Verificação de Diagramas UML utilizando redes de Petri

Larissa Gabriela Dias Vidal - 43364

Dissertação apresentada à Escola Superior de Tecnologia e Gestão de Bragança para obtenção do Grau de Mestre em Informática no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná

Trabalho orientado por:

José Eduardo Moreira Fernandes

Franck Carlos Velez Benito

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2021-2023

Verificação de Diagramas UML utilizando redes de Petri

Larissa Gabriela Dias Vidal - 43364

Dissertação apresentada à Escola Superior de Tecnologia e Gestão de Bragança para obtenção do Grau de Mestre em Informática no âmbito da dupla diplomação com a Universidade Tecnológica Federal do Paraná

Trabalho orientado por:

José Eduardo Moreira Fernandes

Franck Carlos Velez Benito

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2021-2023

Dedicatória

Dedico este trabalho aos meus pais, Seyrio e Alessandra, e aos meus irmãos, Mariana, Henrique e Maria Alice, por toda a força, incentivo, carinho, amor incondicional, união e momentos compartilhados.

Agradecimentos

Agradeço a ...

Primeiramente a Deus, por me dar forças todos os dias, pela minha vida e por todas as minhas bênçãos diárias.

Aos meus familiares, minha avó Maria por tantas vezes investir na minha educação e por todo o apoio. Aos meus primos e primas, em especial à Carol, por todos os momentos da infância. Aos meus tios e tias, em especial à tia Solange e sua família por tantas vezes cuidar de mim e dos meus irmãos, à minha tia Rosa por todo o apoio e tantas ajudas desde que eu iniciei no mundo da computação.

Aos meus amigos, em especial à Samara que sempre esteve comigo nos bons e maus momentos e, mesmo de longe, sempre está presente. À Marcelle e à sua família por todo o carinho, em especial ao meu afilhado, Lulu.

Aos amigos da faculdade, Euristenede por toda a competição para sermos cada vez melhor e todos os trabalhos em equipe. João por todos os filmes e trabalhos em equipe. Às mulheres, incríveis, que nos unimos e nos ajudamos, Raissa, Nathalia e Luanna. Ao Will que me acolheu quando eu decidi mudar de país e viver a aventura do mestrado. Ao Alexandre, por todos os trabalhos em equipe e palavras de incentivo. Ao Jecé e ao Matheus, que fizeram parte de tudo isso e estiveram comigo em momentos importantes. À Camila e ao Guto que por tantas vezes me acolheu no Porto.

À todos os amigos do trabalho, em especial o Gabriel, Miguel, Markin, Louback, Doris e Teteu, obrigada por todo o conhecimento compartilhado.

À todos os professores e servidores da Universidade Tecnológica Federal do Paraná (UTFPR), em especial à Cleide que nos acolheu no início do curso e à Carol, Lieges,

Adri e Fernanda, do DEMAP, que por tantas vezes compartilhamos momentos e pude aprender muito sobre a universidade pública. À Arlete por fazer do curso de Ciência da Computação o que é, à Vera por me cobrar tanto na escrita e ao Davi por orientar meus estágios e acompanhar tudo de perto. Gostaria de agradecer, muito, não tenho nem palavras, à professora Leiliane por todo o conhecimento sobre grafos e redes de Petri, em especial às coloridas, e por todas as chamadas e explicações. Aos demais professores, obrigada por todo ensinamento compartilhado.

Aos meus orientadores, Eduardo e Franck, por toda a paciência, ensinamentos, contribuições para o trabalho e pela confiança em mim.

Ao meu namorado, Gabriel, por cada palavra de incentivo, milhares de ajudas, momentos compartilhados, toda a cumplicidade desde o primeiro dia e por todo o companheirismo diário.

Por fim, aos mais importantes, meus pais, Seyrio e Alessandra e meus irmãos, Mariana, Henrique e Maria Alice. Por sempre acreditarem em mim, até mesmo nos momentos que eu não acreditava. Por sempre me lembrarem da minha capacidade. Por cada momento, cada aventura, cada viagem. Por todo o amor, carinho, atenção, imensas ajudas. Por estarmos juntos em todos os momentos, por toda a união, enfim, por tudo. Gratidão imensa por ter vocês, não sei o que seria de mim sem vocês. Pai e mãe, este trabalho foi realizado graças a vocês, obrigada por sempre acreditarem em mim e por investir tanto em mim.

Resumo

A modelação de *software* desempenha um papel importante na garantia da qualidade e confiabilidade de sistemas computacionais. Nesse contexto, a *Unified Modeling Language* (UML) e as redes de Petri são amplamente empregadas como ferramentas para representar e analisar sistemas complexos, visando validar a corretude dos diagramas.

Este trabalho propõe a criação de regras para o mapeamento dos diagramas de casos de uso e sequência da UML 2.0 para redes de Petri seguras equivalentes. As regras são aplicadas na prática, mapeando diagramas de um artigo encontrado através de pesquisas, seguido pela análise formal das redes resultantes. Além disso, é realizado o mapeamento desses mesmos diagramas para redes de Petri coloridas com base em regras criadas por outros autores, bem como a análise formal desse mapeamento. Por fim, é realizada uma comparação entre as abordagens.

Os resultados deste estudo fornecem uma base sólida para a compreensão e aprimoramento dos processos de mapeamento de diagramas da UML para redes de Petri seguras e coloridas, contribuindo para a validação e verificação eficaz de sistemas complexos de *software*.

Palavras-chave: UML, Rede de Petri, Diagrama de Sequência, Diagrama de Casos de Uso

Abstract

Software modeling has an important role in ensuring the quality and reliability of computer systems. In this context, the UML and Petri nets are usually used as tools for representing and analyzing complex systems, aiming to validate the correctness of the diagrams.

This work proposes the creation of rules for mapping UML 2.0 use case and sequence diagrams to equivalent secure Petri nets. The rules are applied in practice by mapping diagrams from a research paper, followed by the formal analysis of the resulting nets. Additionally, the mapping of these same diagrams to coloured Petri nets is performed based on rules created by other authors, along with the formal analysis of this mapping. Finally, a comparison between the approaches is conducted.

The results of this study provide a solid foundation for understanding and improving the mapping processes of UML diagrams to secure and coloured Petri nets, contributing to the effective validation and verification of complex software systems.

Keywords: UML, Petri Nets, Sequence Diagrams, Use case

Conteúdo

Lista de Figuras

Siglas

ALT *Alternatives*. 11, 23, 36, 37, 42, 53, 56, 75, 79

ATL *Atlas Transformation Language*. 21

CPN *Coloured Petri Net*. 22

LOOP *Looping*. 11, 23, 32, 35, 39, 40, 42, 44, 90, 91

OPT *Option*. 11, 35, 41

PAR *Parallel*. 11, 23, 37

PIPE *Platform Independent Petri Net Editor*. 19, 57, 60, 63, A1

REF *Reference*. 10, 35, 37, 52, 55

SEQ *Weak Sequencing*. 23

UML *Unified Modeling Language*. viii, ix, 1–3, 5, 6, 10, 21–27, 32, 33, 35, 36, 90, 91, 93

UTFPR Universidade Tecnológica Federal do Paraná. vi

Capítulo 1

Introdução

Neste capítulo será apresentado a introdução do trabalho. O capítulo foi dividido em três seções: a seção 1.1 refere-se ao enquadramento do trabalho, a seção 1.2 refere-se aos objetivos do trabalho e a seção 1.3 refere-se à estrutura utilizada no documento.

1.1 Enquadramento

Os sistemas computacionais são capazes de solucionar problemas, facilitar operações e automatizar processos. Na atualidade, eles estão presentes em diversas áreas e de diferentes maneiras. A modelação de *software* permite visualizar a estrutura do sistema antes de desenvolvê-lo. Com os diagramas da UML é possível entender como será o comportamento e quais serão as características do sistema, ou seja, a UML possibilita ter uma visão ampla do *software* a ser desenvolvido [1]. Dessa forma, o desenvolvimento do *software* fica mais simples e organizado.

De acordo com "Unified Modeling Language 2.5.1" [2], a UML tem como objetivo oferecer aos arquitetos de *software*, engenheiros de *software* e desenvolvedores de *software*, ferramentas para análise, projeto e implementação de sistemas baseados em *software* como também, para a modelação de negócios e processos similares. Pelo fato dos diagramas descreverem o comportamento ou as características do *software* é importante que eles sejam

desenvolvidos corretamente. Entretanto, é muito comum que ocorra erros ao desenvolver os diagramas da UML. Estes erros podem acarretar em problemas ao desenvolver o *software* o que pode causar retrabalho, pois o sistema deverá ser todo reconstruído para corrigir as falhas que ocorrem devido ao mal desenvolvimento da modelação.

Os diagramas da UML são catorze, divididos em diagramas estruturais e diagramas comportamentais. Entretanto, não é obrigatório o uso de todos os diagramas para a modelação do *software*, no geral, são utilizados apenas alguns deles. Dentre os diagramas existentes, há uma forte relação entre os diagramas de sequência, casos de uso e diagrama de classes, pois todos os objetos existentes no diagrama de sequência pertencem às classes presentes no diagrama de classes. Além disso, é possível utilizar o diagrama de sequência para expressar as iterações que ocorrem nos diagramas de casos de uso [3].

A rede de Petri é um modelo matemático formal capaz de obter resultados precisos, pois a rede proporciona uma análise precisa dos modelos além de possibilitar a visualização gráfica dos processos [4].

A UML é a linguagem padrão utilizada na modelação de sistemas enquanto que, a rede de Petri possui como principal característica o formalismo matemático, possibilitando obter resultados precisos. Muitos autores utilizam a rede de Petri para apoiar os designers na melhoria da arquitetura de seu *software*, transformando os diagramas da UML em um modelo de uma rede de Petri.

1.2 Objetivos

No âmbito científico, um modelo é uma representação de um sistema, que busca capturar algumas de suas características e fornecer conhecimento sobre ele. Em geral, os modelos possuem um nível de abstração que pode ser refinado através das etapas de transformação, resultando em modelos mais detalhados e complexos [5].

De acordo com os autores Kleppe, Warmer e Bast [6], a transformação de modelos consiste no processo de geração de um novo modelo a partir de um modelo existente. Essa transformação é realizada por meio de uma coleção de regras de transformação, que são

especificações claras de como um modelo pode ser utilizado para criar outro.

Dessa forma, é possível afirmar que a transformação de modelos é um importante instrumento para a análise e compreensão de sistemas complexos, permitindo a construção de modelos mais sofisticados e precisos.

A partir disso, o objetivo deste trabalho é propor um conjunto de regras para realizar o mapeamento de diagramas de casos de uso e diagramas de sequência da UML 2.0 em redes de Petri seguras. Adicionalmente, propõe-se o mapeamento dos mesmos diagramas em redes de Petri coloridas, com base nas regras criadas por Ribeiro, Fernandes, Tjell et al. [7]. Após o mapeamento, é realizada uma análise formal de cada uma das redes, com o intuito de verificar o comportamento do *software* em questão e detetar possíveis erros de forma antecipada. Por fim, realiza-se uma comparação entre as abordagens de redes de Petri seguras e redes de Petri coloridas, visando identificar as vantagens e desvantagens de cada uma delas.

O uso das redes de Petri como ferramenta para análise formal de comportamento de *software* é uma prática realizada por muitos autores. Dessa forma, o mapeamento dos diagramas de casos de uso e sequência da UML para as redes de Petri é uma técnica promissora para antecipar e detetar problemas no *software*. Com isso, espera-se contribuir para o desenvolvimento de *softwares* mais seguros e confiáveis.

1.3 Estrutura do Documento

O conteúdo deste projeto está organizado da seguinte forma: no capítulo 2 são apresentadas todas referências bibliográficas necessárias para entendimento do escopo geral do projeto, iniciando por conceitos da UML (Linguagem de Modelação Unificada), seguindo para os conceitos de redes de Petri e terminando com o estado da arte; no capítulo 3 é apresentado as regras criadas para a mapeamento dos diagramas de casos de uso e de sequência da UML 2.0 para a rede de Petri segura; no capítulo 4 são apresentadas a aplicação das regras de mapeamento para as redes de Petri seguras e coloridas, a análise das redes, bem como discussões e interpretações acerca dos resultados obtidos; por fim, no

capítulo 5, são fornecidas informações sobre todo o trabalho realizado, abordando aspetos relacionados a trabalhos futuros e finalizando em uma conclusão que destaca os resultados e objetivos alcançados.

Capítulo 2

Revisão Bibliográfica

Neste capítulo será descrito a revisão bibliográfica necessária para a compreensão do trabalho. O capítulo foi dividido em três seções: a seção 2.1 refere-se às características fundamentais da UML, a seção 2.2 refere-se às principais características da rede de Petri e suas extensões e a seção 2.3 refere-se ao estado da arte.

2.1 Linguagem de Modelação Unificada (UML)

A UML é uma linguagem de modelação de *software* amplamente utilizada para o desenvolvimento de sistemas orientados a objetos [8]. Segundo Vargas [1], a UML é uma linguagem que permite a especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos. É considerada uma das linguagens mais expressivas para a modelação de sistemas orientados a objetos. A utilização da UML no desenvolvimento de *software* orientado a objetos oferece vantagens significativas, como maior clareza e compreensão do processo de desenvolvimento de *software*, facilitando a comunicação entre desenvolvedores e gerentes de projetos. Além disso, a UML permite que os desenvolvedores de *software* criem modelos precisos e eficientes de sistemas complexos, facilitando a deteção de problemas e a tomada de decisões importantes durante o processo de desenvolvimento.

A linguagem UML 2.0 é formada por catorze diagramas e esses diagramas são classificados em diagramas estruturais e diagramas comportamentais. As subseções 2.1.1 e 2.1.2 descrevem, respectivamente, as características dos diagramas estruturais e comportamentais.

2.1.1 Diagramas Estruturais

A linguagem UML fornece sete tipos de diagramas para modelar a estrutura do sistema de diferentes visões. O comportamento dinâmico dos elementos em questão, não são considerados nesses diagramas [8].

Os diagramas estruturais abordam o aspeto estrutural do ponto de vista do sistema. O objetivo deles é visualizar, especificar, construir e documentar os aspetos estáticos de um sistema. Estes aspetos estáticos englobam a existência e a colocação de itens como classes, interfaces, colaborações e componentes.

A figura 2.1 apresenta os tipos de diagramas estruturais descritos na literatura.

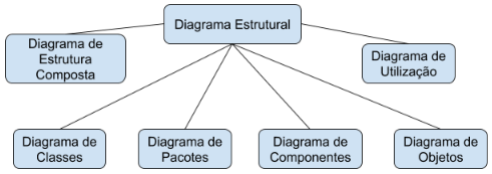

Figura 2.1: Diagramas Estruturais.

2.1.2 Diagramas Comportamentais

De acordo com a Seidl, Scholz, Huemer et al. [8], com os diagramas de comportamento, a UML oferece a infraestrutura que permite definir o comportamento do sistema em detalhes. O comportamento se refere a uma consequência direta de uma ação de pelo menos um objeto. Isto afeta como os estados dos objetos alteram ao longo do tempo. O comportamento pode ser especificado através de ações de um único objeto ou resultar a partir de interações entre múltiplos objetos.

A figura 2.2 apresenta os diagramas comportamentais existentes da UML.

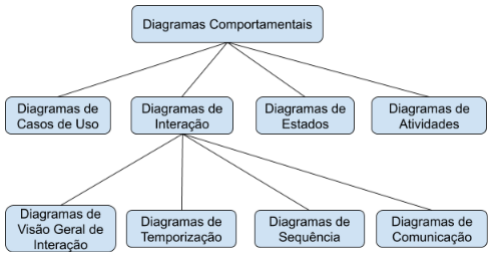

Figura 2.2: Diagramas Comportamentais.

Neste trabalho apenas os diagramas de casos de uso e sequência serão descritos devido ao foco do trabalho.

Diagrama de Casos de Uso

Segundo Seidl, Scholz, Huemer et al. [8] o diagrama de caso de uso possibilita a descrição de possíveis cenários de uso (casos de uso) para os quais um sistema é desenvolvido. Ele modela qual utilizador do sistema utiliza qual funcionalidade, expressando quem realmente irá trabalhar com o sistema que está sendo construído.

O diagrama de casos de uso é composto por:

- Atores: simbolizam os papéis realizados por pessoas, sistemas ou organizações que, em algum momento, interagem com as funcionalidades do sistema;
- Caso de Uso: descreve as funcionalidades esperadas do sistema a ser desenvolvido de acordo com os desejos do cliente responsável pelo desenvolvimento do sistema;
- Associações: descrevem as interações que ocorrem entre os atores e os casos de uso.

A figura 2.3 apresenta o diagrama de casos de uso de um sistema de academia. Os atores são representados por aluno, funcionário e gerente. Os controladores e os cadastros representam os casos de uso. As setas entre os atores e os casos de uso representam as associações existentes entre os atores e os casos de uso.

Em um diagrama de casos de uso pode-se expressar o relacionamento entre casos de uso das seguintes maneiras [3]:

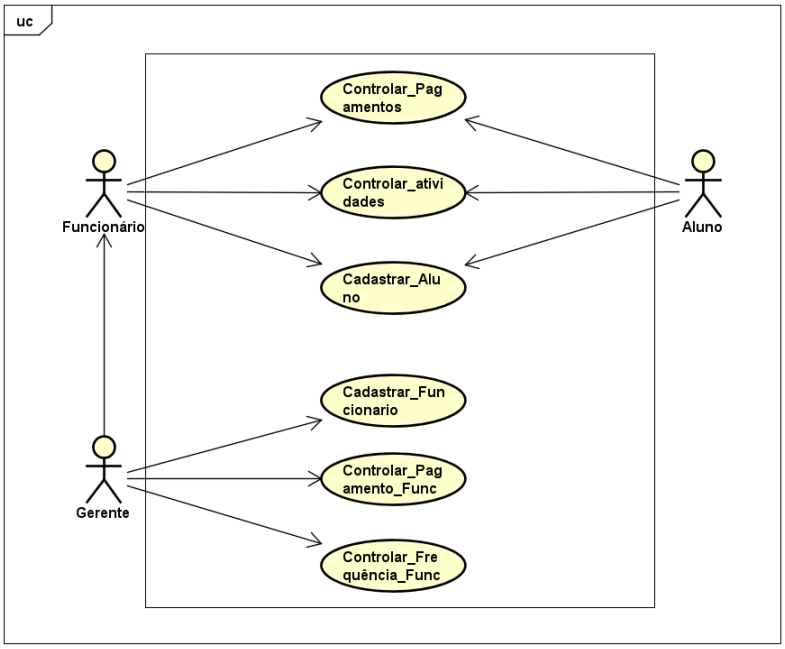

Figura 2.3: Diagrama de Caso de Uso.

- Inclusão: quando existe uma situação, rotina ou cenário comum a mais de um caso de uso, utiliza-se o relacionamento de inclusão. O relacionamento de inclusão expressa obrigatoriedade onde deve-se executar o caso de uso incluído quando o caso base for executado;

- Extensão: esses relacionamentos são utilizados para descrever cenários opcionais de um caso de uso. Os casos de uso estendidos retratam cenários que ocorrem somente em situações específicas se determinada condição for satisfeita;

- Generalização: neste tipo de relacionamento, o caso de uso filho herda todos os cenários possíveis do caso de uso pai.

Diagrama de Sequência

O diagrama de sequência estabelece a disposição dos eventos, identificando as mensagens que devem ser disparadas e em qual ordem. De acordo com Guedes [3], o diagrama de sequência é um diagrama que se atenta a ordem temporal em que as mensagens são

trocadas entre os objetos que estão envolvidos em um determinado processo.

Estes diagramas são compostos por [3] [8]:

- Objetos: representam os participantes. O objeto do diagrama de sequência pode ser um ator, entidade, fronteira, entre outros;

- Linha de vida (*Lifeline*): um participante individual em uma interação que existe durante um período de tempo. É exibida como uma linha vertical, geralmente tracejada, que representa o tempo de vida do objeto associado a ela;

- Mensagens: usadas para representar a ocorrência de eventos que forçam a chamada de um método em algum dos objetos que estão no processo. É representada como uma seta do remetente para o destinatário. O tipo da seta expressa o tipo da comunicação envolvida:

  - Mensagem síncrona: é representada por uma seta de linha contínua e uma ponta de seta triangular preenchida. Neste tipo de mensagem, o remetente da mensagem aguarda até que o destinatário envia uma mensagem de resposta antes de continuar;

  - Mensagem assíncrona: é representada por uma seta de linha contínua e uma ponta de seta triangular aberta. Neste tipo de mensagem, o remetente continua após o envio da mensagem;

- Fragmentos Combinados: noções abstratas de unidades de interação geral. É representado através de um retângulo que abrange toda a interação e apresenta uma aba no canto superior esquerdo contendo um operador que indica qual o diagrama de interação que ele se refere.

Dentre os diagramas existentes, há uma forte relação entre os diagramas de sequência, casos de uso e diagrama de classes, pois todos os objetos existentes no diagrama de sequência pertencem às classes presentes no diagrama de classes. Além disso, é possível

utilizar o diagrama de sequência para expressar as iterações que ocorrem nos diagramas de casos de uso [3].

O principal objetivo deste diagrama é determinar a ordem que os eventos irão ocorrer, quais as mensagens que serão enviadas, os métodos que serão chamados e demonstrar a forma como os objetos interagem em um processo. Todos os objetos existentes no diagrama de sequência pertencem às classes presentes no diagrama de classes, além disso, a partir do diagrama de sequência é possível expressar as interações que ocorrem nos diagramas de casos de uso [3].

A figura 2.4 representa o exemplo do diagrama de sequência para o sistema de uma academia.

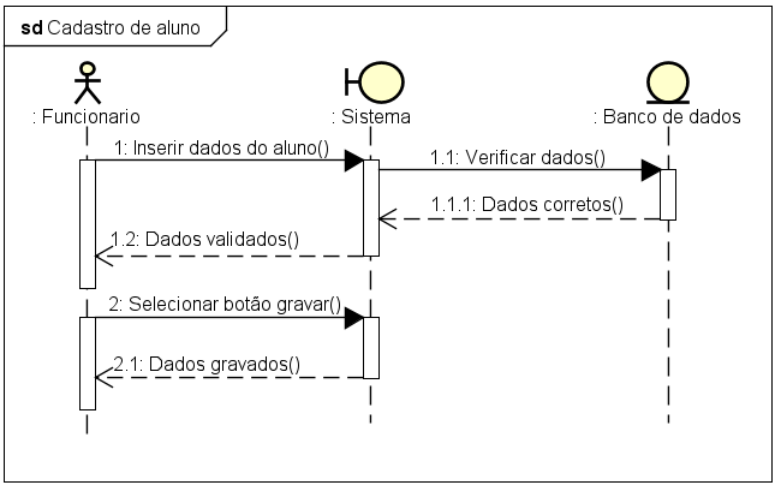

Figura 2.4: Diagrama de Sequência.

Fragmentos Combinados e Operadores de Interação

A UML 2.0 inseriu vários novos elementos gráficos para a construção modular de diagramas de interação relativos aos diagramas de sequência como os fragmentos combinados e os operadores de interação.

Alguns dos operadores de interação são [3] [8]:

- *Reference* (REF): uma referência de interação permite integrar um diagrama de

sequência em outro diagrama de sequência. Por um lado, isso permite que seja possível reutilizar interações que já foram modeladas e, por outro lado, permite dividir sequências de interação complexas em módulos e descrevê-las de forma simples;

- *Alternatives* (ALT): determina que o fragmento combinado representa uma escolha entre dois ou mais comportamentos. Este fragmento costuma utilizar condições de guarda (texto entre colchetes que determina uma regra ou condição), conhecido também como restrição de interação, para determinar o teste a se considerar na escolha de um dos comportamentos;

- *Parallel* (PAR): define que o fragmento combinado retrata uma execução paralela de dois ou mais comportamentos. Cada operação paralela é dividida por uma linha tracejada;

- *Looping* (LOOP): representa que um fragmento combinado terá um laço que poderá ser repetido várias vezes. Este fragmento possui exatamente um operando. A palavra-chave LOOP é seguida por uma especificação opcional de números de iteração dentro do LOOP. Esta especificação possui (min...max) ou (min, max), onde min especifica o número mínimo de iterações e max especifica o número máximo de iterações.

- *Break*: mostra uma "quebra" na execução normal do processo. É principalmente utilizado para modelar o tratamento de exceções. Este fragmento consiste em um operador único mais uma guarda. Se a guarda é verdadeira, as interações dentro deste operando são executadas, as operações restantes do fragmento circundante são omitidas e a interação continua no próximo fragmento de nível superior;

- *Option* (OPT): o fragmento OPT retrata a execução opcional de um evento, ou seja, o comportamento do fragmento pode ou não ser executado.

2.2 Redes de Petri

Uma rede de Petri é uma ferramenta amplamente utilizada na modelação de sistemas dinâmicos, permitindo a representação de processos que envolvem concorrência, paralelismo e sincronização de informações [4]. Rezende [4] também destaca que devido ao seu formalismo matemático, a rede de Petri é capaz de fornecer uma análise precisa dos modelos, verificando propriedades estruturais e comportamentais do sistema em questão. Além disso, a rede de Petri oferece uma visualização gráfica dos processos, facilitando a comunicação entre as partes interessadas no projeto do sistema.

É importante ressaltar que o uso da rede de Petri requer uma compreensão sólida dos conceitos matemáticos envolvidos, mas seus benefícios em relação à análise precisa e visualização gráfica dos processos justificam seu uso na modelação de sistemas dinâmicos.

Os elementos básicos de uma rede de Petri são [9]:

- Lugar: sua interpretação pode ser como uma condição, um estado parcial, uma espera, um procedimento, um conjunto de recursos, um estoque, uma posição geográfica num sistema de transporte, entre outros;
- Transição: relacionada a um evento que ocorre no sistema;
- Ficha: indica que a condição associada ao lugar é verificada. Pode representar um objeto numa certa posição geográfica ou uma estrutura de dados que se manipula.

Graficamente, a rede de Petri é representada por círculos para denominar os lugares, retângulos para denominar as transições e uma seta ligando os lugares e transições para demonstrar a direção da rede. Elas possuem em sua estrutura uma marcação inicial que retrata o estado inicial da rede.

As fichas são representadas com círculos totalmente preenchidos dentro dos lugares e o peso dos arcos é demonstrado a partir de números próximos dos mesmos. Caso não exista nenhum número próximo ao arco, por definição este arco possui peso 1. A figura 2.5 apresenta um exemplo de uma rede de Petri.

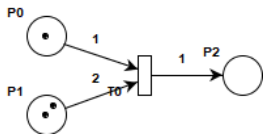

Figura 2.5: Rede de Petri Marcada.

É possível resumir algumas das vantagens da rede de Petri através das seguintes considerações [9]:

- Capaz de descrever uma ordem parcial entre vários eventos, o que proporciona flexibilidade;
- Estados e eventos são demonstrados explicitamente;
- Com uma única família de ferramentas, realiza a especificação, modelação, análise, avaliação do desempenho e implementação;

2.2.1 Definição

Uma Rede de Petri é uma quádrupla $RP = (P, T, A_r, W)$ em que:

- $P = \{p_1, p_2, ..., p_n\}$ é um conjunto finito de lugares;
- $T = \{t_1, t_2, ..., t_n\}$ é um conjunto finito de transições;
- $A_r \subseteq (P \times T) \cup (T \times P)$ é um conjunto finito de arcos;
- $W : A_r \to \mathbb{N}^+$ é a função de ponderação;

O A_r é composto por pré-conjuntos e post-conjuntos que, segundo Cardoso e Valette [9] e Rezende [4], define-se:

- $Pre : P \times T \to \mathbb{N}$ determina os arcos de entrada das transições (ocorrência anterior) sendo $\mathbb{N}$ o conjunto dos números naturais. Formalmente é denotado por $\cdot p$ para representar o conjunto de transições que compartilham p como lugar de entrada e $\cdot t$ para representar o conjunto de lugares de entrada de uma transição t;

- $Post : P \times T \rightarrow \mathbb{N}$ determina os arcos de saída das transições (ocorrência posterior). Formalmente é denotado por $p\cdot$ para representar o conjunto de transições que compartilham p como lugar de saída e $t\cdot$ para representar o conjunto de lugares de saída de uma transição t;

Uma rede de Petri marcada é um par $RPM = (RP, M_0)$ que contém uma rede finita $RP = (P, T, A_r, W)$ e uma marcação inicial $M_0 \rightarrow \mathbb{N}$ que satisfaz [10] [11]:

$$\forall p \in P : M_0(p) \leq K(p)$$

2.2.2 Sensibilização

Uma transição está sensível (ou habilitada) se cada lugar de entrada p de t está marcado com ao menos $w(p, t)$ fichas, onde $w(p, t)$ define o peso do arco que liga um lugar p a uma transição t [12]. Se o evento ocorrer, uma transição habilitada será disparada. Caso o evento não ocorra, a transição habilitada não será disparada. O disparo de uma transição t habilitada remove $w(p, t)$ fichas de cada lugar de entrada p de uma transição t, e adiciona $w(t, p)$ fichas em cada lugar de saída p de uma transição t, onde $w(t, p)$ define o peso do arco de t para p [12].

2.2.3 Disparo e sequência de disparos

Se uma transição t está sensibilizada por uma marcação M, isso significa que a transição pode ser disparada e, ao ser disparada, obtém-se uma nova marcação M' de t tal que [9]:

$\forall p \in P, M'(p) = M(p) - Pre(p, t) + Post(p, t)$

A nova marcação M' é dada pela equação [9]:

$M' = M - Pre(., t) + Post(., t) = M + C(., t)$.

A matriz de incidência de uma rede de Petri representa os números de fichas que serão acrescentadas e removidas de um lugar. Dessa forma, ${a_{ij}}^+ = w(i, j)$ define o número de fichas que são acrescentadas ao lugar p_j quando a transição t_i dispara e ${a_{ij}}^- = w(j, i)$ define o número de fichas que são removidas do lugar p_j quando t_i dispara. Ou seja, a_{ij} é o

número de fichas alteradas no lugar p_j com um disparo da transição t_i. Outra observação é a de que a transição t_i é habilitada numa marcação M, se e somente se [11]:

$a_{ij}{}^{-} \leq M(p_j)$.

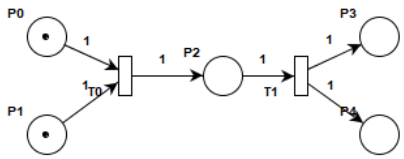

Figura 2.6: Rede de Petri Sensibilizada.

De forma simples, uma sequência de disparos é o conjunto das transições que foram disparadas a partir de uma marcação M_x levando a uma marcação M_n.

É possível representar uma sequência de disparos σ através de um vetor chamado *vetor característico* da sequência σ. O seu tamanho é o número de transições da rede de Petri com cada posição do vetor, caracterizando o número de vezes que o disparo da transição t ocorreu na sequência de disparo σ. O vetor característico não considera a ordem de disparo das transições. Assim, diferentes sequências de disparos terão o mesmo vetor característico [4]. Dessa forma, o vetor característico da rede apresentada na figura 2.6 é $V_s^T = [11]$.

2.2.4 Conflito

O conflito ocorre quando um dos elementos de um pré-conjunto de uma transição é também um dos elementos do pré-conjunto de uma outra transição. Ou seja, duas ou mais transições cujos pré-conjuntos possuem interseção.

De acordo com Costa [11], um conflito é composto por um lugar com mais de uma transição de saída e o disparo de uma dessas transições desabilita as outras transições. Os conflitos são utilizados para modelar sistemas em que um determinado recurso material necessita ser utilizado por mais de uma linha de produção. É possível observar o conflito na figura 2.7 através das transições T_0 e T_1, pois ao disparar uma das duas transições, automaticamente, a outra será desabilitada.

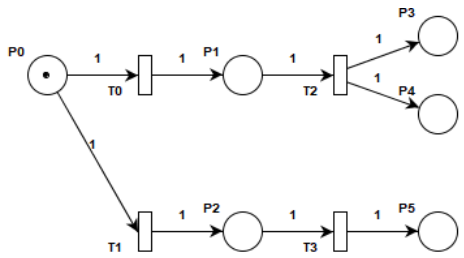

Figura 2.7: Conflito em uma rede de Petri.

2.2.5 Concorrência

Quando em uma rede de Petri existem duas transições que podem ser disparadas ao mesmo tempo, denomina-se concorrência. Ou seja, duas transições estão sensibilizadas ao mesmo tempo. Entretanto, em uma concorrência não pode existir conflito.

Pela definição, duas transições são concorrentes em uma rede de Petri se elas são casualmente independentes, de outro modo, uma pode disparar antes ou depois da outra [11]. Na figura 2.8 é possível observar a concorrência nas transições T_0 e T_1.

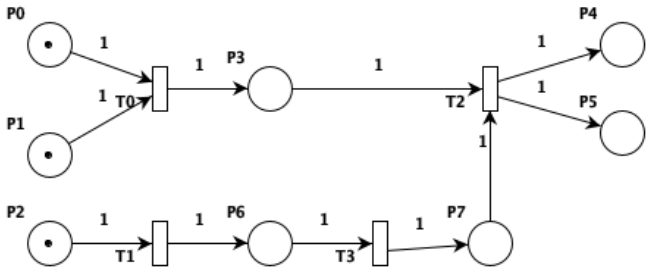

Figura 2.8: Concorrência em uma rede de Petri.

2.2.6 Propriedades

Alcançabilidade

A alcançabilidade é primordial para o estudo das propriedades dinâmicas seja qual for o sistema. O disparo de uma transição habilitada irá alterar a distribuição de fichas, em outras palavras, as marcações de uma rede. Uma sequência de disparos se transforma em uma sequência de marcações. Uma marcação M_n é dita alcançável a partir de uma

marcação M_0 apenas se existe uma sequência de disparos que transformam M_0 para M_n [12].

Um disparo ou uma sequência de ocorrências é denotado por $\sigma = M_0[t_1\rangle M_1[t_2\rangle M_2[...\rangle M_n$ ou apenas $\sigma = t_1t_2...t_n$. Neste caso, M_n é alcançável de M_0 por σ e denota-se $M_0[\sigma > M_n$. O conjunto de todas as possíveis marcações alcançáveis a partir de M_0 em uma rede (N, M_0) é denotado por $R(N, M_0)$ ou apenas $R(M_0)$. Além desse conjunto, tem-se o conjunto de todas as possíveis sequências de disparos partindo de M_0 em uma rede (N, M_0) que é denotada por $L(N, M_0)$ ou simplesmente $L(M_0)$ [12].

O problema da alcançabilidade para uma rede de Petri é encontrar se $M_n \in R(M_O)$ para uma dada marcação M_n em uma rede (N, M_0). Em algumas aplicações, apenas um dos subconjuntos de lugares pode ser interessante para determinada marcação e os demais lugares da rede podem não importar. Isso resulta no problema da sub-marcação alcançável, o qual é o problema de encontrar se $M'_n \in R(M_0)$, onde M'_n é qualquer subconjunto de lugares que pertence a uma marcação M_n [12].

O grafo de alcançabilidade é a representação gráfica do conjunto de marcações alcançáveis. No grafo, os nós são as marcações e os arcos representam as transições que foram disparadas e geraram uma marcação alcançável [10].

A figura 2.9 apresenta o grafo de alcançabilidade do sistema da figura 2.6. O s_0 é composto por $\{p_0, p_1\}$, o s_1 é composto por $\{p_2\}$ e o s_2 é composto por $\{p_3, p_4\}$.

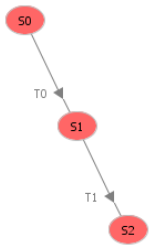

Figura 2.9: Grafo de alcançabilidade da figura 2.6.

Limitabilidade

Uma rede de Petri (N, M_0) é dita k-limitada ou apenas limitada se o número de fichas em cada lugar não excede um número finito k para qualquer marcação alcançável a partir de M_0, em outras palavras, $M(p) \leq k$ para cada lugar p e para cada marcação $M \in R(M_0)$. Uma rede de Petri (N, M_0) é dita segura se ela é 1-limitada [12]. A rede apresentada na figura 2.6, por exemplo, é uma rede de Petri 1-limitada.

2.2.7 Equação Fundamental

A definição da equação fundamental é dada por:

$M' = M_0 + C.\Omega$

Onde:

C representa a matriz de incidência que é determinada pela diferença entre o pós-conjunto e o pré-conjunto, isto é, $C = x \bullet - \bullet x$ [10]. A matriz de incidência do sistema da figura 2.6 é exemplificada a seguir.

$$Pre = \begin{array}{c} \begin{matrix} t_0 & t_1 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \qquad Pos = \begin{array}{c} \begin{matrix} t_0 & t_1 \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \begin{array}{l} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \qquad C = \begin{array}{c} \begin{matrix} t_0 & t_1 \end{matrix} \\ \begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{array} \begin{array}{l} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array}$$

Ω representa o vetor característico, também chamado de vetor Parick, que caracteriza a sequência de disparos de transições. Em outras palavras, seja $\sigma = t_1...t_k$ uma sequência de disparos de transições, $\Omega = (\#_{t1}\sigma, ..., \#_{tk}\sigma)$ sendo que $\#t_i$ é o número de ocorrências de t_i em $t_1...t_k$ [10].

Ainda de acordo Benito [10], através da equação fundamental é possível analisar a acessibilidade das marcações e a representação dos aspetos comportamentais da rede,

dado que esta especifica a dinâmica da inserção e remoção de marcas nos lugares além da sequência de disparos de transições.

2.2.8 Platform Independent Petri Net Editor

Platform Independent Petri Net Editor (PIPE) é um software de código aberto para criar e analisar redes de Petri [13]. Neste trabalho será utilizado o PIPE para desenhar a rede de Petri clássica gerada a partir dos diagramas já mencionados anteriormente além disso, será utilizado o PIPE para gerar o grafo de alcançabilidade para realizar as análises da rede desenvolvida.

2.2.9 Rede de Petri Colorida

Rede de Petri coloridas é a combinação entre a teoria da rede de Petri e a linguagem de programação funcional com a finalidade de solucionar os casos em que é necessário representar processos que são similares mas são distintos. A rede de Petri fornece a base formal para modelar concorrência e sincronização e a linguagem funcional fornece as primitivas para modelar a manipulação de dados e criação de modelos compactos e parametrizáveis [4].

As cores associadas a uma ficha (números inteiros ou conjunto de etiquetas) tem por objetivo diferenciar as fichas. Consequentemente, cada lugar se associa ao conjunto de cores das fichas que podem pertencer a este lugar. A cada transição é associado um conjunto de cores que equivale às diversas formas de disparar uma transição. Nos casos mais simples, quando todos os processos possuem a mesma estrutura e são independentes uns dos outros, as cores das transições estão diretamente relacionadas aos processos, e o conjunto de cores dos lugares e das transições são idênticos [9].

Formalmente, Cardoso e Valette [9] definem uma rede de Petri colorida associada a uma marcação inicial como uma sêxtupla:

$N_c =< P, T, C_{or}, C_{sc}, W, M_0 >$, onde:

- P é um conjunto finito de lugares;

- T é um conjunto finito de transições;
- C_{or} é um conjunto finito de cores;
- C_{sc} é a função subconjunto de cores que a cada lugar e a cada transição associa um subconjunto de C_{or} (cores possíveis para o lugar ou transição): $C_{sc} : P \cup T \rightarrow \rho(C_{or})$;
- W é a função de incidência: $W(p,t) : C_{sc}(t) \times C_{sc}(p) \rightarrow \mathbb{N}$;
- M_0 é a marcação inicial que associa, para cada lugar e para cada cor possível neste lugar, um número de fichas: $M_0(p) : C_{sc}(p) \rightarrow \mathbb{N}$.

Além disso, a rede de Petri colorida é composta por:

- Estrutura: uma rede de Petri;
- Inscrições: variáveis associadas aos lugares, transições e arcos;
- Declarações: tipos de variáveis, funções e operações sobre as variáveis.

Em uma rede de Petri colorida, um lugar só pode receber fichas que possuem valores que respeitam a cor associada ao lugar. As transições definem a dinâmica da rede de Petri colorida e podem apresentar expressões *booleanas* que indicam os tipos de fichas que podem ativar uma transição, restringindo assim o disparo das transições [4]. A figura 2.10 da autora Rezende [4] exemplifica uma rede de Petri colorida.

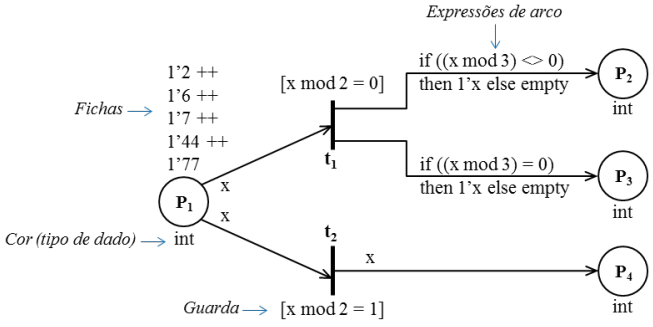

Figura 2.10: Rede de Petri Colorida.
(Autoria: Rezende [4])

2.2.10 CPN TOOLS

CPN TOOLS é uma ferramenta para editar, simular e analisar redes de Petri coloridas [14]. Neste trabalho será utilizado o CPN TOOLS para a transformação dos diagramas já mencionados anteriormente em uma rede de Petri colorida.

2.3 Estado da arte

Para o presente trabalho foi realizada uma pesquisa na ACM e IEEE, bases de dados de grande relevância na área da computação, utilizando as palavras chaves "UML" e *"Petri Net"*. Foram encontrados 67.770 resultados na base da ACM e 306 resultados na base da IEEE. A maioria das propostas mostram o mapeamento de um diagrama específico para um tipo de rede de Petri.

Portanto, escolheu-se diagrama de sequência e casos de uso para o mapeamento para uma Rede de Petri pois, estes diagramas possuem relação entre eles e permitem validar o diagrama de classes. Realizou-se então, uma pesquisa nas mesmas bases utilizando as palavras chaves *"Use case and Sequence Diagrams"* e *"Petri Nets"* obtendo um resultado de 555.037 na base da ACM e 25 na base da IEEE. Destes resultados, foram selecionados os trabalhos mais citados que serão descritos a seguir.

Os autores Shailesh, Nayak e Prasad [15] propuseram uma abordagem para transformar diagramas de sequência da UML/MARTE em redes de Petri temporizadas equivalentes. A transformação foi realizada utilizando a linguagem de transformação *Atlas Transformation Language* (ATL) e a linguagem de programação Java. Para realizar a transformação, os autores definiram os metamodelos de entrada e saída e especificaram as regras de transformação em ATL. Posteriormente, a transformação foi executada, seguindo os passos descritos no processo. Essa abordagem permitiu a análise e verificação do comportamento dos sistemas de *software* modelados, tornando-se uma técnica eficaz para o desenvolvimento de sistemas de *software* mais confiáveis e robustos.

No trabalho de Emadi e Shams [16], os autores propuseram uma abordagem inovadora para a transformação de modelos executáveis. Os autores propuseram um algoritmo

baseado na sintaxe geral das redes de Petri para converter um *software* modelado, descrito por um conjunto de caso de uso e diagramas de sequência, em modelos executáveis. A principal contribuição deste trabalho é a transformação de todas as estruturas e interfaces dos diagramas em componentes das redes de Petri, o que possibilita a execução do modelo. Essa abordagem apresenta vantagens como a deteção antecipada de erros e a facilidade de manutenção dos modelos. Além disso, o algoritmo proposto pode ser aplicado em diversas áreas, como a modelação de sistemas embarcados e a engenharia de *software*.

No trabalho de Alhroob, K. e Hossain [17] foi proposto uma nova metodologia para a representação de sistemas de *software* utilizando redes de Petri de alto nível. Nessa abordagem, os lugares contêm valores e os arcos das transições contêm expressões e variáveis, diferenciando-se das redes de Petri convencionais. A metodologia proposta pelos autores permite a transformação precisa de diagramas de sequência e classes da UML em redes de Petri de alto nível, o que possibilita uma representação mais fiel do sistema de *software*. Essa abordagem apresenta vantagens como a deteção antecipada de erros e a facilidade de verificação e análise do comportamento de um sistema de *software*.

Na proposta do trabalho de Ribeiro e Fernandes [18], utilizou-se os operadores de alto nível da UML 2.0 para transformar o diagrama de sequência em uma rede de Petri colorida. Primeiramente os autores consideraram os fragmentos de interação sem operadores e, então, consideraram para o diagrama de sequência a semântica de relação de ordem entre as mensagens. Cada mensagem do diagrama foi associada a uma transição, assim, o disparo da transição representa a execução de uma mensagem. Os lugares garantem a ordem entre os disparos das transições e representam o objeto de mudança durante a execução. Em seguida, os autores desenvolveram uma *Coloured Petri Net* (CPN) para cada operador de interação estabelecendo algumas regras.

Na proposta de Abrar [19] o autor apresentou uma abordagem para modelar requisitos de *software* de sistemas usando uma rede de Petri orientada a objetos. O autor utiliza diagramas de casos de uso da UML para modelar requisitos funcionais e cenários de caso de uso e, em seguida, realiza a modelação para a rede de Petri. Essa transformação de modelos possibilita a validação das especificações dos requisitos do mundo real de forma

efetiva e sistemática.

Marek [20] propôs uma ferramenta de *software* para modelação de sistemas que permite a criação de diagramas de sequência e instalação por meio de uma interface gráfica do utilizador. Posteriormente, ele transformou esses diagramas em uma rede de Petri estocástica, visando à verificação dos modelos criados.

Os autores Gómez, Rodríguez, Cambronero et al. [21] propuseram regras para modelar diagramas de sequência da UML em uma rede de Petri colorida, com o objetivo de detetar falhas nos diagramas modelados a fim de evitar erros nos estágios iniciais do desenvolvimento do sistema. Além disso, os autores desenvolveram uma ferramenta de transformação de modelo a modelo que segue as regras criadas e fornece a rede de Petri colorida do diagrama.

Os autores Silva, Salmon, Foyo et al. [22] propuseram uma análise e verificação de requisitos de *software* utilizando a técnica de mapeamento de diagramas da UML para uma rede de Petri. Adicionalmente, eles compararam essa abordagem com uma outra que utiliza a Engenharia de Requisitos Orientado a Objetivos (GORE), representada por diagramas KAOS. O objetivo principal era avaliar a eficácia das duas técnicas em termos de cobertura de requisitos e capacidade de identificar problemas em casos de estudo selecionados. A metodologia utilizada envolveu a transformação dos diagramas da UML e KAOS para redes de Petri e a execução de simulações para avaliar os requisitos do *software* em relação ao comportamento do sistema. Os resultados indicaram que a abordagem UML-Petri forneceu uma cobertura mais abrangente dos requisitos e permitiu a identificação mais precisa de problemas em comparação com a abordagem KAOS.

No trabalho de Baidada, Bouziane e Jakimi [23], os autores propõem uma abordagem para a validação de diagramas de sequência da UML por meio da sua equivalente rede de Petri. A abordagem desenvolvida pelos autores é capaz de transformar os diagramas de sequência, considerando operadores ALT, *Weak Sequencing* (SEQ), LOOP e PAR. O objetivo principal é fornecer uma técnica para verificar a correção do comportamento especificado pelo diagrama de sequência. Para isso, é utilizada a teoria da rede de Petri, que

permite modelar sistemas com concorrência e sincronização. A abordagem proposta é ilustrada em um estudo de caso com um sistema de controle de tráfego aéreo, demonstrando sua aplicabilidade na validação de diagramas de sequência de sistemas críticos.

Diversos autores buscam antecipar e detetar problemas de *software* através do mapeamento dos diagramas da UML para diversas redes de Petri. Com o objetivo de identificar os diagramas da UML mais frequentemente utilizados em mapeamentos para redes de Petri, os autores Vidal, Benito e Fernandes [24] conduziram uma pesquisa abrangente na literatura. Através da análise de diversos trabalhos que empregam a UML e as redes de Petri, os autores concluíram que a rede de Petri colorida em conjunto com o diagrama de casos de uso é a mais utilizada.

Capítulo 3

Regras para o mapeamento em redes de Petri seguras

Neste capítulo, serão apresentadas as regras desenvolvidas para o mapeamento dos diagramas da UML em uma rede de Petri segura. O capítulo foi dividido em duas seções: a seção 3.1 refere-se às regras criadas de mapeamento dos diagramas de casos de uso e a seção 3.2 refere-se às regras criadas de mapeamento dos diagramas de sequência.

3.1 Mapeamento Diagrama de Casos de Uso

O diagrama de casos de uso é uma ferramenta da UML que representa a interação entre o ator (ou atores) e o sistema, com o objetivo de alcançar um objetivo de interesse do ator. O principal objetivo desse diagrama é apresentar as funcionalidades gerais do sistema, sem se preocupar com fluxos de trabalho ou uma ordem de execução.

Para realizar o mapeamento deste diagrama em uma rede de Petri, é fundamental compreender como cada elemento do diagrama é representado na rede. Sendo assim, a seguir, será apresentado uma descrição detalhada do mapeamento dos elementos do diagrama de casos de uso para as redes de Petri.

Vale ressaltar que, ao realizar o mapeamento do diagrama de casos de uso para a rede de Petri segura, não é pretendido representar um fluxo de trabalho ou uma ordem de

execução, mas sim os passos que ocorrem na interação entre o ator e o caso de uso. Isso porque, a rede de Petri segura é uma ferramenta de análise formal do comportamento do *software*, que permite avaliar a corretude do sistema em diferentes cenários e condições. Dessa forma, o mapeamento deve refletir adequadamente as interações entre o ator e o sistema, para que se possa realizar uma análise precisa e eficiente do comportamento do *software*.

3.1.1 Regra 1 - Ator

No diagrama de casos de uso da UML, os atores desempenham um papel fundamental, pois representam as entidades externas que interagem com o sistema no contexto de seus casos de uso específicos. Desse modo, utilizou-se um lugar na rede Petri para representar cada ator conforme ilustrado na Figura 3.1.

3.1.2 Regra 2 - Casos de Uso

Os casos de uso descrevem as funcionalidades esperadas do sistema a ser desenvolvido. Para o mapeamento deste recurso para a rede de Petri, utilizou-se igualmente um lugar conforme descrito na figura 3.1.

Caso seja necessário representar o mesmo caso de uso mais de uma vez na rede de Petri, cria-se um lugar com o nome do caso de uso existente seguido de um numeral sequencial a partir de 0. Por exemplo, "casoUso0" representa o mesmo caso de uso que "casoUso". Tal abordagem permite uma melhor organização e visualização da interação entre os casos de uso e os demais elementos na rede de Petri.

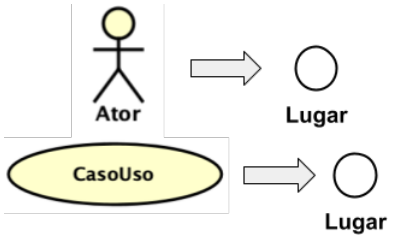

Figura 3.1: Mapeamento ator e caso de uso.

3.1.3 Regra 3 - Associações

Em um diagrama de casos de uso da UML, um ator é conectado a casos de uso via associações. Estas associações expressam que o ator se comunica com o sistema e utiliza certas funcionalidades. Para o mapeamento deste recurso propõe-se utilizar uma transição, como descrito na figura 3.2. O disparo desta transição representa a interação que está ocorrendo entre o ator e o sistema.

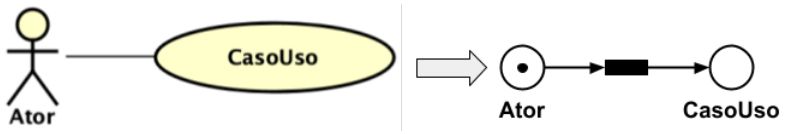

Figura 3.2: Mapeamento associação entre um ator e um caso de uso.

3.1.4 Regra 4 - Relacionamento entre atores

No diagrama de casos de uso, os atores podem ser representados em uma relação de herança (generalização) uns com os outros para expressar propriedades comuns entre estes atores.

Para mapear esse tipo de relacionamento para a rede de Petri, utilizou-se os atores como lugares. Em seguida, todas as associações existentes entre o super ator e seus casos de uso foram copiadas para o ator filho, inserindo cópias dos lugares desses casos de uso, conforme ilustrado na figura 3.3.

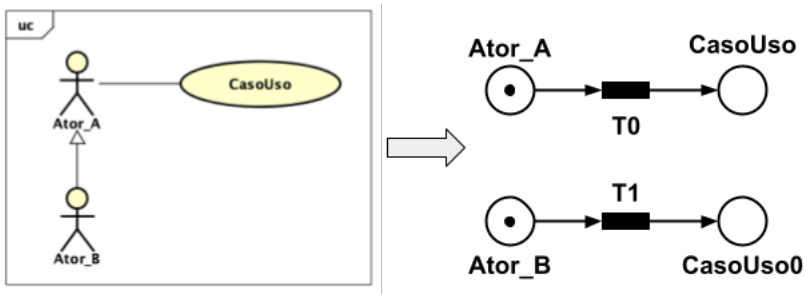

Figura 3.3: Mapeamento generalização entre atores.

3.1.5 Regra 5 - Mais de um ator associado a um mesmo caso de uso

No diagrama de casos de uso, é comum que mais de um ator esteja associado ao mesmo caso de uso. Para mapear esse tipo de relacionamento, o caso de uso é representado por um lugar, conforme mencionado anteriormente, e é criada uma cópia desse lugar para cada ator que deve interagir com ele.

Conforme dito anteriormente, quando é necessário representar o mesmo caso de uso na rede de Petri mais de uma vez, as cópias dos lugares do caso de uso terão o mesmo nome seguido por um numeral iniciando em 0. A figura 3.4 exemplifica o relacionamento de mais de um ator em um mesmo caso de uso na rede de Petri.

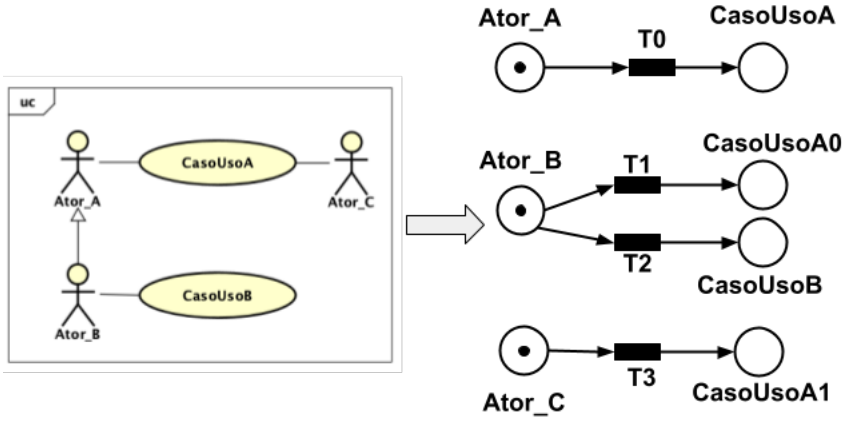

Figura 3.4: Mapeamento mais de um ator associado ao mesmo caso de uso.

3.1.6 Relacionamento entre casos de uso

- **Regra 6 - Inclusão:** o relacionamento de inclusão no diagrama de casos de uso indica que sempre que o caso de uso base for executado, o caso de uso que está sendo incluído também será executado, permitindo uma melhor modularização do sistema e evitando a repetição desnecessária de casos de uso em diferentes diagramas.

  Para realizar o mapeamento deste relacionamento para a rede de Petri, foi inserido o lugar do caso de uso incluído após uma transição de saída do caso de uso base.

Em seguida, foi criada uma cópia do lugar do caso de uso base, representando o mesmo caso de uso base na rede de Petri. Essa cópia é identificada com um numeral sequencial iniciado em 0, seguindo o mesmo padrão utilizado para o mapeamento de casos de uso em que é necessário representá-los mais de uma vez na rede de Petri. A figura 3.5 ilustra esse processo para o relacionamento de inclusão entre os casos de uso "Caso de Uso A"e "Caso de Uso B";

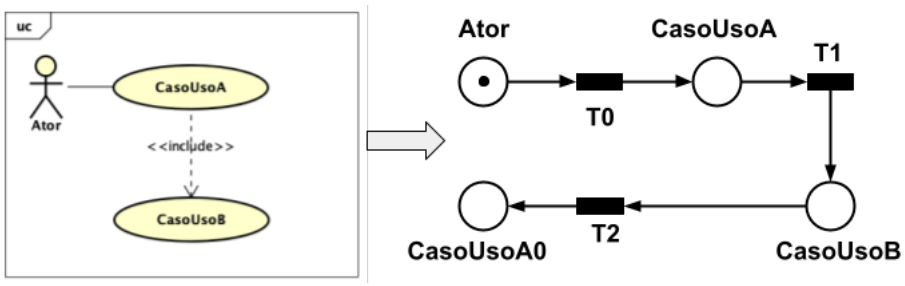

Figura 3.5: Mapeamento inclusão.

- **Regra 7 - Extensão:** O relacionamento de extensão no diagrama de casos de uso é utilizado para representar processos opcionais que podem ocorrer após a execução do primeiro caso de uso. Quando o processo de extensão é executado, é preciso verificar se a condição de extensão especificada foi atendida para que o caso de uso de extensão seja executado. Caso a condição não seja atendida, o fluxo de execução segue o caminho normal.

  Dessa forma, para realizar o mapeamento desse relacionamento inseriu-se um lugar para o caso de uso base seguindo a mesma lógica descrita nas seções anteriores. Em seguida, utilizou-se o conceito de conflito da rede de Petri. A partir do lugar do caso de uso base, foi inserida uma transição que possui como saída dois lugares. Um desses lugares representa o caso de uso base e o outro lugar representa o caso de uso estendido. O lugar do caso de uso estendido possui uma transição com saída para o caso de uso base também. A figura 3.6 ilustra o mapeamento do relacionamento de extensão para a rede de Petri;

- **Regra 8 - Generalização:** No diagrama de casos de uso, os casos de uso também

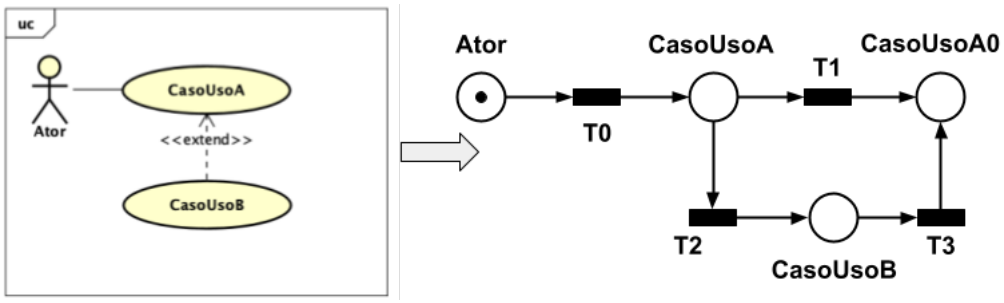

Figura 3.6: Mapeamento extensão.

podem ser representados em uma relação de generalização para expressar propriedades comuns entre diferentes casos de uso. Nesse tipo de relacionamento, o caso de uso especializado herda todos os comportamentos, estrutura e relacionamentos do caso de uso generalizado. Ou seja, quando existir um include ou extend no caso de uso generalizado, o mapeamento do caso de uso especializado apresentará esses relacionamentos também. Para realizar este mapeamento, foram inseridos todos os comportamentos de ambos os casos de uso. Primeiro, mapeia-se o comportamento do caso de uso generalizado. Em seguida, mapeia-se o comportamento do caso de uso especializado, conforme ilustrado na figura 3.7.

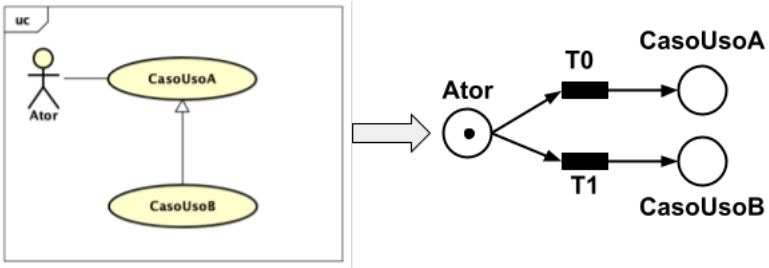

Figura 3.7: Mapeamento generalização caso de uso.

3.1.7 Exemplo detalhado do mapeamento do diagrama de casos de uso

A figura 3.8 apresenta um exemplo do diagrama de casos de uso para um sistema que envolve um utilizador e um cliente como atores principais. Os casos de uso incluem "fazer

login", "cadastrar cliente", "checar cliente", "enviar e-mail" e "receber e-mail".

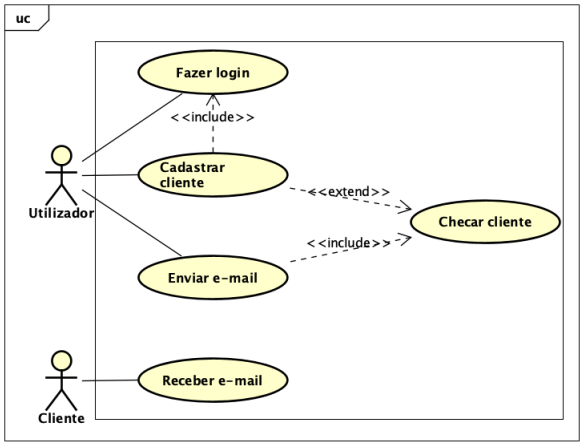

Figura 3.8: Exemplo diagrama de casos de uso.

Para mapear esse diagrama para a rede de Petri segura, cada ator e caso de uso foi mapeado como um lugar, seguindo as regras descritas nas seções anteriores. Além disso, foram incluídas transições na rede de Petri para representar as associações entre os atores e os casos de uso. Por exemplo, a transição "T0" conecta o lugar "Utilizador" ao caso de uso "FazerLogin", indicando que o utilizador pode interagir com esse caso de uso. Outras transições, como "T4" e "T12" representam as interações do "Utilizador" com "EnviarE-mail" e "Cliente" com "Receber e-mail", respectivamente. A transição "T5" representa o evento "ChecarCliente", que pode ser desencadeado pelos casos de uso anteriores.

Esse mapeamento, possibilita a visualização de forma clara como os atores interagem com os casos de uso em diferentes etapas do sistema. A figura 3.9 ilustra a rede de Petri mapeada para este diagrama.

É importante destacar que na figura 3.8, o caso de uso enviar e-mail possui uma inclusão de checar cliente. Na rede de Petri (figura 3.9) é possível notar que o utilizador chama o caso de uso "EnviarEmail" através da transição T4 e este lugar possui uma transição de saída para o lugar de "ChecarCliente".

Além disso, "ChecarCliente" possui também uma extensão com "CadastrarCliente". Devido a isso, o lugar "ChecarCliente" possui um conflito com duas transições, a transição "T6" é o retorno ao caso de uso base da inclusão e a transição "T7" é a execução da extensão

de cadastrar cliente.

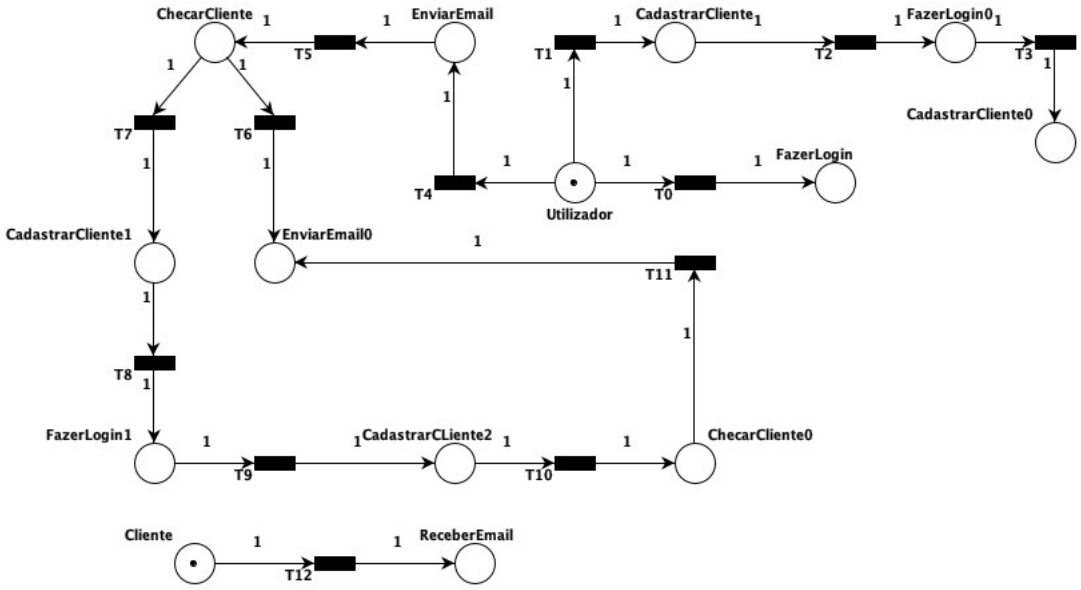

Figura 3.9: Mapeamento diagrama de casos de uso.

3.2 Mapeamento Diagrama de Sequência

O diagrama de sequência é uma ferramenta de modelação da UML utilizada para descrever as interações entre os objetos do sistema, representando como as mensagens e dados são trocados entre eles. Esses objetos podem ser tanto humanos como não humanos, como, por exemplo, um servidor.

Para a representação deste diagrama em uma rede de Petri, uma abordagem que considera a sequência dos eventos presentes no diagrama é proposta. Com a introdução dos operadores de alto nível nos diagramas de sequência da UML 2.0, como o LOOP, a abordagem de mapeamento criada baseia-se nessas atualizações. A seguir, será apresentado uma descrição detalhada do mapeamento dos elementos do diagrama de sequência para as redes de Petri.

3.2.1 Regra 9 - Parceiros de Interação

No diagrama de sequência da UML todas as interações que ocorrem são por meio dos parceiros de interação. Na rede de Petri, o lugar é interpretado como uma condição, um estado parcial, uma espera, um procedimento, um conjunto de recursos, entre outros. Desse modo, utilizou-se um lugar na rede de Petri para representar cada parceiro de interação conforme ilustrado na figura 3.10.

Quando um parceiro de interação já está representado por um lugar na rede de Petri, mas é necessário utilizá-lo novamente para interações subsequentes, é criada uma cópia desse lugar com o mesmo nome seguido de um numeral sequencial a partir de 0. Por exemplo, "LinhaVidaA0" representa o mesmo parceiro de interação "LinhaVidaA". Tal abordagem permite uma melhor organização e visualização da interação entre objetos de interação na rede de Petri.

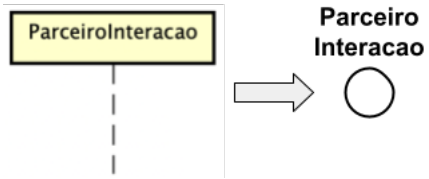

Figura 3.10: Mapeamento parceiros de interação.

3.2.2 Regra 10 - Mensagens

No diagrama de sequência, uma mensagem é retratada por uma seta do remetente para o destinatário. Na rede de Petri este componente foi mapeado através de uma transição, como ilustrado na figura 3.11.

No diagrama de sequência, o tipo da seta da mensagem expressa o tipo da comunicação envolvida. Para o mapeamento, as mensagens síncronas possuem o retorno para o parceiro de interação mesmo quando não há esse retorno explícito. Para mensagens assíncronas, a mensagem apenas é enviada.

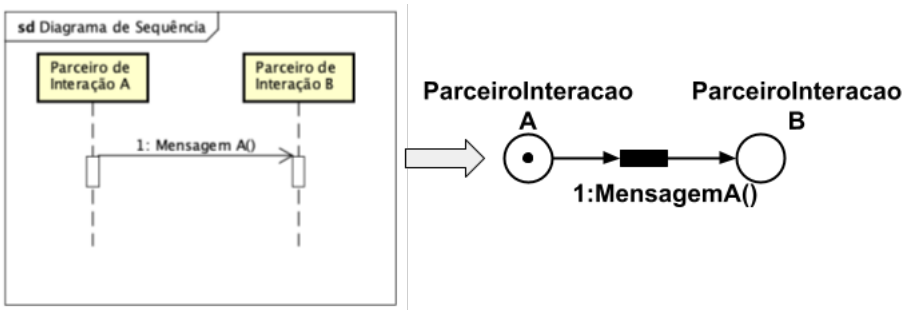

Figura 3.11: Mapeamento mensagem.

3.2.3 Regra 11 - Associações expressas na linha de vida

A linha de vida representa a sequência e duração das mensagens em um diagrama de sequência. Ela pode descrever a ocorrência de um evento que não está explícito no diagrama através do envio de uma mensagem. Essa associação entre dois objetos de interação é possível ser visualizada no diagrama de sequência através da duração da linha de vida.

Nos casos em que a linha de vida indica uma interação entre parceiros de interação, mas não há o envio explícito de mensagens, essa interação é mapeada utilizando uma transição "TRLife", como demonstrado na Figura 3.12.

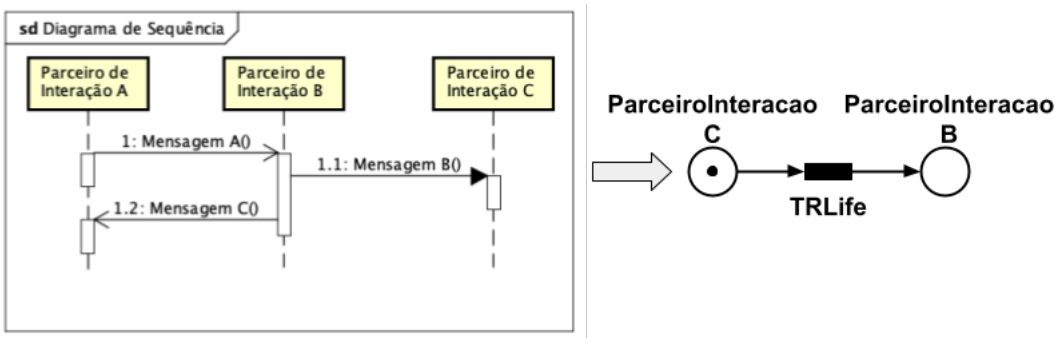

Figura 3.12: Mapeamento linha de vida.

Na figura 3.12, é possível observar que existe uma interação da Linha de Vida C para a Linha de Vida B mesmo não tendo o envio explícito de uma mensagem do elemento C para o B, dessa forma utiliza-se a transição TRLife na rede de Petri.

3.2.4 Fragmentos de Interação

No diagrama de sequência da UML, é possível utilizar fragmentos combinados para modelar estruturas de controle. Esses fragmentos, como OPT e LOOP, permitem descrever de forma compacta e precisa os possíveis caminhos de execução do sistema. Com a utilização desses fragmentos, é possível modelar cenários mais complexos e evitar a repetição de sequências de mensagens em diferentes caminhos. A modelação desses fragmentos no diagrama de sequência pode ajudar na compreensão do comportamento do sistema e facilitar a identificação de erros de lógica.

- **Regra 12 - REF:** no diagrama de sequência da UML, é possível utilizar referências de interação para integrar um diagrama de sequência em outro. Isso permite reutilizar interações já modeladas e dividir sequências de interação complexas em módulos mais simples. Para mapear esse elemento em uma rede de Petri, propõe-se inserir um lugar com o nome do diagrama referenciado e uma transição auxiliar "TRRef". Ao disparar essa transição, é possível inserir o restante do fluxo do diagrama, como demonstrado na figura 3.13.

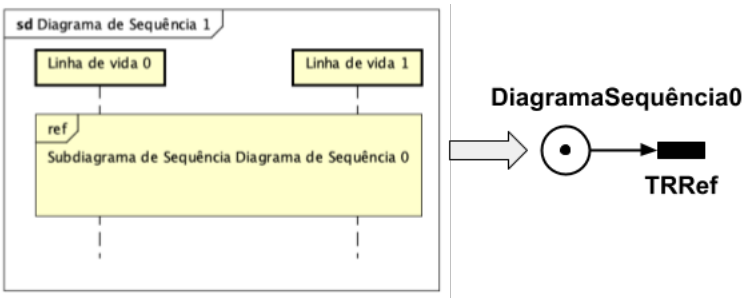

Figura 3.13: Mapeamento REF.

Na figura 3.13, o diagrama de sequência 1 faz referência ao diagrama de sequência 0. Desse modo, na rede de Petri, foi inserido um lugar "DiagramaSequencia0" e, em seguida, inserida uma transição "TRRef". O disparo desta transição leva para o restante do fluxo do diagrama ilustrado na figura 3.13.

Um exemplo mais detalhado pode ser visto na figura 3.15, o diagrama de cadastrar

cliente faz referência ao diagrama de realizar login e após o login do utilizador, executa o cadastro do cliente. Isso ocorre, pois, para cadastrar um cliente é necessário primeiro o utilizador estar logado no sistema. Dessa forma, na rede de Petri do diagrama, mapeou-se a realização do login com um lugar "RealizarLogin" e uma transição "TRRef" e o disparo desta transição leva para o lugar "Utilizador" que continua o fluxo do cadastro de cliente;

- **Regra 13 - ALT:** este fragmento determina que o fragmento combinado representa uma escolha entre dois ou mais comportamentos, possuindo pelo menos dois operandos. Cada operando representa um caminho alternativo na execução e cada um possui uma guarda. A guarda é uma expressão *booleana* entre colchetes. Se não tem guarda, então [*true*] é assumido como valor *default*.

  Para realizar o mapeamento deste elemento, utilizou-se o conceito do conflito de uma rede de Petri. Para descrever as alternativas, foi incluído transições de saída do lugar onde existem essas alternativas. Cada transição representa uma das alternativas. No diagrama de sequência da UML, o fragmento ALT determina alternativas que o utilizador pode escolher apenas 1. Desse modo, elas foram representadas nas redes de Petri com os conceitos de conflito, pois o conflito é composto por um lugar com mais de uma transição de saída e o disparo de uma dessas transições desabilita as outras transições. Este mapeamento pode ser visualizado na figura 3.14.

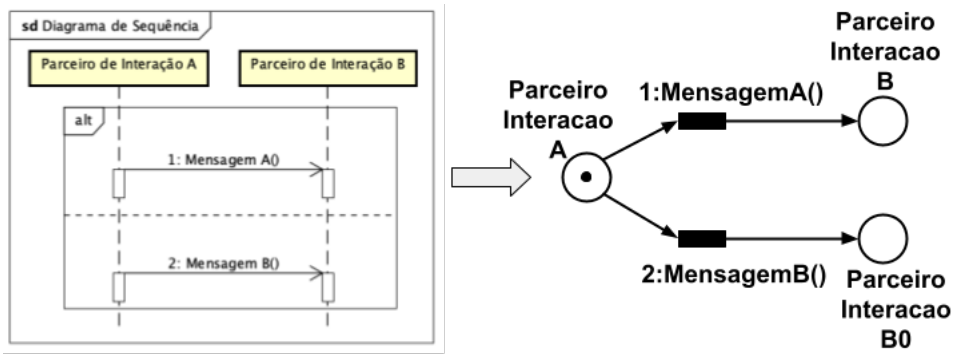

Figura 3.14: Mapeamento ALT.

A figura 3.15 apresenta um exemplo do diagrama de sequência com os operadores

de interação REF e ALT para um processo de cadastrar cliente.

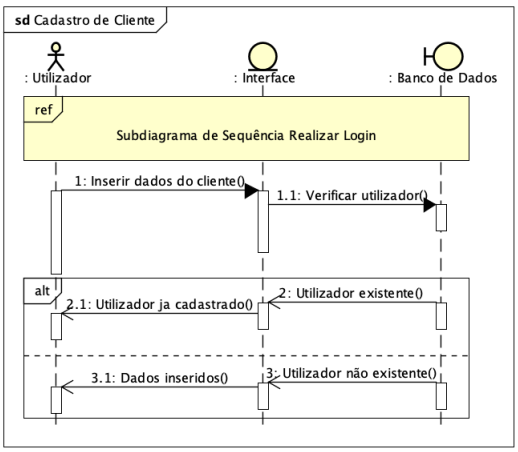

Figura 3.15: Diagrama de Sequência Cadastro de cliente.

Na figura 3.15 é possível observar que a fronteira banco de dados possui duas alternativas no momento de verificar os dados do cliente. A primeira alternativa é o utilizador existente e a segunda alternativa é o utilizador não existente. Dessa forma, a partir do lugar "BancoDados", foram puxadas duas transições. A primeira transição representa o fluxo da primeira e a segunda transição representa o fluxo da segunda alternativa.

O mapeamento deste diagrama foi realizado seguindo todas as regras descritas anteriormente. A figura 3.16 ilustra a rede de Petri mapeada desse diagrama;

- **Regra 14 - PAR:** esse fragmento define que o fragmento combinado retrata uma execução paralela de dois ou mais comportamentos. O fragmento PAR permite deixar de lado qualquer ordem cronológica entre mensagens em operandos diferentes. Para o mapeamento desse fragmento foram utilizados os conceitos de concorrência e paralelismo da rede de Petri, onde duas transições estão ativas e podem ser disparadas paralelamente, como ilustrado na figura 3.17.

  A figura 3.18 ilustra um exemplo do diagrama de sequência para um processo de dirigir. Este diagrama de sequência possui um fragmento paralelo para descrever os processos paralelos de soltar embreagem e pressionar acelerador.

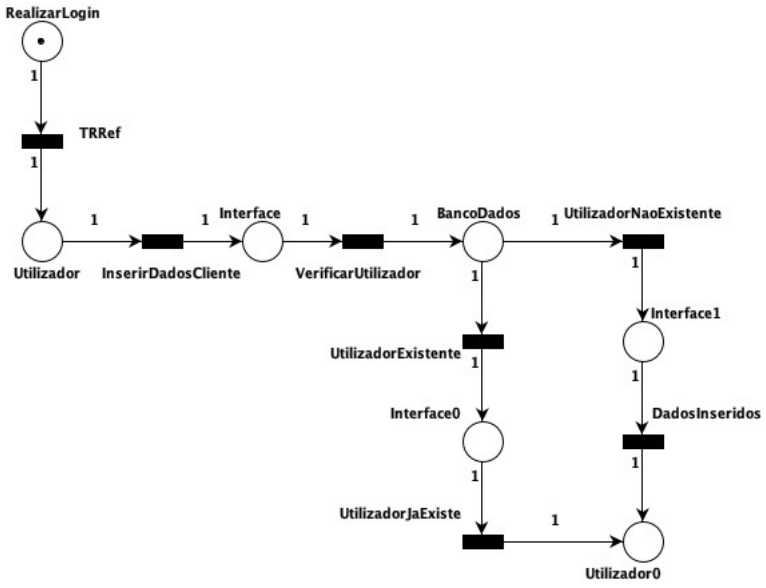

Figura 3.16: Rede de Petri Cadastro de cliente.

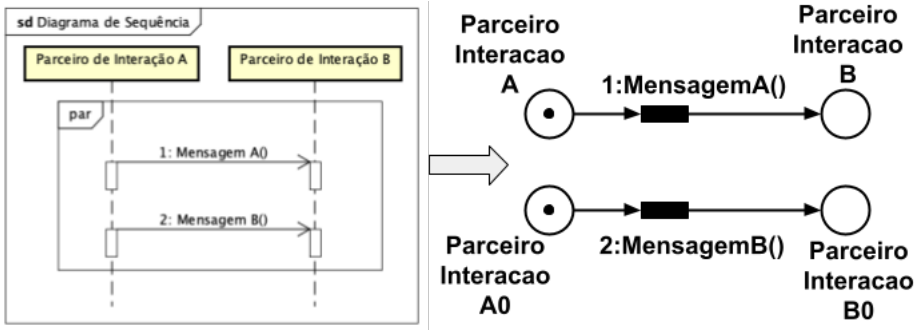

Figura 3.17: Mapeamento PAR.

O diagrama de sequência para um processo de dirigir ilustrado na figura 3.18 possui um fragmento paralelo para descrever os processos paralelos de soltar a embreagem e pressionar o acelerador. Para mapear esse diagrama para uma rede de Petri, foram inseridos dois lugares, "Motorista" e "Motorista0", que representam o motorista. Cada um desses lugares leva para uma transição diferente, "soltarEmbreagem"e "pressionarAcelerador", que inserem uma ficha em "Carro" e "Carro0", respectivamente.

A transição da linha de vida "TRLife" possui como entrada os lugares "Carro" e "Carro0", o que define que é preciso que cada um desses lugares insira uma ficha

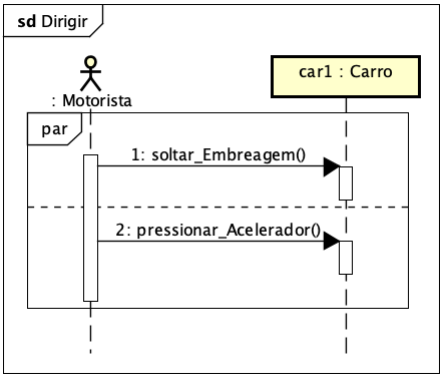

Figura 3.18: Diagrama de Sequência Dirigir **(Autoria: Guedes [3])**.

na transição para que ela possa ser disparada e levar para o lugar que finaliza o paralelismo, neste caso, o "Motorista". Isso permite que cada processo de soltar a embreagem e pressionar o acelerador seja executado independentemente e paralelamente.

Portanto, esse mapeamento permite visualizar claramente como representar um processo paralelo na rede de Petri, utilizando lugares, transições e a marcação das fichas. A figura 3.19 ilustra a rede de Petri mapeada desse diagrama;

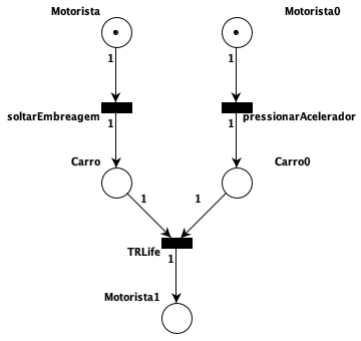

Figura 3.19: Rede de Petri Dirigir.

- **Regra 15 - LOOP:** esse fragmento representa uma ação que poderá ser executada uma ou várias vezes, possuindo exatamente um operando. A palavra-chave LOOP é seguida por uma especificação opcional do número de iterações do LOOP. Esta especificação assume a forma (min..max) ou (min,max), onde min específica o

número mínimo de iterações que o LOOP deve percorrer e max denota o número máximo de iterações.

Para o mapeamento foi realizado um LOOP que exemplifica a ação que pode ser executada várias vezes, inserindo uma transição que conecta dois lugares, como pode ser visualizado na figura 3.20 com a transição "TRLoop" entre o lugar "ParceiroInteracaoA" e "ParceiroInteracaoB" retornando ao início e possibilitando a ação ser feita novamente.

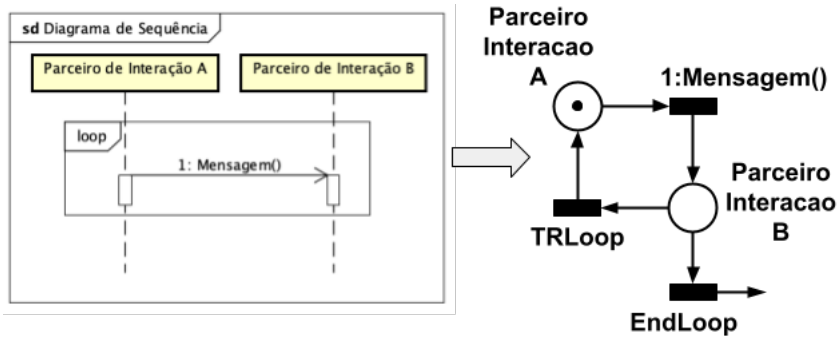

Figura 3.20: Mapeamento LOOP.

Nesta figura, a mensagem enviada entre os parceiros de interação possui um LOOP. Na rede de Petri, o lugar "ParceiroInteracaoA" dispara uma transição com a mensagem para o "ParceiroInteracaoB". O LOOP é representado na transição que está entre esses lugares, possibilitando o envio de "1:Mensagem()" novamente. Caso seja finalizado o LOOP, a transição "EndLoop" permite finalizar o processo;

- **Regra 16 - Break:** esse fragmento interrompe a execução normal do processo e consiste em um único operando e uma guarda. Se a guarda for verdadeira, as operações dentro do operando serão executadas, as operações do fragmento envolvente serão omitidas e a interação continuará no fragmento de nível superior. Caso não haja um fragmento de nível superior, quando o operador BREAK é finalizado, a execução do diagrama de sequência será encerrada.

  Para mapear esse fragmento, foi utilizada uma transição que representa o envio da mensagem contida no fragmento BREAK entre os lugares que representam os

objetos envolvidos no processo e finaliza no lugar que representa o objeto que encerra a execução da sequência no diagrama, como ilustrado na figura 3.21.

Na figura 3.21, é possível observar o mapeamento do fragmento BREAK. A transição "1: Mensagem()" dispara o envio da mensagem do "ParceiroInteracaoA" para o "ParceiroInteracaoB", e depois disso, o processo é finalizado.

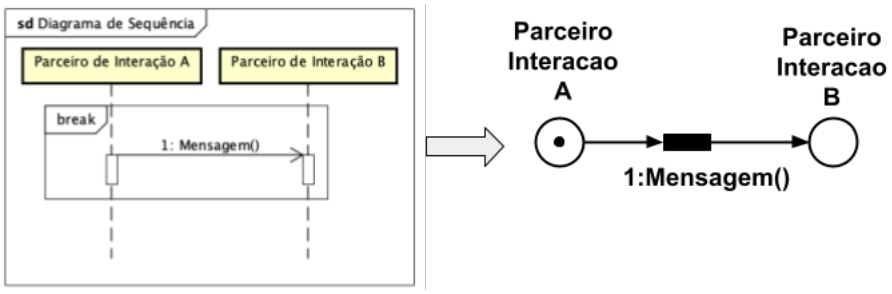

Figura 3.21: Mapeamento BREAK.

- **Regra 17 - OPT:** o fragmento OPT retrata a execução opcional de um evento, ou seja, o comportamento do fragmento pode ou não ser executado. Para o mapeamento deste fragmento utilizou-se também o conceito de conflito, possibilitando a execução do fragmento OPT mas também possibilitando a execução das demais operações sem ser necessário executar o OPT. A figura 3.22 ilustra o mapeamento do fragmento OPT em que o lugar "ParceiroInteracaoA" possui duas transições de saída. Uma dessas transições leva para a execução do operador OPT e após finalizar esta execução leva para a execução do restante do fluxo que é o envio da mensagem B. Caso o OPT não seja executado, será disparada a transição que leva para o envio da mensagem B diretamente.

  A figura 3.23 apresenta um exemplo do diagrama de sequência com o operador OPT para o processo de cadastrar produtos.

  No diagrama da figura 3.23, o utilizador possui como opção inserir fotos do produto ao cadastrar um novo produto. Dessa forma, para realizar o mapeamento, os dados do produto são inseridos e depois de inseridos, o local "Utilizador0" possui duas transições, uma que finaliza o processo com uma transição auxiliar "TRLife" e outra

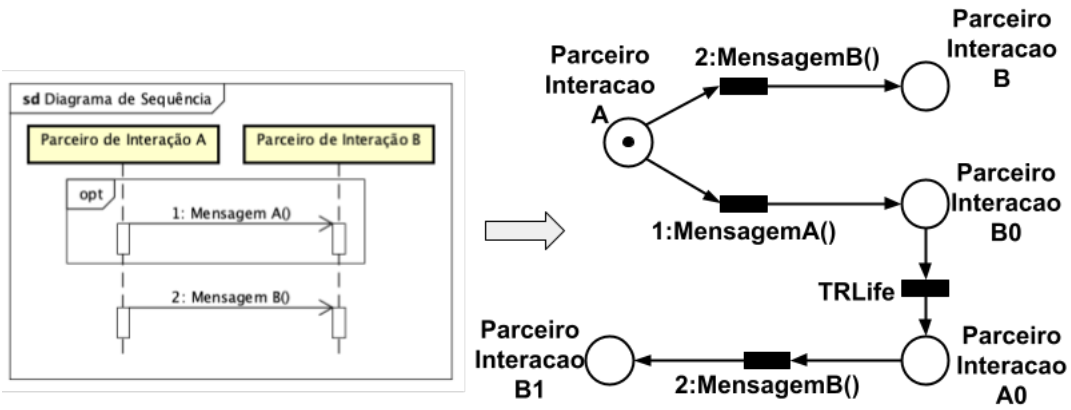

Figura 3.22: Mapeamento OPT.

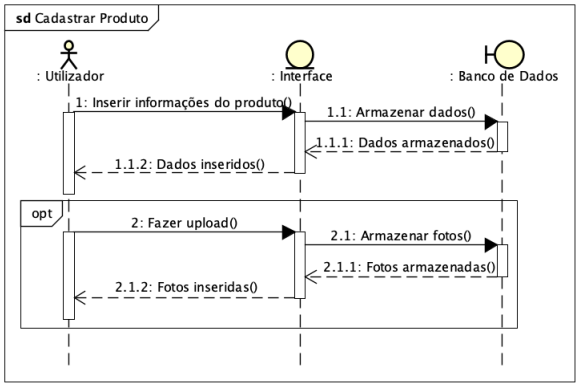

Figura 3.23: Diagrama de Sequência Cadastrar Produto.

que leva para o fluxo de inserir fotos do produto finalizando no mesmo lugar que a transição "TRLife".

O mapeamento deste diagrama foi realizado seguindo todas as regras descritas anteriormente. A figura 3.24 ilustra a rede de Petri mapeada desse diagrama.

3.2.5 Exemplo detalhado do mapeamento do diagrama de sequência

A figura 3.25 ilustra um diagrama de sequência mais detalhado com os operadores ALT, BREAK e LOOP para um processo de enviar e-mail.

Para realizar o mapeamento do diagrama da figura 3.25, cada parceiro de interação foi

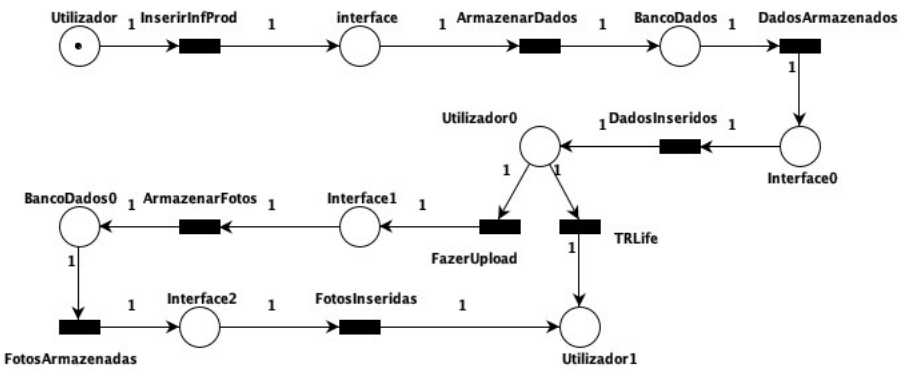

Figura 3.24: Rede de Petri Cadastrar Produto.

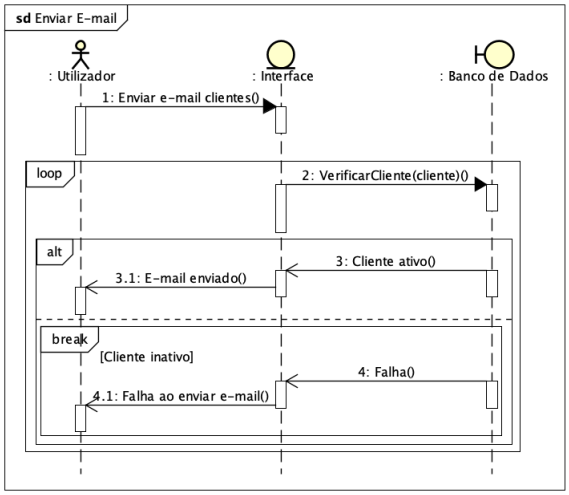

Figura 3.25: Diagrama de Sequência Enviar E-mail.

mapeado como um lugar, seguindo as regras descritas anteriormente. Além disso, foram incluídas transições na rede de Petri para representar as interações ou envio de mensagens entre os objetos.

É importante destacar o mapeamento dos operadores de alto nível presentes neste diagrama. O operador BREAK é representado quando o cliente é inativo e o e-mail não poderá ser enviado. Para o mapeamento, o banco de dados possui uma transição de saída "ClienteInativo" e depois "Interface1" possui uma transição de "FalhaEnviarEmail" e finaliza o processo.

É possível observar na figura 3.25 que o lugar "Utilizador0" possui uma transição "TR-Loop" levando para o lugar "Interface" que retorna o fluxo para verificar se o cliente está

ativo. Se o cliente for inativo, o processo irá finalizar na transição de saída "FalhaEnviarEmail" inserindo uma ficha no lugar "Utilizador1". A figura 3.26 ilustra a rede de Petri mapeada do diagrama de sequência de enviar e-mail;

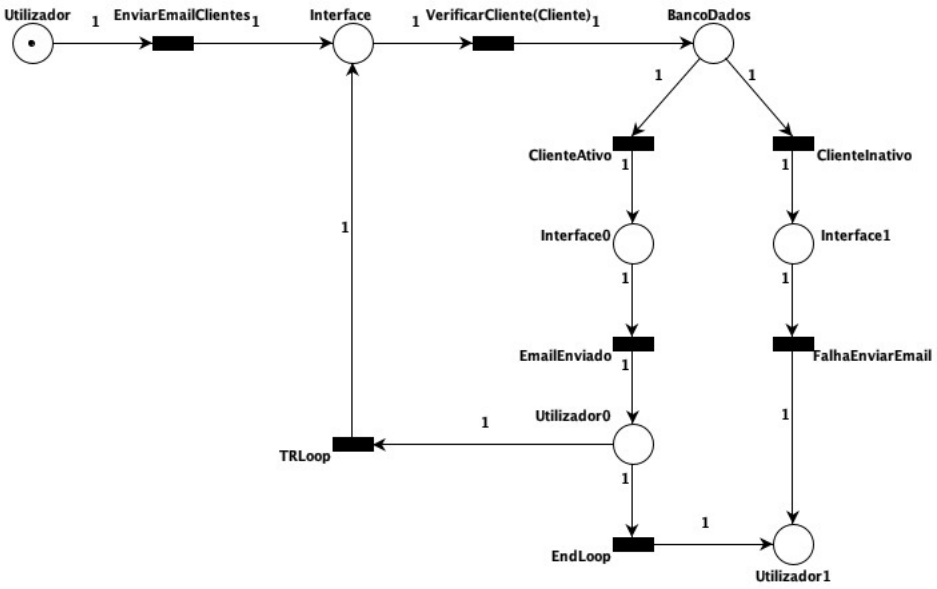

Figura 3.26: Rede de Petri Enviar E-mail.

Esse mapeamento representa de forma clara e concisa as interações entre os objetos do sistema, permitindo a análise e verificação da lógica de funcionamento do sistema representado pelo diagrama de sequência.

3.2.6 Limitações da rede de Petri segura

A rede de Petri clássica 1-limitada (rede segura) não possui recursos para representar o número mínimo e máximo de interações de um LOOP, sendo possível realizar o mapeamento de LOOP apenas quando não há um número máximo e mínimo de interações. Isso ocorre devido ao fato da rede ser controlada pela ficha presente no lugar habilitando ou não uma transição. Como em uma rede segura, apenas pode existir 1 ficha por vez nos lugares e todas as transições precisam de 1 ficha para serem disparadas, não tem como representar o número mínimo e máximo das interações. Outra limitação da rede segura é o fato de não detalhar a condição de disparo da transição, uma vez que para disparar a

transição só é necessário ter 1 ficha. Ou seja, não é possível expressar com mais detalhes as condições de disparo da transição, através do tipo da ficha ou algo do gênero.

Capítulo 4

Estudo de caso

Neste capítulo será realizada a prática das regras elaboradas no capítulo anterior. Para isso, utilizou-se diagramas de um trabalho encontrado a partir de pesquisas realizadas no Google. O trabalho de Souza [25], consiste em uma monografia de especialização da Universidade Tecnológica Federal do Paraná para o departamento de Informática e realizou um protótipo de aplicativo mobile a fim de gerenciar relatórios para a Polícia Militar do estado do Paraná.

Além disso, será realizado o mapeamento dos mesmos diagramas para uma rede de Petri colorida equivalente baseando-se nas regras criadas por Ribeiro, Fernandes, Tjell et al. [7]. Por fim, será feito as análises formais dos grafos de alcançabilidade de cada uma das redes.

O capítulo foi dividido em sete seções: a seção 4.1 refere-se ao mapeamento do diagrama de casos de uso para a rede de Petri segura, a seção 4.2 refere-se ao mapeamento do diagrama de sequência para a rede de Petri segura, a seção 4.3 refere-se às análises das redes de Petri seguras mapeadas. A seção 4.4 refere-se ao mapeamento do diagrama de casos de uso para a rede de Petri colorida, a seção 4.5 refere-se ao mapeamento do diagrama de sequência para a rede de Petri colorida, a seção 4.4 refere-se às análises das redes de Petri coloridas mapeadas e a seção 4.7 refere-se à discussão dos resultados obtidos e comparação das duas abordagens.

4.1 Mapeamento Diagrama de Casos de Uso para Rede de Petri Segura

De acordo com Souza [25] os casos de uso do sistema foram desenvolvidos com base nos requisitos funcionais do sistema. Neste trabalho, para o mapeamento do diagrama de casos de uso do aplicativo para a polícia, como já dito anteriormente, todos os atores e casos de uso foram mapeados como lugares na rede de Petri e a interação entre eles foram mapeados através de transições entre o ator e o caso de uso.

A figura 4.1, ilustra o diagrama de casos de uso definido por Souza [25] no sistema para a Polícia Militar do Paraná. Desse modo, para realizar o mapeamento desse diagrama para uma rede de Petri segura, o ator policial e os casos de uso foram mapeados como lugares na rede de Petri. As associações que ocorrem entre o ator e os casos de uso foram mapeadas através das transições.

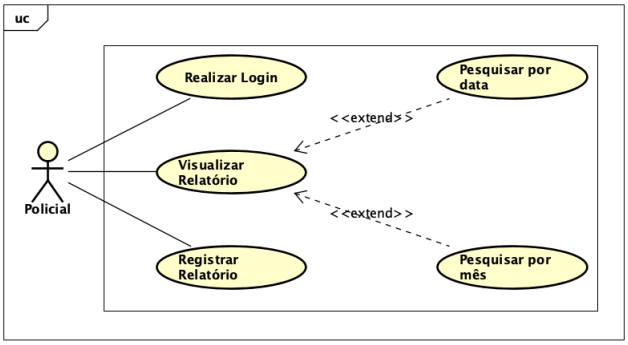

Figura 4.1: Diagrama de Casos de Uso **(Autoria: Souza [25])**.

A figura 4.2 ilustra a rede de Petri mapeada a partir do diagrama ilustrado na figura 4.1. Tal abordagem permite uma melhor visualização da interação entre o ator e o sistema, bem como facilita a análise formal do comportamento do software em questão.

Ao analisar a figura 4.2, é possível observar que o ator policial, representado como um lugar, interage com outros lugares, como registrar relatório, realizar login, pesquisar por mês, visualizar relatório e pesquisar por data. Cada interação é exemplificada pelo disparo das transições, onde o ator policial aciona a transição T1, por exemplo, e é levado

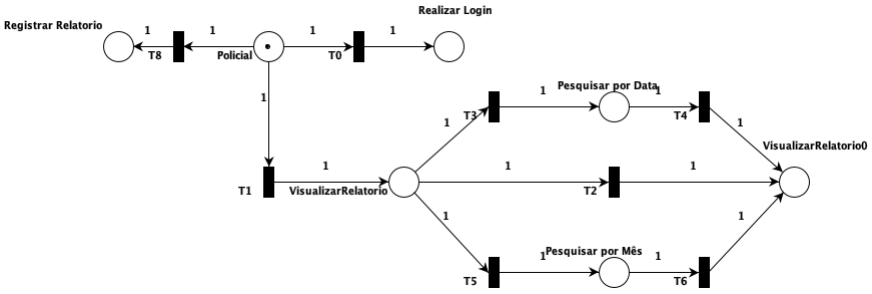

Figura 4.2: Rede de Petri Diagrama de Casos de Uso.

ao lugar de visualizar relatório. O caso de uso "Visualizar Relatório" possui duas extensões: pesquisar por data e pesquisar por mês. Dessa forma, foram incluídas três transições no caso de uso "Visualizar Relatório": uma que leva à execução da extensão de pesquisar por data, outra que leva à execução da extensão de pesquisar por mês e outra que leva ao caso de uso base sem executar as extensões.

4.2 Mapeamento Diagrama de Sequência para Rede de Petri Segura

Nesta seção serão apresentados os mapeamentos dos diagramas de sequência do trabalho utilizado como base (o protótipo da Polícia Militar) para a rede de Petri segura.

4.2.1 Realizar Login

A figura 4.3 ilustra o diagrama de sequência de “Realizar Login” definido por Souza [25] no sistema para a Polícia Militar do Paraná. Para realizar o mapeamento do diagrama de sequência “Realizar Login” em uma rede de Petri segura foram realizados os seguintes passos:

1. Para este diagrama, os elementos "Policial", "CIntPolLogin", "CCtrlLoginPol", "CPolicial", "CIntSBD"e "SBD" foram mapeados como lugares e as mensagens que ocorrem entre eles foram mapeados como transições;

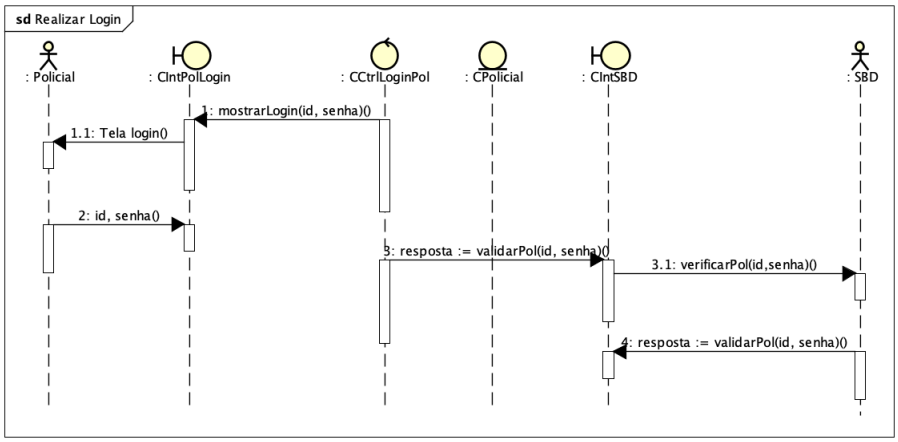

Figura 4.3: Realizar login **(Autoria: Souza [25])**.

2. Inicia se o mapeamento através do ":CCtrlLoginPol" transformando-o em um lugar na rede de Petri e inserindo uma ficha nele, pois é onde inicia a troca de mensagens no diagrama;

3. Este controle envia uma mensagem para o lugar ":CIntPolLogin", que também foi adicionado à rede. Entre esses dois lugares, uma transição foi criada para representar a mensagem "mostrarLogin(id,senha)", enviada do controle ":CCtrlLoginPol"para o lugar ":CIntPolLogin";

4. ": CIntPolLogin" envia outra mensagem através da transição "telaLogin" para o ator "Policial" (também um lugar na rede);

5. "Policial" envia para "CIntPolLogin0" a mensagem "2: id,senha" através da transição "idSenha";

6. Neste momento, tem-se o lugar "CIntPolLogin0" representando a fronteira "CIntPolLogin" porém a próxima mensagem enviada no diagrama é a partir de "CCtrlLoginPol". Dessa forma, é inserida uma transição "TRLife" para representar a linha de vida e a partir dela inserido um lugar "CCtrlLoginPol0". Após este lugar, é inserida a transição "validarPol(id,senha)". Em seguida, insere-se o lugar "CIntSBD";

7. "CIntSBD" envia então a mensagem "3.1: verificarPol(id,senha)" através da transição "verificarPol(id,senha)" para "SBD";

8. "SBD" envia a mensagem "4: resposta" através da transição "resposta" para o lugar "CIntSBD0";

9. Como as mensagens são síncronas, elas aguardam respostas. Após a mensagem "4: resposta" enviada para o ":CIntSBD" retorna-se para ":CCtrlLoginPol" a resposta da mensagem "3: resposta := validarPol(id,senha)". Dessa forma, foi inserida a transição "TRLife0" representando a linha de vida e o lugar "CCtrlLoginPol1";

10. A rede inicia com 1 ficha inserida em "CCtrlLoginPol", representado como um lugar na rede de Petri. Este lugar insere uma ficha na transição "mostrarLogin(id, senha)" tornando possível a mesma ser disparada. O disparo desta transição insere uma ficha em "CIntPolLogin", sensibilizando a transição "telaLogin";

11. A transição "telaLogin", quando disparada, insere uma ficha em "Policial" que sensibiliza a transição "idSenha". A sequência dos fatos são mapeadas com lugares e transições até concluir no lugar "CCtrlLoginPol1".

A figura 4.4 ilustra a rede de Petri segura mapeada a partir do diagrama de sequência "Realizar Login" ilustrado na figura 4.3.

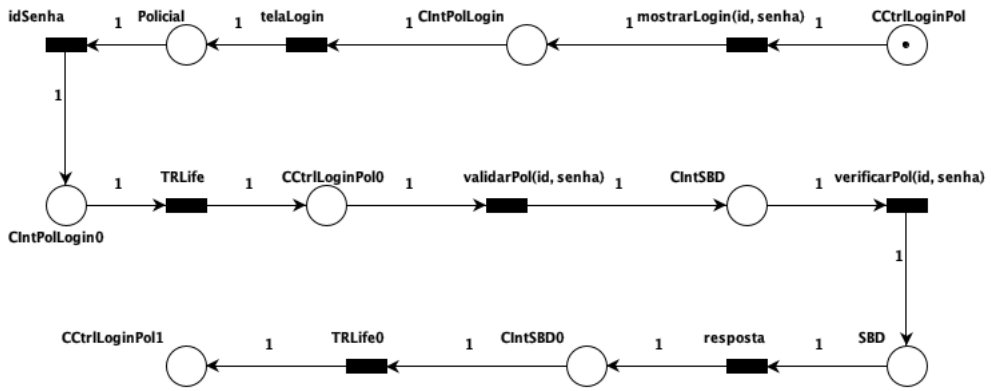

Figura 4.4: Rede de Petri Realizar Login.

4.2.2 Registrar Relatório

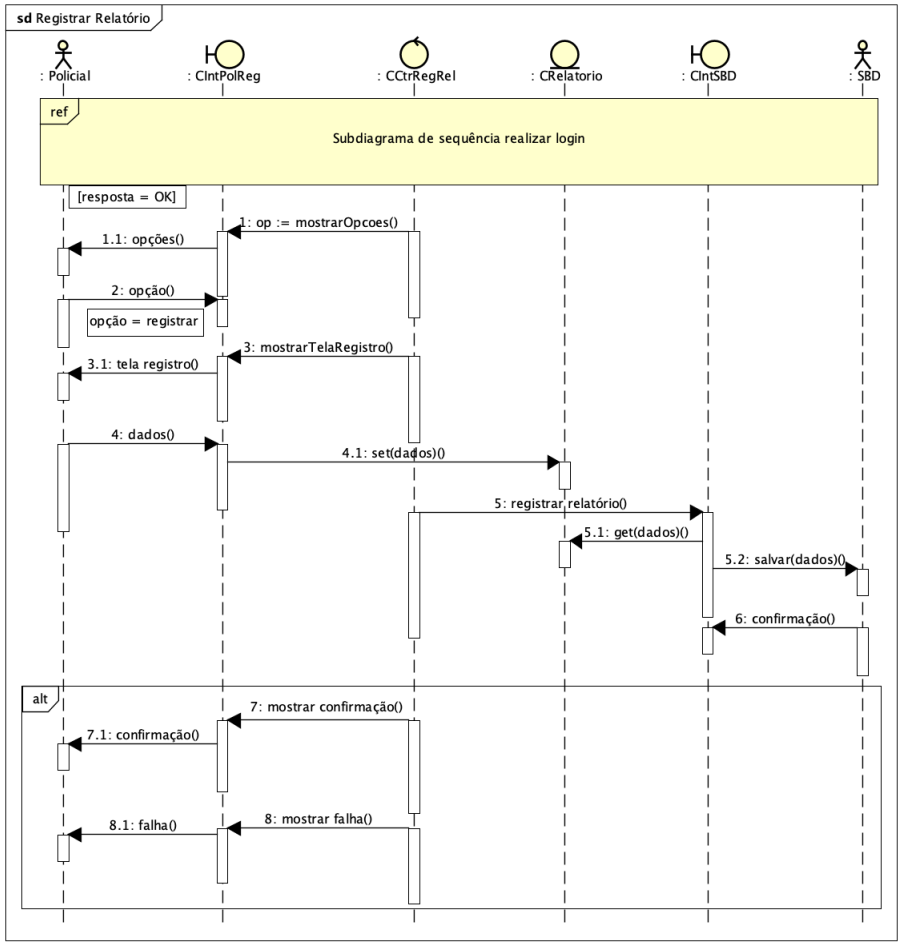

Figura 4.5: Registrar Relatório **(Autoria: Souza [25])**.

Para realizar o mapeamento do diagrama “Registrar Relatório” (figura 4.5), para a rede de Petri segura, seguiu-se os seguintes passos:

1. Para este diagrama, os lugares "Policial", "CIntPolReg", "CCtrlRegRel", "CRelatorio", "CIntSBD"e "SBD" foram também mapeados como lugares e os eventos entre eles foram mapeados por meio de transições;

2. Neste diagrama existe o fragmento de interação REF que faz referência ao diagrama de realizar login. Para mapear a referência, foi incluído um lugar com o nome do diagrama referenciado e uma transição "TRRef" que representa o fluxo do diagrama

de realizar login. No lugar do diagrama referenciado, foi inserida uma ficha pois é onde deve iniciar a sequência de eventos;

3. Após realizar o login, o controle "CCtrlRegRel" exibe as opções através de uma transição "mostrarOpcoes" para "CIntPolReg" e este elemento envia uma mensagem "1:1: opções" para o ator "Policial" através da transição "opcoes";

4. "Policial" envia então a mensagem "2: opção" para o lugar "CIntPolReg0" através da transição "opcao";

5. Na rede, a ficha está em "CIntPolReg0" mas a próxima mensagem é enviada a partir de ": CCtrlRegRel", então é inserido uma transição "TRLife" e após ela o lugar "CCtrlRegRel0". Este lugar contém uma transição "mostrarTelaRegistro", essa transição representa a mensagem "3: mostrar tela registro". A transição "mostrarTelaRegistro" possui uma ligação com o lugar "CIntPolReg1" o qual é o receptor da mensagem;

6. "CIntPolReg1" envia a mensagem "3.1: tela registro" através da transição "telaRegistro" para "Policial0";

7. Em seguida, para o restante do fluxo do diagrama de sequência, foram aplicadas as regras previamente estabelecidas, levando em consideração a sequência das operações;

8. É importante destacar o fragmento de interação ALT presente nesse diagrama, onde são apresentadas duas alternativas. Para este mapeamento, foi utilizado o conceito de conflito da rede de Petri, como mencionado anteriormente, inserindo duas transições de saída no lugar auxiliar "CCtrlRegRel2". A transição de "mostrarConfirmacao" e a transição "mostrarFalha";

9. Cada fluxo de uma alternativa foi mapeado com lugares representando os objetos do diagrama e transições representando a troca de mensagens entre eles. Os dois fluxos alternativos finalizam no mesmo lugar "Policial1" pois é onde finaliza no diagrama;

10. A rede inicia com 1 ficha inserida em "RealizarLogin" pois é o lugar na rede de Petri representando o diagrama referenciado, ele então insere uma ficha na transição "TRRef" tornando possível a mesma ser disparada. O disparo desta transição insere uma ficha em "CCtrlRegRel", sensibilizando a transição "mostrarOpcoes";

11. A transição "mostrarOpcoes", quando disparada, insere uma ficha em "CIntPolReg" que sensibiliza a transição "opcoes". E então, a sequência dos fatos são mapeados com lugares e transições até concluir no lugar "SBD0".

A figura 4.6 ilustra a rede de Petri mapeada a partir do diagrama de sequência da figura 4.5.

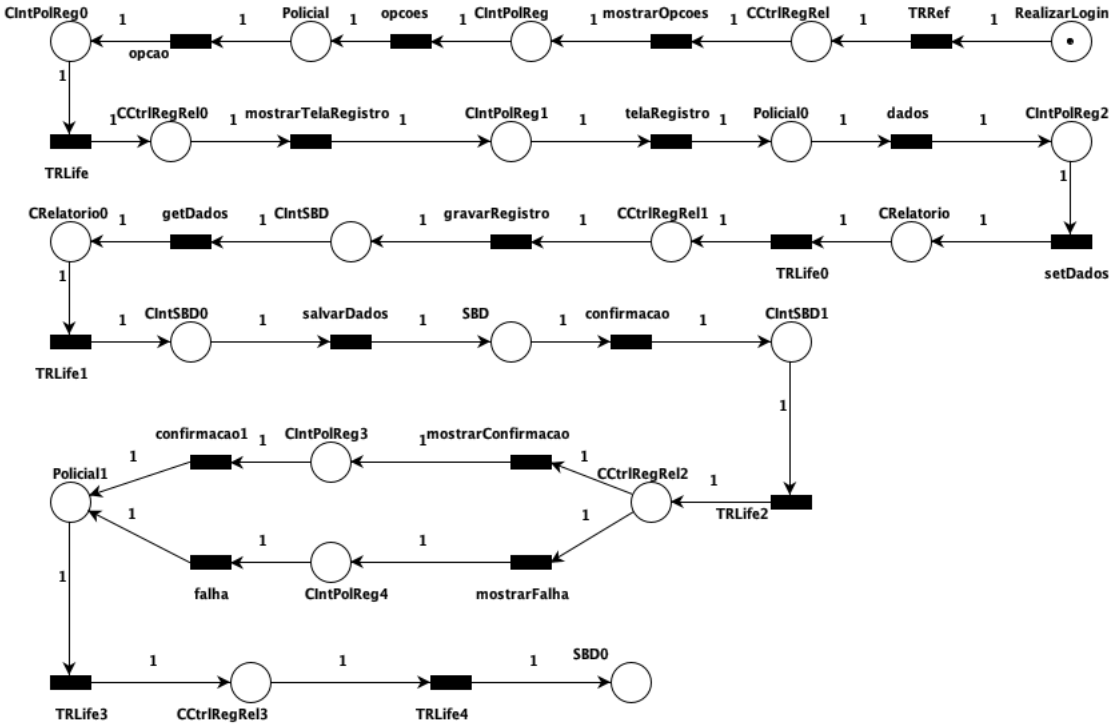

Figura 4.6: Rede de Petri Registrar Relatório.

4.2.3 Visualizar Relatório

Para o diagrama “Visualizar Relatório” (figura 4.7), o mapeamento foi realizado da seguinte forma:

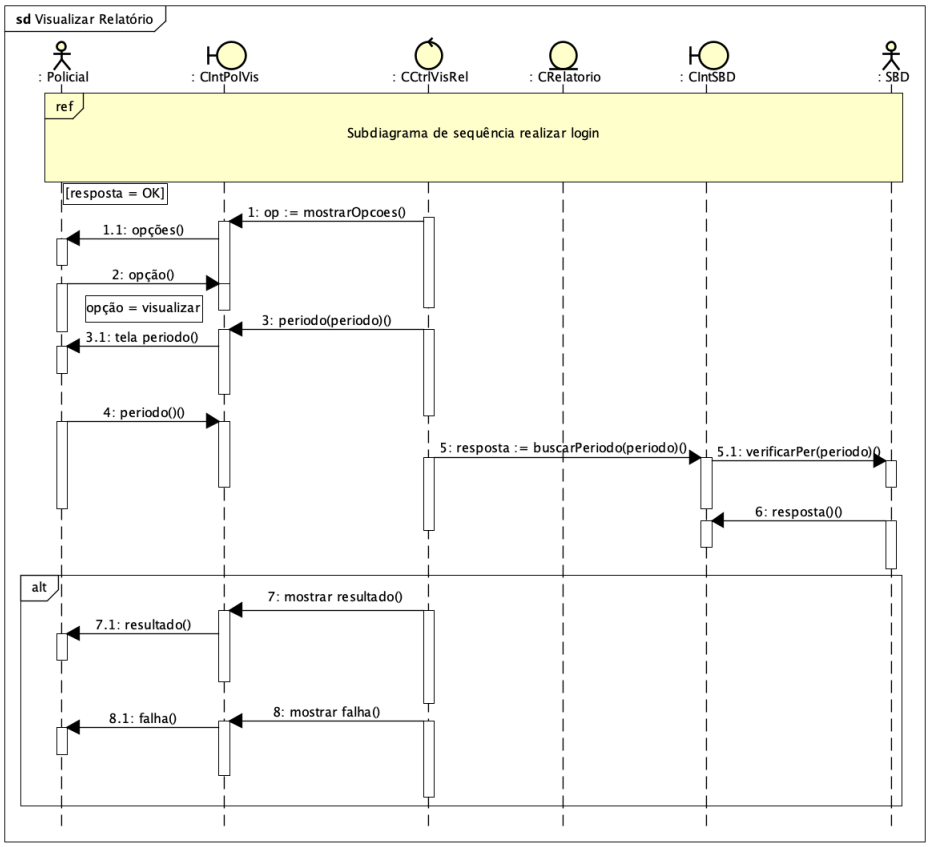

Figura 4.7: Visualizar Relatório **(Autoria: Souza [25])**.

1. Neste diagrama existe o fragmento de interação REF que faz referência ao diagrama de realizar login. Dessa maneira, foi incluído um lugar com o nome do diagrama referenciado (RealizarLogin) e uma transição "TRRef". No lugar do diagrama referenciado, foi inserido uma ficha pois é onde deve iniciar a sequência de eventos;

2. Após realizar o login, o controle "CCtrlVisRel" exibe as opções através da transição "mostrarOpcoes" para a fronteira "CIntPolVis" e este retorna as opções para o ator Policial através da transição "opcoes";

3. O ator "Policial" envia a opção escolhida para a fronteira "CIntPolVis0" através da transição "opcao". É inserido então uma transição "TRLife" para "CCtrlVisRel0";

4. O controle "CCtrlVisRel0" envia o período através da transição "periodo(periodo)";

5. Em seguida, "CIntPolVis1" envia a mensagem "3:1 tela periodo" para o ator "Policial0" através da transição "telaPeriodo";

6. Dessa forma, os lugares e as transições continuam a sequência de eventos respeitando a sequência do diagrama de sequência;

7. Este diagrama possui o fragmento de interação ALT que é representado com o conflito no lugar "CCtrlVisRel2". Este lugar possui duas transições de saída: "motrarResultado" e "mostrarFalha". A primeira transição representa o fluxo de mostrar resultado e a segunda o fluxo de mostrar falha. É importante destacar que estes dois fluxos finalizam no mesmo lugar "Policial1";

8. A rede inicia com 1 ficha inserida em "RealizarLogin", ele então insere uma ficha na transição "TRRef" tornando possível a mesma ser disparada. O disparo desta transição insere uma ficha em "CCtrlVisRel", sensibilizando a transição "mostrarOpcoes";

9. A transição "mostrarOpcoes", quando disparada, insere uma ficha em "CIntPolVis" que sensibiliza a transição "opcoes". A sequência dos fatos são mapeados com lugares e transições até concluir no lugar "SBD0".

A figura 4.8 ilustra a rede de Petri de visualizar relatório mapeada a partir do diagrama de sequência visualizar relatório exibido na figura 4.7.

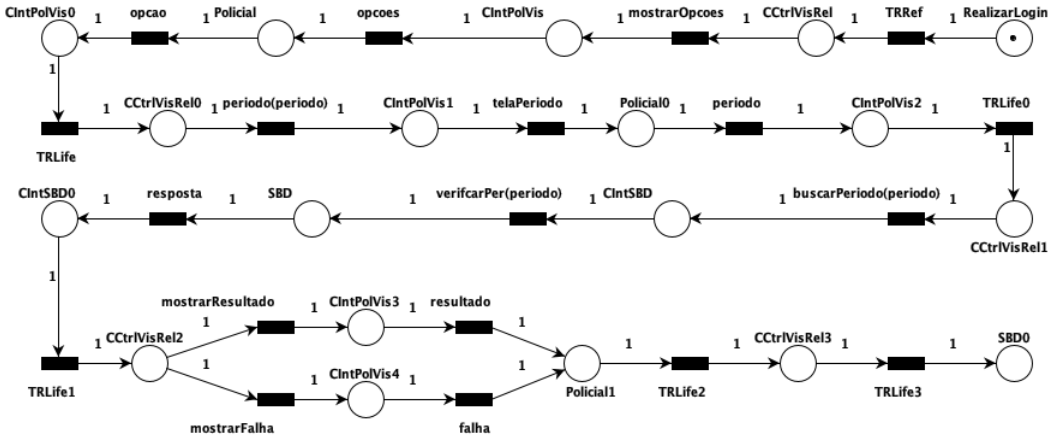

Figura 4.8: Rede de Petri Visualizar Relatório.

4.3 Análise das Redes de Petri Seguras

Nesta seção será descrito a análise das redes de Petri mapeadas na seção anterior. Para isso, utilizou-se o software PIPE a fim de gerar os grafos de alcançabilidade e utilizá-los como base nas análises. Para analisar as redes mapeadas, serão ilustrados novamente o diagrama base e a rede mapeada. A figura 4.9 a seguir, por exemplo, ilustra o diagrama de casos de uso do trabalho utilizado como base.

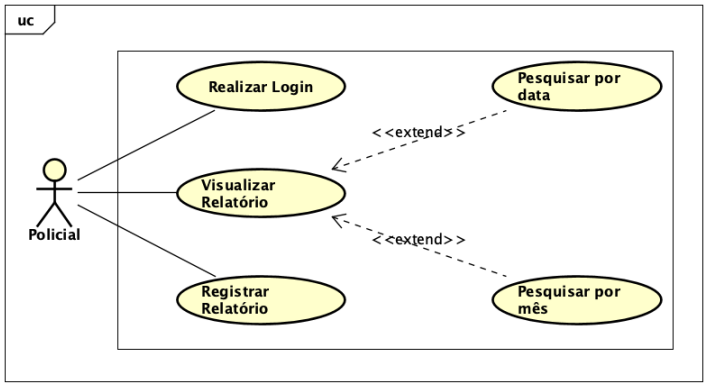

Figura 4.9: Diagrama de Casos de Uso **(Autoria: Souza [25])**.

A figura 4.10 ilustra a rede de Petri obtida a partir do diagrama de casos de uso ilustrado na figura 4.9.

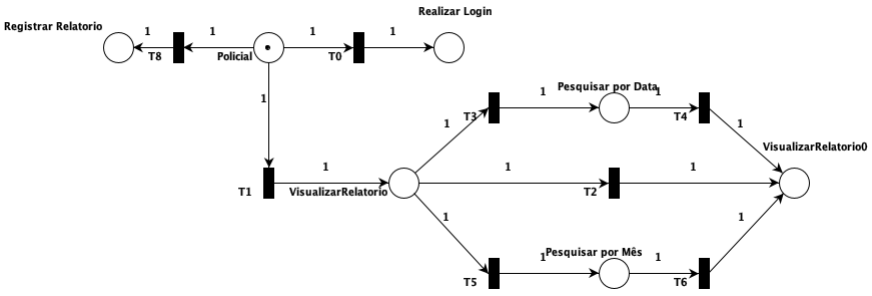

Figura 4.10: Rede de Petri Diagrama Caso de Uso.

Utilizando a rede da figura 4.10 foi gerado o grafo de alcançabilidade, através do PIPE, obtendo o grafo ilustrado na figura 4.11. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações

possíveis da rede através do disparo de uma determinada transição. A rede possui então uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

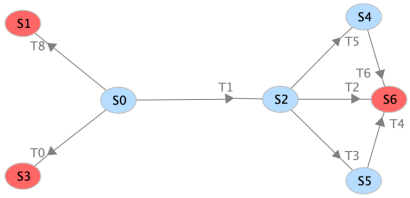

Figura 4.11: Grafo de alcançabilidade Caso de Uso.

Para o entendimento das análises, é necessário a compreensão do que cada estado do grafo representa. Eles representam lugares da rede como é detalhado a seguir:

- O estado S0 no grafo de alcançabilidade é composto pelo lugar “Policial” da rede de Petri da figura 4.10;
- O estado S1 é composto pelo lugar “Registrar relatório” da rede;
- O estado S2 é composto pelo lugar “VisualizarRelatorio”;
- O estado S3 é composto pelo lugar “Realizar Login” da rede;
- O estado S4 é composto pelo lugar “Pesquisar por mês” da rede;
- O estado S5 é composto pelo lugar “Pesquisar por Data” da rede;
- O estado S6 é composto pelo lugar “VisualizarRelatorio0” da rede.

A partir disso, define-se a marcação inicial da rede avaliando as transições habilitadas. Cada lugar que possui fichas que habilitam uma transição recebem 1 e os lugares que não possuem recebem 0.

A marcação inicial da rede então é M0 = [1, 0, 0, 0, 0, 0, 0], considerando a ordem dos lugares como "Policial", "Realizar Login", "Registrar Relatório", "VisualizarRelatorio", "Pesquisar por Data", "Pesquisar por Mês"e "VisualizarRelatorio0".

A cada transição disparada, obtém-se uma nova marcação pois, disparando uma transição é removida as fichas de um lugar e inseridas em outro lugar de acordo com o peso

definido no arco. Diante disso, a sequência de disparos da rede ilustrada na figura 4.2 é dada pelo seguinte:

M0 [T0> M1	M0 = [1, 0, 0, 0, 0, 0, 0]
M0 [T1> M2	M1 = [0, 1, 0, 0, 0, 0, 0]
M0 [T8> M3	M2 = [0, 0, 0, 1, 0, 0, 0]
M2 [T2> M4	M3 = [0, 0, 1, 0, 0, 0, 0]
M2 [T3> M5	M4 = [0, 0, 0, 0, 0, 0, 1]
M2 [T5> M6	M5 = [0, 0, 0, 0, 1, 0, 0]
M5 [T4> M4	M6 = [0, 0, 0, 0, 0, 1, 0]
M6 [T6> M4	

Com base nas marcações e sequência de disparos listados acima, é possível verificar que o disparo da transição T0 alcança o estado S3 (Realizar login), o disparo da transição T1 alcança o estado S2 (Visualizar relatório). A partir de T3 é possível visualizar o relatório pesquisando por data e a partir de T5 é possível visualizar o relatório pesquisando por mês. O disparo da transição T8 alcança o estado S1 (Registrar relatório). Os disparos das transições presentes em "Policial" representam a invocação do caso de uso pelo ator.

Ao analisar o diagrama de casos de uso, pode-se constatar que o ator "Policial" possui as seguintes funcionalidades: realizar login, registrar relatório e visualizar relatório. No caso da visualização do relatório, é possível fazê-lo por mês ou por data. Esses cenários também foram analisados na rede de Petri, a partir da qual verificou-se a alcançabilidade de todos esses cenários no grafo de alcançabilidade, mantendo-se as mesmas condições presentes no diagrama.

Para a próxima análise, será seguido os mesmos passos descritos acima. A figura 4.12 a seguir, por exemplo, ilustra o diagrama de sequência de realizar login do trabalho utilizado como base.

A figura 4.13 ilustra a rede de Petri do diagrama de sequência de realizar login.

Utilizando a rede da figura 4.13 foi gerado o grafo de alcançabilidade, através do

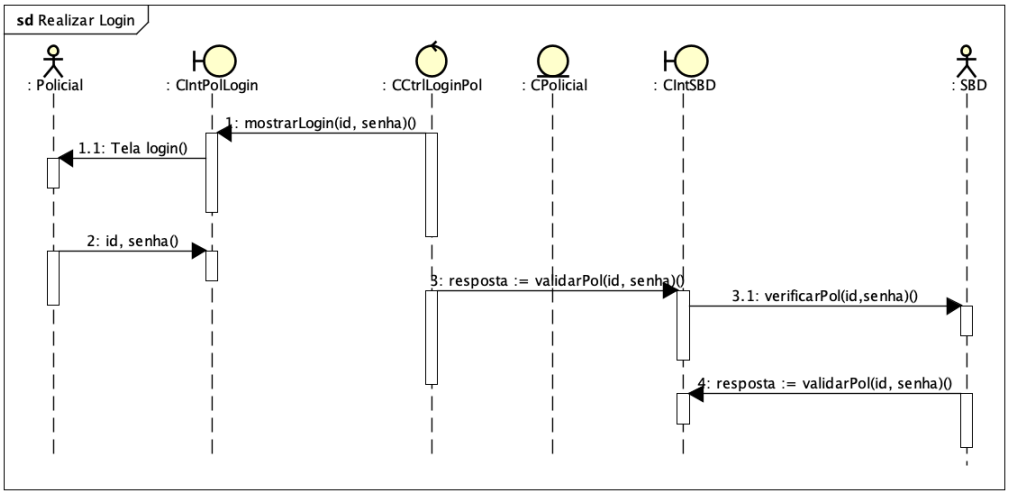

Figura 4.12: Realizar login **(Autoria: Souza [25])**.

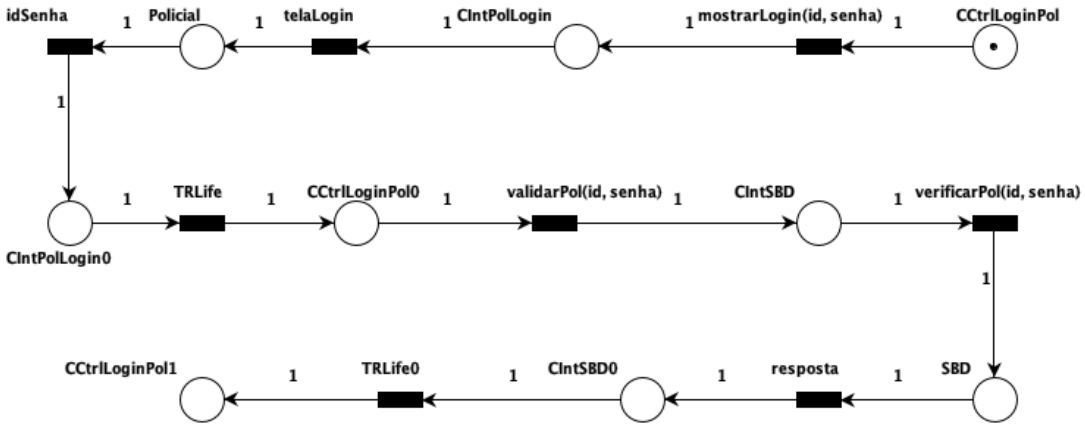

Figura 4.13: Rede de Petri Realizar Login.

PIPE, obtendo o grafo ilustrado na figura 4.14. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações possíveis da rede através do disparo de uma determinada transição. A rede possui então uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para o entendimento das análises, é necessário a compreensão do que cada estado do grafo representa. Eles representam lugares da rede como é detalhado a seguir:

- O estado S0 do grafo de alcançabilidade é composto por "CCtrlLoginPol" da rede de Petri;

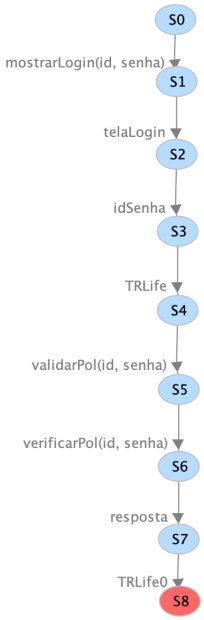

Figura 4.14: Grafo de alcançabilidade rede Realizar Login.

- O estado S1 do grafo de alcançabilidade é composto por "CIntPolLogin";
- O estado S2 do grafo de alcançabilidade é composto por "Policial";
- O estado S3 do grafo de alcançabilidade é composto por "CIntPolLogin0";
- O estado S4 do grafo de alcançabilidade é composto por "CCtrlLoginPol0";
- O estado S5 do grafo de alcançabilidade é composto por "CIntSBD";
- O estado S6 do grafo de alcançabilidade é composto por "SBD";
- O estado S7 do grafo de alcançabilidade é composto por "CIntSBD0";
- O estado S8 do grafo de alcançabilidade é composto por "CCtrlLoginPol1".

A partir disso, define-se a marcação inicial da rede avaliando as transições habilitadas. A marcação inicial da rede é dada então por: M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0], considerando a ordem dos lugares como listado acima. A sequência de disparos é dada por:

M0 [mostrarLogin> M1	M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0]
M1 [telaLogin> M2	M1 = [0, 1, 0, 0, 0, 0, 0, 0, 0]
M2 [idSenha> M3	M2 = [0, 0, 1, 0, 0, 0, 0, 0, 0]
M3 [TRLife> M4	M3 = [0, 0, 0, 1, 0, 0, 0, 0, 0]
M4 [validarPol> M5	M4 = [0, 0, 0, 0, 1, 0, 0, 0, 0]
M5 [verificarPol> M6	M5 = [0, 0, 0, 0, 0, 1, 0, 0, 0]
M6 [resposta> M7	M6 = [0, 0, 0, 0, 0, 0, 1, 0, 0]
M7 [TRLife0> M8	M7 = [0, 0, 0, 0, 0, 0, 0, 1, 0]
	M8 = [0, 0, 0, 0, 0, 0, 0, 0, 1]

Com base nas marcações e sequência de disparos listados acima é possível verificar então que, a marcação inicial sensibiliza a transição “mostrarLogin” e o disparo desta transição leva para a marcação M2 sensibilizando a transição “telaLogin”. Após isso, a transição “idSenha” é habilitada levando para a marcação M3 quando disparada. M3 sensibiliza a transição “TRLife” que é a representação da linha de vida e o disparo desta transição leva para a marcação M4. A marcação M4 sensibiliza a transição “validarPol” que ao ser disparada leva para a marcação M5 que sensibiliza a transição “verificarPol” e quando disparada leva para a marcação M6. M6 sensibiliza a transição “resposta” e quando disparada leva para a marcação M7. M7 sensibiliza a transição "TRLife0" e quando disparada leva para a marcação final, M8.

O diagrama de sequência apresentado na figura 4.12 ilustra a sequência necessária para realizar o login no sistema. Para isso, é enviado o método "mostrarLogin(id, senha)" para o objeto CIntPolLogin, que exibe a tela de login para o policial. Em seguida, o policial insere o id e a senha, que são validados por CIntSBD e verificados por SBD. A resposta da validação e verificação é enviada por SBD.

O grafo de alcançabilidade inicia em S0 (CCtrlLoginPol) e alcança o estado S1 (CIntPolLogin) ao disparar a transição "mostrarLogin(id, senha)". O estado S1, ao disparar a

transição "telaLogin", alcança o estado S2 (Policial), e S2, ao disparar a transição "idSenha", alcança o estado S3 (CIntPolLogin0). S3, ao disparar a transição "TRLife", alcança o estado S4 (CCtrlLoginPol0), e S4, ao disparar a transição "validarPol(id, senha)", alcança o estado S5 (CIntSBD). S5, ao disparar a transição "verificarPol(id, senha)", alcança o estado S6 (SBD), e S6, ao disparar a transição "resposta", alcança o estado S7 (CIntSBD0). Por fim, S7, ao disparar a transição "TRLife0", alcança o estado S8 (CCtrlLoginPol1).

Assim, é possível destacar que a sequência de eventos descritos no diagrama de sequência coincide com a sequência de eventos do grafo de alcançabilidade. Todos os estados no grafo são alcançáveis, o que significa que é possível atingir todas as marcações permitidas através do disparo de uma determinada transição.

Para a análise da rede de registrar relatório, será ilustrado novamente o diagrama de sequência e a rede de Petri de "Registrar relatório". A figura 4.15 ilustra o diagrama de sequência de registrar relatório do trabalho utilizado como base.

A figura 4.16 ilustra a rede de Petri segura do diagrama de sequência de registrar relatório.

A partir da rede ilustrada na figura 4.16 foi gerado o grafo de alcançabilidade, através do PIPE, obtendo o grafo ilustrado na figura 4.17. Avaliando o grafo, é possível perceber que todos os estado do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações possíveis da rede através do disparo de uma determinada transição. A rede possui então uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para o entendimento das análises, é necessário a compreensão do que cada estado do grafo representa. Eles representam lugares da rede como é detalhado a seguir:

- S0 por "RealizarLogin";
- S1 por "CCtrlRegRel";
- S2 por "CIntPolReg";
- S3 por "Policial";

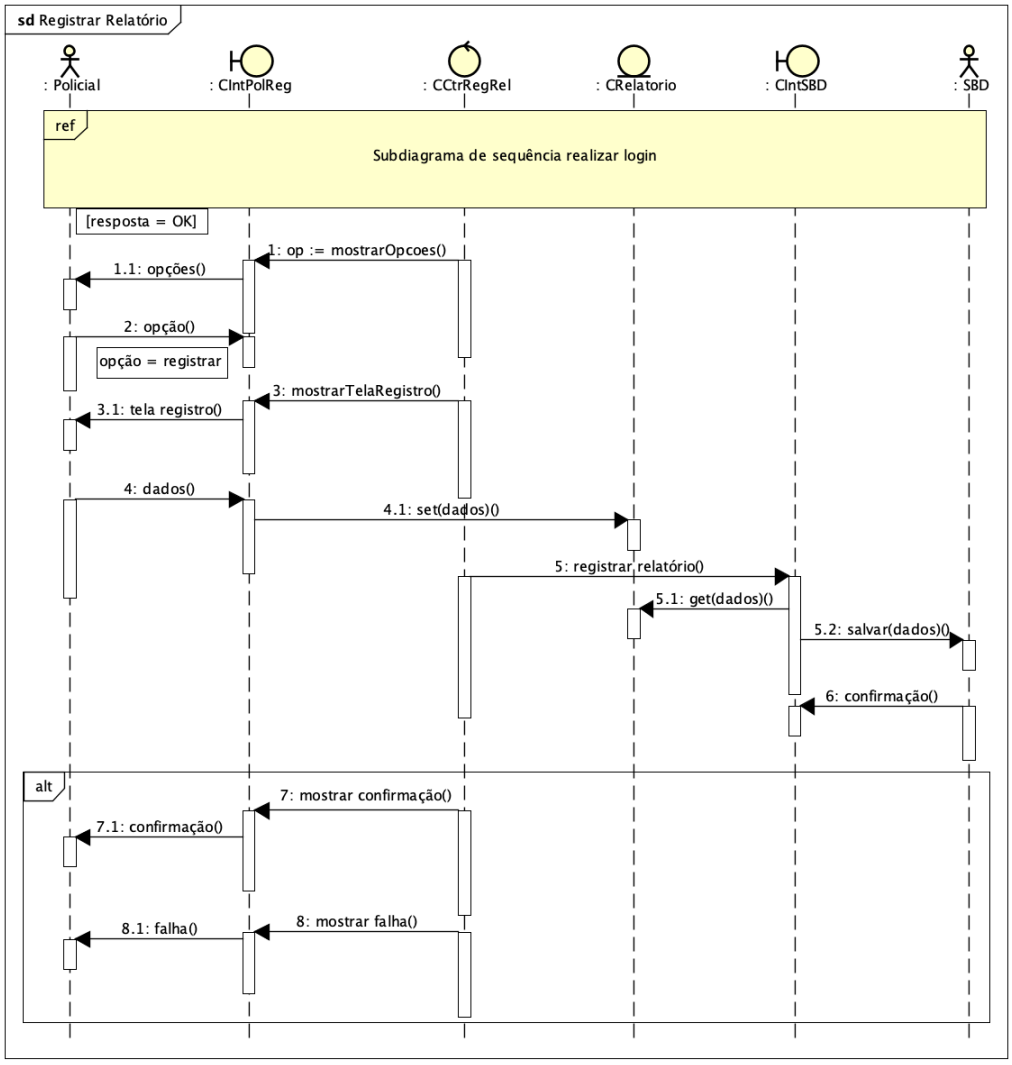

Figura 4.15: Diagrama de sequência registrar relatório **(Autoria: Souza [25])**.

- S4 por "CIntPolReg0";
- S5 por "CCtrlRegRel0";
- S6 por "CIntPolReg1";
- S7 por "Policial0";
- S8 por "CIntPolReg2";
- S9 por "CRelatorio";

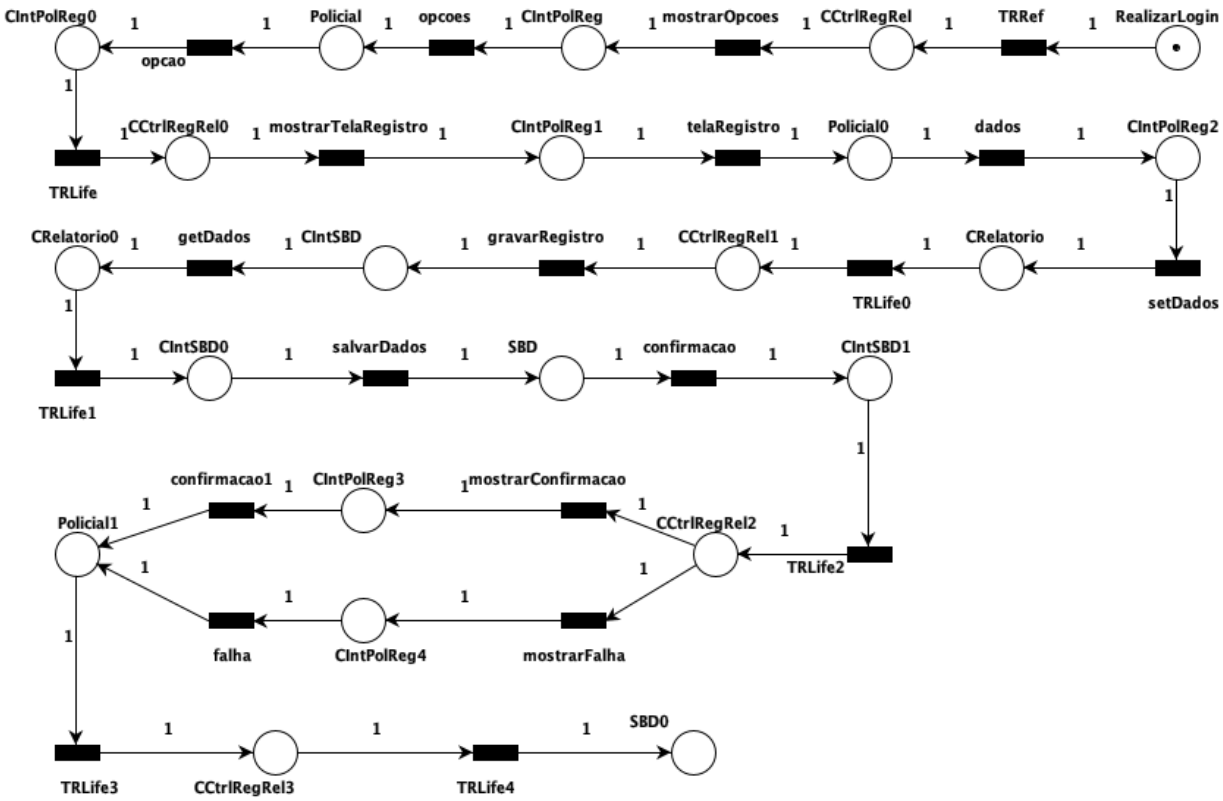

Figura 4.16: Rede de Petri registrar relatório.

- S10 por "CCtrlRegRel1";
- S11 por "CIntSBD";
- S12 por "CRelatorio0";
- S13 por "CIntSBD0";
- S14 por "SBD";
- S15 por "CIntSBD1";
- S16 por "CCtrlRegRel2";
- S17 por "CIntPolReg4";
- S18 por "CIntPolReg3";

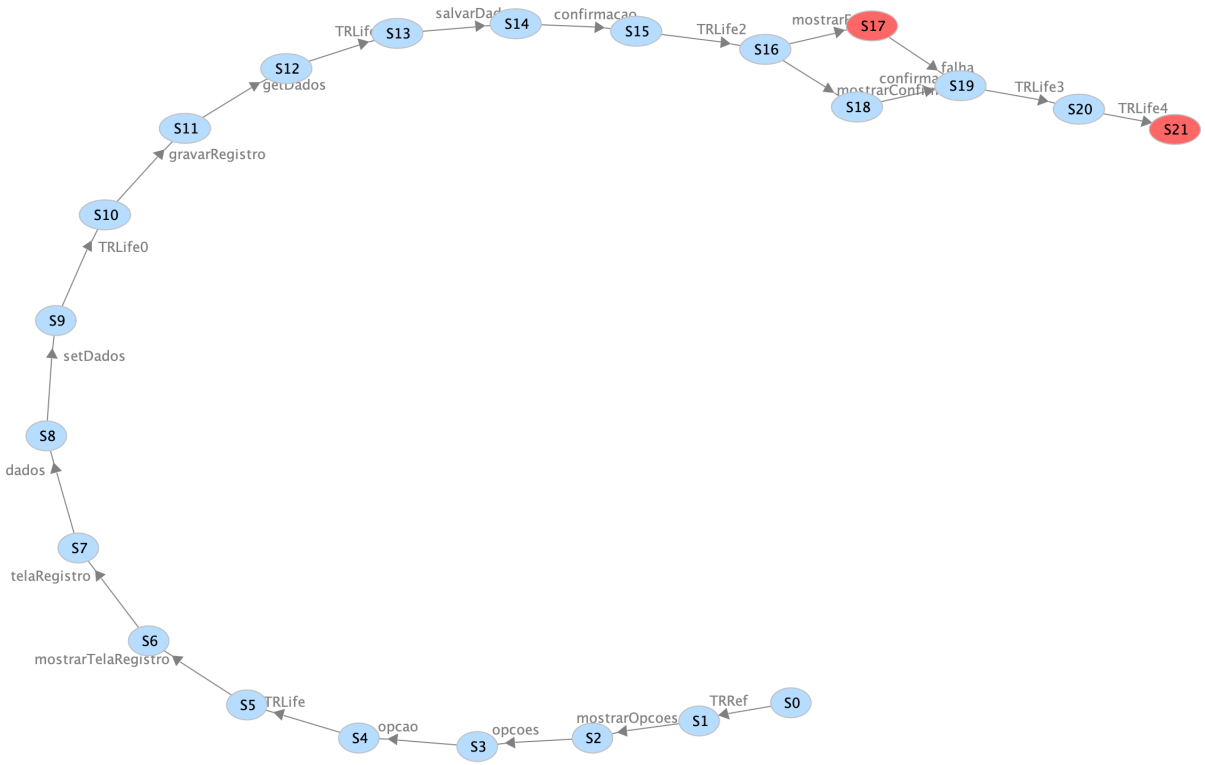

Figura 4.17: Grafo de alcançabilidade rede registrar relatório.

- S19 por "Policial1";
- S20 por "CCtrlRegRel3";
- S21 por "SBD0".

A partir disso, define-se a marcação inicial da rede avaliando as transições habilitadas. A marcação inicial da rede é dada então por: M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], considerando a ordem dos lugares como listado acima. A sequência de disparos é dada por:

- M0 [TRRef> M1;
- M1 [mostrarOpcoes> M2;
- M2 [opcoes> M3;
- M3 [opcao> M4;

- M4 [TRLife> M5;
- M5 [mostrarTelaRegistro> M6;
- M6 [TelaRegistro> M7;
- M7 [dados>M8;
- M8 [setDados> M9;
- M9 [TRLife0> M10;
- M10 [gravarRegistro> M11;
- M11 [getDados> M12;
- M12 [TRLife1> M13;
- M13 [salvarDados> M14;
- M14 [confirmacao> M15;
- M15 [TRLife2> M16;
- M16 [mostrarConfirmacao> M18;
- M16 [mostrarFalha> M17;
- M17 [falha> M19;
- M18 [confirmacao1> M19;
- M19 [TRLife3> M20;
- M20 [TRLife4> M21.

Onde,

- M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

- M1 = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M2 = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M3 = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M4 = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M5 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M6 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M7 = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M8 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M10 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M11 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M12 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0];
- M13 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0];
- M14 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0];
- M15 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0];
- M16 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
- M17 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0];
- M18 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0];
- M19 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0];
- M20 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];

- M21 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];

Com base nas marcações e sequência de disparos listados acima é possível verificar então que, a marcação inicial da rede sensibiliza a transição “TRRef” para realizar o login e o disparo desta transição leva para a marcação M2 sensibilizando a transição “mostrarOpções”. Após isso, a transição “opções” é habilitada levando para a marcação M3 quando disparada. M3 sensibiliza a transição “opcao” e o disparo dessa transição leva para a marcação M4. A marcação M4 sensibiliza a transição “TRLife” que é a representação da linha de vida e que ao ser disparada leva para a marcação M5 que sensibiliza a transição “mostrarTelaRegistro” e quando disparada leva para a marcação M6. M6 sensibiliza a transição “telaRegistro” e quando disparada leva para a marcação M7. M7 sensibiliza a transição "dados" e quando disparada leva para a marcação M8.

A marcação M8, sensibiliza a transição "setDados" levando para a marcação M9. M9 sensibiliza a transição "TRLIfe0" e quando disparada leva para a marcação M10. M10 sensibiliza a transição "gravarRegistro" indo para M11 e M11 sensibiliza a transição "getDados"indo para M12. M12 sensibiliza a transição "TRLife1" levando para a marcação M13. M13 sensibiliza "salvarDados" e quando disparada, leva para a marcação M14. M14 sensibiliza "salvarDados" indo para M15 que sensibiliza "confirmacao" levando para M16. A marcação M16 sensibiliza a transição "TRLife2"onde possui um conflito. O primeiro caminho é mostrar a falha a partir da marcação M17 com a transição "mostrarFalha" onde é levado para a marcação M19. A marcação M18 possui a transição "mostrarConfirmacao" que também leva para M19. M19 sensibiliza a transição "TrLife3" levando para marcação M20 e por fim, M20 sensibiliza a transição "TRLife4"que leva para a marcação final, M21.

O diagrama de sequência da figura 4.15 ilustra a sequência necessária para registrar o relatório no sistema. Analisando o disparo das transições que levam da marcação inicial até a final é possível verificar que a sequência de eventos necessária para registrar o relatório é atendida conforme descrito no diagrama de sequência. Ou seja, o grafo descreve todos os cenários descritos no diagrama de sequência para o registro do relatório e descreve todos os estados alcançáveis a partir de determinado momento descrito no diagrama. A análise do mapeamento do diagrama de sequência de visualizar relatório é descrito no

anexo A.1.

4.4 Mapeamento Diagrama de Casos de Uso para Rede de Petri Colorida

Para o mapeamento do diagrama de casos de uso e do diagrama de sequência em uma rede colorida, utilizou-se como base o artigo dos autores Ribeiro, Fernandes, Tjell et al. [7]. Neste artigo, os autores propuseram uma abordagem para traduzir diagramas de casos de uso e diagramas de sequência em um modelo de rede de Petri colorida equivalente. A seguir será descrito brevemente as regras criadas pelos autores e os mapeamentos realizados de acordo com as regras.

Para o mapeamento dos diagrama de casos de uso, os autores propões mapear cada caso de uso presente no diagrama de casos de uso para uma transição de substituição. Cada transição de substituição representa o fluxo do seu respectivo diagrama de sequência. Além disso, os autores consideram apenas os casos de uso primários.

É importante destacar que, as redes de Petri coloridas permitem a criação de sub-redes através da transição de substituição. Cada rede pode conter várias sub-redes, que por sua vez podem ser aninhadas em diferentes níveis de detalhamento. Para estabelecer a conexão entre a transição de substituição e a sub-rede correspondente, são utilizados lugares-porta. Esses lugares representam a entrada e saída da sub-rede, delimitando o início e o fim da mesma. Após o término da sub-rede, o fluxo retorna à rede principal [26] [27].

Dessa forma, cada caso de uso da figura 4.9 foi traduzido como uma transição de substituição. A figura 4.18 ilustra a rede de Petri mapeada. Os autores propõem iniciar a rede com um lugar do tipo de uma lista de casos de uso possíveis executar e este lugar é conectado a uma transição que carrega uma função para remover da lista a opção escolhida. Após isso, tem-se as transições de substituição. Para o diagrama de casos de uso

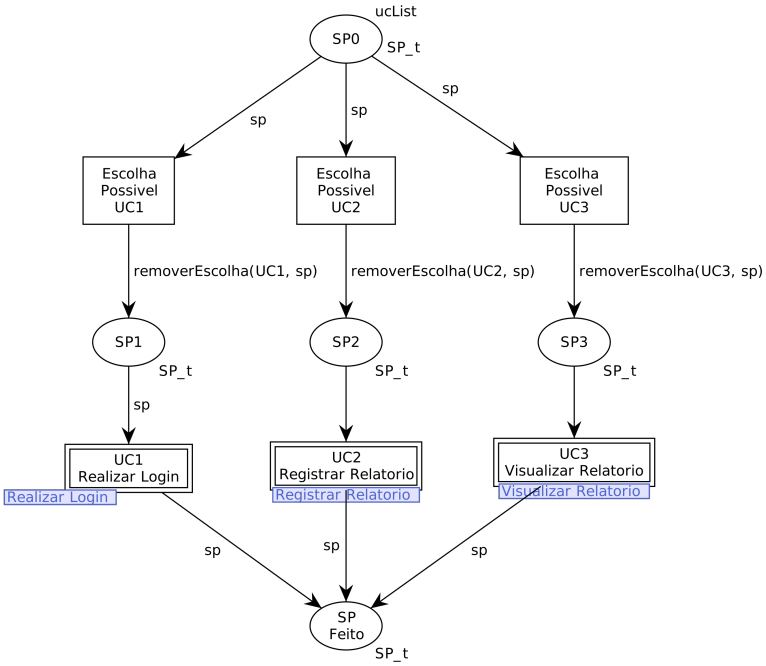

Figura 4.18: Rede de Petri colorida diagrama de casos de uso.

da figura 4.9 os casos de uso primário são: "Realizar Login", "Registrar Relatório" e "Visualizar Relatório". Devido a isso, tem-se na rede colorida três transições de substituição, cada uma representa respectivamente um caso de uso. O mapeamento de cada transição de substituição pode ser visto no anexo A.2. A seção a seguir explica melhor como foi feito o mapeamento dos diagramas de sequência para a rede de Petri colorida, a diferença entre o mapeamento dos diagramas de sequência para uma rede de Petri colorida e esses mesmos diagramas correspondentes às transições de substituição é que no diagrama de casos de uso os diagramas carregam a lista de casos de uso até o final.

O conjunto de cores, variáveis e funções da rede colorida do diagrama de casos de uso é descrito no apêndice B.

4.5 Mapeamento Diagrama de Sequência para Rede de Petri Colorida

Para mapear diagramas de sequência em redes de Petri coloridas, Ribeiro, Fernandes, Tjell et al. [7] propuseram um conjunto de regras que permitem a conversão de um diagrama de sequência em sua equivalente rede de Petri colorida. Os mesmos autores descreveram com mais detalhes as regras de mapeamento em [28]. Eles propuseram uma conversão baseada em uma semântica para diagramas de sequência que considera uma relação de ordem entre as mensagens, onde a emissão de uma determinada mensagem requer a recepção da mensagem anterior, se esta existir. É importante destacar que os autores assumiram que os diagramas de sequência possuem mensagens síncronas.

Na abordagem proposta pelos autores, cada mensagem em um diagrama de sequência é representada por uma transição na rede de Petri colorida. Além disso, a abordagem inclui locais na rede de Petri colorida que contêm os valores atuais dos objetos presentes no diagrama de sequência. Cada mensagem em um diagrama de sequência carrega uma função que pode alterar o valor do objeto receptor, de modo que, sempre que uma transição é disparada, os valores dos objetos podem mudar. Ao lado de cada transição na rede de Petri colorida, há um local que contém uma representação do valor atual do objeto no destino da mensagem.

4.5.1 Realizar Login

Para o diagrama de sequência de "Realizar Login" ilustrado na figura 4.4, criou-se os respectivos lugares de "Policial", "CIntPolLogin", "CCtrlLoginPol", "CIntSBD" e "SBD" para representar os objetos presentes no diagrama de sequência em uma rede de Petri colorida. Em seguida, foram inseridas as transições para representar as trocas de mensagens presentes no diagrama como, por exemplo,"1: mostrarLogin() 1.1: Tela login()", "2: id, senha()", e as demais mensagens presentes no diagrama. A figura 4.19 ilustra a rede de Petri colorida do diagrama de sequência de "Realizar Login". Entre as transições existem

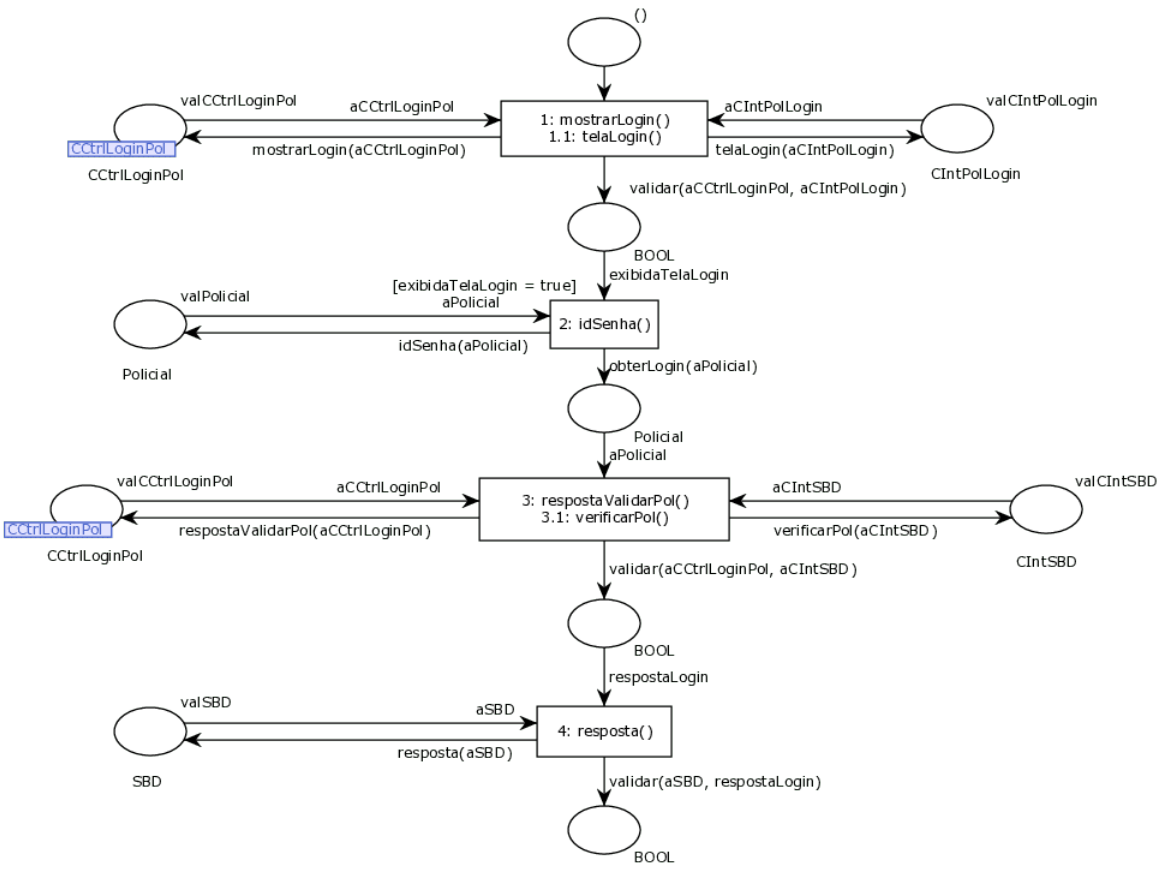

Figura 4.19: Rede de Petri colorida realizar login.

os lugares que controlam a rede, ou seja, a ficha presente nesse lugar habilita ou não uma transição, seguindo as regras propostas pelos autores. O mapeamento de um diagrama de sequência para uma rede de Petri requer uma compreensão precisa das ações e condições envolvidas no diagrama. No caso específico do mapeamento do diagrama de sequência de realizar login, a primeira transição da rede é acionada para exibir a tela de login. A transição de inserir os dados do login só é exibida se a tela tiver sido exibida corretamente e, para fazer essa verificação utiliza-se a função "validar()" com os parâmetros "aCCtrlLoginPol"e "aCIntPolLogin". Caso essa função retorne um valor *true*, a próxima transição pode ser disparada. A próxima transição é a inserção dos dados de id e senha. Inseridos o id e senha, a próxima transição irá verificar e validar os dados. Quando esta transição é disparada, o resultado é inserido no próximo lugar da rede, contendo uma ficha com o

valor *booleano*. Somente após essa verificação, a próxima transição pode ser disparada. A última transição é o resultado do login, do tipo *booleano*. Caso o último lugar da rede tenha 1 ficha *true*, significa que o login foi realizado. Caso contrário, significa que o login não foi realizado.

4.5.2 Registrar Relatório

Para o diagrama de sequência de "Registrar Relatório" ilustrado na figura 4.5, utilizou-se as mesmas regras de tradução obtendo a rede ilustrada na figura 4.20 a seguir.

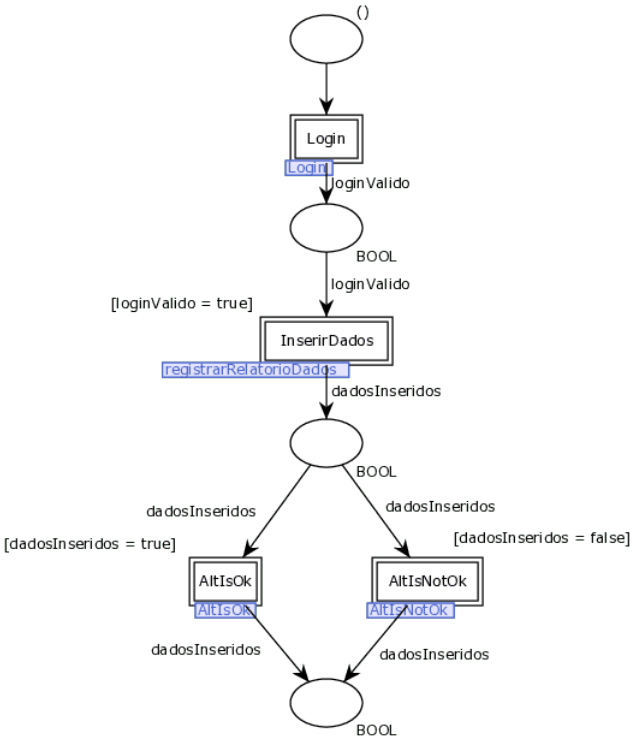

Figura 4.20: Rede de Petri colorida registrar relatório.

Para esse mapeamento, inseriu-se transições de substituições para uma melhor leitura da rede. A primeira transição corresponde ao "Login" que é um operador de alto nível no diagrama de sequência, o operador de referência. A rede obtida é similar à rede apresentada anteriormente na figura 4.19, o que diferencia é que como ela é uma sub-página, ela possui um lugar de entrada e um lugar de saída como ilustrado na figura

4.21.

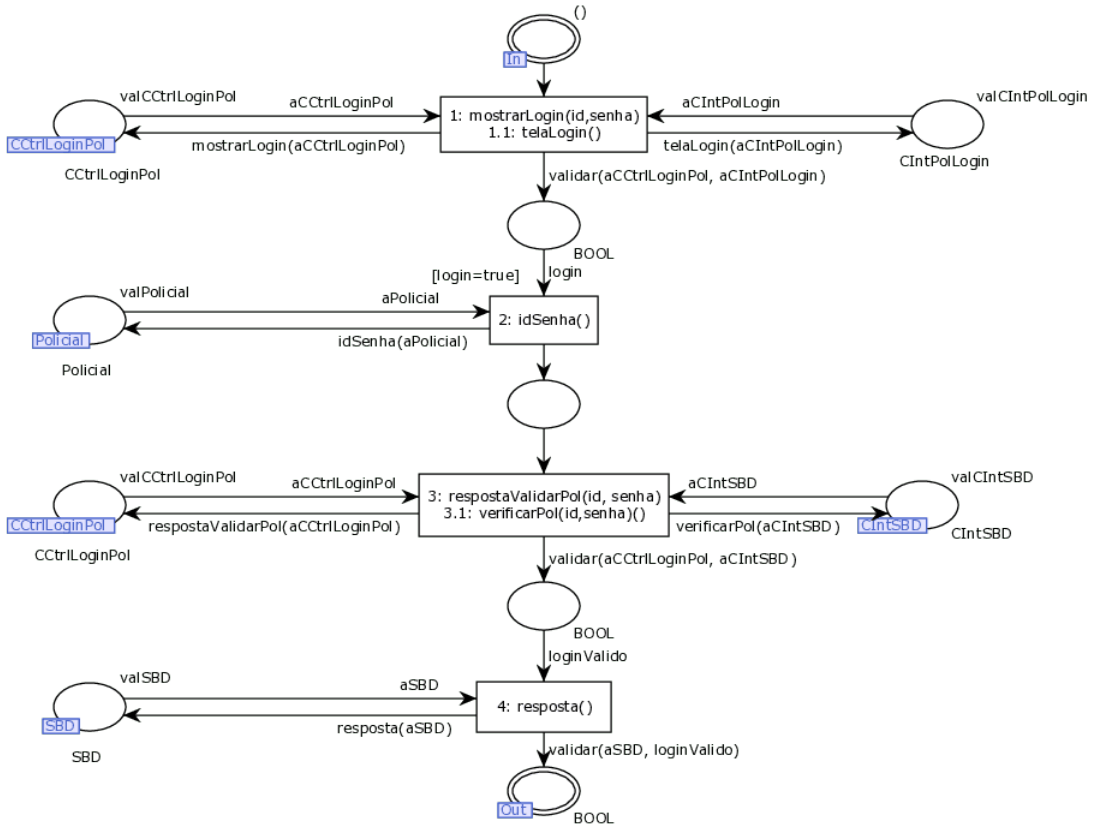

Figura 4.21: Rede de Petri colorida: transição de substituição "Login".

A segunda transição de substituição na rede de Petri é a de "Inserir Dados", que representa o fluxo de inserção dos dados. Para mapear essa transição, foram utilizadas as mesmas regras definidas por Ribeiro e Fernandes [28], inserindo lugares que controlam a rede e lugares que representam os objetos do diagrama de sequência. Cada lugar da rede contém uma ou mais fichas, representando o estado atual da transição. Quando a transição "Inserir Dados" é disparada, a função associada a cada objeto da interação é acionada, alterando seu valor na rede. Dessa forma, é possível simular o fluxo de dados representado pelo diagrama de sequência. A Figura 4.22 ilustra a rede de Petri resultante desse mapeamento.

É importante destacar que o diagrama de "Registrar Relatório" possui o operador ALT, apresentando duas alternativas para seguir a sequência dos eventos. Para o mapeamento,

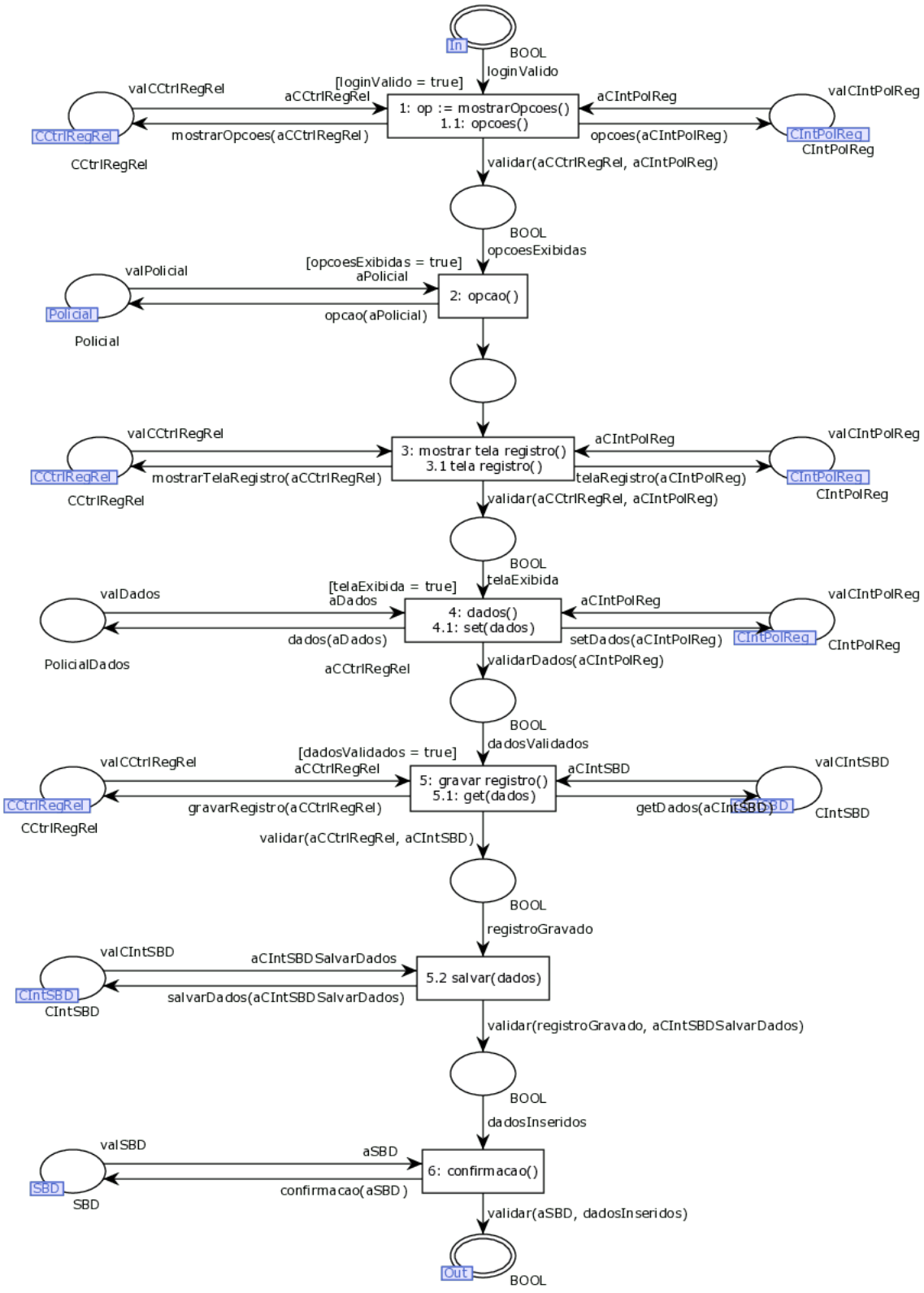

Figura 4.22: Rede de Petri colorida: transição de substituição "InserirDados".

assim como descrito pelos autores, inseriu-se duas transições onde só é possível a execução de apenas uma delas. Essas transições possuem condições, a transição de confirmação possui a condição dos dados inseridos ser igual a *true* e a transição de falha possui como condição os dados inseridos serem iguais a *false*. O fluxo de cada uma dessas transições também foi inserido em uma transição de substituição.

Para o fluxo de confirmação, o mapeamento foi realizado como ilustrado na figura 4.23. Para entrar nessa transição, é necessário ter 1 ficha do tipo *true*, o que determina que os dados foram inseridos e será apresentado a confirmação.

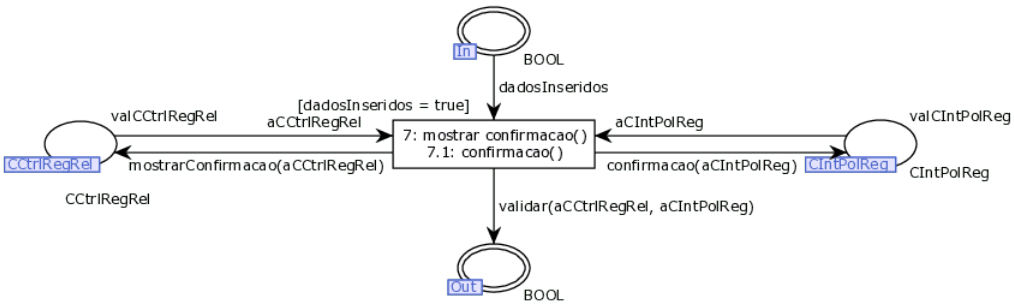

Figura 4.23: Rede de Petri colorida: transição de substituição "AltIsOk".

O fluxo de falha foi mapeado seguindo o de confirmação, porém para esse fluxo é necessário que a ficha seja do tipo *false*. Esse mapeamento é ilustrado na figura 4.24.

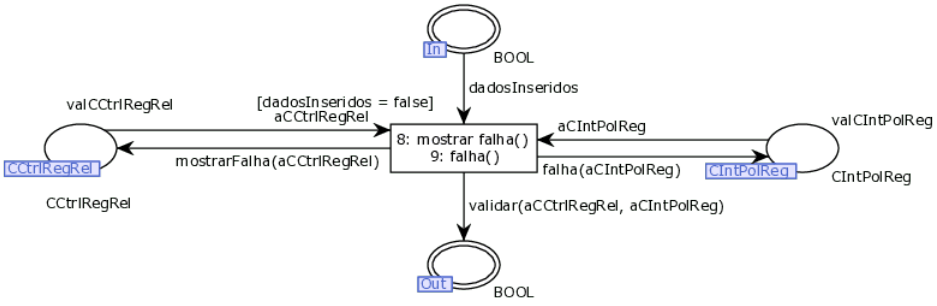

Figura 4.24: Rede de Petri colorida: transição de substituição "AltIsNotOk".

4.5.3 Visualizar Relatório

Para o diagrama de sequência de "Visualizar Relatório" ilustrado na figura 4.7 , utilizou-se também as regras dos autores Ribeiro, Fernandes, Tjell et al. [7], obtendo a rede de Petri colorida ilustrada na figura 4.25.

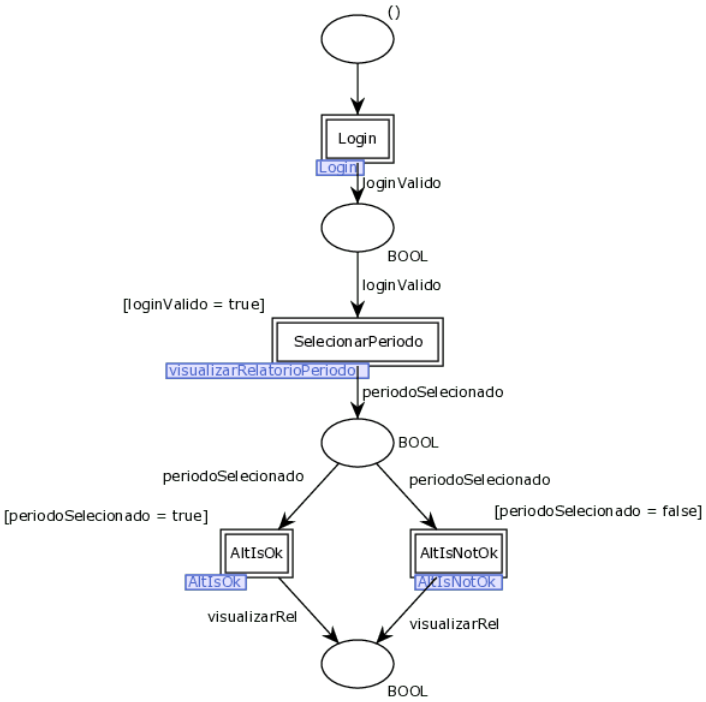

Figura 4.25: Rede de Petri colorida visualizar relatório.

Este mapeamento também foi realizado utilizando as transições de substituição para a melhor leitura da rede. A primeira transição corresponde ao "Login" que é um operador de alto nível no diagrama de sequência, o operador de referência. A rede obtida foi a mesma apresentada na figura 4.21.

A segunda transição de substituição na rede de Petri é a de "Selecionar Período", que representa o fluxo de seleção de período para a visualização do relatório. Para mapear o fluxo dessa transição, utilizou-se também os lugares que controlam a rede e os lugares que representam os objetos do diagrama de sequência. Cada lugar da rede contém uma ou mais fichas, representando o estado atual da transição. A rede possui funções que alteram os valores dos objetos possibilitando simular o fluxo dos dados representados pelo

diagrama de sequência. A 4.26 ilustra a rede de Petri resultante desse mapeamento.

É importante destacar que o diagrama de "Visualizar Relatório" possui o operador ALT, apresentando duas alternativas para seguir a sequência dos eventos. Para o mapeamento, assim como descrito pelos autores, inseriu-se duas transições onde só é possível a execução de apenas uma delas. O fluxo de cada uma dessas transições também foi inserido em uma transição de substituição.

Para o fluxo de resultado, o mapeamento foi realizado como ilustrado na figura 4.27. Esta transição possui como guarda o valor *true*, ou seja, para ser habilitada é necessário 1 ficha de valor *true*, o que determina que o relatório pode ser visualizado.

O fluxo de falha foi mapeado seguindo o de confirmação, porém para esse fluxo é necessário que a ficha seja do tipo *false*. Esse mapeamento é ilustrado na figura 4.28.

4.6 Análise das redes de Petri coloridas

Nesta seção será descrito a análise das redes de Petri coloridas mapeadas na seção anterior. Para isso, utilizou-se o *software* CPN Tools a fim de gerar os grafos de alcançabilidade e utilizá-los como base das análises.

Diante disso, a partir da rede da figura 4.18 foi gerado o grafo de alcançabilidade, através do CPN Tools, obtendo o grafo ilustrado na figura 4.29. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações possíveis da rede através do disparo de uma determinada transição. A rede possui então uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para o entendimentos das análises, é necessário a compreensão do que cada estado do grafo representa, para isso, foi gerado o relatório do grafo no CPN Tools e ele pode ser visto no apêndice B. A figura 4.30 ilustra as propriedades do grafo. É importante destacar que, de acordo com a marcação inicial definida para gerar esse grafo, algumas transições ficaram "mortas". Isso ocorreu devido ao fato da transição "AltIsNotOkRegRel", por exemplo, só ser habilitada se possuir 1 ficha do tipo *false*. Para a marcação inicial

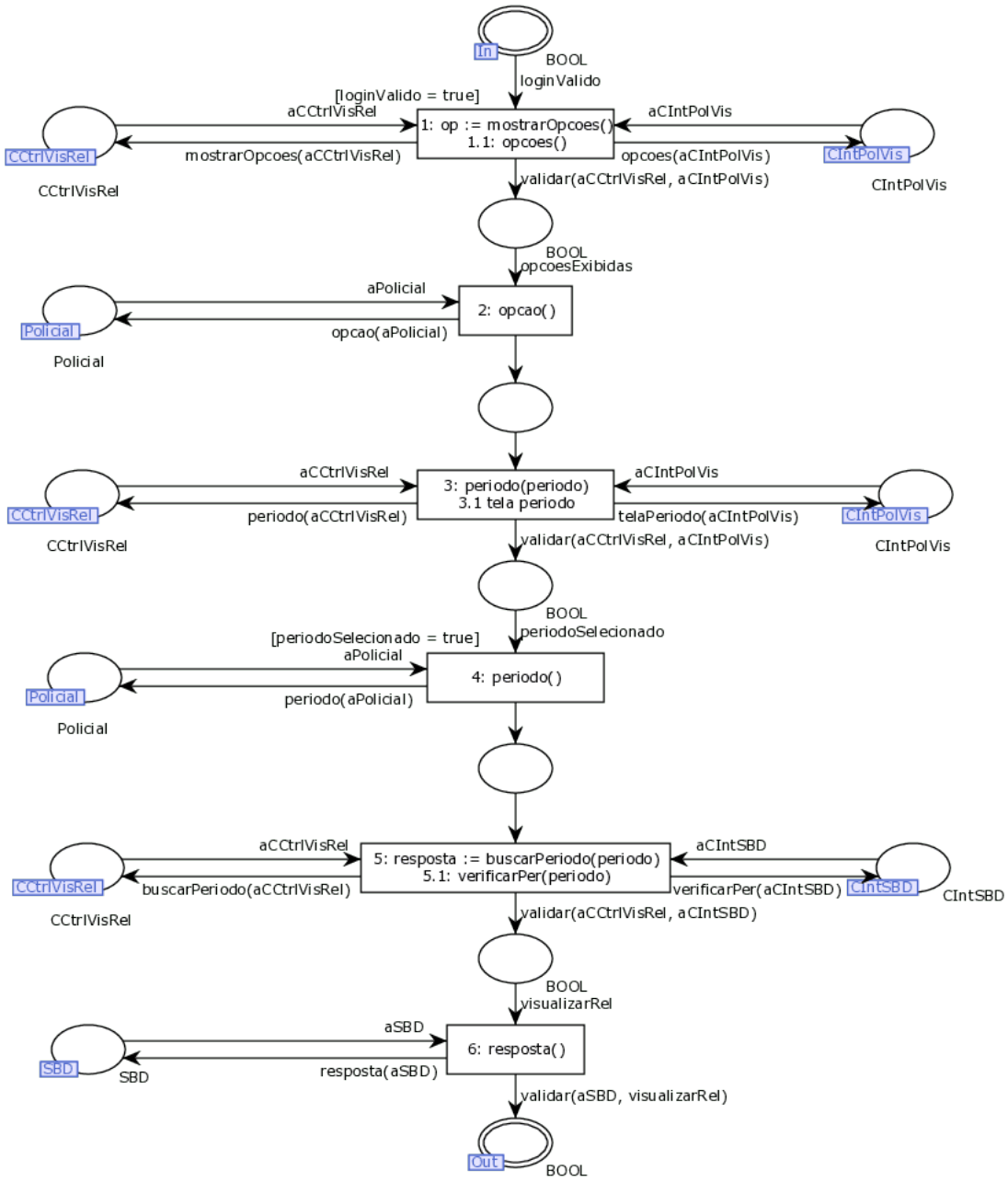

Figura 4.26: Rede de Petri colorida: transição de substituição "SelecionarPeriodo".

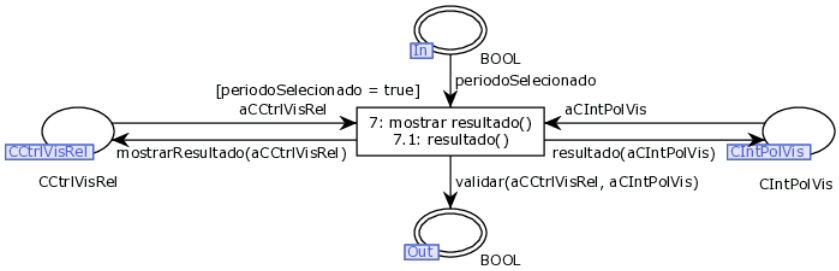

Figura 4.27: Rede de Petri colorida: transição de substituição "AltIsOk".

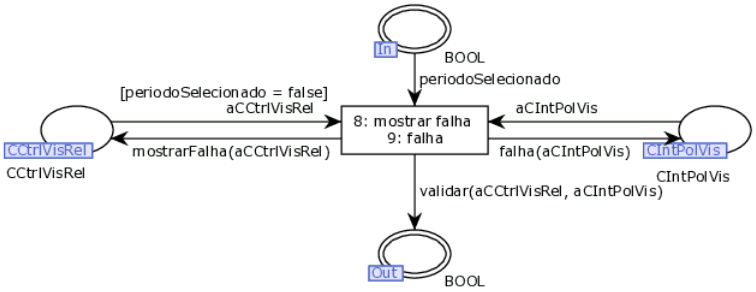

Figura 4.28: Rede de Petri colorida: transição de substituição "AltIsNotOk".

definida e a sequência de disparos realizada até essa etapa, o valor da ficha no lugar antecessor a essa transição era *true*.

Outro ponto importante, é o fato do grafo não apresentar sequência de ocorrências infinitas. Por fim, é possível identificar que através de uma sequência de disparos a rede leva da marcação inicial até a final representando todos os passos descritos no diagrama de casos de usos e seus respectivos diagramas de sequência.

Para o diagrama de sequência de realizar login ilustrado na figura 4.19, foi obtido o grafo de alcançabilidade ilustrado na figura 4.31. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações da rede através do disparo de uma determinada transição. A rede possui então, uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para o entendimentos das análises do mapeamento do diagrama de realizar login, é necessário a compreensão do que cada estado do grafo representa. Diante disso, foi gerado o relatório do grafo no CPN Tools, como anteriormente, e ele pode ser visto no apêndice B. A figura 4.32 ilustra as propriedades do grafo.

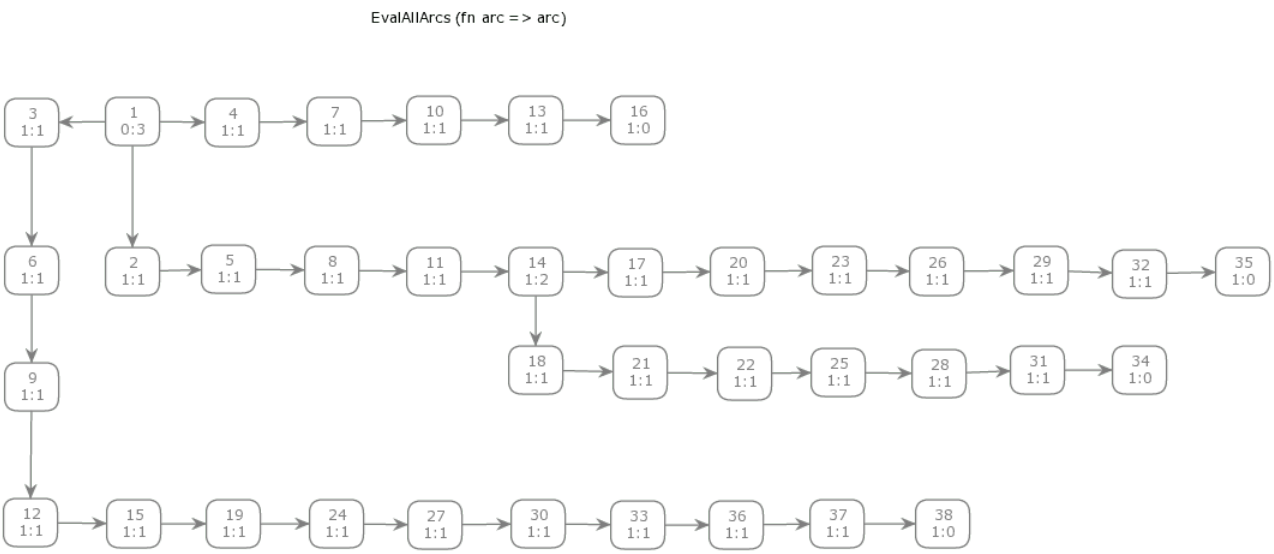

Figura 4.29: Grafo de alcançabilidade diagrama de casos de uso.

Para as propriedades do grafo de alcançabilidade de realizar login, é importante destacar o fato do grafo não apresentar sequência de ocorrências infinitas. Além disso, é possível identificar que através de uma sequência de disparos a rede leva da marcação inicial até a final representando todos os passos descritos no diagrama de sequência com o objetivo de realizar login.

Para o diagrama de sequência de registrar relatório ilustrado na figura 4.20, foi obtido o grafo de alcançabilidade ilustrado na figura 4.31. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações da rede através do disparo de uma determinada transição. A rede possui então, uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para entender as análises do mapeamento do diagrama de registrar relatório, é necessário a compreensão do que cada estado do grafo representa e, novamente, foi gerado o relatório do grafo no CPN Tools. O relatório pode ser visto no apêndice B. A figura 4.34 ilustra as propriedades do grafo.

Para as propriedades do grafo de alcançabilidade de registrar relatório, é importante destacar o fato do grafo não apresentar sequência de ocorrências infinitas. Além disso, é importante destacar que, de acordo com a marcação inicial definida para gerar esse grafo,

```
 Home Properties
------------------------------------------------------------------------

  Home Markings
     None


 Liveness Properties
------------------------------------------------------------------------

  Dead Markings
     [14,19,20,21]

  Dead Transition Instances
     AltIsNotOkRegRel'mostrarFalha_falha 1
     AltIsNotOkVisRel'mostrarFalha_falha 1
     AltIsOkRegRel'mostrar_confirmacao_confirmacao 1
     AltIsOkVisVisRel'mostrarResultado_resultado 1
     registrarRelatorioDados'confirmacao 1
     registrarRelatorioDados'dados_setDados 1
     registrarRelatorioDados'gravar_registro_getDados 1
     registrarRelatorioDados'mostrarTelaRegistro_telaRegistro 1
     registrarRelatorioDados'salvarDados 1
     visualizarRelatorioDados'buscarPeriodo_verificarPer 1
     visualizarRelatorioDados'periodo 1
     visualizarRelatorioDados'periodo_tela_periodo 1
     visualizarRelatorioDados'resposta 1

  Live Transition Instances
     None


 Fairness Properties
------------------------------------------------------------------------
     No infinite occurrence sequences.
```

Figura 4.30: Propriedades grafo de alcançabilidade casos de uso.

a transição "AltIsNotOk" ficou morta pois, essa transição só é habilitada se possuir 1 ficha do tipo *false* no lugar antecessor a ela. Para a marcação inicial definida e a sequência de disparos realizada até essa etapa, o valor da ficha no lugar antecessor a essa transição era *true* o que não habilitou em nenhum momento a transição.

Por fim, é possível identificar que através de uma sequência de disparos a rede leva da marcação inicial até a final representando todos os passos descritos no diagrama de sequência com o objetivo de registrar o relatório.

Para o diagrama de sequência de visualizar relatório ilustrado na figura 4.25, foi obtido o grafo de alcançabilidade ilustrado na figura 4.31. Avaliando o grafo, é possível destacar

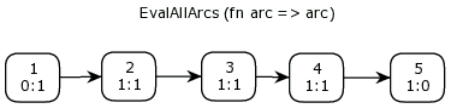

Figura 4.31: Grafo de alcançabilidade diagrama de sequência realizar login.

```
Home Properties
--------------------------------------------------------------------------------

 Home Markings
    [5]


Liveness Properties
--------------------------------------------------------------------------------

 Dead Markings
    [5]

 Dead Transition Instances
    None

 Live Transition Instances
    None


Fairness Properties
--------------------------------------------------------------------------------
    No infinite occurrence sequences.
```

Figura 4.32: Propriedades grafo de alcançabilidade diagrama de sequência realizar login.

que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações da rede através do disparo de uma determinada transição. A rede possui então, uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para compreender melhor as análises do mapeamento do diagrama de visualizar relatório, é necessário a compreensão do que cada estado do grafo representa, outra vez, foi gerado o relatório do grafo no CPN Tools e ele pode ser visto no apêndice B. A figura

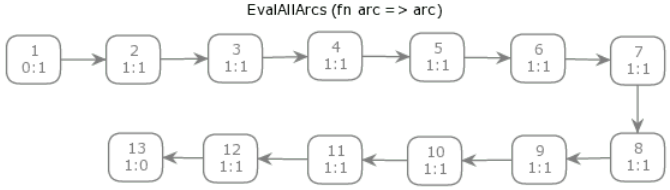

Figura 4.33: Grafo de alcançabilidade diagrama de sequência registrar relatório.

```
Home Properties
------------------------------------------------------------------------

 Home Markings
    [13]


Liveness Properties
------------------------------------------------------------------------

 Dead Markings
    [13]

 Dead Transition Instances
    AltIsNotOk'mostrarFalha_falha 1

 Live Transition Instances
    None


Fairness Properties
------------------------------------------------------------------------
    No infinite occurrence sequences.
```

Figura 4.34: Propriedades grafo de alcançabilidade diagrama de sequência registrar relatório.

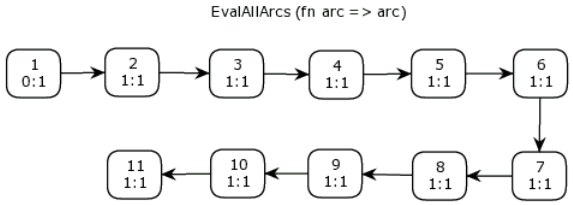

Figura 4.35: Grafo de alcançabilidade diagrama de sequência visualizar relatório.

4.36 ilustra as propriedades do grafo.

Para as propriedades do grafo de alcançabilidade de visualizar relatório, é importante destacar o fato do grafo não apresentar sequência de ocorrências infinitas. Além disso, é importante destacar que, de acordo com a marcação inicial definida para gerar esse grafo, a transição "AltIsNotOk" ficou morta pois, essa transição só é habilitada se possuir 1 ficha do tipo *false* no lugar antecessor a ela. Para a marcação inicial definida e a sequência de disparos realizada até essa etapa, o valor da ficha no lugar antecessor a essa transição era *true* o que não habilitou em nenhum momento a transição.

Por fim, é possível identificar que através de uma sequência de disparos a rede leva da marcação inicial até a final representando todos os passos descritos no diagrama de

Figura 4.36: Propriedades grafo de alcançabilidade diagrama de sequência visualizar relatório.

sequência com o objetivo de visualizar o relatório.

4.7 Discussão dos resultados

O mapeamento dos diagramas de casos de uso e sequência para as redes de Petri segura e colorida possibilitou a análise das vantagens e desvantagens de cada uma das abordagens. É possível destacar que a rede de Petri segura apresenta algumas limitações enquanto a rede de Petri colorida apresenta mais recursos. A rede de Petri colorida é a combinação entre a teoria da Rede de Petri e a linguagem de programação funcional, ou seja, a rede colorida apresenta tudo o que a rede segura possui mas, tem a linguagem de programação. O fato da rede ter a linguagem de programação, traz mais capacidade de modelar modelos complexos.

As figuras 4.10 e 4.18 ilustram, respectivamente, o mapeamento do diagrama de casos de uso para a rede de Petri segura e a rede de Petri colorida. Ao analisar as figuras, é possível notar diferenças significativas na representação do mesmo diagrama em cada modelo

de rede de Petri. Na figura 4.10, a rede segura é mapeada por meio do uso de lugares, que representam o ator e seus casos de uso, e transições para modelar os relacionamentos entre eles. O disparo de cada transição na rede segura representa a execução de um caso de uso. Essa transformação é um modelo simplificado de representação dos eventos que ocorrem durante a execução de caso de uso.

Por outro lado, a rede colorida ilustrada na figura 4.18 utiliza transições de substituição para representar os casos de uso. Cada transição na rede colorida representa um caso de uso específico, e inclui o mapeamento completo do fluxo descrito no diagrama de sequência correspondente. Por exemplo, o caso de uso "Realizar Login" é descrito em um diagrama de sequência, que é então mapeado para a rede colorida, onde é representado como uma transição de substituição no mapeamento do diagrama de casos de uso. Dentro desta transição de substituição, é descrito todo o fluxo de realizar login. É importante observar que a rede colorida considera apenas os casos de uso diretamente ligados aos atores, ignorando casos de uso de extensão, por exemplo.

Na análise dos grafos de alcançabilidade resultantes do mapeamento do diagrama de casos de uso para ambas as abordagens (figuras 4.11 e 4.29), observam-se as diferenças significativas descritas nos parágrafos anteriores. No grafo de alcançabilidade da rede segura (figura 4.11), identificam-se mais de uma ramificação no estado S0, relacionados ao ator "Policial". Essas ramificações decorrem do fato de que a presença de uma ficha no lugar "Policial" habilita todas as transições dos casos de uso. Na rede de Petri, as ramificações são representadas pela propriedade de conflito das redes de Petri. Para o mapeamento do diagrama de casos de uso, embora seja possível executar um caso de uso e seus relacionamentos, os eventos detalhados do caso de uso não são descritos nesse mapeamento.

Já no grafo de alcançabilidade da rede colorida (figura 4.29), também é observado ramificações no estado 1. Isso ocorreu porque a marcação inicial da rede foi definida com uma lista completa de todos os casos de uso, habilitando todas as transições dos casos de uso. Na rede mapeada, também é possível observar a presença de conflitos nessas ramificações. No entanto, é notável que esse grafo apresenta mais estados, representando os

eventos detalhados de cada caso de uso. Embora esse mapeamento não descreva os relacionamentos entre os casos de uso, oferece uma descrição detalhada dos eventos ocorridos em cada caso de uso conforme representado em um diagrama de sequência.

Ambas as abordagens apresentam diferentes formas de descrever os diagramas de casos de uso e permitem uma análise formal dos mesmos. No entanto, a abordagem da rede segura é especialmente notável pela sua simplicidade. Ao utilizar lugares para representar atores e casos de uso, e transições para indicar a execução de cada caso de uso, a rede segura adota uma abordagem simplificada. Apesar de sua simplicidade, a abordagem da rede segura ainda possibilita uma análise formal dos casos de uso, o que torna essa abordagem uma opção atrativa a considerar. Enquanto a rede colorida oferece uma maior riqueza de detalhes ao representar o fluxo de cada caso de uso de forma precisa, a abordagem da rede segura simplifica a representação dos modelos.

Ao optar por uma das abordagens, é importante levar em consideração a diferença na representação dos modelos. A abordagem da rede segura, embora mais simplória, mantém a capacidade de análise formal, o que é essencial no processo de escolha entre as abordagens disponíveis. Em resumo, a abordagem da rede segura se destaca pela sua simplicidade, proporcionando uma representação mais direta e fácil de entender dos casos de uso. É fundamental ressaltar essa vantagem ao considerar as diferentes abordagens disponíveis para análise formal.

Analisando as figuras 4.13 e 4.19, pode-se observar que as abordagens de rede de Petri segura e colorida apresentam semelhanças na representação dos diagramas de sequência. Ambas abordagens utilizam lugares para descrever os objetos de interação do diagrama de sequência e transições para descrever a troca de mensagens entre eles. A rede de Petri segura descreve a sequência de eventos utilizando uma ordem entre os lugares e transições. A rede carrega uma única ficha que habilita o disparo da transição e, esta ficha leva do estado inicial ao estado final.

A rede colorida, por sua vez, utiliza tipos de lugares, fichas e funções para descrever a sequência de eventos. Cada ficha, dependendo de seu tipo, habilita ou não uma transição, permitindo levar a rede de um estado inicial a um estado final. Isso é possível graças

à linguagem de programação da rede colorida, que permite limitar os tipos de fichas necessários para disparar uma transição, bem como limitar os tipos de lugar. Ou seja, é possível controlar a rede a partir dos conjuntos de cores associados aos lugares e transições, o que pode ser observado nos mapeamentos de diagramas apresentados anteriormente, como ilustrado nas figuras 4.19, 4.20 e 4.25. Nessas redes, existem os lugares que servem principalmente para controlar o fluxo, como por exemplo os lugares do tipo *BOOL* entre as transições, permitindo apenas a passagem de fichas pela rede. Além disso, as redes também possuem funções que podem modificar as fichas durante a execução da rede.

Analisando as figuras que ilustram o mapeamento das redes seguras e coloridas para um mesmo diagrama de sequência, é possível destacar que a rede colorida é mais específica na descrição dos eventos através da linguagem de programação.

Por outro lado, a rede de Petri segura não possui a linguagem de programação o que limita um pouco representar os modelos. O controle da rede é feito através dos números de fichas inseridos em cada lugar e o número de fichas necessários para disparar uma transição, o que habilita e desabilita as transições. Pelo fato do mapeamento aqui apresentado ser 1-limitado, é possível controlar a rede a partir dessa única ficha transferida de um lugar para outro na rede. Mas, apesar disso, a rede de Petri segura consegue mapear o diagrama de sequência e possibilita uma análise formal dos modelos.

É possível constatar a limitação na representação dos modelos na rede segura comparando os grafos de alcançabilidade gerados por ambas as redes. Ao analisar os grafos apresentados na figura 4.15 e na figura 4.33, que representam o mesmo diagrama, é observado que na rede segura, uma ficha inserida no estado que representa o lugar antecessor das transições de "mostrar resultado" ou "mostrar falha" possui duas bifurcações, uma para cada fluxo. Isso ocorre porque essa única ficha habilita ambas as transições na rede. No entanto, na rede colorida, a ficha inserida no lugar antecessor dessas transições habilita apenas uma delas, devido à limitação do valor da ficha. Cada transição na rede colorida possui um valor distinto, possibilitando o disparo de apenas uma delas de acordo com o valor da ficha.

Na rede segura, o fato de ter 1 ficha inserida no lugar, já habilita ambas as transições.

Isso porque a ficha não possui diferença de valor, possibilitando disparar ambas as transições. Além disso, é possível analisar a sequência ocorrida de acordo com as transições habilitadas e disparadas na rede segura. Na rede colorida, é possível analisar esses mesmos pontos e além disso, analisar o tipo de cada ficha e limitar o disparo de cada transição. É possível descrever a sequência dos eventos com mais detalhes.

Além disso, como descrito na seção das regras elaboradas, a rede segura não consegue abstrair muito os modelos apresentando limitações como por exemplo no operador de alto nível LOOP do diagrama de sequência. A rede de Petri 1-limitada não possui recursos para representar o número mínimo e máximo de interações de um LOOP, sendo possível realizar o mapeamento de LOOP apenas quando não há um número máximo e mínimo de interações.

Outro ponto importante a destacar são os lugares das redes de Petri. O CPN Tools possui a fusão (*Fusion*) dos lugares o que, de acordo com a própria documentação do CPN Tools, é um método para definir o conjunto dos locais, de modo que qualquer coisa que aconteça a cada local de um conjunto também aconteça a todos os outros locais do conjunto. Os lugares são então funcionalmente idênticos [29].

A fusão permitiu agrupar os lugares iguais da rede. Isso pode ser visualizado nas figuras ilustradas nos mapeamentos. As figuras 4.19, 4.20 e 4.25 ilustram as redes coloridas mapeadas a partir dos diagrama de sequência de "Realizar Login", "Registrar Relatório" e "Visualizar Relatório". Ao analisar essas figuras, é possível notar que para representar um mesmo lugar, utilizou-se a fusão, o que significa que aquele lugar representa o mesmo objeto do diagrama e possui o mesmo tipo de ficha.

Por outro lado, nas redes seguras ilustradas nas figuras 4.13, 4.16 e A.2 obtidas através dos mesmos diagramas mencionados no parágrafo anterior, foi necessário criar cópias dos lugares utilizando numerais a partir de 0 para representar um mesmo lugar.

Apesar de suas limitações, a rede segura mostrou-se capaz de mapear a maioria dos elementos do diagrama de sequência, incluindo os operadores de alto nível da UML 2.0. É importante destacar que muitos dos conceitos utilizados na rede segura são semelhantes aos da rede colorida, uma vez que a teoria por trás de ambas as redes é a mesma. Isso

sugere que a abordagem da rede segura pode ser uma alternativa viável para validar diagramas de sequência da UML, uma vez que é possível representar grande parte dos elementos em um modelo formal e verificável.

É possível destacar também que ambas as redes possui limitações para representar um diagrama de casos de uso. O diagrama de casos de uso não apresenta um fluxo de trabalho o que dificulta o mapeamento dele para uma rede de Petri. Foi possível representar a sequência do que aconteceria caso o utilizador selecionasse determinado caso de uso. Nesse caso, é possível observar também que a rede colorida é mais robusta, possuindo transições de substituição capaz de descrever todo o fluxo de um determinado caso de uso descrito a partir do diagrama de sequência.

Para o diagrama de sequência ambas as redes apresentaram recursos para o mapeamento. O diagrama de sequência é mais fácil ser mapeado pois possui uma sequência de eventos, o que possibilita demonstrar mais facilmente na rede de Petri. Mais uma vez a rede de Petri colorida possui mais recursos para esse mapeamento pois, como dito anteriormente, a rede segura não possui recursos suficientes para mapear, por exemplo, o número mínimo e máximo de interações em um LOOP.

Capítulo 5

Conclusão

A modelação de *software* é uma tarefa essencial para garantir a qualidade e confiabilidade de sistemas computacionais. Nesse contexto, a UML e as redes de Petri são ferramentas amplamente utilizadas para representar e analisar sistemas complexos.

Este trabalho contribui para a pesquisa de mapeamento dos diagramas de sequência e casos de uso da UML 2.0 para as redes de Petri segura e colorida. Durante o desenvolvimento deste estudo, uma pesquisa foi realizada e está descrita em [24]. Essa pesquisa teve o objetivo de identificar os diagramas da UML mais utilizados em conjunto com as redes de Petri, assim como os tipos de redes de Petri empregados. De acordo com os resultados obtidos, o diagrama de sequência é o mais comumente utilizado em conjunto com a rede de Petri colorida. No entanto, a pesquisa evidenciou a carência de regras para realizar o mapeamento o diagrama de sequência da UML 2.0 para as redes de Petri seguras.

O presente trabalho representou um desafio, já que, apesar de diversas pesquisas na área de mapeamento de diagramas de sequência da UML 2.0 para redes de Petri, não foi encontrado nenhum estudo que se dedicasse especificamente à rede segura. Em contrapartida, foram encontrados diversos trabalhos que utilizavam as redes de alto nível, que se baseiam na mesma teoria da rede segura. Dessa forma, foram criadas regras para o mapeamento dos diagramas de sequência e casos de uso para a rede segura. A aplicação dessas regras permitiu verificar a viabilidade do mapeamento e, apesar de algumas limitações, possibilitou a representação dos elementos de ambos os diagramas.

Realizar o mapeamento dos mesmos diagramas para as redes de Petri seguras e coloridas permitiu constatar que as redes de Petri coloridas apresentam uma capacidade maior de detalhamento para descrever os modelos, em comparação com as redes de Petri seguras. Essa capacidade se deve à possibilidade de associar um conjunto de fichas a cada lugar, funções e condições específicas a cada transição, fusão de lugares e outras características. Mesmo com as limitações encontradas na representação dos modelos na rede segura, como mencionado anteriormente, os resultados obtidos nesse trabalho demonstram que ambas as abordagens são relevantes para a análise e verificação dos requisitos do *software*.

Existem diversas possibilidades de avanços na pesquisa de mapeamento de diagramas de sequência e casos de uso para redes de Petri. Como trabalho futuro, sugere-se a criação de um *software* que possa realizar esse mapeamento de forma automática, seguindo as regras estabelecidas nesse trabalho. Esse *software* poderia ajudar a simplificar o processo de mapeamento, tornando-o mais eficiente e reduzindo a necessidade de intervenção manual. Além disso, uma vez que o mapeamento tenha sido concluído, seria possível realizar análises automatizadas para verificar a corretude do diagrama e identificar possíveis problemas ou falhas. Dessa forma, o uso de um *software* desse tipo poderia trazer benefícios significativos para a modelação e análise de sistemas complexos.

Bibliografia

[1] T. C. S. Vargas, "A história da UML e seus diagramas," *Rev. Odontol. Univ. Federal de Santa Catarina*, vol. 18, n.º 3, pp. 265–274, 2006.

[2] "Unified Modeling Language 2.5.1," em Object Management Group, 2017. URL: https://www.omg.org/spec/UML/2.5.1/About-UML.

[3] G. T. A. Guedes, *UML2 Uma abordagem prática*, Novatec editora Ltda. São Paulo, SP, Brasil: Novatec, 2011, ISBN: 978-85-7522-281-2.

[4] L. P. Rezende, "WorkFlow net Possibilística aplicada aos Sistemas de Gerenciamento de Processos de Negócios Flexíveis," tese de doutoramento, Universidade Federal de Uberlândia, jun. de 2017.

[5] F. Jouault, F. Allilaire, J. Bézivin e I. Kurtev, "ATL: A model transformation tool," *Science of Computer Programming*, vol. 72, n.º 1, pp. 31–39, 2008, Special Issue on Second issue of experimental software and toolkits (EST), ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2007.08.002. URL: https://www.sciencedirect.com/science/article/pii/S0167642308000439.

[6] A. Kleppe, J. Warmer e W. Bast, *MDA Explained: The Model Driven Architecture - Practice and Promise*, 3rd. Boston, MA, U.S: Pearson Education, Inc., 2004, ISBN: 032119442X.

[7] O. Ribeiro, J. Fernandes, S. Tjell e J. B. Jorgensen, "Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net," 2007. URL: https://ieeexplore.ieee.org/document/4273282.

[8] M. Seidl, M. Scholz, C. Huemer e G. Kappel, *UML@ Classroom*, Series Editor. Heidelberg, Germany: Springer, 2012, ISBN: 978-3-319-12742-2.

[9] J. Cardoso e R. Valette, *Redes de Petri*. Editora da UFSC, 1997, ISBN: 9788532800954.

[10] F. C. V. Benito, "Desdobramento para Redes de Petri k-limitadas," tese de mestrado, Universidade Federal do Paraná, ago. de 2010.

[11] E. M. M. Costa, *Redes de Petri e aplicações aos sistemas a eventos discretos*. 2011.

[12] T. Murata, "Petri Nets: Properties, Analysis and Applications," *PROCEEDINGS OF THE IEEE, VOL. 77, NO. 4, APRIL 1989*, 1989.

[13] "Platform Independent Petri net Editor." (), URL: `http://pipe2.sourceforge.net/` (acedido em 04/03/2020).

[14] "CPN Tools." (), URL: `https://cpntools.org/` (acedido em 04/03/2020).

[15] T. Shailesh, A. Nayak e D. Prasad, "Transformation of sequence diagram to timed Petri net using Atlas Transformation Language metamodel approach," *Journal of Software: Evolution and Process*, vol. 34, n.º 1, e2412, 2022. DOI: `https://doi.org/10.1002/smr.2412`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.2412`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2412`.

[16] S. Emadi e F. Shams, "Transformation of Usecase and Sequence Diagrams to Petri Nets," *ISECS International Colloquium on Computing, Communication, Control, and Management*, 2009.

[17] A. Alhroob, D. K. e A. Hossain, "Transforming UML Sequence Diagram to High Level Petri Net," *2nd International Conference on Software Technology and Engineering(ICSTE)*, 2010.

[18] O. Ribeiro e J. Fernandes, "Some Rules to Transform Sequence Diagrams into Coloured Petri Nets," 2006. URL: `http://hdl.handle.net/1822/43652`.

[19] M. A. Abrar, "Model Validation of System Requirements using Object Oriented Petri Nets," *Virtual University of Pakistan*, 2019. DOI: FEB2019@VUMULTAN.COM. URL: https://www.researchgate.net/profile/Asif-Abrar/publication/340132243_Model_Validation_of_System_Requirements_using_Object_Oriented_Petri_Nets_Model_Validation_of_System_Requirements_using_Object_Oriented_Petri_Nets/links/5e7a40eb92851cdfca2f27b6/Model-Validation-of-System-Requirements-using-Object-Oriented-Petri-Nets-Model-Validation-of-System-Requirements-using-Object-Oriented-Petri-Nets.pdf.

[20] O. Marek, "Generating Petri Nets for Reliability Analysis of Smart Grids From UML Diagrams," tese de mestrado, Masaryk University Faculty of Informatics, 2021.

[21] A. Gómez, R. J. Rodríguez, M.-E. Cambronero e V. Valero, "Profiling the publish/-subscribe paradigm for automated analysis using colored Petri nets," *Software & Systems Modeling*, vol. 18, pp. 2973–3003, 2019. DOI: https://doi.org/10.1007/s10270-019-00716-1. URL: https://link.springer.com/article/10.1007/s10270-019-00716-1.

[22] J. M. Silva, A. Z. O. Salmon, P. M. G. del Foyo e R. Silva, "REQUIREMENTS ENGINEERING AT A GLANCE: COMPARING GORE AND UML METHODS IN THE DESIGN OF AUTOMATED SYSTEMS," *Anais do XXII Congresso Brasileiro de Automática*, vol. 1, n.º 1, pp. 31–39, 2020, ISSN: 2525-8311. URL: https://www.sba.org.br.

[23] C. Baidada, E. M. Bouziane e A. Jakimi, "A New Approach for Recovering High-Level Sequence Diagrams from Object-Oriented Applications Using Petri Nets," *Procedia Computer Science*, vol. 148, pp. 323–332, 2019, THE SECOND INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2018, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.01.040. URL: https://www.sciencedirect.com/science/article/pii/S1877050919300407.

[24] L. Vidal, F. Benito e J. E. Fernandes, "Feasibility Study of the Use of Petri Nets in the Verification of UML Diagrams," em *Developments and Advances in Defense and Security*, Á. Rocha, C. H. Fajardo-Toro e J. M. Riola, eds., Singapore: Springer Nature Singapore, 2023, pp. 29–38.

[25] I. P. Souza, *Protótipo de Aplicativo para Gerenciamento de Relatórios a Partir de Dispositivos Móveis Para a Polícia Militar do Estado do Paraná*, Especialização, Curitiba, 2013.

[26] N. Marranghello, "Redes de Petri: Conceitos e Aplicações," rel. téc., mar. de 2005.

[27] Q. S. Mahdi, "Using Hierarchical Object Oriented Timed Colored Petri Nets for Design Radar Display," *NeuroQuantology*, vol. 20, pp. 1162–1174, 2022, ISSN: 1303-5150. DOI: 10.48047/NQ.2022.20.20.NQ109116.

[28] O. Ribeiro e J. Fernandes, "Animation-based Validation of Reactive Software Systems using Behavioural Models," tese de doutoramento, Universidade do Minho - Escola de Engenharia, 2009. URL: https://hdl.handle.net/1822/10218.

[29] C. Tools. "Fusion places." (jan. de 2018), URL: https://cpntools.org/2018/01/09/fusion-places/.

Apêndice A

Análise Rede de Petri segura

Nesta seção será apresentada a análise da rede de Petri segura de visualizar relatório.

A.1 Visualizar Relatório

Para a a análise da rede de visualizar relatório, será ilustrado o diagrama e a rede novamente. A figura A.1 ilustra o diagrama de sequência de visualizar relatório.

A figura A.2 ilustra a rede de Petri do diagrama de sequência ilustrado na figura A.1.

Utilizando a rede da figura A.2, foi gerado o grafo de alcançabilidade utilizando o PIPE, obtendo o grafo ilustrado na figura A.3. Avaliando o grafo, é possível destacar que todos os estados do grafo são alcançáveis, ou seja, é possível alcançar todas as marcações possíveis da rede através do disparo de uma determinada transição. A rede possui então uma marcação inicial que leva até a marcação final através de uma sequência de disparos.

Para o entendimento das análises, é necessário a compreensão do que cada estado do grafo representa. Eles representam lugares da rede como é detalhado a seguir:

- S0 por "RealizarLogin";
- S1 por "CCtrlVisRel";
- S2 por "CIntPolVis";

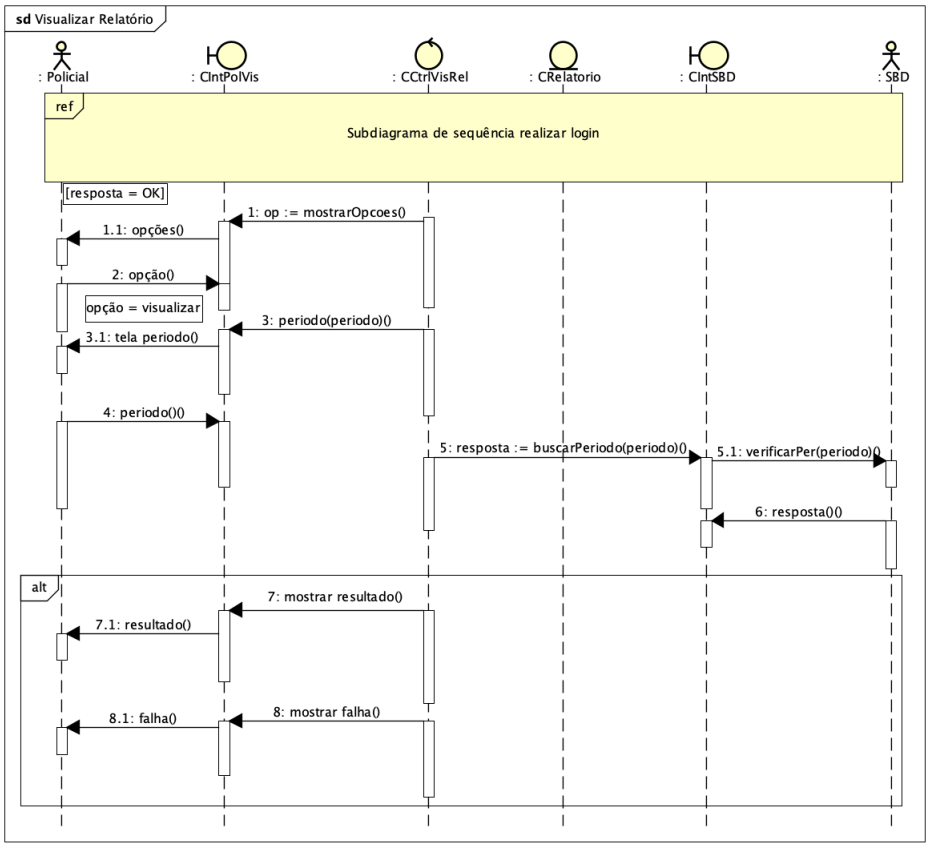

Figura A.1: Diagrama de sequência visualizar relatório **(Autoria: Souza [25])**.

- S3 por "Policial";
- S4 por "CIntPolVis0";
- S5 por "CCtrlVisRel0";
- S6 por "CIntPolVis1";
- S7 por "Policial0";
- S8 por "CIntPolVis2";
- S9 por "CCtrlVisRel1";
- S10 por "CIntSBD";
- S11 por "SBD";

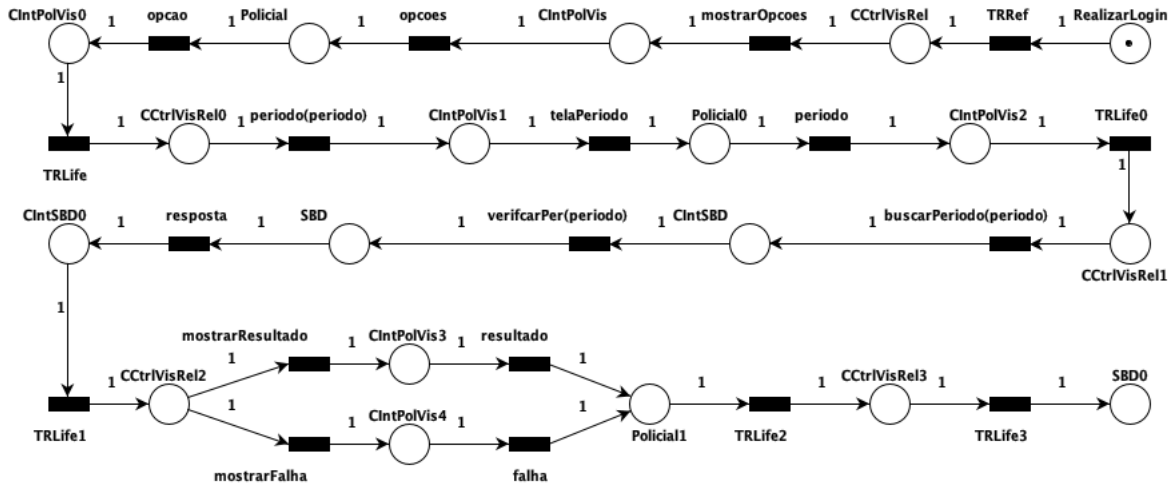

Figura A.2: Rede de Petri visualizar relatório.

- S12 por "CIntSBD0";
- S13 por "CCtrlVisRel2";
- S14 por "CIntPolVis3";
- S15 por "CIntPolVis4";
- S16 por "Policial1";
- S17 por "CCtrlVisRel3";
- S18 por "SBD0".

A partir disso, define-se a marcação inicial da rede avaliando as transições habilitadas. A marcação inicial da rede é dada então por: M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], considerando a ordem dos lugares como listado acima. A sequência de disparos é dada por:

- M0 [TRRef> M1;
- M1 [mostrarOpcoes> M2;
- M2 [opcoes> M3;

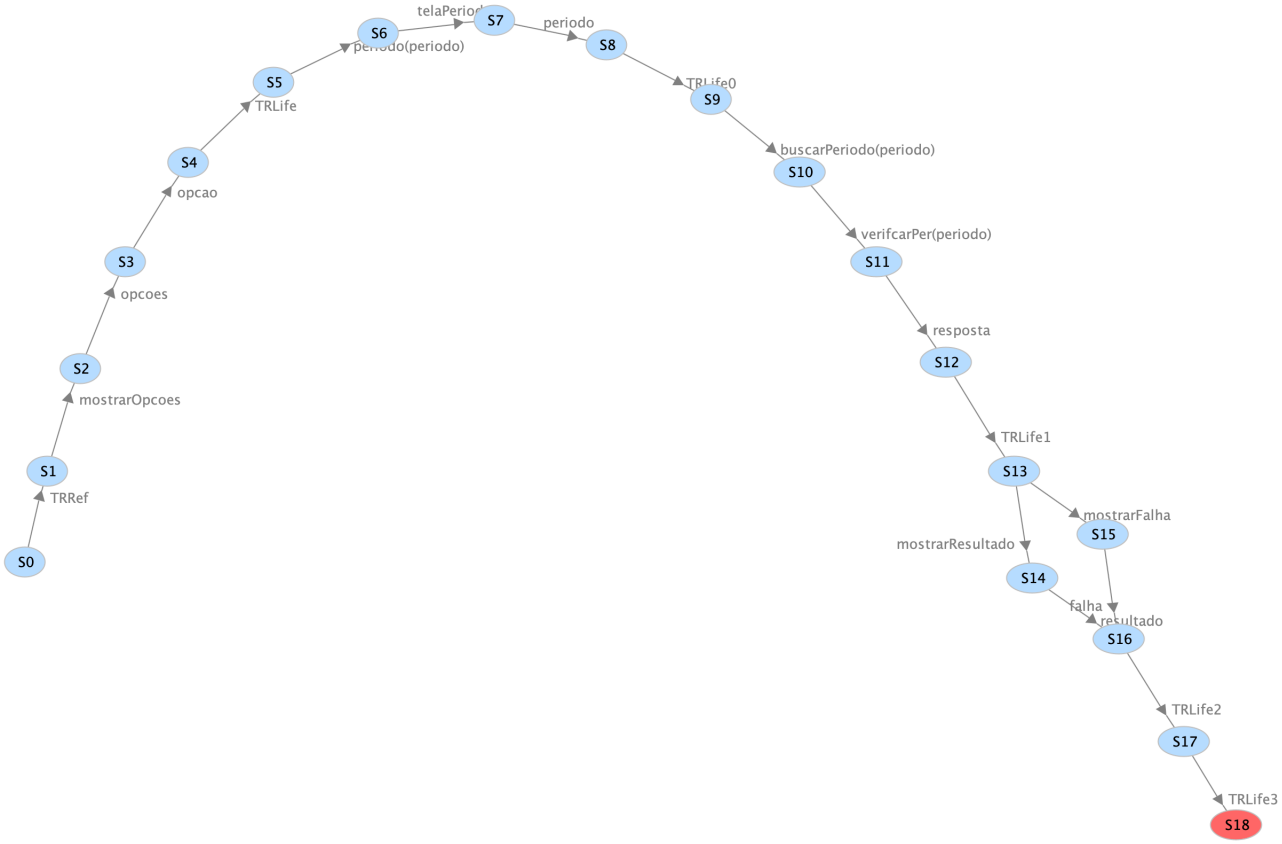

Figura A.3: Grafo de alcançabilidade rede visualizar relatório.

- M3 [opcao> M4;
- M4 [TRLife> M5;
- M5 [periodo(periodo)> M6;
- M6 [TelaPeriodo> M7;
- M7 [periodo>M8;
- M8 [TRLife0> M9;
- M9 [buscarPeriodo(periodo)> M10;
- M10 [verificarPer(periodo)> M11;
- M11 [resposta> M12;

- M12 [TRLife1> M13;
- M13 [mostrarResultado> M14;
- M13 [mostrarFalha> M15;
- M14 [resultado> M16;
- M15 [falha> M16;
- M16 [TRLife2> M17;
- M17 [TRLife3> M18.

Onde,

- M0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M1 = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M2 = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M3 = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M4 = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M5 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M6 = [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M7 = [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M8 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M9 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M10 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0];
- M11 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0];

- M12 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0];
- M13 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0];
- M14 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0];
- M15 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0];
- M16 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0];
- M17 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0];
- M18 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1].

Com base nas marcações e sequência de disparos listados acima é possível verificar então que, a marcação inicial da rede sensibiliza a transição “TRRef” para realizar o login e o disparo desta transição leva para a marcação M1 sensibilizando a transição “mostrarOpções” e levando para a marcação M2.

Após isso, a transição “opções” é habilitada levando para a marcação M3 quando disparada. M3 sensibiliza a transição “opção” e o disparo dessa transição leva para a marcação M4. A marcação M4 sensibiliza a transição “TRLife” que é a representação da linha de vida e que ao ser disparada leva para a marcação M5 que sensibiliza a transição “periodo(periodo)” e quando disparada leva para a marcação M6. M6 sensibiliza a transição “telaPeriodo” e quando disparada leva para a marcação M7. M7 sensibiliza a transição "periodo" e quando disparada leva para a marcação M8.

A transição M8 sensibiliza a transição "TRLife0" levando para a marcação M9. M9 sensibiliza a transição "buscarPeriodo(periodo)" e quando disparada leva para a marcação M10. M10 sensibiliza a transição "verificarPer(periodo)" indo para M11 e M11 sensibiliza a transição "resposta"indo para M12. M12 sensibiliza a transição "TRLife1" levando para a marcação M13. M13 possui um conflito, onde M14 sensibiliza a transição "mostrarResultado" e M15 sensibiliza a transição "mostrarFalha". Ambas as marcações levam para M16 com o disparo da transição "falha" ou "resultado"Ṁ16 sensibiliza "TrLife2" levando para a marcação M17 e M17 através de "TRLife3" vai para a marcação final, M18.

O diagrama de sequência da figura A.1 descreve a sequência necessária para visualizar o relatório no sistema. Analisando o disparo das transições que levam da marcação inicial até a final é possível verificar que a sequência de eventos necessária para visualizar o relatório é atendida conforme descrito no diagrama de sequência. Ou seja, o grafo descreve todos os cenários descritos no diagrama de sequência para a visualização do relatório e descreve todos os estados alcançáveis a partir de determinado momento descrito no diagrama.

A.2 Redes de Petri coloridas

Nesta seção, será descrito as transições de substituição presentes no mapeamento do diagrama de casos de uso para a rede de Petri colorida. Para facilitar a compreensão, será apresentado novamente a rede de Petri colorida correspondente ao caso de uso em questão. A Figura A.4 ilustra a rede de Petri colorida gerada a partir do diagrama de casos de uso.

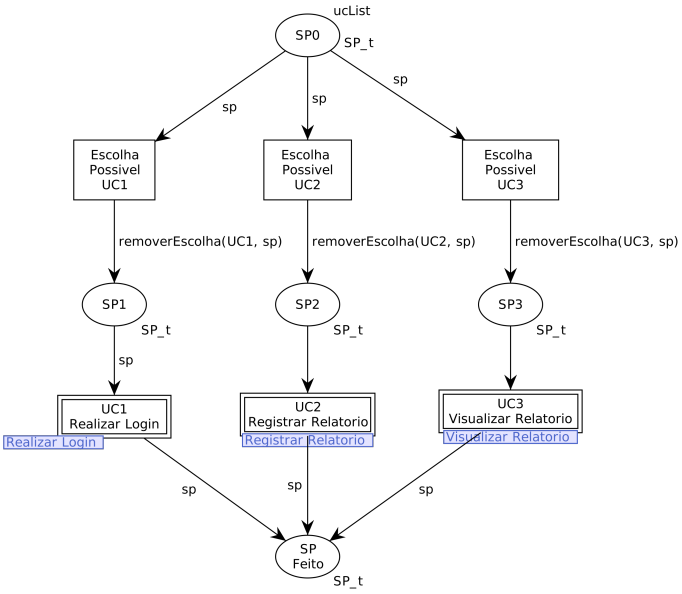

Figura A.4: Rede de Petri colorida diagrama de casos de uso.

A transição de substituição para o caso de uso "Realizar Login" foi mapeada de acordo

com o procedimento descrito na seção 4.5. Os objetos de interação foram mapeados como lugares e as trocas de mensagens entre eles foram representadas por transições, conforme ilustrado na figura A.5.

Uma diferença significativa é que, como essa rede é uma sub-rede da transição de substituição, ela possui um local de entrada e um local de saída. Além disso, a rede carrega a lista de casos de uso para que possa ser enviada novamente à rede superior.

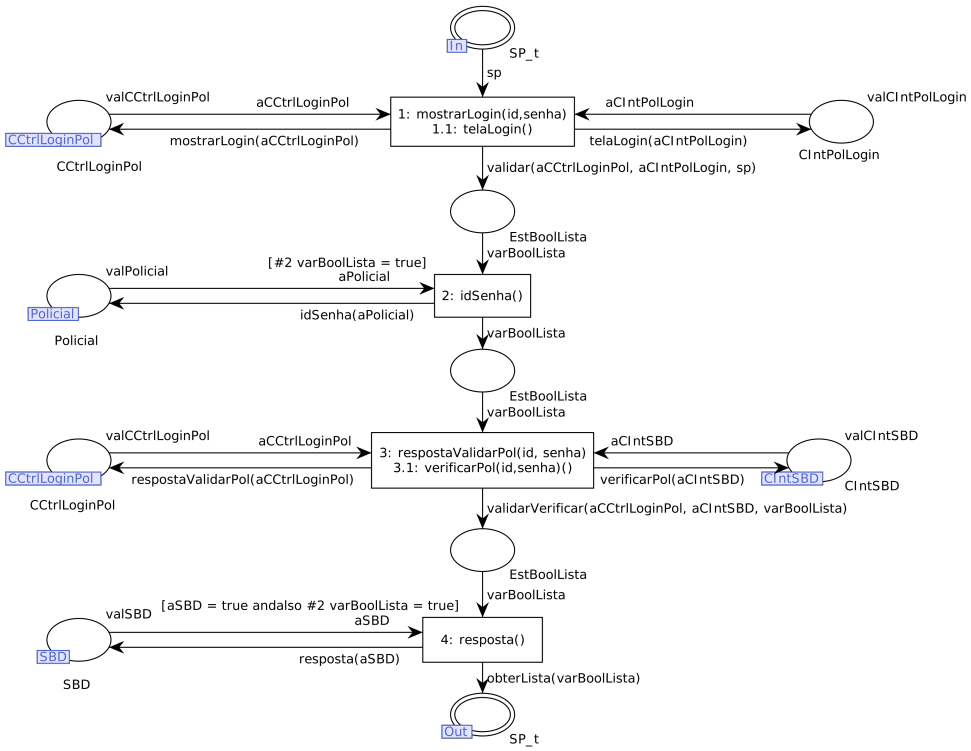

Figura A.5: Transição de substituição "Realizar Login".

Em seguida, para mapear a transição de substituição de "Registrar Relatório", seguiu-se os mesmos procedimentos descritos na seção 4.5, obtendo a rede ilustrada na figura A.6. Novamente, a diferença dessa rede para a rede apresentada na seção 4.5 são os locais de entrada e saída e a lista de casos de uso carregada na sub-rede e enviada novamente para a rede superior.

Da mesma forma como descrito na seção 4.5, foram inseridas as transições de substituição na rede colorida ilustrada na figura A.6 para a melhor compreensão da rede. Estas sub-redes também carregam a lista de casos de uso possíveis. O mapeamento completo é

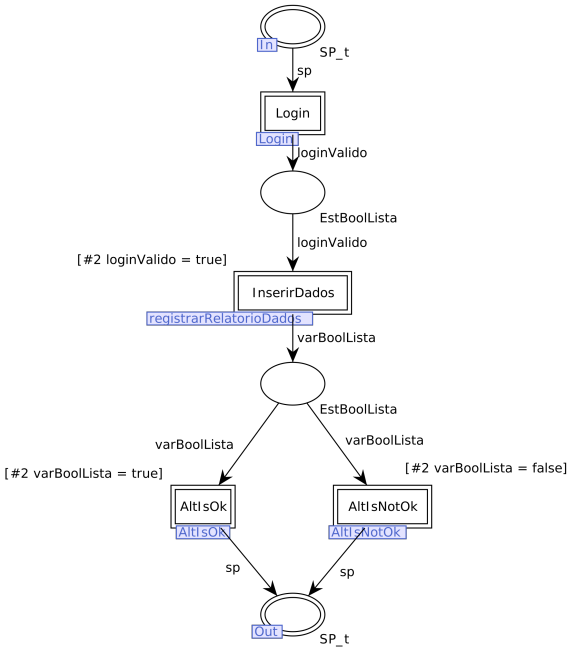

Figura A.6: Transição de substituição "Registrar Relatório".

ilustrado nas figuras A.7, A.8, A.9 e A.10.

A sub-rede da transição de substituição de "Realizar Login" presente na sub-rede de "Registrar Relatório", pode ser observada na figura A.7.

A sub-rede da transição de substituição de "Inserir Dados" presente na sub-rede de "Registrar Relatório", pode ser observada na figura A.8.

A sub-rede da transição de substituição de "AltIsOk" presente na sub-rede de "Registrar Relatório", pode ser observada na figura A.9.

A sub-rede da transição de substituição de "AltIsNotOk" presente na sub-rede de "Registrar Relatório", pode ser observada na figura A.10.

Em seguida, realizou-se o mapeamento da transição de substituição para "Visualizar Relatório", seguindo os mesmos procedimentos descritos na seção 4.5. O resultado desse mapeamento pode ser observado na figura A.11, ilustrando a rede de Petri correspondente ao caso de uso "Visualizar Relatório". Novamente, utilizou-se transições de substituição para a melhor compreensão do diagrama.

A sub-rede da transição de substituição de "Realizar Login" presente na sub-rede de

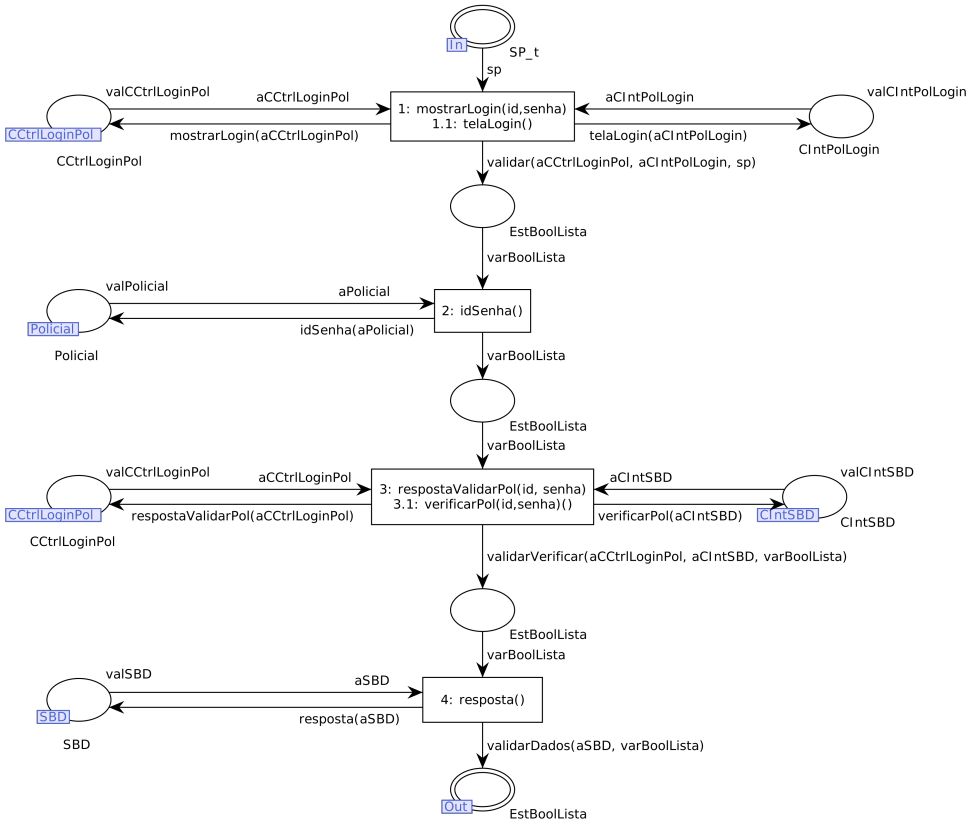

Figura A.7: Transição de substituição "Registrar Relatório - Login".

"Visualizar Relatório" pode ser observada na figura A.12.

A sub-rede da transição de substituição de "Selecionar Período" presente na sub-rede de "Visualizar Relatório" pode ser visualizado na figura A.13.

A sub-rede da transição de substituição de "AltIsOK" presente na sub-rede de "Visualizar Relatório" pode ser visualizado na figura A.14.

A sub-rede da transição de substituição de "AltIsNotOK" presente na sub-rede de "Visualizar Relatório" pode ser visualizado na figura A.15.

Como dito na seção 4.4, o conjunto de cores, variáveis e funções da rede colorida do diagrama de casos de uso é descrito no anexo B.

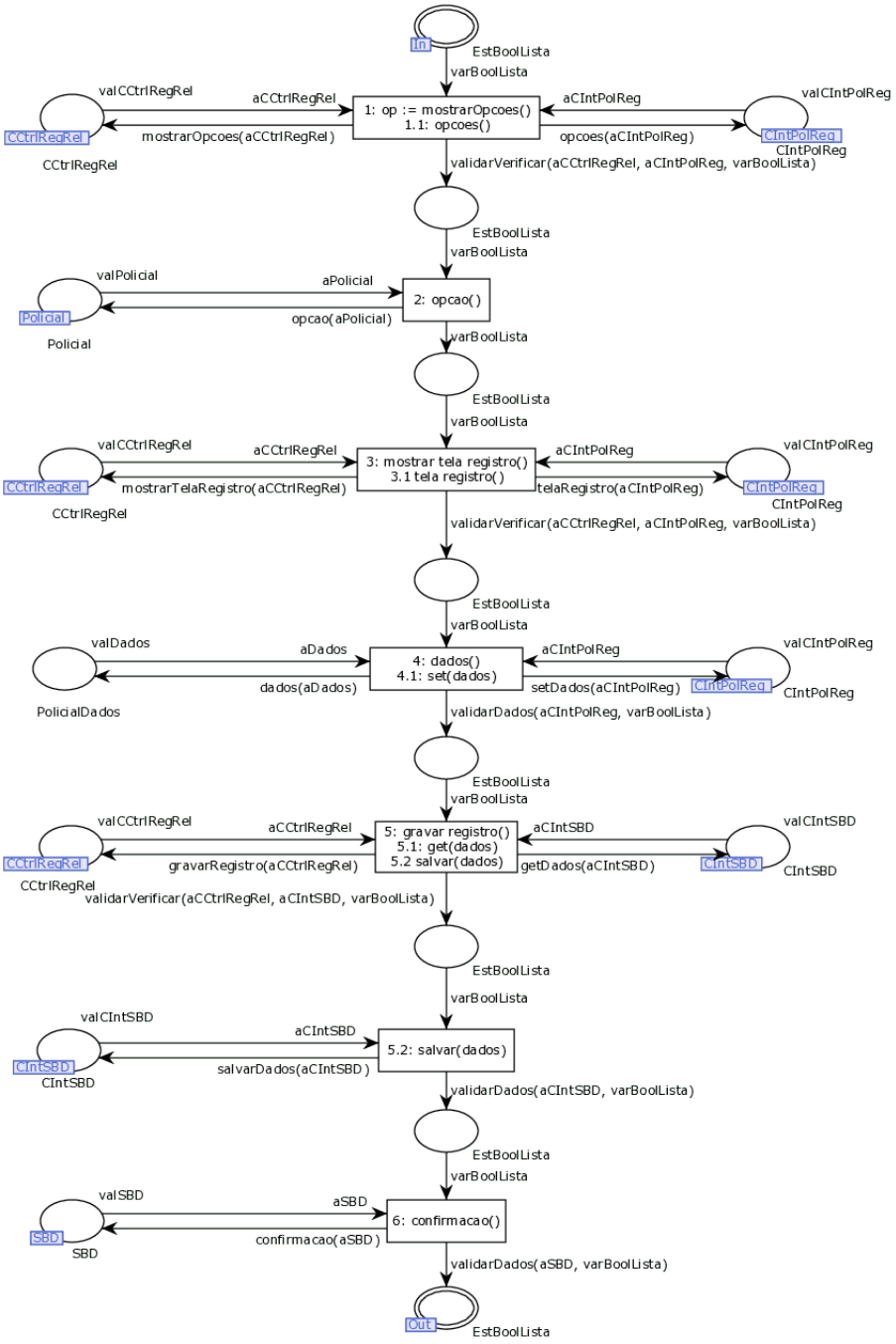

Figura A.8: Transição de substituição "Registrar Relatório - Dados".

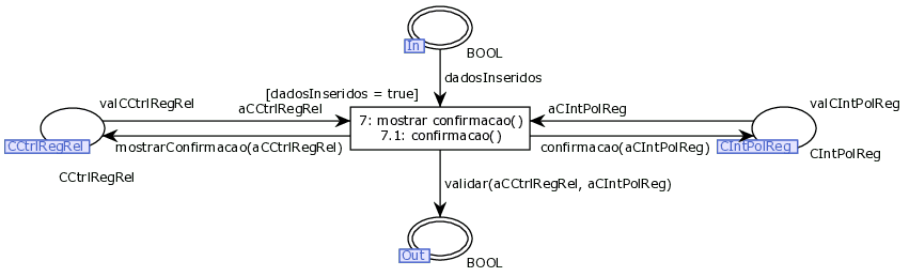

Figura A.9: Transição de substituição "Registrar Relatório - AltIsOk".

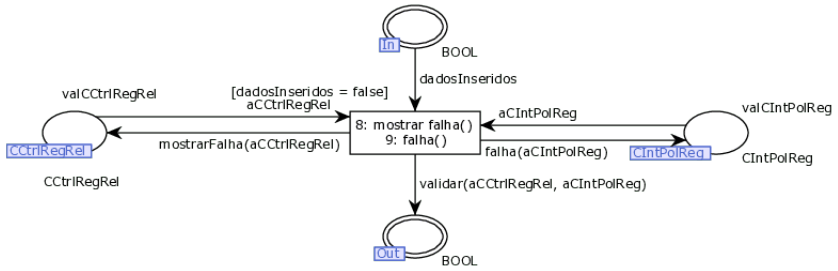

Figura A.10: Transição de substituição "Registrar Relatório - AltIsNotOk".

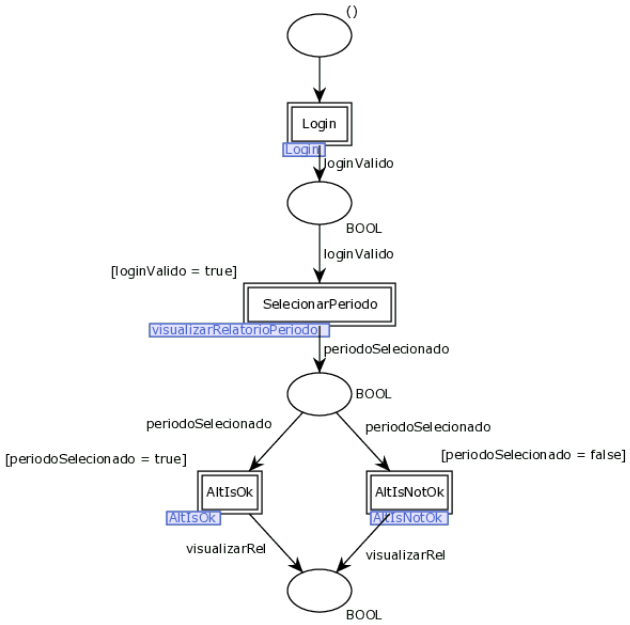

Figura A.11: Transição de substituição "Visualizar Relatório".

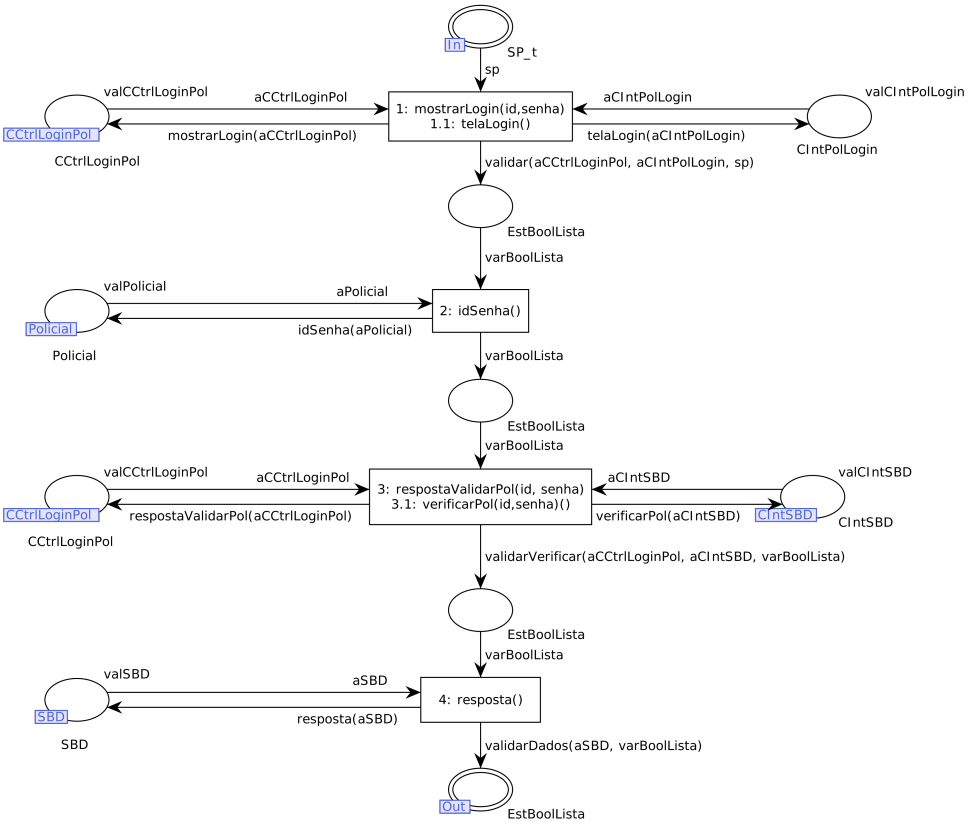

Figura A.12: Transição de substituição "Visualizar Relatório - Login".

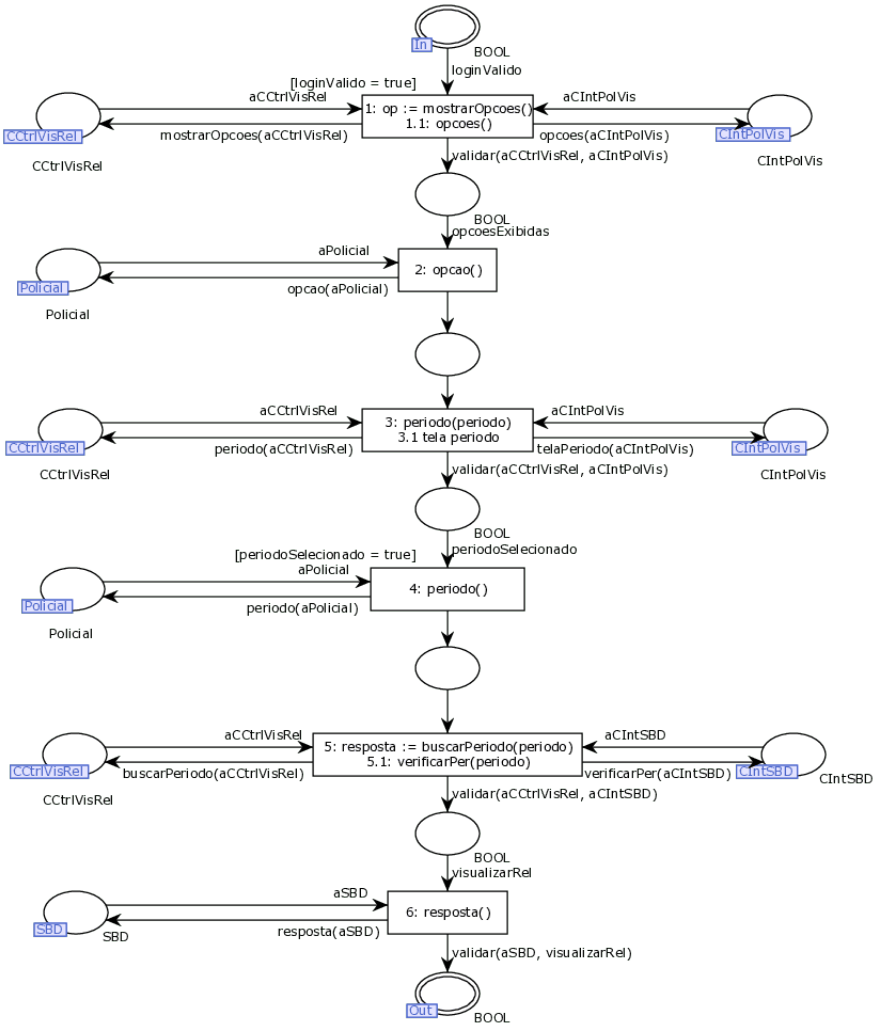

Figura A.13: Transição de substituição "Visualizar Relatório - Período".

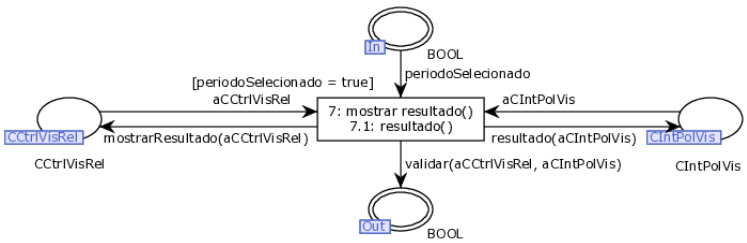

Figura A.14: Transição de substituição "Visualizar Relatório - AltIsOk".

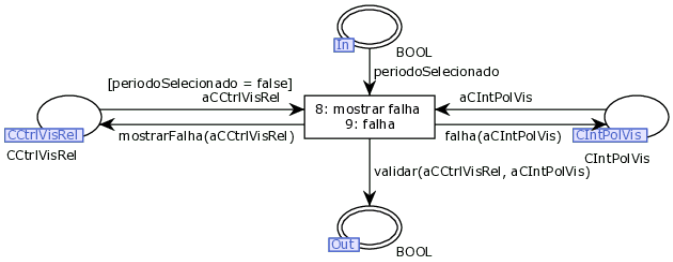

Figura A.15: Transição de substituição "Visualizar Relatório - AltIsNotOk".

Apêndice B

Códigos e relatórios das redes coloridas

O listing a seguir apresenta o conjunto de cores, variáveis e funções definidos na modelação do diagrama de casos de uso para a rede de Petri colorida.

Listing B.1: Código fonte rede de Petri colorida casos de uso

```
(* Standard priorities *)
     val P_HIGH = 100;
     val P_NORMAL = 1000;
     val P_LOW = 10000;
(* Standard declarations *)
     colset UNIT = unit;
     colset BOOL = bool;
     colset INT = int;
     colset INTINF = intinf;
     colset TIME = time;
     colset REAL = real;
     colset STRING = string;
(* Common Declarations *)
     (* Common Declarations *)
          (* PolicialEstruturas *)
               colset id = int;
               colset senha = string;
               colset PolicialLogin = product
                                                    id*
                                                    senha;
```

```
        colset Policial = PolicialLogin;
    colset CIntSBD = bool;
    colset SBD = bool;
    colset sp = string;
    colset Valor = string;
    colset Valores = list Valor;
    colset SP_t = Valores;
    colset EstBoolLista = product SP_t * BOOL;
    colset EstLoginLista = product SP_t *Policial;
(* Variables *)
    var aCIntSBD: CIntSBD;
    var aSBD: SBD;
    var param1: BOOL;
    var param2: BOOL;
    var param3: BOOL;
    var varBoolLista: EstBoolLista;
    var lista: Valores;
(* Functions *)
    fun mostrarOpcoes(obj) = obj
    fun opcoes(obj) = obj
    fun resposta(obj) = obj
    fun obterLista(estBoolLista: EstBoolLista) =
       #1 estBoolLista;
    fun validarDados(param1, estLista) =
      let val lista = obterLista(estLista)
      in
        if param1=true then
          (lista, true)
        else
          (lista, false)
      end;
    fun validar(param1, param2, SP_t) = if param1=true andalso param2=true then
      (SP_t, true) else (SP_t, false);
    fun obterListaLogin(loginLista: EstLoginLista) =
      #1 loginLista;
    fun validarVerificar(param1, param2, estLista): EstBoolLista =
      let val lista = obterLista(estLista)
      in
        if (param1 = true) andalso (param2 = true) then
          (lista, true)
        else
          (lista, false)
      end;
    fun gravarDados(param1, param2, param3) =
```

```
            if param1=true andalso param2=true andalso param3=true then
              true
            else
              false;
          fun mostrarFalha(obj) = obj
          fun falha(obj) = obj
(* Caso de Uso *)
    (* Colset *)
          colset uc = string;
    (* Variables *)
          var sp: Valores;
    (* Functions *)
          fun removerEscolha(elem: Valor, lst: Valores): Valores =
          case lst of
              [] => []
              | [x] => if (x = elem) then [] else lst
              | x::xs => if (x = elem) then xs else x :: removerEscolha(elem, xs);
          fun validar1(param1: bool, param2: bool, lista: Valores) = [if param1=true andalso param2=true tl
    (* Values *)
          val UC1 = "UC1";
          val UC2 = "UC2";
          val UC3 = "UC3";
          val ucList = [UC1, UC2, UC3];
(* Realizar Login *)
    (* Colset *)
          colset CCtrlLoginPol = bool;
          colset CIntPolLogin = bool;
    (* Variables *)
          var aCCtrlLoginPol: CCtrlLoginPol;
          var aCIntPolLogin: CIntPolLogin;
    (* Functions *)
          fun mostrarLogin(aCCtrlLoginPol) = aCCtrlLoginPol
          fun telaLogin(aCIntPolLogin) = aCIntPolLogin
          fun idSenha(aPolicial) = aPolicial
          fun respostaValidarPol(aCCtrlLoginPol) = aCCtrlLoginPol
          fun verificarPol(aCIntSBD) = aCIntSBD
          fun inserirLoginLista(login: Policial, estBoolLista) =
             let val lista = obterLista(estBoolLista)
             in
               (lista, login)
             end;
    (* Values *)
          val valCCtrlLoginPol = true: CCtrlLoginPol;
          val valCIntPolLogin = true: CIntPolLogin;
```

```
(* Registrar Relatorio *)
    (* Colset *)
        (* Policia estruturas *)
            colset descricao = string;
            (* Estruturas *)
                colset Viatura = product
                                                id*
                                                descricao;
                colset DelegaciaStruct = product
                                                 id*
                                                 descricao;
                colset Delegacia = DelegaciaStruct;
                colset Policiais = list PolicialLogin;
                colset Relatorio = product
                                               id*
                                               Policiais*
                                               Viatura*
                                               descricao;
        colset isOkConfirmacao = bool;
        colset PolicialDados = Relatorio;
        colset isOk = bool;
        colset CIntPolReg = bool;
        colset CCtrlRegRel = bool;
        colset CPolicial = PolicialLogin;
    (* Variaveis *)
        var aCIntSBDSalvarDados: CIntSBD;
        var dadosInseridos: BOOL;
        var loginValido: BOOL;
        var aDados: Relatorio;
        var aCIntPolReg: CIntPolReg;
        var aCCtrlRegRel: CCtrlRegRel;
        var aPolicial: Policial;
    (* Funcoes *)
        fun confirmacao1(aCIntPolReg) = aCIntPolReg
        fun confirmacao(aSBD) = aSBD
        fun salvarDados(aCIntSBDSalvarDados) = aCIntSBDSalvarDados
        fun getDados(aCIntSBD) = aCIntSBD
        fun gravarRegistro(aCCtrlRegRel) = aCCtrlRegRel
        fun setDados(aCIntPolReg) = aCIntPolReg
        fun dados(aDados) = aDados
        fun telaRegistro(aCIntPolReg) = aCIntPolReg
        fun mostrarTelaRegistro(aCCtrlRegRel) = aCCtrlRegRel
        fun opcao(aPolicial) = aPolicial
        fun respostaValidarPolSBD(aSBD) = aSBD
```

```
        fun mostrarConfirmacao(aCCtrlRegRel) = aCCtrlRegRel
    (* Values *)
        val valDelegacia = (1, "Delegacia 1"): Delegacia;
        val valViatura = (1, "Viatura 1"): Viatura;
        val valPolicial = (1, "Senha 1"): Policial;
        val valPoliciais = [(valPolicial)]: Policiais;
        val valDados = (1, valPoliciais, valViatura, "Relatorio 1"): PolicialDados;
        val valCIntPolReg =true: CIntPolReg;
        val valCCtrlRegRel = true: CCtrlRegRel;
        val valSBD = true: SBD;
        val valCIntSBD = true: CIntSBD;
(* Visualizar Relatorio *)
    (* Colset *)
        colset CCtrlVisRel = bool;
        colset CIntPolVis = bool;
        colset CCtrlRegVis = bool;
    (* Variables *)
        var aCCtrlVisRel: CCtrlVisRel;
        var aCIntPolVis: CIntPolVis;
        var aCCtrlRegVis: CCtrlRegVis;
        var periodoValidado: BOOL;
    (* Functions *)
        fun periodo(obj) = obj;
        fun telaPeriodo(obj) = obj;
        fun buscarPeriodo(obj) = obj;
        fun verificarPer(obj) = obj;
        fun mostrarResultado(obj) = obj;
    (* Values *)
        val valCCtrlVisRel = true: CCtrlVisRel;
        val valCIntPolVis =true: CIntPolVis;
```

O listing a seguir apresenta o conjunto de cores, variáveis e funções definidos na modelação do diagrama de de sequência de realizar login para a rede de Petri colorida.

Listing B.2: Código fonte rede de Petri colorida diagrama de sequência realizar login

```
(* Policia estruturas *)
    colset id = int;
    colset senha = string;
    (* Estruturas *)
        colset PolicialStruct = product
                                    id*
                                    senha;
(* Standard priorities *)
```

```
    val P_HIGH = 100;
    val P_NORMAL = 1000;
    val P_LOW = 10000;
(* Standard declarations *)
    colset UNIT = unit;
    colset BOOL = bool;
    colset INT = int;
    colset INTINF = intinf;
    colset TIME = time;
    colset REAL = real;
    colset STRING = string;
    (* Colset *)
        colset CIntSBD = bool;
        colset Policial = PolicialStruct;
        colset CCtrlLoginPol = bool;
        colset SBD = bool;
        colset CPolicial = unit;
        colset CIntPolLogin = bool;
(* Variables *)
    var loginValido: BOOL;
    var param1: BOOL;
    var param2: BOOL;
    var aSBD: SBD;
    var aCIntSBD: CIntSBD;
    var aPolicial: Policial;
    var aCIntPolLogin: CIntPolLogin;
    var aCCtrlLoginPol: CCtrlLoginPol;
(* Functions *)
    fun validar(param1, param2) = [if param1=true andalso param2=true then true else false];
    fun resposta(obj) = obj;
    fun respostaValidarPolSBD(obj) = obj;
    fun verificarPol(obj) = obj;
    fun respostaValidarPol(obj) = obj;
    fun idSenha(obj) = obj;
    fun telaLogin(obj) = obj;
    fun mostrarLogin(obj) = obj;
(* Values *)
    val valSBD = true: SBD;
    val valCIntSBD = true: CIntSBD;
    val valPolicial = (1, "Senha"): Policial;
    val valCCtrlLoginPol = true: CCtrlLoginPol;
    val valCIntPolLogin = true: CIntPolLogin;
```

O listing a seguir apresenta o conjunto de cores, variáveis e funções definidos na modelação do diagrama de de sequência de registrar relatório para a rede de Petri colorida.

Listing B.3: Código fonte rede de Petri colorida diagrama de sequência registrar relatório

```
(* Standard priorities *)
     val P_HIGH = 100;
     val P_NORMAL = 1000;
     val P_LOW = 10000;
(* Standard declarations *)
     colset UNIT = unit;
     colset BOOL = bool;
     colset INT = int;
     colset INTINF = intinf;
     colset TIME = time;
     colset REAL = real;
     colset STRING = string;
(* Common Declarations *)
     (* Policia estruturas *)
          colset id = int;
          colset senha = string;
          colset descricao = string;
          (* Estruturas *)
               colset Viatura = product
                                        id*
                                        descricao;
               colset DelegaciaStruct = product
                                         id*
                                         descricao;
               colset Delegacia = DelegaciaStruct;
               colset PolicialLogin = product
                                                 id*
                                                 senha;
               colset Policiais = list PolicialLogin;
               colset Relatorio = product
                                       id*
                                       Policiais*
                                       Viatura*
                                       descricao;
     (* Colset *)
          colset Policial = PolicialLogin;
          colset CIntSBD = bool;
          colset SBD = bool;
     (* Variables *)
```

```
        var login: BOOL;
        var aPolicial: Policial;
        var aCIntSBD: CIntSBD;
        var aSBD: SBD;
        var param1: BOOL;
        var param2: BOOL;
        var param3: BOOL;
        var opcoesExibidas: BOOL;
        var telaExibida: BOOL;
        var dadosValidados: BOOL;
        var registroGravado: BOOL;
    (* Functions *)
        fun validarDados(param1) = if param1=true then true else false;
        fun validar(param1, param2) =
           if param1=true andalso param2=true then true else false;
        fun gravarDados(param1, param2, param3) =
           if param1=true andalso param2=true andalso param3=true then true else false;
(* Realizar Login *)
    (* Colset *)
        colset CCtrlLoginPol = bool;
        colset CIntPolLogin = bool;
    (* Variables *)
        var aCCtrlLoginPol: CCtrlLoginPol;
        var aCIntPolLogin: CIntPolLogin;
    (* Functions *)
        fun mostrarLogin(obj) = obj;
        fun telaLogin(obj) = obj;
        fun idSenha(obj) = obj;
        fun respostaValidarPol(obj) = obj;
        fun verificarPol(obj) = obj;
        fun resposta(obj) = obj;
    (* Values *)
        val valCCtrlLoginPol = true: CCtrlLoginPol;
        val valCIntPolLogin = true: CIntPolLogin;
(* Registrar Relatorio *)
    (* Colset *)
        colset PolicialDados = Relatorio;
        colset teste = Relatorio;
        colset isOk = bool;
        colset isOkConfirmacao = bool;
        colset CIntPolReg = bool;
        colset CCtrlRegRel = bool;
        colset CPolicial = PolicialLogin;
    (* Variables *)
```

```
        var dadosInseridos: BOOL;
        var aCCtrlRegRel: CCtrlRegRel;
        var aCIntPolReg: CIntPolReg;
        var aCIntSBDSalvarDados: CIntSBD;
        var loginValido: BOOL;
        var aDados: PolicialDados;
    (* Functions *)
        fun mostrarOpcoes(obj) = obj;
        fun opcoes(obj) = obj;
        fun opcao(obj) = obj;
        fun mostrarTelaRegistro(obj) = obj;
        fun telaRegistro(obj) = obj;
        fun dados(obj) = obj;
        fun setDados(obj) = obj;
        fun gravarRegistro(obj) = obj;
        fun getDados(obj) = obj;
        fun salvarDados(obj) = obj;
        fun confirmacao(obj) = obj;
        fun mostrarConfirmacao(obj) = obj;
        fun mostrarFalha(obj) = obj;
        fun falha(obj) = obj;
    (* Values *)
        val valDelegacia = (1, "Delegacia 1"): Delegacia;
        val valViatura = (1, "Viatura 1"): Viatura;
        val valPolicial = (1, "Senha 1"): Policial;
        val valPoliciais = [(valPolicial)]: Policiais;
        val valDados = (1, valPoliciais, valViatura, "Relatorio 1"): Relatorio;
        val valCIntPolReg =true: CIntPolReg;
        val valCCtrlRegRel = true: CCtrlRegRel;
        val valSBD = true: SBD;
        val valCIntSBD = true: CIntSBD;
```

O listing a seguir apresenta o conjunto de cores, variáveis e funções definidos na modelação do diagrama de de sequência de visualizar relatório para a rede de Petri colorida.

Listing B.4: Código fonte rede de Petri colorida diagrama de sequência visualizar relatório

```
(* Standard priorities *)
    val P_HIGH = 100;
    val P_NORMAL = 1000;
    val P_LOW = 10000;
(* Standard declarations *)
    (* Estruturas *)
        colset id = int;
```

```
        colset senha = string;
        colset PolicialStruct = product
                                    id*
                                    senha;
    colset UNIT = unit;
    colset BOOL = bool;
    colset INT = int;
    colset INTINF = intinf;
    colset TIME = time;
    colset REAL = real;
    colset STRING = string;
    (* Declarations *)
        colset CIntPolVis = bool;
        colset CCtrlVisRel = bool;
        colset CIntSBD = bool;
        colset Policial = PolicialStruct;
        colset CCtrlLoginPol = bool;
        colset SBD = bool;
        colset CPolicial = unit;
        colset CIntPolLogin = bool;
(* Variaveis *)
    var visualizarRel: BOOL;
    var loginValido: BOOL;
    var param1: BOOL;
    var param2: BOOL;
    var aCIntPolVis: CIntPolVis;
    var aCCtrlVisRel: CCtrlVisRel;
    var aSBD: SBD;
    var aCIntSBD: CIntSBD;
    var aPolicial: Policial;
    var aCIntPolLogin: CIntPolLogin;
    var aCCtrlLoginPol: CCtrlLoginPol;
    var periodoSelecionado: BOOL;
    var opcoesExibidas: BOOL;
(* Funcoes *)
    fun validar(param1, param2) = [if param1=true andalso param2=true then true else false];
    fun falha(aCIntPolVis) = aCIntPolVis
    fun mostrarFalha(aCCtrlVisRel) = aCCtrlVisRel
    fun resultado(aCIntPolVis) = aCIntPolVis
    fun mostrarResultado(aCCtrlVisRel) = aCCtrlVisRel
    fun salvarDados(aCIntSBD) = aCIntSBD
    fun verificarPer(aCIntSBD) = aCIntSBD
    fun buscarPeriodo(aCCtrlVisRel) = aCCtrlVisRel
    fun setDados(aCIntPolReg) = aCIntPolReg
```

```
fun periodo1(aPolicial) = aPolicial
fun telaPeriodo(aCIntPolVis) = aCIntPolVis
fun periodo(aCCtrlVisRel) = aCCtrlVisRel
fun opcao(aPolicial) = aPolicial
fun opcoes(aCIntPolVis) = aCIntPolVis
fun mostrarOpcoes(aCCtrlVisRel) = aCCtrlVisRel
fun resposta(aSBD) = aSBD
fun respostaValidarPolSBD(aSBD) = aSBD
fun verificarPol(aCIntSBD) = aCIntSBD
fun respostaValidarPol(aCCtrlLoginPol) = aCCtrlLoginPol
fun idSenha(aPolicial) = aPolicial
fun telaLogin(aCIntPolLogin) = aCIntPolLogin
fun mostrarLogin(aCCtrlLoginPol) = aCCtrlLoginPol
```

O listing a seguir apresenta o relatório do grafo de alcançabilidade da rede colorida do diagrama de casos de uso gerado a partir do CPN Tools.

Listing B.5: Relatório grafo de alcançabilidade diagrama de casos de uso

```
Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  21
     Arcs:   20
     Secs:   0
     Status: Partial

  Scc Graph
     Nodes:  21
     Arcs:   20
     Secs:   0


Boundedness Properties
------------------------------------------------------------------------

  Best Integer Bounds
                                Upper       Lower
     AltIsNotOkRegRel'P49 1     1           1
     AltIsNotOkRegRel'P50 1     1           1
     AltIsNotOkVisRel'P88 1     1           1
     AltIsNotOkVisRel'P89 1     1           1
     AltIsOkRegRel'P45 1        1           1
```

```
AltIsOkRegRel'P46 1      1          1
AltIsOkVisVisRel'P84 1   1          1
AltIsOkVisVisRel'P85 1   1          1
Caso_de_Uso'SP0 1        1          0
Caso_de_Uso'SP1 1        1          0
Caso_de_Uso'SP2 1        1          0
Caso_de_Uso'SP3 1        1          0
Caso_de_Uso'SP_Feito 1   1          0
LoginRegRel'P13 1        1          1
LoginRegRel'P14 1        1          1
LoginRegRel'P15 1        1          0
LoginRegRel'P16 1        1          1
LoginRegRel'P17 1        1          0
LoginRegRel'P18 1        1          1
LoginRegRel'P19 1        1          1
LoginRegRel'P20 1        1          0
LoginRegRel'P21 1        1          1
LoginVisRel'P55 1        1          1
LoginVisRel'P56 1        1          1
LoginVisRel'P57 1        1          0
LoginVisRel'P58 1        1          1
LoginVisRel'P59 1        1          0
LoginVisRel'P60 1        1          1
LoginVisRel'P61 1        1          1
LoginVisRel'P62 1        1          0
LoginVisRel'P63 1        1          1
Realizar_Login'P1 1      1          1
Realizar_Login'P2 1      1          1
Realizar_Login'P3 1      1          0
Realizar_Login'P4 1      1          1
Realizar_Login'P5 1      1          0
Realizar_Login'P6 1      1          1
Realizar_Login'P7 1      1          1
Realizar_Login'P8 1      1          0
Realizar_Login'P9 1      1          1
Registrar_Relatorio'P23 1
                         1          0
Registrar_Relatorio'P43 1
                         0          0
Visualizar_Relatorio'P65 1
                         1          0
Visualizar_Relatorio'P82 1
                         0          0
registrarRelatorioDados'P25 1
```

```
                          1          1
registrarRelatorioDados'P26 1
                          1          1
registrarRelatorioDados'P27 1
                          1          0
registrarRelatorioDados'P28 1
                          1          1
registrarRelatorioDados'P29 1
                          1          0
registrarRelatorioDados'P30 1
                          1          1
registrarRelatorioDados'P31 1
                          1          1
registrarRelatorioDados'P32 1
                          0          0
registrarRelatorioDados'P33 1
                          1          1
registrarRelatorioDados'P34 1
                          1          1
registrarRelatorioDados'P35 1
                          0          0
registrarRelatorioDados'P36 1
                          1          1
registrarRelatorioDados'P37 1
                          1          1
registrarRelatorioDados'P38 1
                          0          0
registrarRelatorioDados'P39 1
                          1          1
registrarRelatorioDados'P40 1
                          0          0
registrarRelatorioDados'P41 1
                          1          1
visualizarRelatorioDados'P67 1
                          1          1
visualizarRelatorioDados'P68 1
                          1          1
visualizarRelatorioDados'P69 1
                          1          0
visualizarRelatorioDados'P70 1
                          1          1
visualizarRelatorioDados'P71 1
                          1          0
visualizarRelatorioDados'P72 1
```

```
                          1          1
   visualizarRelatorioDados'P73 1
                          1          1
   visualizarRelatorioDados'P74 1
                          0          0
   visualizarRelatorioDados'P75 1
                          1          1
   visualizarRelatorioDados'P76 1
                          0          0
   visualizarRelatorioDados'P77 1
                          1          1
   visualizarRelatorioDados'P78 1
                          1          1
   visualizarRelatorioDados'P79 1
                          0          0
   visualizarRelatorioDados'P80 1
                          1          1

Best Upper Multi-set Bounds
   AltIsNotOkRegRel'P49 1
                       1`true
   AltIsNotOkRegRel'P50 1
                       1`true
   AltIsNotOkVisRel'P88 1
                       1`true
   AltIsNotOkVisRel'P89 1
                       1`true
   AltIsOkRegRel'P45 1 1`true
   AltIsOkRegRel'P46 1 1`true
   AltIsOkVisVisRel'P84 1
                       1`true
   AltIsOkVisVisRel'P85 1
                       1`true
   Caso_de_Uso'SP0 1   1`["UC1","UC2","UC3"]
   Caso_de_Uso'SP1 1   1`["UC2","UC3"]
   Caso_de_Uso'SP2 1   1`["UC1","UC3"]
   Caso_de_Uso'SP3 1   1`["UC1","UC2"]
   Caso_de_Uso'SP_Feito 1
                       1`["UC2","UC3"]
   LoginRegRel'P13 1   1`true
   LoginRegRel'P14 1   1`true
   LoginRegRel'P15 1   1`(["UC1","UC3"],true)
   LoginRegRel'P16 1   1`(1,"Senha 1")
   LoginRegRel'P17 1   1`(["UC1","UC3"],true)
```

```
LoginRegRel'P18 1    1`true
LoginRegRel'P19 1    1`true
LoginRegRel'P20 1    1`(["UC1","UC3"],true)
LoginRegRel'P21 1    1`true
LoginVisRel'P55 1    1`true
LoginVisRel'P56 1    1`true
LoginVisRel'P57 1    1`(["UC1","UC2"],true)
LoginVisRel'P58 1    1`(1,"Senha 1")
LoginVisRel'P59 1    1`(["UC1","UC2"],true)
LoginVisRel'P60 1    1`true
LoginVisRel'P61 1    1`true
LoginVisRel'P62 1    1`(["UC1","UC2"],true)
LoginVisRel'P63 1    1`true
Realizar_Login'P1 1  1`true
Realizar_Login'P2 1  1`true
Realizar_Login'P3 1  1`(["UC2","UC3"],true)
Realizar_Login'P4 1  1`(1,"Senha 1")
Realizar_Login'P5 1  1`(["UC2","UC3"],true)
Realizar_Login'P6 1  1`true
Realizar_Login'P7 1  1`true
Realizar_Login'P8 1  1`(["UC2","UC3"],true)
Realizar_Login'P9 1  1`true
Registrar_Relatorio'P23 1
                     1`(["UC1","UC3"],true)
Registrar_Relatorio'P43 1
                     empty
Visualizar_Relatorio'P65 1
                     1`(["UC1","UC2"],true)
Visualizar_Relatorio'P82 1
                     empty
registrarRelatorioDados'P25 1
                     1`true
registrarRelatorioDados'P26 1
                     1`true
registrarRelatorioDados'P27 1
                     1`(["UC1","UC3"],true)
registrarRelatorioDados'P28 1
                     1`(1,"Senha 1")
registrarRelatorioDados'P29 1
                     1`(["UC1","UC3"],true)
registrarRelatorioDados'P30 1
                     1`true
registrarRelatorioDados'P31 1
                     1`true
```

```
     registrarRelatorioDados'P32 1
                          empty
     registrarRelatorioDados'P33 1
                          1`(1,[(1,"Senha 1")],(1,"Viatura 1"),"Relatorio 1")
     registrarRelatorioDados'P34 1
                          1`true
     registrarRelatorioDados'P35 1
                          empty
     registrarRelatorioDados'P36 1
                          1`true
     registrarRelatorioDados'P37 1
                          1`true
     registrarRelatorioDados'P38 1
                          empty
     registrarRelatorioDados'P39 1
                          1`true
     registrarRelatorioDados'P40 1
                          empty
     registrarRelatorioDados'P41 1
                          1`true
     visualizarRelatorioDados'P67 1
                          1`true
     visualizarRelatorioDados'P68 1
                          1`false++
1`true
     visualizarRelatorioDados'P69 1
                          1`(["UC1","UC2"],true)
     visualizarRelatorioDados'P70 1
                          1`(1,"Senha 1")
     visualizarRelatorioDados'P71 1
                          1`(["UC1","UC2"],true)
     visualizarRelatorioDados'P72 1
                          1`true
     visualizarRelatorioDados'P73 1
                          1`true
     visualizarRelatorioDados'P74 1
                          empty
     visualizarRelatorioDados'P75 1
                          1`(1,"Senha 1")
     visualizarRelatorioDados'P76 1
                          empty
     visualizarRelatorioDados'P77 1
                          1`true
     visualizarRelatorioDados'P78 1
```

```
                        1`true
    visualizarRelatorioDados'P79 1
                        empty
    visualizarRelatorioDados'P80 1
                        1`true

Best Lower Multi-set Bounds
    AltIsNotOkRegRel'P49 1
                        1`true
    AltIsNotOkRegRel'P50 1
                        1`true
    AltIsNotOkVisRel'P88 1
                        1`true
    AltIsNotOkVisRel'P89 1
                        1`true
    AltIsOkRegRel'P45 1 1`true
    AltIsOkRegRel'P46 1 1`true
    AltIsOkVisVisRel'P84 1
                        1`true
    AltIsOkVisVisRel'P85 1
                        1`true
    Caso_de_Uso'SP0 1   empty
    Caso_de_Uso'SP1 1   empty
    Caso_de_Uso'SP2 1   empty
    Caso_de_Uso'SP3 1   empty
    Caso_de_Uso'SP_Feito 1
                        empty
    LoginRegRel'P13 1   1`true
    LoginRegRel'P14 1   1`true
    LoginRegRel'P15 1   empty
    LoginRegRel'P16 1   1`(1,"Senha 1")
    LoginRegRel'P17 1   empty
    LoginRegRel'P18 1   1`true
    LoginRegRel'P19 1   1`true
    LoginRegRel'P20 1   empty
    LoginRegRel'P21 1   1`true
    LoginVisRel'P55 1   1`true
    LoginVisRel'P56 1   1`true
    LoginVisRel'P57 1   empty
    LoginVisRel'P58 1   1`(1,"Senha 1")
    LoginVisRel'P59 1   empty
    LoginVisRel'P60 1   1`true
    LoginVisRel'P61 1   1`true
    LoginVisRel'P62 1   empty
```

```
LoginVisRel'P63 1    1`true
Realizar_Login'P1 1  1`true
Realizar_Login'P2 1  1`true
Realizar_Login'P3 1  empty
Realizar_Login'P4 1  1`(1,"Senha 1")
Realizar_Login'P5 1  empty
Realizar_Login'P6 1  1`true
Realizar_Login'P7 1  1`true
Realizar_Login'P8 1  empty
Realizar_Login'P9 1  1`true
Registrar_Relatorio'P23 1
                     empty
Registrar_Relatorio'P43 1
                     empty
Visualizar_Relatorio'P65 1
                     empty
Visualizar_Relatorio'P82 1
                     empty
registrarRelatorioDados'P25 1
                     1`true
registrarRelatorioDados'P26 1
                     1`true
registrarRelatorioDados'P27 1
                     empty
registrarRelatorioDados'P28 1
                     1`(1,"Senha 1")
registrarRelatorioDados'P29 1
                     empty
registrarRelatorioDados'P30 1
                     1`true
registrarRelatorioDados'P31 1
                     1`true
registrarRelatorioDados'P32 1
                     empty
registrarRelatorioDados'P33 1
                     1`(1,[(1,"Senha 1")],(1,"Viatura 1"),"Relatorio 1")
registrarRelatorioDados'P34 1
                     1`true
registrarRelatorioDados'P35 1
                     empty
registrarRelatorioDados'P36 1
                     1`true
registrarRelatorioDados'P37 1
                     1`true
```

```
registrarRelatorioDados'P38 1
                    empty
registrarRelatorioDados'P39 1
                    1`true
registrarRelatorioDados'P40 1
                    empty
registrarRelatorioDados'P41 1
                    1`true
visualizarRelatorioDados'P67 1
                    1`true
visualizarRelatorioDados'P68 1
                    empty
visualizarRelatorioDados'P69 1
                    empty
visualizarRelatorioDados'P70 1
                    1`(1,"Senha 1")
visualizarRelatorioDados'P71 1
                    empty
visualizarRelatorioDados'P72 1
                    1`true
visualizarRelatorioDados'P73 1
                    1`true
visualizarRelatorioDados'P74 1
                    empty
visualizarRelatorioDados'P75 1
                    1`(1,"Senha 1")
visualizarRelatorioDados'P76 1
                    empty
visualizarRelatorioDados'P77 1
                    1`true
visualizarRelatorioDados'P78 1
                    1`true
visualizarRelatorioDados'P79 1
                    empty
visualizarRelatorioDados'P80 1
                    1`true
```

O listing a seguir apresenta o relatório do grafo de alcançabilidade da rede colorida do diagrama de sequência de "Realizar Login" gerado a partir do CPN Tools.

Listing B.6: Relatório grafo de alcançabilidade diagrama de sequência "Realizar Login"

```
CPN Tools state space report for:
```

/cygdrive/C/Users/gabri/OneDrive/Documentos/Larissa/sd-realizar-login-new-final-grafo.cpn
Report generated: Thu May 11 08:06:39 2023

Statistics

State Space

Nodes: 5
Arcs: 4
Secs: 0
Status: Full

Scc Graph

Nodes: 5
Arcs: 4
Secs: 0

Boundedness Properties

Best Integer Bounds

	Upper	Lower
login'P0 1	1	0
login'P10 1	1	0
login'P1 1	1	1
login'P2 1	1	1
login'P3 1	1	0
login'P4 1	1	1
login'P5 1	1	0
login'P6 1	1	1
login'P7 1	1	1
login'P8 1	1	0
login'P9 1	1	1

Best Upper Multi-set Bounds

login'P0 1	1`()
login'P10 1	1`true
login'P1 1	1`true
login'P2 1	1`true
login'P3 1	1`true
login'P4 1	1`(1,"Senha")
login'P5 1	1`(1,"Senha")

```
    login'P6 1          1`true
    login'P7 1          1`true
    login'P8 1          1`true
    login'P9 1          1`true

  Best Lower Multi-set Bounds
    login'P0 1          empty
    login'P10 1         empty
    login'P1 1          1`true
    login'P2 1          1`true
    login'P3 1          empty
    login'P4 1          1`(1,"Senha")
    login'P5 1          empty
    login'P6 1          1`true
    login'P7 1          1`true
    login'P8 1          empty
    login'P9 1          1`true


Home Properties
--------------------------------------------------------------------------------

  Home Markings
    [5]


Liveness Properties
--------------------------------------------------------------------------------

  Dead Markings
    [5]

  Dead Transition Instances
    None

  Live Transition Instances
    None


Fairness Properties
--------------------------------------------------------------------------------

    No infinite occurrence sequences.
```

O listing a seguir apresenta o relatório do grafo de alcançabilidade da rede colorida do

diagrama de sequência de "Registrar Relatório" gerado a partir do CPN Tools.

Listing B.7: Relatório grafo de alcançabilidade diagrama de sequência "Registrar Relatório"

```
CPN Tools state space report for:
/cygdrive/C/Users/gabri/OneDrive/Documentos/Larissa/sd-registrar-relatorio-new-final-grafo.cpn
Report generated: Thu May 11 08:07:34 2023


 Statistics
------------------------------------------------------------------------


  State Space
     Nodes:  13
     Arcs:   12
     Secs:   0
     Status: Full

  Scc Graph
     Nodes:  13
     Arcs:   12
     Secs:   0


 Boundedness Properties
------------------------------------------------------------------------


  Best Integer Bounds
                          Upper       Lower
     AltIsNotOk'P38 1     1           1
     AltIsNotOk'P39 1     1           1
     AltIsOk'P34 1        1           1
     AltIsOk'P35 1        1           1
     Login'P10 1          1           1
     Login'P2 1           1           1
     Login'P3 1           1           1
     Login'P4 1           1           0
     Login'P5 1           1           1
     Login'P6 1           1           0
     Login'P7 1           1           1
     Login'P8 1           1           1
     Login'P9 1           1           0
```

```
   registrarRelatorio'P0 1 1          0
   registrarRelatorio'P12 1
                           1          0
   registrarRelatorio'P32 1
                           1          0
   registrarRelatorio'P41 1
                           1          0
   registrarRelatorioDados'P14 1
                           1          1
   registrarRelatorioDados'P15 1
                           1          1
   registrarRelatorioDados'P16 1
                           1          0
   registrarRelatorioDados'P17 1
                           1          1
   registrarRelatorioDados'P18 1
                           1          0
   registrarRelatorioDados'P19 1
                           1          1
   registrarRelatorioDados'P20 1
                           1          1
   registrarRelatorioDados'P21 1
                           1          0
   registrarRelatorioDados'P22 1
                           1          1
   registrarRelatorioDados'P23 1
                           1          1
   registrarRelatorioDados'P24 1
                           1          0
   registrarRelatorioDados'P25 1
                           1          1
   registrarRelatorioDados'P26 1
                           1          1
   registrarRelatorioDados'P27 1
                           1          0
   registrarRelatorioDados'P28 1
                           1          1
   registrarRelatorioDados'P29 1
                           1          0
   registrarRelatorioDados'P30 1
                           1          1

Best Upper Multi-set Bounds
   AltIsNotOk'P38 1     1`true
```

```
AltIsNotOk'P39 1     1`true
AltIsOk'P34 1        1`true
AltIsOk'P35 1        1`true
Login'P10 1          1`true
Login'P2 1           1`true
Login'P3 1           1`true
Login'P4 1           1`true
Login'P5 1           1`(1,"Senha 1")
Login'P6 1           1`()
Login'P7 1           1`true
Login'P8 1           1`true
Login'P9 1           1`true
registrarRelatorio'P0 1
                     1`()
registrarRelatorio'P12 1
                     1`true
registrarRelatorio'P32 1
                     1`true
registrarRelatorio'P41 1
                     1`true
registrarRelatorioDados'P14 1
                     1`true
registrarRelatorioDados'P15 1
                     1`true
registrarRelatorioDados'P16 1
                     1`true
registrarRelatorioDados'P17 1
                     1`(1,"Senha 1")
registrarRelatorioDados'P18 1
                     1`()
registrarRelatorioDados'P19 1
                     1`true
registrarRelatorioDados'P20 1
                     1`true
registrarRelatorioDados'P21 1
                     1`true
registrarRelatorioDados'P22 1
                     1`(1,[(1,"Senha 1")],(1,"Viatura 1"),"Relatorio 1")
registrarRelatorioDados'P23 1
                     1`true
registrarRelatorioDados'P24 1
                     1`true
registrarRelatorioDados'P25 1
                     1`true
```

```
    registrarRelatorioDados'P26 1
                          1`true
    registrarRelatorioDados'P27 1
                          1`true
    registrarRelatorioDados'P28 1
                          1`true
    registrarRelatorioDados'P29 1
                          1`true
    registrarRelatorioDados'P30 1
                          1`true

Best Lower Multi-set Bounds
    AltIsNotOk'P38 1      1`true
    AltIsNotOk'P39 1      1`true
    AltIsOk'P34 1         1`true
    AltIsOk'P35 1         1`true
    Login'P10 1           1`true
    Login'P2 1            1`true
    Login'P3 1            1`true
    Login'P4 1            empty
    Login'P5 1            1`(1,"Senha 1")
    Login'P6 1            empty
    Login'P7 1            1`true
    Login'P8 1            1`true
    Login'P9 1            empty
    registrarRelatorio'P0 1
                          empty
    registrarRelatorio'P12 1
                          empty
    registrarRelatorio'P32 1
                          empty
    registrarRelatorio'P41 1
                          empty
    registrarRelatorioDados'P14 1
                          1`true
    registrarRelatorioDados'P15 1
                          1`true
    registrarRelatorioDados'P16 1
                          empty
    registrarRelatorioDados'P17 1
                          1`(1,"Senha 1")
    registrarRelatorioDados'P18 1
                          empty
    registrarRelatorioDados'P19 1
```

```
                        1`true
    registrarRelatorioDados'P20 1
                        1`true
    registrarRelatorioDados'P21 1
                        empty
    registrarRelatorioDados'P22 1
                        1`(1,[(1,"Senha 1")],(1,"Viatura 1"),"Relatorio 1")
    registrarRelatorioDados'P23 1
                        1`true
    registrarRelatorioDados'P24 1
                        empty
    registrarRelatorioDados'P25 1
                        1`true
    registrarRelatorioDados'P26 1
                        1`true
    registrarRelatorioDados'P27 1
                        empty
    registrarRelatorioDados'P28 1
                        1`true
    registrarRelatorioDados'P29 1
                        empty
    registrarRelatorioDados'P30 1
                        1`true
```

Home Properties

Home Markings

[13]

Liveness Properties

Dead Markings

[13]

Dead Transition Instances

AltIsNotOk'mostrarFalha_falha 1

Live Transition Instances

None

```
Fairness Properties
------------------------------------------------------------------------

    No infinite occurrence sequences.
```

O listing a seguir apresenta o relatório do grafo de alcançabilidade da rede colorida do diagrama de sequência de "Visualizar Relatório" gerado a partir do CPN Tools.

Listing B.8: Relatório grafo de alcançabilidade diagrama de sequência "Visualizar Relatório"

```
CPN Tools state space report for:
/cygdrive/C/Users/gabri/OneDrive/Documentos/Larissa/sd-visualizar-relatorio-new-final-grafo.cpn
Report generated: Thu May 11 08:08:23 2023


Statistics
------------------------------------------------------------------------

  State Space
    Nodes:  12
    Arcs:   11
    Secs:   0
    Status: Full

  Scc Graph
    Nodes:  12
    Arcs:   11
    Secs:   0


Boundedness Properties
------------------------------------------------------------------------

  Best Integer Bounds
                          Upper       Lower
    AltIsNotOk'P35 1      1           1
    AltIsNotOk'P36 1      1           1
    AltIsOk'P31 1         1           1
    AltIsOk'P32 1         1           1
    Login'P10 1           1           1
    Login'P2 1            1           1
    Login'P3 1            1           1
```

```
Login'P4 1                  1           0
Login'P5 1                  1           1
Login'P6 1                  1           0
Login'P7 1                  1           1
Login'P8 1                  1           1
Login'P9 1                  1           0
visualizarRelatorio'P0 1
                            1           0
visualizarRelatorio'P12 1
                            1           0
visualizarRelatorio'P29 1
                            1           0
visualizarRelatorio'P38 1
                            1           0
visualizarRelatorioPeriodo'P14 1
                            1           1
visualizarRelatorioPeriodo'P15 1
                            1           1
visualizarRelatorioPeriodo'P16 1
                            1           0
visualizarRelatorioPeriodo'P17 1
                            1           1
visualizarRelatorioPeriodo'P18 1
                            1           0
visualizarRelatorioPeriodo'P19 1
                            1           1
visualizarRelatorioPeriodo'P20 1
                            1           1
visualizarRelatorioPeriodo'P21 1
                            1           0
visualizarRelatorioPeriodo'P22 1
                            1           1
visualizarRelatorioPeriodo'P23 1
                            1           0
visualizarRelatorioPeriodo'P24 1
                            1           1
visualizarRelatorioPeriodo'P25 1
                            1           1
visualizarRelatorioPeriodo'P26 1
                            1           0
visualizarRelatorioPeriodo'P27 1
                            1           1

Best Upper Multi-set Bounds
```

```
AltIsNotOk'P35 1     1`true
AltIsNotOk'P36 1     1`true
AltIsOk'P31 1        1`true
AltIsOk'P32 1        1`true
Login'P10 1          1`true
Login'P2 1           1`true
Login'P3 1           1`true
Login'P4 1           1`true
Login'P5 1           1`(1,"Senha 1")
Login'P6 1           1`()
Login'P7 1           1`true
Login'P8 1           1`true
Login'P9 1           1`true
visualizarRelatorio'P0 1
                     1`()
visualizarRelatorio'P12 1
                     1`true
visualizarRelatorio'P29 1
                     1`true
visualizarRelatorio'P38 1
                     1`true
visualizarRelatorioPeriodo'P14 1
                     1`true
visualizarRelatorioPeriodo'P15 1
                     1`true
visualizarRelatorioPeriodo'P16 1
                     1`true
visualizarRelatorioPeriodo'P17 1
                     1`(1,"Senha 1")
visualizarRelatorioPeriodo'P18 1
                     1`()
visualizarRelatorioPeriodo'P19 1
                     1`true
visualizarRelatorioPeriodo'P20 1
                     1`true
visualizarRelatorioPeriodo'P21 1
                     1`true
visualizarRelatorioPeriodo'P22 1
                     1`(1,"Senha 1")
visualizarRelatorioPeriodo'P23 1
                     1`()
visualizarRelatorioPeriodo'P24 1
                     1`true
visualizarRelatorioPeriodo'P25 1
```

```
                        1`true
    visualizarRelatorioPeriodo'P26 1
                        1`true
    visualizarRelatorioPeriodo'P27 1
                        1`true

Best Lower Multi-set Bounds
    AltIsNotOk'P35 1    1`true
    AltIsNotOk'P36 1    1`true
    AltIsOk'P31 1       1`true
    AltIsOk'P32 1       1`true
    Login'P10 1         1`true
    Login'P2 1          1`true
    Login'P3 1          1`true
    Login'P4 1          empty
    Login'P5 1          1`(1,"Senha 1")
    Login'P6 1          empty
    Login'P7 1          1`true
    Login'P8 1          1`true
    Login'P9 1          empty
    visualizarRelatorio'P0 1
                        empty
    visualizarRelatorio'P12 1
                        empty
    visualizarRelatorio'P29 1
                        empty
    visualizarRelatorio'P38 1
                        empty
    visualizarRelatorioPeriodo'P14 1
                        1`true
    visualizarRelatorioPeriodo'P15 1
                        1`true
    visualizarRelatorioPeriodo'P16 1
                        empty
    visualizarRelatorioPeriodo'P17 1
                        1`(1,"Senha 1")
    visualizarRelatorioPeriodo'P18 1
                        empty
    visualizarRelatorioPeriodo'P19 1
                        1`true
    visualizarRelatorioPeriodo'P20 1
                        1`true
    visualizarRelatorioPeriodo'P21 1
                        empty
```

```
    visualizarRelatorioPeriodo'P22 1
                        1`(1,"Senha 1")
    visualizarRelatorioPeriodo'P23 1
                        empty
    visualizarRelatorioPeriodo'P24 1
                        1`true
    visualizarRelatorioPeriodo'P25 1
                        1`true
    visualizarRelatorioPeriodo'P26 1
                        empty
    visualizarRelatorioPeriodo'P27 1
                        1`true


Home Properties
--------------------------------------------------------------------------------

  Home Markings
    [12]


Liveness Properties
--------------------------------------------------------------------------------

  Dead Markings
    [12]

  Dead Transition Instances
    AltIsNotOk'mostrarFalha_falha 1

  Live Transition Instances
    None


Fairness Properties
--------------------------------------------------------------------------------

    No infinite occurrence sequences.
```