UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

A Bottom Up Approach for Modeling Business Process using Time Petri Nets

Dissertação de Mestrado

Danillo Siqueira Ramos

São Cristóvão – Sergipe

2023

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Danillo Siqueira Ramos

A Bottom Up Approach for Modeling Business Process using Time Petri Nets

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Michel dos Santos Soares
Coorientador(a): Prof. Dr. Fábio Gomes Rocha

São Cristóvão – Sergipe

2023

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Ramos, Danillo Siqueira
R175b A bottom up approach for modeling business process using time Petri nets / Danillo Siqueira Ramos ; orientador Michel dos Santos Soares. - São Cristóvão, 2023.
84 f.; il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2023.

1. UML (Computação). 2. Petri, Redes de. 3. Modelos de capacitação e maturidade (Software). I. Soares, Michel dos Santos, orient. II. Título.

CDU 004.43

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: DANILLO SIQUEIRA RAMOS

Em 25 dias do mês de novembro do ano de dois mil e vinte dois, com início às 09h00min, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **DANILLO SIQUEIRA RAMOS**, que desenvolveu o trabalho intitulado: "***A Bottom Up Approach for Modeling Business Process using 'Time Petri Nets"***, sob a orientação do Prof. Dr. **Michel dos Santos Soares**. A Sessão foi presidida pelo Prof. Dr. **Michel dos Santos Soares** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Guillermo H. Rodriguez** (UBA) e, em seguida, o Prof. Dr. **Kalil Araujo Bispo** (PROCC/UFS) e o Prof. Dr. **Fábio Gomes Rocha** (PROCC/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovado "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos*", 25 de novembro de 2022.

Prof. Dr. Michel dos Santos Soares
(PROCC/UFS)
Presidente

Prof. Dr. Kalil Araujo Bispo
(PROCC/UFS)
Examinador Interno

Prof. Dr. Guillermo H. Rodriguez
(UBA)
Examinador Externo

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Examinador Interno

Danillo Siqueira Ramos
Candidato

Coordenador do PROCC
SIAPE - 1194034
PORTARIA Nº 1129 de 20 de Outubro de 2022

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Acknowledgements

E mais uma vez chega a hora de agradecer. Mais uma etapa da vida acadêmica chega ao fim, mas isso não quer dizer que seja o fim da caminhada, pois a estrada do conhecimento jamais se finda. Primeiramente, gostaria de agradecer a Deus, que sempre me deu forças, me fazendo levantar e seguir quando algumas vezes pensei em desistir. Em seguida, agradeço à minha família, que sempre me apoiou em tudo. Meus pais, Antônio e Josefa, por sempre estarem ao meu lado me incentivando a nunca parar, e sempre se esforçando para proporcionar os meios para que isto fosse possível. Meu irmão, Jonnathan, que do seu jeito, sempre me apoiou nessa caminhada. Minha esposa, Gislaine, que sempre esteve ao meu lado e apoia as minhas decisões, nunca soltando a minha mão. Aos meus orientadores, o professor Michel e o professor Fábio, por sempre estarem presentes, sempre dispostos a tirar todas as dúvidas, marcando reuniões extras fora do horário e dando broncas sempre que preciso. Se não fosse a excelente orientação por parte dos professores, não seria possível chegar ao fim desta caminhada desenvolvendo um excelente trabalho. A todos os meus amigos, os quais gostaria de mencionar aqui um a um, porém iria estender este texto a inúmeras páginas. Muito obrigado pelas palavras e gestos de apoio, durante esta e muitas outras caminhadas anteriores. Amigos que fiz durante o mestrado, amigos que tenho desde a graduação e amigos que me acompanham a vida toda, meu muito obrigado mais sincero. Vocês foram fundamentais para esta conquista. Aos colegas de trabalho, que sempre compreenderam as minhas ausências para assistir às aulas remotas, ou para dar andamento aos artigos, aos trabalhos e atividades do mestrado. Mesmo em épocas turbulentas, não reclamavam por eu não estar muito presente por muitas vezes. Vocês foram incríveis e agradeço muito pela contribuição nesta caminhada. Agradeço também aos professores Gilton e Rogério e também a Marianne, pelo incentivo do ingresso no mestrado, em uma época em que acreditava que não teria capacidade para tal. Enfim, a todos que fizeram parte, direta ou indiretamente desta caminhada, e que porventura tenha esquecido de mencionar, peço desculpas e sintam-se agradecidos. Para encerrar, repito o meu muito obrigado a todos.

“Temos que continuar aprendendo.
Temos que estar abertos.
E temos que estar prontos para espalhar nosso
conhecimento a fim de chegar a uma
compreensão mais elevada da realidade.”
(Thich Nhat Hanh)

Resumo

A UML é considerada uma linguagem de modelagem padrão de fato, oferecendo uma variedade de diagramas que fornecem aos engenheiros uma visão detalhada de vários aspectos do software modelado. Com a UML é possível modelar processos de negócios, modelos de dados, casos de uso, atividades, cenários de usos, entre outros, fornecendo um alto nível de abstração. Apesar de oferecer variados tipos de diagramas, a UML não está livre de inconsistências semânticas, ambiguidades ou notações impróprias. Tais problemas acabam criando interpretações equivocadas dos diagramas UML, as quais se tornam obstáculos na construção de um software confiável e adoção de uma arquitetura que seja adequada ao projeto. Uma forma de mitigar essas lacunas da UML é o uso de métodos formais, como as Redes de Petri, na modelagem de sistemas. Utilizar métodos formais permite, entre outras vantagens, simular o comportamento do sistema modelado e analisar suas propriedades. O uso de métodos formais auxilia na escolha de componentes arquiteturais adequados ao software, os quais atendem as suas necessidades de forma satisfatória. Nesta dissertação, o método formal usado para desenvolver os modelos são as Redes de Petri Temporais. Para modelar uma visão geral do sistema, foram elaborados modelos menores de forma a representar cada detalhe do sistema, os quais foram unidos posteriormente utilizando uma abordagem *bottom-up*. Para investigar qual o tipo de Rede de Petri com tempo é mais adequada e mais utilizada, um mapeamento sistemático da literatura foi realizado sobre o estado da arte sobre Redes de Petri com tempo nas últimas duas décadas (2001-2021), no qual ficou constatado que as Redes de Petri com tempo associado às transições são as mais utilizadas. Após esta investigação, um estudo de caso foi realizado para verificar o comportamento e analisar as propriedades dos modelos em Redes de Petri com tempo associados às transições em uma arquitetura de microserviços. Os Diagramas de Atividades e Diagramas de Casos de Uso do Sistema de Gerenciamento Eletrônico de Documentos foram transcritos em diagramas em Redes de Petri com tempo associado às suas transições. O sistema utiliza a arquitetura de microserviços, na qual pequenos serviços são implementados e executam uma única tarefa, se comunicando entre si por meio de mensagens assíncronas. Dos modelos transcritos, foram analisadas as propriedades e o comportamento da arquitetura de microserviços modelada em Redes de Petri Temporais foi verificado. Também foi possível observar problemas de temporização do software e identificar gargalos, o que auxiliou na escolha de componentes arquiteturais adequados ao software.

Palavras-chave: UML. Redes de Petri Temporais. Modelagem de Software. Microserviços.

Abstract

UML is considered a de facto standard software modeling language, offering a wide range of diagrams that provide engineers with a detailed view of various aspects of the modeled software. With the UML, it is possible to model business processes, data models, use cases, activities, scenarios, among others, providing a high level of abstraction. Despite offering many types of diagrams, UML is not free from semantic inconsistencies, ambiguities, or inappropriate notations. Such problems end up creating misinterpretations of UML diagrams, becoming obstacles in the construction of reliable software and adoption of an architecture that is suitable for the project. One way to mitigate these gaps left by the UML is the use of formal methods, such as Petri Nets, for modeling systems. Using formal methods allows, among other advantages, to simulate the behavior of the modeled system and analyze its properties. In addition, the use of formal methods helps in choosing the appropriate architectural components for the software, which satisfactorily meet the systems needs. In this dissertation, the formal method used to develop the models is the Time Petri Nets. For modeling an overview of the system, smaller models representing each microservice were modeled, and later joined using a bottom-up approach. To investigate in the literature which is the most used and most suitable type of Time Petri Nets, a systematic mapping was carried out on the state of the art of the last two decades (2001-2021), in which it was found that Petri Nets with time associated with its transitions are the most used. After the investigation, an industry case study was conducted in order to verify the behavior and analyze the properties of the models in Petri Nets with time associated with transitions on a microservices architecture. Activity diagrams and UML Use Cases of the Electronic Document Management System were transcribed to diagrams in Petri Nets with time associated with their transitions. The system uses the microservices architecture, in which small services are implemented that perform a single task and communicate with each other through asynchronous messages. Given the transcribed models, their properties were analyzed and microservices behavior modeled in Time Petri Nets was verified. In addition, it was also possible to observe software timing problems and identify bottlenecks, which helped in the choice of appropriate architectural components for the software.

Keywords: UML. Time Petri Nets. System Modeling. Microservices.

List of Figures

List of abbreviations and acronyms

SGED	*Sistema de Gerenciamento Eletrônico de Documentos* (Electronic Document Management System)
UC	Use Case
DCOMP	Departamento de Computação
UFS	Universidade Federal de Sergipe
UML	Unified Modeling Language
SEFAZ	*Secretaria do Estado da Fazenda*
HR	Human Resources
RQ	Research Question
TINA	Time Petri Nets Analyzer
SMD	State Machine Diagrams
ISO	International Organization of Standardization
IEC	International Electrotechnical Commission
MS	Microservice
OCR	Optical Character Recognition
AD	Activity Diagram

Contents

1 Introduction

In this Chapter, the main motivations for the development of this work are presented, in which the contextualization, research problem, objectives, methodology and the structure of the work are described.

1.1 Contextualization

Software is playing an increasingly important role in our daily lives, and a failure, no matter how small, can result in important economic losses or even put lives at risk. In this way, reliability becomes an essential requirement for software. Improving reliability means refining the software design, identifying errors earlier in the design life-cycle and thus reducing costs and efforts. Building models and adopting an architecture helps stakeholders to identify these errors early (WANG; PAN; CHEN, 2006; VENTERS et al., 2018).

Considered as the software blueprint, the software architecture helps developers and designers to correctly understand requirements, as well as detect and handle errors early (JAISWAL, 2019). Evolution of software systems is inevitable. From the monolithic systems, very common in the 1980s, to the current distributed systems, the way of developing systems has undergone significant changes. While such changes provide major advances to software systems, it is critical that developers understand them correctly, as well as understand the impacts the changes can have. Such an understanding is important as changes add to the complexity of the software, and understanding them improperly will likely result in an increase in system bugs, offering greater robustness and reliability to the software (WILLIAMS; CARVER, 2010; WOODS, 2016).

Reliability is a strong indicator of software quality. Providing greater reliability to the software increases its complexity, and obtaining an overview of the system becomes a critical factor for the early identification and correction of errors. Detecting and correcting errors has the

positive consequence of increasing the reliability of the software. In order for the software to meet this fundamental requirement for its success, it is important that the architecture for the software is suitable. Software reliability is linked to the architecture of its internal components (ZHANG et al., 2017).

Software architecture aims to mitigate the setbacks that changes may bring, as well as assist in the quick and effective resolution of such problems. Adopting a suitable architecture for the system to be developed will lead to the success of the project, as possible errors will soon be detected and the requirements will be correctly understood. However, a wrong choice of architectural style will lead to the failure of the project. The architecture serves as a bridge between requirements and software implementation (GARLAN, 2014). In order for the adopted architecture to adequately satisfy the requirements, it is necessary to model the architecture (SEDAGHATBAF; AZGOMI, 2019). Construction of models also helps in a better understanding of the complexity of the system to be developed (ALETI et al., 2012).

UML was proposed as a general-purpose software modeling language, combining existing notations and diagrams, providing several views of the software to be developed (FUENTES-FERNÁNDEZ; VALLECILLO-MORENO, 2004). An advantage of using UML diagrams is that it allows engineers to visualize the design of the software (ERMEL et al., 2018; GONCALES; FARIAS; BISCHOFF, 2019). The current version of the UML during the writing of this dissertation is 2.5.1, and it classifies diagrams into two main groups: structural diagrams, which show the static structure of the software and its parts, at different levels of abstraction, and behavioral diagrams, which show the operation of the software at runtime.

Despite offering a wide variety of diagrams, UML is not free from ambiguities in its diagrams, semantic inconsistencies, or inappropriate notations. Such problems end up creating obstacles for the correct understanding of the UML models (SIAU; LOO, 2006). One way to mitigate such problems is to use formal methods. Formal methods are mathematical tools and play an important role in ensuring the correctness of the system to be developed, in addition to providing an unambiguous model (BORGES; MOTA, 2007).

For example, if a model is found to be incorrect or inconsistent, formal verification is advantageous as it allows such problems to be corrected before implementation, reducing costs (ALHUMAIDAN et al., 2012; YANG et al., 2012; SINGH; SHARMA; SAXENA, 2016). Formal methods are a field of Computer Science, which focus on rigorous mathematical specification, design and verification of systems. Using formal methods, in addition to providing early finding and eliminating errors, also allows the construction of more robust and secure software (HALL, 1990; SESHIA; SADIGH; SASTRY, 2015; Ochem; Perlade, 2015). As an example of formal methods, the Z Notation and Petri Nets can be cited (MURATA, 1989; HALL, 1990).

Proposed by Carl Adam Petri in 1962, Petri Nets allow the simulation of behavior and formal verification of the modeled system (LIU; BARKAOUI, 2016). Time Petri Nets are a variation of Petri Nets, which use time constraints on system actions, being used to model systems

in which the addition of time constraints are necessary for their correct functioning (WANG; MAHULEA; SILVA, 2015). In this work, Time Petri Nets are chosen as a modeling language due to its characteristics, possibility of modeling temporal constraints and formal verification of properties, which are absent in UML diagrams.

1.2 Research Problem

Even having several types of diagrams, in order to provide the maximum understanding to engineers, UML does not allow formal modeling (ABBAS et al., 2021). The use of formal methods allows models to be formally verified. The mathematical rigor of formal methods allows software models to be verified at any stage of their life cycle, from requirements elicitation to software system evolution, thus ensuring greater reliability and robustness to the software before it is implemented. Especially in systems that need strict time constraints, formal methods become an enabler to ensure that constraints are not violated (WOODCOCK et al., 2009; PELED, 2019).

Despite having several diagrams that provide different points of view of a software, for some types of software systems UML is insufficient, especially for software where timing actions and responses are necessary, such as the case study, Electronic Document Management System (SGED). SGED manages documents for the HR sector of a public organization in the State of Sergipe, which must respect deadlines established in business rules. In addition to meeting deadlines, it is also necessary to identify possible bottlenecks in software requests, in order to avoid crashes and choose architectural components that are more suitable for the SGED.

Thus, to fill this UML gap, the models of the Electronic Document Management System (SGED) were transcribed, using Petri Nets with time, as well as their properties were verified. The system uses the microservices architecture, in which small services are implemented that perform a single task and communicate with each other through asynchronous messages. The complete model was designed using a bottom-up approach, which consists of building smaller models, involving small parts of the system and later joining them, providing a complete view of all parts of the system.

1.3 Objectives

The objective of this dissertation is to carry out a case study, through the transcription of the SGED UML Activity diagrams in Petri Nets with time, to analyze the behavior of the microservices architecture modeled in Petri Nets and to formally verify its properties. In addition, identify possible bottlenecks, through time constraints, and with that choose an asynchronous queuing system in order to avoid system crashes. To fulfill this general objective, the following specific objectives were proposed:

- Conducting a systematic mapping of the literature, in order to investigate which types

of Petri Nets with time are most used. The systematic mapping of the literature aims to investigate which type of Petri Net with time is most used for modeling microservices and which is the most used formal verification method to analyze the properties of the models, thus helping in the adoption of the Petri Net type which is most adequate for the problem described;

- Transcribe the UML Activity Diagrams which design the SGED, carry out the validation of the new models with the development team and then analyze the behavior of the models in Petri Nets in the microservices architecture, as well as analyze their properties;
- Identify, through the models, where asynchronous queues of requests should be established, in order to avoid system crashes;
- Verify the feasibility and scalability of models in Petri Nets with time in an industrial system, which needs to add time constraints to comply with its business rules, conducting an industry case study for such findings.

1.4 Methodology

In the Information Systems area, two paradigms characterize most researches: behavioral science and design science. While behavioral science focuses on human and organizational behavior, explaining or predicting them, design science focuses on extending human and organizational boundaries, developing and expanding their capabilities, through the creation or innovation of their artifacts (HEVNER et al., 2004).

The behavioral science paradigm seeks to develop and prove theories that involve human or organizational behavior. Its main objective is to develop and justify theories that predict or explain the phenomena surrounding the analysis, implementation, design, management and use of information systems. The behavioral science paradigm seeks to develop and prove theories that involve human or organizational behavior. Its main objective is to develop and justify theories that predict or explain the phenomena surrounding the analysis, implementation, design, management and use of information systems. It has its roots in the field of research methods in the natural sciences (HEVNER et al., 2004).

The design science paradigm has its roots in the field of engineering. This paradigm seeks to create solutions, innovations that define ideas, technical capabilities and the creation of products through which the analysis, design, implementation and use of information systems can be effectively carried out (HEVNER et al., 2004). This dissertation follows the design science paradigm.

Initially, with the purpose of investigating the state of the art on Petri Nets with time in the last two decades (2001-2021), a systematic mapping of the literature was carried out.

The systematic mapping is further detailed in Chapter 3.

After performing the systematic mapping, an industry case study was accomplished. The objective of the case study is to analyze the behavior of microservices modeled in Petri Nets. For this, the UML diagrams of the Electronic Document Management System (SGED), which uses the microservices architecture, were transcribed into Petri Nets with time associated with the transitions (T-time Petri Net). Such transcription was performed due to the impossibility of formal verification and simulation of behavior through UML diagrams. To analyze in detail each of the SGED microservices, small models were designed for each of them.

Time constraints were added to each of these models, an essential factor for the system to function as desired. From this, it was possible to observe the behavior of microservices modeled in T-time Petri Nets, analyze their properties and observe the feasibility of modeling large systems using Petri Nets. A merge of the smaller models was made, to build a model of the complete system. To accomplish this join, a bottom-up approach was adopted (SOARES; VRANCKEN, 2012).

Bottom-up approach was combined with the place fusion technique, which consists of merging places that have similar functionality into smaller Petri Nets. This approach, in addition to enabling the construction of models for large systems, facilitates scalability. As new functionalities are being inserted into the system, models can be built that represent it and soon after the smaller model is inserted into the larger model. Case study is further explained in Chapter 4.

1.5 Related Works

Yang et al. (2012) (YANG et al., 2012) performed the transcription of UML Sequence Diagrams in Timed Colored Petri Nets with Inhibitor Arcs. The authors also proposed formal mapping rules for the transcription of diagrams, in order to ensure the correctness of formal verifications. The focus of the work are real-time embedded systems. A case study is carried out for validation using ad-hoc vehicular networks systems, which allow vehicle occupants to communicate with other vehicles, bus stations or nearby internet hosts, in which the Sequence Diagrams are transcribed into Timed Colored Petri Nets with Inhibitor arcs and properties are analyzed and necessary time constraints are added. What differs the study cited from this work are the type of Petri nets, the type of transcribed UML diagrams and the type of modeled system. It is noteworthy that, according to the nature of the problem studied in this dissertation, the appropriate type of Petri Net is the T-Time Petri Net (Subsection 1.3), differing from the type of Net used in the aforementioned reported work.

Rahmoune, Chaoui and Kerkouche (2015) (RAHMOUNE; CHAOUI; KERKOUCHE, 2015) proposes a framework to automatically transform UML Activity diagrams into Petri Nets. The authors emphasize the lack of tools for analysis and verification of UML models, which makes

it impossible to formalize and simulate the behavior of models, for example. The formalization of the models also guarantees the absence of any ambiguities and their validation. Also, according to the authors, the formalization of the models guarantees more security and compliance for the project. The INA tool was used to check the conformity of the transformed models, and a case study was made with models representing telephone calls that were transformed and verified.

Meghzili et al (2017) (MEGHZILI et al., 2017) present an approach to transform UML State Machine Diagrams (UML SMD) into Colored Petri Nets (CPN) models. According to the authors, the UML can be used for modeling and the CPN can be used to verify the models' conformity. Also, according to the authors, techniques such as theorem proof and model checker are the appropriate solution to prove the correctness of transformations, since model transformation is one of the most error-prone tasks. Semantic transformation rules are defined, which are validated using the Isabelle/HOL theorem prover. Such rules help the target model, which are the CPNs, to preserve the properties of the source model. This dissertation, in addition to differing the type of Petri Net used, also differs in the properties analyzed. The properties analyzed by the cited authors are transformation properties, not the behavioral properties that will be analyzed in this dissertation. A case study will also be accomplished, with the transcripts of the models used in this dissertation.

Noulamo et al. (2018) (NOULAMO et al., 2018) propose a rigorous approach to automatically transform UML State Machine Diagram into Colored Temporal Petri Nets using composition patterns. To achieve automatic transformation, the authors establish composition rules, in order to make the approach rigorous, preserving the original properties of the UML model. Reuse methods are also used, so that in the proposed approach, it is possible to use models from other software. The authors validate the proposed approach in a software model of a hot beverage machine, detailing the transformation patterns and showing that it is also possible to combine the UML State Machine Diagrams with UML Activity Diagrams to represent the dynamic behavior of the system and also transform this combination of diagrams. The type of Petri Net used in this dissertation differs from the work cited, as well as the type of system that validates the case study carried out in this dissertation.

Shailesh, Nayak and Prazad (2020) (SHAILESH; NAYAK; PRASAD, 2020) performed the mapping of UML Sequence Diagrams to Generalized Stochastic Petri Nets, a model of Timed Petri Nets. The diagrams are from a real-time system. The purpose of the mapping was to perform a performance analysis of complex real-time systems. Although it is a non-functional parameter, it is critical for this type of system, making performance analysis an important task. The authors, in addition to mapping the Sequence Diagrams, present a performance evaluation methodology. The authors state that the analysis of performance in the initial stage of the system helps to improve its quality, and the use of the formality of timed Petri Nets allows such analysis in early stages of development. To demonstrate the proposed methodology, the authors carried out a case study on a manufacturing system on a production-on-demand strategy. What differs

this dissertation from the work is the analysis of properties that will be carried out, as well as the type of system and identification of suitable architectural elements.

Lopéz-Grao, Merseguer and Campos (2022) (LÓPEZ-GRAO; MERSEGUER; CAMPOS, 2022) propose a compositional approach to transcribe various types of UML diagrams into Stochastic Petri Nets. The authors emphasize the importance of the approach, since the UML is a semi-formal notation widely used for modeling various types of systems, but it does not have a scope for formal verification of the models. The focus of the work is the UML Activity Diagrams, which demonstrate the dynamic behavior of the system. Rules are defined for the transcription of Activity Diagrams, and these rules are validated in a case study in a basic email client model. After obtaining the complete model, the performance is analyzed in a prototype of a CASE tool developed by the authors, in which the performance between the two models is compared.

1.6 Dissertation Structure

This dissertation is structured as follows: in Chapter 2, the theoretical background is presented, with relevant definitions for understanding the work as a whole; in Chapter 3, the data obtained from a systematic mapping of the related literature are presented; in Chapter 4 the case study is presented, as well as the approach chosen to build the model is conceptualized; and in Chapter 5, the conclusions of this work are presented.

2

Theoretical Background

This Chapter presents the theoretical background of this dissertation, showing important concepts for a better understanding of the research.

2.1 Software Architecture

According to ISO/IEC 42010 (ISO/IEC/IEEE, 2011), software architecture is defined as “fundamental properties or concepts of a system in its environment, which are embodied in its elements, relationships, and principles of its design and evolution”. The architecture meets technical and operational needs, optimizing the performance and security attributes of the project, and showing the path traced between the requirements and the final system. It also involves serious decisions related to the organization of system development, in which each decision will have a considerable impact on the quality, maintenance, performance and overall success of the project (HASSELBRING, 2018; JAISWAL, 2019).

As computer systems evolve and become larger, their complexity grows in direct proportion. As a result, issues such as design, general analysis, and system specifications become critical issues (GARLAN, 2007). The adoption of an appropriate architectural style for software design is essential to correctly understand system requirements, as well as preventing and correcting possible problems (GARLAN, 2007).

Software Architecture has emerged as an important subfield of Software Engineering, gaining increasing interest from researchers in recent decades. Software systems are directly dependent on their architectural design, which ensures their long-term use, maintenance, and proper evolution, in a dynamic execution environment. Thus, the choice of an inappropriate architectural model will lead to the failure of the software project (GARLAN, 2000; GARLAN, 2007; VENTERS et al., 2018).

The Software Engineering community has been developing several techniques, approaches,

and tools that help the understanding of the architecture and the communication between project stakeholders (SHAHIN; LIANG; BABAR, 2014). Medvidovic and Taylor (2010) (MEDVIDOVIC; TAYLOR, 2010) state that at the heart of every well-designed software system is a good software architecture, which is a set of correct design decisions. An example of an architectural style that has been gaining prominence in industry and academia is microservices. In this architectural style, the software is developed in the form of small services that act independently and communicate through asynchronous messages (JAMSHIDI et al., 2018; HASSELBRING, 2018; JAISWAL, 2019). More details about this architectural style are described in Section 2.2.

2.2 Microservices

Considered a new architectural style for the development of computational applications based on small independent services communicating with each other through asynchronous messages, the microservices architecture aims to facilitate the maintenance and scalability of systems (DMITRY; MANFRED, 2014; ADERALDO et al., 2017).

Migrating systems to the microservices architecture brings several benefits, such as a better understanding of the base code, better management of availability and scalability, and use of different technologies, thus avoiding the imprisonment to just one (BALALAIE et al., 2018).

In addition to the advantages already mentioned, more advantages of adopting the microservices architecture can be mentioned, such as small services developed and implemented independently, allowing more efficiency for development teams. The services developed are small, have low coupling, and are specialized in a single task, unlike the monolithic architecture, in which the systems are developed in large modules and may present high coupling (KRYLOVSKIY; JAHN; PATTI, 2015; MAZLAMI; CITO; LEITNER, 2017; JAMSHIDI et al., 2018; PONCE; MáRQUEZ; ASTUDILLO, 2019).

Microservices architecture has received significant attention in academia and has also been widely used in industry (JAMSHIDI et al., 2018). For industry, in particular, the microservices architecture allows for faster delivery, as it reduces time-to-market (BALALAIE et al., 2018). The industry's competitiveness demands that their software products be robust, as well as more efficient and flexible, as well as more scalable, reconfigurable, and faster (CIAVOTTA et al., 2017).

Dragoni et al. (DRAGONI et al., 2017) highlight relevant features of monolithic architecture and points out its disadvantages. Also according to Dragoni et al. (DRAGONI et al., 2017), the microservices architecture emerged to deal with such problems:

1. Large monolithic systems are difficult to evolve and maintain. Mapping bugs requires lengthy code analysis;

2. Monolithic systems suffer from the so called “dependency hell”, in which adding or updating libraries can cause unwanted behavior on the system, which may not compile/run or, worse, cause unexpected behavior;

3. Any change to one module of a monolithic system requires a reboot of the entire system. For large projects, rebooting often leads to considerable downtime. These outages make development, testing, and maintenance difficult;

Despite all the advantages offered by the microservices architecture, this architectural style may not be suitable in all cases. For example, a system can be better developed in a monolithic way, not because it is not modular, but because it is not necessary to isolate its modules as much as in the microservices architecture. There will be cases where the costs of adopting the architecture may outweigh the benefits. There will also be cases where microservices are the right solution, but teams do not implement them correctly (JAMSHIDI et al., 2018).

Another factor that must be faced as a significant challenge of the microservices architecture is finding the right size of the modules. Assigning modules the right responsibilities, and right size, and designing well their interfaces is a challenge for microservices architecture, as well as other approaches where poorly designed boundaries can lead to increased network traffic.

This increase may result in a system unsuitable for the assigned tasks, due to poor performance and large instabilities. Another challenge also comes into this question: the granularity of services. There is a lack of agreement on the right size of a microservice among teams. Some teams understand that a microservice should only have a few dozen LOCs, while others encapsulate a few KLOCs as well as dozens of database classes and entities in their microservices (JAMSHIDI et al., 2018).

Jamshidi et al. (JAMSHIDI et al., 2018) state that to mitigate such challenges, since interest in microservices has been growing continuously and becoming a very important research topic, is the existence of an interaction between industry and academia. Despite an increasing number of published articles, these have little or no impact on microservices practice. If there is an effort between researchers and professionals to share a microservices infrastructure that can emulate, as accurately as possible, the production environments of typical microservices applications, it would give the possibility for academic teams not only to address the most representative problems faced by professionals but also to carry out more empirical and sector-focused studies.

2.3 Petri Nets

Proposed by Carl Adam Petri in 1962 in his doctoral thesis, Petri Nets are a mathematical and graphical modeling tool applicable to various types of systems (MURATA, 1989).

A Petri Net is graphically represented by a bipartite multigraph, composed of two types of nodes: places, which are represented by circles and transitions, represented by bars. Joining

places to transitions are arcs, represented by unidirectional arrows. Within the places, one or more tokens can be allocated. Tokens are dynamic elements, consumed and produced with the firing of transitions, which represent the behavior of the net (ZHOU; WU, 2018).

Petri nets are formally defined (MURATA, 1989; ZHOU; WU, 2018):

Def.: A Petri Net is a 5-tuple, PN = (P,T,F,W,M_0), in which:

- P = {p_1, p_2, p_3, ..., p_n} is a finite set of places;
- T = {t_1, t_2, t_3, ..., t_n} is a finite set of transitions;
- $F \subseteq$ (P x T) $\cup$ (T x P) is the set of arcs and establishes a flow relationship between places and transitions;
- $W : F \rightarrow \{1, 2, 3, ..., n\}$ represents the weight of the arcs;
- M_0: P $\rightarrow \{1, 2, 3, ..., n\}$ represents initial marking;
- P $\cap$ T = $\emptyset$ e P $\cup$ T $\neq \emptyset$.

According to flow relation F, an arc connects the place to the transition only, and not to another place. The same occurs with the transition, which is linked only to the place. A place can have n tags. An arc can move n tokens at a time. A transition is enabled for firing if there are tags at the input location that meet fire conditions (MURATA, 1989; ZHOU; WU, 2018).

Petri nets have some behavioral properties (MURATA, 1989):

- **Reachability:** this property is the basis for studying the dynamic properties of a system. The firing of an enabled transition will change the place of the token, respecting the firing rules. A sequence of fires will result in a sequence of marks. A tag M_n is said to be reachable by the initial tag M_0 if there is a firing sequence that allows M_0 to become M_n.
- **Boundedness:** a net is said to be $k - bounded$ or simply $bounded$ if the number of tokens in each place does not exceed a finite number of k at any mark reachable from M_0. A net is said to be secure if it is $1 - bounded$.
- **Liveness:** a net is said to be alive if, no matter what mark is reached by M_0, it is possible to fire any transition of the net, progressing any sequence of fires. Liveness property ensures that the system is deadlock free.
- **Reversibility/Home State:** a net is said to be reversible if M_0 is reachable from a M mark, that is, the net is able to return to its initial state.

Murata (MURATA, 1989) also presents some analysis methods for verifying Petri Nets:

- **Coverability Tree:** given a Petri Net with initial marking M_0, new tags can be obtained through the number of transitions enabled. For each new marking, new markings can be reached. This process will result in a tree-like representation of the tags. The nodes represent the markings reached from M_0, and the arcs represent the fired transitions. If the net is unbounded, the tree will be infinite. In limited nets, coverability tree is called the reachability tree and contains all reachable markings.

- **Incidence Matrix and State Equation:** given a Petri Net with n transitions and m places, the incidence matrix $A = [a_{ij}]$ is a matrix whose entries are given by:

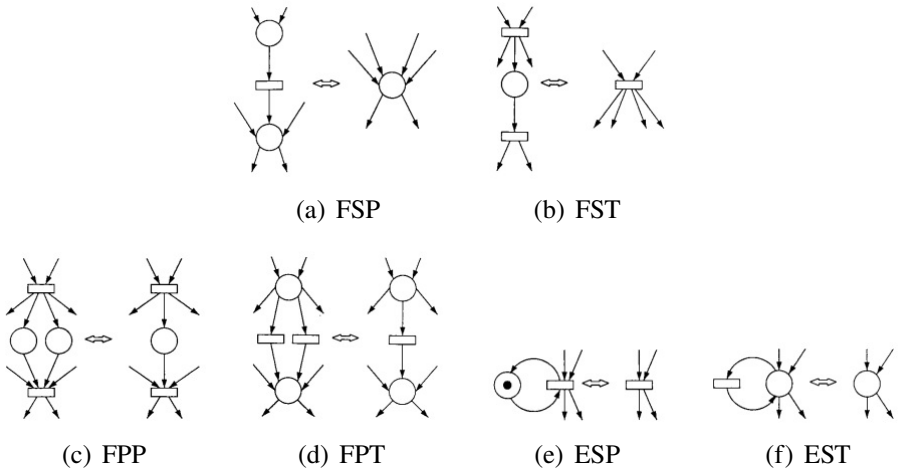
$$a_{ij} = a_{i_j}^{+} - a_{i_j}^{-}$$

Where $a_{i_j}^{+} = w(i, j)$ is the arc weight of the transition i to the exit place j and $a_{i_j}^{-} = w(i, j)$ is the arc weight of the transition i to the input place j.

State Equation: the M_k marking is written as an array of columns $m \times 1$. The order entry j of M_k denotes the number of tokens in place j immediately after the firing of order k, in any firing sequence. The order fire k, or control vector u_k is a column vector $n \times 1$ with $n - 1$ zeros, and a non-zero entry, 1 at the order position i, indicating that the transition i fires on the firing of order k. Since the order line i of the incidence matrix A denotes the change in marking as a result of firing the transition i, we can write the following equation of state for a Petri net:

$$M_k = M_k\_1 + A^T u_k, k = 1, 2, \ldots$$

- **Simple Reduction rules for Analysis:** Reduction rules aim to simplify the analysis of large systems, preserving the properties of the Petri Net. Six basic operations can be performed to reduce a net:



(a) FSP (b) FST

(c) FPP (d) FPT (e) ESP (f) EST

Figure 1 – Reduction Techniques

1. Fusion of Series Places (FSP), illustrated in Figure 1(a);
2. Fusion of Series Transitions (FST), illustrated in Figure 1(b);

3. Fusion of Parallel Places (FPP), illustrated in Figure 1(c);
4. Fusion of Parallel Transitions (FPT), illustrated in Figure 1(d);
5. Elimination of Self-Loop Places (ESP), illustrated in Figure 1(e);
6. Elimination of Self-Loop Transitions (EST), illustrated in Figure 1(f).

2.4 Petri Nets with time

According to Murata (MURATA, 1989), Petri Nets are applicable to various types of systems, with some variations. Among the existing variations, we can mention the Petri nets with time. This variation of Petri Nets are ideal for modeling systems where time constraints are crucial for their correct functioning (BASILE; CABASINO; SEATZU, 2015; ZHOU; WU, 2018).

Petri nets with time can be temporal, in which time is represented by an associated numerical interval. The interval can be linked to transitions (t-time Petri Net) or to places (p-time Petri Net). Time can also be set to an integer value, and the Petri Net is said to be timed. Likewise, the value can be associated with transitions (t-timed Petri Net) or with places (p-timed Petri Net) (BASILE; CABASINO; SEATZU, 2015; ZHOU; WU, 2018).

2.4.1 P-time Petri Nets

A time interval $\theta_{min} \leq \theta \leq \theta_{max}$ is associated with the places in the Petri Net (KHANSA, 1997). θ_{min} represents the minimum time since the token arrives at the place to enable the output transition to be fired, and θ_{max} represents the maximum time in which the token is available to be fired. As long as the token does not reach the minimum stay time, it will be available for firing other transitions.

In the interval $[\theta_{min}, \theta_{max}]$, the token will be available, making it possible to fire transitions. After the instant θ_{max}, the token is considered dead, and can no longer be used to fire any transition. The dead token means that there was a time constraint violation in the model. In this model, the main advantage is to represent a minimum and maximum interval between two events.

Figure 2 illustrates the evolution of a p-time Petri Net. Static intervals are associated with the letter e and visibility intervals with the letter v. It is also possible to observe in Figure 2 that the place P_1 has a static and visibility time interval equal to $[3, 5]$, that is, the token will only be available in this time interval. Before, for example in $\theta = 2$, the token is unavailable for use. After this interval, for example at $\theta = 6$, the token will be dead and cannot be used to fire t_1.

Still in Figure 2, place P_2 contains two associated intervals: $[2, 3]$ above and $[5, 6]$ below. The interval below, $[5, 6]$ is obtained by adding the fire instant in P_1, $theta = 3$ to the static interval of P_2 which is $[2, 3]$. The interval obtained by $[2 + 3, 3 + 3] = [5, 6]$ is the visibility interval of the token in place P_2.

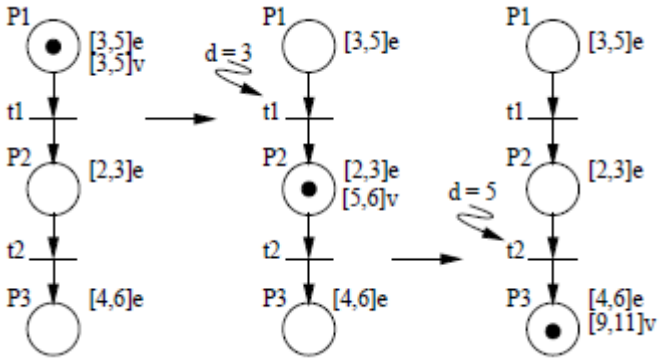


Figure 2 – Evolution of p-time Petri Net

Similar to P_2 place, the P_3 place in Figure 2 also has two associated intervals: $[4, 6]$ and $[9, 11]$. The visibility interval $[9, 11]$ shows when the token is available in P_3. This is because the token in P_2 was used to fire at the instant $\theta = 5$, which is added to the static interval $[4, 6]$ in P_3.

2.4.2 T-time Petri Nets

A t-time Petri net has a time interval associated with each transition. The fire is performed immediately, however the transition must be enabled for this during the time interval associated with it. An interval $[\theta_{min}, \theta_{max}]$ is associated with each transition; the sensitization duration must be greater than θ_{min} and less than θ_{max}, for example firing can only happen within the interval $[\theta_{min}, \theta_{max}]$.

Figure 3 shows the evolution of a t-time Petri Net. The transition t_1 that has the associated interval $[2, 3]$ is fired at the instant $\theta = 2$. Thus, with the subsequent firing of t_2, whose associated time interval is $[4, 6]$, the sensitization interval of t_2 is given by $[4 + 2, 6 + 2]$, resulting in the interval $[6, 8]$.

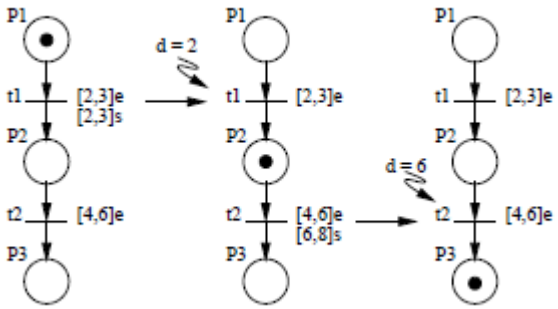


Figure 3 – Evolution of t-time Petri Net

2.4.3 P-timed Petri Nets

In a place that represents a certain activity A, a time θ is associated that represents the duration of the activity A. When a token arrives at this place at the instant τ, it can only be moved after the instant $\tau' = \tau + \theta$. While the activity A is being executed, the token cannot be used to fire the next transition, thus becoming unavailable. When the A activity is complete, the token is available and can fire the next transition. Therefore, the token can be in two states: available or unavailable. It is worth mentioning that only available tokens can fire transitions (RAMCHANDANI, 1973; CARDOSO; VALETTE, 1997).

2.4.4 T-timed Petri Nets

Fire duration is associated with each transition of the Petri Net. The fire is no longer instantaneous, but has a duration. Assigning a duration to the transition only makes sense if the transition is interpreted as being an activity and not an instantaneous event. The storage of tokens in the output places only takes place after a time has been associated.

The tokens of the transition's input places can be in two states: reserved, when the transition fires, remaining in this state until the associated time passes; unreserved, while the transition is not fired (RAMAMOORTHY; HO, 1980; CARDOSO; VALETTE, 1997).

2.4.5 Modeling using Petri Nets

Modeling architecture using a formal method, such as Petri Nets, ensures that requirements are correctly met, creating robust and reliable models that adequately meet the needs (WOODCOCK et al., 2009). Modeling using formal methods also helps in choosing architectural components suitable for software's purpose. For modeling using Petri Nets, two main approaches can be used: **top-down**, which consists of decomposing large systems into subsystems, and these subsystems are decomposed into even smaller subsystems, so that latter can be modeled as Petri Nets. The second approach is to build small models and then combine them to obtain a larger, more complex model representing the entire system. This approach is called **bottom-up** (VOGLER, 1992; GIRAULT; VALK, 2013). For the modeling presented in this work, a bottom-up approach was used (GIRAULT; VALK, 2013). Transitions as well as places can represent sub-nets, allowing top-down modeling at several levels of detail (CHRISTENSEN; PETRUCCI, 2000).

To combine smaller nets to form a larger net that represents the system as a whole, three main techniques can be employed: place fusion, arc addition, and transition fusion. The choice of appropriate technique will be according to communication method of the modeled system, that is, whether the system is synchronous or asynchronous (GIRAULT; VALK, 2013). First technique, place fusion, consists of merging places in the net that have similar tasks. This technique is simple and considered an effective way for asynchronous communications between subnets, in which the action that produces a token and the action that consumes it cannot occur at same

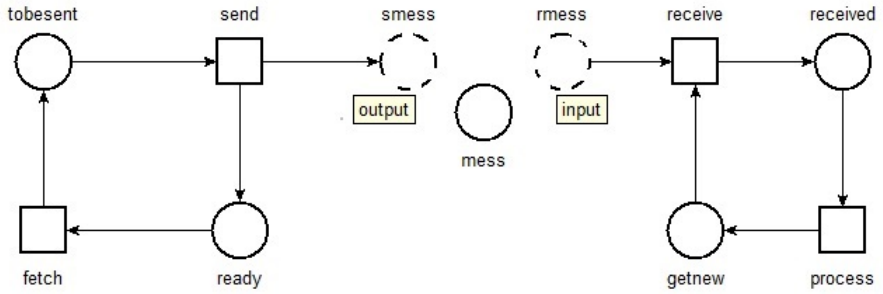


Figure 4 – Place Fusion Technique. Adapted from (GIRAULT; VALK, 2013)

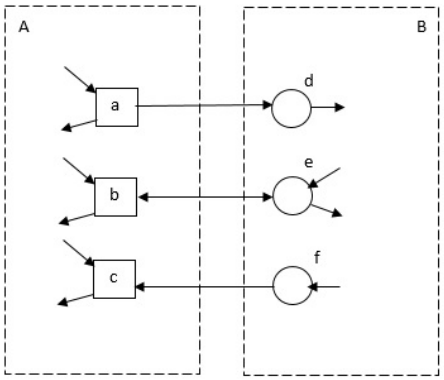


Figure 5 – Ways of adding arcs. Adapted from (GIRAULT; VALK, 2013)

time. When using this technique, it is recommended as a good practice to consider one place as the main one, which will be the merging of the two places coming from the subnet (HUANG; KIRCHNER, 2009; GIRAULT; VALK, 2013).

Figure 4 shows an example of place fusion technique.

The second technique, arc addition, is used for models that require synchronous communication. One can add an input arc at one place on the subnet, linking it to a transition on another subnet. For this new arc, firing transition will be restricted as it will need an extra token. As a result, events from one subnet become dependent on other subnet. An output arc can also be added. While adding an incoming arc restricts shots, the output arc does opposite effect, extending them. In this form of arc addition, the state of the subnet is changed by the occurrence of some event in the other subnet, used to control the sending of messages.

It is also possible to add an I/O arc. The behavior of an I/O arc is similar to input arc, restricting the fires of the involved transition (GIRAULT; VALK, 2013). Figure 5 shows three ways of adding arcs.

The third technique, transition fusion, is also used in models that require synchronous communication. When all subnet merge transitions are enabled, the resulting transition will be enabled. The consumption of transition resulting from merger will be the sum of the consumption

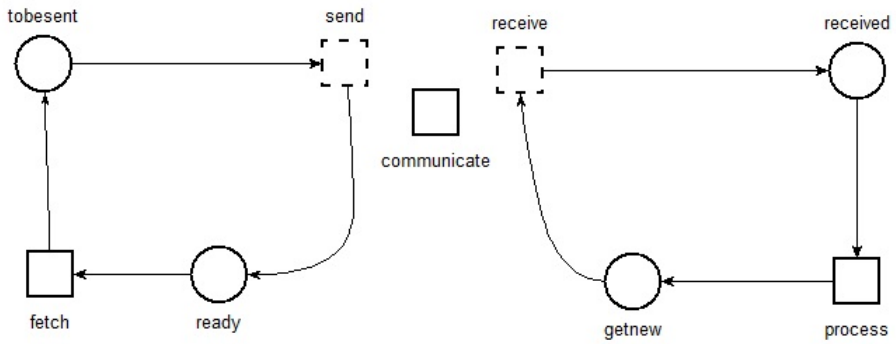


Figure 6 – Transition Fusion example. Adapted from (GIRAULT; VALK, 2013)

of origin transitions, and its production will be in same way (GIRAULT; VALK, 2013). Figure 6 shows an example of transition fusion.

2.5 Business Processes

A business process can be defined as a set of related and structured tasks, which produce services or products in order to meet the needs of a particular actor or a set of actors. Business processes are considered crucial points for organizations. However, processes should not be a set of random activities. The processes must be standardized and coordinated, in order to add positive values to the organization.

Furthermore, standardizing and coordinating business processes makes processes reusable, maximizing the value they create while reducing costs when compared to non-standard processes. Standardizing processes makes them measurable. Non-measurable processes do not allow determining what values they add to the organization. Coordinating and standardizing processes are activities that are part of process management. All organizations, whether governmental or not, have a large number of business processes. (DUMAS et al., 2013; CHANG, 2016; WEISSBACH et al., 2016).

Business Process Models (BPM) are graphical representations of existing processes in the organization. Some notations for the construction of these models can be used, such as Petri Nets and Business Process Modeling Notation (BPMN) for example. Process models are used as a basis for preparing documentation, conducting analysis and for the continuous improvement of business processes, with business process modeling being an integral part of Business Process Management initiatives.

It is also worth mentioning that the elaboration and analysis of models are considered critical factors for organizational improvement (SOLA et al., 2022; PUFAHL et al., 2022). Business Process Management has been gaining significant attention due to its potential to

increase productivity and reduce costs within the organization. Business Process Management paradigm is concerned with the design, monitoring, execution and continuous improvement of business processes and is considered by a significant majority of organizations as a factor of strategic advantage to support their operations (AREVALO et al., 2016; MENDLING et al., 2018).

In addition to such concerns and process optimizations, Business Process Management has the ability to adjust business processes to market requirements, as these requirements are dynamic, allowing organizations to respond to them more quickly and economically (TRIGO; BELFO; ESTéBANEZ, 2016).Business processes are also known to be the central point to reap the benefits offered by new information systems deployed in the organization. Projects that bring new information technologies to the organization significantly affect productivity by improving the corresponding business processes (MALINOVA; GROSS; MENDLING, 2022). Improving processes and aligning them with the organization's Information Systems (IS) is critical to the organization's competitiveness.

The high degree of alignment positively influences IT effectiveness, and leads to better business process performance as there is a tight coupling between IT applications and business processes in an organization (AALST, 2013; AVERSANO; GRASSO; TORTORELLA, 2016).

Rosemann and Broke (2015) (ROSEMANN; BROCKE, 2015) state that BPM has six core elements:

- Strategic Alignment: it is critical that BPM is aligned with the overall organizational strategy. Processes must be designed, executed, managed and measured according to the organization's specific priorities and strategic situations.
- Governance: governance establishes transparent and functionally appropriate and accountable accountability for different levels of BPM.
- Methods: set of tools and techniques to support and enable activities within the lifecycle of the BPM process and initiatives.
- Information Technology: focusing on analysis and process modeling support, IT solutions for BPM increasingly manifest in the form of Process-Aware Information Systems (PAIS), that is, the software has an explicit understanding of the process that needs to be executed.
- People: seen as a core element of BPM, are individuals or groups that improve and apply their knowledge and management skills to improve business performance.
- Culture: the culture aims to create a welcoming environment that embodies collective values and beliefs in relation to the organization. Although still considered a minor factor, comparative case studies demonstrate the strong impact of the culture factor on BPM success.

3

Systematic Mapping: Petri Nets with Time in the Past Two Decades

In order to investigate the existing literature on Petri Nets in the last 20 years, a systematic mapping was carried out, presented below.

As defined by Kitchenham (2004) (KITCHENHAM et al., 2009) and Petersen (2008) (PETERSEN et al., 2008), a systematic mapping provides a comprehensive view of a particular area of research, as well as offering several benefits. We can group the works reviewed by places of publication, such as journals and conferences, number of works published in a given year, performing a quantitative analysis and also highlighting the results of the analyzed research.

Among the benefits, we can highlight themes for new mapping or systematic reviews, identification of gaps in areas of research and assist in planning new research (TOFAN et al., 2014). Keele (2007) (KEELE et al., 2007) defines three main phases for conducting a literature mapping: planning the evaluation of articles, reviewing these articles and creating evaluation reports. In the first phase, research questions are defined. In the second phase, the primary studies necessary to satisfy the defined questions are identified.

In the third and last phase, the results of the research are presented (BAKAR; KASIRUN; SALLEH, 2015; CALDERÓN; RUIZ, 2015; SEPÚLVEDA; CRAVERO; CACHERO, 2016). Liberati (LIBERATI et al., 2009) stresses the importance of defining an evaluation protocol, according to which all forms of evaluation and selection must be described, such as databases used, research date, screening processes and inclusion and exclusion criteria.

Following the protocol defined by Kitchenham (2004) (KITCHENHAM et al., 2009), we carried out a study of Systematic Mapping of the Literature (SML) in order to obtain an overview of state of the art on use of Petri Nets (PN) with time in the last 20 years. To achieve the proposed objective, seven Research Questions (RQ) were defined:

- RQ1 - In which application domains Petri Nets with time are most used?

- RQ2 - Which types of Petri Nets with time are most common?
- RQ3 - What are the most used software tools for modeling Petri Nets with time?
- RQ4 - Which bibliometric key facts of the publications on Petri Nets with time?
- RQ5 - Where are Petri Nets with time scientific communities?
- RQ6 - Which formal verification methods are most used?
- RQ7 - Which good properties are most considered in formal verifications?

3.1 Search and Selection Strategy

In order to find articles relevant to mapping, a generic search string was created, containing the keywords considered relevant:

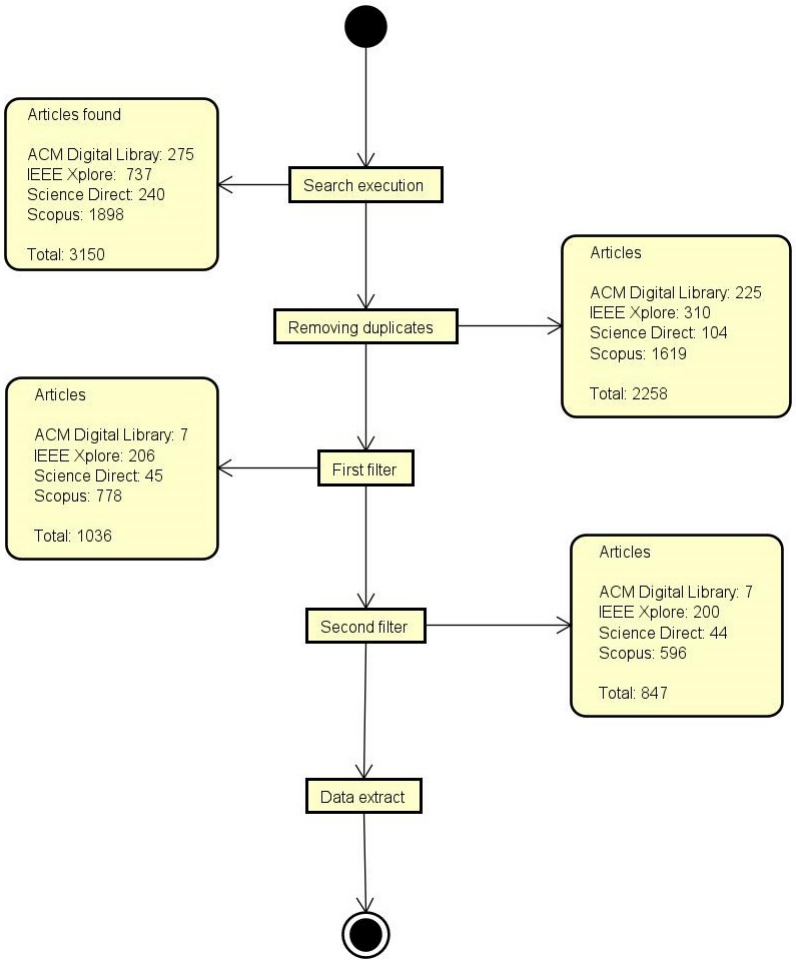


Figure 7 – Flowchart of activities execution

"time Petri Net" OR "time Petri Nets" OR "timed Petri Net" OR "timed Petri Nets"

The defined string was used in following scientific bases: ACM Digital Library, IEEE, Science Direct and Scopus. Considering the particularities of each bases, the generic string was adapted so that it could satisfactorily fulfill the objective.

Databases	**Found papers**
ACM	275
IEEE	737
Science Direct	240
Scopus	1898
total	3.150

Table 1 – Databases and Found papers

Table 1 shows the number of papers found in each database searched. In order to select the appropriate papers to fulfill the objectives of this research, some inclusion (IC) and exclusion (EC) criteria were developed. Table 2 shows defined criteria:

Inclusion Criteria	**Exclusion Criteria**
Use Petri Nets with time	Abstract Unavailable
	Books or Thesis
	Duplicate papers
	Language other than english
	Secondary studies

Table 2 – Selection Criterias

The flowchart of activities in the Figure 7 shows the steps taken to build this paper. As depicted in Figure 7, a total of 3150 papers were initially found. After removing the duplicated, there were 2258 papers to be analyzed. After applying the first filter (partial reading: title and abstract), 1036 papers remained. After the application of the second filter (complete reading and application of the selection criteria presented in Table 2), 847 articles were left for the data extraction necessary to obtain the desired results of this research.

3.2 Results

This Section and following subsections presents the obtained results.

3.2.1 RQ1: In which application domains Petri Nets with time are most used?

To satisfy this RQ, the application domains presented by the analyzed works were identified. Figure 8 shows which domains are most found in the literature.

As depicted in Figure 8, theoretical papers (with no defined application domain) are the majority, with 424 papers analyzed. Most authors suggest new approaches and new types of Petri

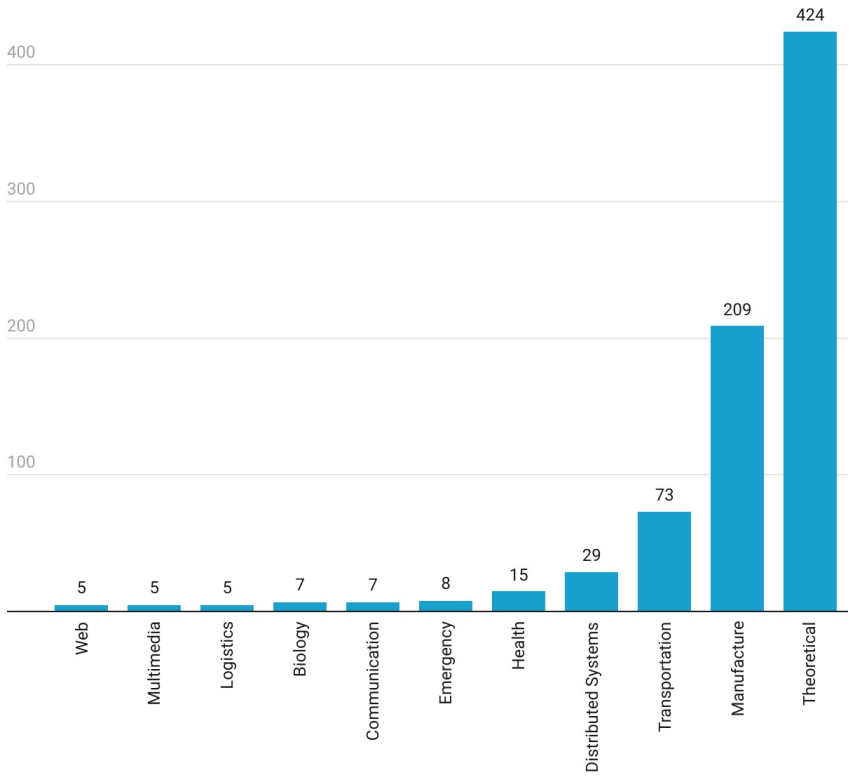


Figure 8 – Application Domains

Nets with time, or new algorithms and tools for analysis, as well as integration of Petri nets with time to current systems processes, and did not carry out application experiments in the real world. Second, the manufacturing sector is the most common, being cited in 209 of the analyzed papers, followed by all forms of Transportation, mentioned in 91 papers.

Other domains were mentioned in smaller numbers, as shown in Figure 8. They are: biological systems and communication systems, found in 7 papers, distributed systems (distr sys) found in 29 papers, logistics and multimedia systems found in 3 papers, web systems found in 5 papers and healthcare systems, found in 15 papers.

3.2.2 RQ2 - Which types of Petri Nets with time are most common?

As depicted in Figure 9, the most common type of Petri Net with time is the Transition Timed Petri Net (T-Timed Petri Net), having been mentioned in 335 papers. Then, the Time Transition Petri Net (T-Time Petri Net) was the most found in the analyzed papers, with 321 papers mentioning their use. The main difference between these two types of Petri nets with time is the representation of time. In T-Timed Petri Net, time is represented in a timed manner, that is, that transition must fire within that duration. In the T-Time Petri Net, time is represented in the form of intervals, with a minimum instant and a maximum instant for firing the transition.

Petri Nets with time in places are also among the most used, but appearing in a considerably

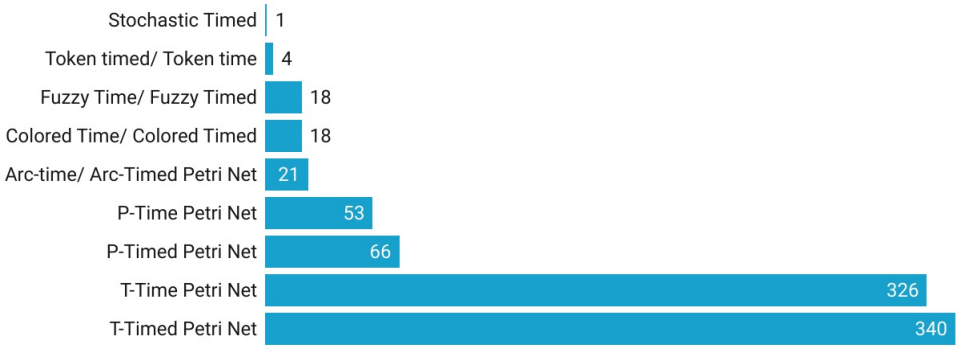


Figure 9 – Most used Petri Nets with time

smaller number than Petri Nets with time in transitions. Petri Nets with timed places (P-timed Petri Net) have been mentioned in 66 papers, and Petri Nets with time places (P-time Petri Nets) have been mentioned in 53 papers.

Petri Nets with timed arcs (Arc-timed Petri Net) and time arcs (Arc-time Petri Net) are seldom used, according to data obtained in the analysis of the papers. Arc-timed Petri Nets appear in 14 papers and Arc-time Petri Nets in only 6 papers.

Fuzzy Time Petri Nets are mentioned in 8 papers and the Fuzzy Timed Petri Nets appear in 10 papers. Colored Timed Petri Net are mentioned in 17 papers. Petri Nets with time classified as “others” totaled 17 papers. They are variations proposed by the authors, often to serve specific purposes or validate theories proposed by them.

3.2.3 RQ3 - What are the most used software tools for modeling Petri Nets with time?

To answer this RQ, tools used in the analyzed papers are identified. Most of the papers (622) did not specify the tools used, which is why the total number of tools identified is below the number of papers analyzed. Figure 10 shows which tools are most used for modeling problems in Petri Nets with time.

As shown in Figure 10, the most used tool for modeling Petri Nets with time is the TINA (TIme Petri Net Analyzer), mentioned in 47 papers. CPN tools appears as the second most used, mentioned in 29 papers. ROMEO is the third most used tool, mentioned 21 times. As less used, the HYPENS tool appears with only 5 papers mentioning its use.

Tools classified as “Others” are tools proposed by the authors to validate their theories or to meet specific purposes. These tools are in their majority academic tools resulted from a research project.

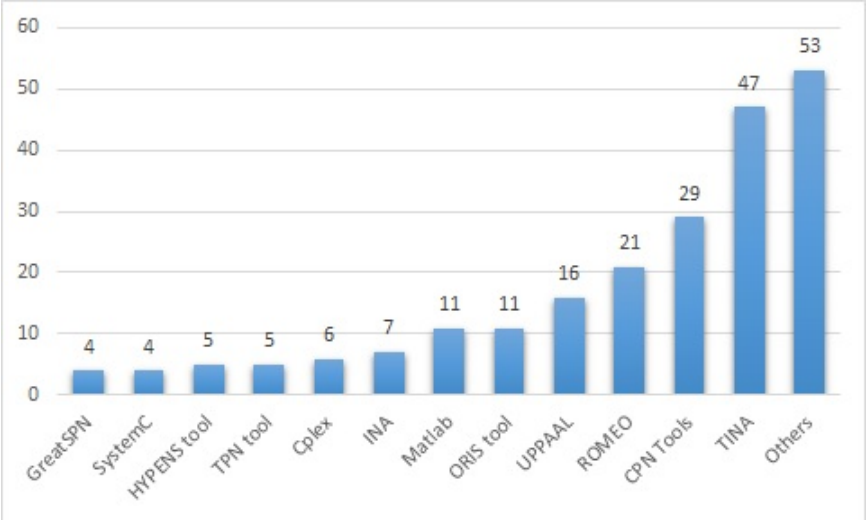


Figure 10 – Most used tools for Petri Nets with time

3.2.4 RQ4 - Which are bibliometric key facts of the publications on Petri Nets with time?

Publications in the area of Petri Nets with time are presented in this Subsection, as well as the type of venue in which it was published (conference or journal). Figure 11 shows the number of articles per year, in the last 20 years, and Figure 12 shows the type of venue of publication.

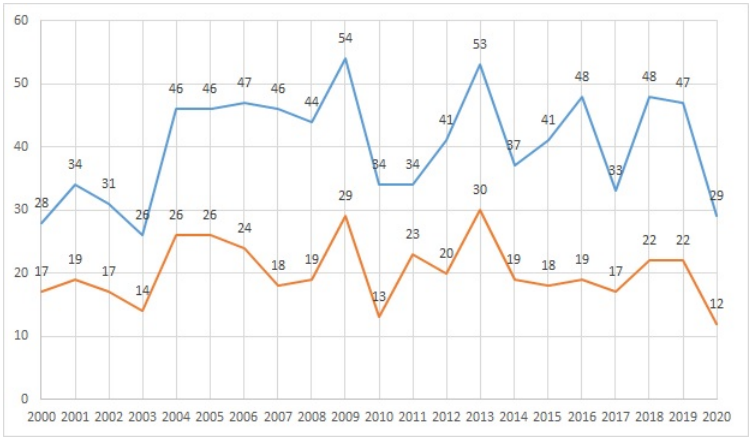


Figure 11 – Number of publications per year/ Theoretical papers per year

According to Figure 11, the year in which there were more publications on Petri Nets with time was 2009, in which there were 54 papers published. Then, the year 2013 had 53 papers published. The year with the fewest publications was 2000, with only 28 papers. Figure 11 also shows the number of theoretical papers per year analyzed (red line). According to the Figure 11, the year in which there were more theoretical papers on Petri Nets was 2009, with 29 publications.

The year 2020 presented only 29 published papers. This may be linked to the fact that the searches for this paper were made in up until October 2020.

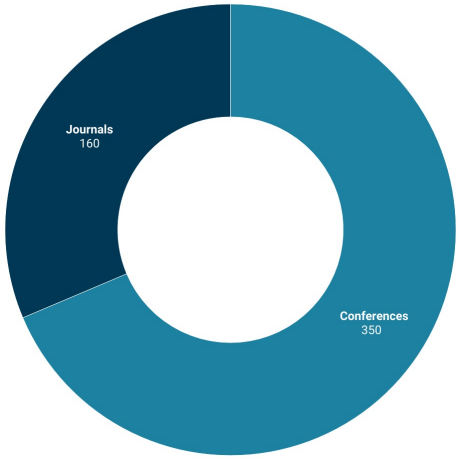


Figure 12 – Publications in conferences/journals

Figure 12 shows that 69% of papers (350 papers) were published in conferences and 31% (160 papers) in journals.

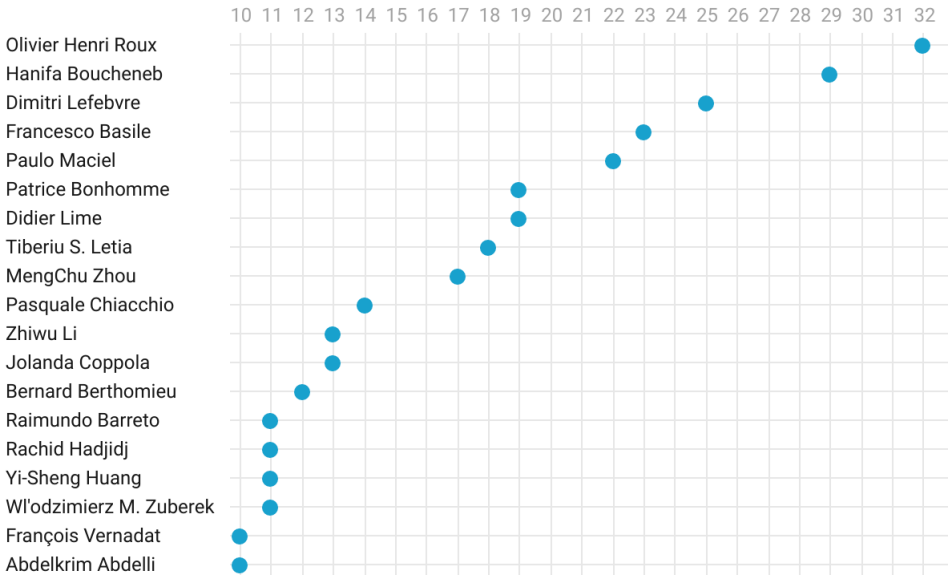


Figure 13 – Publications in conferences/journals per author

Figure 13 shows the most prolific authors in the analyzed period (2000 - October/2020).

3.2.5 RQ5 - Where are Petri Nets with time scientific communities?

In this Subsection, the countries that most publish research on Petri Nets with time are shown. Figure 14 shows data related to the number of publications by countries (blue line).

According to Figure 14, the country with the largest number of authors on Petri Nets with time is China, with 561 researchers. France appears in second place, with 465 researchers. Third, with 265 researchers, comes Italy. Figure 15 shows the distribution of the number of researchers

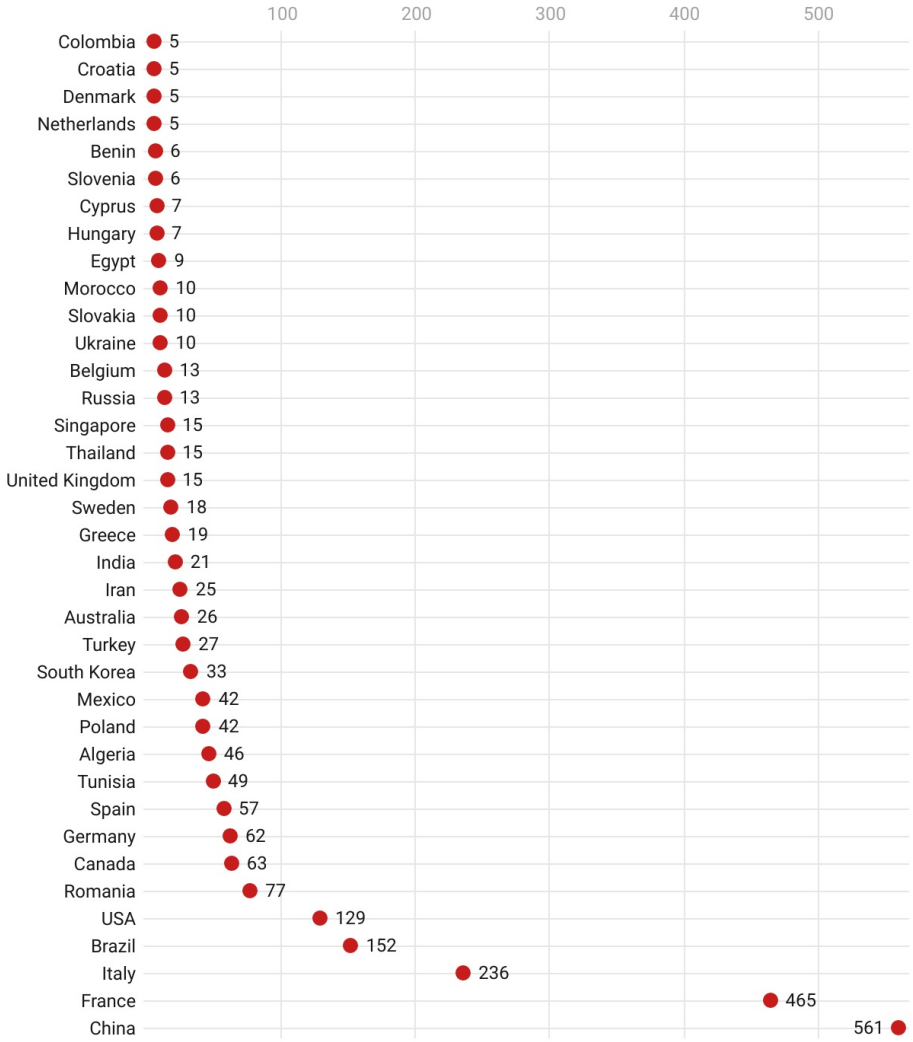


Figure 14 – Scientific communities of Petri Nets with time

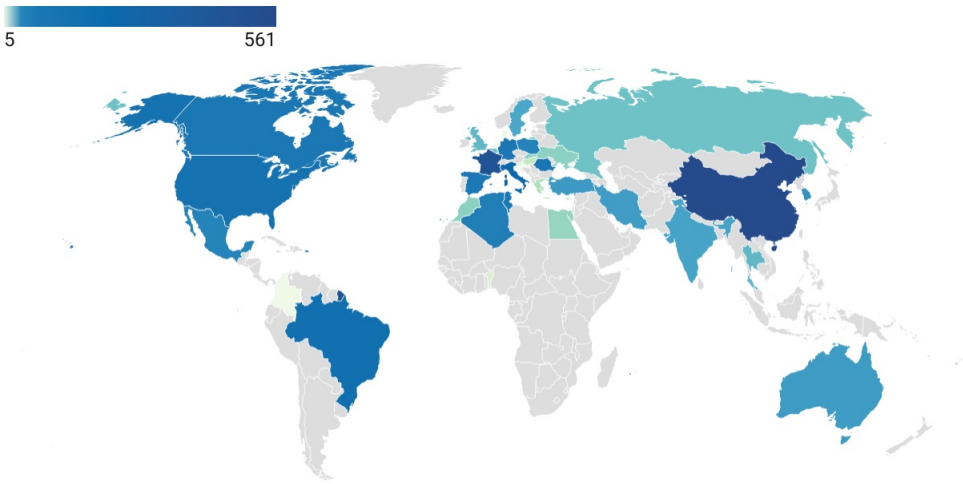


Figure 15 – Worldwide distribution of papers

around the world. In order to facilitate visualization, Figure 16 highlights the countries of Europe according to the number of published papers.

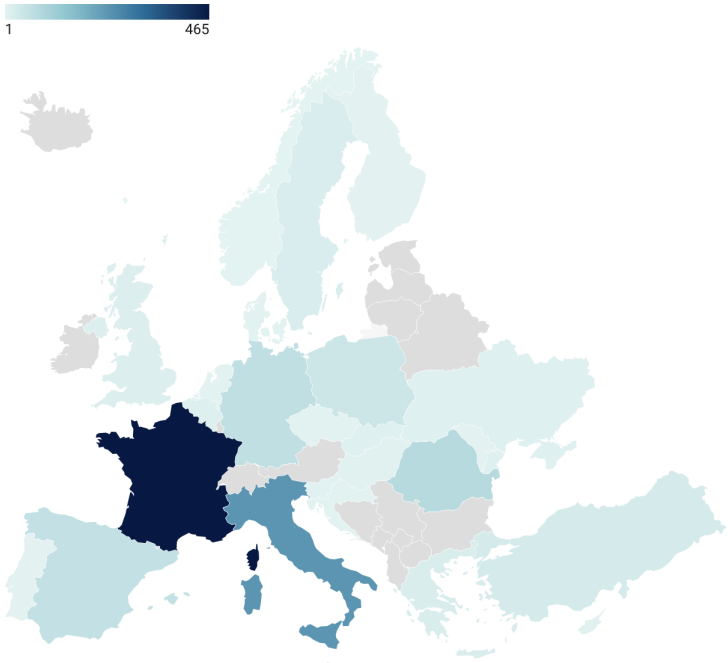


Figure 16 – Distribution of papers published in Europe

For each paper, all authors were counted. Therefore, for instance, in case of a paper A, with author 1 from France, and author 2 from China, even tough this is one paper, each country is counted.

3.2.6 RQ6 - Which formal verification methods are most used?

According to the data collected in the analysis, the most used verification method is the state space (reachability graph or coverability tree), mentioned in 363 papers. However, most of the papers (396 papers) do not mention the use of any formal verification method.

The least used method is Linear Algebra, mentioned in only 9 papers. The methods classified as "others" are those proposed by the authors, to meet specific purposes or to validate their proposals.

It is worth mentioning that the total number of methods exceeds the number of papers due to the fact that some papers mention the use of more than one formal verification method.

3.2.7 RQ7 - Which good properties are most considered in formal verification?

Here, it is shown which properties of Petri Nets with time have been used the most in the last 20 years. Figure 18 shows the properties extracted from the analyzed papers.

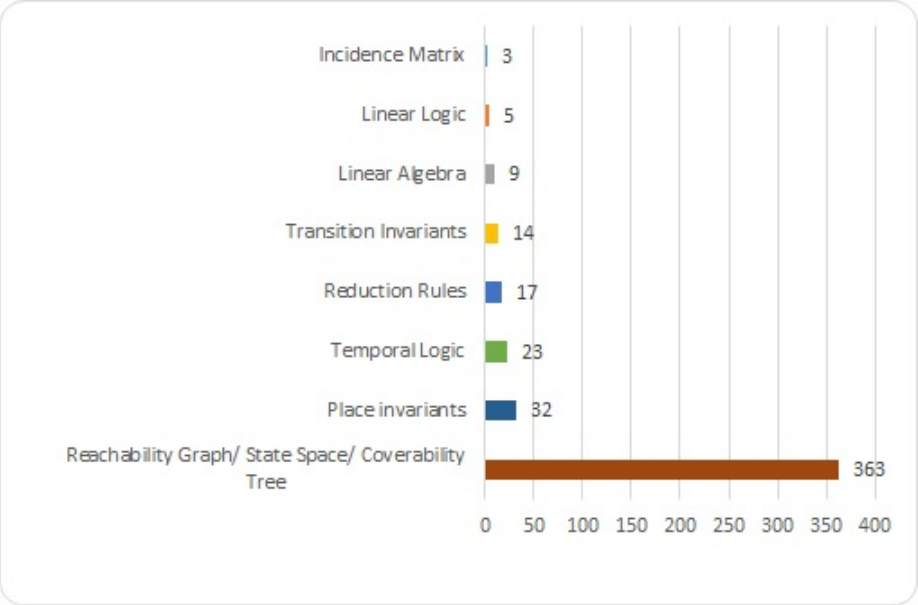


Figure 17 – Most used verification methods

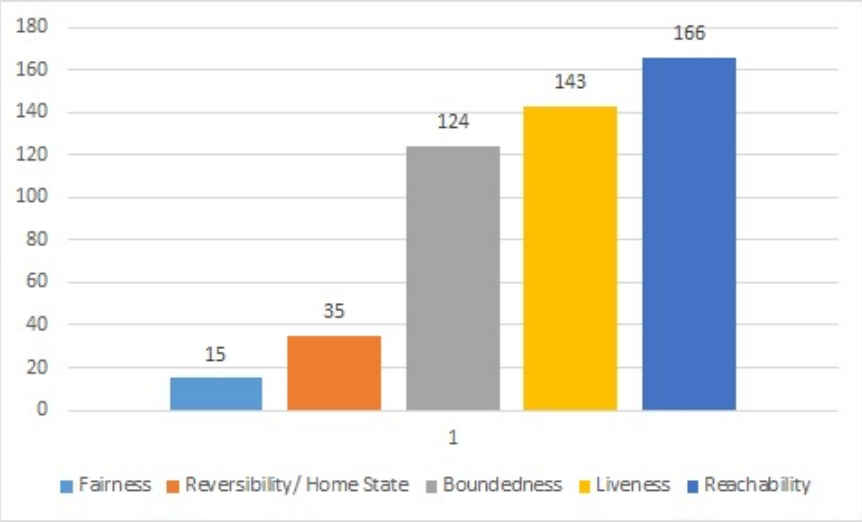


Figure 18 – Most analyzed properties

According to Figure 18, the Reachability property was the most verified in the last 20 years, being mentioned in 166 papers. The second most verified, with 143 mentions was the Liveness property. Most of the papers (526) did not verify the properties. The properties classified as "others" are proposed by the authors, based on the properties already existing in the literature.

It is worth noting that the number of properties is greater than the number of papers analyzed due to the fact that there are papers that verify more than one property.

3.3 Conclusion

Regarding the relationship of time and Petri nets, the choices between the variety of time approaches is vast. Once the solution to a given problem calls for the use of Petri nets with time, one still has to make a number of decisions, frequently with few notions on known options. For instance, the time nature as deterministic or interval, associated to places, arcs or transitions, tool support, previous works on the specific domain of interest, and so on. Therefore, for new users, or even experienced ones, a synthesis of Petri nets and time is most welcome, as described in this article.

Our protocol retrieved 3150 papers in ACM Digital Library, IEEE, Science Direct and Scopus. From these, after excluding duplicated papers and applying other exclusion criteria, we found 847 articles left for the data extraction necessary to obtain the desired results of this research. Among other results, we can mention the necessity of more applied research, as we found a high proportional number of theoretical papers, and also the need for better and improved software tools, which are most often academic solutions.

Future work will focus on specific Systematic Literature Reviews on Petri Nets with time, with main focus on the advantages and disadvantages on the most common models found in this article, t-time and t-timed, when used in practice in industry, as well as the relationship of these models with other modeling languages that are used in industry, such as UML and SysML.

4

Case Study

This Chapter presents a case study carried out with the SGED system, as well as transcribed models, used approach to build the complete model, results obtained, and discussion about results.

4.1 Problem Description

SGED is a system that is currently in use in the Human Resources (HR) sector of the Secretary of State for Finance of the State of Sergipe (*Secretaria da Fazenda do Estado de Sergipe* - SEFAZ/SE), a public agency administered by the Government of the State of Sergipe, Brazil. All documentation related to the agency's employees is managed by the system. Sending documents to the system is done through scanned documents, involving image processing of these documents for data extraction. The SGED is an open source system, developed by undergraduate students who make up the DigitalSE research group of the Information Systems course at Tiradentes University (UNIT) and made available to SEFAZ/SE.

HR Manager will be able to register new HR analysts, who will be able to validate or invalidate the documents sent. Agency's business rules define validation and invalidation criteria, which generally depends on deadlines to be met.

It is also possible to request reports in SGED. Core activities of SGED, such as image processing, requesting queries, generating reports, and extracting and analyzing information are time-consuming, which leads timing to become a central point in SGED modeling. As noted, SGED image processing requests and activities end up demanding time, which can lead to bottlenecks.

Thus, it is necessary to establish asynchronous queues, and the choice of a queuing system must be adequate so that there are no impediments that hinder the execution of other activities. The queue system must establish asynchronous queues, which avoids system crashes

and allows other activities and requests to be executed in parallel.

Timing models become an important factor, as it helps in the proper choice of architectural components of the SGED. The choice of suitable architectural components culminates in the success of the project and ensures that the system will satisfactorily comply with the business rules for which it is intended.

Current SGED models were designed using UML. Despite providing several diagrams, the UML does not allow the formal establishment of time intervals required in asynchronous communications, which are necessary for the correct functioning of the SGED. This gap makes it difficult, in addition to identifying possible bottlenecks, to choose appropriate architectural components for the proper functioning of the software.

It is also important to simulate behavior to map possible states of the system, in addition to checking possible violations of time constraints. SGED is an industry system, used in a public organization and was chosen because it is a system that needs time constraints and uses the microservices architecture, thus requiring formal modeling to identify a possible violation of timing constraints with business processes, as well as observing the behavior of microservices architecture when modeled in Petri Nets with time.

Therefore, the UML diagrams of the SGED were transcribed in Petri nets with time intervals associated with the transitions, to identify the execution points that need timing, to implement asynchronous queues essential to the desired operation. The states of the models were also mapped and their respective graphs were constructed, helping to analyze the properties inherent to the SGED.

The modeling approach used in this dissertation to develop the models that will serve as a basis for the analysis is the bottom-up approach. Such an approach consists of building smaller models, representing small parts of the system, which are later joined to form the model that represents the complete system. Such an approach allows a better understanding of the parts of the system and facilitates its scalability, facilitating the modeling of large systems that have several functionalities, as is the case with SGED.

In this dissertation, techniques adopted to combine models produced to represent a system as a whole is place fusion, as there is no need for synchronous communication in the modeled system, as shown in Figure 4. Also, using the bottom-up approach, the models that represent parts of the system were combined in order to represent the complete system

Smaller models were developed, encompassing all aspects of the system. With the place fusion technique, places that performed similar tasks were merged, becoming a single place, thus obtaining the complete model for each Use Case. In a similar way, the complete model of the SGED was elaborated. A bottom-up approach was combined with the place fusion technique, so that the smaller models were joined and, in turn, the complete model of each Use Case and SGED was obtained.

4.2 System Modeling

SGED business rules were initially modeled using UML diagrams. Models are represented in a Use Case (UC) diagram, representing an SGED overview. Figure 19 shows the UC diagram with all aspects that compose SGED and provides an overview. From this UC diagram (Figure 19), models were modeled that provide a view of the activities performed by the SGED.

According to the UC diagram in Figure 19, Employee selects the document type that will send to SGED for analysis, therefore, document type must be previously registered. HR Analyst is the actor responsible for registering document types that will be analyzed.

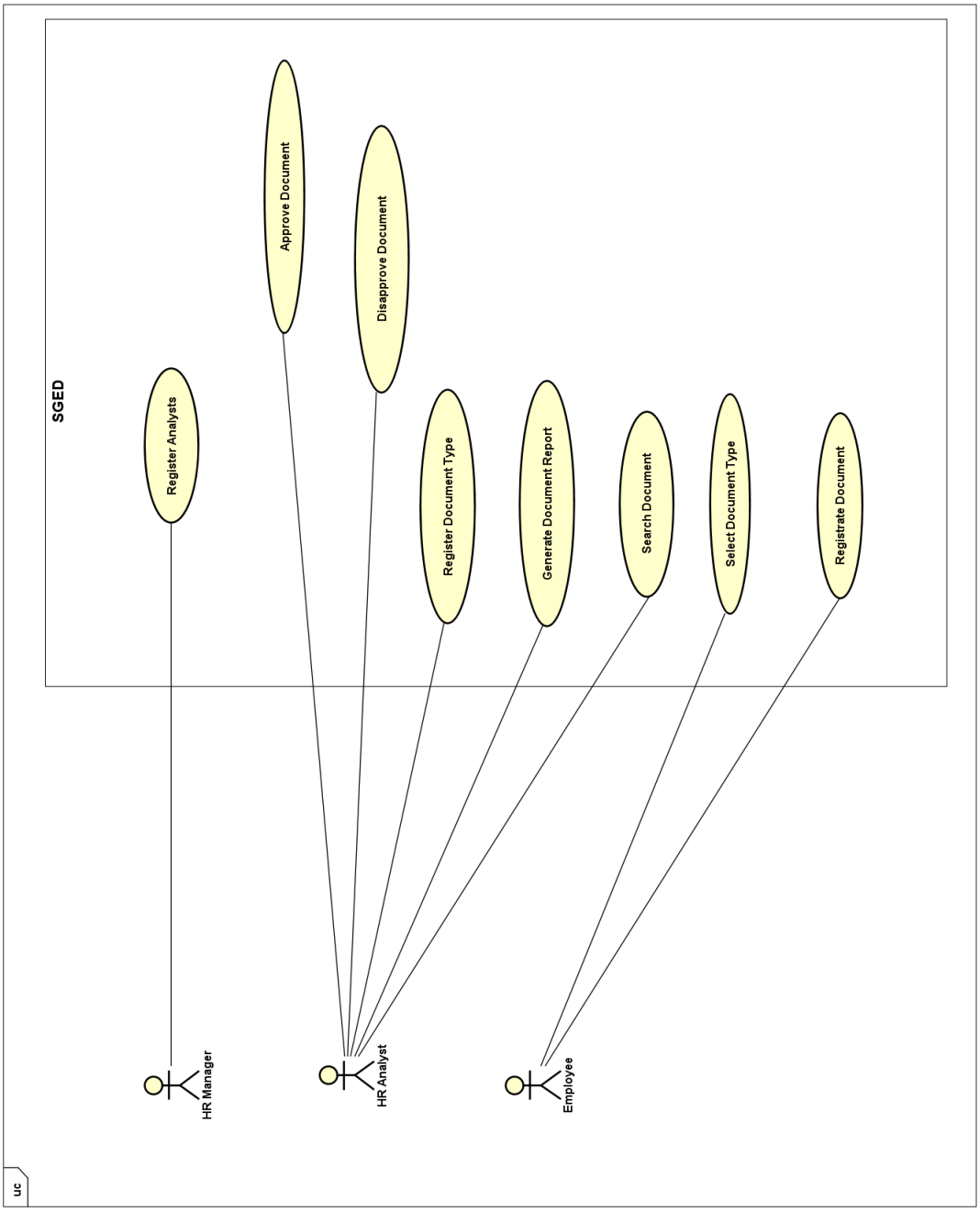


Figure 19 – SGED Use Case Diagram

After selecting which document type to send (medical certificate, identification document, among others), the Extract Information microservice (MS) will pre-process the image of the scanned document, proceeding to extract data with OCR. After extracting data, MS will still do indexing, that is, a process of sending it to the database and filtering so that only relevant data are sent for analysis.

Soon after, indexed data is sent to MS Document Analytics which will analyze data received from MS Extract Information. The data will be analyzed by previously defined business rules and will be made available to the HR Analyst, who will then approve or not the submitted document. Whether approved or not, the document will be registered in the employee's functional folder.

In addition to approving or disapproving sent documents according to business rules of the organization, the HR Analyst is also responsible for researching documents and generating reports with searches performed, as necessary. HR Analysts are registered in the system by the HR Manager. Details of each phase of the processing performed will be described throughout this dissertation.

Such activities require the use of microservices to extract and analyze data and generate reports, which are essential to comply with business rules, for example. The first Activity Diagram (AD) of the SGED, represented by Figure 20, shows the flow of activities to perform the registration of the HR analyst in the system's database. HR analyst is the actor responsible for analyzing and validating medical certificates sent by employees to justify absences.

Once the HR Manager sends the document to the SGED, the latter sends it to MS Extract Information. This microservice performs adjustments to the document image since the document is sent in form of an image. After adjusting the image, to improve its visibility, Extract Information applies the Optical Character Recognition (OCR) mechanism, to extract data relevant to the registration process. Extract Information Microservice (MS) still performs indexing of the extracted data. Indexing performs a kind of data cleaning so that only the relevant data are sent to the analysis performed by MS Document Analytics, making data analysis a faster process. Then, as illustrated in Figure 20, data is sent to MS Document Analytics.

MS Document Analytics then sends the analyzed information back to the SGED, which will notify the responsible actor (HR Manager) to validate the document and thus register the analyst. Registration must be completed within 24 hours, in compliance with the established business rules. If this implementation does not take place within the period established in the business rule, the document is deleted and it will be necessary to make another request and send it.

The second UC concerns the types of documents that will be accepted for analysis by the SGED. HR Analyst will register types of documents that meet the demand of the organization's business rules and will be accepted for sending and analyzing data. The activity diagram for UC2

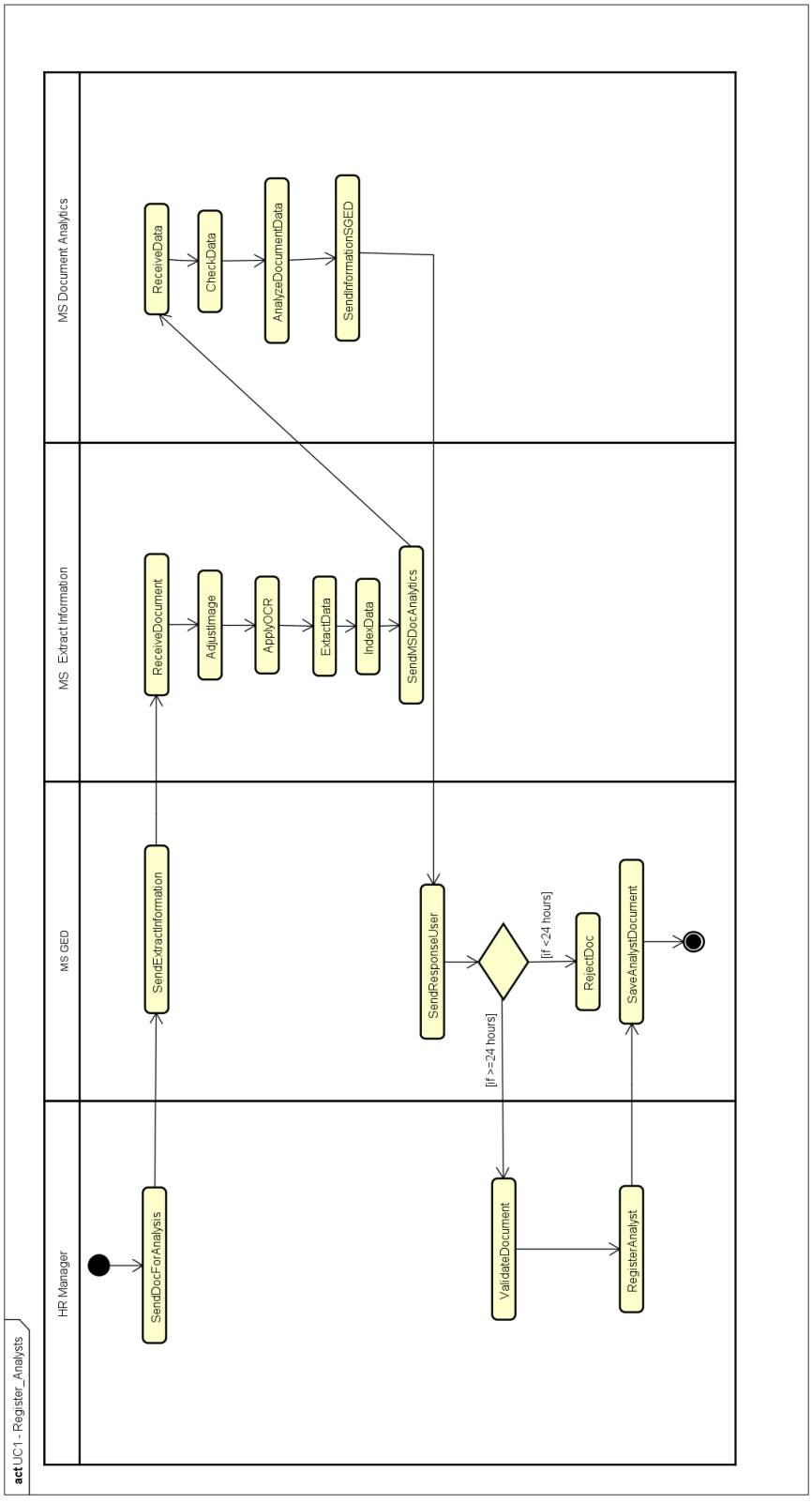


Figure 20 – SGED - UC1: Register Analysts

is shown in Figure 21.

UC3 and UC4 are interdependent use cases. UC4 searches for documents registered in

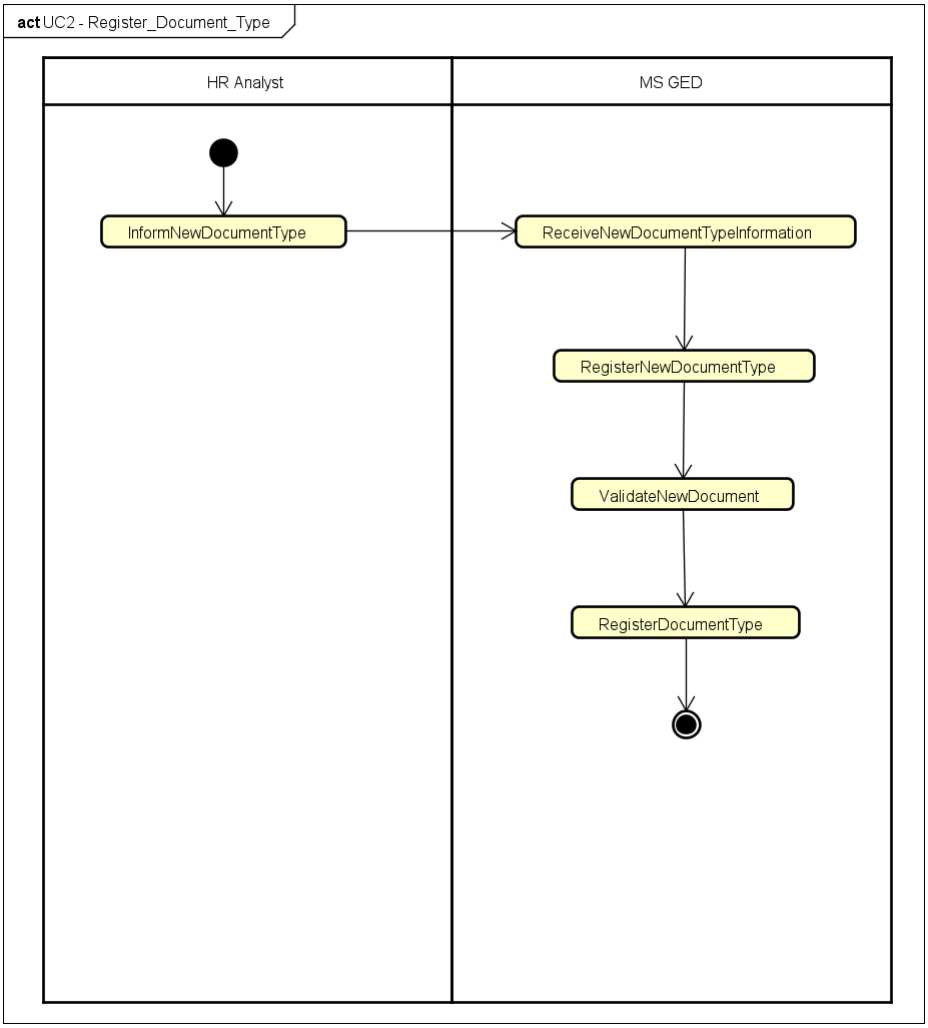


Figure 21 – SGED - UC2: Register Document Type

the SGED. Search ranges from the employee's documents to the history of medical certificates sent by the employee for validation. If the HR Manager considers it necessary and according to the business rule applied to the need at the time the search is carried out, he can generate a report with the data searched. UC3 is the use case that handles reporting by SGED.

For reporting, SGED sends data found in the search to the JS Reports microservice. This microservice will generate reports in three formats (PDF, PPTX, or DOCX) and send information back to SGED, which will make it available to the user actor who requested it. Figure 22 shows Activity diagram representing the described flow. Also according to Figure 22, the generated report is available for 48 hours in SGED, allowing the user actor who requested it to be viewed. After this period, the document is automatically deleted by SGED, respecting the business rules.

Use Case 5 (Approve Document) and 6 (Register Document) are Use Cases that complement each other. UC5, represented in Figure 23, deals with sending the medical certificate to the SGED. With a flow of activities similar to UC1, the SGED will receive the document and send it to the microservices responsible for extracting and analyzing data. The UC5, specifically,

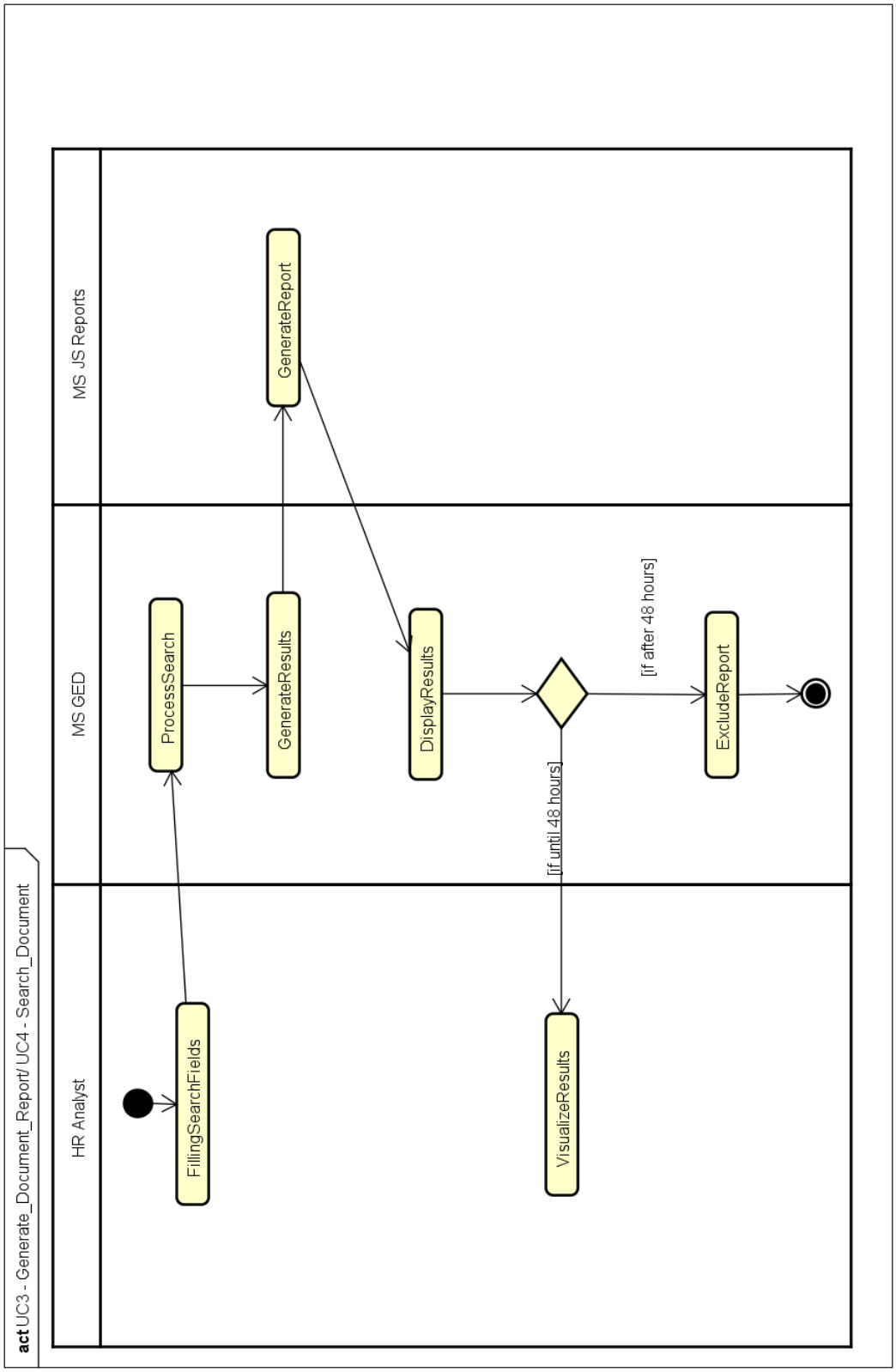


Figure 22 – SGED - UC3/UC4: Generate Document Report/ Search Document

will verify if the document sent is a medical certificate, verifying that it contains data that qualify it as such (patient's name, description of the reasons that led him to obtain the certificate, such as

the code of the International Statistical Classification of Diseases and Related Health Problems or ICD, doctor's stamp/signature and date). Similar to UC1, data is also indexed before it is sent for analysis.

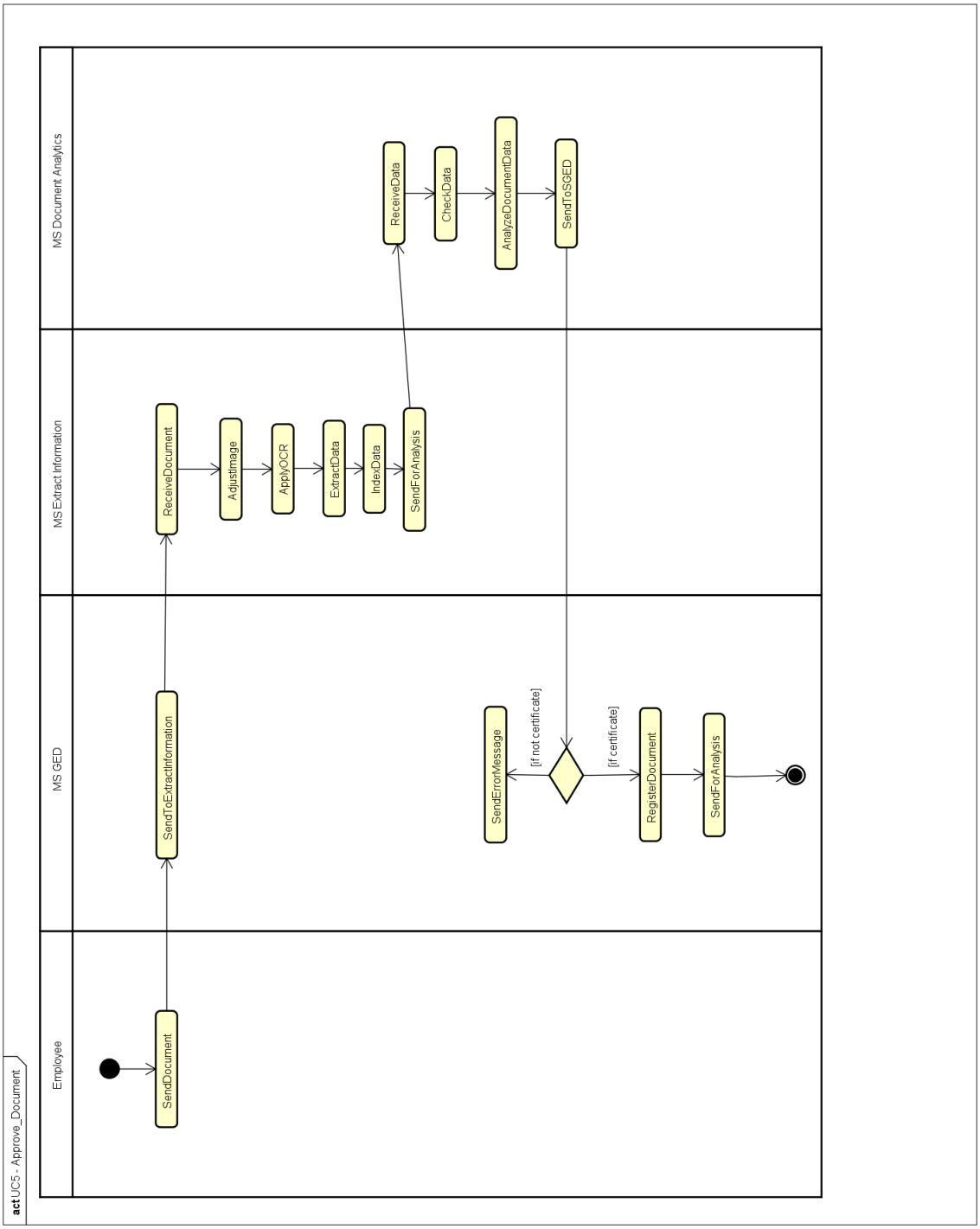


Figure 23 – SGED - UC5: Approve Document

If a document is within the imposed business rules, it is approved and sent for analysis, which will be carried out by the UC6. In turn, UC6, represented in Figure 24, after registration and approval made at UC5, will analyze the certificate according to deadlines set out in the

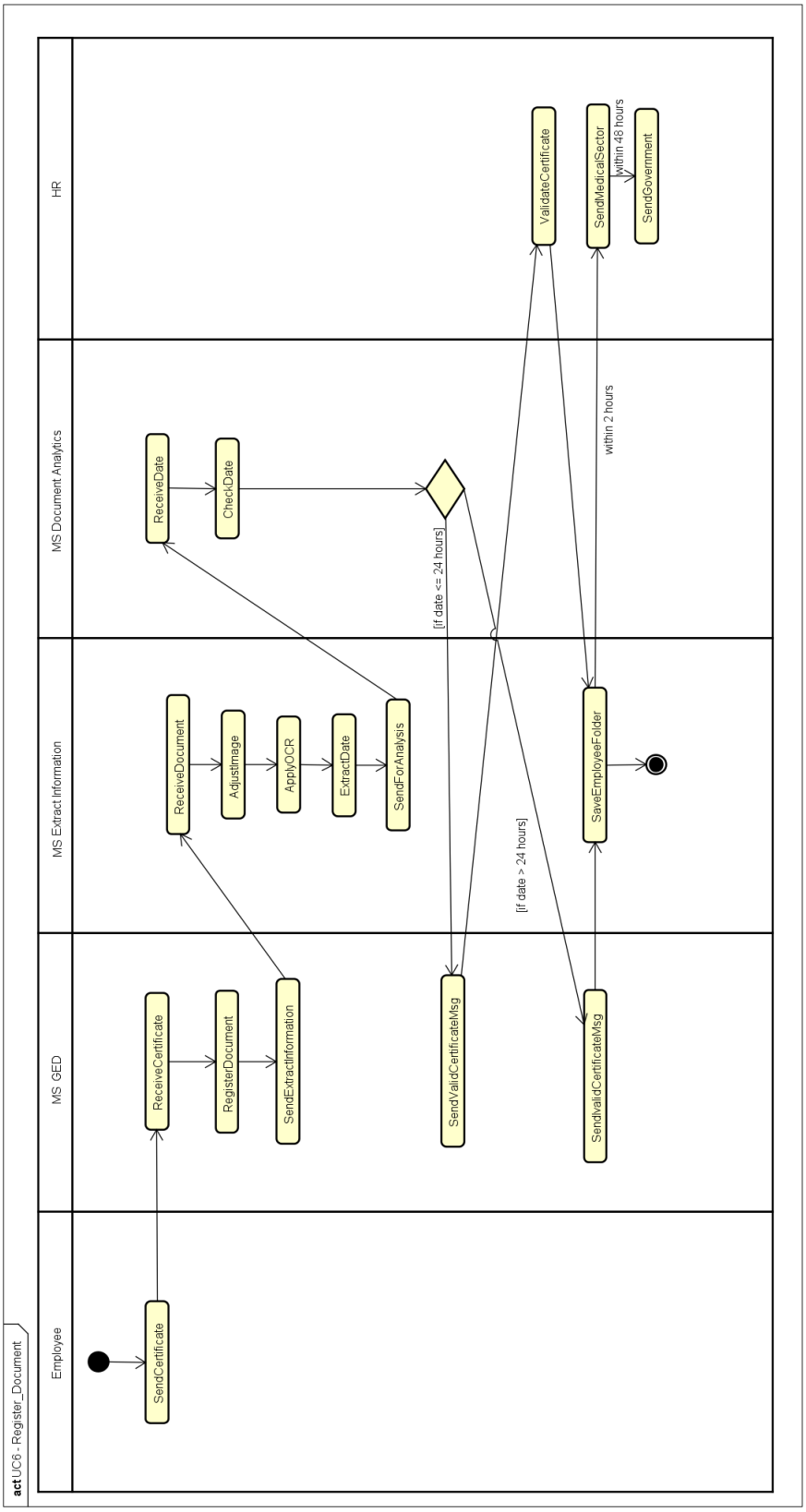


Figure 24 – SGED - UC6: Register Document

business rule for validating the medical certificate. After the employee sends it to the SGED and its approval is described in UC5 (in Figure 24 (UC6), UC5 (Figure 23) is represented by the

activity “RegisterDocument”), the certificate is sent for data extraction and analysis. In this UC, indexing is not necessary as the data received is already indexed. Flow in these activities is similar to that already described in UC1, with image adjustment, application of OCR and analysis of extracted data, performed by the Extract Information and MS Document Analytics microservices, respectively. In UC6 (Figure 24), unlike UC1, in which more data was extracted for analysis, Extract Information will identify and send date of document for analysis. The analysis of date is an important business rule, as medical certificate to be valid must be sent within 24 hours of its issuance.

MS Document Analytics analyzes the date of document, making a comparison with date of system and verifying that document respects this business rule. If it is valid, a notification is sent to SGED and the employee is alerted about document validity. After that, HR sector is informed and will comply with the other business rules, sending the document to medical sector within 2 hours and the medical sector will send it to Federal Government within 48 hours. If certificate does not meet the deadline of business rule, the employee will be alerted about document invalidity. In both cases, document will be saved in the employee’s functional folder.

UML Activity diagrams presented in this section were transcribed for Time Petri Nets, adding time constraints necessary for the correct functioning of the system to net transitions. The purpose of transcript is to analyze the behavior of business processes that use the microservices architecture, verify properties of models and simulate the behavior of modeled system, observing temporal constraints applied to the model.

4.2.1 Modeling using Time Petri Nets

In order to analyze the behavior of business processes that use the microservices architecture, as well as verify the properties of the models and analyze the time constraints of the SGED, the UML Activity diagrams were transcribed into Time Petri Nets. Transcribed models were presented in Section 4.2. Due to the large flow of activities represented in diagrams, the UC models were built through designing smaller models, so that they offer a detailed view of SGED. After building small models, for a complete representation of each UC, a bottom-up approach was used to produce the complete model. The union of the smaller models used the place fusion technique, joining places of similar tasks between the Petri Nets.

4.2.1.1 Use Case 1 models - Register Analyst

Figure 25 represents one of the UC1 models. The model represented in Figure 25 represents the SGED waiting state, while the document necessary to register the HR analyst in the system is not sent. In this model, all transitions have zero time constraints, indicating that they fire immediately. Therefore, the system will change the state immediately after sending a document, leaving the waiting state and going to the sending state to the data extraction microservice.

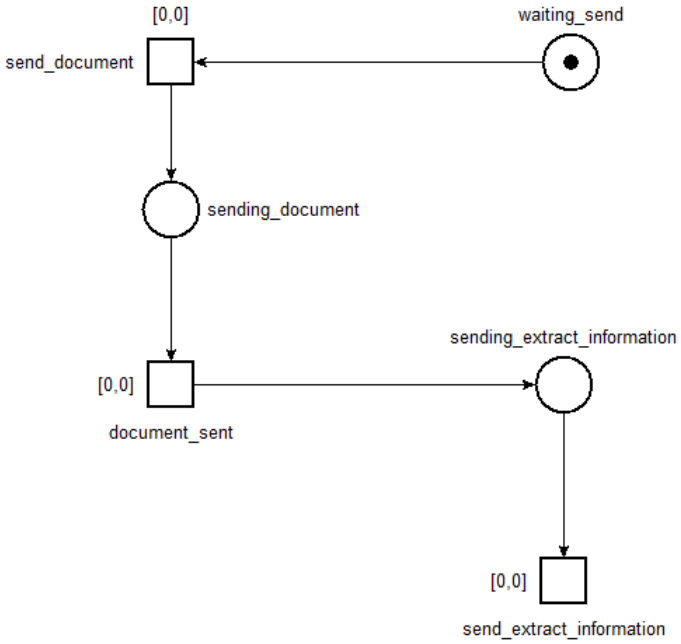


Figure 25 – UC1 - Model 1: SGED Wait State

An interesting aspect in Petri Nets is the possibility of modeling waiting states and adding temporal constraints, even if null. Time constraints are added as per business rules defined in business processes.

Still, with representation of UC1, Figure 26 shows the second model of UC1. In the model in Figure 26 it is possible to observe that there are non-zero temporal constraints. These restrictions mean that the system must perform described actions in place within the time limits defined in the Petri Net transitions. When the system completes that action, the transition is fired and the system state changes, that is, it will perform the next action defined in the model. Time intervals in transitions of all models presented in this work are represented in seconds. It can be observed in Figure 26, three transitions with time constraints: adjust_image [0,10], apply_OCR [5,300], and extract_data [0,30], meaning that the Extract Information microservice will have up to 10 seconds to adjust the image so that it is possible to extract necessary data, between 5 seconds and 5 minutes (300 seconds) to apply the OCR and recognize the characters and up to 30 seconds to extract necessary data to send for analysis, respectively. It is worth mentioning that the time data reported here were obtained through tests carried out with documents sent to the system.

If these restrictions are not respected, the system will detect a timeout and issue an error alert, that is, the system did not behave as expected. To represent abnormal situations in the behavior of a system, a mechanism called Watchdog is used (CARDOSO; VALETTE, 1997). Figure 27 represents the first UC1 watchdog model. The model shows temporal constraint violation added to transition adjust_image. In cases of a temporal constraint violation, timeout transition will be fired, indicating violation existence and marking will be considered “dead”,

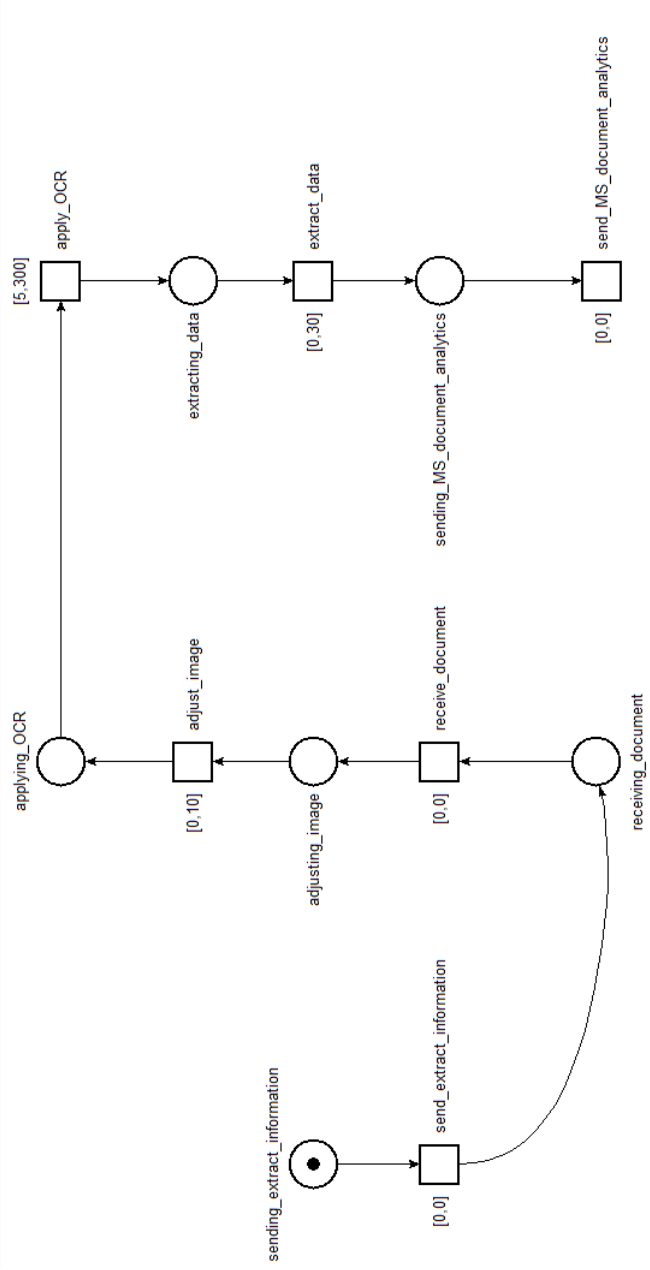


Figure 26 – UC1 - Model 2: Extract Information Microservice

being transferred to a place that represents an error.

In the watchdog case represented by Figure 27, if there is a temporal constraint violation of transition adjust_image, that is, if the Extract Information microservice takes more than 10 seconds to perform image adjustment, it will send this information to SGED, which will alert the user about the error.

Figure 28 represents second possible temporal constraint violation of Figure 26 model. Extract Information microservice should take between 5 and 300 seconds to recognize document characters so that it is possible to send data to be analyzed. If it exceeds this time, similarly to the watchdog in Figure 27, the temporal violation will be reported and the marking will be

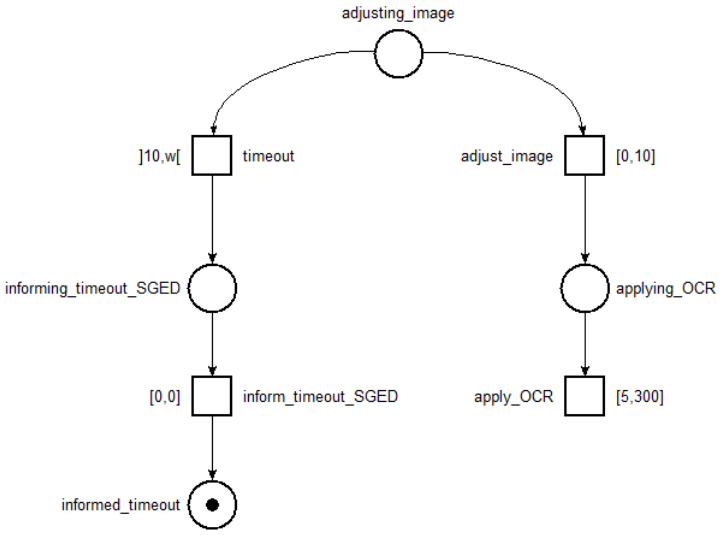


Figure 27 – UC1/UC5/UC6 - Watchdog 1: adjust_image

considered dead, being transferred to a place that indicates an error.

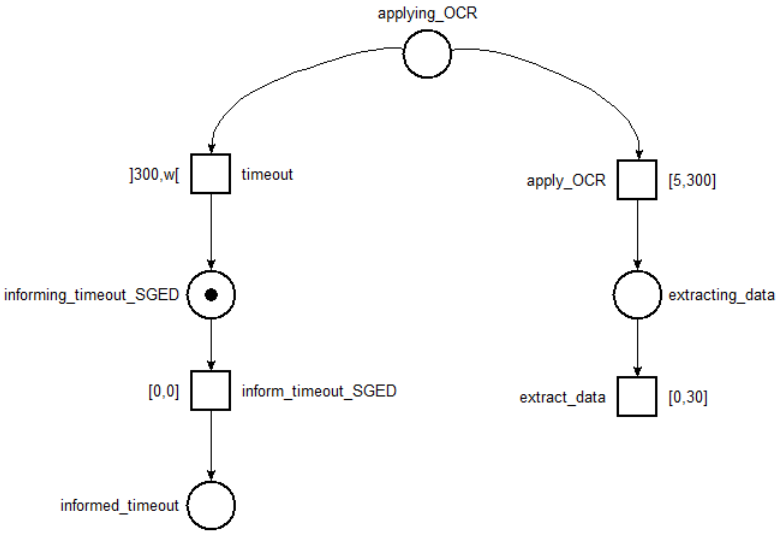


Figure 28 – UC1/UC5/UC6 - Watchdog 2: apply_OCR

Third watchdog of Figure 26 shows temporal violation in transition extract_data. If there is a violation of time constraint, that is, the Extract Information microservice takes more than 30 seconds to data extraction, it will fire a timeout and the information will be sent to SGED, transferring the dead marking to a place that represents an error. Figure 29 shows this temporal violation.

The third UC1 model shows the activity flow of the MS Document Analytics microservice, which, after receiving data extracted by Extract Information, performs the checking and verification of the data received, to validate the document according to business rules. Figure 30 shows this model representation.

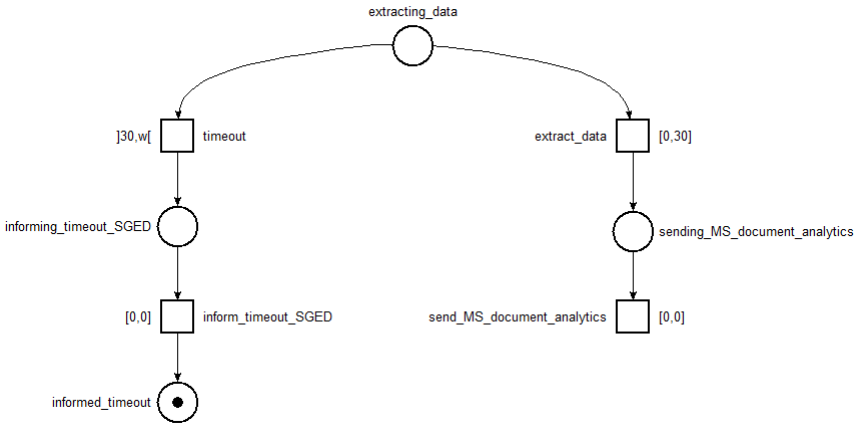


Figure 29 – UC1/UC5/UC6 - Watchdog 3: extract_data

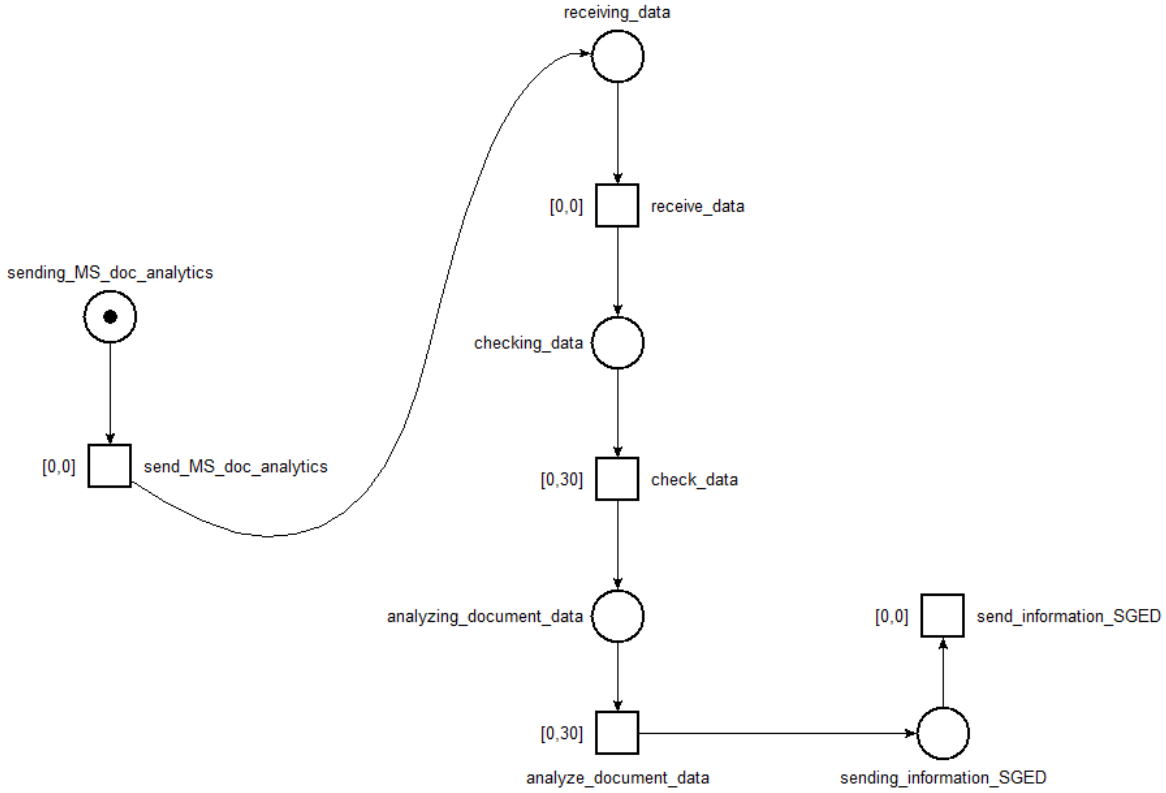


Figure 30 – UC1 - Model 3: MS Document Analytics

There are two significant time constraints in this model (Figure 30): transition check_data [0,30] and transition analyze_document_data [0,30]. Both transitions represent fundamental actions to comply with business rules, as the data analyzed comes from professionals who will validate other types of documents sent to the SGED.

Indexed data, that is, data that has already been filtered by MS Extract Information will be sent to MS Document Analytics, which will analyze such data. After analyzing data by MS Document Analytics, information will be sent to SGED, which will validate and subsequently

register the HR Analyst. As mentioned, it is possible to observe two temporal constraints in the model in Figure 30.

In both watchdogs, flow is similar to the ones already presented: a timeout is fired, indicating temporal constraint violation, and marking of that place is removed to a place that represents an unexpected system behavior. Figure 31 and Figure 32 shows the models of watchdogs.

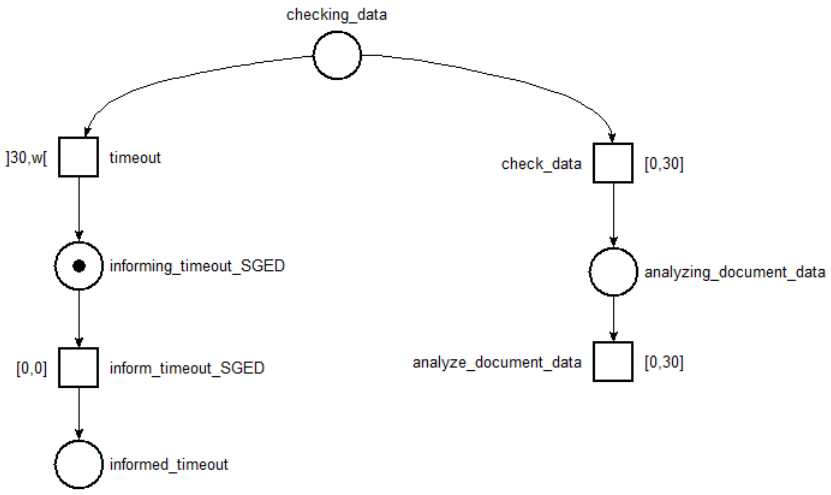


Figure 31 – UC1/UC5/UC6 - Watchdog 3: Check Data

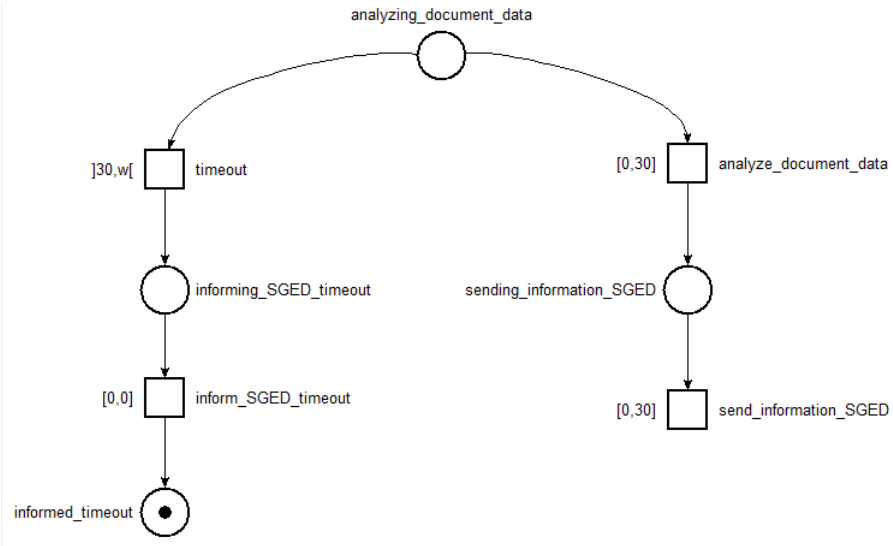


Figure 32 – UC1/UC5/UC6 - Watchdog 5: Analyze Document Data

The fourth and last UC1 model represents information reception sent by MS Document Analytics to SGED and sending this information to HR Manager to validate the HR Analyst's registration via SGED. For effectivity of registration, manager will have up to 24 hours to validate document sent, according to business processes. After this period, the document is automatically invalidated. Figure 33 represents the described model with time constraints.

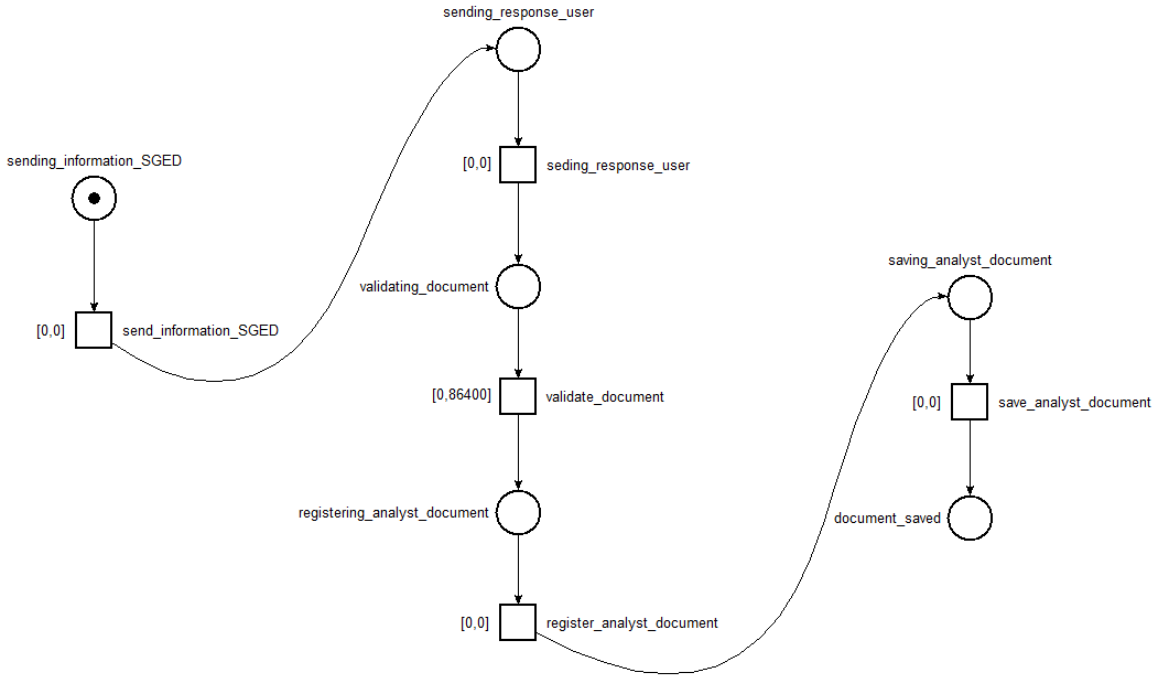


Figure 33 – UC1 model 4: SGED information reception and validate document

According to model in Figure 33, after receiving MS Document Analytics information, SGED will give a 24-hour deadline for document validation. Time constraint of transition validate_document represents time for validation, expressed in seconds. For the model in Figure 33, there is only one possible watchdog, referring to transition validate_document. If there is no validation by the HR Manager within the period stipulated in the business rules, the SGED will automatically invalidate the document, firing the timeout and not performing the registration of the HR analyst, being necessary to redo the procedure. Figure 34 represents the watchdog that corresponds to the described temporal violation.

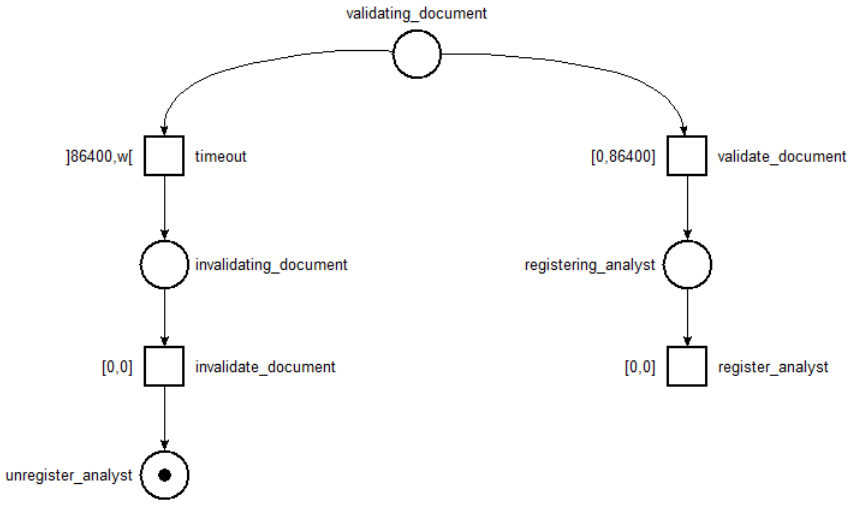


Figure 34 – UC1 watchdog 6: SGED invalidate document

Finishing the sequence of models referring to UC1, Figure 35 shows the complete model of this Use Case, with all analysis and data extraction processes separated by actors. Figure 35 also shows the sending and receiving of data by the microservices that work with the SGED, complying with business rules represented by transitions with time restrictions.

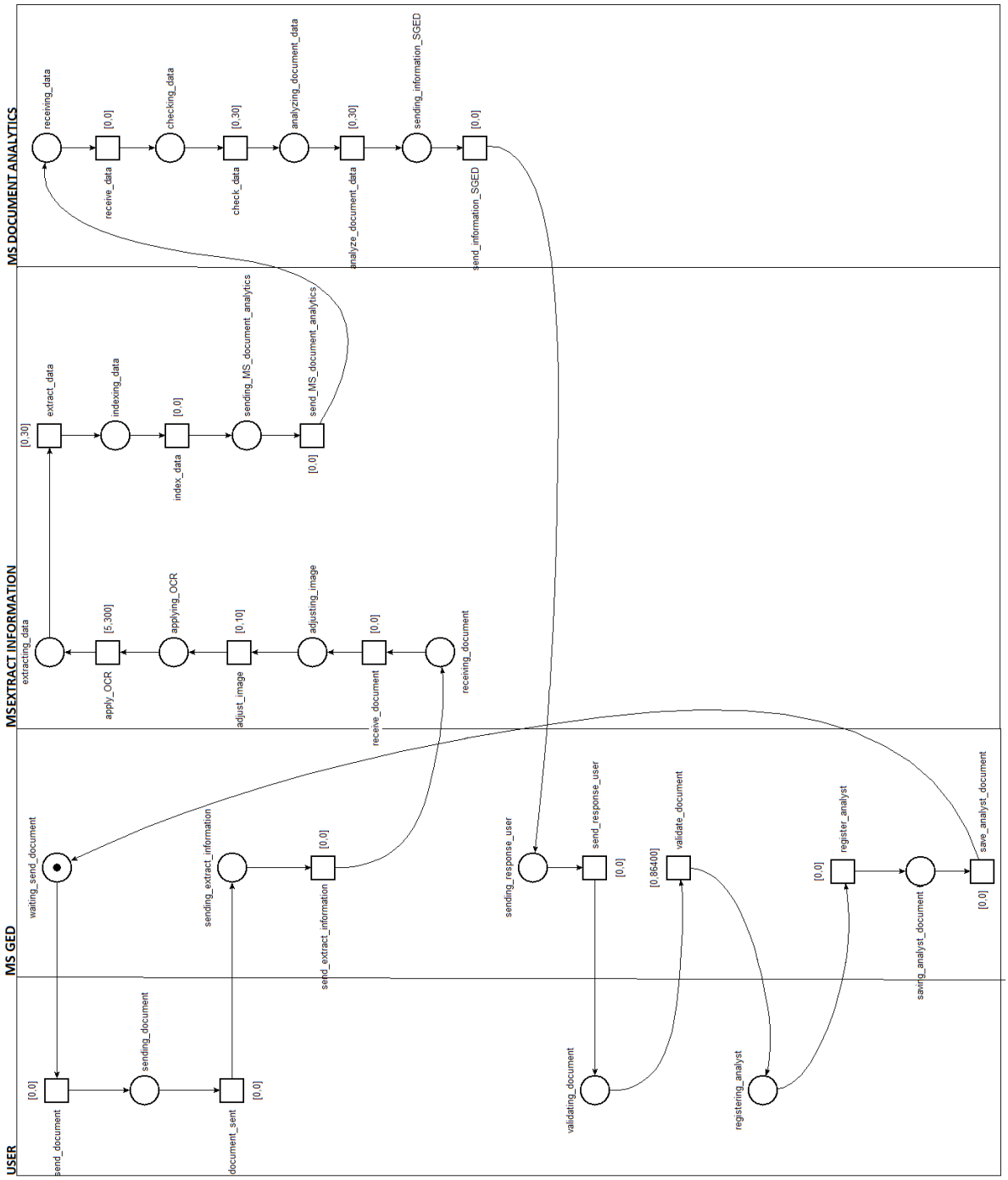


Figure 35 – UC1 Complete Model

It is relevant to note in Figure 35 that after extracting process, data analyzing, and validating, in this use case system will return to the waiting state, being available for sending a new document for registration.

4.2.1.2 Use Case 2 models - Register Documthe ent Type

UC2, despite being a simple model and smaller in size than others, was also divided into smaller models to facilitate the understanding of its functionalities, as well as to approach them completely.

The first model of this UC is shown in Figure 36. The model in Figure 36 represents the waiting state, while Microservice (MS) GED waits for the user to send a new document that will be accepted for the system. The model also shows the moment when the user (HR Analyst) will send information about which type of document will be registered.

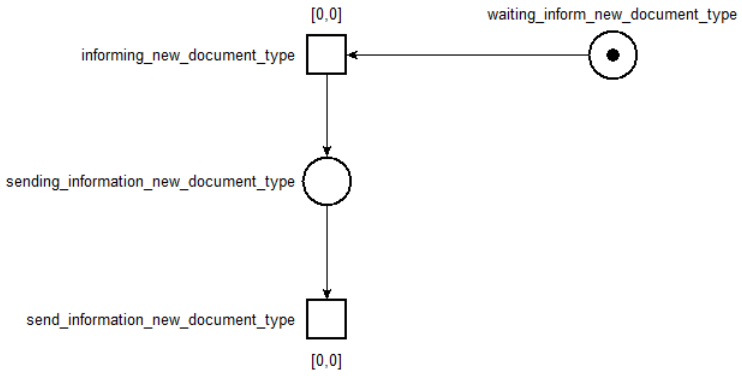


Figure 36 – UC2 model 1 - Waiting State and New Document Information

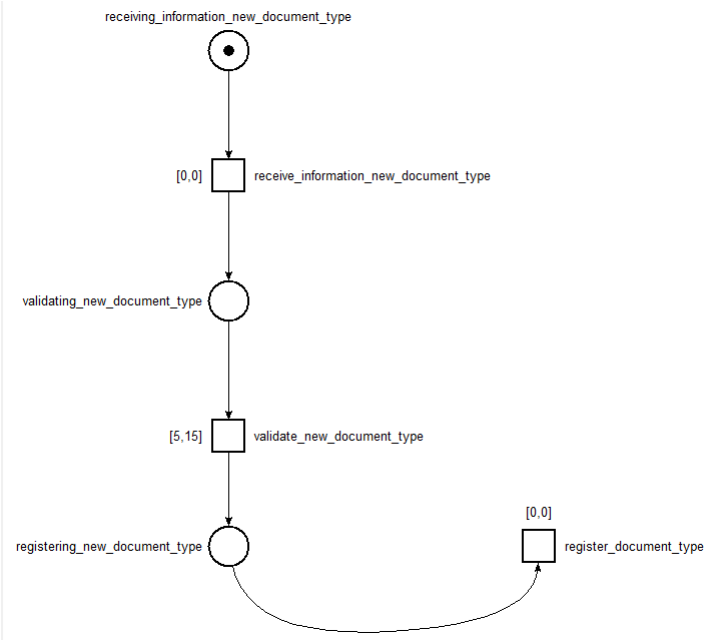


Figure 37 – UC2 model 2 - Receive Information and Validate Document Type

Second model of the UC2 is shown in Figure 37. Model shown in Figure 37 presents reception of information by the SGED. In this model, there is a temporal constraint added to

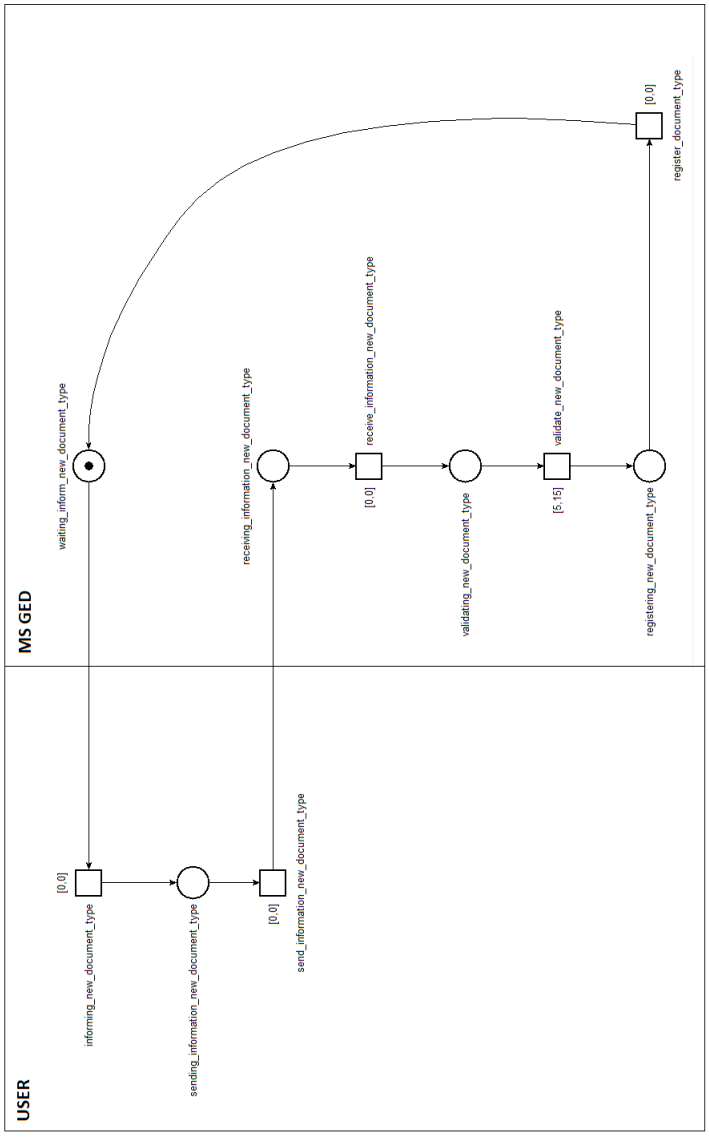


Figure 38 – UC2 Model Complete

transition validate_new_document_type, which contains a time interval [5,15]. This means that the processing of information sent by the user to the system should take between 5 seconds and 15 seconds to be processed and later registered in the system.

If validation of information sent exceeds the 15-second limit, a timeout is fired, as shown in the watchdog in Figure 39. After firing a timeout, an error message is sent to the user and it is necessary to restart the registration of a new type of document.

For modeling UC2 complete model, shown in Figure 38, a bottom-up approach was used in conjunction with the Petri Nets fusion technique called place fusion, as well as a separation of functions of each actor that composes this use case. Such approaches allow for a better understanding of the modeled functionality, as well as providing a complete view of each use case.

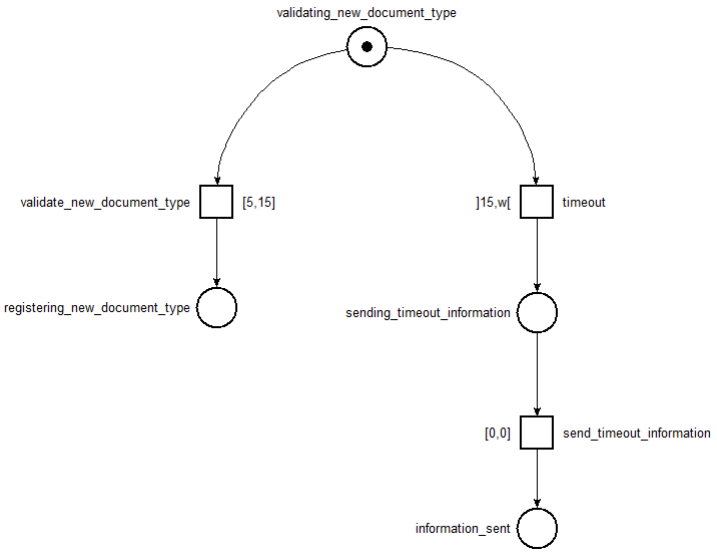


Figure 39 – UC2 watchdog - Validating Document Type

4.2.1.3 UC3/UC4 - Generate Document Report/ Search Document

Use cases presented in this subsection, UC3 - Generate Document Report and UC4 - Search Document, are interdependent use cases. Because they are interdependent, their models were unified, so that their flow of execution and functioning could be better understood. These use cases work in conjunction with the JS Reports microservice, which performs report generation.

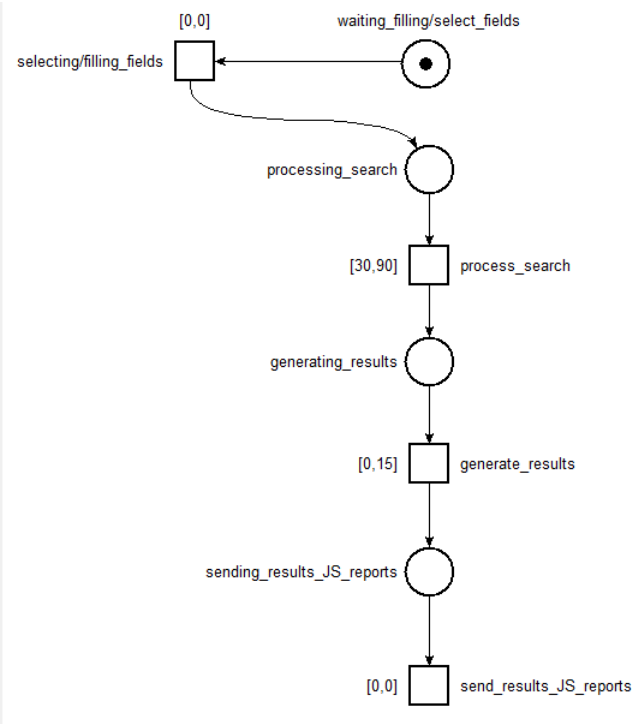


Figure 40 – UC3/UC4 model 1 - Process Search/ Generate Results

First model of UC3/UC4 is shown in Figure 40. The model shows the waiting state, while

the system waits for the filling or selection of search fields so that this search can be processed with items desired by the HR Analyst. This model contains two temporal constraints: transition process_search [30,90] and transition generate_results [0,15]. To process the search according to what the user wants, SGED will have a minimum time of 30 seconds and a maximum time of 90 seconds. After that, a timeout is fired and an error alert is sent to the user, is necessary to redo the search.

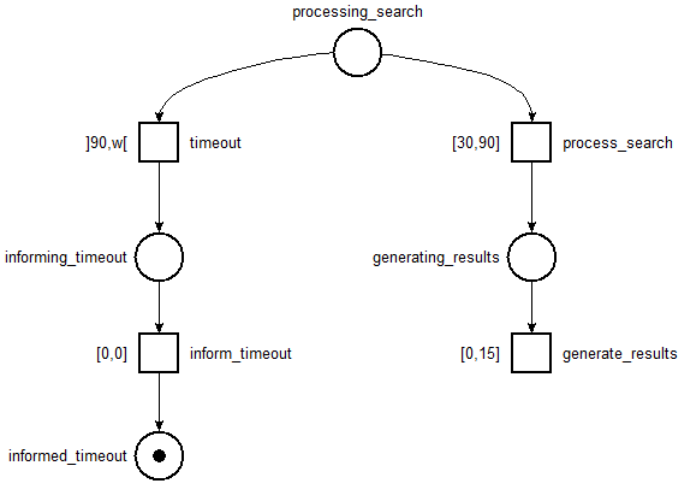


Figure 41 – UC3/UC4 watchdog 1 - Process Search Time Constraint Violation

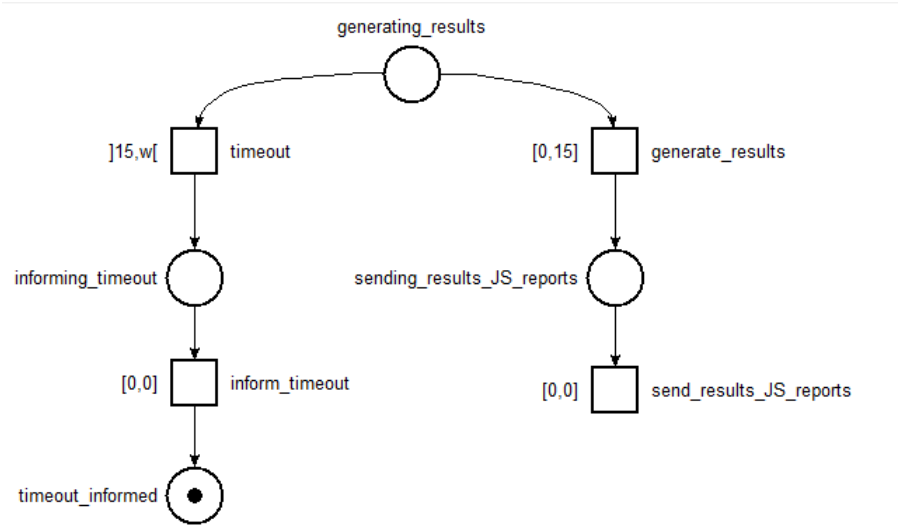


Figure 42 – UC3/UC4 watchdog 2 - Generate Results

To deal with temporal constraint violation of transition process_search, watchdog in Figure 41 was modeled. Similar to other watchdogs already presented, a timeout is fired and an error alert is sent to the user. The treatment for temporal constraint violation of transition generate_results is shown in the watchdog of Figure 42. When there is no temporal constraint violation of transition generate_results, the search is processed and its results are sent to the JS

Reports microservice for report generation. When a violation occurs, a timeout is fired and an error alert is sent, similarly to other temporal constraint violation treatments

Second UC3/UC4 model, shown in Figure 43, corresponds to report generation flow by MS JS Reports. In this model, only a temporal constraint on transition generate_report is observable. This restriction means that MS JS Reports will have between 60 seconds and 600 seconds, that is, between 1 and 10 minutes, to generate a report and make it available to the user. In the expected execution flow, the report will be sent to SGED and made available to the user in three file formats: DOCX, PPTX, and PDF. If there is a time constraint violation, shown in the watchdog of Figure 44, a timeout is fired that will generate an error alert. It is worth mentioning that the time constraint added to transition generate_report creates an asynchronous queue of requests for generating reports, thus preventing the system from crashing and the consequent impossibility of using the same or other SGED functionalities.

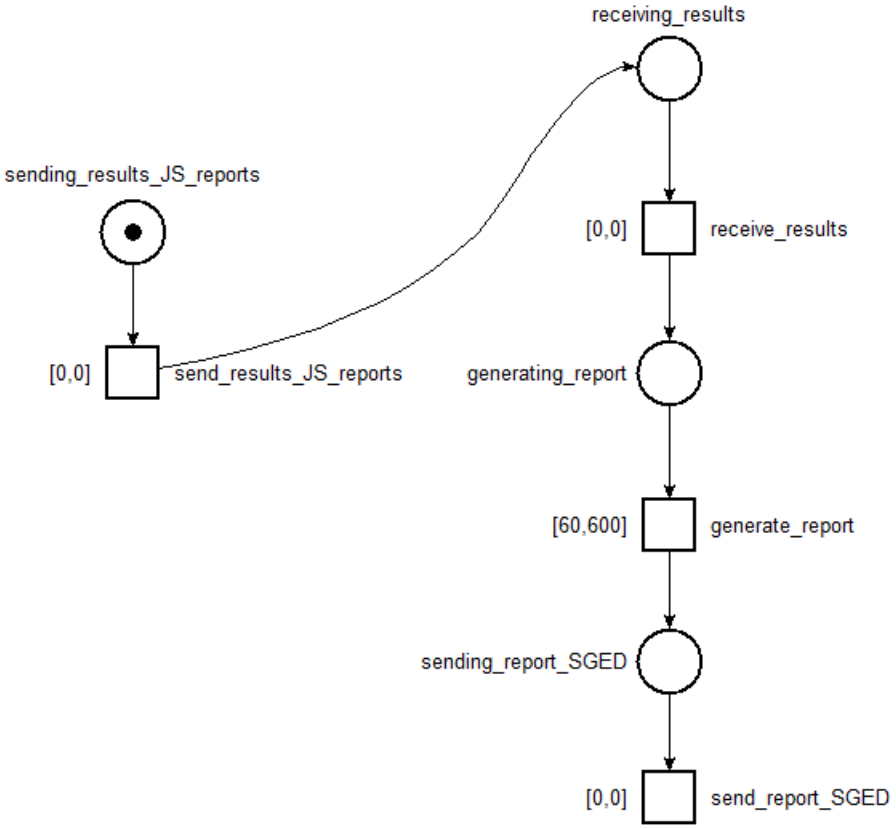


Figure 43 – UC3/UC4 Model 2 - JS Reports Generate Report

The third and last model of UC3/UC4 shows report receipt by SGED. At this stage, the report will be available in three formats, as already mentioned. Model is shown in Figure 45. In this model, there is a significant temporal constraint on transition display_report. When generated, the report will be available to the user for up to 48 hours (172800 seconds) for visualization. After this period, SGED automatically deletes the report. Report deletion is represented in watchdog in

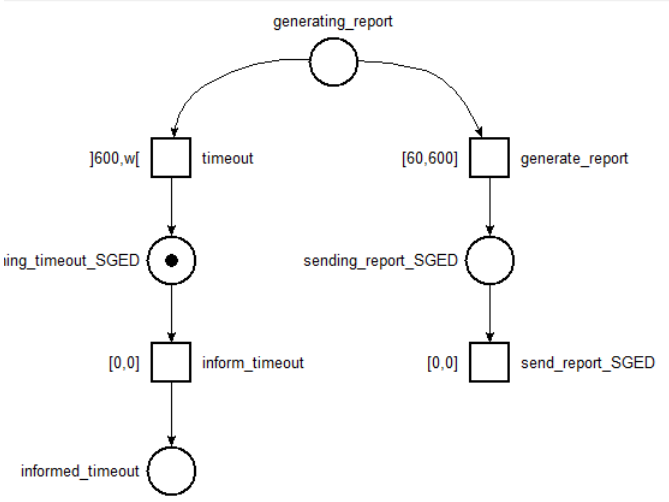


Figure 44 – UC3/UC4 Watchdog 3 - Generate Report

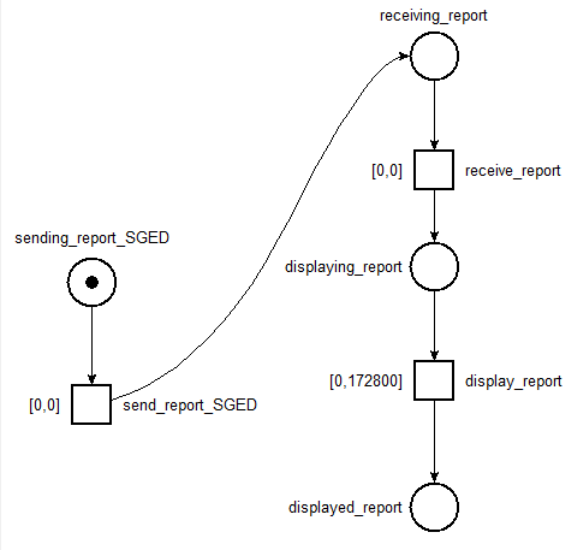


Figure 45 – UC3/UC4 Model 3 - SGED Reception Report and Displaying Report

Figure 46. This restriction is part of the business rules, which carry out a periodic cleaning, thus avoiding an accumulation of documents that will no longer be needed after a certain period.

Complete model of UC3/UC4 is shown in Figure 47. It was modeled using a bottom-up approach and place fusion technique to unite places of similar tasks, resulting in a complete model that addresses all necessary characteristics of UC. The roles of each actor are also detailed in the model on Figure 47.

4.2.1.4 UC5 - Approve Document

UC5 - Approve Document is a bigger use case than others since the flow of activities involves more steps and also another use case, UC6. Microservices that run alongside UC5 are the same as those that run on UC1, for data extraction and analysis, as well as their watchdogs.

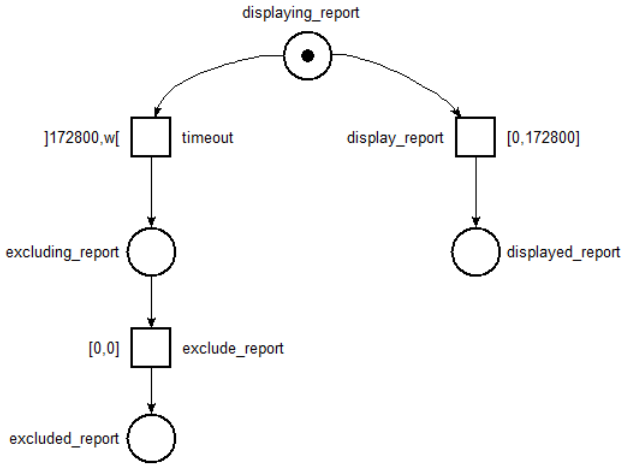


Figure 46 – UC3/UC4 Watchdog 4 - Exclude Report

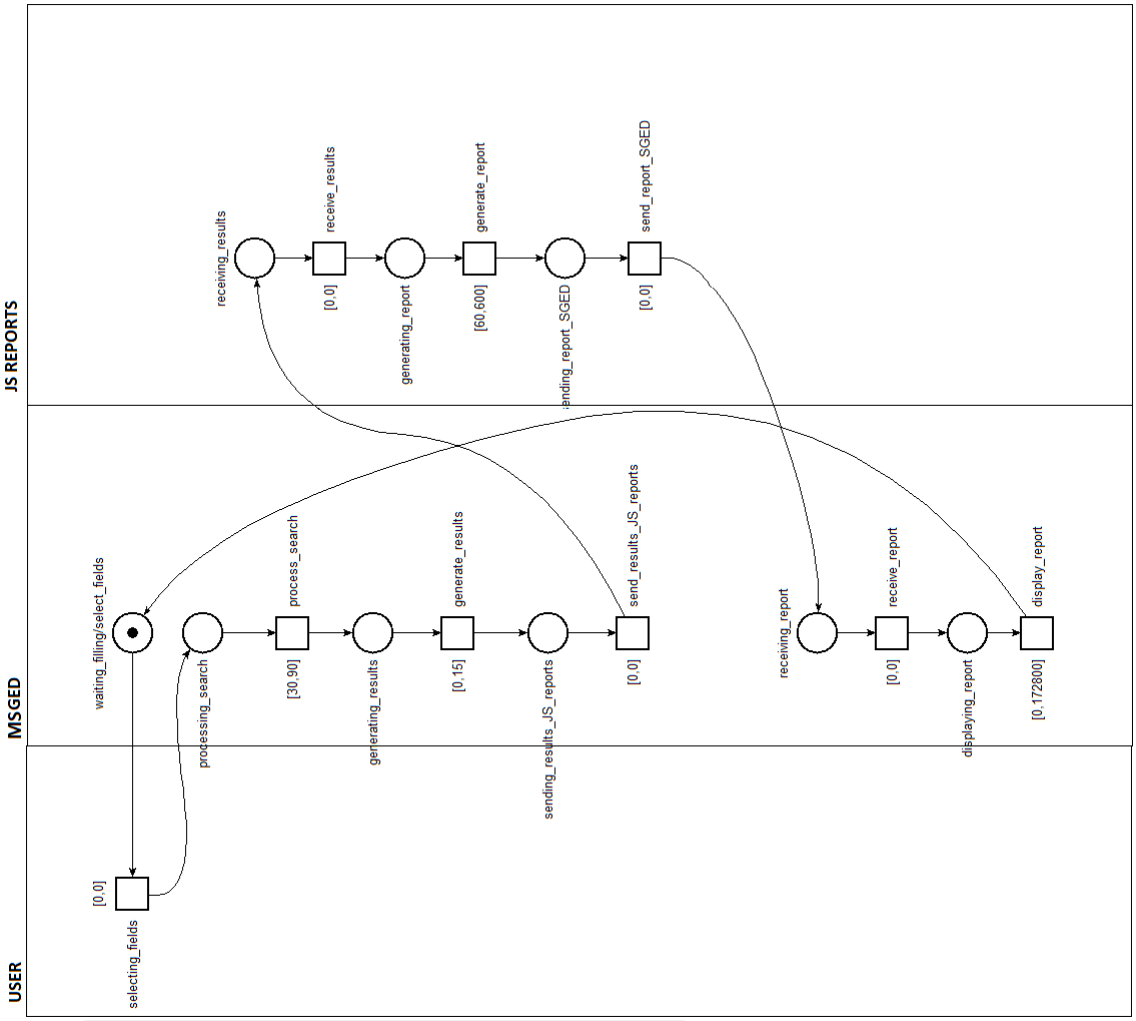


Figure 47 – UC3/UC4 Complete Model

Models are shown in Figure 25, Figure 26 and Figure 30. Their respective watchdogs are shown in Figure 27, Figure 28, Figure 29, Figure 31 and Figure 32.

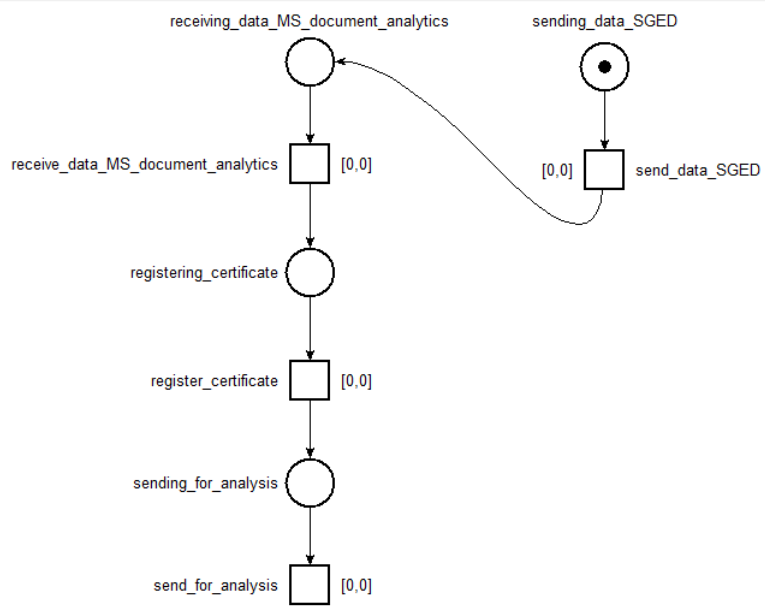


Figure 48 – UC5 - Reception and Register Certificate

The model that is specific to UC5 concerns the reception of information received by MS Document Analytics, recording medical certificates, and sending to UC6, which deals with the validation of the medical certificate. Model is presented in Figure 48. In the model in Figure 48, it is observed that there are only null constraints, indicating that transitions are fired instantaneously. The medical certificate is recorded in the system and then sent for validation, carried out at UC6.

Complete model of UC5 is shown in Figure 49. In this model are all activity flows, with the action of MS Extract Information and MS Document Analytics and their respective time constraints that meet business rules. The roles of each actor are also specified in the model.

4.2.1.5 UC6 - Register Document

UC6 - Register Document is the last UC of the SGED. Its flow of activities is extensive, encompassing the entire UC5. Due to the considerable size of the two diagrams,it was decided to do not unify UC5 and UC6, to facilitate visualization of their activity flows. Similar to the UCs already presented, UC6 was divided into smaller models to encompass all its aspects.

The first model of UC6 is shown in Figure 50 and represents the waiting state, while the system waits for the employee to send his medical certificate for analysis. SGED receives the certificate and registers the document. Registration of medical certificate process is performed according to UC5 described in Subsection 4.2.1.4 and is represented by transition register_document in Figure 50.

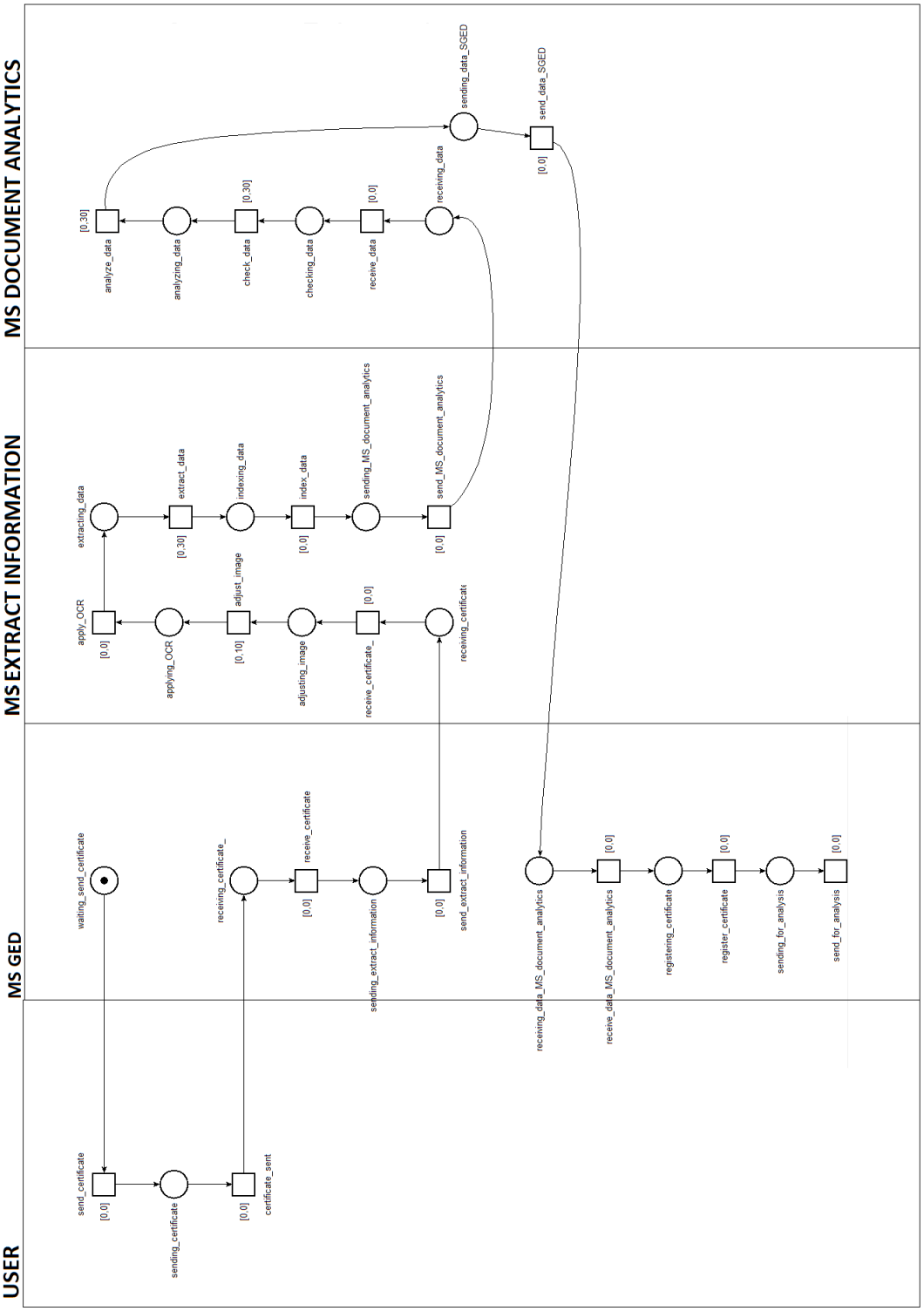


Figure 49 – UC5 Complete Model

After receiving and processing the document, it is forwarded to MS Extract Information. Specifically for this UC, MS Extract Information will extract the date from the document, as it will serve to validate or invalidate the medical certificate, according to business rules. This activity flow is available in Figure 51.

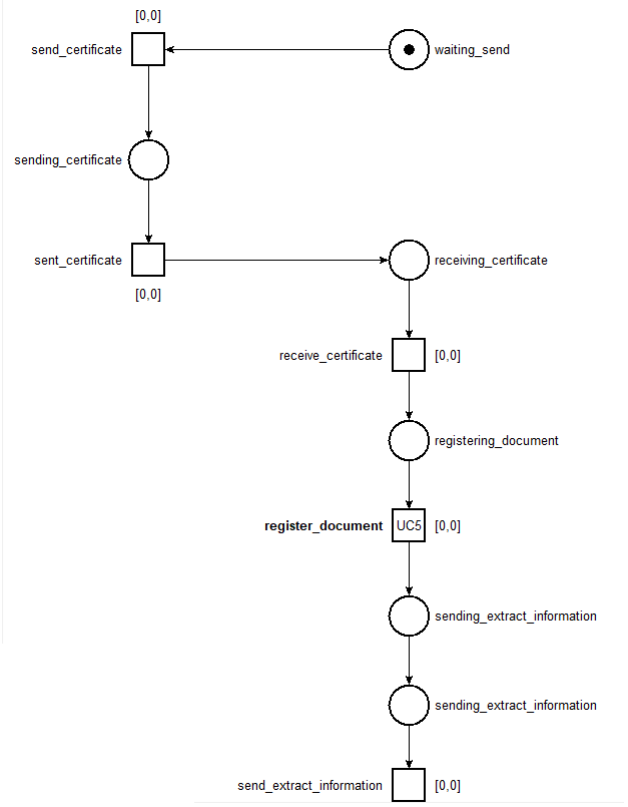


Figure 50 – UC6 model 1 - Waiting State

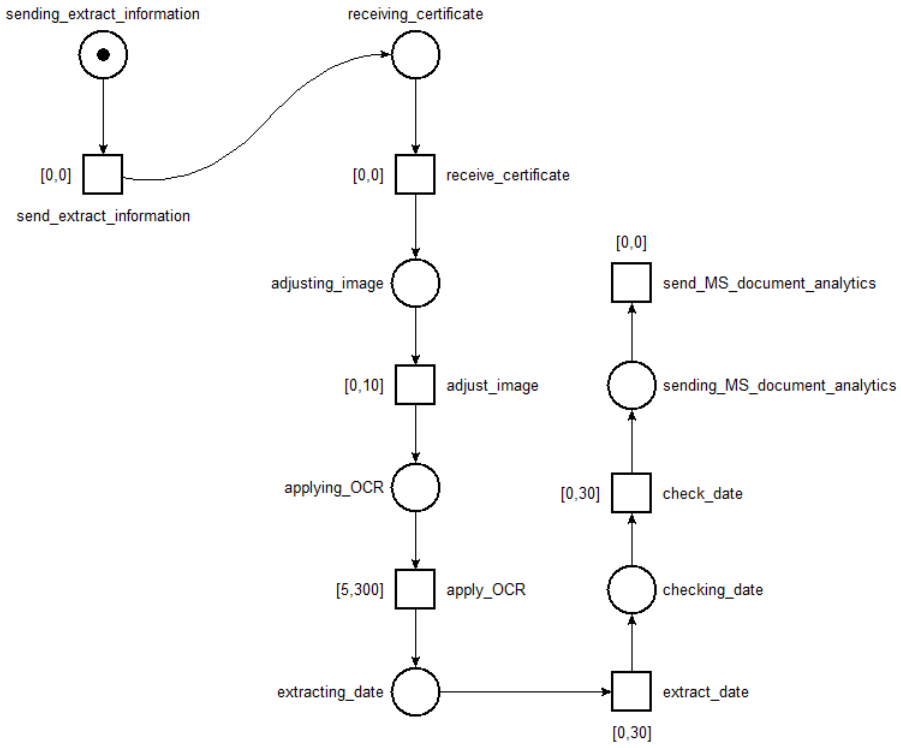


Figure 51 – UC6 model2 - Extract Date

However, although the extraction is only one specific data as shown in Figure 51, the processing is similar to those already presented, as well as the watchdogs referring to this part. The watchdogs are represented in Figure 27, 28, 29 and 31.

Model 3, shown in Figure 52, shows the processing of receiving data extracted by MS Extract Information by MS Document Analytics, which cleans and analyzes data. In addition to receiving data, MS Document Analytics also captures the date of the system and compares it with the received date, for purpose of validating medical certificates according to business rules.

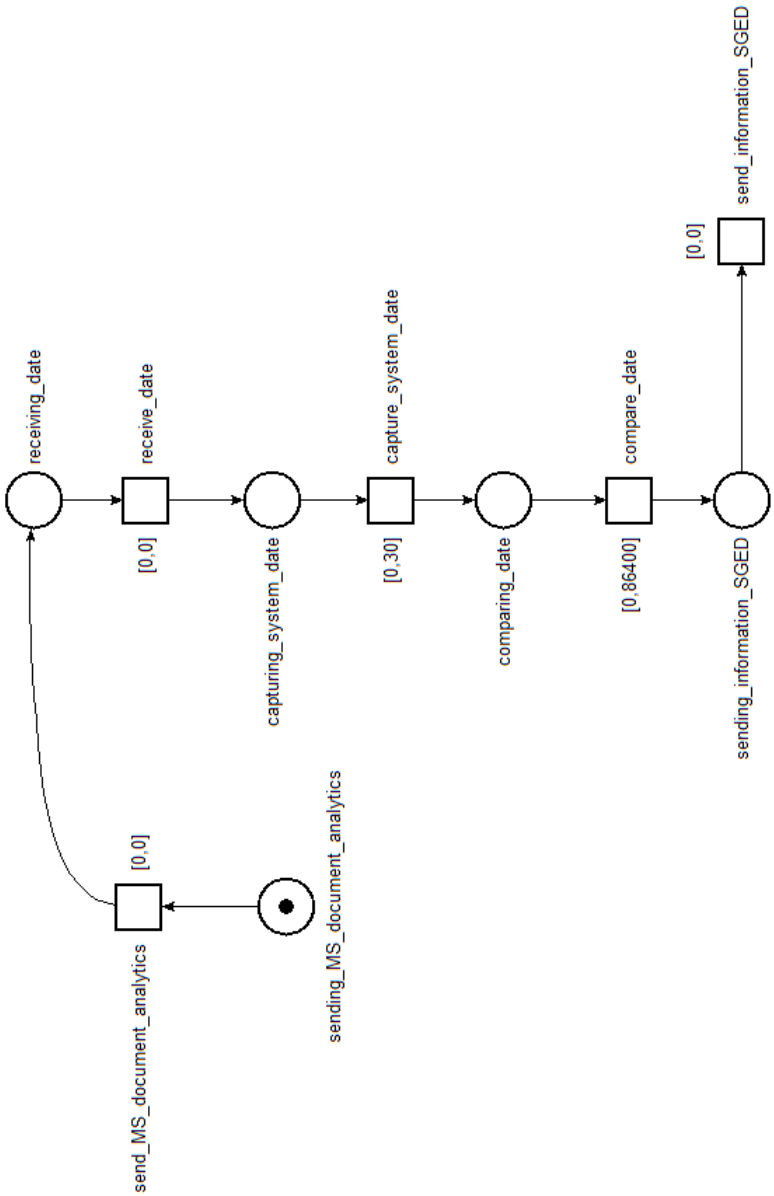


Figure 52 – UC6 model 3 - Compare Date

The first temporal restriction of the Figure 52 model concerns capturing the system date, for purposes of comparison with the date extracted from the medical certificate. MS Document

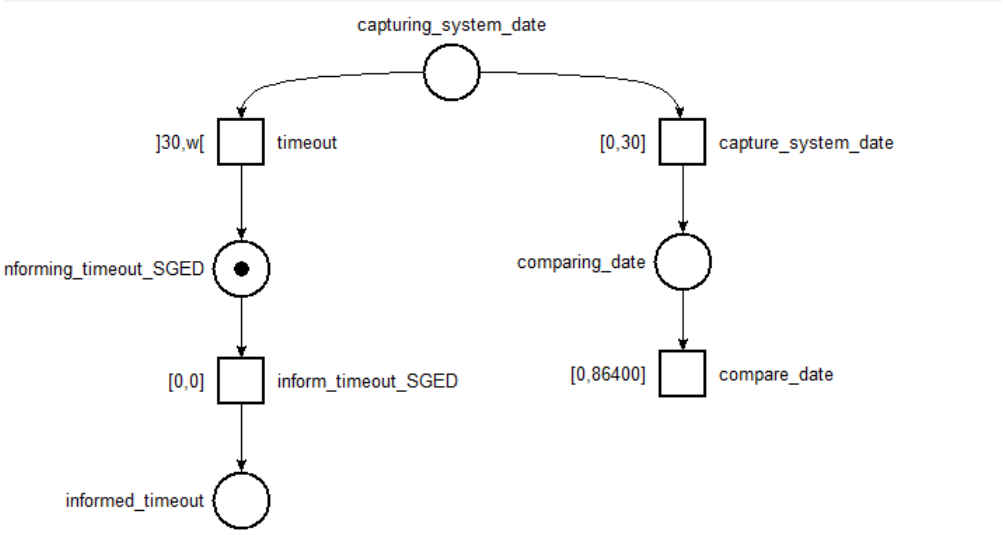


Figure 53 – UC6 watchdog - Capture System Date

Analytics will have up to 30 seconds to capture and compare the dates. Violation of this time restriction is represented in the watchdog in the Figure 53. The second temporal constraint of the model in Figure 52 concerns the validity period for the medical certificate, which is 24 hours after issuance, according to business rules. If the certificate is within 24 hours, information is sent to SGED for validation. If the medical certificate is outside the 24 hours, a timeout is fired with a document invalidity alert and information is sent to the SGED, which in turn saves the document in the functional folder of the employee. Violation of this time constraint is represented in Figure 54.

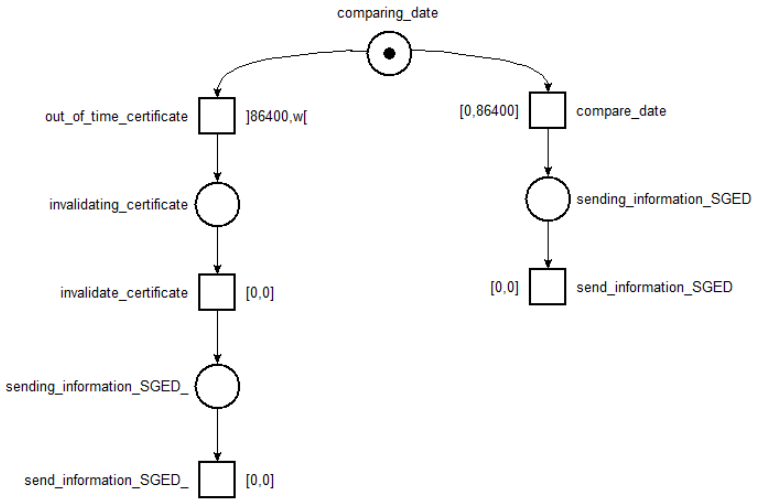


Figure 54 – UC6 watchdog - Compare Date

Model 4 of the UC6 is shown in Figure 56 and concerns the action of saving the certificate sent, regardless of whether is valid or not, in the employee's functional folder. Save in folder

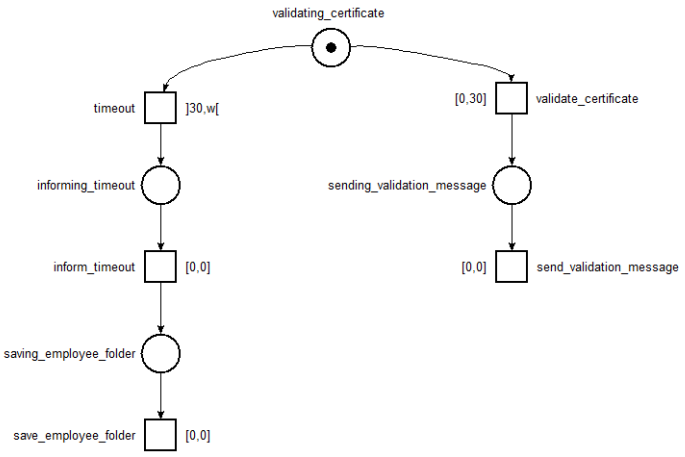


Figure 55 – UC6 watchdog - Validate Certificate

action is part of the organization's business rules. In this model, there is a single temporal constraint on transition validate_certificate. SGED must perform validation of medical certificate within 30 seconds. If this does not occur, a timeout will be fired and an error alert will be sent, as shown in Figure 55.

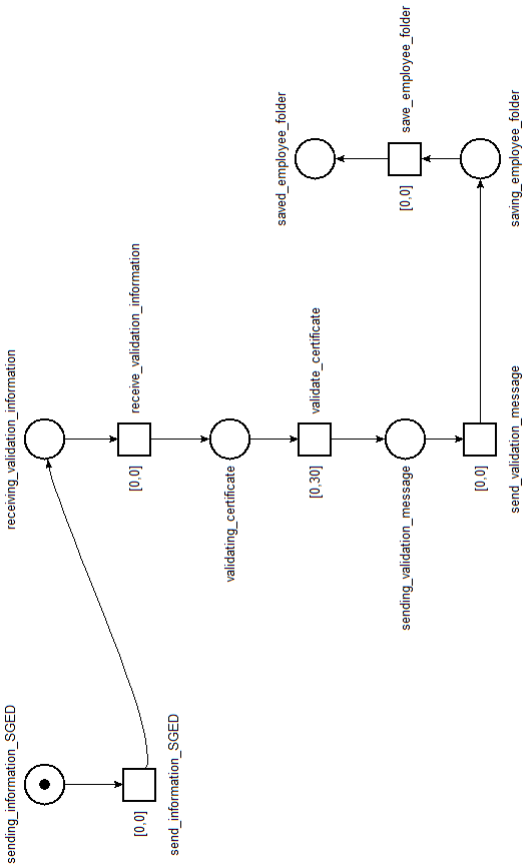


Figure 56 – UC6 Model 4 - Validate Certificate and Save Employee Folder

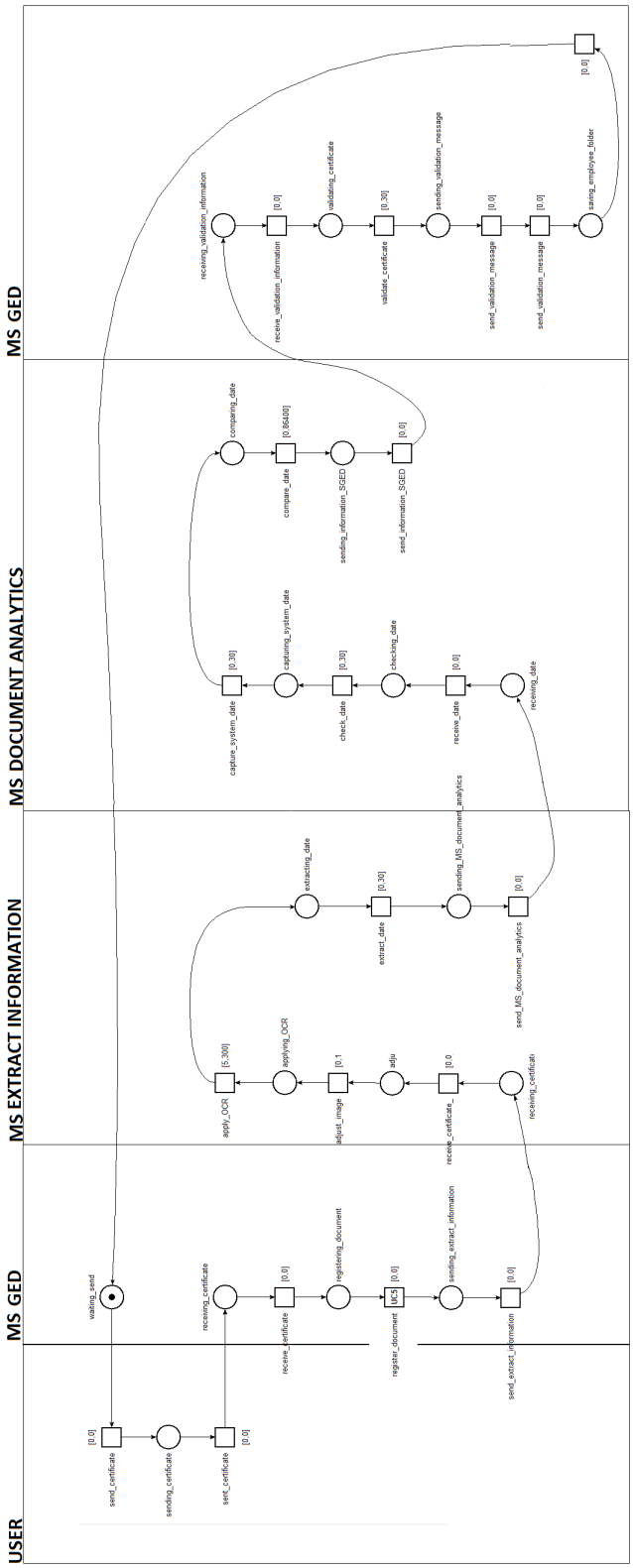


Figure 57 – UC6 Complete Model

To conclude UC6, Figure 57 shows the complete model, modeled with the union of the smaller models presented. In this model, actions are separated according to the actor who performs them. Transition register_document represents all processing performed on the UC5.

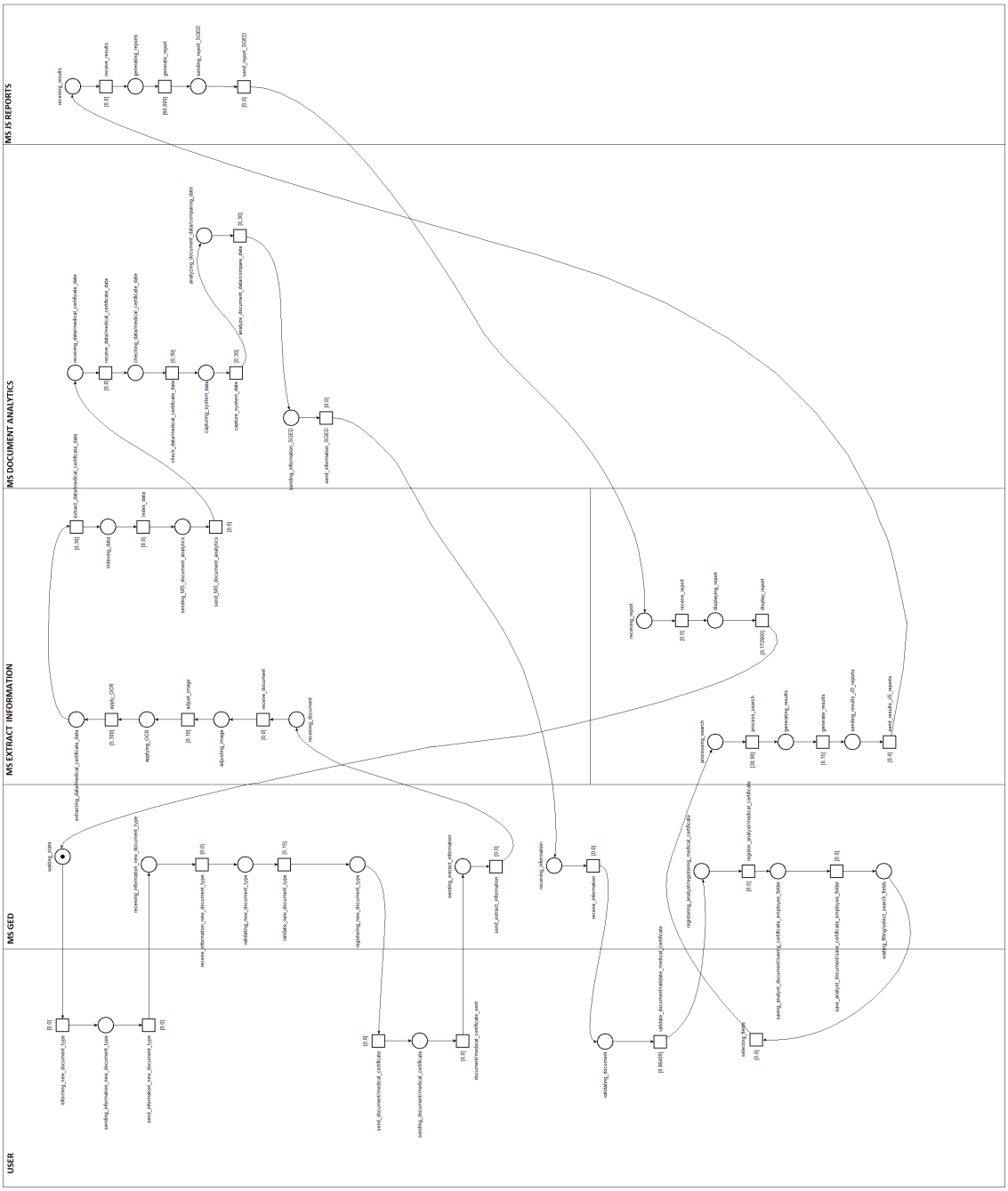


Figure 58 – SGED Complete Model

4.3 Results

After modeling in Petri Nets of each UC, the properties of each model were analyzed individually, as well as the properties of the complete model. The complete SGED model is shown in Figure 58. For modeling each Use Case and analyzing good properties of Time Petri Nets, Time Petri Nets Analyzer (TINA) tool, version 3.6.0 - 64 bits, was used.

The property verification method used is the space state graph. The state space graph

maps all possible states of the models, demonstrating if there is any unavailable or unreachable space. State space analysis also shows if the model is as desired, that is, if Petri Net markings are reaching everywhere. Furthermore, the state space graph proves the analyzed properties. Arcs of state space graphs represent the firing of transitions. States are numbered and represent places to which marking is moved after each net transition fires. For all models, analyzed and verified properties were liveness, boundedness, and reversibility.

Liveness property refers to the complete absence of deadlocks, being possible to fire any transition of model progressing through any additional fire sequence.

Boundedness concerns the limitability of Petri Net. If a number k of tokens present in one or more places in the net is finite, the net is considered k-bounded or simply bounded. Net is said secure if it is $1 - bounded$. All models presented are $1 - bounded$ and considered safe.

Reversibility is a property that concerns the ability of the net to return to its initial state after any firing sequence. In many models, there is no need to go back to its initial state. In an SGED case, only one of the models does not meet this property, as it is not necessary for its need.

For UC1 (Figure 35), was found that the net meets the liveness, boundedness, and reversibility properties, as shown in the TINA tool verification report shown in Figure 59. Its respective state space graph is shown in Figure 64. Also according to the analysis shown in Figure 59, the model is composed of 17 places and 17 transitions, being interconnected by 34 arcs. The model also has 17 states, with all states considered "live", that is, reachable.

digest | help

places	17	transitions	17	net	bounded	Y	live	Y	reversible	Y

abstraction	count	props	psets	dead	live
states	17	17	17	0	17
transitions	17	17	17	0	17

Figure 59 – UC1 Good Properties

digest | help

places	5	transitions	5	net	bounded	Y	live	Y	reversible	Y

abstraction	count	props	psets	dead	live
states	5	5	5	0	5
transitions	5	5	5	0	5

Figure 60 – UC2 Good Properties

digest | help

places	9	transitions	9	net	bounded	Y	live	Y	reversible	Y

abstraction	count	props	psets	dead	live
states	9	9	9	0	9
transitions	9	9	9	0	9

Figure 61 – UC3/UC4 Good Properties

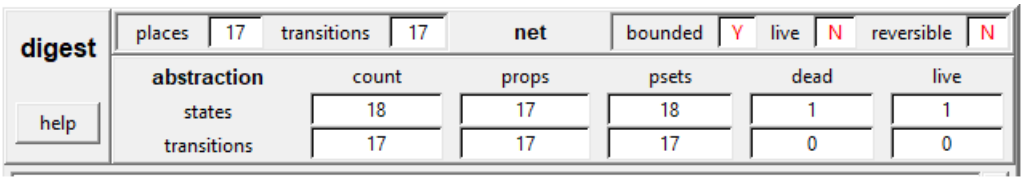


Figure 62 – UC5 Good Properties

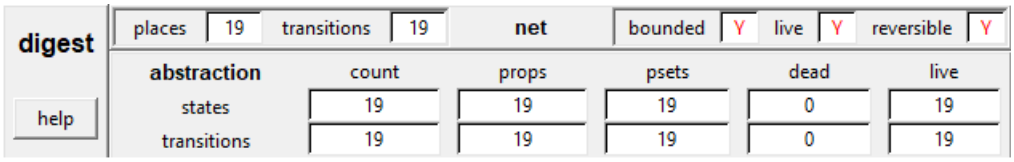


Figure 63 – UC6 Good Properties

For UC2 (Figure 38), was found that the model in Petri Net meets the liveness, reversibility, and boundedness properties, as shown in Figure 60. Also according to Figure 60, this model has 5 places and 5 transitions that are interconnected by 10 arcs. This model also has 5 possible states and has no "dead" transitions, that is, transitions that do not transfer marking any model place. State space graph is shown in Figure 65.

For UC3 and UC4 (Figure 47), was verified that model also meets three properties analyzed: liveness, boundedness, and reversibility. The model for these UCs has only 9 transitions and 9 places interconnected by 18 arcs, as well as 9 states. In this model, there are also no "dead" transitions. Analysis is shown in Figure 61 and state space graph is presented in Figure 66.

As shown in Figure 62, UC5 (Figure 49) only meets Boundedness property. As it is a UC that is linked to another UC, it does not meet Liveness and Reversibility properties. The model that represents it ends in transition send_for_analysis, which forwards a medical certificate for analysis that is performed at UC6, and the model is not reversible, as it does not return to its initial M_0 marking. For the same reason, the net is not considered live, as the aforementioned transition will not transfer the marking to another location in the same net.

Also according to Figure 62, the net has 17 places and 17 transitions, interconnected by 33 arcs. Particularly in this net, the number of arcs is less than the sum of places number and transitions because there is a "dead" transition in its composition. Finally, its state space graph is shown in Figure 67 and has 18 possible states. The number of states differs from the number of places and transitions due to the state considered final, that is, dead-marking.

Model properties for the UC6 (Figure 57) are shown in Figure 63. According to Figure 63, UC meets the Liveness, Boundedness, and Reversibility properties and is the largest of the nets ever presented, with 19 places and 19 transitions interconnected by 38 arcs. Also according to Figure 63, the net has 19 possible states and no dead transitions. State space graph is shown in Figure 68.

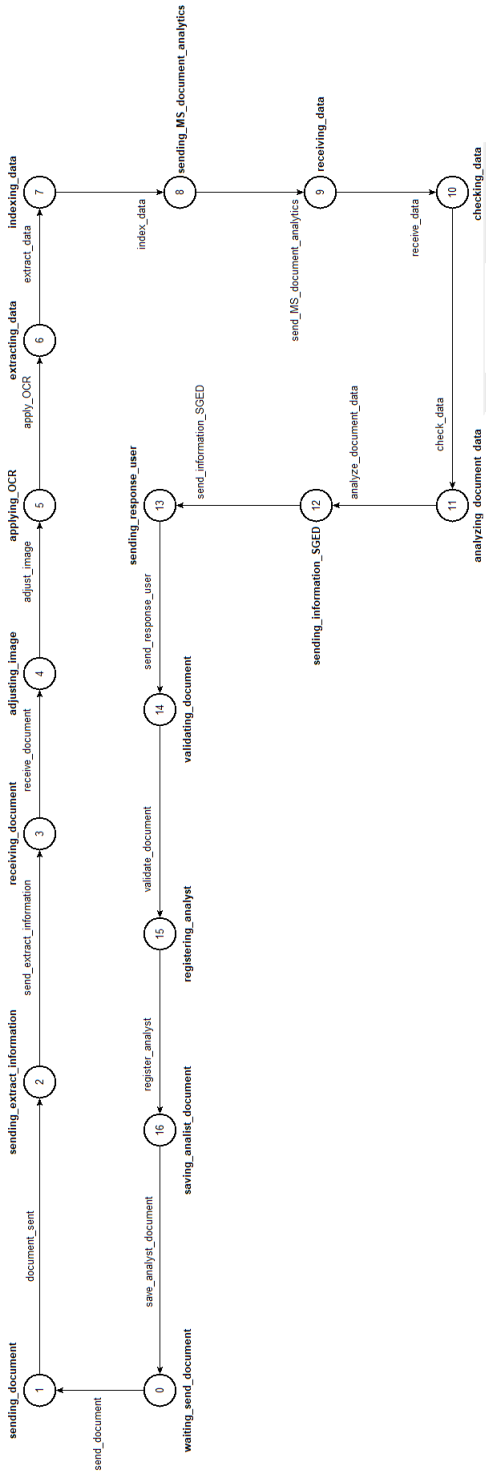


Figure 64 – UC1 State Space Graph

Finally, properties of complete SGED model (Figure 58) are shown in Figure 69. As shown in Figure 69, the net meets the liveness, boundedness, and reversibility properties, similar to smaller models. Also according to Figure 69, the net has 31 places and 31 transitions interconnected by 62 arcs. This number is because some UCs use the same microservices for data extraction and analysis and also the place fusion technique, which reduces the number of

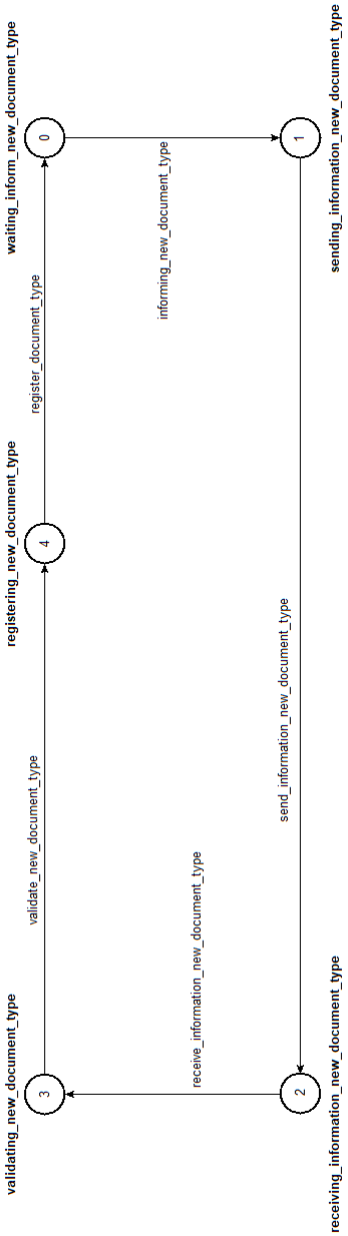


Figure 65 – UC2 State Space Graph

transitions and places. Net has no dead transitions. SGED state space graph is shown in Figure 70 and has 26 possible states.

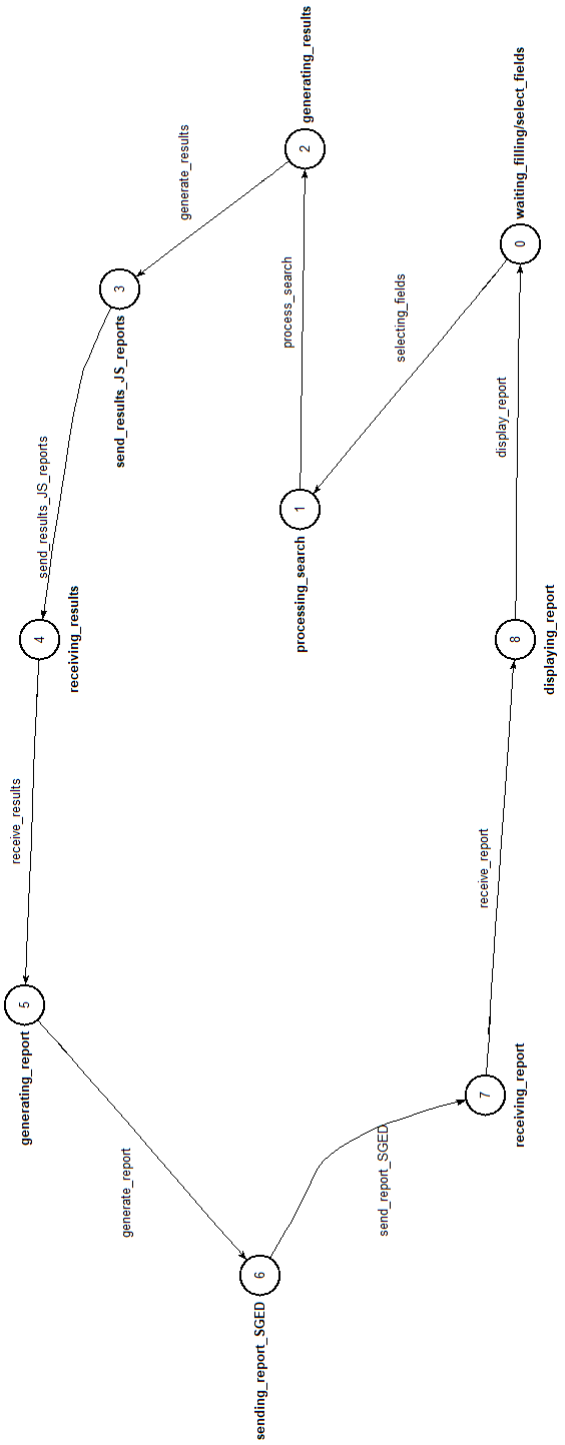


Figure 66 – UC3/UC4 State Space Graph

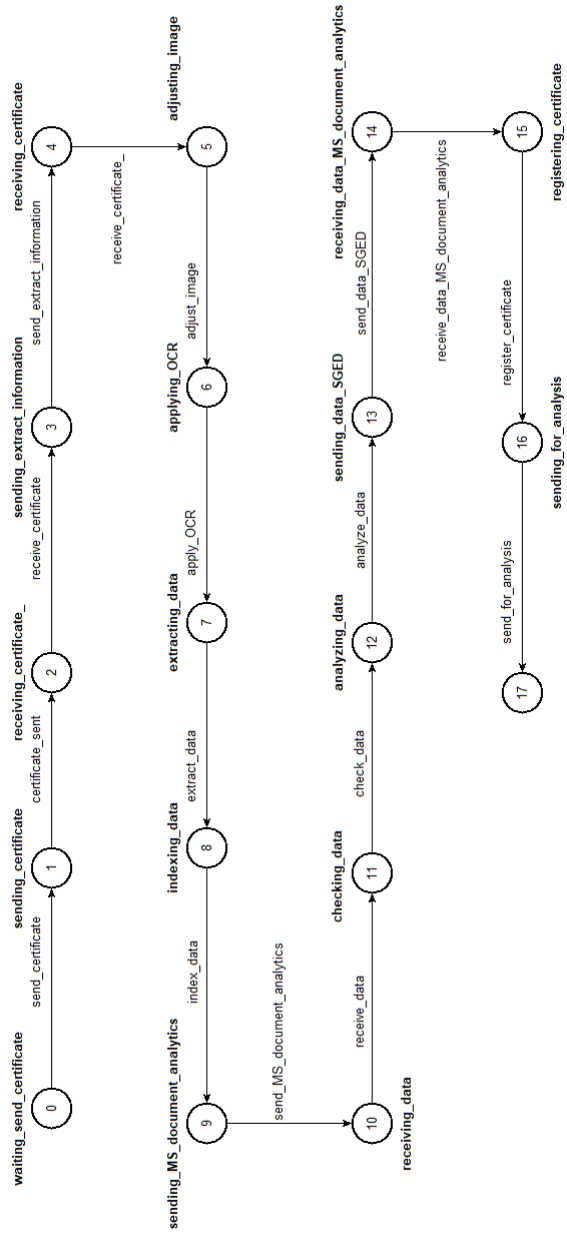


Figure 67 – UC5 State Space Graph

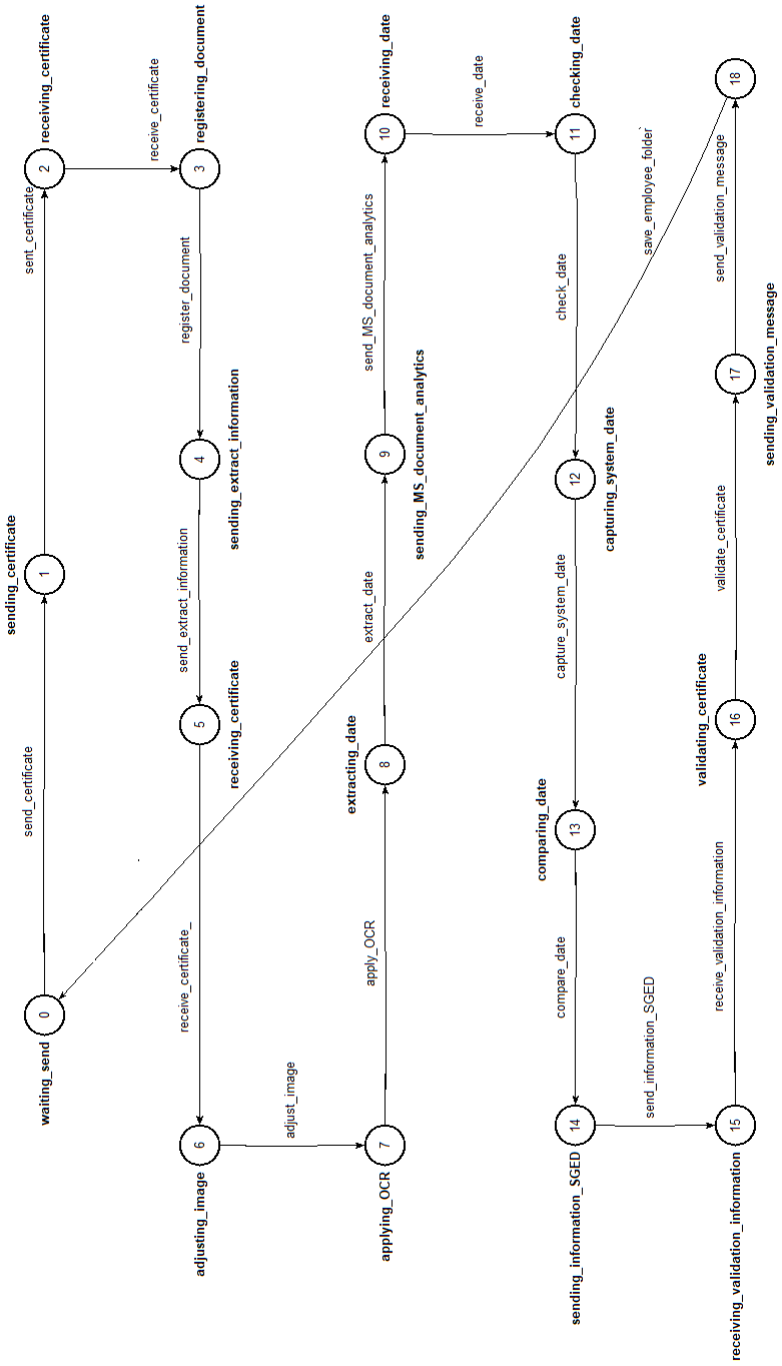


Figure 68 – UC6 State Space Graph

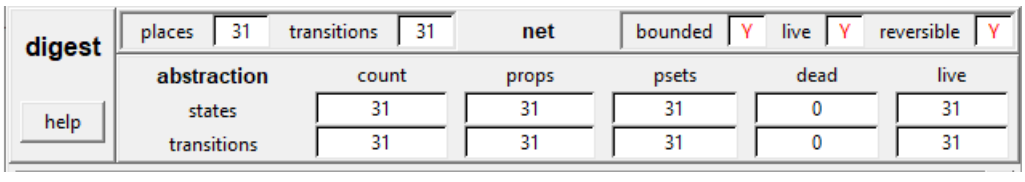


Figure 69 – SGED Good Properties

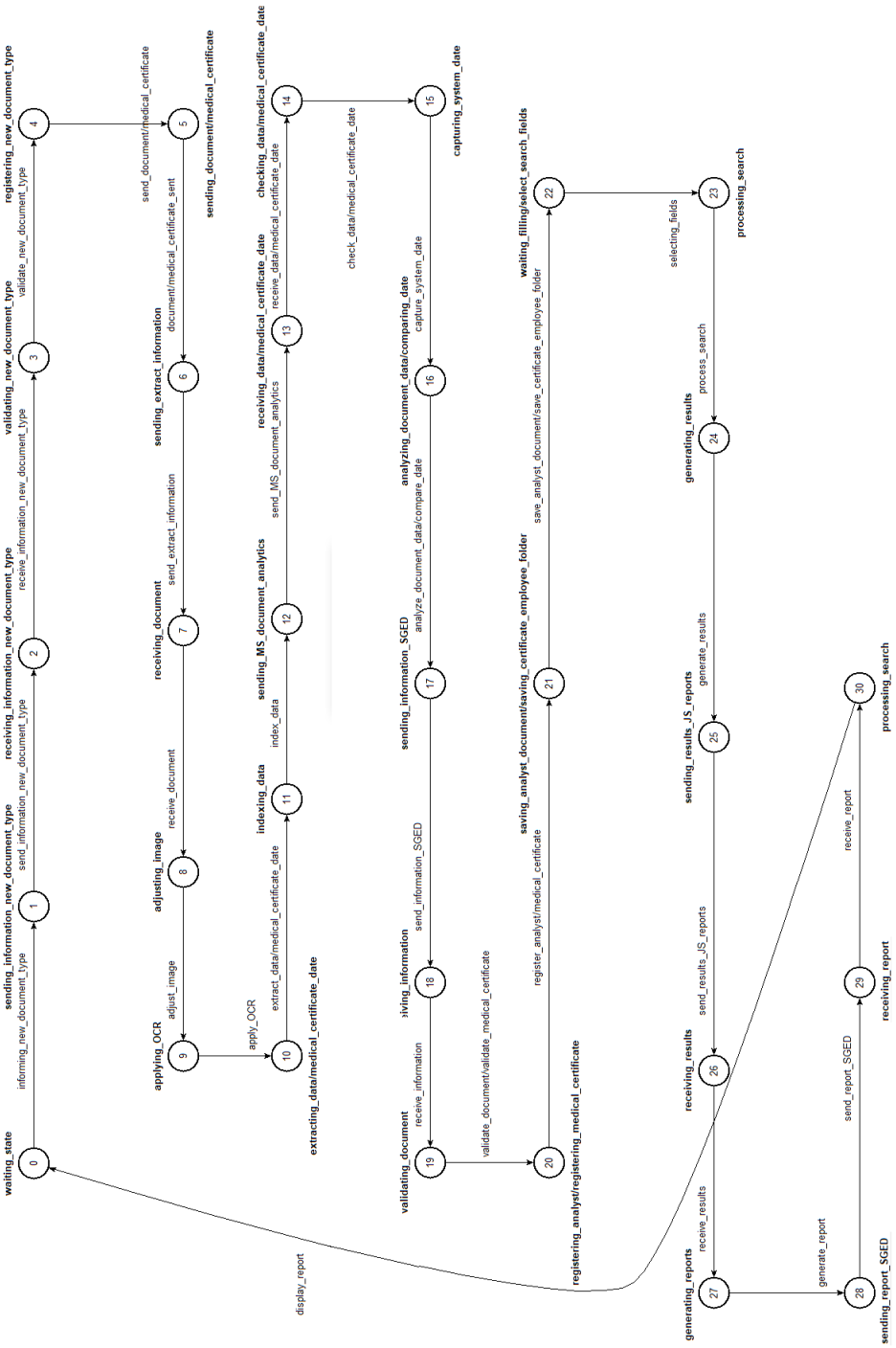


Figure 70 – SGED State Space Graph

5

Conclusions, contributions and future work

Modeling the case study using Petri nets with time allowed us to observe several aspects of the SGED, analyze its behavior, and early detect and prevent deadlocks. The bottom-up approach, used to build the complete model, allowed each part of the system to be completely addressed by building smaller models. Integrating the smaller models, using the place fusion in the net that have similar tasks, combined with the bottom-up approach, allows for a complete system view. Although UML is a graphical modeling language widely used for modeling different types of systems, and offers different types of diagrams that make it possible to provide views of various aspects of the modeled system, there are considerable gaps that do not allow to detect possible problems that may affect the system when in use.

To mitigate the UML gaps and adapt the SGED to business rules, UML Activity diagrams were transcribed to Petri Nets with time associated with transitions. Transcription allowed early detection of problems, such as the need for a report request queue at UC5, shown in Figure 49. Timing models were also an important factor in helping to design appropriate architectural components, such as asynchronous queuing systems. The queue system was implemented asynchronously so that crashes would not occur, which would not allow the system to be used until it was fixed. The transcript also made it possible for temporal business rules to be respected, adapting the SGED to such rules and determining that measures are taken in a timely manner if there is a temporal rules violation.

Another important aspect observed in modeling in Time Petri Nets is the obligation to comply with business rules, imposed with time restrictions. In UC1 (Figure 35) and UC6 (Figure57), a restriction of 24 hours (86400 seconds, as written in transitions of respective UCs) for approval of documents sent to SGED is part of specific business rules for such cases. This allows, in addition to enforcing compliance with such rules, that in case of changes SGED or any other system is adapted to new rules, without the need for major changes.

It is worth mentioning that the addition of time constraint to models is essential for the

fulfillment of business rules and that all timed transitions of the models represent the fulfillment of such rules. Modeling was performed using the TINA tool version 3.6.0 for 64-bit systems, developed by a research group at the Laboratoire d'analyse et d'architecture des systèmes (LAAS-CNRS). The tool, in addition to allowing construction of models in Petri Nets with time associated with transitions, also performs properties analysis of models and builds graphs to verify their conformity. The purpose of transcribing diagrams, in addition to adapting SGED to business rules, was to analyze the behavior of Petri Nets in modeling microservices allied to business rules. It was verified the feasibility of modeling microservices using Petri Nets, its scalability, and integration of system functions, as well as, if necessary, adding other functions or modifying existing functions before implementing the system.

The main contribution of this work was to show the feasibility of building models of large systems using Petri Nets with time, as well as to prove the scalability of the models and observe the behavior of Petri Nets for modeling of microservices allied to business processes.

It was also possible to conclude that Petri Nets are viable for the construction of models that use microservices in business processes. The addition of time restrictions to models helps to ensure that business rules are respected, meeting rules of the organization or specific legislation that regulates the deadline for sending documents. Although formal methods such as Petri Nets are still considered complex, their use in building models leads to significant improvements for the reliability and robustness of the system.

The use of Petri Nets with time for modeling microservices allied to business processes also served to detect problems such as possible bottlenecks, which would lead to system crashes. It also helped to handle exceptions early, leading to better performance in extracting and analyzing data sent to the system.

It is important to highlight another significant contribution of this work for the Software Engineering area with the publication of the article **A Bottom Up Approach for Modeling Business Process using Time Petri Nets** with a study on the first model of SGED, representing UC6. The model was later improved to better represent its functions. The article was published on *Simpósio Brasileiro de Sistemas de informação* (SBSI), classified as Qualis A4 and is available in the scientific base ACM Digital Library under reference:

RAMOS, Danillo Siqueira; ROCHA, Fabio; SOARES, Michel dos Santos. A Bottom Up Approach for Modeling Business Process using Time Petri Nets. In: XVIII Brazilian Symposium on Information Systems. 2022. p. 1-8.

For the production of the models and conclusion of this work, some limitations can be observed. The first one concerns in the impossibility of a face-to-face meeting to understand the requirements of the SGED, due to the COVID-19 pandemic, which may have led to their misunderstandings.

Another limitation observed is the fact that some functionalities of SGED were not

addressed in modeling, due to mere forgetfulness. However, the main functionalities are modeled and presented in this work. Another limitation of this work concerns system delay points that were mapped. There may be more execution points where unmapped delays are generated, without implementing asynchronous communication.

As a suggestion for future works, we intend to expand our study focus to other types of systems in microservices that need time constraints to adapt to the business rules. Many systems in which time constraints are essential to their functioning still use semi-formal languages for modeling, making it difficult to study their properties and add time constraints to models.

Also as future work, it would be interesting to demonstrate in other types of systems, either with transcription of models or elaboration of new models, scalability of Petri Nets with time, in systems that use the microservices architecture, demystifying the use of formal methods.

Finally, as another future work, it is intended to add functionalities to SGED for complete verification of the documents sent, to prevent fraud, as well as add functionalities that adapt to other business rules with time restrictions for sending and receiving more document types.

Bibliography

AALST, W. M. Van der. Business Pprocess Management: A Comprehensive Survey. *International Scholarly Research Notices*, Hindawi, v. 2013, 2013. Citado na página 29.

ABBAS, M. et al. Formal modeling and Verification of UML Activity Diagrams (UAD) with FoCaLiZe. *Journal of Systems Architecture*, Elsevier, v. 114, p. 101911, 2021. Citado na página 14.

ADERALDO, C. M. et al. Benchmark Requirements for Microservices Architecture Research. In: IEEE. *2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*. [S.l.], 2017. p. 8–13. Citado na página 20.

ALETI, A. et al. Software Architecture Optimization Methods: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, IEEE, v. 39, n. 5, p. 658–683, 2012. Citado na página 13.

ALHUMAIDAN, F. et al. State Based Static and Dynamic Formal Analysis of UML State Diagrams. *Journal of Software Engineering and Applications*, Scientific Research Publishing, v. 5, n. 07, p. 483, 2012. Citado na página 13.

AREVALO, C. et al. A Metamodel to Integrate Business Processes Time Perspective in BPMN 2.0. *Information and Software Technology*, v. 77, p. 17–33, 2016. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584916300842>. Citado na página 29.

AVERSANO, L.; GRASSO, C.; TORTORELLA, M. Managing the Alignment Between Business Processes and Software Systems. *Information and Software Technology*, v. 72, p. 171–188, 2016. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584915002189>. Citado na página 29.

BAKAR, N. H.; KASIRUN, Z. M.; SALLEH, N. Feature Extraction Approaches from Natural Language Requirements for Reuse in Software Product Lines: A Systematic Literature Review. *Journal of Systems and Software*, Elsevier, v. 106, p. 132–149, 2015. Citado na página 30.

BALALAIE, A. et al. Microservices Migration Patterns. *Software: Practice and Experience*, Wiley Online Library, v. 48, n. 11, p. 2019–2042, 2018. Citado na página 20.

BASILE, F.; CABASINO, M. P.; SEATZU, C. State Estimation and Fault Diagnosis of Labeled Time Petri Net Systems With Unobservable Transitions. *IEEE Transactions on Automatic Control*, v. 60, n. 4, p. 997–1009, 2015. Citado na página 24.

BORGES, R. M.; MOTA, A. C. Integrating UML and Formal Methods. *Electronic Notes in Theoretical Computer Science*, Elsevier, v. 184, p. 97–112, 2007. Citado na página 13.

CALDERÓN, A.; RUIZ, M. A Systematic Literature Review on Serious Games Evaluation: An Application to Software Project Management. *Computers & Education*, Elsevier, v. 87, p. 396–422, 2015. Citado na página 30.

CARDOSO, J.; VALETTE, R. *Redes de Petri*. [S.l.]: Editora da UFSC Florianópolis, 1997. Citado 2 vezes nas páginas 26 and 51.

CHANG, J. F. *Business Process Management Systems: Strategy and Implementation*. [S.l.]: Auerbach Publications, 2016. Citado na página 28.

CHRISTENSEN, S.; PETRUCCI, L. Modular Analysis of Petri Nets. *The Computer Journal,*, v. 43, n. 3, p. 224–242, 2000. Citado na página 26.

CIAVOTTA, M. et al. A Microservice-Based Middleware for the Digital Factory. *Procedia manufacturing*, Elsevier, v. 11, p. 931–938, 2017. Citado na página 20.

DMITRY, N.; MANFRED, S.-S. On Microservices Architecture. *International Journal of Open Information Technologies*, Laboratory of Open Information Technologies of the faculty of the CMC of Moscow State University., v. 2, n. 9, p. 24–27, 2014. Citado na página 20.

DRAGONI, N. et al. Microservices: Yesterday, Today, and Tomorrow. *Present and ulterior software engineering*, Springer, p. 195–216, 2017. Citado na página 20.

DUMAS, M. et al. *Fundamentals of Business Process Management*. [S.l.]: Springer, 2013. v. 1. Citado na página 28.

ERMEL, G. et al. Supporting the Composition of UML Component Diagrams. In: *Proceedings of the XIV Brazilian Symposium on Information Systems*. [S.l.: s.n.], 2018. p. 1–9. Citado na página 13.

FUENTES-FERNÁNDEZ, L.; VALLECILLO-MORENO, A. An Introduction to UML Profiles. *UML and Model Engineering*, Citeseer, v. 2, n. 6-13, p. 72, 2004. Citado na página 13.

GARLAN, D. Software Architecture: A Roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2000. (ICSE '00), p. 91–101. ISBN 1581132530. Disponível em: <https://doi-org.ez20.periodicos.capes.gov.br/10.1145/336512.336537>. Citado na página 19.

GARLAN, D. Software architecture. *Wiley Encyclopedia of Computer Science and Engineering*, Wiley Online Library, 2007. Citado na página 19.

GARLAN, D. Software Architecture: A Travelogue. In: *Future of Software Engineering Proceedings*. [S.l.: s.n.], 2014. p. 29–39. Citado na página 13.

GIRAULT, C.; VALK, R. *Petri Nets for Systems Engineering: a guide to modeling, verification, and applications*. [S.l.]: Springer Science & Business Media, 2013. Citado 4 vezes nas páginas 7, 26, 27, and 28.

GONCALES, L. J.; FARIAS, K.; BISCHOFF, V. Towards a Hybrid Approach to Measure Similarity Between UML Models. In: *Proceedings of the XV Brazilian Symposium on Information Systems*. New York, NY, USA: Association for Computing Machinery, 2019. (SBSI'19). ISBN 9781450372374. Citado na página 13.

HALL, A. Seven Myths of Formal Methods. *IEEE Software*, IEEE, v. 7, n. 5, p. 11–19, 1990. Citado na página 13.

HASSELBRING, W. Software Architecture: Past, Present, Future. In: *The Essence of Software Engineering*. [S.l.]: Springer, Cham, 2018. p. 169–184. Citado 2 vezes nas páginas 19 and 20.

HEVNER, A. R. et al. Design Science in Information Systems Research. *MIS Quarterly*, JSTOR, p. 75–105, 2004. Citado na página 15.

HUANG, H.; KIRCHNER, H. Policy Composition Based on Petri Nets. In: *2009 33rd Annual IEEE International Computer Software and Applications Conference*. [S.l.: s.n.], 2009. v. 2, p. 416–421. Citado na página 27.

ISO/IEC/IEEE. *ISO/IEC/IEEE Systems and Software Engineering—Architecture Description*. [S.l.]: IEEE Piscataway, NJ, USA, 2011. Citado na página 19.

JAISWAL, M. Software Architecture and Software Design. *International Research Journal of Engineering and Technology (IRJET) e-ISSN*, p. 2395–0056, 2019. Citado 3 vezes nas páginas 12, 19, and 20.

JAMSHIDI, P. et al. Microservices: The Journey so Far and Challenges Ahead. *IEEE Software*, IEEE, v. 35, n. 3, p. 24–35, 2018. Citado 2 vezes nas páginas 20 and 21.

KEELE, S. et al. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007. Citado na página 30.

KHANSA, W. *Réseaux de Pétri P-Temporels: Contribution à L'étude des Systèmes à Évènements Discrets*. Tese (Doutorado) — Chambéry, 1997. Citado na página 24.

KITCHENHAM, B. et al. Systematic Literature Reviews in Software Engineering–A Systematic Literature Review. *Information and Software Technology*, Elsevier, v. 51, n. 1, p. 7–15, 2009. Citado na página 30.

KRYLOVSKIY, A.; JAHN, M.; PATTI, E. Designing a Smart City Internet of Things Platform With Microservice Architecture. In: IEEE. *2015 3rd International Conference on Future Internet of Things and Cloud*. [S.l.], 2015. p. 25–30. Citado na página 20.

LIBERATI, A. et al. The PRISMA Statement for Reporting Systematic Reviews and Meta-Analyses of Studies that Evaluate Health Care Interventions: Explanation and Elaboration. *Journal of clinical epidemiology*, Elsevier, v. 62, n. 10, p. e1–e34, 2009. Citado na página 30.

LIU, G.; BARKAOUI, K. A Survey of Siphons in Petri Nets. *Information Sciences*, Elsevier, v. 363, p. 198–220, 2016. Citado na página 13.

LÓPEZ-GRAO, J. P.; MERSEGUER, J.; CAMPOS, J. Performance Engineering Based on UML & SPN's: A software Performance Tool. In: CRC PRESS. *Proceedings of the 17th International Symposium on Computer and Information Sciences*. [S.l.], 2022. p. 405–409. Citado na página 18.

MALINOVA, M.; GROSS, S.; MENDLING, J. A Study into the Contingencies of Process Improvement Methods. *Information Systems*, v. 104, p. 101880, 2022. ISSN 0306-4379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306437921001022>. Citado na página 29.

MAZLAMI, G.; CITO, J.; LEITNER, P. Extraction of Microservices from Monolithic Software Architectures. In: IEEE. *2017 IEEE International Conference on Web Services (ICWS)*. [S.l.], 2017. p. 524–531. Citado na página 20.

MEDVIDOVIC, N.; TAYLOR, R. N. Software Architecture: Foundations, Theory, and Practice. In: *2010 ACM/IEEE 32nd International Conference on Software Engineering*. [S.l.: s.n.], 2010. v. 2, p. 471–472. Citado na página 20.

MEGHZILI, S. et al. On the Verification of UML State Machine Diagrams to Colored Petri Nets Transformation Using Isabelle/HOL. In: *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. [S.l.: s.n.], 2017. p. 419–426. Citado na página 17.

MENDLING, J. et al. Blockchains for Business Process Management-Challenges and Opportunities. *ACM Transactions on Management Information Systems (TMIS)*, ACM New York, NY, USA, v. 9, n. 1, p. 1–16, 2018. Citado na página 29.

MURATA, T. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, 1989. Citado 4 vezes nas páginas 13, 21, 22, and 24.

NOULAMO, T. et al. Formalization Method of the UML Statechart by Transformation Toward Petri Nets. *IAENG International Journal of Computer Science*, v. 45, n. 4, p. 32, 2018. Citado na página 17.

Ochem, Q.; Perlade, E. Formal Methods for Informal Developpers: A Case-Study Driven by the French Defense Agency (DGA). In: *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. [S.l.: s.n.], 2015. p. 16–16. Citado na página 13.

PELED, D. A. Formal Methods. In: *Handbook of Software Engineering*. [S.l.]: Springer, 2019. p. 193–222. Citado na página 14.

PETERSEN, K. et al. Systematic Mapping Studies in Software Engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. [S.l.: s.n.], 2008. p. 1–10. Citado na página 30.

PONCE, F.; MáRQUEZ, G.; ASTUDILLO, H. Migrating from Monolithic Architecture to Microservices: A Rapid Review. In: *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*. [S.l.: s.n.], 2019. p. 1–7. Citado na página 20.

PUFAHL, L. et al. BPMN in Healthcare: Challenges and Best Practices. *Information Systems*, v. 107, p. 102013, 2022. ISSN 0306-4379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306437922000217>. Citado na página 28.

RAHMOUNE, Y.; CHAOUI, A.; KERKOUCHE, E. A Framework for Modeling and Analysis UML Activity Diagram using Graph Transformation. *Procedia Computer Science*, v. 56, p. 612–617, 2015. ISSN 1877-0509. The 10th International Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) Affiliated Workshops. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050915017421>. Citado na página 16.

RAMAMOORTHY, C.; HO, G. S. Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets. *IEEE Transactions on Software Engineering*, IEEE, n. 5, p. 440–449, 1980. Citado na página 26.

RAMCHANDANI, C. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.* Tese (Doutorado) — Massachusetts Institute of Technology, 1973. Citado na página 26.

ROSEMANN, M.; BROCKE, J. v. The Six Core Elements of Business Process Management. In: *Handbook on Business Process Management 1*. [S.l.]: Springer, 2015. p. 105–122. Citado na página 29.

SEDAGHATBAF, A.; AZGOMI, M. A. SQME: A Framework for Modeling and Evaluation of Software Architecture Quality Attributes. *Software & Systems Modeling*, Springer, v. 18, n. 4, p. 2609–2632, 2019. Citado na página 13.

SEPÚLVEDA, S.; CRAVERO, A.; CACHERO, C. Requirements Modeling Languages for Software Product lines: A Systematic Literature Review. *Information and Software Technology*, Elsevier, v. 69, p. 16–36, 2016. Citado na página 30.

SESHIA, S. A.; SADIGH, D.; SASTRY, S. S. Formal Methods for Semi-Autonomous Driving. In: IEEE. *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. [S.l.], 2015. p. 1–5. Citado na página 13.

SHAHIN, M.; LIANG, P.; BABAR, M. A. A Systematic Review of Software Architecture Visualization Techniques. *Journal of Systems and Software*, v. 94, p. 161–185, 2014. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121214000831>. Citado na página 20.

SHAILESH, T.; NAYAK, A.; PRASAD, D. An UML Based Performance Evaluation of Real-Time Systems Using Timed Petri Net. *Computers*, MDPI, v. 9, n. 4, p. 94, 2020. Citado na página 17.

SIAU, K.; LOO, P.-P. Identifying Difficulties in Learning UML. *Information Systems Management*, Taylor & Francis, v. 23, n. 3, p. 43–51, 2006. Citado na página 13.

SINGH, M.; SHARMA, A.; SAXENA, R. Formal Transformation of UML Diagram: Use Case, Class, Sequence Diagram with Z Notation for Representing the Static and Dynamic Perspectives of System. In: SPRINGER. *Proceedings of International Conference on ICT for Sustainable Development*. [S.l.], 2016. p. 25–38. Citado na página 13.

SOARES, M. S.; VRANCKEN, J. A Modular Petri Net to Modeling and Scenario Analysis of a Network of Road Traffic Signals. *Control Engineering Practice*, v. 20, n. 11, p. 1183–1194, 2012. ISSN 0967-0661. Special Section: Wiener-Hammerstein System Identification Benchmark. Citado na página 16.

SOLA, D. et al. Exploiting Label Semantics for Rule-Based Activity Recommendation in Business Process Modeling. *Information Systems*, v. 108, p. 102049, 2022. ISSN 0306-4379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0306437922000436>. Citado na página 28.

TOFAN, D. et al. Past and Future of Software Architectural Decisions–a Systematic Mapping Study. *Information and Software Technology*, Elsevier, v. 56, n. 8, p. 850–872, 2014. Citado na página 30.

TRIGO, A.; BELFO, F.; ESTéBANEZ, R. P. Accounting Information Systems: Evolving towards a Business Process Oriented Accounting. *Procedia Computer Science*, v. 100, p. 987–994, 2016. ISSN 1877-0509. International Conference on Enterprise Information Systems/International Conference on Project MANagement/International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN / HCist 2016. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050916324334>. Citado na página 29.

VENTERS, C. C. et al. Software sustainability: Research and Practice from a Software Architecture Viewpoint. *Journal of Systems and Software*, v. 138, p. 174–188, 2018. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121217303072>. Citado 2 vezes nas páginas 12 and 19.

VOGLER, W. *Modular Construction and Partial Order Semantics of Petri Nets*. Secaucus, NJ, USA: Springer-Verlag, 1992. ISBN 0387557679. Citado na página 26.

WANG, W.-L.; PAN, D.; CHEN, M.-H. Architecture-based software reliability modeling. *Journal of Systems and Software*, v. 79, n. 1, p. 132–146, 2006. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121205001421>. Citado na página 12.

WANG, X.; MAHULEA, C.; SILVA, M. Diagnosis of Time Petri Nets Using Fault Diagnosis Graph. *IEEE Transactions on Automatic Control*, IEEE, v. 60, n. 9, p. 2321–2335, 2015. Citado na página 14.

WEISSBACH, R. et al. Challenges in Business Processes Modeling – Is Agile BPM a Solution? In: SPRINGER. *International Conference on Business Process Management*. [S.l.], 2016. p. 157–167. Citado na página 28.

WILLIAMS, B. J.; CARVER, J. C. Characterizing Software Architecture Changes: A Systematic Review. *Information and Software Technology*, Elsevier, v. 52, n. 1, p. 31–51, 2010. Citado na página 12.

WOODCOCK, J. et al. Formal Methods: Practice and Experience. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 4, p. 1–36, 2009. Citado 2 vezes nas páginas 14 and 26.

WOODS, E. Software Architecture in a Changing World. *IEEE Software*, IEEE, v. 33, n. 6, p. 94–97, 2016. Citado na página 12.

YANG, N. et al. Modeling UML Sequence Diagrams Using Extended Petri Nets. *Telecommunication Systems*, Springer, v. 51, n. 2, p. 147–158, 2012. Citado 2 vezes nas páginas 13 and 16.

ZHANG, C. et al. Software Architecture Modeling and Reliability Evaluation Based on Petri Net. In: IEEE. *2017 International Conference on Dependable Systems and Their Applications (DSA)*. [S.l.], 2017. p. 51–56. Citado na página 13.

ZHOU, M.; WU, N. *System Modeling and Control with Resource-Oriented Petri Nets*. [S.l.]: Crc Press, 2018. v. 35. Citado 2 vezes nas páginas 22 and 24.