs.org/wp-content/uploads/tools/Mercury_Tool_Manual_v4.6.5.pdf>.

HAMBY, D. M. *A review of techniques for parameter sensitivity analysis of environmental models*. [S.l.]: Environmental Monitoring and Assessment, pages 135–154, 1994.

HOFFMAN F.; GARDNER, R. *Evaluation of Uncertainties in Environmental Radiological Assessment Models*. [S.l.]: In: TILL, J.; MEYER, H. (Ed.). Radiological Assessments: a textbook on environmental dose assessment, 1983.

JACOBSON I., B. G.; RUMBAUGH, J. *The Unified Software Development Process*. 1. ed. [S.l.]: Addison-Wesley, 1999.

JAIN, R. *The art of computer system performance analysis: techniques for experimental design, measurement, simulation and modeling*. 1. ed. [S.l.]: John Wiley & Sons, 1991.

JIN Y., Z. H. Y. H. Z. S.; GE, J. An approach to locating delayed activities in software processes. *International Journal of Automation and Computing*, 2017.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. [S.l.]: Technical Report – Department of Computer Science, University of Durham, 2007.

KLEINROCK, L. *Queueing Systems*. [S.l.]: Wiley-Interscience New York, 1975. ISBN 0471491101.

KURTEL, K. Measuring and monitoring software maintenance services: An industrial experience. *Joint Conference of the 23nd International Workshop on Software Measurement (IWSM) and the Eighth International Conference on Software Process and Product Measurement (Mensura)*, 2013.

LARMAN, C. *Utilizando UML e padrões*. [S.l.]: Livraria Tempo Real Inform, 2006. ISBN 8560031529.

LILJA, D. *Measuring computer performance: a practitioner's guide*. [S.l.]: Cambridge university press, 2005. ISBN 978-0521646703.

LIMA-JUNIOR, F.; CARPINETTI, L. Quantitative models for supply chain performance evaluation: A literature review. 2017.

MACIEL, P. e. a. *Performance and dependability in service computing: Concepts, techniques and research directions*. [S.l.]: Information Science Reference (Premier Reference Source), 2011. 52-97 p. ISBN 1609607945.

MARSAN M.A., B. G. C. G. D. S.; FRANCESCHINIS, G. *Modelling with Generalized Stochastic Petri Nets*. [S.l.]: John Wiley & Sons, 1995. ISBN 0471930598.

MARTIN, J.; MCCLURE, C. *Software Maintenance: The Problem and Its Solutions*. [S.l.]: Prentice Hall, 1983.

MASON-JONES, R.; TOWILL, D. Enlightening supplies. IET, p. 156–160, 1997.

MATOS R., A. J. O. D. M. P. T. K. Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory*, 2015.

MENASCE D.A., D. L.; ALMEIDA, V. *Performance by design: computer capacity planning by example*. [S.l.]: Prentice Hall Professional, 2004. ISBN 978-0130906731.

MURATA, T. Petri Nets: properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, 1989.

NIESSINK, F.; VLIET, H. V. Software maintenance from a service perspective. *Journal of Software Maintenance and Evolution: Research and Practice*, v. 12, p. 103–120, 2000.

PENDER, T. *UML Bible*. [S.l.]: John Wiley & Sons, 2003. ISBN 0764526049.

PEREIRA, L. *Análise e Modelagem de Sistemas com a UML*. 1. ed. [S.l.]: EdiÇão do Autor, 2011.

PIANOSI F., B. k. F. J. W. J. R. J. B. D. W. T. Sensitivity analysis of environmental models: a systematic review with practical workflow. *Environmental Modelling & Software*, v. 79, p. 214–232, 2016.

PRESSMAN, R. *Software Engineering: a practitioner's approach*. 7. ed. [S.l.]: McGraw-Hill Higher Education, 2009.

RACHDI A., E.-N. A.; DAHCHOUR, M. Liveness and reachability analysis of bpmn process models. *CIT. Journal of Computing and Information Technology*, v. 24, n. 2, p. 195–207, 2016.

RAMAMOORTHY, C.; HO, G. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Transactions on Software Engineering*, 1980.

REISIG, W. *Petri Nets - An Introduction*. [S.l.]: Springer Verlag, 1992. ISBN 978-0387137230.

SAPNA, P.; MOHANTY, H. Automated scenario generation based on uml activity diagrams. *International Conference on Information Technology*, 2008.

SETH D., S.-N.; DHARIWAL, P. Application of value stream mapping (vsm) for lean and cycle time reduction in complex production environments: a case study. 2017.

SIEGEL, J. *Introduction to OMG's Unified Modeling Language*. 2019. Disponível em: <http://www.uml.org/>.

SILVA B., M.-R. C. G. F. J. O. D. F. J. D. J. L. A. A. V.; MACIEL, P. Mercury: An integrated environment for performance and dependability evaluation of general systems. *IEEE 45th Dependable Systems and Networks Conference (DSN-2015)*, 2015.

SILVA F.A., K.-S. R. M. O. D. M. T. M. A.; MACIEL, P. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, 2017.

SILVA, G. Modelo construtivista de avaliação de desempenho do processo de planejamento de compras no poder judiciário do estado de santa catarina. 2017.

SLACK N., C.-S. H. C. H. A.; JOHNSTON, R. *Operations Management*. [S.l.]: FT Prentice Hal, 1995. ISBN 0273603167.

SMITH, C. Software performance engineering. performance evaluation of computer and communication systems. *IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation*, p. 509–536, 1993.

SOMMERVILLE, I. *Software Engineering*. 9. ed. [S.l.]: Pearson, 2011.

TAKANG A.A.AND GRUBB, P. *Software Maintenance: Concepts and Practic*. 2. ed. [S.l.]: World Scientific, Singapore, 2003.

TRACY, B. *Eat That Frog*. New York: Berrett-Koehler Publishers, 2017. ISBN 978-1626569416.

TRIVEDI, K. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. [S.l.]: John Wiley & Sons, 2001. ISBN 978-0-471-33341-8.

VASAVA, P. Optimizing the process flow in heat treatment plant through value stream mapping via simulation. 2017.

VLIET, H. V. *Software Engineering: Principles and Practices*. 2. ed. [S.l.]: John Wiley & Sons, 2000.

APPENDIX A – SYSTEMATIC REVIEW RESULTS

Table 19 – Search results in digital libraries.

Step 1	Step 2	#	Library	Title	Autor(s)
[Incl]	[Incl]	1	Google Scholar	Mobile Cloud Performance Evaluation Using Stochastic Models	Francisco Airton Silva, Sokol Kosta, Matheus Rodrigues, Danilo Oliveira, Teresa Maciel, Alessandro Mei and Paulo Martins Maciel
[Incl]	[Excl]	2	Google Scholar	Quantitative models for supply chain performance evaluation: A literature review	Francisco Rodrigues Lima-Junior and Luiz Cesar Ribeiro Carpinetti
[Excl]		3	Google Scholar	Designing software for operational decision support through coloured Petri nets	F.M. Maggi and M. Westergaard
[Excl]		4	Google Scholar	The International Journal of Production Research at 55: a content-driven review and analysis	Jian (Jeff) Guan, Andrew S. Manikas and Lynn H. Boyd
[Excl]		5	Google Scholar	Quality Assessment in DevOps: Automated Analysis of a Tax Fraud Detection System	Diego Perez-Palacin, Youssef Ridene and José Merseguer
[Excl]		6	Google Scholar	Modelling and analysis of sustainable manufacturing system using a digraph-based approach	Vimal K. E. K., Vinodh S. and Anand Gurumurthy

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Incl]	[Incl]	7	Google Scholar	Hierarchical Colored Petri Nets for Modeling and Analysis of Transit Signal Priority Control Systems	Yisheng An, Naiqi Wu, Xiangmo Zhao, Xuan Li and Pei Chen
[Excl]		8	Google Scholar	Application of Lean and JIT principles in supply chain management	Chandan Deep Singh, Rajdeep Singh, Jaskanwal Singh Mand and Sukhvir Singh
[Excl]		9	Google Scholar	Model-Based Systems Engineering: Motivation, Current Status, and Needed Advances	Azad M. Madn and Michael Sievers
[Excl]		10	Google Scholar	Cloud manufacturing system for sheet metal processing	Petri Helo and Yuqiuge Hao
[Excl]		11	Google Scholar	Data envelopment analysis with multiple modes of functioning. Application to reconfigurable manufacturing systems	Sebastián Lozano, Gabriel Villa ORCID Icon and Ignacio Eguía
[Excl]		12	Google Scholar	Process models in design and development	David C. Wynn and P. John Clarkson
[Excl]		13	Google Scholar	Self-Aware Resource Management in Virtualized Data Centers	Simon Spinner
[Excl]		14	Google Scholar	Strategies and Techniques for Quality and Flexibility	Miryam Barad
[Excl]		15	Google Scholar	Development of sustainable platform for modular product family: a case study	Ahm Shamsuzzoha and Petri Helo

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		16	Google Scholar	Infrastructure Resilience Assessment, Management and Governance – State and Perspectives	Hans R. Heinimann and Kirk Hatfield
[Excl]		17	Google Scholar	Performance evaluation of holonic control of a switch arrival system	Carlos Indriago, Olivier Cardin, Odile Bellenguez-Morineau, Naly Rakoto, Pierre Castagna and Edgar Chacòn
[Excl]		18	Google Scholar	Safety and Reliability. Theory and Applications	Marko Cepin and Radim Bris
[Incl]	[Excl]	19	Google Scholar	Optimizing the process flow in heat treatment plant through value stream mapping via simulation: A case study at Volvo Group Trucks Operations	Pratikchandra Vasava
[Excl]		20	Google Scholar	Development and analysis of system and human architectures for critical infrastructure vulnerability assessment	Johnathon D. Huff
[Excl]		21	Google Scholar	A Context-Driven Framework for Proactive Decision Support With Applications	Manisha Mishra, David Sidoti, Gopi Vinod Avvari, Pujitha Mannaru, Diego Fernando Martínez Ayala, Krishna R. Pattipati and David L. Kleinman

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		22	Google Scholar	BME VIK Annual Research Report on Electrical Engineering and Computer Science 2016	Hassan Charaf, Gábor Harsányi, András Poppe, Sándor Imre and Bálint Kiss
[Excl]		23	Google Scholar	Applications of Cyber-Physical System: A Literature Review	Hong Chen
[Incl]	[Excl]	24	Google Scholar	Application-Aware Dynamic Fine-Grained Resource Provisioning in a Virtualized Cloud Data Center	Jing Bi, Haitao Yuan, Wei Tan, MengChu Zhou, Yushun Fan, Jia Zhang and Jianqiang Li
[Excl]		25	Google Scholar	Automated Validation of Minimum Risk Model-Based System Designs of Complex Avionics Systems	Nils Fischer
[Excl]		26	Google Scholar	Cloud-based smart asset management for urban flood control	Gangyan Xu, George Q. Huang, Ji Fang and Ji Chen
[Excl]		27	Google Scholar	A Case for Asynchronous Many Task Runtimes: A Modeling Approach for High Performance Computing and Big Data Analytics	Joshua Suetterlein
[Excl]		28	Google Scholar	A CRM Performance Measurement in Banking Using Integrated BSC and Customized ANP-BOCR Approach	Vesna Tornjanski, Snežana Knežević and Boris Delibašić

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		29	Google Scholar	Mining and Matching Relationships From Interaction Contexts in a Social Manufacturing Paradigm	Jiewu Leng and Pingyu Jiang
[Excl]		30	Google Scholar	Architecture Design and Interoperability Analysisof a SCADA System for the Power Network Control and Management	Pablo Albiol Graullera
[Incl]	[Excl]	31	Google Scholar	A Research on Simulation Framework for the Advancement of Supplying Management Competency	Jong Hun Woo, Youngmin Kim, Yong-Kuk Jeong and Shin Jong-Gye
[Excl]		32	Google Scholar	A Cost Model for Expressing and Estimating Ecological Costs of Software-Driven Systems	Thomas Schulze
[Excl]		33	Google Scholar	Modeling and evaluation of software system gamification elements	Darius Ašeriškis
[Excl]		34	Google Scholar	A survey on search-based model-driven engineering	Ilhem Boussaïd, Patrick Siarry and Mohamed Ahmed-Nacer
[Excl]		35	Google Scholar	Computational Frameworks: Systems, Models and Applications	Mamadou Kaba Traore
[Excl]		36	Google Scholar	Reverse logistics network redesign under uncertainty for wood waste in the CRD industry	Julien Trochu, Amin Chaabane and Mustapha Ouhimmou

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		37	Google Scholar	A Contribution to Resource-Aware Architectures for Humanoid Robots	Manfred Kroehnert
[Excl]		38	Google Scholar	Qualitative and Multi-Attribute Learning from Diverse DataCollections	Hao Zhang
[Excl]		39	Google Scholar	Automatic scaling in cloud computing	Tomáš Vondra
[Excl]		40	Google Scholar	Network performance & Quality of service in data networks involving spectrum utilization techniques	Hassan Fadel
[Excl]		41	Google Scholar	Designing a Self-Optimization System for Cognitive Wireless Home Networks	Elena Meshkova, Zhou Wang, Krisakorn Rerkrai, Junaid Ansari, Jad Nasreddine, Daniel Denkovski, Tim Farnham, Janne Riihijärvi, Liljana Gavrilovska and Petri Mähönen
[Incl]	[Excl]	42	Google Scholar	Application of value stream mapping (VSM) for lean and cycle time reduction in complex production environments: a case study	Dinesh Seth, Nitin Seth and Pratik Dhariwal

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		43	Google Scholar	A Complete Bibliography of Publications in Computer Systems Science and Engineering and International Journal of Computer Systems Science	-
[Excl]		44	Google Scholar	Real-Time Adaptive Data-Driven Perception for Anomaly Priority Scoring at Scale	Seyed Ali Miraftabzadeh
[Excl]		45	Google Scholar	Control-Theoretical Software Adaptation: A Systematic Literature Review	Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns and Martina Maggio
[Excl]		46	Google Scholar	Cloud manufacturing – a critical review of recent development and future trends	Göran Adamson, Lihui Wang, Magnus Holm and Philip Moore
[Excl]		47	Google Scholar	Model-Based Testing for Internet of Things Systems	Abbas Ahmad, Fabrice Bouquet, Elizabeta Fourneret and Bruno Legeard
[Excl]		48	Google Scholar	Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems	Stefan Hillmann
[Excl]		49	Google Scholar	A Generalized Proceduralization Framework for Urban Models with Applications in Procedural Modeling, Synthesis, and Reconstruction	Ilke Demir

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		50	Google Scholar	On the Move to Meaningful Internet Systems: OTM 2016 Workshops: Confederated ...	Ioana Ciuciu, Christophe Debruyne, Hervé Panetto, Georg Weichhart, Peter Bollen, Anna Fensel and Maria-Esther Vidal
[Incl]	[Excl]	51	Google Scholar	Modelo construtivista de avaliação de desempenho do processo de planejamento de compras no Poder Judiciário do Estado de Santa Catarina	Guilherme Mattos da Silva
[Excl]		52	Google Scholar	List of Recently Published Quality Research Papers	Dr. M. N. Ansari
[Excl]		53	Google Scholar	A Bibliography of Publications in Scientific Programming	-
[Excl]		54	Google Scholar	A Bibliography of Publications in ACM SIGOPS Operating Systems Review	-
[Excl]		55	Google Scholar	A Bibliography of Publications in Computer Languages	-
[Excl]		56	Google Scholar	A Complete Bibliography of Publications in Science of Computer Programming	-
[Excl]		57	Google Scholar	A Complete Bibliography of Publications in the Journal of Network and Computer Applications	-

Continues on the next page

Table 19 – *Continued from previous page*

Step 1	Step 2	#	Library	Title	Autor(s)
[Excl]		58	IEEE	A Context-Driven Framework for Proactive Decision Support With Applications	Manisha Mishra, David Sidoti, Gopi Vinod Avvari, Pujitha Mannaru, Diego Fernando Martínez Ayala, Krishna R. Pattipati and David L. Kleinman
[Excl]		59	IEEE	iCAST 2017 proceedings	-
[Excl]		60	IEEE	Systematic mapping study on MBT: tools and models	Maicon Bernardino, Elder M. Rodrigues, Avelino F. Zorzo and Luciano Marchezan
[Excl]		61	IEEE	A Consensus-Based Multicriteria Group Decision Model for Information Technology Management Committees	Alberto Sampaio Lima, José Neuman de Souza, J. Antão B. Moura and Igor P. da Silva
[Excl]		62	Periódicos	Perspectives and relationships in Supply Chain Simulation: A systematic literature review	Josenildo Brito Oliveira, Renato Silva Lima and José Arnaldo Barra Montevechi
[Excl]		63	Periódicos	A Survey on Spot Pricing in Cloud Computing	Dinesh Kumar, Gaurav Baranwal, Zahid Raza and Deo Prakash Vidyarthi
[Excl]		64	Periódicos	The evolution and future of manufacturing: A review	Behzad Esmaeilian, Sara Behdad and Ben Wang
[Excl]		65	Periódicos	Decision-making models for supply chain risk mitigation: A review	Varthini Rajagopal, Prasanna Venkatesan, Shanmugam and Mark Goh

APPENDIX B – ARTICLES INCLUDED IN RELATED WORKS

Table 20 – Articles included in the list of Related Works.

Title	Autor(s)
An Approach to Locating Delayed Activities in Software Processes	Yun-Zhi Jin, Hua Zhou, Hong-Ji Yang, Si-Jing Zhang, Ji-Dong Ge
Liveness and reachability analysis of BPMN process models	Anass Rachdi, Abdeslam En-Nouaary and Mohamed Dahchour
Software Maintenance Maturity Model (SMmm): the software maintenance process model	Alain April, Jane Huffman Hayes, Alain Abran, Reiner Dumke
Um modelo de avaliação de desempenho para suporte ao planejamento do processo de mudança de software	Marcely Dias

APPENDIX C – MERCURY MODELS AND SCRIPTS

Listing C.1 – Mercury Team One Process Model Script

```
arrival_delay = 3.44;
anls_delay = 0.5;
db_delay = 6.0;
dev_delay = 8.0;
tst_delay = 4.0;
prod_delay = 2.0;
anls = 1;
db = 1;
dev = 5;
tst = 1;
prod = 2;
weight_db = 0.25;
weight_m = 0.75;


SPN Model{

    place P0;
    place P1;
    place P10;
    place P11( tokens= 30 );
    place P12;
    place P13( tokens= dev );
    place P14;
    place P15( tokens= 30 );
    place P16;
    place P17( tokens= tst );
    place P18;
    place P19( tokens= 30 );
    place P2( tokens= 40 );
    place P20;
    place P21( tokens= prod );
    place P3( tokens= anls );
    place P4;
    place P5( tokens= 40 );
    place P6;
    place P7( tokens= 10 );
    place P8;
    place P9( tokens= db );


    immediateTransition TI0(
        inputs = [P0, P3],
        outputs = [P1, P2]
    );

    immediateTransition TI1(
        inputs = [P4, P7],
        outputs = [P5, P6]
    );
```

```
51
    immediateTransition TI2(
53      inputs = [P4, P11],
        outputs = [P5, P10]
55  );

57  immediateTransition TI3(
        inputs = [P6, P9],
59      outputs = [P8, P7]
    );
61
    immediateTransition TI4(
63      inputs = [P10, P13],
        outputs = [P12, P11]
65  );

67  immediateTransition TI5(
        inputs = [P14, P17],
69      outputs = [P15, P16]
    );
71
    immediateTransition TI6(
73      inputs = [P18, P21],
        outputs = [P19, P20]
75  );

77  timedTransition TE0(
        inputs = [P2],
79      outputs = [P0],
        delay = arrival_delay
81  );

83  timedTransition TE1(
        inputs = [P1, P5],
85      outputs = [P3, P4],
        delay = anls_delay,
87      serverType = "InfiniteServer"
    );
89
    timedTransition TE2(
91      inputs = [P8],
        outputs = [P9],
93      delay = db_delay,
        serverType = "InfiniteServer"
95  );

97  timedTransition TE3(
        inputs = [P12, P15],
99      outputs = [P13, P14],
        delay = dev_delay,
101     serverType = "InfiniteServer"
    );
103
    timedTransition TE4(
105     inputs = [P16, P19],
        outputs = [P18, P17],
107     delay = tst_delay,
```

```
        serverType = "InfiniteServer"
    );

    timedTransition TE5(
        inputs = [P20],
        outputs = [P21],
        delay = prod_delay,
        serverType = "InfiniteServer"
    );

    metric IUanls = stationaryAnalysis( expression = "(E{#P1})/anls" );
    metric IUdb = stationaryAnalysis( expression = "(E{#P8})/db" );
    metric IUdev = stationaryAnalysis( expression = "(E{#P12})/dev" );
    metric IUtst = stationaryAnalysis( expression = "(E{#P16})/tst" );
    metric IUprod = stationaryAnalysis( expression = "(E{#P20})/prod" );
    metric AUanls = stationaryAnalysis( expression = "P{#P1>0}" );
    metric AUdb = stationaryAnalysis( expression = "P{#P8>0}" );
    metric AUdev = stationaryAnalysis( expression = "P{#P12>0}" );
    metric AUtst = stationaryAnalysis( expression = "P{#P16>0}" );
    metric AUprod = stationaryAnalysis( expression = "P{#P20>0}" );
    metric TPm = stationaryAnalysis( expression = "(1/arrival_delay)*(1-P{((#P0=40)
        AND(#P1=anls))})*(weight_m)" );
    metric TPdb = stationaryAnalysis( expression = "(1/arrival_delay)*(1-P{((#P0=40)
        AND(#P1=anls))})*(weight_db)" );
    metric RTm = stationaryAnalysis( expression = "((E{#P0})+(E{#P1})+(E{#P4})+(E{#
        P10})+(E{#P12})+(E{#P14})+(E{#P16})+(E{#P18})+(E{#P20}))/((1/arrival_delay)
        *(1-P{((#P0=40)AND(#P1=anls))})*(weight_m))" );
    metric RTdb = stationaryAnalysis( expression = "((E{#P0})+(E{#P1})+(E{#P4})+(E{#
        P6})+(E{#P8}))/((1/arrival_delay)*(1-P{((#P0=40)AND(#P1=anls))})*(weight_db)
        )" );
}

main {
    setIntegerParameters("anls", "db", "dev", "tst", "prod");

    IUanls = solve( Model,IUanls );
    println(IUanls);

    IUdb = solve( Model,IUdb );
    println(IUdb);

    IUdev = solve( Model,IUdev );
    println(IUdev);

    IUtst = solve( Model,IUtst );
    println(IUtst);

    IUprod = solve( Model,IUprod );
    println(IUprod);

    AUanls = solve( Model,AUanls );
    println(AUanls);

    AUdb = solve( Model,AUdb );
    println(AUdb);

    AUdev = solve( Model,AUdev );
```

```
159     println(AUdev);

161     AUtst = solve( Model,AUtst );
        println(AUtst);
163
        AUprod = solve( Model,AUprod );
165     println(AUprod);

167     TPm = solve( Model,TPm );
        println(TPm);
169
        TPdb = solve( Model,TPdb );
171     println(TPdb);

173     RTm = solve( Model,RTm );
        println(RTm);
175
        RTdb = solve( Model,RTdb );
177     println(RTdb);

179 }
```

Listing C.2 – Mercury Sensitivity Analysis Script

```
    percentageDifference (
2       model_ = "Model",
        metric_ = "\textit{<metric>}",
4       samplingPoints = 10,
        parameters = (
6           anls = [ <interval> ],
            dbas = [ <interval> ],
8           devs = [ <interval> ],
            tsts = [ <interval> ],
10          engs = [ <interval> ]
        ),
12  output = (
            type = "R",
14          yLabel = "\textit{<metric>}",
            baselineValue = \textit{<metric>},
16          format = "png"
        )
18  );
```