
Modelo de Simulação de Vídeo Games baseado em um Ambiente 3D e em Redes de Petri Coloridas

Felipe Nedopetalski



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2024

Felipe Nedopetalski

**Modelo de Simulação de Vídeo Games baseado
em um Ambiente 3D e em Redes de Petri
Coloridas**

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Stéphane Julia

Coorientador: Franciny Medeiros Barreto

Uberlândia

2024

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

N371m
2024 Nedopetalski, Felipe, 1998-
Modelo de Simulação de Vídeo Games baseado em um Ambiente 3D
e em Redes de Petri Coloridas [recurso eletrônico] / Felipe Nedopetalski.
- 2024.

Orientadora: Stéphane Julia.
Coorientadora: Franciny Medeiros Barreto.
Dissertação (Mestrado) - Universidade Federal de Uberlândia,
Programa de Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: <http://doi.org/10.14393/ufu.di.2025.5071>
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Julia, Stéphane (Orient.). II. Barreto, Franciny
Medeiros (Coorient.). III. Universidade Federal de Uberlândia. Programa
de Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

André Carlos Francisco
Bibliotecário-Documentalista - CRB-6/3408



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da
Computação

Av. João Naves de Ávila, 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG,
CEP 38400-902

Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br



ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Dissertação, 32/2024, PPGCO				
Data:	26 de agosto de 2024	Hora de início:	09:10	Hora de encerramento:	11:53
Matrícula do Discente:	12212CCP005				
Nome do Discente:	Felipe Nedopetalski				
Título do Trabalho:	Modelo de Simulação de Vídeo Games baseado em um Ambiente 3D e em Redes de Petri Coloridas				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Engenharia de Software				
Projeto de Pesquisa de vinculação:	Modelagem e Simulação de Video Games				

Reuniu-se por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Franciny Medeiros Barreto - Ciências da Computação - UFJ (Coorientadora), Márcia Aparecida Fernandes- FACOM/UFU, José Reinaldo Silva Eng. Mecatrônica e de Sistemas Mecânicos - Poli-USP e Stéphane Julia - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Franciny Medeiros Barreto - Jataí/GO e José Reinaldo Silva - São Paulo/SP. O aluno participou da cidade de Jataí/GO outros membros da banca participaram da cidade de Uberlândia.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Stéphane Julia, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir ao candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos,

conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **Jose Reinaldo Silva, Usuário Externo**, em 28/08/2024, às 12:43, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Franciny Medeiros Barreto, Usuário Externo**, em 28/08/2024, às 13:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Stéphane Julia, Professor(a) do Magistério Superior**, em 28/08/2024, às 16:00, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Márcia Aparecida Fernandes, Professor(a) do Magistério Superior**, em 29/08/2024, às 14:20, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5625467** e o código CRC **4E064C83**.

Este trabalho é dedicado à todas as pessoas que me auxiliaram durante toda a minha trajetória até aqui, principalmente à minha esposa, meus pais, professores e orientadores.

Agradecimentos

Agradeço a Capes pela oportunidade de cursar este mestrado com bolsa.

“Sua vida pode ser dividida em dois períodos: antes de agora e a partir de agora.”
(Prof. Obvious Stating)

Resumo

Este trabalho tem como objetivo apresentar uma abordagem para a modelagem e análise de cenários de vídeo games utilizando modelos de simulação 2D e 3D. Tanto jogos com um único jogador (*singleplayer*) quanto jogos com dois ou mais jogadores (*multiplayer*) são considerados nesta pesquisa. Pode-se considerar um cenário de vídeo game como as ações realizadas pelo jogador de forma sequencial. Para representar estas ações realizadas pelos jogadores, ferramentas que utilizam métodos formais como as redes de Petri coloridas, ou ferramentas que utilizam cenários 2D e 3D podem ser utilizadas. Para representar as áreas que os jogadores poderão acessar para realizar as atividades, um Mapa Topológico foi criado. Trabalhos anteriores ilustram como as ações e os locais do mapa podem se comunicar corretamente utilizando redes de Petri coloridas. Para este trabalho, além de explorar os modelos já existentes, foram apresentados modelos em 2D e 3D dos cenários de jogos *singleplayer* e *multiplayer* assim como seus resultados de simulações e comparações com os modelos formais feitos com redes de Petri coloridas. Para ilustrar as diversas abordagens apresentadas neste trabalho foi utilizado o primeiro nível do jogo *Silent Hill II* tanto utilizando redes de Petri coloridas quanto SIMIO nos casos *singleplayer* e *multiplayer*.

Palavras-chave: Redes de Petri. SIMIO. Simulação. Multiplayer. 3D.

Abstract

This work aims to present an approach for modeling and analysis of video game scenarios using 2D and 3D simulation models. Singleplayer and multiplayer games are considered in this research. A video game scenario can be considered as the actions performed by the player in a sequential manner. To represent these actions carried out by players, tools that use formal methods such as colored Petri nets, or tools that use 2D and 3D scenarios can be used. To represent the areas that players will be able to access to carry out activities, a Topological Map was created. Previous works have illustrated how actions and locations on the map can communicate correctly using colored Petri nets. For this work, in addition to exploring existing models, new models were presented in 2D and 3D of singleplayer and multiplayer game scenarios as well as their results simulations and comparisons with formal models made with colored Petri nets. To illustrate the different approaches presented in this work, the first level of the game Silent Hill II was used to create the models in colored Petri nets and SIMIO in the cases singleplayer and multiplayer.

Keywords: Petri nets. SIMIO. Simulation. Multiplayer. 3D.

Lista de ilustrações

Figura 1 – Exemplo de uma rede de Petri e seus elementos básicos.	30
Figura 2 – Exemplo de uma Rede de Petri Marcada	31
Figura 3 – Exemplo do disparo da transição T1.	31
Figura 4 – Elementos básicos de uma rede de Petri colorida.	35
Figura 5 – Disparo da transição T1 e a atribuição do valor a variável x	35
Figura 6 – Estado da rede após o disparo da transição T1.	36
Figura 7 – Rede de Petri colorida com adição de tempo.	38
Figura 8 – Disparo da transição T1 com tempo de 5 unidades.	38
Figura 9 – Disparo da transição T2 com tempo de 15 unidades no arco de saída. .	39
Figura 10 – Exemplo de um modelo utilizando <i>fusion places</i> no CPN Tools.	40
Figura 11 – Exemplo da execução do modelo com <i>fusion places</i> no CPN Tools. . . .	41
Figura 12 – <i>Workflow net</i> para o Processo de Tratamento de Reclamações apresen- tado por (AALST; HEE, 2004)	43
Figura 13 – Tokens coloridos no SIMIO	47
Figura 14 – Arquitetura de um objeto no SIMIO (FONTE: Mota et al. (2017)) . . .	48
Figura 15 – Definição das <i>stations</i> no SIMIO	48
Figura 16 – Definição das <i>stations</i> no SIMIO na aba <i>Facility View</i>	49
Figura 17 – Propriedades do <i>Decide Steps</i> no SIMIO.	50
Figura 18 – Exemplo de uma rede de Petri com sincronismo.	51
Figura 19 – Propriedades do <i>Decide Steps</i> no SIMIO.	51
Figura 20 – Processos em paralelo em uma CPN.	52
Figura 21 – Propriedades do <i>Decide Steps</i> associado ao objeto <i>Server</i> no SIMIO. .	52
Figura 22 – Exemplo de recurso em uma CPN.	53
Figura 23 – Step <i>Seize</i> no SIMIO.	54
Figura 24 – <i>Delay Step</i> no SIMIO.	54
Figura 25 – Ativadores de Processos no SIMIO.	55
Figura 26 – Exemplo de transição CPN.	56
Figura 27 – Exemplo de transição no SIMIO.	56

Figura 28 – Atribuição de valor na transição no SIMIO.	57
Figura 29 – Rede de Petri colorida do modelo lógico do primeiro nível do jogo Silent Hill II (Fonte: Barreto e Julia (2020b))	64
Figura 30 – Mapa parcial do primeiro nível de Silent Hill II (Fonte: Barreto e Julia (2020b))	65
Figura 31 – Rede de Petri colorida do modelo topológico do primeiro nível do jogo Silent Hill II (Fonte: Barreto e Julia (2020b))	67
Figura 32 – Exemplo da comunicação entre modelo Lógico e modelo Topológico (Fonte: Barreto e Julia (2020b))	68
Figura 33 – Comunicação entre modelo lógico e topológico utilizando o conceito de <i>fusion places</i>	68
Figura 34 – Rede de Petri Colorida do Modelo Lógico do primeiro nível do jogo Silent Hill II.	69
Figura 35 – Rede de Petri Colorida do Modelo Topológico do primeiro nível do jogo Silent Hill II.	70
Figura 36 – Resultados da análise das propriedades de limitabilidade e vivacidade .	72
Figura 37 – Replicações da simulação no modelo <i>singleplayer</i> com tempo.	72
Figura 38 – Exemplo de rede de Petri Colorida.	73
Figura 39 – Exemplo de modelo no SIMIO equivalente a uma rede de Petri Colorida.	74
Figura 40 – Processo <i>SortearCaso</i> no SIMIO.	74
Figura 41 – Processo <i>Process_Tempo_Tunel</i> no SIMIO.	75
Figura 42 – Modelo lógico 2D <i>singleplayer</i> feito utilizando SIMIO (FONTE: (NE-DOPETALSKI et al., 2024))	75
Figura 43 – Modelo Topológico 2D <i>singleplayer</i> feito utilizando SIMIO	76
Figura 44 – Configuração da variável de controle da área <i>Platarforma de Observação</i> .	76
Figura 45 – Modelo Lógico 3D <i>singleplayer</i> feito utilizando SIMIO	77
Figura 46 – Lista dos possíveis nós de saída do local “Igreja” no modelo topológico 3D.	77
Figura 47 – Propriedades do nó de saída do local “Igreja” no Modelo Topológico 3D.	78
Figura 48 – Modelo 3D <i>singleplayer</i> criado utilizando SIMIO	78
Figura 49 – Exemplo da Atividade inicial do Modelo Global no SIMIO.	79
Figura 50 – Exemplo da Atividade inicial do Modelo Global em rede de Petri Colorida.	79
Figura 51 – Área Floresta do Modelo Global no SIMIO.	79
Figura 52 – Área Igreja do Modelo Global no SIMIO.	80
Figura 53 – Área Jardim do Modelo Global no SIMIO.	80
Figura 54 – Área Floresta do Modelo Global em rede de Petri Colorida.	80
Figura 55 – Área Igreja do Modelo Global em rede de Petri Colorida.	81
Figura 56 – Área Quintal do Modelo Global em rede de Petri Colorida.	81
Figura 57 – Área Tunel do Modelo Global em rede de Petri Colorida.	82

Figura 58 – Área Tunel do Modelo Global em rede de Petri Colorida.	82
Figura 59 – Botão de simulação do SIMIO.	82
Figura 60 – Tabela de resultados da simulação realizada no <i>software</i> SIMIO no cenário <i>singleplayer</i>	83
Figura 61 – Resultado da simulação realizada no <i>software</i> SIMIO no cenário <i>singleplayer</i>	83
Figura 62 – Exemplo de um modelo de simulação global no caso multiplayer.	86
Figura 63 – Mecanismo de comunicação no Modelo Global Multiplayer.	87
Figura 64 – Estado da rede após o disparo da transição <i>Fim de A1</i>	88
Figura 65 – Estado da rede ao final da <i>quest</i>	89
Figura 66 – Modelo de rede de Petri Colorida global hierárquico Multiplayer do jogo <i>Silent Hill II</i>	90
Figura 67 – Modelo Lógico Multiplayer do primeiro nível do jogo Silent Hill II implementado no CPN Tools.	92
Figura 68 – Região incerta associada a um pseudo-disparo e representação da atividade “ <i>Find Map</i> ”	93
Figura 69 – Modelo Topológico Multiplayer do primeiro nível do jogo Silent Hill II implementado no CPN Tools.	95
Figura 70 – Lugares e transições adicionais no Modelo Topológico Multiplayer para mecanismo do pseudo-disparo.	96
Figura 71 – Replicações no modelo de rede de Petri colorida.	97
Figura 72 – Modelo 3D <i>multiplayer</i> feito utilizando SIMIO	97
Figura 73 – Atributo <i>Maximum Arrivals</i> do objeto <i>Source</i>	98
Figura 74 – Processo <i>Ctrl</i> responsável pela sequência de disparo das funções do pseudo-disparo.	99
Figura 75 – Processo <i>psDisp</i>	99
Figura 76 – Processo <i>rtDisp</i>	100
Figura 77 – Processo de cancelamento dos pseudo-disparos na área significativa <i>Apartamento</i>	101
Figura 78 – Pseudo-disparo da atividade <i>Encontrar mapa</i> no SIMIO.	101
Figura 79 – Pseudo-disparo na área significativa <i>Tunel</i> no SIMIO.	102
Figura 80 – Cancelamento do pseudo-disparo.	102
Figura 81 – Resultado das simulações das atividades hierárquicas multiplayer no SIMIO.	103

Lista de siglas

CPN - Coulored Petri Nets

ITNG - International Conference on Information Technology

UML - Unified Modelling Language

3D - Three Dimensional

Sumário

1	INTRODUÇÃO	21
1.1	Motivação	23
1.2	Objetivos e Desafios da Pesquisa	25
1.3	Contribuições	27
1.4	Organização da Dissertação	27
2	FUNDAMENTAÇÃO TEÓRICA	29
2.1	Redes de Petri	29
2.1.1	Propriedades das redes de Petri	32
2.2	Rede de Petri Colorida	33
2.2.1	Redes de Petri Temporizadas	36
2.2.2	Fusion Places	39
2.3	<i>Workflow nets</i>	41
2.3.1	<i>Soundness</i>	42
2.4	SIMIO	44
2.4.1	Objetos	45
2.4.2	Elementos úteis para implementação	46
2.4.3	Equivalência entre SIMIO e Redes de Petri	47
2.4.4	Eventos condicionais e Satisfação de restrições	49
2.4.5	Modelando Sincronismo	50
2.4.6	Modelando Paralelismo	51
2.4.7	Modelando recursos em processos	53
2.4.8	Consumo de tempo	54
2.4.9	Inserindo uma transição de rede de Petri no SIMIO	54
2.5	Considerações Finais	57
3	TRABALHOS CORRELATOS	59
3.1	Considerações Finais	60

4	MODELO <i>SINGLEPLAYER</i>	63
4.1	Modelo em rede de Petri	63
4.2	Modelo SIMIO	71
4.3	Considerações Finais	83
5	MODELO MULTIPLAYER	85
5.1	Mecanismo de Pseudo-Disparo	85
5.2	Modelo de rede de Petri colorida Multiplayer	89
5.3	Modelo 3D Multiplayer	96
5.3.1	Conversão do modelo CPN para o modelo 3D SIMIO	98
5.3.2	Resultados das simulações do modelo 3D	103
5.4	Considerações Finais	103
6	CONCLUSÃO	105
6.1	Principais Contribuições	105
6.2	Trabalhos Futuros	106
6.3	Produções Bibliográficas	106
	REFERÊNCIAS	107

CAPÍTULO 1

Introdução

Um jogo, ou vídeo game, pode ser considerado como um sistema fechado e formal que representa um subconjunto da realidade (CRAWFORD, 1982). Por fechado entende-se que o jogo possui um universo próprio com regras e leis que direcionam o jogador a alcançar o objetivo principal em um ambiente formal e controlado. Kanode Christopher M. e Haddad (2009) afirmam que o jogo se trata de uma síntese de código, imagens, música e atuação que se reúnem em forma de entretenimento. Pode-se dizer que os jogos são um tipo especial de aplicação multimídia que se trata de um produto de entretenimento que requer participação ativa do usuário (CALLELE; NEUFELD; SCHNEIDER, 2005). Lewis e Whitehead (2011) afirmam que o único aspecto original de um jogo que o diferencia do desenvolvimento de um software tradicional é a necessidade que ele seja divertido. No entanto, para (KANODE CHRISTOPHER M. E HADDAD, 2009), as aplicações de jogos se diferem de softwares tradicionais não apenas pelo uso e integrações de recursos multimídias, mas também na sua fase de pré-produção. Esta fase é um tipo de coleta de requisitos e criação de protótipos do jogo. Do ponto de vista clássico da indústria, o desenvolvimento de jogos é composto por duas principais etapas, que são denominadas *game design* e *level design* (GAL et al., 2002).

Na etapa de *game design* são definidos os principais aspectos do universo do jogo e cada detalhe de como ele funcionará (GAL et al., 2002). Nessa fase define-se o contexto do jogo (por exemplo época, estilo, referências históricas, etc.), o cenário global como a topologia, gráficos de navegação, principais personagens e as principais características do jogo que fazem ele se tornar único, com os seus princípios de jogabilidade, de ergonomia, assim como as imagens e sons que serão utilizados. Em síntese, na fase de *game design* é determinada a natureza do jogo.

Ainda de acordo com Gal et al. (2002), na etapa de *level design* todos os diferentes componentes do jogo se unem. Nesta fase é definida a descrição de como o jogador irá interagir com os objetos do jogo, a descrição das missões que o jogador executará, assim como o nível de dificuldade para a execução destas tarefas. São nessas duas etapas (*game* e *level design*) que são elaborados e descritos aspectos importantes e decisivos para a história

do jogo. Logo, essas duas etapas do processo de criação de jogos são fundamentais para o sucesso do projeto.

Um jogo possui um ou mais níveis (*levels*) com uma narrativa que fornece a dinâmica do jogo. Cada nível de jogo consiste em uma topologia, ou seja, o espaço no qual o jogador irá se movimentar, enigmas, ações principais e um conjunto de itens/objetos que irão interagir para o objetivo ser alcançado. Cada nível desse precisa ser significativo e é necessário que mantenha a coerência com o tema e a meta central do jogo. Com isso, é necessário levar em consideração a lógica de sequenciamento das ações relativas aos objetos, o posicionamento desses objetos no universo fechado do jogo, assim como as limitações do espaço virtual. Já a narrativa é uma sequência ordenada de eventos, no qual cada evento é um elemento atômico da história ou alguma coisa significativa que acontece (COLLÉ; CHAMPAGNAT; PRIGENT, 2005).

Além da narrativa, o cenário no qual o jogador se encontra também tem grande influência na motivação do jogador, pois pode proporcionar uma sensação de realidade. Um cenário é um conjunto de sequências ordenadas de eventos (COLLÉ; CHAMPAGNAT; PRIGENT, 2005). Em termos de *game design*, cada cenário é uma descrição clássica de uma sequência de ações das fases do jogo e como ocorre a navegação entre essas fases. Já em termos de *level design*, os cenários correspondem ao posicionamento dos objetos em relação às missões do jogo. Isso faz com que o jogador seja induzido a fazer uma sequência de ações parcialmente ordenadas, que precisam ser executadas pelo jogador para alcançar o final de um determinado nível (GAL et al., 2002).

Para Collé, Champagnat e Prigent (2005), os editores tendem a criar uma quantidade maior de jogos baseados em cenários, na esperança de deixá-los mais atrativos para o público alvo. Para Rollings e Morris (2003), é necessário criar desafios vinculados ao cenário que instiguem o jogador. Ao propor um desafio é preciso levar em consideração que diferentes tarefas podem mudar a topologia do espaço virtual de uma forma bastante complexa e, conseqüentemente, pode gerar erros de jogabilidade que precisam ser prevenidos (NATKIN; VEGA; GRÜNVOGEL, 2004).

Também é fundamental garantir que a experiência do jogo leve a uma sucessão de metas dentro de um prazo razoável. O jogador precisa ter a sensação de liberdade no mundo virtual, mesmo quando ele é orientado para uma solução de uma forma inconsciente (ROLLINGS; MORRIS, 2003). Cumprir todas as exigências para fazer com que o jogo seja atrativo não é uma tarefa trivial. Dessa forma, os projetos de jogos se tornam mais complexos.

Kanode Christopher M. e Haddad (2009) afirmam que a utilização de metodologias pobres é um dos maiores problemas do desenvolvimento de jogos. Segundo os autores, isso faz com que os projetos não consigam ser finalizados no tempo previsto, deixando os projetos mais caros. Segundo os autores, isso ocorre devido ao gerenciamento incorreto das atividades na fase de pré-produção do jogo (engenharia de requisitos, estimativas,

prototipagem, etc.). Lewis e Whitehead (2011) citam que as maiores falhas de desenvolvimento de jogos se encontram na etapa de levantamento de requisitos. Por exemplo, alguns requisitos não-funcionais, como entretenimento e imersão do usuário, criam exigências especiais no processo de engenharia de requisitos de um jogo, diferentes daquelas de projetos de software tradicionais (CALLELE; NEUFELD; SCHNEIDER, 2005). Logo, é necessário encontrar novas ferramentas e conceitos para atender aos desafios enfrentados no desenvolvimento de jogos (HORROCKS, 1999). Pensando nisso, a Engenharia de Software se torna imprescindível para alcançar os desafios do desenvolvimento de jogos, melhorando a gerência dos projetos, diminuindo os riscos e garantindo o sucesso futuro da indústria de jogos (KANODE CHRISTOPHER M. E HADDAD, 2009).

1.1 Motivação

A indústria dos jogos vem crescendo cada vez mais durante os últimos anos (CHUECA et al., 2023). Somente em 2021, teve uma receita de \$180.3 bilhões, superando as receitas das indústrias da música e do cinema de 2019. Além disso, os autores afirmam que a distância entre a indústria de jogos e a pesquisa está diminuindo cada vez mais (CHUECA et al., 2023).

Apesar da grande diferença entre o desenvolvimento de softwares convencionais e jogos (CHUECA et al., 2023), uma prática comum no processo de desenvolvimento é construir diagramas e gráficos para especificar visualmente conjuntos de requisitos. No contexto do desenvolvimento de jogos alguns autores como Ang e Rao (2004) e Rucker (2003) relatam o uso da UML (*Unified Modelling Language* - Linguagem de Modelagem Unificada) para mostrar como os diferentes objetos de um jogo irão interagir entre si de acordo com as ações realizadas pelo jogador dentro de um cenário.

A modelagem formal é outra abordagem utilizada para a descrição de cenários de jogos. Esses tipos de modelos são rigorosamente definidos e não contém especificações ambíguas (LEWIS; WHITEHEAD, 2011). É possível que, ao utilizar os modelos formais, sejam exploradas as passagens entre níveis do jogo e que sejam avaliadas certas propriedades em termos de jogabilidade, utilizando técnicas de simulação manuais ou automáticas. Neste contexto, alguns trabalhos exploram a utilização das redes de Petri como uma ferramenta eficiente para modelagem e análise de vídeo games, como por exemplo (BARRETO; JULIA, 2014; BARRETO; JULIA, 2020a).

As redes de Petri são consideradas como uma ferramenta gráfica e matemática de representação formal que permite modelagem, análise e controle de sistemas a eventos discretos que comportam atividades paralelas, concorrentes e assíncronas (MURATA, 1989). Araújo e Roque (2009) utilizaram diagramas baseados em redes de Petri para representar as possíveis ações dos jogadores em relação aos objetivos do jogo, sendo possível mapear o fluxo do jogo por meio de simulação. Além disso, os autores afirmam que ao simular o

comportamento do jogo é possível detectar problemas ainda na fase de projeto. Já Thomas, Yessad e Labat (2011) utilizaram as redes de Petri para modelar o comportamento de um jogador durante a execução do jogo e, a partir disso, acompanhar o progresso de aprendizado do jogador.

Já Natkin, Vega e Grünvogel (2004), apresentaram a modelagem de cenários de jogos baseada em uma rede de transações, que pode ser considerada como um tipo de rede de Petri. Ela permite modelar transações lógicas e temporais enquanto que um mapa topológico do jogo é modelado por meio de um tipo específico de grafo, os hiper-grafos. Os autores estabeleceram um mecanismo de comunicação entre os modelos para relacioná-los e definir a influência de um modelo sobre o outro.

Em Oliveira, Julia e Passos (2011), é utilizada uma abordagem baseada em Workflow net (AALST; HEE, 2004) para especificar os cenários existentes nos jogos. Nesta abordagem, as transições da rede representam as ações que o jogador executa para atingir um objetivo específico do jogo. Para validar os possíveis cenários que o jogador pode executar durante uma missão, os autores usaram o formalismo da Lógica Linear.

Redes de Petri também foram utilizadas para modelar atividades existentes em um nível de jogo, bem como o mapa topológico correspondente do mundo virtual no trabalho de Barreto e Julia (2014). Foi adicionado um mecanismo de comunicação entre os dois modelos (modelo de atividades e modelo de mapa topológico), formando um único modelo global. Este modelo foi analisado considerando algumas propriedades que garantem a consistência da execução correta das atividades e da disposição das diversas áreas existentes no mundo virtual do jogo. Em Barreto e Julia (2020a), os autores aplicaram os mesmos princípios para a modelagem apresentados em (BARRETO; JULIA, 2014) em um cenário com mais de um jogador (*multiplayer*). Segundo os autores, em muitos jogos *multiplayer*, as ações de cada jogador influenciam diretamente nas ações dos outros jogadores. Devido a este fato, o fluxo do jogo pode mudar, gerando um comportamento de incerteza.

Uma outra forma de visualizar o fluxo de um jogo é realizando a simulação em um ambiente com gráficos tridimensionais (3D). Nedopetalski et al. (2024) apresenta um cenário de vídeo game, modelado utilizando redes de Petri, num ambiente 3D. Para realizar a simulação no modelo 3D, foi utilizado o software SIMIO. Segundo Nedopetalski et al. (2024), as propriedades do modelo feito em redes de Petri são mantidas ao serem criadas no software SIMIO. Além disso, os resultados obtidos pelas simulações utilizando as redes de Petri e os resultados utilizando o software SIMIO foram similares. Esta abordagem foi aplicada utilizando um cenário de jogo *singleplayer*.

Com exceção do trabalho de Nedopetalski et al. (2024), todos os outros trabalhos citados acima limitaram-se à representação do fluxo de atividades utilizando ferramentas de modelagem formais ou diagramas, fazendo com que o entendimento das fases do jogo dependessem de uma notação gráfica e formal. Além disso, a maioria considera apenas a modelagem de jogos *singleplayer*, ou seja, com um único jogador. Como mostrado em

Barreto e Julia (2020a) e Araújo e Roque (2009), a complexidade em jogos *multiplayer* é maior, principalmente em atividades concorrentes e, apesar de ser possível a modelagem formal de jogos *multiplayer*, Reuter, Göbel e Steinmetz (2015) e Rezin et al. (2018) afirmam que a grande quantidade de possibilidades de caminhos que se formam com mais de um jogador ao mesmo tempo no jogo, ou seja, quando é detalhado o conjunto de atividades possíveis que um grupo de jogadores poderá realizar quando eles colaboram para alcançar uma mesma meta de jogo, pode levar a explosão de estados quando se considera a análise dos modelos.

Apesar deste problema, utilizar métodos formais como as redes de Petri para auxiliar no desenvolvimento do fluxo das atividades dos jogos, faz com que seja possível prever uma maior quantidade de erros na jogabilidade, como mostrado em (BARRETO; JULIA, 2014) e (BARRETO; JULIA, 2020a). Apesar disso, num modelo formal pode não ficar tão claro o significado da sequência de atividades do jogo. Para isso, Nedopetalski et al. (2024) transformaram uma sequência de atividades de um jogo modelado em redes de Petri em um modelo de simulação 3D. Desta forma, as atividades que serão executadas ficam claras, pois há uma representação no qual as pessoas envolvidas no desenvolvimento desta sequência de atividades podem visualizar o que acontece durante determinado nível do jogo. No entanto, não há desenvolvimento apenas de jogos *singleplayer*. Os jogos *multiplayers* também ocupam uma grande parcela de desenvolvimento de jogos. Além disso, uma ação de um jogador no jogo, pode influenciar diretamente as ações dos outros jogadores, o que dificulta ainda mais o entendimento da sequência de atividades que devem ser realizadas. Para isso, realizar o desenvolvimento de modelos de simulação 3D para jogos *multiplayer* é de suma importância para o entendimento do processo de desenvolvimento do jogo, fazendo com que reduza a quantidade de erros ainda durante o levantamento de requisitos do jogo.

1.2 Objetivos e Desafios da Pesquisa

Os jogos contêm uma parte lógica e outra parte topológica que atuam simultaneamente para representar as ações que os jogadores poderão executar e os locais em que essas ações acontecerão. É importante definir esses modelos durante o levantamento de requisitos e a elaboração de protótipos para que fique claro qual será a evolução dos jogadores de acordo com as atividades e as interações. Ao criar esses modelos separadamente, é possível analisá-los de forma independente. Outra vantagem é a possibilidade de alterá-los sem interferir diretamente no outro modelo. Após a definição dos modelos, é necessário fazer com que eles se comuniquem de forma harmoniosa, gerando assim um modelo com uma perspectiva global do jogo. Além disso, os modelos lógico e topológico devem ser preparados para receber mais de um jogador, uma vez que as atividades realizadas por um jogador podem influenciar nas atividades e locais disponíveis para um outro jogador.

Dessa forma, criar um modelo global *multiplayer* se faz necessário.

Ao criar uma fase de um jogo em um ambiente 3D, espera-se que a parte lógica se mescle com a parte topológica, uma vez que o jogador terá acesso ao mapa e precisará interagir e realizar as atividades necessárias para prosseguir no jogo. Realizar este processo na parte de *level design* é de suma importância para que testes sejam feitos antes do jogo entrar em fase de desenvolvimento. Dessa forma, análises podem ser feitas no modelo a fim de gerar informações que contribuam para o processo de desenvolvimento do jogo, resultando em um desenvolvimento mais eficiente, diminuindo o retrabalho. Sendo assim, criar modelos em ambiente 3D *multiplayer* para verificar os possíveis cenários, inserir e testar as atividades do jogo nos mapas se faz importante.

A compatibilidade entre ferramentas é essencial para que uma tradução fidedigna seja realizada com sucesso. Ao utilizar duas ou mais ferramentas, que buscam obter resultados semelhantes de formas diferentes, é necessário que sejam compatíveis e que haja possibilidade de realizar a tradução entre elas. Os modelos formais das redes de Petri trazem a confiabilidade do formalismo matemático que permite a implementação de importantes métodos de análises. Já os modelos 3D, expõem visualmente qual será o comportamento das atividades e interações feitas pelos jogadores ao realizar interações pelo mapa. Nesse sentido, é fundamental realizar a tradução adequada entre o modelo de rede de Petri e o modelo 3D.

Dentro deste contexto, a pesquisa descrita neste documento tem como objetivo apresentar modelos de simulação formal e 3D de cenários de jogos *singleplayer* e *multiplayer* utilizando para isso o formalismo das redes de Petri e também um ambiente de modelagem 3D. Para tanto, objetiva-se:

1. definir um modelo de simulação hierárquico *multiplayer* global, contendo um modelo lógico e outro topológico para representar as atividades e o mapa do cenário do jogo. Com isso, será possível analisar e realizar uma verificação formal do cenário utilizando o formalismo das redes de Petri;
2. reproduzir e amadurecer o modelo de simulação definido em um ambiente 3D para verificar a jogabilidade do cenário do jogo. O cenário hierárquico formal estabelecido irá conter os modelos lógico e topológico comunicando entre si e, portanto, simular o mesmo comportamento em um ambiente 3D e mostrar qual será o comportamento do jogador tanto do ponto de vista do sequenciamento das atividades previstas, quanto do espaço virtual onde tais atividades acontecerão;
3. definir um método de análise qualitativa (corretude e jogabilidade) e quantitativa (tempo de jogo) baseado nos modelos produzidos (modelos formais baseados em redes de Petri e modelos 3D). Será importante também verificar um certo nível de equivalência entre os resultados de análise produzidos pelos modelos formais baseados em redes de Petri e os modelos 3D.

1.3 Contribuições

A realização deste trabalho vai contribuir para a área de Engenharia de Software no âmbito do processo de desenvolvimento de jogos, em especial nas etapas iniciais do desenvolvimento que são denominadas *game* e *level design*, nas quais são definidos os requisitos e a arquitetura utilizada no sistema. A principal contribuição exposta neste trabalho é a criação de modelo de simulação 3D *singleplayer/multiplayer* baseado em um modelo formal de simulação, permitindo que os responsáveis pelo desenvolvimento do jogo em sua fase embrionária possam ter maior clareza de como o jogo será do ponto de vista dos jogadores. Este modelo, além de trazer clareza em relação à jogabilidade, também permitirá certas verificações de boas propriedades das atividades do jogo tanto qualitativamente quanto quantitativamente, no caso de um ou mais jogadores (*singleplayer* e *multiplayer*).

Baseado nos objetivos da pesquisa que foram apresentados, também são contribuições deste trabalho:

- ❑ a criação de um modelo de rede de Petri colorida *singleplayer* e *multiplayer* hierárquico utilizando a ferramenta de edição, simulação e análise de redes de Petri coloridas CPN Tools.
- ❑ regras de transformação do modelo de jogo baseado em redes de Petri coloridas num modelo visual 3D baseado no ambiente de simulação SIMIO.

1.4 Organização da Dissertação

Este documento se encontra dividido em 5 capítulos organizados da seguinte forma:

- ❑ no capítulo 2 são apresentados os conceitos básicos necessários para entender os modelos apresentados neste trabalho. Conceitos como redes de Petri e suas propriedades, Workflow nets, CPN Tools e SIMIO são apresentados neste capítulo;
- ❑ o capítulo 3 apresenta a literatura correlata deste trabalho. Neste capítulo são apresentadas as diferentes abordagens que foram apresentadas por outros autores, assim como uma discussão sobre as vantagens e limitações dos trabalhos realizados;
- ❑ o capítulo 4 apresenta os modelos de rede de Petri e 3D dos cenários de jogo *singleplayer* além dos resultados das simulações dos modelos. Neste capítulo será apresentado também os mecanismos de conversão do modelo de jogo baseado em redes de Petri coloridas para o modelo 3D baseado na ferramenta de simulação SIMIO;
- ❑ o capítulo 5 apresenta os modelos de rede de Petri e 3D dos cenários de jogo *multiplayer* além dos resultados das simulações dos modelos. Neste capítulo será apre-

sentado também os mecanismos de conversão do modelo de jogo baseado em redes de Petri coloridas para o modelo 3D baseado na ferramenta de simulação SIMIO;

- ❑ o capítulo 6, apresenta as considerações finais deste trabalho, as principais contribuições, trabalhos futuros e contribuições bibliográficas;
- ❑ por fim são listadas as referências bibliográficas utilizadas no decorrer deste trabalho.

Fundamentação Teórica

2.1 Redes de Petri

As redes de Petri têm origem na tese de Carl Adam Petri intitulada *Comunicação com Autômatos* apresentada em 1962 na Universidade de Darmsdadt. Apesar de ser considerada uma teoria relativamente jovem, ela se adapta bem a um grande número de aplicações em que as noções de eventos e de evoluções simultâneas são importantes (CARDOSO; VALETTE, 1997).

Redes de Petri são consideradas como uma ferramenta gráfica e matemática de representação formal que permite modelagem, análise e controle de sistemas a eventos discretos que comportam atividades paralelas, concorrentes e assíncronas (MURATA, 1989). Uma rede de Petri pode ser vista em particular como um grafo bipartido e direcionado com dois tipos de nós chamados de lugares e transições. Os nós são conectados por meio de arcos direcionados. Conexões entre dois nós do mesmo tipo não são permitidas. Dessa forma, os elementos básicos que permitem a definição das redes de Petri são:

- a) **Lugar:** em uma rede de Petri um lugar é representado por um círculo. Um lugar pode ser interpretado como uma condição, um estado parcial, um procedimento, a existência de um recurso, etc.
- b) **Transição:** uma transição é representada graficamente por uma barra ou retângulo. As transições são associadas a eventos que ocorrem em um sistema.
- c) **Ficha:** a ficha é representada por um ponto em um lugar. É um indicador significando que a condição associada ao lugar é verificada, como, por exemplo, um recurso disponível em um certo processo.

A Figura 1 ilustra um exemplo de uma rede de Petri e seus elementos básicos.

Formalmente, as redes de Petri são definidas da seguinte forma (CARDOSO; VALETTE, 1997):

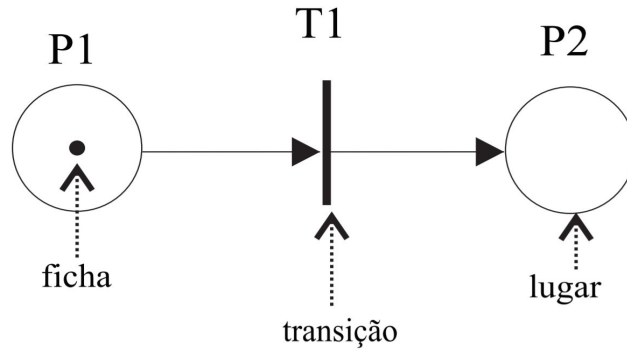


Figura 1 – Exemplo de uma rede de Petri e seus elementos básicos.

Definição 1 (Rede de Petri) Uma rede de Petri é uma quádrupla $R = \langle P, T, \text{Pré}, \text{Pós} \rangle$ onde:

- a) P é um conjunto finito de lugares de dimensão n ;
- b) T é um conjunto finito de transições de dimensão m ;
- c) $\text{Pré}: P \times T \rightarrow \mathbf{N}$ é a aplicação da entrada (lugares precedentes ou incidência anterior), com $\mathbf{N}$ sendo o conjunto dos números naturais;
- d) $\text{Post}: T \times P \rightarrow \mathbf{N}$ é a aplicação de saída (lugares seguintes ou incidência posterior).

Um lugar p é chamado lugar de entrada de uma transição t se existe um arco direcionado de p para t . Por exemplo, na Figura 1, o lugar $P1$ é lugar de entrada da transição $T1$. Já um lugar é chamado lugar de saída de uma transição se existe um arco direcionado de t para p . Ainda na Figura 1, o lugar $P2$ é um lugar de saída de $T1$.

No caso das redes de Petri marcadas, a definição formal para (CARDOSO; VALETTE, 1997) é a seguinte:

Definição 2 (Rede de Petri Marcada) Uma rede marcada N é uma dupla $N = \langle R, M \rangle$ onde:

- a) R é uma rede de Petri,
- b) M é a marcação inicial dada pela aplicação $M : P \rightarrow \mathbf{N}$.

$M(p)$ é então o conjunto de fichas contidas no lugar de p . A marcação M é a distribuição das fichas dos diversos lugares, sendo representada por um vetor coluna cuja dimensão é o número de lugares e os elementos são $M(p)$. Um exemplo de rede de Petri marcada é dado na Figura 2 onde a marcação é dada por $M^T = [1 \ 0 \ 3 \ 0 \ 1]$ (M^T é o valor transposto).

Formalmente, a evolução dinâmica de uma rede de Petri é dada pelas definições seguintes (CARDOSO; VALETTE, 1997):

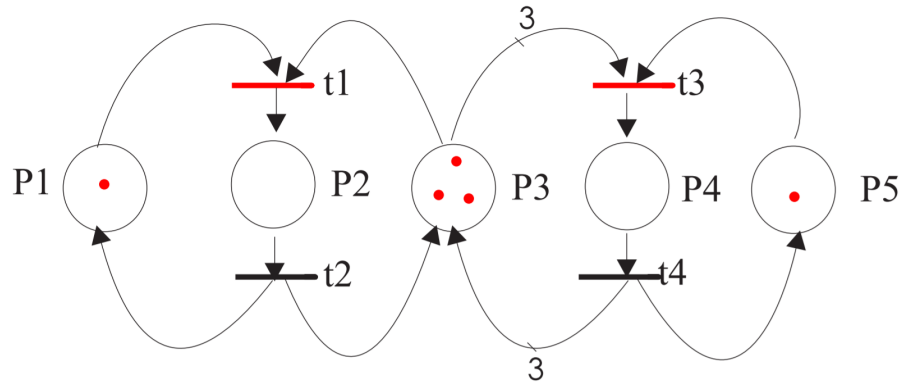


Figura 2 – Exemplo de uma Rede de Petri Marcada

Definição 3 Uma transição t está sensibilizada ou habilitada se e somente se:

$$\forall p \in P, M(p) \geq \text{Pre}(p, t) \quad (1)$$

Isto é, uma transição está sensibilizada se o número de fichas em cada um dos seus lugares de entrada for maior ou igual ao peso do arco que liga este lugar à transição.

Definição 4 Se t está sensibilizada por uma marcação M , uma nova marcação M' é obtida por meio do disparo de t de maneira que:

$$\forall p \in P, M'(p) = M(p) - \text{Pre}(p, t) + \text{Post}(p, t) \quad (2)$$

O disparo de uma transição t consiste então em retirar as fichas dos lugares de entrada ($\text{Pré}(p, t)$), e em depositar fichas em cada lugar de saída ($\text{Post}(p, t)$). Um exemplo de disparo de transição pode ser visto na Figura 3, em que a transição T1 tem como lugares de entrada P1 e P2. Os arcos de entrada de T1 tem peso 1, ou seja, é necessário uma ficha de cada lugar de entrada para que a transição seja disparada. como os arcos de entrada, os arcos de saída de T1 também possuem peso 1, gerando assim uma ficha em cada lugar de saída da transição.

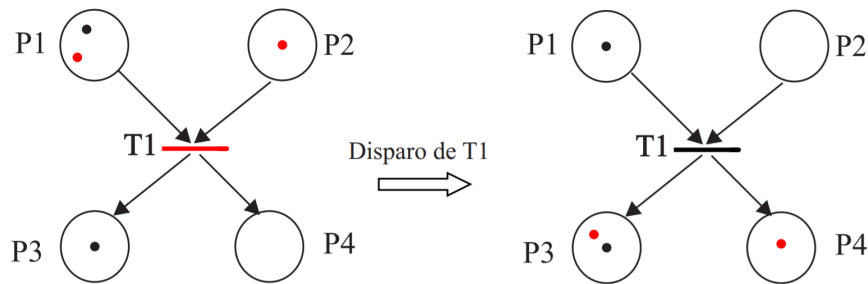


Figura 3 – Exemplo do disparo da transição T1.

Uma observação importante a ser feita é que as interpretações dos lugares e fichas podem ser bastante variadas. Podem ser usadas para descrever entidades abstratas como

condições ou estados, mas também entidades físicas como peças ou ambientes. Uma transição pode ser associada a um evento que ocorre no sistema, ou até mesmo a execução de uma operação com duração estritamente positiva (não nulas). Uma ficha pode representar um objeto ou um indicador. Neste sentido, as redes de Petri permitem uma visão interpretada e gráfica do sistema que está sendo modelado, além de permitir procedimentos formais de análise.

2.1.1 Propriedades das redes de Petri

Além das redes de Petri serem utilizadas para resolver problemas que tenham atividades concorrentes, elas podem ser utilizadas também para analisar e descrever um sistema (DAVID; ALLA, 2005). Para isso, foram desenvolvidos vários métodos que permitem a análise de diferentes propriedades de sistemas de eventos discretos ((MURATA, 1989) e (RADFORD, 1982)).

A dinâmica de um sistema modelado por uma rede de Petri se dá pela evolução das marcações. A partir da marcação inicial e do conjunto de marcações acessíveis, são definidas algumas propriedades. Murata (1989) define essas propriedades como: alcançabilidade, limitabilidade, reiniciabilidade e vivacidade. Na literatura, esse conjunto de propriedades é conhecido como boas propriedades.

- ❑ **Alcançabilidade** (*Reachability*): indica possibilidade de alcançar uma marcação específica. Uma marcação é considerada alcançável caso exista uma sequência finita de disparos que conduza ela a partir da marcação inicial. Essa propriedade verifica quais serão os estados que serão alcançados ou não.
- ❑ **Limitabilidade** (*Boundedness*): pode-se considerar uma rede de Petri limitada caso o número total de fichas nunca exceder um inteiro positivo k . Neste caso, a rede é considerada k -limitada. Se k for igual a 1, então pode-se dizer que a rede é binária.
- ❑ **Reiniciabilidade** (*Reversibility*): uma rede é considerada reiniciável se é sempre possível voltar para a marcação inicial por meio de uma sequência de disparos, considerando qualquer marcação.
- ❑ **Vivacidade** (*Liveness*): se todas as transições da rede de Petri forem vivas, então ela é considerada viva. Uma transição T é considerada viva se para toda marcação alcançável da rede, existe uma sequência de disparo que sensibiliza esta transição T . Portanto, uma rede de Petri é considerada viva caso todas as suas transições forem disparáveis. Com esta propriedade é possível garantir que a rede não contenha *deadlock*.

2.2 Rede de Petri Colorida

As fichas em redes de Petri podem representar objetos ou recursos em um sistema modelado (AALST, 1992). Esses objetos podem ter atributos que podem não ser facilmente representados por uma ficha ordinária de uma rede de Petri. A fim de resolver este problema, Jensen (1981) definiu o que hoje se conhece por redes de Petri Coloridas, ou *Colored Petri Nets* (CPN).

A CPN combina o formalismo da rede de Petri com uma linguagem de programação funcional para obter uma linguagem de modelagem para sistema concorrentes. As redes de Petri fornecem a base formal para modelar concorrência e sincronização. Já a linguagem de programação fornece as primitivas para manipulação de dados e criação de modelos compactos e parametrizáveis. Sendo assim, em CPN as fichas possuem valores que são conhecidos como cores. Essas cores não significam apenas cores ou padrões, elas podem representar tipos de dados complexos (JENSEN; KRISTENSEN, 2009). Desta forma, as CPNs foram projetadas para tornar mais compactos modelos de sistemas complexos, permitindo a individualização de fichas, usando tipos de dados. Assim, diferentes processos ou recursos podem ser representados em uma mesma estrutura gráfica.

Formalmente, as redes de Petri Coloridas podem ser definidas como se segue (JENSEN; KRISTENSEN, 2009):

Definição 5 (Rede de Petri Colorida) *Uma rede de Petri colorida não hierárquica (CPN) é uma tupla $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ onde:*

- P é o conjunto finito de lugares;
- T é o conjunto finito de transições T tal que $P \cap T = \emptyset$;
- $A \subseteq P \times T \cup T \times P$ é o conjunto de arcos direcionados;
- Σ é um conjunto finito de conjunto de cores (color sets) não vazios;
- V é um conjunto finito de variáveis tipadas tal que $Tipo[v] \in \Sigma$ para toda variável $v \in V$;
- $C: P \rightarrow \Sigma$ é a função do conjunto de cores que associa a cada lugar um conjunto de cor;
- $G: T \rightarrow EXPR_v$ é uma função de guarda (expressão lógica) que associa uma guarda a cada transição t tal que $Tipo[G(t)] = Bool$;
- $E: A \rightarrow EXPR_v$ é uma função de expressão de arco que associa uma expressão a cada arco tal que $Tipo[E(a)] = C(p)_{MS}$, onde p é o lugar conectado ao arco a ;
- $I: P \rightarrow EXPR$ é uma função de inicialização que atribui uma expressão de inicialização a cada lugar p tal que $Tipo[I(p)] = C(p)_{MS}$.

A aplicação prática da modelagem e análise das CPNs depende fortemente da existência de ferramentas computacionais que ofereçam suporte à criação e manipulação de tais modelos. O CPN Tools é software adequado para edição, simulação e para análise de espaço de estado (*state space*) dos modelos de redes de Petri Coloridas (JENSEN; KRISTENSEN, 2009).

Há várias razões para o uso de Redes de Petri Coloridas e CPN Tools:

- a) O uso de redes de Petri fornece uma análise para correção do modelo por meio de, por exemplo, uma análise do *state space*. É importante salientar que esta análise é realizada a partir do modelo de alto nível por meio da replicação de cenário e não do estudo do modelo de rede de Petri ordinário subjacente. Isso é então um tipo de análise de propriedade de um modelo de alto nível que é o equivalente a um programa produzido em uma linguagem de programação de alto nível.
- b) É viável fazer pequenas adaptações dos modelos e depois comparar os resultados, dadas várias propostas de modelagem.
- c) A integração de processos e de dados complexos é essencial para modelagem de processos complexos. O CPN Tools oferece a possibilidade de integrá-los em um único modelo.
- d) O CPN Tools contém um ambiente de simulação, em que se pode seguir passo-a-passo uma sequência de disparos. O *software* permite simulação do tipo Monte Carlo, onde replicações de um mesmo cenário aleatório e o cálculo de estimativas de valores médios que pertencem a intervalos de confiança, podem ser calculados automaticamente.

Os estados de uma rede de Petri colorida são representados pela marcação dos lugares. Cada lugar tem um tipo, que por sua vez poderá ter um número variado de fichas. Cada ficha tem um valor que é denominado pelas cores no contexto das rede de Petri coloridas (AALST; STAHL, 2011). Essas cores representam desde dados primitivos como números inteiros e *strings*, até tipos de dados complexos como produtos cartesianos ou estruturas de dados (JENSEN; KRISTENSEN, 2009). A Figura 4 ilustra um exemplo dos elementos básicos de uma rede de Petri colorida.

A rede de Petri colorida da Figura 4 possui dois lugares denominados *Place 1* e *Place 2*. Esses lugares possuem o tipo de dado (cor) *INTxSTRING*, assim como a variável *x* associada aos arcos. Este tipo de dado é formado pelo produto cartesiano do tipo *INT* (inteiro) e *STRING* (cadeia de caracteres). Os lugares da rede que são definidos com este tipo de dado só aceitam fichas do mesmo tipo. Neste caso, a ficha com a inscrição *1'(2024, "exemplo")* faz parte do tipo de dado aceito pelo lugar, uma vez que é composta por um inteiro (2024) e uma cadeia de caracteres ("*exemplo*").

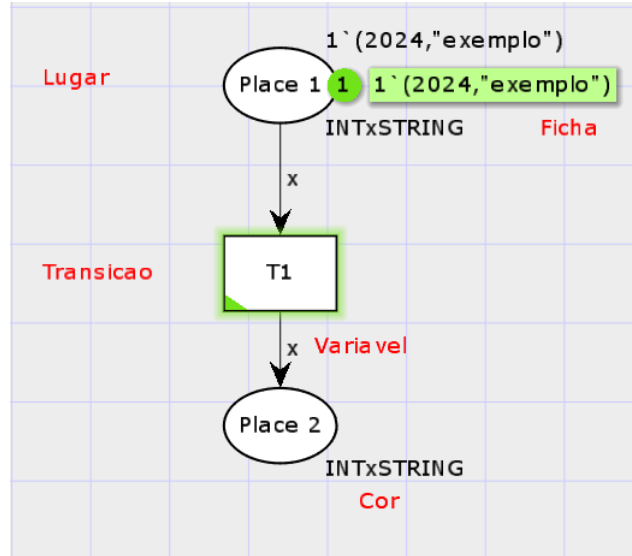


Figura 4 – Elementos básicos de uma rede de Petri colorida.

Para criar novas redes, a declaração de novas cores e variáveis se torna indispensável. O CPN Tools tem uma área gráfica, onde são construídas as redes e outra parte para a declaração de cores, variáveis, funções, entre outros. Foi necessário declarar três tipos de *cores* e uma variável para criar o modelo da Figura 4 da seguinte forma:

```

1 colset INT = int;
2 colset STRING = string;
3 colset INTxSTRING = product INT * STRING;
4 var x: INTxSTRING;

```

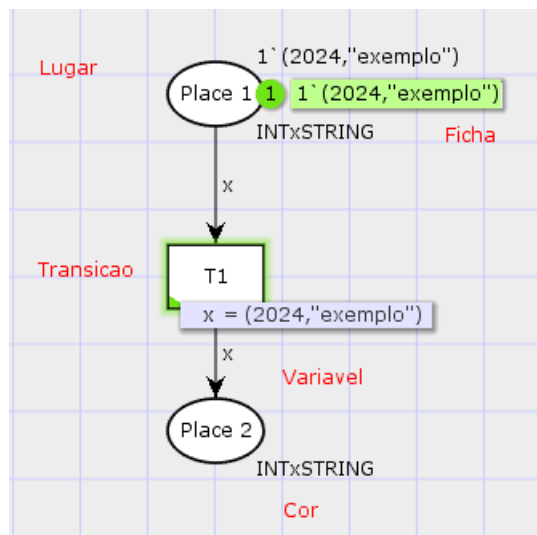


Figura 5 – Disparo da transição T1 e a atribuição do valor a variável x .

Assim como em uma rede de Petri clássica, na rede de Petri colorida uma transição é disparada retirando as fichas dos lugares de entrada e depositando nos lugares de saída da transição. Nos exemplos das Figuras 5 e 6 a transição T1 ficará habilitada somente se no seu lugar de entrada houver pelo menos uma ficha. No disparo de uma transição, as

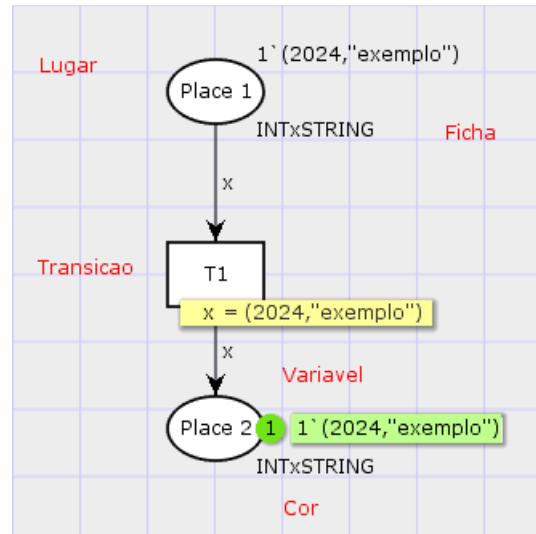


Figura 6 – Estado da rede após o disparo da transição T1.

variáveis dos arcos de entrada são substituídas pelo valor da ficha. É possível observar este comportamento nas Figuras 5 em que o valor da ficha de *Place 1* é associado à variável x e na Figura 6, uma ficha é produzida no lugar *Place 2*.

2.2.1 Redes de Petri Temporizadas

As redes de Petri clássicas não permitem a modelagem do tempo de forma explícita (durações de atividades e períodos de ocorrência de início e fim de operações), mas podem ser estendidas em relação ao tempo, já que informações temporais são úteis para sequenciar eventos, comparar suas durações, bem como determinar o intervalo existente entre elas. Desta forma, é possível representar e analisar problemas ligados a diversas atividades onde a avaliação do tempo é um critério importante. Vários trabalhos foram realizados no sentido de utilizar explicitamente o tempo como parâmetro contínuo e quantificável em redes de Petri, como apresentado em (DOTOLI; FANTI, 2006; WU; ZHOU, 2007; WANG; FANG; GUO, 2020).

Ao adicionar tempo nas redes de Petri, o indeterminismo com relação ao disparo de uma transição é reduzido, uma vez que esta informação adiciona uma nova relação entre as ordens de disparos das transições (FREITAS, 2017).

Um aspecto importante, seja qual for a forma de especificar o tempo na rede, é a forma como é interpretado. Este aspecto é denominado de semântica do tempo e, para as redes de Petri, existem basicamente duas semânticas distintas (CARDOSO; VALETTE, 1997):

- a) Tempo de Sensibilização: determina o tempo que deve transcorrer desde o momento da habilitação de uma transição até o momento do seu disparo. O disparo ocorre de forma atômica, ou seja, disparo instantâneo, indivisível.

- b) Tempo de Disparo: o disparo de uma transição ocorre em três fases, retirada da marca indisponível, disparo atômico e colocação da marca disponível.

Outro aspecto importante, em relação ao tempo, é a efetivação do disparo de uma transição sensibilizada (CARDOSO; VALETTE, 1997). Para alguns sistemas, o disparo de uma transição sensibilizada deve ocorrer em determinada data. Nestes casos, a semântica do tempo é denominada semântica forte, em oposição à semântica fraca, na qual o disparo de uma transição sensibilizada não é obrigatório; porém, se ocorrer, deve ser em uma determinada data também.

As primeiras redes de Petri com interferência do tempo foram apresentadas por Merlin (1974) e Ramchandani (1974). Desde então, diferentes modelos de redes de Petri com tempo tem sido propostos na literatura. Geralmente, os modelos se diferenciam em aspectos, tais como: tipo de temporização, localização da restrição temporal e propriedade da restrição (CERONE; MAGGIOLO-SCHETTINI, 1999). Independente desses aspectos, podem-se agrupar as extensões temporais das redes de Petri em quatro grandes categorias:

- a) Redes de Petri Temporais (*Time Petri Nets*): a restrição temporal é um intervalo de tempo associado a cada transição. Foram inicialmente usadas na descrição de protocolos de comunicação; entretanto, seu campo de aplicação tem se ampliado para áreas como a manufatura, sistemas em tempo real, validação e verificação de sistemas (BERTHOMIEU; MENASCHE, 1983; CERONE; MAGGIOLO-SCHETTINI, 1999; BOWDEN, 2000; BERARD et al., 2013).
- b) Redes de Petri Temporizadas (*Timed Petri Nets*): permitem que transições ou lugares retenham marcas durante um intervalo de tempo. São comumente usadas na análise de sistemas de manufatura (ZUBEREK, 1991; LAHAYE; KOMENDA; BOIMOND, 2014; DOTOLI et al., 2014).
- c) Marcas Temporizadas: a restrição temporal é associada à ficha ao se propagar na rede. São amplamente usadas em problemas de produção (CELKO, 1982; TSAI; YANG; CHANG, 1995).
- d) Modelos Probabilísticos: são as chamadas redes de Petri Estocásticas e são usadas para estimar o desempenho de grandes sistemas. Uma probabilidade é então associada com uma transição, lugar ou marca (MURATA, 1989).

Para um modelo de jogo, as redes de Petri com tempo são essenciais, uma vez que é importante ter noção do tempo necessário para realizar as atividades que devem ser seguidas para atingir o objetivo do jogo. Cada atividade tem um tempo médio para ser realizada, assim como a movimentação pelo espaço do mapa topológico possui um tempo médio esperado.

Nas redes de Petri coloridas é possível associar tempo aos modelos. Para tanto, usa-se o conceito de timestamp que é associado à ficha. (JENSEN; KRISTENSEN, 2009). Uma inscrição de tempo associada às transições ou aos arcos é utilizada para calcular o *timestamp* associado à ficha do modelo. Se a inscrição de tempo estiver associada a uma transição, todas as fichas geradas por aquela transição terão um tempo adicionado a elas. Caso a inscrição estiver associada a um arco, o tempo será adicionado nas fichas geradas naquele arco (JENSEN; KRISTENSEN, 2009).

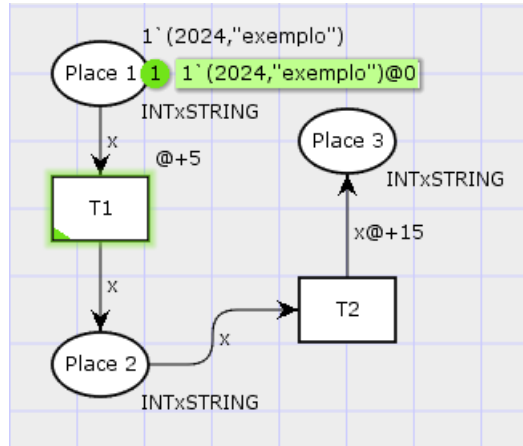


Figura 7 – Rede de Petri colorida com adição de tempo.

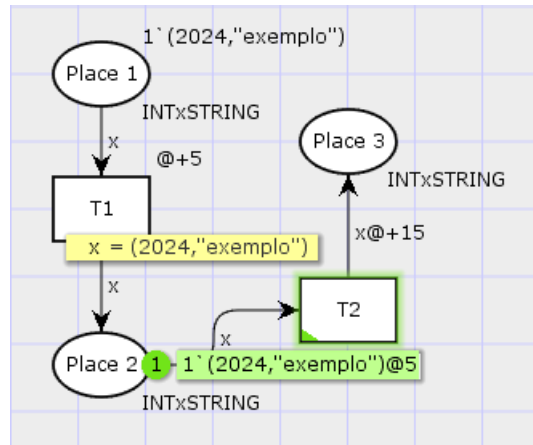


Figura 8 – Disparo da transição T1 com tempo de 5 unidades.

Acrescentar tempo em um modelo de rede de Petri colorida é um processo simples. Para isso, é preciso definir quais conjuntos de cores são temporizados no modelo. Essa definição é feita adicionando a palavra reservada *timed* na declaração do conjunto de cores. Desta forma, o conjunto de cor será temporizado e poderá receber os valores de tempo nas fichas. Para o exemplo utilizado nas Figuras 7, 8 e 9, as declarações das cores ficaram da seguinte forma:

```
1 colset INT = int;
2 colset STRING = string;
3 colset INTxSTRING = product INT * STRING timed;
```

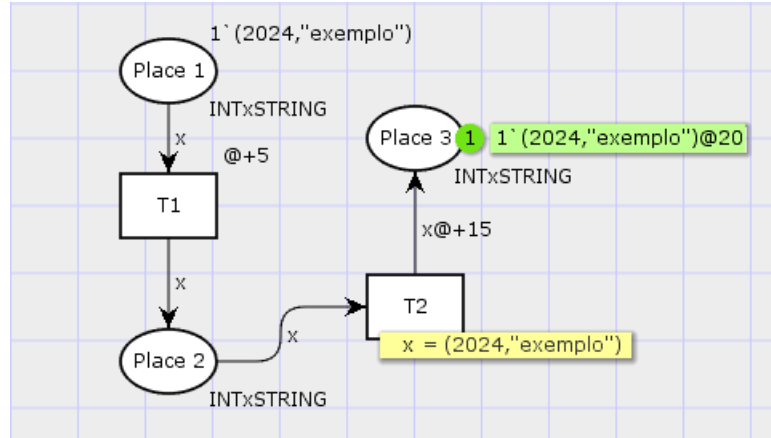


Figura 9 – Disparo da transição T2 com tempo de 15 unidades no arco de saída.

```
4 var x: INTxSTRING;
```

As Figuras 7, 8 e 9 exemplificam o funcionamento de uma rede de Petri colorida com adição de tempo. Na transição T1 foi adicionada a inscrição de tempo $@+5$, que significa que a operação correspondente a transição leva 5 unidades de tempo para ser concluída. Deste modo, toda vez que alguma ficha for disparada pela transição T1, será adicionado 5 unidades de tempo a esta ficha. Na Figura 7, por exemplo, o timestamp da ficha é $@0$. Após o disparo de T1, na Figura 8, o timestamp passa a ser 5. No caso do arco de saída da transição T2, representado na Figura 9, o arco possui uma inscrição de tempo adicional de $@+15$ unidades de tempo. Desta forma, será somada a quantidade especificada pela inscrição do arco, que neste caso é $@+15$, totalizando $@20$ unidades de tempo à ficha.

2.2.2 Fusion Places

A modelagem de redes de Petri coloridas utilizando a ferramenta CPN Tools se torna ainda mais interessante pela possibilidade de estruturar os modelos separadamente utilizando um mecanismo de estruturação hierárquica. Basicamente, a construção deste tipo de modelo permite construir modelos maiores combinando pequenas redes de Petri coloridas em um único modelo (JENSEN; KRISTENSEN, 2009). Este tipo de característica facilita a modelagem de sistemas complexos (AALST; STAHL, 2011).

Uma das ferramentas para a modelagem hierárquica é chamado *fusion places*. Ela permite identificar múltiplos lugares que são idênticos semanticamente, significando que todos esses lugares representam um único lugar conceitual, mesmo tendo vários desenhos dele na rede (KRISTENSEN; CHRISTENSEN; JENSEN, 1998). O conjunto de *fusion places* é chamado de *fusion set* e qualquer ação que acontecer em algum lugar deste conjunto, também acontecerá nos outros lugares deste mesmo conjunto. Desta forma, sempre que uma ficha é alocada ou consumida por um lugar, outra ficha idêntica vai ser adicionada ou consumida em todos os outros lugares que pertencem ao mesmo conjunto.

As Figuras 10 e 11 exemplificam o funcionamento dos *fusion places*. Nelas são representados dois modelos distintos que produzem pacotes. Cada um destes pacotes possuem um identificador. Ambos os modelos têm como objetivo montar os pacotes para serem entregues a uma única fila e gerar um pacote final único para ser enviado. O modelo à esquerda apenas monta os seus pacotes e despacha para a fila. Já o modelo à direita, além de montar seus pacotes, recebe os pacotes do primeiro modelo e prepara o envio.

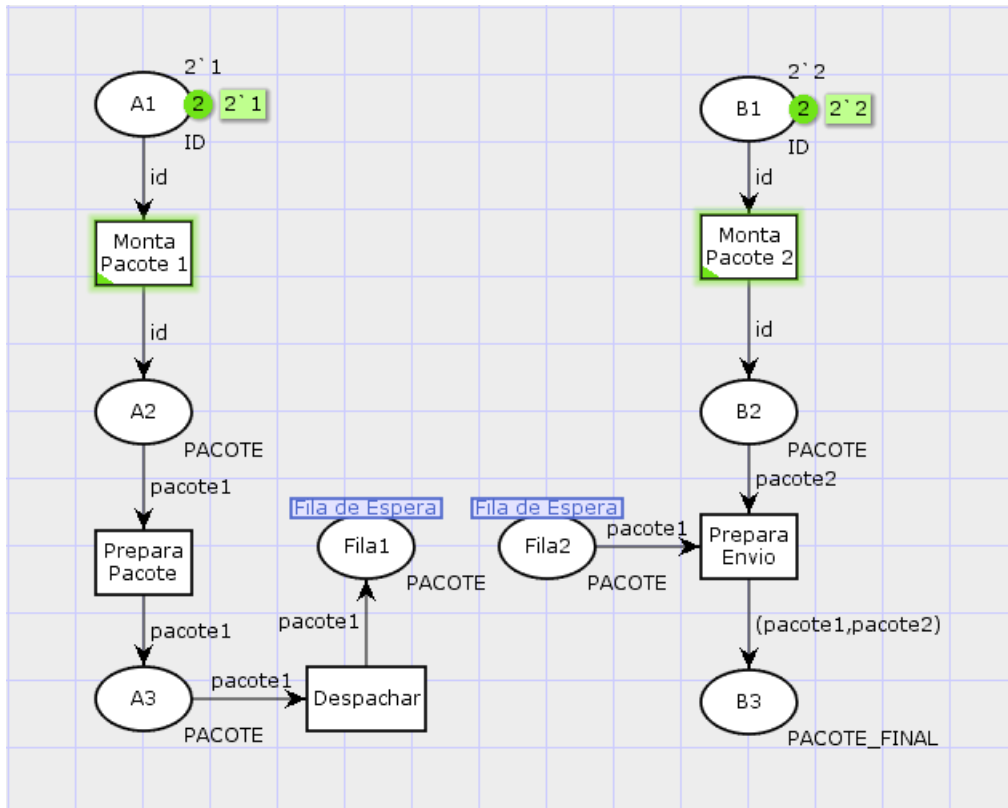


Figura 10 – Exemplo de um modelo utilizando *fusion places* no CPN Tools.

Os lugares *Fila1* e *Fila2* são marcados com a *fusion tag* “Fila de Espera”, isso significa que eles pertencem ao mesmo fusion set. Quando a transição *Despachar* for disparada, uma ficha será adicionada ao lugar *Fila1* e, conseqüentemente também será adicionada uma ficha no lugar *Fila2*. A implementação das cores utilizadas neste modelo são apresentadas a seguir:

```

1 colset ID = INT;
2 colset PACOTE = ID;
3 colset PACOTE_FINAL = product PACOTE*PACOTE;
4 var id: ID;
5 var pacote1, pacote2: PACOTE;

```

Além da possibilidade de criar modelos hierárquicos, os fusion places funcionam como uma ferramenta adequada de desenho permitindo que não haja cruzamento de arcos, deixando o modelo visualmente mais organizado, sem mudar o seu significado e sem ser necessário realizar nenhuma declaração especial (JENSEN; KRISTENSEN, 2009).

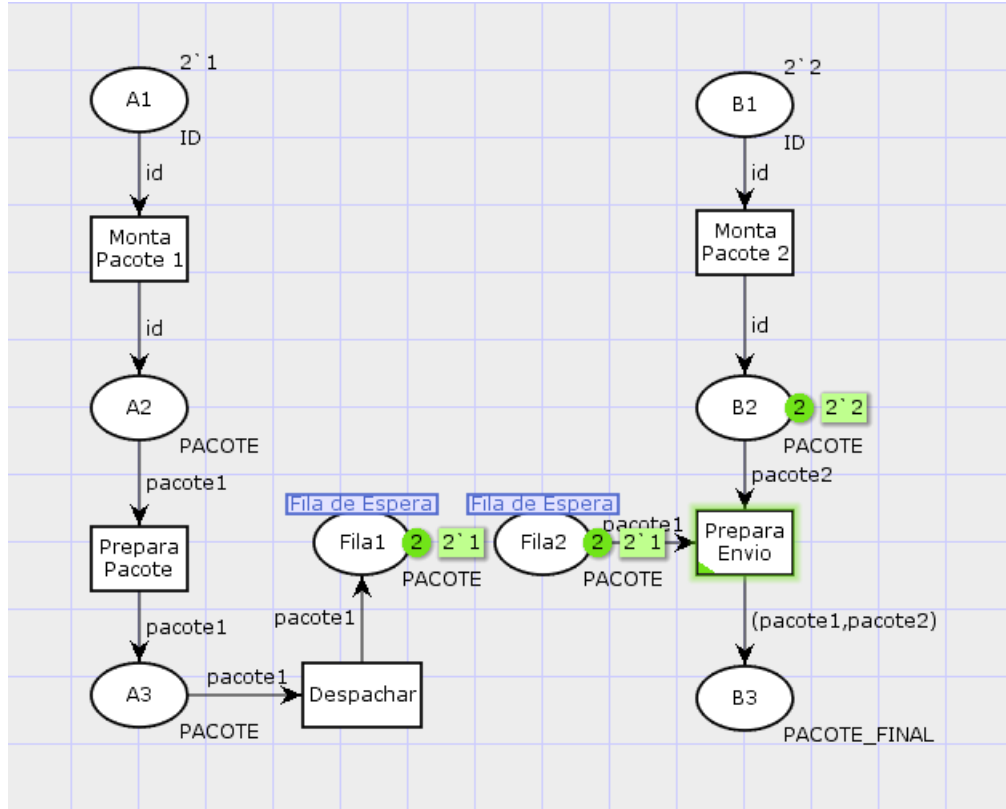


Figura 11 – Exemplo da execução do modelo com *fusion places* no CPN Tools.

2.3 Workflow nets

Uma rede de Petri que modela um processo de negócio é chamada de *Workflow net* (AALST, 1998; AALST; HEE, 2004). Uma *Workflow net* satisfaz os seguintes requisitos (AALST, 1998):

- a) possui apenas um lugar de início (i) e apenas um lugar de fim (o). Esses dois lugares são tratados como lugares especiais. O lugar i tem apenas arcos de saída e o lugar o tem apenas arcos de entrada. Uma ficha em i representa um caso que precisa ser tratado. Uma ficha em o representa um caso que já foi tratado;
- b) em cada tarefa (transição) e condição (lugar) deve estar em um caminho que se encontra entre o lugar de início (i) e o lugar de término (o).

A definição formal de uma *Workflow net* é apresentada a seguir (AALST; HEE, 2004):

Definição 6 (Workflow net) A rede de Petri $R = (P, T, F)$ é uma *Workflow net* se e somente se:

1. existe um único lugar de início $i \in P$ tal que $\bullet i = \phi$;
2. existe um único lugar de saída $o \in P$ tal que $o \bullet = \phi$;
3. cada nó $x \in P \cup T$ está em um caminho entre os lugares i e o .

Um processo especifica quais tarefas precisam ser executadas e em qual ordem executá-las. Modelar um processo de negócio em termos de um *Workflow net* é bem direto: as tarefas são modeladas por transições, condições são modeladas por lugares e os casos são modelados pelas fichas (AALST, 1998).

Para ilustrar o mapeamento de um processo por uma *Workflow net*, considera-se o processo de tratamento de reclamações (*Handle Complaint Process*) apresentado por Aalst e Hee (2004) que registra as atividades feitas quando uma nova reclamação é gerada, passando por diferentes departamentos até concluir o caso, realizando a compensação ou enviando uma carta.

A Figura 12 ilustra a *Workflow net* que representa este processo. De acordo com Aalst e Hee (2004), as seguintes construções básicas para o roteamento de tarefas devem ser consideradas:

- a) sequencial: a forma mais simples de execução de tarefas, onde uma tarefa é executada após a outra, havendo, claramente, dependência entre elas;
- b) paralela: mais de uma tarefa pode ser executada simultaneamente, ou em qualquer ordem;
- c) condicional (ou rota seletiva): quando há uma escolha entre duas ou mais tarefas;
- d) iterativa: quando é necessário executar uma mesma tarefa (ou um conjunto de tarefas) múltiplas vezes.

Considerando o processo de tratamento de reclamações mostrado na Figura 12, as tarefas *Contact_Client* e *Contact_Department* são um exemplo de roteamento paralelo. As tarefas *Collect* e *Assess* são um exemplo de roteamento sequencial. Já as tarefas *Pay* e *Send_letter* são um exemplo de roteamento condicional.

2.3.1 *Soundness*

Soundness é o principal critério de correção definido para as *Workflow nets*. Uma *Workflow net* é *Sound* se, e somente se, os três requisitos a seguir são satisfeitos (AALST; HEE, 2004):

- a) para cada ficha colocada no lugar de início, uma (e apenas uma) aparece no lugar de término;
- b) quando uma ficha aparece no lugar, todos os outros lugares estão vazios, considerando o caso em questão;
- c) considerando uma tarefa associada a uma transição, é possível evoluir da marcação inicial até uma marcação que sensibiliza tal transição, ou seja, não deve haver nenhuma transição que não pode ser disparada na *Workflow net*.

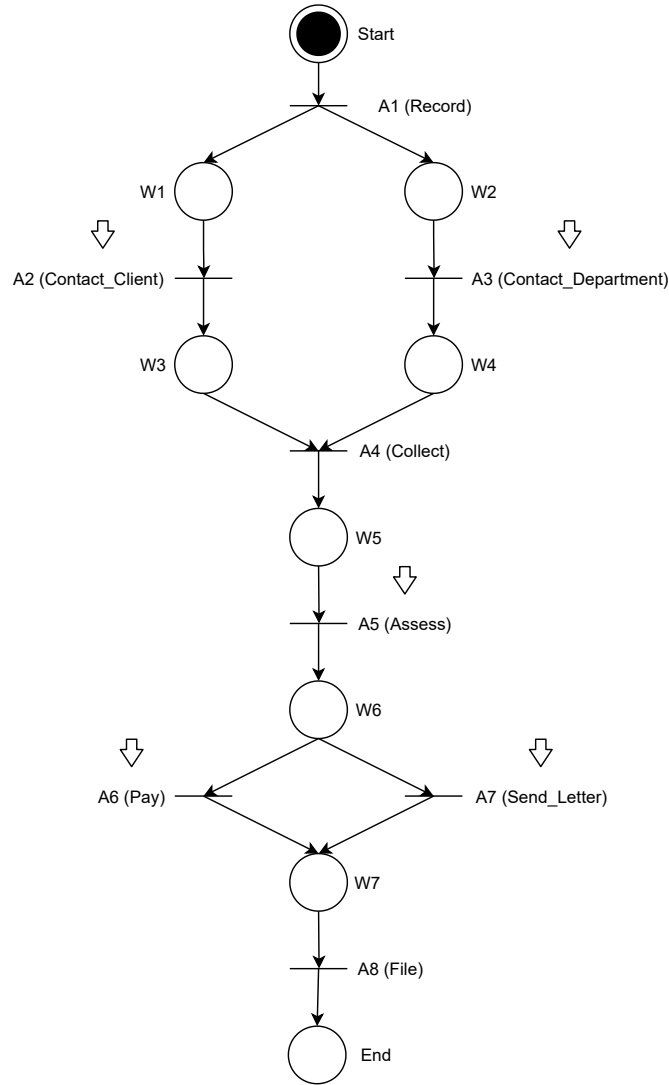


Figura 12 – *Workflow net* para o Processo de Tratamento de Reclamações apresentado por (AALST; HEE, 2004)

A definição formal do critério de correção *Soundness* no contexto das *Workflow nets*, proposto por Aalst (1998), é apresentada a seguir:

Definição 7 (*Soundness*) Um processo de workflow modelado por uma *Workflow net* $PN = (P, T, F)$ é *Sound* se, e somente se:

- a) para cada marcação M alcançável a partir da marcação i , existe uma sequência de disparo que leva da marcação M para a marcação o . Formalmente:

$$\forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o);$$

- b) a marcação o é o único estado alcançável a partir da marcação i com pelo menos uma ficha no lugar o . Formalmente:

$$\forall_M (i \xrightarrow{*} M \wedge M \geq o) \Rightarrow (M = o);$$

c) não há nenhuma transição morta em (PN, i) . Formalmente:

$$\forall_{t \in T} \exists_{M, M'} (i \xrightarrow{*} M \xrightarrow{t} M');$$

A *Workflow net* apresentada na Figura 12 é *Sound*, pois todos os critérios de definição de *soundness* foram contemplados.

2.4 SIMIO

O SIMIO é um *software* de modelagem e simulação que é empregado em diversas áreas do conhecimento como logística, gestão, pesquisa operacional, entre outras. É geralmente utilizado em empresas e indústrias da área da saúde, mineração e transporte. Apesar de ser um *software* relativamente novo, o SIMIO contém abordagens de modelagem avançadas que são indisponíveis nos softwares tradicionais (MOTA et al., 2017).

O SIMIO fornece um ambiente de modelagem 2D ou 3D multi-paradigma, combinando orientação a objetos e orientação a processos. Cada objeto declarado representa um componente físico do sistema e seu comportamento caracterizado a partir de suas definições. As características declaradas do objeto são passadas para os objetos derivados devido à propriedade de herança do paradigma. Dentre as várias vantagens do SIMIO como *software* de modelagem 3D, destacam-se as seguintes: (MOTA et al., 2017):

- ❑ É um software relativamente novo que tem abordagem de modelagens avançadas que não são encontradas em outros *softwares* tradicionais.
- ❑ Não é necessário conhecimento em nenhuma linguagem de programação específica para desenvolver as funcionalidades necessárias de um modelo.
- ❑ Contém uma poderosa interface gráfica, que faz com que os modelos desenvolvidos sejam entendidos facilmente, mesmo por pessoas que não são familiarizadas com modelagem e simulação.
- ❑ Integrar redes de Petri coloridas com o SIMIO permite que as funcionalidades de cada um complementem um ao outro suficientemente para que o designer tenha um sistema de análise e simulação muito poderoso.

SIMIO foi criado com uma ferramenta de modelagem de multi-paradigmas que suporta tanto orientação a objetos quanto orientação a processos. Em outras palavras, todos os objetos utilizados na área de trabalho do SIMIO (*facility view*) são, em essência, objetos que contém toda a lógica baseada em processos que governam seus comportamentos. Esses processos são encapsulados nos objetos, mas novos processos podem ser utilizados para estender a lógica do comportamento do objeto (PEGDEN, 2007; KELTON; SMITH;

STURROCK, 2010). Segundo Mota et al. (2017), o modo como as redes de Petri são integradas com a modelagem de ambientes não é bom o suficiente em ajudar o analista a entender a relação de causa e efeito de relacionamento entre os elementos (objetos) dos sistemas. Além disso segundo os autores, o formalismo das redes de Petri é muito abstrato para ser apresentado às pessoas que decidem, que na grande maioria dos casos são os usuários finais.

Com a combinação de uma abordagem de orientação a objetos com o paradigmas de processos, é possível um alto grau de flexibilidade no desenvolvimento de modelos. Isso é particularmente interessante para implementar semânticas de redes de Petri (MOTA et al., 2017). SIMIO, na sua versão completa, é composto por 6 seções: a *facility view* ou área de trabalho (é onde os modelos são desenvolvidos); a área dos processos é utilizada para estender as funcionalidades dos objetos; a área das definições, onde variáveis, propriedades e outros elementos são adicionados; a área da informação que é utilizada para definir propriedades utilizando tabelas; o quadro que serve para monitorar a evolução dos objetos no SIMIO; e a área de resultados, em que os resultados podem ser analisados.

2.4.1 Objetos

Um objeto é utilizado no SIMIO na área do *Facility View* e representa um componente físico do sistema, como um cliente, um trabalhador, uma máquina, um veículo ou um caminho. O comportamento do objeto é controlado pela sua definição. Se a definição mudar, todos os objetos derivados desta definição serão modificados devido a propriedade de herança.

No SIMIO, todos os objetos são derivados de uma classe geral determinada por sua principal funcionalidade. Em particular, os diferentes objetos disponíveis na biblioteca inicial tem um comportamento específico derivado do comportamento básico de cada classe. Os objetos tem os seguintes três principais componentes:

- ❑ Modelo (comportamento)
- ❑ Interface (propriedades, estados, eventos)
- ❑ Representação externa (entrada e saída de nós, gráficos)

Qualquer modelo pode ser transformado em um objeto se uma interface ou uma representação externa for adicionada. Um projeto pode ter um “modelo principal” e um ou mais “sub-modelos” que são utilizados como objetos. Por definição, um projeto do SIMIO contém um “modelo entidade” que pode ser utilizado como uma entidade “burra” ou pode ser melhorada com estados, propriedades, atributos, representações externas ou uma lógica particular.

2.4.2 Elementos úteis para implementação

Alguns dos elementos mais importantes de um modelo de simulação do SIMIO que serão utilizados para criar um modelo compatível com uma rede de Petri são apresentados a seguir (MOTA et al., 2017):

1. **Objetos Fixos:** Um objeto fixo tem uma localização estática na área de trabalho. Os objetos fixos têm nós associados ou portas pelos quais entidades entram/saem. Dentre eles, pode ser mencionado os objetos *SOURCE*, *SERVER* e *SINK*.
2. **Links:** *Links* definem um caminho para entidades e as transportam entre dois nós. Podem ser unidirecionais ou bidirecionais e as conexões podem ser utilizadas para criar redes. Estes *links* têm um peso que pode ser utilizado para selecionar um caminho.
3. **Nós:** Os nós definem o início e o final de um ou mais *links*. Eles podem ser utilizados também para modelar interseções de entradas/saídas múltiplas de *links*. Os nós também podem definir a entrada e saída de pontos de um objeto fixo associado.
4. **Entidades:** Entidades são objetos que fluem pelos *links* e podem entrar e sair de objetos. Também podem pertencer a uma rede particular determinada por nós e *links*, e seguem uma rota sequencial específica. Dada que as entidades são objetos no SIMIO, elas podem ter uma lógica específica que controla seu comportamento. Geralmente, entidades são geradas por um objeto *SOURCE*, mas podem também ser geradas através do uso de um processo interno de um objeto e sob condições específicas.
5. **Estações:** As estações podem ser encontradas nas definições de área do SIMIO. O conceito de estação no SIMIO representam o lugar onde as entidades que fluem pelo modelo podem ser localizadas dentro de um objeto. Neste sentido, e dado que elas são parte da anatomia de um objeto no SIMIO, elas podem ser utilizadas para estender as funcionalidades de um objeto por meio da herança para desenvolver um novo objeto e ser utilizado de forma independente no modelo para armazenar entidades.
6. **Atributos:** O conceito de atributo é implementado no SIMIO através do que é chamado de *estados*. Essas variáveis no SIMIO correspondem a atributos já associados ao modelo global como um objeto pai ou com objetos que fazem parte de um modelo global que conceitualmente pode ser associado ao local de variáveis em qualquer linguagem de programação. A principal característica dos atributos são os valores adquiridos que serão modificados com a execução da simulação.
7. **Propriedades:** O conceito de propriedade é implementado no SIMIO através das *propriedades* do objeto. Estas propriedades são parâmetros que definem a morfologia

do objeto e o principal objetivo para estes parametros é ser definido durante o processo de desenvolvimento do modelo. Por estas propriedades serem parâmetros que definem o comportamento do objeto, elas podem ser adicionadas em sua propria descrição para ter um modelo mais ajustado. Sua principal característica é que seu valor é definido no começo da simulação e não muda mais durante a execução da simulação.

8. **Tokens:** o conceito de tokens é inerente ao SIMIO, mas o conceito é diferente da semântica utilizada nas redes de Petri. No SIMIO, este conceito é utilizado para governar os passos (*steps*) na área de processos. Não é possível ver estes tokens graficamente, mas os diferentes processos que definem a função lógica de um objeto são executados através de um fluxo virtual de passos que contemplam o processo.

2.4.3 Equivalência entre SIMIO e Redes de Petri

Utilizar o SIMIO junto com um modelo formal permite que se tenha vantagens de capacidade das duas abordagens. De um lado é possível formalizar um modelo de simulação baseado em eventos discretos. Do outro é possível empregar as ferramentas representações gráficas poderosas em programas como SIMIO. Tudo isso considerando abordagens hierárquicas e orientadas a objetos para a elaboração progressiva de modelos complexos (MOTA et al., 2017).

A seguir são apresentados meios para implementar os principais elementos do formalismo da rede de Petri colorida no ambiente de simulação do SIMIO (MOTA et al., 2017):

1. **Tokens:** As entidades do fluxo no SIMIO são utilizadas para alcançar o relacionamento correspondente com os tokens das redes de Petri. Quando estas entidades não têm nenhum atributo, elas correspondem a um token ordinário da rede de Petri, mas elas podem corresponder também à tokens do formalismo das redes de Petri coloridas se foram adicionados atributos a utilizar os *estados* do SIMIO. A Figura 13 mostra a correspondência das entidades com os tokens que tem atributos (cores). Caso os tokens sejam utilizados como cores, é necessário definir estados para as entidades correspondentes do modelo SIMIO. Estes atributos podem ter diferentes tipos de dados, como *Integer*, *Boolean*, *String*, entre outros.

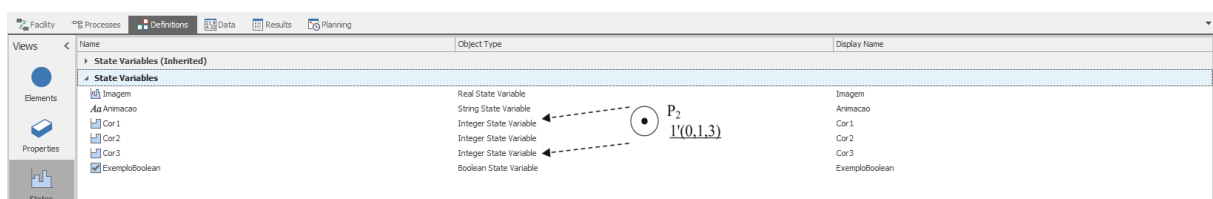


Figura 13 – Tokens coloridos no SIMIO

2. **Lugares:** Para implementar modelos de rede de Petri, os lugares corresponderão no SIMIO às *estações*. A funcionalidade das estações no SIMIO corresponde a armazenar as entidades e fazer parte de quase todo objeto da biblioteca padrão do SIMIO. A Figura 14 ilustra a arquitetura de um objeto no SIMIO.

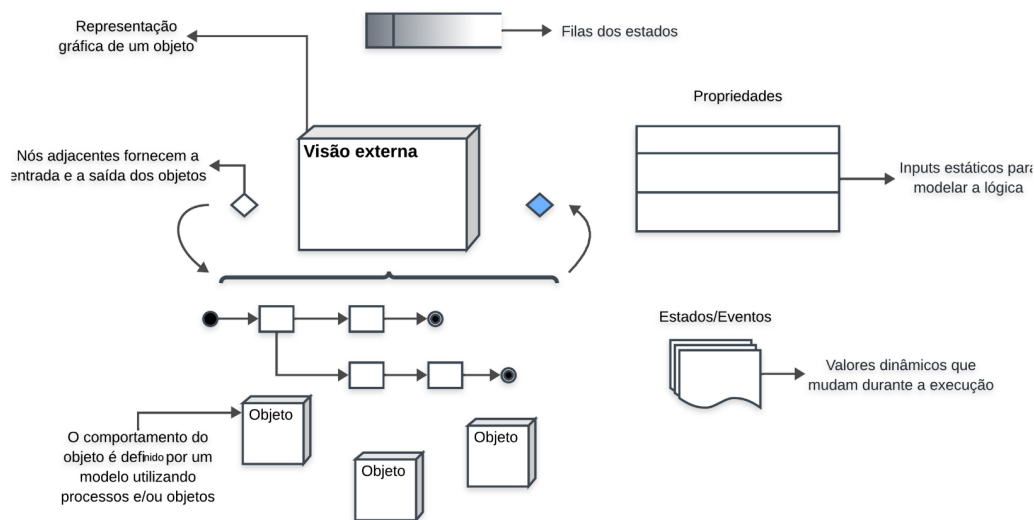


Figura 14 – Arquitetura de um objeto no SIMIO (FONTE: Mota et al. (2017))

As estações são elementos que podem existir como um objeto independente ou como um objeto que faz parte de outro objeto mais completo do ambiente. Quando uma representação gráfica de uma rede de Petri é utilizada para modelar uma atividade, os tokens são localizados nos lugares. Se uma correspondência direta com as estações do SIMIO for alcançada, as entidades do SIMIO que modelam os tokens serão localizadas nas estações e podem ser visualizadas por uma fila.

As Figuras 15 e 16 mostram um exemplo da definição uma estação no SIMIO para armazenar os tokens durante a simulação.

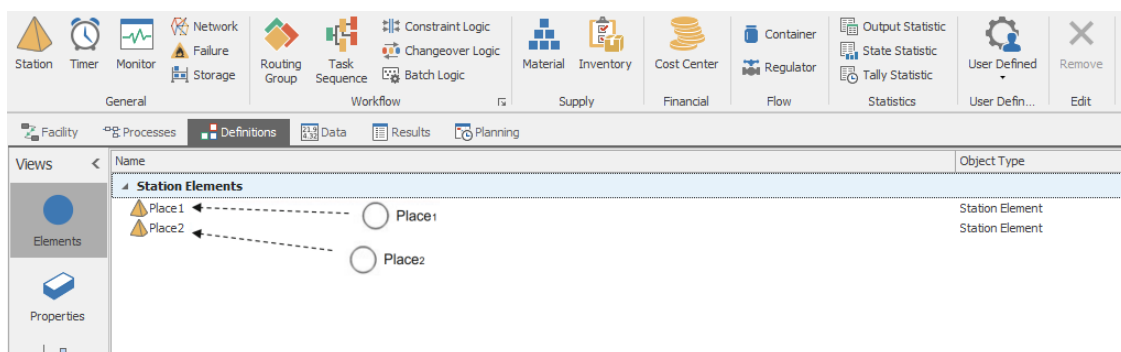


Figura 15 – Definição das *stations* no SIMIO

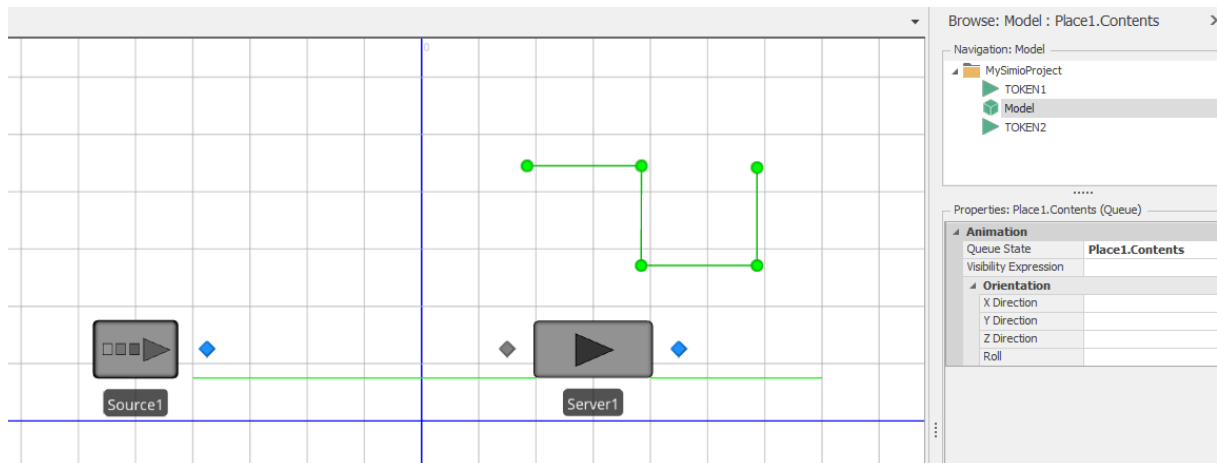


Figura 16 – Definição das *stations* no SIMIO na aba *Facility View*

A associação de uma fila com as entidades que são encontradas nas estações é especificada no campo de propriedade das filas onde aparece a instrução **QueueState** |**PLACE1.Contents**, na qual *PLACE1* é a representação do lugar no modelo.

2.4.4 Eventos condicionais e Satisfação de restrições

Eventos condicionais são todos aqueles onde um conjunto de condições devem ser atingidas para que os eventos possam continuar. Quando um sistema é analisado, o analista geralmente encontra as condições para liberar a ativação dos eventos. Os eventos condicionais são adaptados para o modelo CPN por meio do uso de restrições que são implementadas como nós de entrada, guardas, peso de arcos e expressões do modelo CPN. Diferente das redes de Petri ordinárias, os eventos condicionais nas CPNs são especificados não somente pelo peso dos arcos e pelos arcos de entrada de uma transição, mas também pelas expressões do arco e guardas associados a transição. As expressões dos arcos impõem restrições nas cores dos tokens que participam na verificação das condições que sensibilizam a transição. Separadamente, as guardas são expressões Booleanas que garantem restrições associadas com a transição e os valores das variáveis dos arcos. As restrições associadas com a guarda devem ser satisfeitas junto com todas as restrições para que a transição seja sensibilizada.

Para desenvolver um modelo equivalente no SIMIO, é necessário utilizar os seguintes elementos:

- ❑ Estações
- ❑ Steps de decisão

As estações (*Stations*) podem ser utilizadas de modo independente como elemento do modelo, ou também podem ser pré-alocadas nos objetos do SIMIO. Se estações dos objetos da biblioteca padrão forem utilizadas, as funcionalidades implementadas na biblioteca de

objetos pode, ser utilizadas assim como é o caso dos processos *seize*, *delay* e *release* que já vêm prontos no objeto *Server*. Por este motivo, neste trabalho são utilizados os objetos do tipo *Server* no lugar de *Stations* para aproveitar as funcionalidades pré implementadas dos objetos.

A implementação das restrições definidas pelos arcos de um modelo CPN pode ser feito utilizando os *Decide Steps*. Eles são encontrados na área dos processos e são equivalentes a estrutura de controle *IF.THEN.ELSE* das linguagens de programação. A propriedade *DecideType* deve então ser especificada como uma condição de guarda de uma rede de Petri colorida. O atributo *Condition or Probability* deve conter as condições lógicas que irão satisfazer a condição de guarda do modelo.

O processo em que o *Decide Step* será configurado irá seguir com a sua execução baseado no resultado da condição imposta. Caso verdadeiro, irá transferir a entidade para outro local; caso falso irá destruí-la. A Figura 17 indica como isso pode ser implementado usando o ambiente de desenvolvimento do SIMIO.

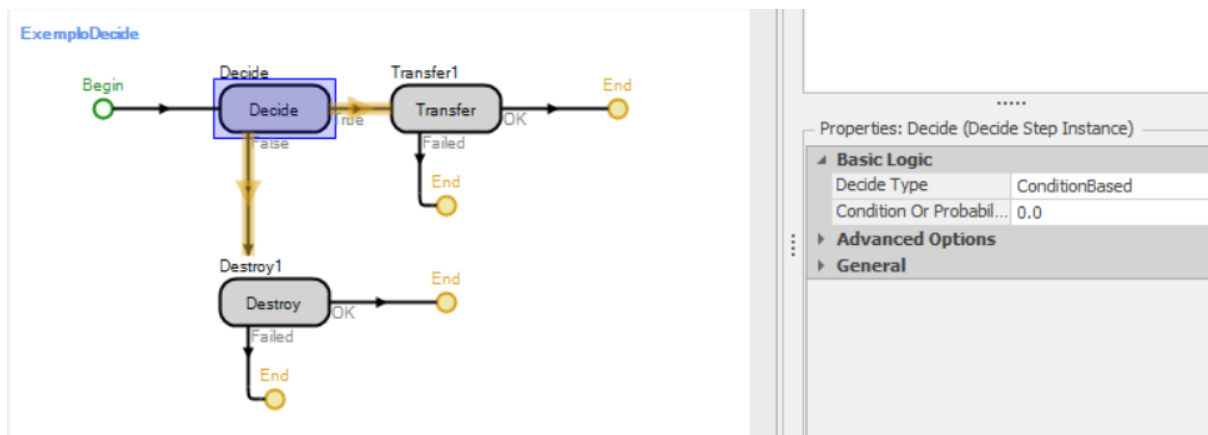


Figura 17 – Propriedades do *Decide Steps* no SIMIO.

2.4.5 Modelando Sincronismo

Sincronização é a característica que pode ser modelada usando o formalismo ordinário das redes de Petri. O formalismo torna isso possível sem nenhum problema devido ao uso de uma simples transição de sincronização. A Figura 18 mostra um exemplo de dois sub-processos que são sincronizados quando se encontram numa mesma transição de sincronização.

Neste exemplo, a transição entre $W3$, $W4$ e $W5$ sincroniza as atividades que continuam o seu fluxo de forma única. Esta transição não é disparada até que o token das atividades $W3$ e $W4$ estejam em suas respectivas atividades.

As características da sincronização de processos podem ser modeladas no SIMIO de diferentes formas. Uma das formas é a utilização do objeto *Server* e a propriedade *Secondary Resources*. Esta propriedade é utilizada para especificar que um processo não pode

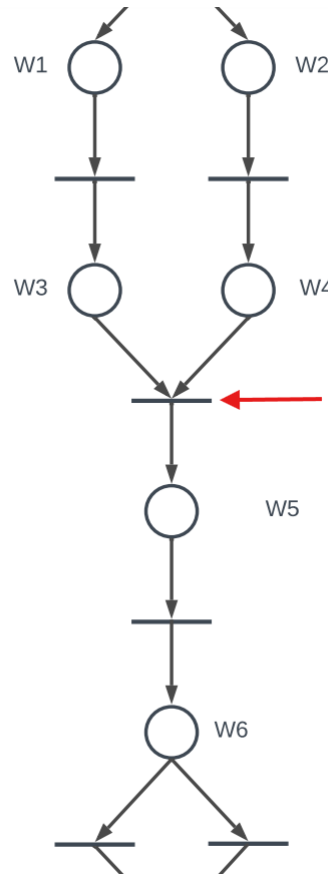


Figura 18 – Exemplo de uma rede de Petri com sincronismo.

começar até que os recursos especificados pela propriedade estejam disponíveis. Porém, no caso da tradução de uma situação de sincronização numa rede de Petri colorida, é possível também utilizar os elementos *Decide step* e *Station* para implementar o mecanismo de sincronização.

Na lógica interna do *Decide*, é possível atribuir uma condição ou probabilidade para que algo ocorra. Neste caso, é possível criar uma condição que verifique se há conteúdo nos lugares *W3* e *W4* da Figura 18 para então seguir com o processo. A Figura 19 mostra como pode ser atribuído tal condição.

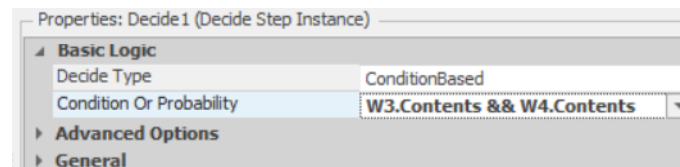


Figura 19 – Propriedades do *Decide Steps* no SIMIO.

2.4.6 Modelando Paralelismo

Paralelismo é a ocorrência de processos que se desenvolvem a partir de transição em comum. A Figura 20 ilustra a implementação do paralelismo nas redes de Petri.

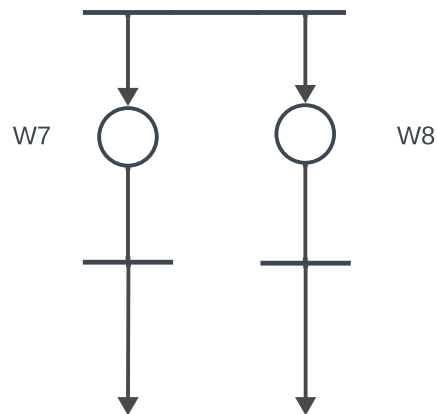


Figura 20 – Processos em paralelo em uma CPN.

A dinâmica do desenvolvimento das atividades em paralelo é bem simples de modelar. No SIMIO, a implementação do paralelismo é simples, pois pode ser realizada pela criação de um nó conectado a dois ou mais objetos do tipo *Servers*. A Figura 21 mostra os *steps* no SIMIO que são responsáveis por realizar tal comportamento, assim como da criação da entidade adicional (duplicação de um token da rede de Petri colorida quando acontece a situação de paralelismo) que irá acompanhar o comportamento da estrutura de controle paralela.

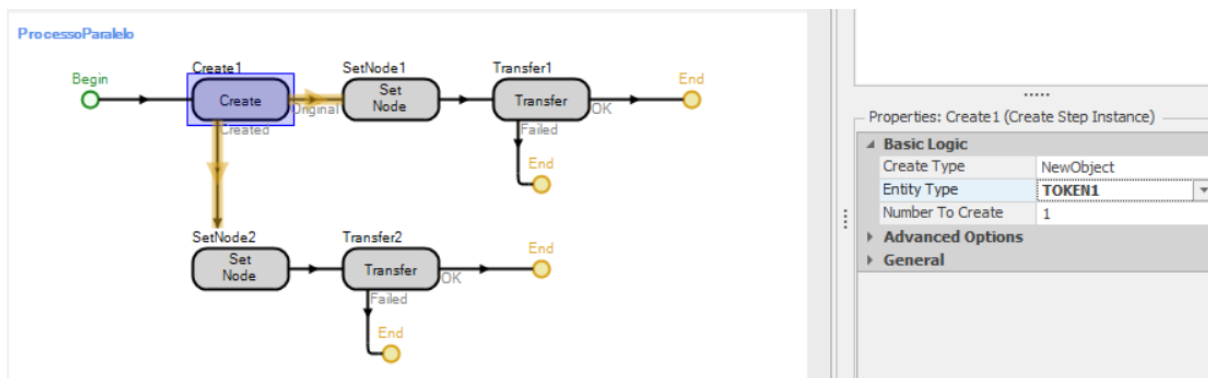


Figura 21 – Propriedades do *Decide Steps* associado ao objeto *Server* no SIMIO.

É possível observar que na implementação do processo na Figura 21, o step *Create* é utilizado para criar um novo objeto (neste caso uma entidade chamada *TOKEN1*). Ao final do processo de criação irão existir duas entidades do tipo *TOKEN1* e cada uma de modo independente no seu respectivo processo, devido ao step *SetNode*.

Finalmente, o step *Transfer* envia a entidade para o ponto desejado do modelo. Basta selecionar nas configurações do step qual será o destino da entidade para que ela seja transferida para lá para seguir seu processo normalmente.

2.4.7 Modelando recursos em processos

Numa rede de Petri ordinária, um processo é um conjunto de transições que modelam a evolução de um sistema, enquanto processos nas CPNs utilizam também elementos como transições, lugares, nós, arcos e todos os elementos inerentes ao formalismo. Uma das vantagens das CPNs é que os processos podem ser modelados de forma mais compacta na versão hierárquica do modelo, até o ponto em que apenas uma transição é necessária para modelar um processo inteiro.

A Figura 22 ilustra um exemplo típico de um recurso sendo utilizado para disponibilizar um processo.

Para implementar este tipo de situação no SIMIO, é possível utilizar os elementos *Seize*, *Delay* e *Release*.

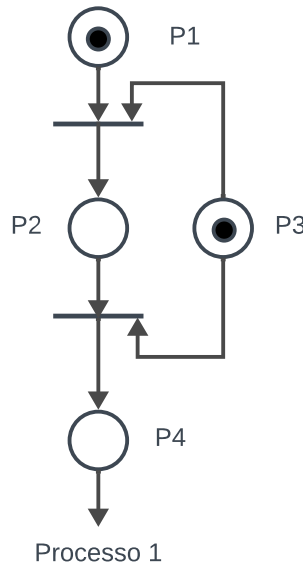
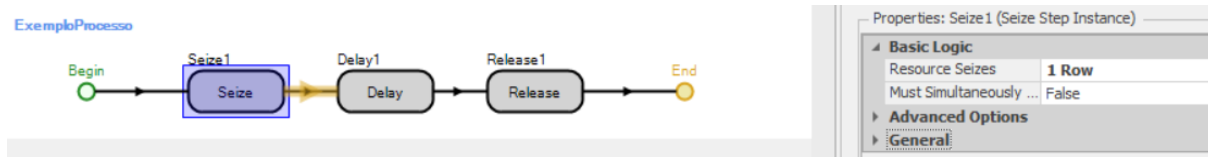


Figura 22 – Exemplo de recurso em uma CPN.

É possível definir um modelo de simulação usando a noção de recurso considerando a lógica interna de um objeto de tipo *Server* que autoriza a especificação de *steps* de simulação. O *step seize* permite especificar qual objeto será o recurso ou os recursos utilizados para este processo. O exemplo da Figura 22 equivalente no SIMIO é representado na Figura 23. Neste caso o *step Seize* irá reservar o recurso configurado na opção *Resource Seizes*. Após o tempo determinado pelo *step Delay*, ele irá liberar o recurso no *step Release*.

Esta sequência pode ser inserida em qualquer objeto do SIMIO (nós, servers, links, entre outros) associado com algum evento.

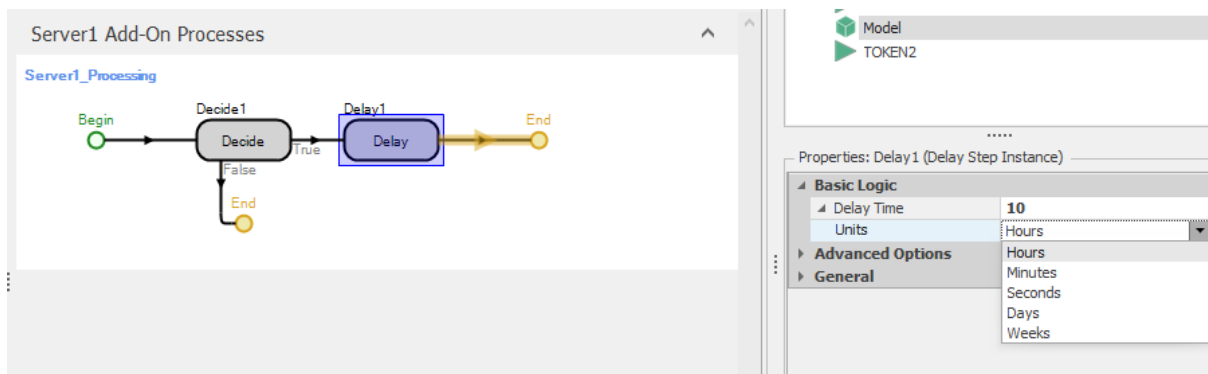
Além disso, existem alguns objetos como *Server* que já contém esta lógica implementada por padrão. Consequentemente, é necessário apenas adicionar estes objetos no

Figura 23 – Step *Seize* no SIMIO.

facility view e especificar as propriedades dos recursos que serão necessários para realizar a operação.

2.4.8 Consumo de tempo

O consumo de tempo é crucial para simular e analisar como os sistemas funcionam. Utilizando redes de Petri colorida, o consumo de tempo acontece toda vez que uma transição é disparada. No caso do SIMIO, o consumo de tempo é feito utilizando o step *delay*. Este step é localizado em um processo logo após a verificação das restrições associadas a objetos do tipo entidade que representam os tokens numa rede de Petri colorida. A Figura 24 mostra a implementação do step *Delay* em um processo simples no qual é possível escolher quanto tempo poderá ser gasto e em qual tipo de unidade será consumido (horas, minutos, segundos, entre outros).

Figura 24 – *Delay Step* no SIMIO.

2.4.9 Inserindo uma transição de rede de Petri no SIMIO

Devido a arquitetura do SIMIO, as regras associadas as CPNs com diferentes eventos podem ser implementadas com a lógica dos objetos. Cada objeto tem seu número particular de eventos, os quais podem conter *EntityEnter*, *Processed*, *EntityExited*, entre outros. Neste caso, é importante determinar qual evento do SIMIO irá executar a lógica associada com a CPN. A lógica de disparo em um objeto será inserida em um ponto no qual as características da entidade (associada ao token usado para realizar o disparo na CPN) não são afetadas pela lógica inerente do objeto onde a entidade reside. Então, sugere-se que a lógica de disparo da CPN seja implementada no evento associado com a entrada da entidade no objeto SIMIO correspondente (MOTA et al., 2017).

A Figura 25 mostra o ambiente do SIMIO chamado *Add-On Process Triggers*, no qual é possível encontrar os processos para estender as funcionalidades do objeto original com os processos desenvolvidos pelo usuário.

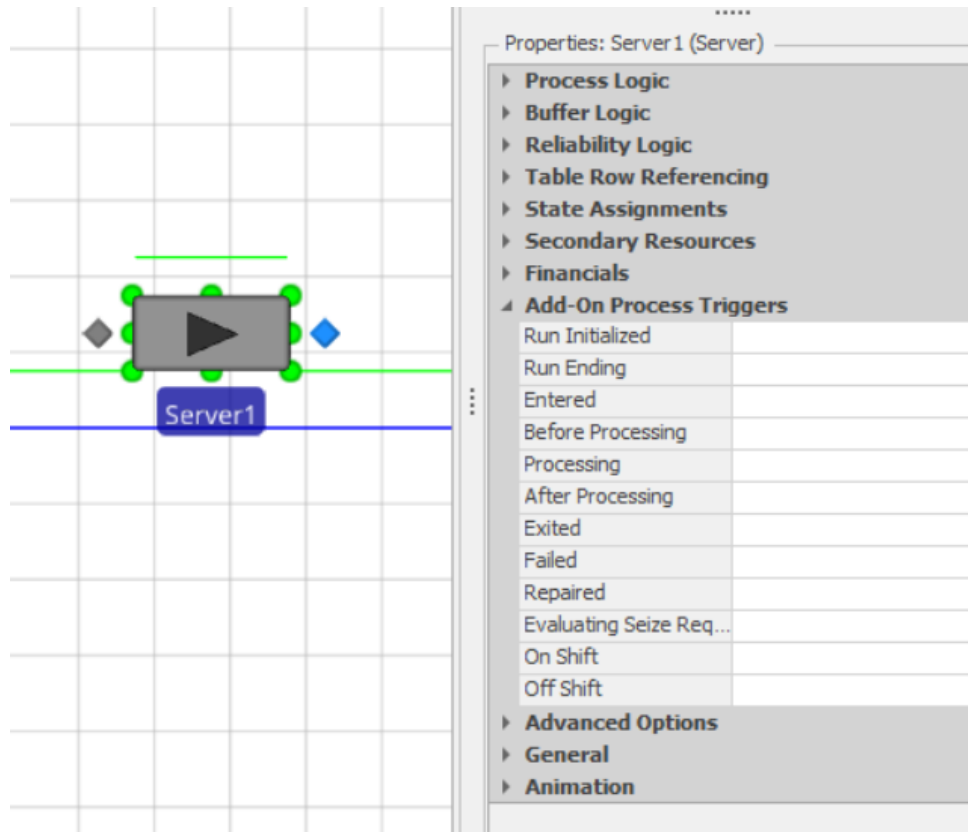


Figura 25 – Ativadores de Processos no SIMIO.

As propriedades da janela do modelo mostram que existem diferentes propriedades que podem ser modificadas. Estas propriedades irão mudar dependendo do objeto envolvido, mas geralmente todos os objetos terão estes ativadores. Ao expandir a seleção do *Add-On Process Triggers*, os eventos associados ao objeto são exibidos. São nestes eventos que a lógica de disparo de uma transição de uma CPN é implementada (MOTA et al., 2017).

A lógica de disparo de uma CPN pode ser integrada com o ambiente de modelagem de duas formas (MOTA et al., 2017):

A primeira forma de implementação de um disparo de transição é chamada de link automático. Isso é feito selecionando um evento, por exemplo o *Entered* ou *Exited*. Após a seleção do evento, o SIMIO muda automaticamente para a tela do processo onde o evento escolhido aparece e poderá ser alterado.

A segunda forma de implementar um disparo de transição, é através da realização de um link manualmente, onde o usuário modela o evento que corresponde ao disparo da transição diretamente na área de processos e depois implementa o link para conectar o evento de disparo a qualquer objeto do ambiente SIMIO envolvido com o disparo da transição (MOTA et al., 2017).

Como exemplo é possível utilizar a rede de Petri ilustrada na Figura 26. Esta é a representação de um modelo simples apenas para dar um passo-a-passo de como uma transição pode ser implementada no SIMIO. Na transição $T2$ a função `defineTempo()` é acionada assim que a transição é disparada. Esta função gera um valor uniforme de tempo entre 0.1 e 1.0.

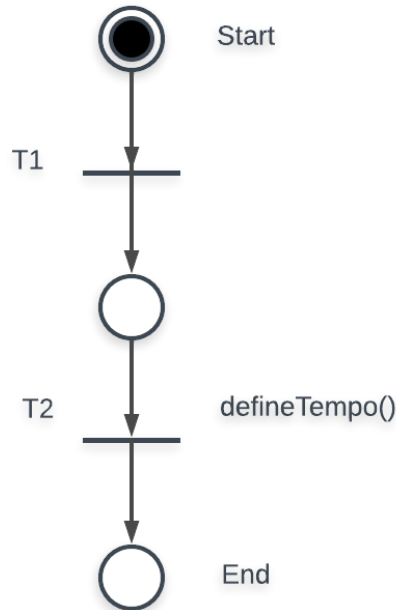


Figura 26 – Exemplo de transição CPN.

No SIMIO, o mesmo modelo é definido conforme ilustra a Figura 27. As transições $T1$ e $T2$ são representadas no SIMIO com objetos *Server*.

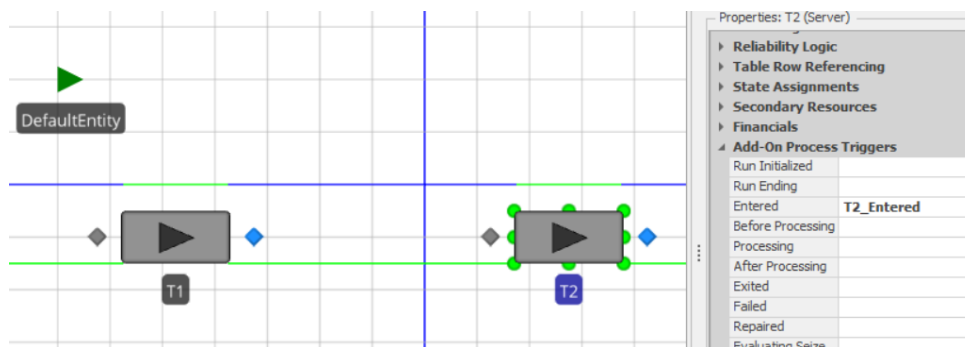


Figura 27 – Exemplo de transição no SIMIO.

O *Server* que representa a transição $T2$ está configurado para disparar a função `defineTempo()` assim que a entidade entrar no *Server* através do processo gerado automaticamente chamado `T2_Entered` ilustrado na Figura 28.

Foi definido uma variável chamada *tempo* para armazenar este valor. Assim que a entidade entra no *Server* $T2$, este processo é disparado, o cálculo da função é realizado

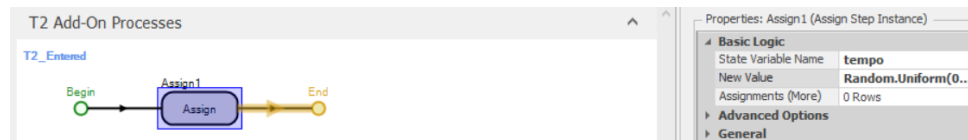


Figura 28 – Atribuição de valor na transição no SIMIO.

e o valor é armazenado na variável configurada. Dessa forma o mesmo comportamento que acontece na rede de Petri pode ser reproduzido no modelo SIMIO. Se for necessário que o evento aconteça em outro momento da simulação, basta escolher entre as opções disponíveis para adicionar um processo.

2.5 Considerações Finais

Neste capítulo foram apresentados os principais conceitos necessários para o desenvolvimento e entendimento desta pesquisa. Para começar foi mostrado o conceito original das redes de Petri, assim como seus recursos. Logo em seguida, foi mostrado também como as redes de Petri com Tempo e rede de Petri Coloridas funcionam, quais as suas diferenças e o motivo de utilizar o CPN Tools como ferramenta de simulação dos modelos de rede de Petri Coloridas.

Também foi apresentado neste capítulo o conceito de Workflow nets. Elas são um tipo particular de rede de Petri que são utilizadas para modelar processos de negócios. Nesta pesquisa foi utilizada nos modelos *singleplayer* e *multiplayer* que foram modelados com rede de Petri para representar a ordem das atividades a serem seguidas pelos jogadores. Além disso, a propriedade *soundness* pode ser verificada no modelo para garantir características importantes quando se considera um modelo de jogo.

Foi apresentado o *software* SIMIO que será utilizado para realizar a modelagem e a simulação dos modelos *singleplayer* e *multiplayer* em 3D. O software em questão possui ambiente amigável, permite a criação e análise de modelos 3D e contém orientação a objetos, que facilita a simulação de diferentes cenários. Além disso, foram apresentados exemplos de como os elementos das CPNs podem ser convertidos em elementos do SIMIO, baseados no trabalho de Mota et al. (2017).

Por fim, os conceitos relacionados a rede de Petri Coloridas, Workflow nets, CPN Tools e SIMIO apresentados neste capítulo serão utilizados para traduzir e simular os modelos *singleplayer* e *multiplayer* utilizando CPN Tools e SIMIO.

Trabalhos Correlatos

Neste capítulo serão apresentados os trabalhos correlacionados com esta pesquisa. O trabalho de Chueca et al. (2023) traz a evolução da Engenharia de Software relacionada a jogos nos últimos anos. Os autores realizaram uma revisão bibliográfica de 2009 à 2021, resultando em 98 estudos publicados e relacionados ao assunto de Engenharia de Software para jogos, sendo que dos 98 trabalhos citados pelos autores, cerca de 14 eram sobre design, 14 sobre estruturas de sistema e 26 sobre verificação e validação. Outra revisão bibliográfica foi realizada por Pieva e Bernardino (2023), ressaltando a importância em ter um documento de design e desenvolvimento para criar um jogo, mostrando a preferência dos trabalhos citados por utilizar este documento como guia para desenvolver o jogo e registrar como ele será feito.

Em Kessing, Tutenel e Bidarra (2012), os autores ressaltaram a falta de consistência que existe entre as representações visuais e como elas se comportam, agem e reagem nos jogos. Como solução, foi proposto um *framework* para facilitar o desenvolvimento de semânticas enquanto o jogo é feito e durante a manutenção do jogo após seu lançamento. No mesmo contexto, Marín-Lora, Chover e Sotoca (2020) apresentaram um sistema de design e desenvolvimento para a lógica de video games 2D. Segundo os autores, com a utilização do sistema, foi possível reduzir a quantidade de ações e condições durante a definição e desenvolvimento do jogo. Vários jogos foram testados para validar o potencial da ferramenta.

Em Abrahão et al. (2016), os autores apresentaram um método para ser usado para avaliar a usabilidade de video games nos estágios iniciais de um processo de desenvolvimento orientado ao modelo. Segundo os autores, o método depende de um modelo que decompõe usabilidade em atributos mensuráveis e métricas específicas para o jogo. Seguindo a mesma linha de desenvolvimento orientado ao modelo, Domingo et al. (2020) utilizaram dados reais de video games para avaliar se utilizar este método para o desenvolvimento e design realmente fazia diferença. Ao aplicar o método em dados e projetos reais, os autores questionaram os resultados obtidos por pesquisas passadas no quesito de limitar o modelo apenas para exercícios acadêmicos e desenvolvedores sem experiência.

Em Luhova et al. (2019) os autores utilizaram a ferramenta Canva para realizar a formalização do processo de *game design*. Segundo os autores, os responsáveis por este trabalho já estão acostumados a utilizar a ferramenta no dia-a-dia sendo então mais conveniente para o processo criativo. Os autores afirmam que é importante criar um algoritmo para automatizar este processo, porém a ferramenta não consegue realizar análises e nem garantir que a sequência de atividades esteja correta.

Redes de Petri foi utilizada também no trabalho de Chandler e Davidrajuh (2024), no qual os autores utilizaram a ferramenta para encontrar a opção mais eficiente para organizar os oleodutos do jogo “Factorio” e assim atingir uma maior pontuação. Devido a explosão de estados que foi gerada na simulação, os autores decidiram por utilizar redes de Petri modulares. Desta forma, foi possível continuar com as simulações sem perder os resultados.

No trabalho de Barreto et al. (2014), foi apresentado um modelo de simulação de cenário de jogo *singleplayer* com mecanismo de comunicação entre modelo lógico e modelo topológico. Os autores relataram que o modelo apresentado resulta em dados similares aos obtidos por jogadores na vida real, fazendo com que o modelo possa ser utilizado para simular cenários de jogos ainda durante a fase de design do jogo. Barreto e Julia (2020a) também apresentaram um mecanismo de pseudo-disparo utilizando redes de Petri coloridas para modelar um cenário de jogo *multiplayer* com comunicação entre modelo lógico e modelo topológico. Com o resultado obtido pelos autores foi possível simular o comportamento de recursos que são compartilhados entre jogadores e que a ação de um jogador influencia diretamente e/ou indiretamente nas ações que o outro jogador deve fazer.

Por fim, (NEDOPETALSKI et al., 2024) apresentou uma equivalência entre um modelo de jogo feito no CPN Tools e um modelo de simulação 3D utilizando o software SIMIO. Os autores criaram uma simulação em 3D para representar uma fase de um jogo *singleplayer* e utilizaram como base, um modelo de jogo *singleplayer* feito utilizando a ferramenta CPN Tools. Os resultados obtidos pelos autores foram similares aos obtidos pelos autores do modelo criado no CPN Tools. Segundo os autores, com o modelo 3D é possível ter um maior entendimento de como o jogo irá funcionar e a ferramenta pode ser utilizada ainda na fase de design para testar diferentes cenários e possibilidades.

3.1 Considerações Finais

Neste capítulo foram apresentados alguns trabalhos relacionados ao tema desta pesquisa que contribuíram para o processo de design de jogos. Apesar de muitos não utilizarem redes de Petri ou SIMIO para atingirem seus objetivos, todos apresentaram propostas para a criação da documentação nas fases iniciais dos jogos.

Os trabalhos de Chueca et al. (2023) e Pieva e Bernardino (2023), apresentaram uma

revisão bibliográfica, analisando a literatura nos últimos 15 anos, a respeito da evolução da área de criação e desenvolvimento de vídeo games. Os autores ressaltam que as pesquisas apontam a importância e necessidade de criar documentos nas fases iniciais do jogo que contribuam para a implementação e análises/testes dos jogos. As abordagens apresentadas por Kessing, Tutenel e Bidarra (2012) e Marín-Lora, Chover e Sotoca (2020), além de ressaltarem a dificuldade da criação de designs eficientes para video games, os autores propuseram um sistema e um framework como solução para este problema. Já Abrahão et al. (2016) propuseram um método para avaliar a usabilidade do jogo que retorna dados e métricas mensuráveis já nos estágios iniciais de desenvolvimento. Domingo et al. (2020) ao aplicar um método semelhante em dados de jogos reais, questiona os trabalhos anteriores por limitarem o método somente para exercícios acadêmicos e desenvolvedores sem experiência, quando poderia ser utilizado em projetos reais. (LUHOVA et al., 2019) abordaram a utilização da ferramenta Canva nas fases iniciais do desenvolvimento de um jogo por ser uma ferramenta popular entre a equipe de projeto. (CHANDLER; DAVIDRAJUH, 2024) utilizaram redes de Petri para encontrar a opção mais eficiente para organizar oleodutos do jogo “Factorio”. A abordagem pode ser aplicada em verdadeiros oleodutos posteriormente.

Por fim, os trabalhos de Barreto et al. (2014) e Barreto e Julia (2020a) propuseram modelos *singleplayer* e *multiplayer* utilizando CPN Tools. Além disso, também foi proposto um mecanismo para a comunicação entre modelo lógico e topológico. Barreto e Julia (2020a) também apresentaram um modelo em rede de Petri colorida com de pseudo-disparo para simular o comportamento de fases *multiplayer* em que os recursos e atividades são compartilhados entre os jogadores em um mesmo cenário. O trabalho de Nedopetalski et al. (2024) apresentou a tradução de um modelo CPN Tools *singleplayer* para um modelo 3D utilizando SIMIO. O modelo 3D não apenas possibilitou uma nova forma de visualizar o cenário de jogo, como também pôde ser simulado e validado, uma vez que a simulação obteve resultados similares aos do modelo em rede de Petri.

Dentro deste contexto, a pesquisa apresentada nesta dissertação propõe um modelo de simulação para jogos *multiplayer* utilizando tanto um modelo de rede de Petri, quanto um modelo em 3D. O modelo em rede de Petri será utilizado para especificar o fluxo de jogo e para obter análises com a simulação. Já o modelo 3D além de obter também os dados de análise por meio da simulação, também permitirá melhor visualização do fluxo de jogo e compreensão sobre quais, como e quando as atividades serão executadas no cenário estabelecido. Será possível garantir por meio do modelo, especificado ainda nas fases iniciais, que o jogo não terão falhas estruturais, em termos de jogabilidade. Dessa forma, a proposta deste trabalho contribuirá para a documentação dos requisitos do jogo, visualização 3D da especificação e análise dos requisitos definidos.

Modelo *Singleplayer*

Um jogo pode ser considerado como um tipo especial de aplicação multimídia que se trata de um produto de entretenimento que requer participação ativa do usuário (CALLELE; NEUFELD; SCHNEIDER, 2005). Também é necessário que um jogo seja divertido (LEWIS; WHITEHEAD, 2011) e que cada nível tenha significado para manter a coerência da sequência de atividades que devem ser feitas para atingir o objetivo do jogo (COLLÉ; CHAMPAGNAT; PRIGENT, 2005). Definir as atividades e analisar o seu fluxo de funcionamento durante o jogo na fase de design é uma tarefa árdua e importante (GAL et al., 2002), portanto é necessário criar novos métodos, técnicas e ferramentas para apoiar os criadores de jogos nas fases iniciais do projeto, facilitando a especificação e análise do jogo.

Este capítulo tem como ponto de partida os trabalhos de Barreto e Julia (2014) e Nedopetalski et al. (2024) que criaram modelos de jogos *singleplayer* utilizando a teoria das redes de Petri Coloridas e o software SIMIO, respectivamente. Ambos os modelos serão apresentados neste capítulo para ilustrar o funcionamento dos cenários de jogos no caso *singleplayer*.

4.1 Modelo em rede de Petri

Para ilustrar a abordagem, o jogo Silent Hill II foi utilizado. Este é um jogo de aventura e terror que começa em uma cidade deserta. O jogador assume o papel de James, que recebe uma estranha carta de sua esposa pedindo para se encontrarem em Silent Hill. Porém, a esposa dele já havia falecido a três anos atrás. Intrigado pelo mistério da carta James vai para Silent Hill. Para alcançar o principal objetivo do jogo, o jogador deve explorar o ambiente, lutar contra inimigos, coletar objetos e também resolver enigmas.

Estes objetivos são representados em formas de atividades (missões), as quais o jogador fica encarregado de cumpri-las. Barreto e Julia (2020b) criaram um modelo lógico para representar as atividades realizadas no primeiro nível do jogo Silent Hill II.

A Figura 29 ilustra o modelo lógico feito utilizando as CPNs do primeiro nível do jogo Silent Hill II. Ele possui 11 lugares (P0 até P10) e 9 atividades (Encontrar mapa de Silent Hill, Matar criatura, Pegar o rádio, Encontrar bilhete, Encontrar mapa no bar, Encontrar chave, Ir ao apartamento, Fork, Join). A ficha utilizada no modelo é do tipo *player* e a variável *p*, que está associada a todos os arcos do modelo, é do tipo (cor) *PLAYER*.

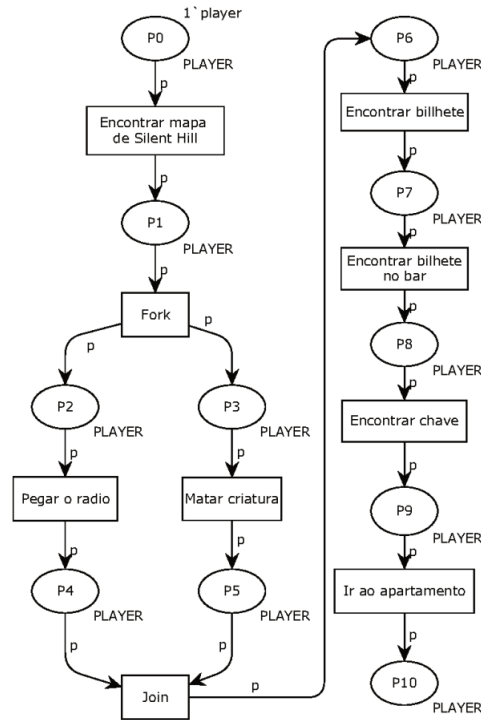


Figura 29 – Rede de Petri colorida do modelo lógico do primeiro nível do jogo Silent Hill II (Fonte: Barreto e Julia (2020b))

Em (BARRETO; JULIA, 2020b), os autores criaram um modelo lógico e um mapa topológico para representar as atividades e os locais do jogo na rede de Petri. Segundo eles, um mapa topológico pode ser dividido em partes de acordo com cada nível. As áreas significativas são lugares no mapa onde o jogador irá executar as atividades do jogo. A Figura 30 ilustra parte do mapa do jogo Silent Hill II, juntamente com as áreas mais significativas do mapa para o primeiro nível do jogo. Essas áreas são:

1. Plataforma de observação;
2. Floresta;
3. Igreja;
4. Quintal;
5. Túnel;
6. Trailer;

7. Bar;
8. Rua Martin;
9. Apartamento;

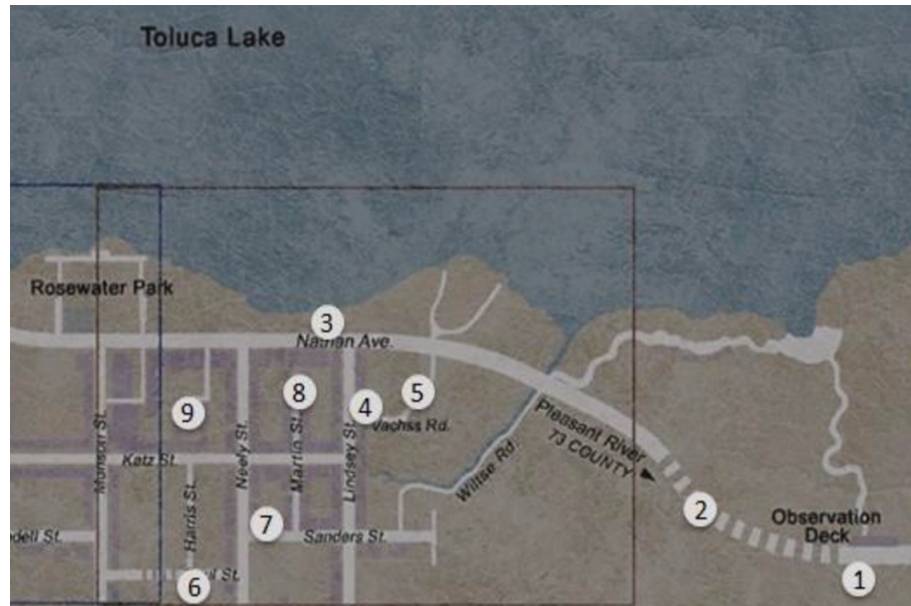


Figura 30 – Mapa parcial do primeiro nível de Silent Hill II (Fonte: Barreto e Julia (2020b))

Todas as ações do jogo acontecem em áreas significativas de um mapa topológico que representa o ambiente do mundo virtual. Esta topologia do jogo representa as áreas pelo qual o jogador irá transitar livremente, desde que satisfaça as condições específicas para conseguir acessar aquela área. Cada lugar do mapa pode ser um local específico como um quarto, ou o interior de uma casa ou simplesmente uma rua deserta. A descrição do mundo virtual, assim como a evolução do jogador dentro do jogo são indispensáveis para o processo de criação de jogos.

Um mapa topológico pode possuir as seguintes características:

- ❑ as áreas de um mapa são fixas durante todo o jogo, ou seja, o jogo é igual do começo ao fim;
- ❑ existem áreas do mapa que são significativas, pois são nelas que acontecem as missões;
- ❑ as áreas possuem ligações entre si;
- ❑ as áreas podem ser livres ou ter restrições para acessá-las. Caso ela seja restrita, o jogador deverá cumprir o requisito da restrição para poder acessar a área desejada.

É possível representar um mapa topológico de um jogo utilizando uma rede de Petri do tipo grafo de estado¹ (BARRETO; JULIA, 2020b). Neste grafo de estado, cada área significativa do mapa topológico é modelada por um lugar específico. As fronteiras entre as áreas são representadas por transições simples e cada transição da rede tem apenas um lugar de entrada e um lugar de saída. A localização do jogador é apresentada por uma ficha que se encontra em um lugar específico da rede. A orientação do arco ilustra qual a direção que o jogador poderá ir, dada as opções disponíveis.

A Figura 31 ilustra o grafo de estado que representa o Mapa Topológico para o primeiro nível de Silent Hill II. Existem 9 áreas significativas onde o jogador poderá transitar (Plataforma de observação, Floresta, Igreja, Túnel, Trailer, Bar, Quintal, Rua Martin e Apartamento). A adjacência das áreas é dada de acordo com o mapa do primeiro nível, representado pela Figura 30.

Inicialmente o jogador está localizado na área chamada *Plataforma de observação*. A ficha neste local representa a localização atual do jogador. Com a execução das atividades, o jogador mudará a sua localização. Porém ao iniciar, o jogador não pode acessar todas as áreas conectadas pois há requisitos pré-definidos para acessar aquelas áreas específicas.

Ao criar o modelo topológico de rede de Petri colorida, é necessário definir o conjunto de cores. Neste caso, foi definido a cor *PLAYER* e a variável p que é do tipo *PLAYER* para representar o jogador passando pelas áreas significativas do mapa.

O modelo de jogo *singleplayer* apresentado por Barreto e Julia (2014) foi baseado na teoria das redes de Petri e simulado na ferramenta CPN Tools. O modelo apresentado pelos autores é considerado global, pois contém a parte lógica e a parte topológica do jogo. Ambos os modelos são criados separadamente; no entanto os autores apresentaram um mecanismo de comunicação responsável por fazer a interação entre as atividades (Modelo Lógico) e os locais do ambiente do jogo (Modelo Topológico) para que a simulação dos modelos fique mais próximo da realidade.

Como exemplo é possível considerar o início do jogo Silent Hill II, no qual o jogador se encontra na área *Plataforma de observação*. No início, esta é a única área no qual o jogador pode ter acesso. A condição para o jogador deixar a primeira área é encontrar o mapa de Silent Hill, que é a primeira atividade do jogo e o jogador precisa estar na *Plataforma de observação* para conseguir realizar esta atividade. Para representar isto formalmente, um disparo de uma transição específica do Modelo Lógico representará a produção do recurso correspondente. Então é possível estabelecer um mecanismo de comunicação assíncrono entre o Modelo Lógico e o Modelo Topológico.

A Figura 32 ilustra o exemplo de especificação do mecanismo de comunicação entre o Modelo Lógico e o Modelo Topológico. A transição *Encontrar Mapa de Silent Hill* é habilitada se existir uma ficha no lugar P0 e uma ficha no lugar *Plataforma de observação*,

¹ Uma rede de Petri marcada conhecida como grafo de estado é equivalente a um grafo de estado que representa um autômato e que está em apenas um estado por vez se, e somente se, ele contém exatamente uma ficha localizada em um de seus lugares.

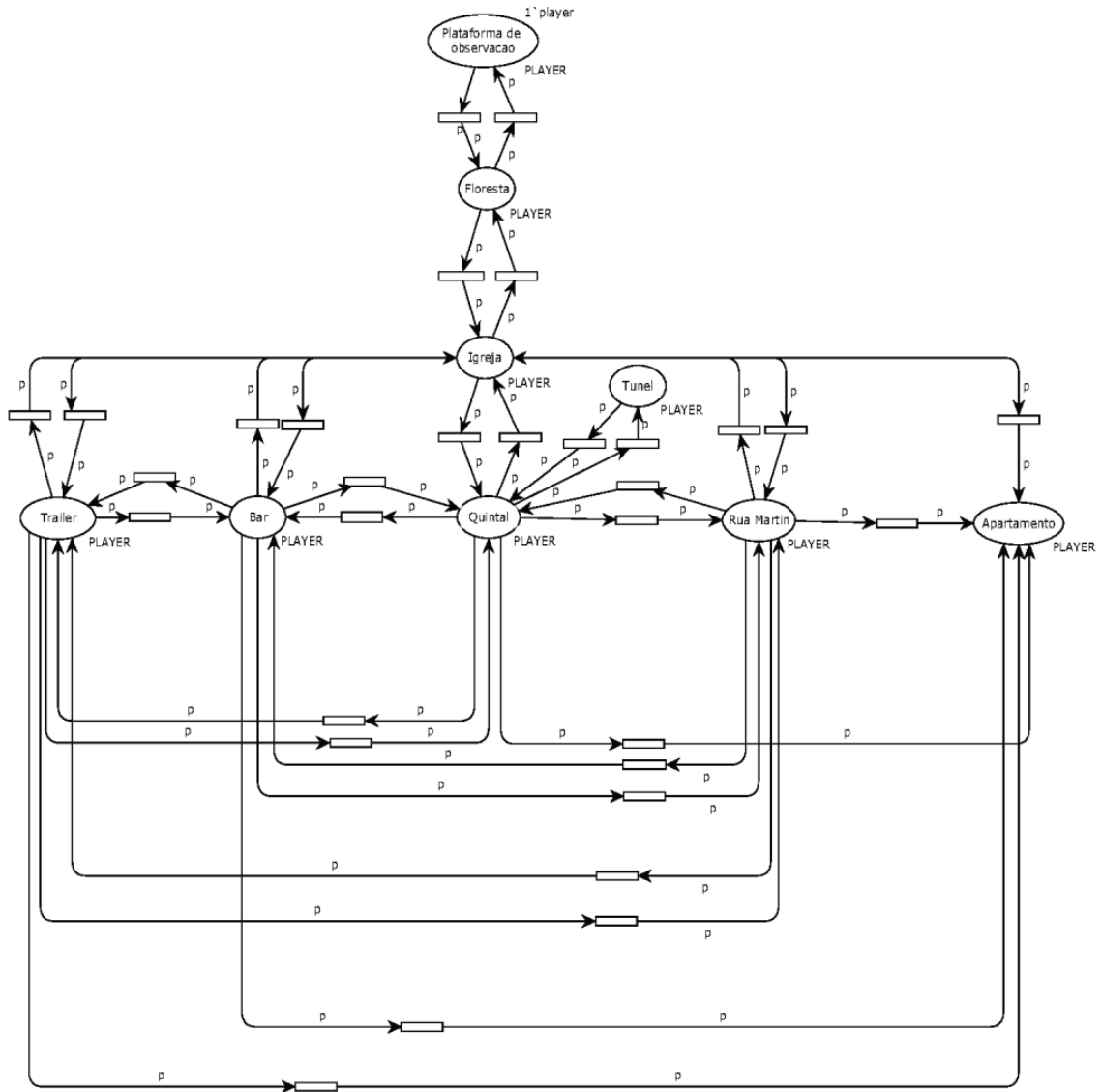


Figura 31 – Rede de Petri colorida do modelo topológico do primeiro nível do jogo Silent Hill II (Fonte: Barreto e Julia (2020b))

como ilustrado na Figura 32(b). Depois que a transição é disparada, são produzidas 3 fichas: uma ficha retorna para o lugar *Plataforma de observação*, a segunda é produzida em *P1* e a última em *C1*, liberando a área *Floresta* (Figura 32(c)).

Assim como neste exemplo, em qualquer outra área que necessitar ser desbloqueada mediante a realização de alguma atividade que satisfaça uma condição, é possível utilizar o mecanismo de comunicação assíncrono ilustrado pela Figura 32 para desbloquear a área desejada. No trabalho de Barreto e Julia (2014) os autores apresentaram um modelo no CPN Tools utilizando o conceito de *fusion places* para realizar a comunicação entre as atividades. A Figura 33 mostra como é representado tal mecanismo de comunicação. Do lado esquerdo da Figura 33 está o Modelo Lógico, enquanto do outro lado está o Modelo Topológico, assim como na Figura 32. Desta forma, evita-se que vários arcos se sobreponham pelo modelo, simplificando a visualização do modelo e permitindo que haja uma comunicação entre o Modelo Lógico e o Modelo Topológico.

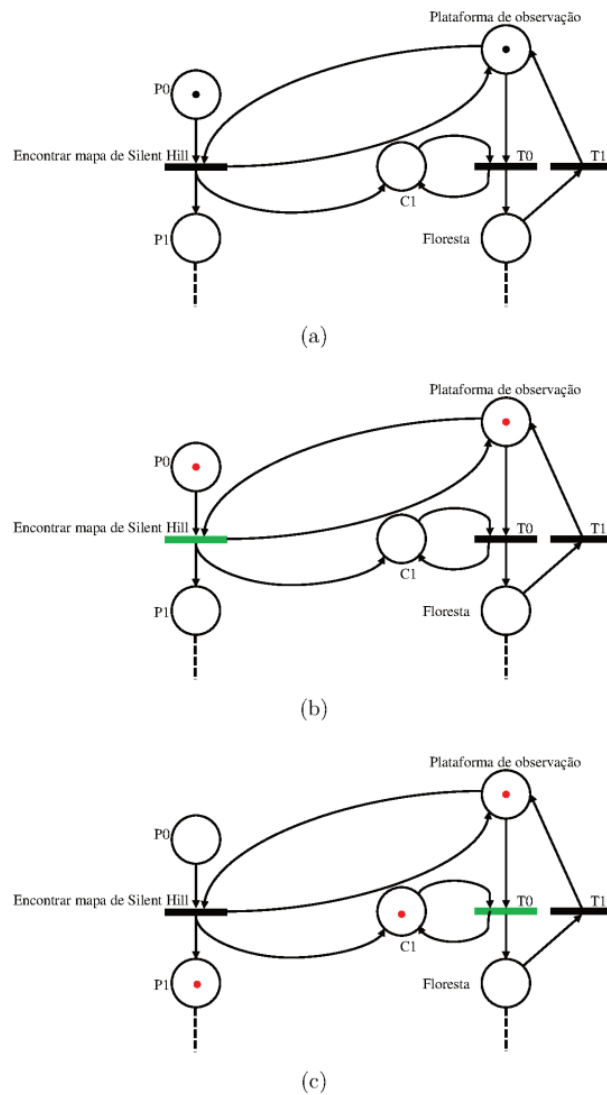


Figura 32 – Exemplo da comunicação entre modelo Lógico e modelo Topológico (Fonte: Barreto e Julia (2020b))

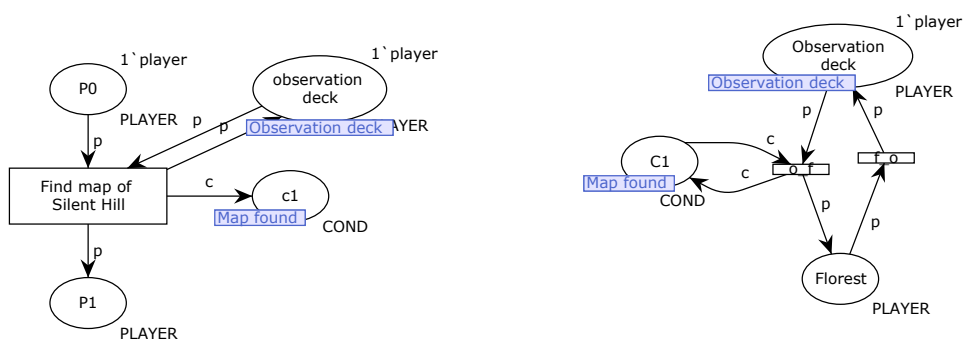


Figura 33 – Comunicação entre modelo lógico e topológico utilizando o conceito de *fusion places*.

As Figuras 34 e 35 ilustram o Modelo Lógico e o Modelo Topológico do primeiro nível do jogo *Silent Hill II* apresentado em (BARRETO; FREITAS; JULIA, 2018) utilizando redes de Petri Coloridas. A Figura 34 corresponde ao Modelo Lógico, com toda as ativi-

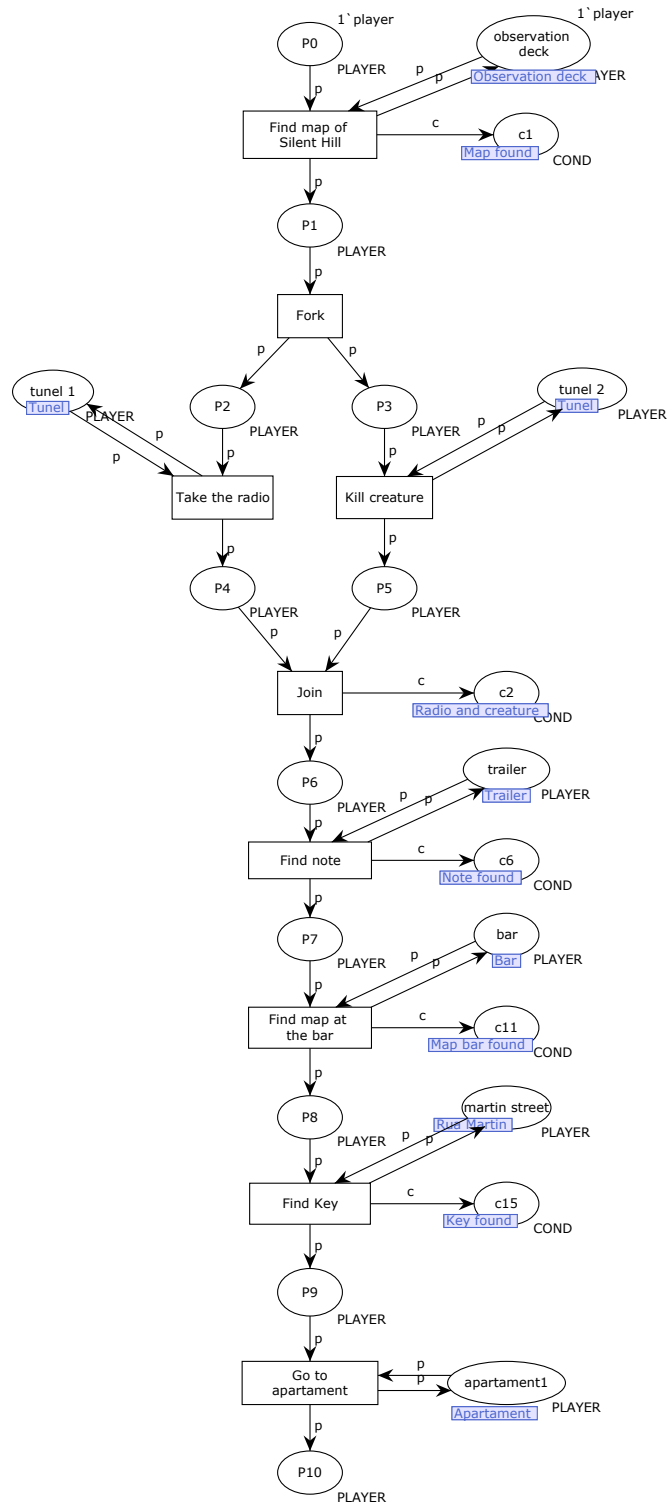


Figura 34 – Rede de Petri Colorida do Modelo Lógico do primeiro nível do jogo Silent Hill II.

dades do nível do jogo representadas por meio das transições. Já a Figura 35 corresponde ao Modelo Topológico, onde cada lugar da rede corresponde a uma área significativa do mapa do jogo.

No Modelo Lógico feito com redes de Petri Coloridas representado pela Figura 34 é

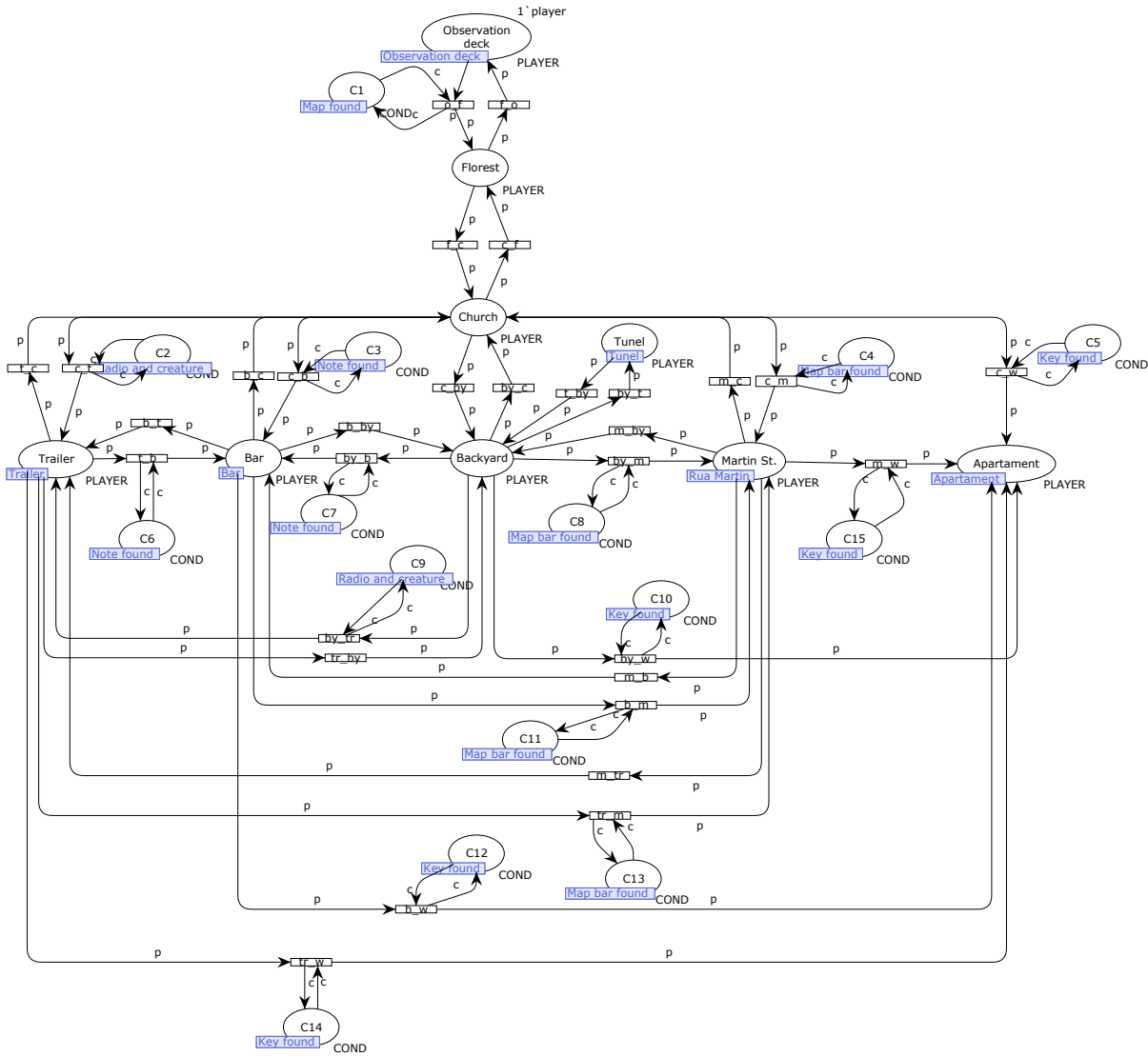


Figura 35 – Rede de Petri Colorida do Modelo Topológico do primeiro nível do jogo Silent Hill II.

possível perceber que a ficha inicial é do tipo $PLAYER\ 1'(p1, 0.5)@0.0$, em que $p1$ pertence à cor P e o segundo parâmetro iniciado com 0.5 é o nível de experiência inicial do jogador; o valor $@0.0$ ao final da ficha significa o tempo adicional inicial que no caso é 0. Todos os lugares do modelo são do tipo $PLAYER$ e os arcos poderão receber os valores do mesmo tipo.

Existem também funções de tempo em cada atividade realizada no modelo. A função $aTime(e)$ recebe o nível de experiência do personagem e utiliza a função $exponential(e)$ para retornar quanto tempo foi gasto para realizar aquela atividade baseado no nível de experiência do jogador naquela atividade. Esta função retornará um valor aleatório e foi escolhida pois é uma das funções de distribuição aleatória mais utilizadas para simular tempos entre eventos na maioria dos problemas de simulação (AALST; STAHL, 2011).

No Modelo Topológico feito com redes de Petri Coloridas ilustrado na Figura 35 também é iniciado com a ficha inicial do tipo $PLAYER\ 1'(p1, 0.5)@0.0$, assim como no Modelo

Lógico. O lugar inicial é na *Plataforma de Observação*, pois é necessário encontrar o mapa para poder acessar a área da *Floresta*. Diferente do Modelo Lógico, no Modelo Topológico é utilizado a função *mpTime()* que retorna uma distribuição uniforme entre 0.1 e 1.0. Esta função é utilizada para gerar um valor para somar ao tempo atual do jogador e representar o tempo que foi gasto para passar pela área significativa.

Para a análise do modelo, Barreto, Freitas e Julia (2018) utilizaram a ferramenta de análise do CPN Tools chamada *state space*. A análise, ilustrada na Figura 36 mostrou que é possível verificar no modelo global um conjunto de propriedades que garantem a corretude do modelo, em termos de jogabilidade. Por exemplo, com o relatório de análise gerado pelo CPN Tools é possível identificar que o modelo analisado não possui duplicação de fichas e que todas as transições da rede podem ser disparadas, o que significa que todas as atividades do jogo podem ser executadas pelo jogador. Como conclusão, os autores mostraram que o modelo pode ser considerado *sound*, uma vez que ele é vivo e 1-limitado. Para o cenário de jogos, o modelo ser *sound* significa que eventualmente, o jogador poderá acessar todas as atividades que estão distribuídas no modelo, garantindo a consistência das atividades do nível de jogo, bem como a boa construção do mapa topológico. A Figura 37 mostra as replicações das simulações do modelo *singleplayer* com adição de tempo nas atividades apresentado em (BARRETO; FREITAS; JULIA, 2018). Em média, as replicações gastaram um total de 63,53 unidades de tempo para finalizar o cenário de jogo no modelo *singleplayer*. Este tempo gasto nas simulações em termos de análise de jogo, mostra ao analista quanto tempo será gasto em média para realizar as atividades propostas e passar pelas áreas significativas do mapa de jogo onde se encontram as atividades a serem realizadas. Realizar este tipo de simulação faz parte do método de análise, uma vez que é necessário saber o tempo de jogo médio das quests propostas em jogos *singleplayer*.

4.2 Modelo SIMIO

Baseado no modelo apresentado em Barreto, Freitas e Julia (2018), o trabalho de Nedopetalski et al. (2024) propõe um método de modelagem 3D do mesmo cenário utilizando o *software* SIMIO. Para criar um modelo 2D/3D que fosse similar ao modelo de rede de Petri colorida, Mota et al. (2017) apresenta em seu trabalho a equivalência entre os componentes do SIMIO e das redes de Petri. Os autores sugerem que os componentes de uma rede de Petri colorida podem ser traduzidos para um modelo 2D/3D utilizando o SIMIO sem perda de significado, mesmo não havendo relação direta entre as ferramentas. Desta forma, é possível criar um modelo 2D/3D baseado em um modelo em rede de Petri colorida e que ambos sejam equivalentes.

Para entender esta equivalência entre componentes, é possível utilizar uma parte do Modelo *singleplayer* para servir de exemplo, ilustrados nas Figuras 38 e 39.

Boundedness Properties		

Best Integer Bounds	Upper	Lower
Silent_Hill'A1 1	1	0
Silent_Hill'A2 1	1	0
Silent_Hill'Apartamento 1	1	0
Silent_Hill'Bar 1	1	0
Silent_Hill'C10 1	1	0
Silent_Hill'C1 1	1	0
Silent_Hill'C11 1	1	0
Silent_Hill'C12 1	1	0
Silent_Hill'C13 1	1	0
Silent_Hill'C14 1	1	0
Silent_Hill'C15 1	1	0
Silent_Hill'C2 1	1	0
Silent_Hill'C3 1	1	0
Silent_Hill'C4 1	1	0
Silent_Hill'C5 1	1	0
Silent_Hill'C6 1	1	0
Silent_Hill'C7 1	1	0
Silent_Hill'C8 1	1	0
Silent_Hill'C9 1	1	0
Silent_Hill'Floresta 1	1	0
Silent_Hill'Igreja 1	1	0
Silent_Hill'F0 1	1	0
Silent_Hill'F10 1	1	0
Silent_Hill'F1 1	1	0
Silent_Hill'F2 1	1	0
Silent_Hill'F3 1	1	0
Silent_Hill'F4 1	1	0
Silent_Hill'F5 1	1	0
Silent_Hill'F6 1	1	0
Silent_Hill'F7 1	1	0
Silent_Hill'F8 1	1	0
Silent_Hill'F9 1	1	0
Silent_Hill		
'Plataforma_de_observacao 1	1	0
Silent_Hill'Quintal 1	1	0
Silent_Hill'Rua Martin 1	1	0
Silent_Hill'Trailer 1	1	0
Silent_Hill'Tunel 1	1	0
Silent_Hill'apartamento2 1	1	0
Silent_Hill'bar 1	1	0
Silent_Hill'c1 1	1	0
Silent_Hill'c11 1	1	0
Silent_Hill'c15 1	1	0
Silent_Hill'c2 1	1	0
Silent_Hill'c6 1	1	0
Silent_Hill		
'plataforma_de_observacao 1	1	0
Silent_Hill'Rua Martin 1	1	0
Silent_Hill'trailer 1	1	0
Silent_Hill'tunel 1 1	1	0
Silent_Hill'tunel 2 1	1	0

Home Properties

Home Markings
All

Liveness Properties

Dead Markings
NoneDead Transition Instances
NoneLive Transition Instances
All

Figura 36 – Resultados da análise das propriedades de limitabilidade e vivacidade

Simulation no.: 1	Simulation no.: 6
Steps.....: 93	Steps.....: 70
Model time....: 62.9249769971	Model time....: 41.1863356298
Stop reason....: No more enabled transitions!	Stop reason....: No more enabled transitions!
Time to run simulation: 0 seconds	Time to run simulation: 0 seconds
Simulation no.: 2	Simulation no.: 7
Steps.....: 99	Steps.....: 72
Model time....: 64.9958197461	Model time....: 49.2293575717
Stop reason....: No more enabled transitions!	Stop reason....: No more enabled transitions!
Time to run simulation: 0 seconds	Time to run simulation: 0 seconds
Simulation no.: 3	Simulation no.: 8
Steps.....: 104	Steps.....: 72
Model time....: 76.4852400115	Model time....: 51.7992009044
Stop reason....: No more enabled transitions!	Stop reason....: No more enabled transitions!
Time to run simulation: 0 seconds	Time to run simulation: 0 seconds
Simulation no.: 4	Simulation no.: 9
Steps.....: 83	Steps.....: 127
Model time....: 50.5384825856	Model time....: 74.9692379343
Stop reason....: No more enabled transitions!	Stop reason....: No more enabled transitions!
Time to run simulation: 0 seconds	Time to run simulation: 0 seconds
Simulation no.: 5	Simulation no.: 10
Steps.....: 152	Steps.....: 123
Model time....: 85.6253770888	Model time....: 77.5787499602
Stop reason....: No more enabled transitions!	Stop reason....: No more enabled transitions!
Time to run simulation: 0 seconds	Time to run simulation: 0 seconds

Figura 37 – Replicações da simulação no modelo *singleplayer* com tempo.

A Figura 38 ilustra as atividades realizadas na área significativa *Túnel*. Nela acontecem duas atividades em paralelo: “Pegar o rádio” e “Matar a criatura”. Inicialmente uma ficha do tipo *player* é criada no lugar *P1*, que leva diretamente para as atividades paralelas. Ao final, a condição de *rádio e criatura* é disparada na atividade *Join*, sinalizando que

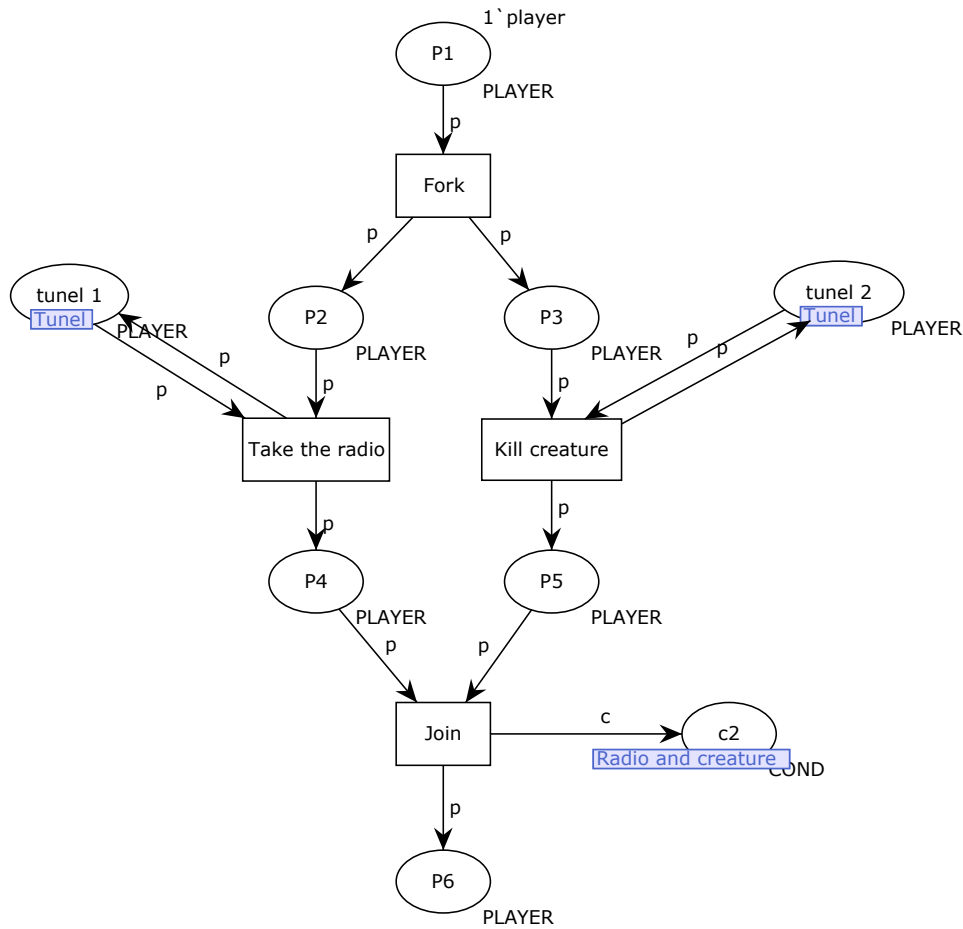


Figura 38 – Exemplo de rede de Petri Colorida.

ambas as atividades já foram concluídas. Todos os arcos do modelo de exemplo da Figura 38 utilizam da variável p do tipo *PLAYER*, com exceção do arco que liga a transição *Join* com o lugar $c2$. Além disso as atividades utilizam da área significativa *Tunel* para completar as tarefas. Se o jogador não estiver na área correta, não é possível disparar as transições das atividades.

Já a Figura 39 ilustra as atividades realizadas área significativa *Tunel* do jogo *Silent Hill II* com a visão do SIMIO. Diferente do modelo de rede de Petri colorida, neste modelo apenas o Modelo Topológico é representado visualmente, enquanto o Modelo Lógico é executado ao passar pelas áreas significativas. Isso se deve ao fato do uso do objeto do tipo *Server* que, por padrão, possui várias possibilidades de disparo de processos facilitando a criação da lógica que trará sentido ao objeto. Para criar esta lógica, dois processos foram criados para reproduzirem o comportamento da rede de Petri colorida no SIMIO: *SortearCaso* e *Process_Tempo_Tunel*. Além disso, os atributos do jogador como a experiência por exemplo, são armazenados em variáveis no sistema, podendo ser utilizadas e alteradas ao realizar as atividades.

Assim que o jogador entra na área significativa *Tunel*, o processo *SortearCaso* (ilustrado na Figura 40) é disparado iniciando a lógica do objeto. Nele são configurados todas

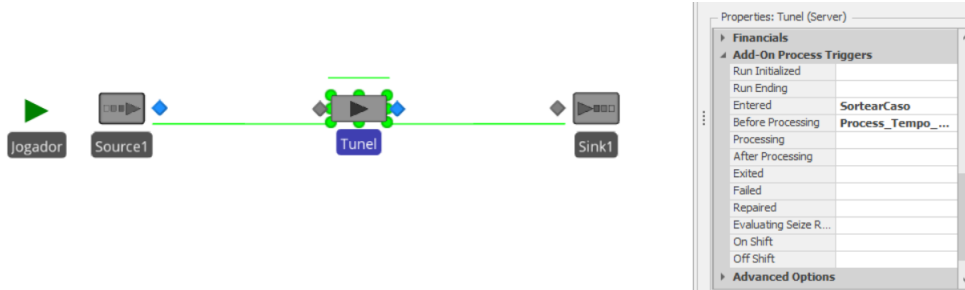


Figura 39 – Exemplo de modelo no SIMIO equivalente a uma rede de Petri Colorida.

as possibilidades que o jogador pode enfrentar ao executar ambas as atividades (“Pegar o rádio” e “Matar a criatura”). É sorteado de forma aleatória uma das possibilidades de resultado para simular a execução das atividades. Cada opção modelada simula um comportamento esperado que pode acontecer durante a execução das atividades paralelas. Dentre estas opções estão: pegar o rádio mas não conseguir matar a criatura, conseguir matar a criatura mas não encontrar o rádio, não conseguir encontrar o rádio nem matar a criatura, entre outras.

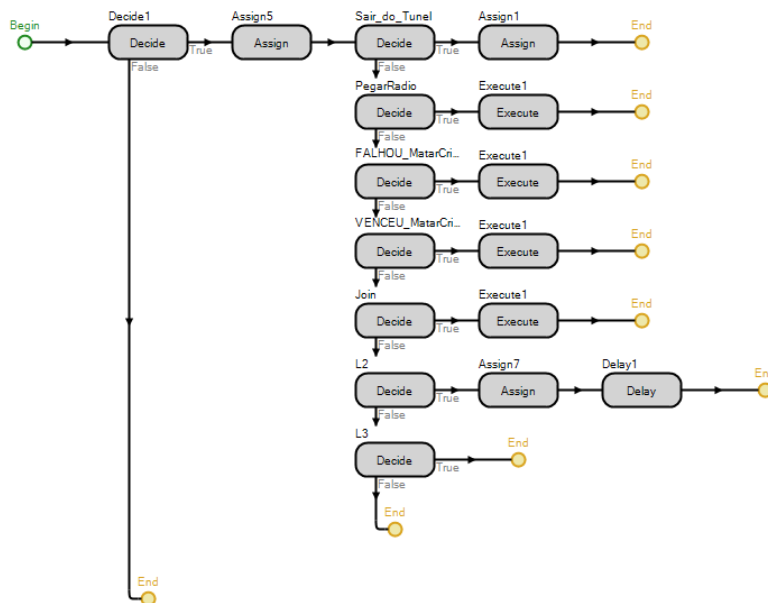


Figura 40 – Processo *SortearCaso* no SIMIO.

Logo após a execução do processo *SortearCaso*, é disparado o processo *Process_Tempo_Tunel*. A Figura 41 ilustra este processo no qual é calculado quanto tempo será gasto para realizar a atividade determinada baseado na escolha aleatória realizada pelo processo *SortearCaso*. É necessário um processo específico para estas atividades pois além de serem paralelas, há a possibilidade de falha durante a execução, acrescentando uma complexidade maior à simulação.

Assim que o processo *Process_Tempo_Tunel* é finalizado, o tempo adicional da atividade é somado ao tempo atual da simulação e caso a atividade seja completada com

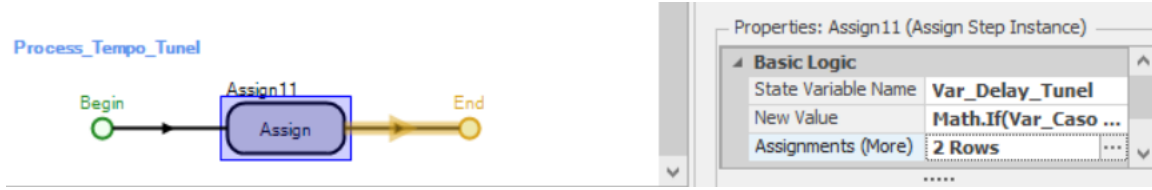


Figura 41 – Processo *Process_Tempo_Tunel* no SIMIO.

sucesso, uma variável de controle é utilizada para armazenar esta informação e sinalizar a conclusão das atividades, assim como é feito no modelo de rede de Petri Colorida. Para armazenar esta informação é utilizado o *step Assign*. Nele é possível alterar o valor de uma ou mais variáveis de ambiente do SIMIO, como é ilustrado na Figura 41.

Para criar o Modelo Lógico completo em 2D do primeiro nível do jogo *Silent Hill II* no SIMIO ilustrado na Figura 42, foi necessário repetir os passos citados no exemplo acima para as outras atividades, ou seja, foram criados outros *Servers* para cada atividade e atribuído a função de tempo lógica (*exponential()*), assim como foi implementado na rede de Petri colorida. Como é apenas o modelo lógico, não há restrição de local para verificar.

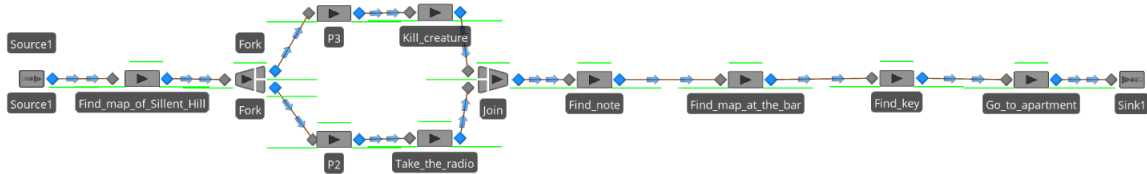


Figura 42 – Modelo lógico 2D *singleplayer* feito utilizando SIMIO (FONTE: (NEDOPE-TALSKI et al., 2024))

Para iniciar o modelo, um objeto *Source* foi adicionado para criar uma instância da entidade *jogador* na simulação. As atividades *Find_map_of_Silent_Hill*, *Kill_creature*, *Take_the_radio*, *Find_note*, *Find_map_at_the_bar*, *Find_key* e *Go_to_apartment* foram criadas como objetos do tipo *Server*, pois por padrão possuem várias opções de disparo de processos ao serem executados. Para realizar as atividades em paralelo, os objetos *Fork* e *Join* foram utilizados para duplicar e sincronizar as entidades produzidas durante a simulação. Ao completar as atividades propostas pelo jogo, um objeto *Sink* é utilizado para consumir a entidade jogador e finalizar o cenário de jogo (a simulação).

Assim como o Modelo Lógico, também foi criado no SIMIO um Modelo Topológico para representar a rede de Petri Colorida equivalente. A Figura 43 representa o modelo 2D feito no SIMIO, que é equivalente ao Modelo Topológico apresentado na Figura 35.

O modelo criado utilizando rede de Petri colorida e ilustrado na Figura 35, corresponde a uma rede de Petri do tipo grafo de estados. No modelo são representados os caminhos possíveis entre áreas de jogos, assim como as condições necessárias para entrar em cada local do mapa. Para criar as restrições dos locais, foram criadas variáveis de controle para cada área significativa com restrição. Elas foram configuradas para que antes de

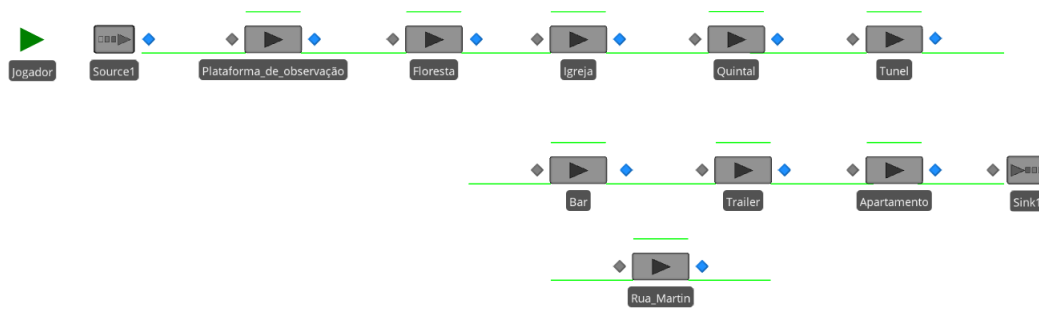


Figura 43 – Modelo Topológico 2D *singleplayer* feito utilizando SIMIO

sair da atividade, a condição necessária seja concedida, como ilustrado na Figura 44. No caso deste exemplo, ao sair da área significativa *Plataforma de Observação*, o jogador encontrará o mapa e irá acessar a próxima área. O modelo 2D ilustrado na Figura 43 é equivalente na versão SIMIO da Figura 35. Em ambos modelos são representados apenas os locais onde os jogadores poderão passar, sem considerar as atividades envolvidas.

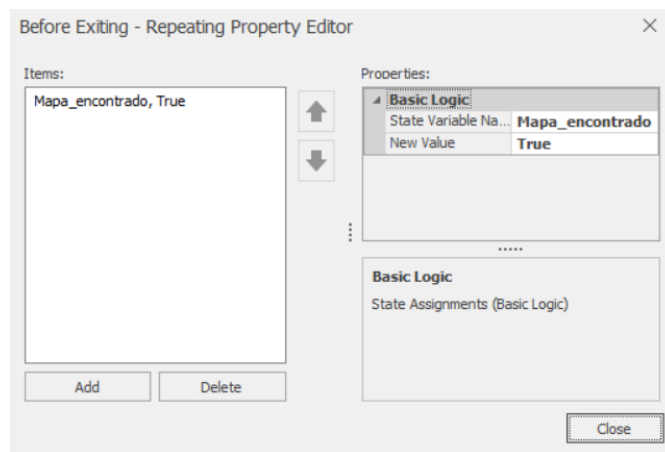


Figura 44 – Configuração da variável de controle da área *Plataforma de Observação*.

O Modelo Lógico 2D servirá como um “esqueleto” para o modelo 3D. O diferencial do SIMIO é a possibilidade de criar modelos e simular o funcionamento do jogo em um ambiente 3D. Dessa forma é possível interpretar o modelo de jogo até mesmo por aqueles que não são especialistas (como os usuários finais do jogo) nas abordagens formais utilizadas para a especificação das fases de *game* e *level design*.

No SIMIO há uma série de recursos, símbolos, desenhos e texturas para que seja possível criar um ambiente 3D e representar o cenário desejado. Apesar de ser possível importar imagens para criar as texturas desejadas, é importante salientar que os modelos 3D apresentados neste trabalho são meramente ilustrativos e buscam apenas representar o fluxo das atividades, não tentando se assemelhar com a aparência real do jogo utilizado.

Ao construir o Modelo Topológico 3D foi utilizado o recurso disponível nos objetos do SIMIO chamado *Select From List* que é responsável por definir o nó de destino para as entidades a partir do local em que se encontram no momento. Para fazer isso, uma lista

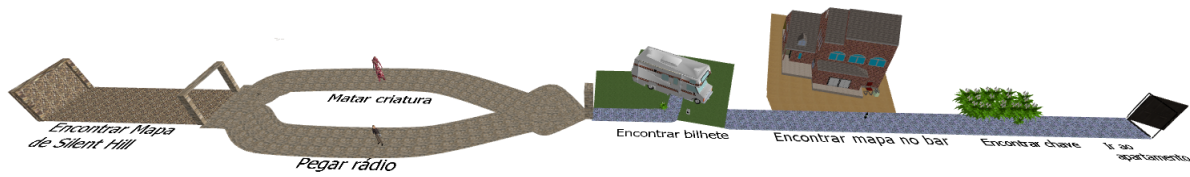


Figura 45 – Modelo Lógico 3D *singleplayer* feito utilizando SIMIO

dos possíveis locais que o jogador poderá ir durante a simulação é criada. A Figura 46 ilustra a lista dos locais da área significativa *Igreja*. Já a Figura 47 ilustra como esta lista pode ser atribuída no objeto *Server* que representa a área *Igreja* para a seleção de qual será o próximo local que o jogador irá acessar.

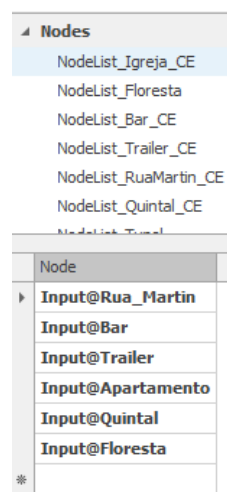


Figura 46 – Lista dos possíveis nós de saída do local “Igreja” no modelo topológico 3D.

Na configuração do *output* do *Server* da área significativa *Igreja*, a opção *Entity Destination* mostra algumas escolhas para selecionar qual o modo de seleção das opções da lista. Para este modelo foi selecionado a opção *Select From List*, no qual é possível selecionar qualquer uma das listas criadas anteriormente. Como foi criado uma lista específica para cada área significativa, a lista *ListReference_Igreja* foi escolhida pois contém as possíveis áreas que o jogador poderá acessar a partir da *Igreja*.

Outra opção modificada nos *Servers* do modelo foi a *Selection Goal* que lista várias opções de escolhas de como deve ser selecionado o item da lista informada. Ao selecionar a opção *Random*, o próprio sistema durante a simulação irá selecionar de forma aleatória uma das opções da lista selecionada.

Na Figura 48 é ilustrado o modelo de simulação 3D do Mapa Topológico feito no software SIMIO que é baseado na rede de Petri colorida apresentada por Barreto e Julia (2014). Neste modelo 3D, um mapa ilustrativo mostra as áreas significativas onde o jogador poderá transitar. É importante ressaltar que as atividades do jogo são feitas em locais específicos do mapa e configuradas internamente no SIMIO por meio dos processos. Logo, o Modelo Topológico e o Modelo Global não terão diferenças visuais aparentes, uma

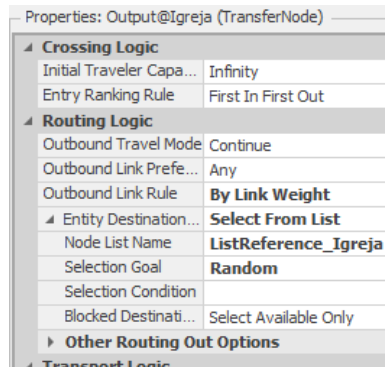


Figura 47 – Propriedades do nó de saída do local “Igreja” no Modelo Topológico 3D.

vez que ambos os modelos ilustrarão apenas as áreas do mapa de forma visual.



Figura 48 – Modelo 3D *singleplayer* criado utilizando SIMIO

Para ter um melhor entendimento entre os modelos utilizando redes de Petri coloridas e SIMIO, uma comparação das atividades será apresentada a seguir.

De início ambos os modelos apresentam seus jogadores na área significativa *Plataforma de Observação* ilustrados nas Figuras 49 e 50. É necessário que nos dois modelos os jogadores encontrem o mapa de Silent Hill para acessar as próximas áreas. Para isso, ao executar a atividade, a condição de *mapa encontrado* é adquirida. No modelo CPN da Figura 50 essa condição é dada no lugar *c1*, enquanto no modelo SIMIO da Figura 49 essa condição acontece quando o gatilho *Before Exiting* do *Server Plataforma_de_Observação* é acionado, alterando o valor da variável *Mapa_encontrado*, como ilustrado na Figura 44.

Para executar a próxima atividade necessária para completar o nível do jogo, é preciso chegar na área significativa *Tunel* para poder fazer as atividades *Matar criatura* e *Encontrar Rádio*. Porém, há três áreas que necessitam ser atingidas antes de chegar na área *Tunel*. Estas áreas são *Floresta*, *Igreja* e *Quintal*. Nelas não há nenhuma condição específica para poder ser acessada, tampouco alguma atividade para ser feita. Portanto o jogador poderá transitar livremente entre essas áreas. As Figuras 51, 52 e 53 ilustram

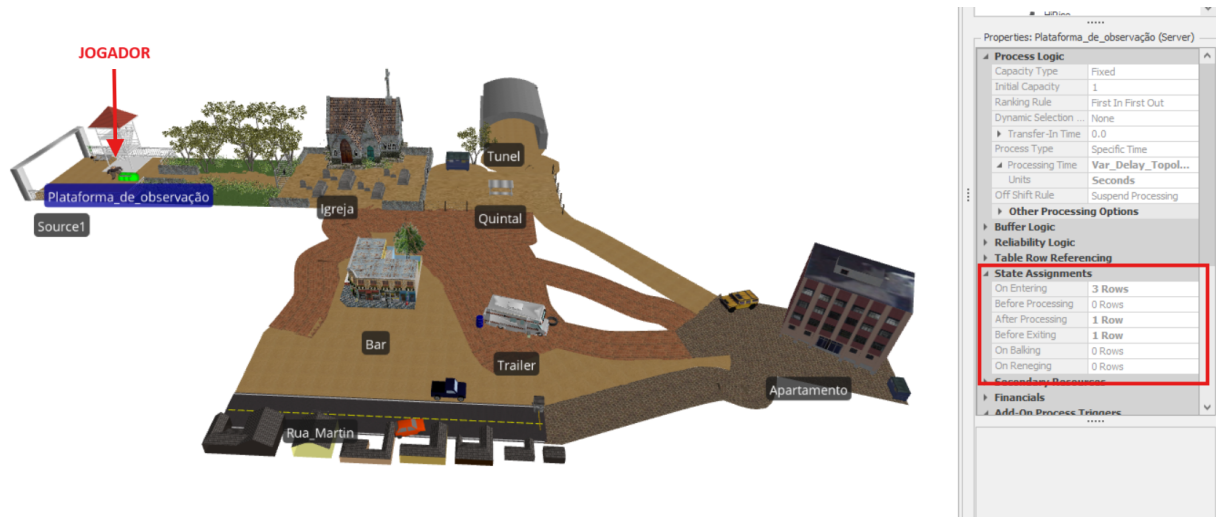


Figura 49 – Exemplo da Atividade inicial do Modelo Global no SIMIO.



Figura 50 – Exemplo da Atividade inicial do Modelo Global em rede de Petri Colorida.

a passagem do jogador (sinalizado com uma seta vermelha indicando seu local atual) nas áreas *Floresta*, *Igreja* e *Quintal* durante a simulação do modelo global. As três áreas contém apenas o disparo do processo *Process_Tempo_Topologico* ao entrar para calcular o tempo gasto para passar pela área. É utilizado a função uniforme para gerar este tempo.



Figura 51 – Área Floresta do Modelo Global no SIMIO.

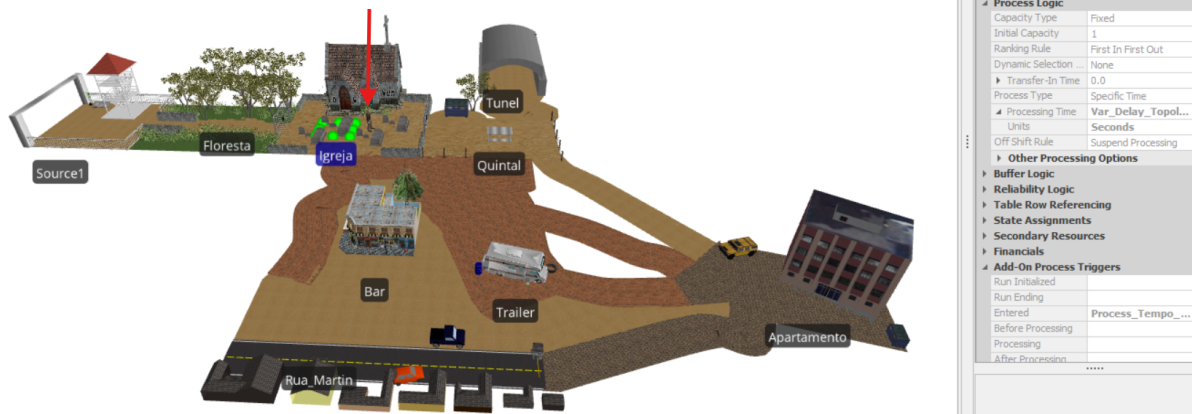


Figura 52 – Área Igreja do Modelo Global no SIMIO.

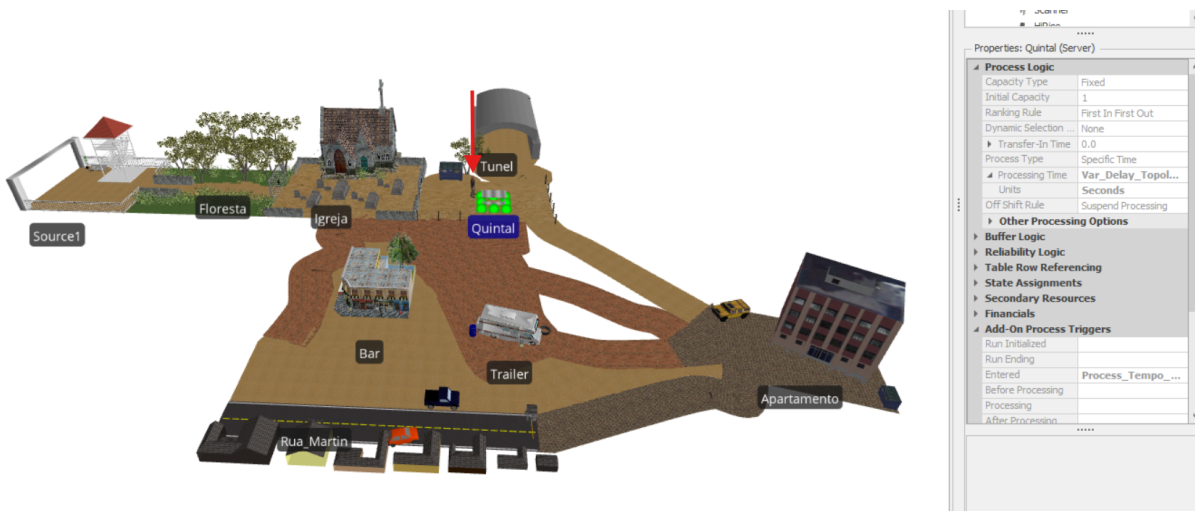


Figura 53 – Área Jardim do Modelo Global no SIMIO.

Já as Figuras 54, 55 e 56 ilustram a passagem do jogador no modelo CPN. Assim como no modelo do SIMIO, não há nenhuma restrição ou atividade que necessita ser realizada nestas áreas.



Figura 54 – Área Floresta do Modelo Global em rede de Petri Colorida.

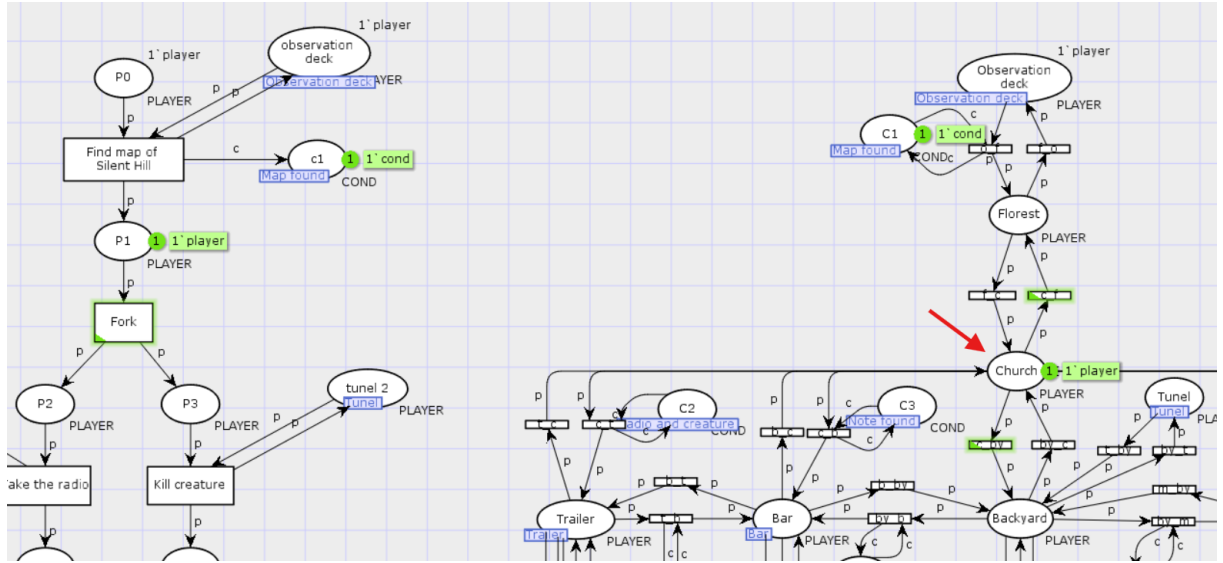


Figura 55 – Área Igreja do Modelo Global em rede de Petri Colorida.

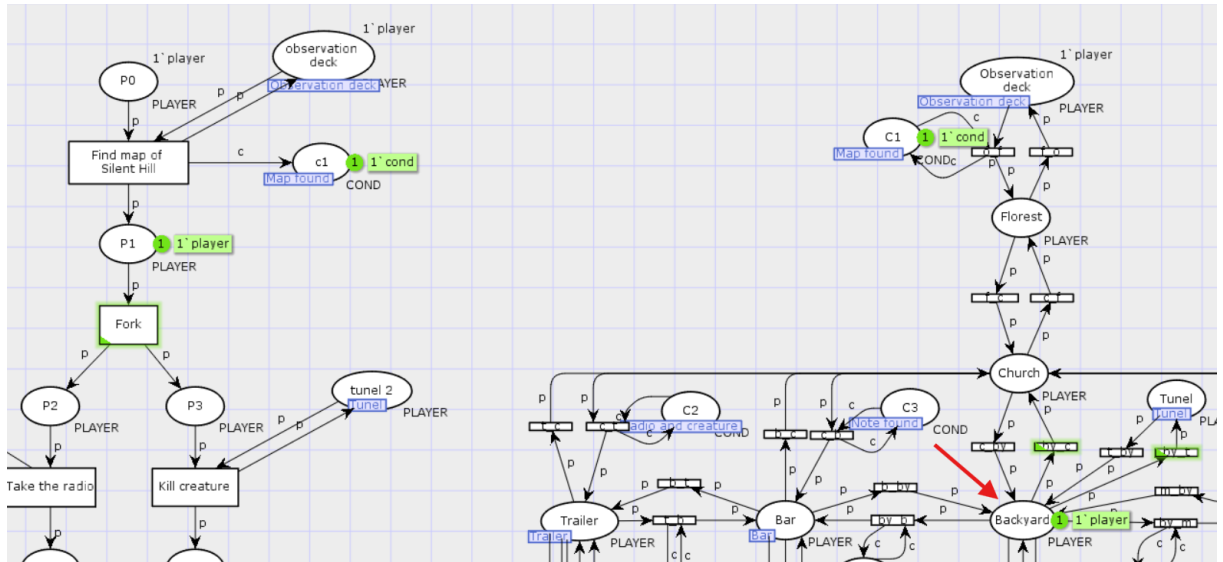


Figura 56 – Área Quintal do Modelo Global em rede de Petri Colorida.

Por fim, o jogador chega na área significativa *Tunel* onde poderá realizar as atividades *Matar criatura* e *Encontrar o rádio*. As Figuras 57 e 58 ilustram esta situação. No modelo do SIMIO, o processo *SortearCaso* é disparado assim que o jogador entra na área e o tempo é calculado logo em seguida. Já no modelo CPN, as atividades ficam habilitadas para serem disparadas. Ao completar a atividade, a simulação segue para as atividades restantes do mesmo modo como foi feito até o momento.

O modelo de jogo global no SIMIO, assim como no modelo CPN contém o modelo lógico e topológico em um único modelo final. Nele é possível ver todos os locais no qual o jogador pode transitar e também as atividades que pode realizar em cada área significativa do mapa. Ao simular este modelo, além de poder acompanhar visualmente onde o jogador está e o que está fazendo, o SIMIO apresenta como saída uma tabela de resultados. Para realizar a simulação no SIMIO, basta clicar no botão verde triangular

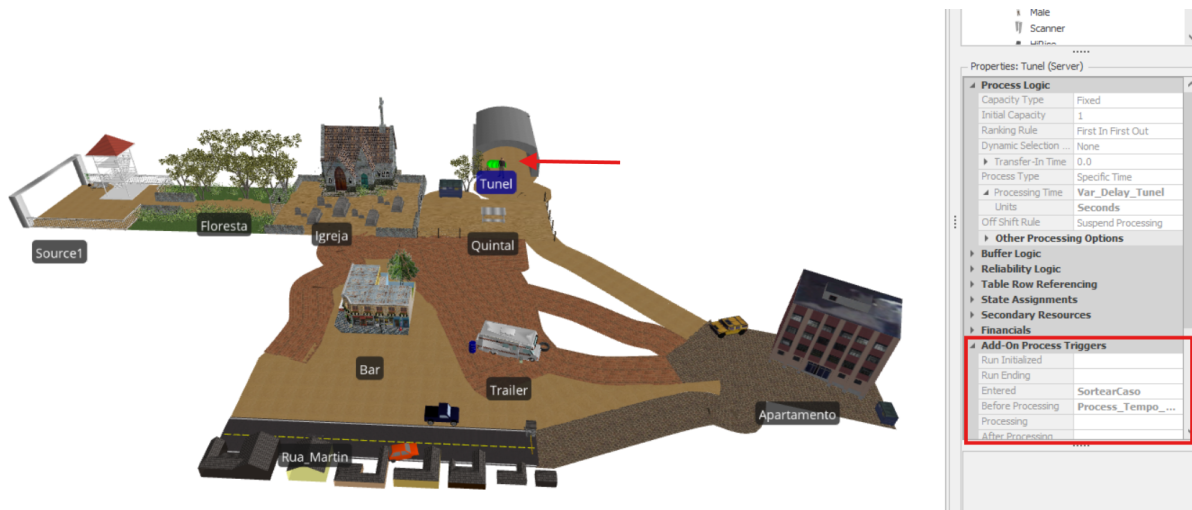


Figura 57 – Área Tunel do Modelo Global em rede de Petri Colorida.

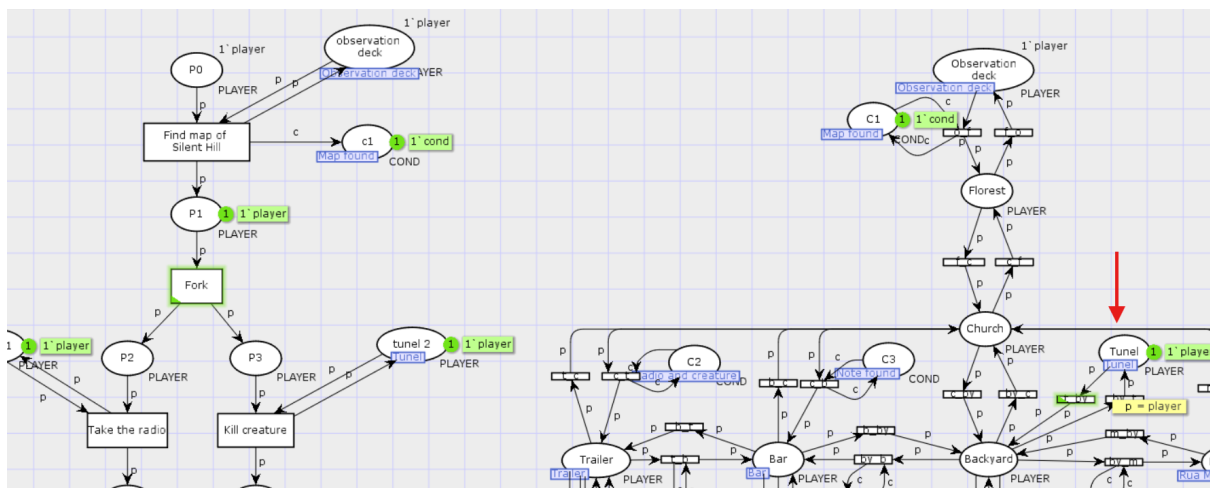


Figura 58 – Área Tunel do Modelo Global em rede de Petri Colorida.

no canto superior esquerdo da aplicação, ilustrado pela Figura 59.

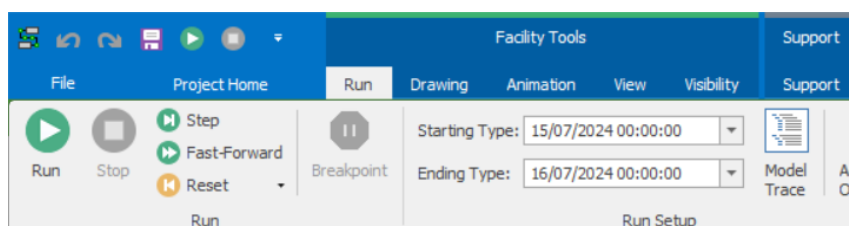


Figura 59 – Botão de simulação do SIMIO.

A tabela de resultados dispõe de vários dados, incluindo qual o momento da simulação que o jogador estava em cada local do mapa. Além disso, ela mostra qual é o tipo do objeto que aqueles dados estão relacionados e também o número de ocorrências que aquele objeto teve durante a simulação. A Figura 60 ilustra como os resultados da simulação são apresentados ao usuário. A partir destes resultados é possível realizar uma análise dos dados, não só numérica, mas também de jogabilidade do modelo e comparar com os

resultados obtidos por meio da simulação no CPN Tools.

Object Type	Object Name	Data Source	Category	Data Item	Statistic	Average Total
ModeEntity	Jogador	[Population]	Content	NumberInSystem	Average	0,0000
					Maximum	1,0000
				FlowTime	Average (Sec...	2,5016
			Throughput	NumberCreated	Maximum (Sec...	2,5016
					Minimum (Sec...	2,5016
				NumberDestroyed	Observations	1,0000
Server	Apartamento	[Resource]	ResourceState	TimeStarved	Total	1,0000
					Total	1,0000
					Average (Sec...	86,400,0000
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	86,400,0000
	Bar	[Resource]	ResourceState	TimeStarved	Average (Sec...	86,400,0000
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	86,400,0000
	Floresta	[Resource]	ResourceState	TimeStarved	Average (Sec...	86,400,0000
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	86,400,0000
	Igreja	[Resource]	ResourceState	TimeStarved	Average (Sec...	86,400,0000
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	86,400,0000
	Plataforma_de_obse...	[Resource]	ResourceState	TimeStarved	Average (Sec...	86,400,0000
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	86,400,0000
	InputBuffer	Throughput			NumberEntered	1,0000
					NumberExited	1,0000

Figura 60 – Tabela de resultados da simulação realizada no *software* SIMIO no cenário *singleplayer*.

Como resultado da simulação do Modelo Global no SIMIO, obteve-se o tempo mostrado na Figura 61 de 86,4 unidades de tempo. Este resultado ilustra o tempo gasto para passar pelas áreas significativas e também para realizar as atividades do nível proposto. Assim como Nedopetalski et al. (2024) mostrou em seu trabalho, o resultado obtido utilizando o *software* SIMIO é compatível com o resultado obtido utilizando o CPN Tools. Dessa forma, os modelos podem ser considerados semelhantes. Além disso, utilizar os dois modelos para realizar simulações pode trazer grandes resultados de análise tanto quantitativos quanto qualitativos para desenvolvedores e usuários finais.

[Resource]	ResourceState	TimeStarved	Average (Sec...	86,400,0000
			Occurrences	1,0000
			Percent	100,0000
			Total (Seconds)	86,400,0000

Figura 61 – Resultado da simulação realizada no *software* SIMIO no cenário *singleplayer*.

4.3 Considerações Finais

Os modelos apresentados neste capítulo resultaram no desenvolvimento deste trabalho. O modelo de rede de Petri colorida apresentado por Barreto e Julia (2014) mostrou que é possível simular um nível de jogo *singleplayer*, além de ter análises sobre a rede que

reforçam a corretude da mesma. Já o modelo apresentado por Nedopetalski et al. (2024), mostra que um modelo de rede de Petri colorida pode ser transformado em um modelo 3D sem perda de sentido e com resultados de simulação semelhantes, resultando em um maior entendimento sobre o que está acontecendo no modelo durante a execução da simulação passo a passo em 3D.

Este capítulo teve como principal resultado a publicação do artigo Nedopetalski et al. (2024) apresentado na conferência ITNG 2024: 21st International Conference on Information Technology-New Generations com o apoio da Universidade Federal de Uberlândia.

Os arquivos bases podem ser encontrado no repositório aberto.

Modelo *Multiplayer*

Além dos jogos *singleplayer*, existem os jogos *multiplayer* nos quais dois ou mais jogadores estão simultaneamente no mesmo local, fazendo as mesmas coisas e dividindo os mesmos recursos. Este fato influencia diretamente nas ações que podem ser tomadas pelos jogadores. Por exemplo, se dois jogadores participam de uma atividade que consiste em encontrar uma chave, os dois vão iniciar a atividade mas somente um dos dois vai finalizar a atividade (quem encontrar a chave primeiro); o outro jogador deverá então abandonar a atividade “encontrar chave” para, junto com o outro jogador, prosseguir no jogo. Portanto, a criação de cenários de jogos *multiplayers* é um desafio na criação de video games, uma vez que com mais de um jogador concorrendo às mesmas atividades, a dinâmica dos eventos pode representar cenários muito complexos do ponto de vista de um modelo de simulação a eventos discretos (ARAÚJO; ROQUE, 2009). Além disso, apesar de ser possível criar e verificar modelos *multiplayer* utilizando modelos de rede de Petri colorida (BARRETO; JULIA, 2020a), o entendimento para todos os envolvidos no processo de criação pode ficar comprometido, já que nem todos os usuários têm familiaridade com abordagens formais (como é o caso da teoria das redes de Petri), principalmente os usuários finais.

Portanto, neste capítulo será apresentado um modelo de rede de Petri colorida hierárquico *multiplayer*, e também um modelo 3D *multiplayer* equivalente baseado na ferramenta SIMIO. Resultados de simulação passo a passo no caso da ferramenta SIMIO ou por replicação no caso do CPN Tools serão apresentados e mostrarão as vantagens e desvantagens de cada abordagem.

5.1 Mecanismo de Pseudo-Disparo

Nesta seção será apresentado o mecanismo de pseudo-disparo e de comunicação entre o Modelo Lógico e o Modelo Topológico no CPN Tools e como ele é traduzido para o SIMIO. Este mecanismo é apresentado originalmente no trabalho de Barreto, Freitas e Julia (2018)

O Modelo Global Multiplayer é formado pelo Modelo Lógico Multiplayer, Modelo

Topológico Multiplayer e um mecanismo de comunicação. Uma vez que o Modelo Global é estabelecido, o cenário de jogo pode ser simulado e analisado como um todo. Para explicar a proposta de implementação do modelo de simulação global no caso multiplayer será utilizado um exemplo simples de jogo com três atividades no modelo lógico e duas áreas no modelo topológico, como ilustra a Figura 62. A Utilização do CPN Tools, além de fornecer ferramentas para o gerenciamento do mecanismo de comunicação, ainda mantém os dois modelos visualmente distintos.

No modelo de simulação da Figura 62, todos os lugares da rede pertencem à classe *PLAYER*. Esta classe é composta por três atributos principais: a identificação do jogador (representado por $p1$ e $p2$), uma lista de atributos de tamanho 2 que representa o status das atividades, e um indicador de experiência (e). Portanto, a ficha que representa o objeto jogador 1 é dada por $1(p1,[0,0],0.5)$. O lado esquerdo da Figura 62 ilustra o Modelo Lógico Multiplayer. Já o lado direito ilustra o Modelo Topológico Multiplayer. As atividades *A1* e *A2* só podem ser executadas uma única vez (por um dos jogadores), portanto as transições *Início de A1*, *Fim de A1*, *Início de A2* e *Fim de A2* correspondem à regiões de incerteza no modelo. Os lugares *M1*, *M2* e *M3* correspondem as áreas do mapa topológico.

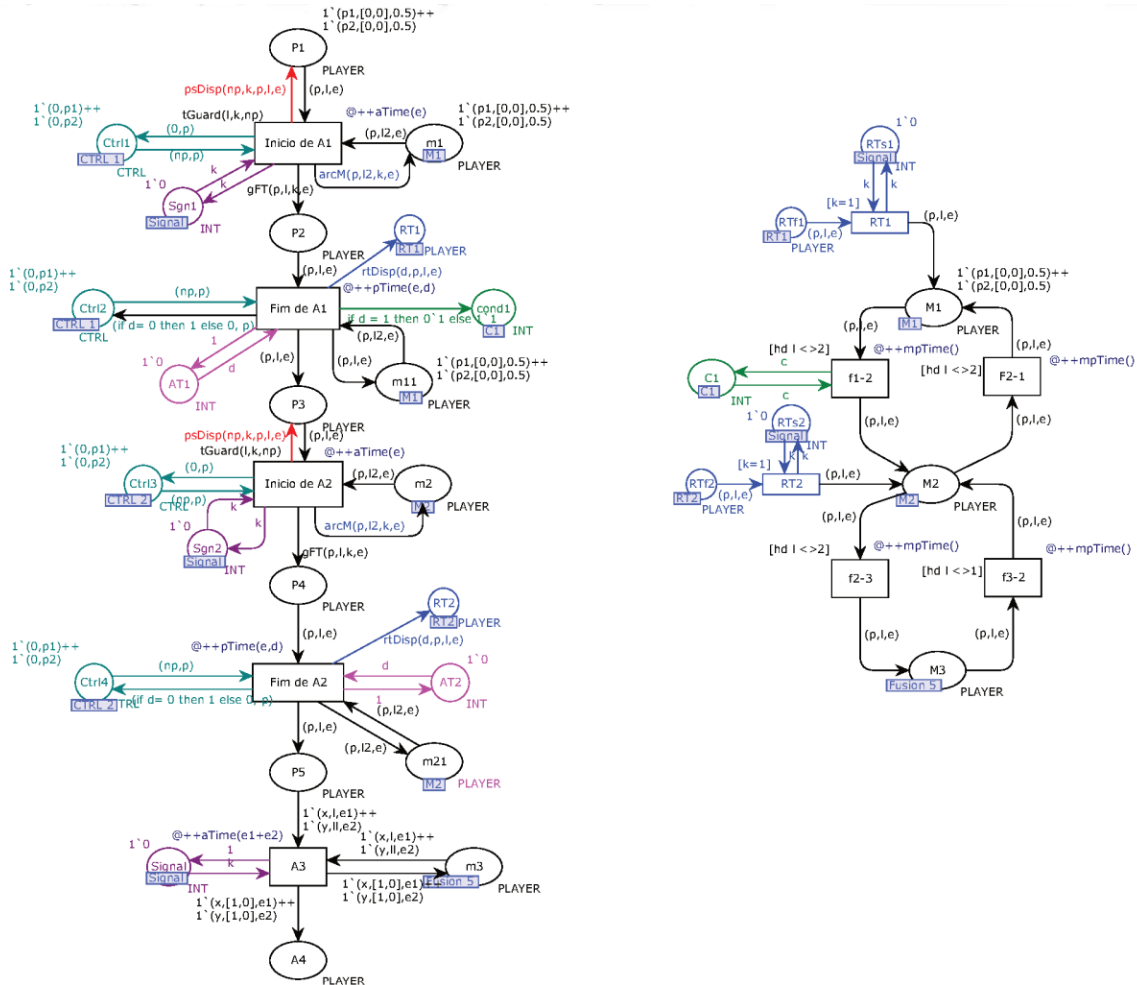


Figura 62 – Exemplo de um modelo de simulação global no caso multiplayer.

Para a implementação do Modelo Global Multiplayer no CPN Tools, o conceito de *fusion places* é utilizado. É possível observar na Figura 63 que é destacado o mecanismo de comunicação implementado no CPN Tools. Os lugares do mapa topológico estão marcados com uma *fusion tag* correspondente à área. Por exemplo, os lugares $M1$, $m1$ e $m11$ estão rotulados com a *fusion tag* $M1$, pois todos representam a área do mapa $M1$. O mesmo acontece com $RT1$ e $RTf1$, ambos pertencentes ao mesmo conjunto de fusão $RT1$. Os lugares que representam uma condição necessária para a liberação de uma área do mapa são $cond1$ e $C1$, ambos pertencentes ao conjunto de fusão $C1$.

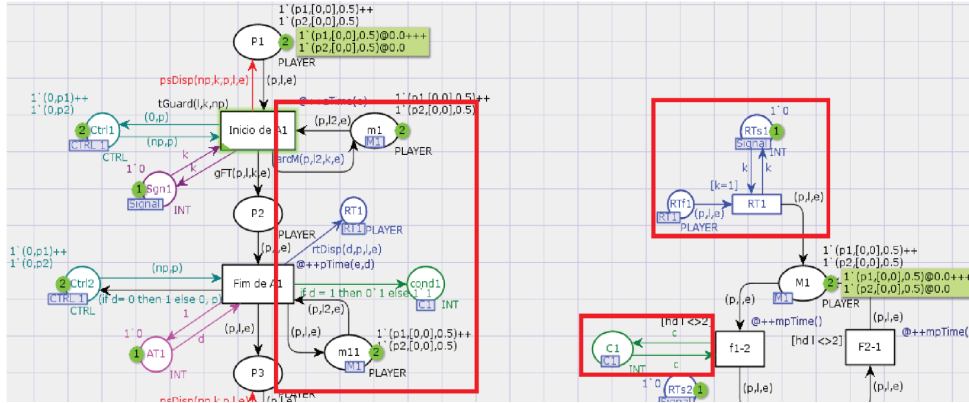


Figura 63 – Mecanismo de comunicação no Modelo Global Multiplayer.

Lugares especiais precisam ser adicionados à estrutura do modelo para que as variáveis sejam atualizadas a cada disparo de transição. Os lugares com prefixo *CTRL* atualizam a variável np , do tipo inteiro, que é responsável por permitir a execução e o cancelamento das atividades associadas aos pseudo-disparos. Os lugares *CTRL* são inicializados com o identificador do jogador e o valor de np (por exemplo, $1'(p1,[0,0],0.5)$), e são associados às transições de início e fim de cada atividade que pertence à região de incerteza do modelo, como ilustrado na Figura 63. Os lugares com prefixo *AT* atualizam a variável d e são associados apenas as transições de fim de cada atividade da região de incerteza. A variável d tem como objetivo indicar quando a transição é disparada com certeza, ou seja, quando a atividade é finalizada. Nas regiões de incerteza também são adicionados lugares com o prefixo *Sgn*, que atualizarão a variável k , responsável por indicar o status da última atividade da missão. Tais lugares são definidos como *fusion places* e fazem parte do mesmo conjunto de fusão, fazendo com que a variável k funcione como um tipo de variável global para o modelo.

Para o gerenciamento dos pseudo-disparos (disparos incertos) no CPN Tools, duas funções são necessárias no mecanismo de comunicação: *rtDisp()* e *arcM()*. A função *rtDisp()* é associada ao arco de entrada dos lugares de prefixo *RT*, com o objetivo de armazenar no lugar a ficha do jogador que executou o pseudo-disparo. Já a função *arcM()* tem como objetivo de gerar fichas apenas enquanto *quest* não for finalizada. A implementação dessas funções é dada como segue:

considerado). O mesmo é feito para as demais transições do modelo que são pseudo-disparadas.

O lugar RT no mecanismo de comunicação tem o objetivo de funcionar como um recurso para armazenar a ficha do jogador que executou o pseudo-disparo da transição. Armazenar essa informação é importante para a realização do cancelamento do pseudo-disparo ao final do nível de jogo. O lugar RT faz então a comunicação com o mapa topológico informando para qual jogador o cancelamento será aplicado. No cenário de jogo, o cancelamento do pseudo-disparo de uma transição não gera novas fichas no modelo lógico nem no modelo topológico.

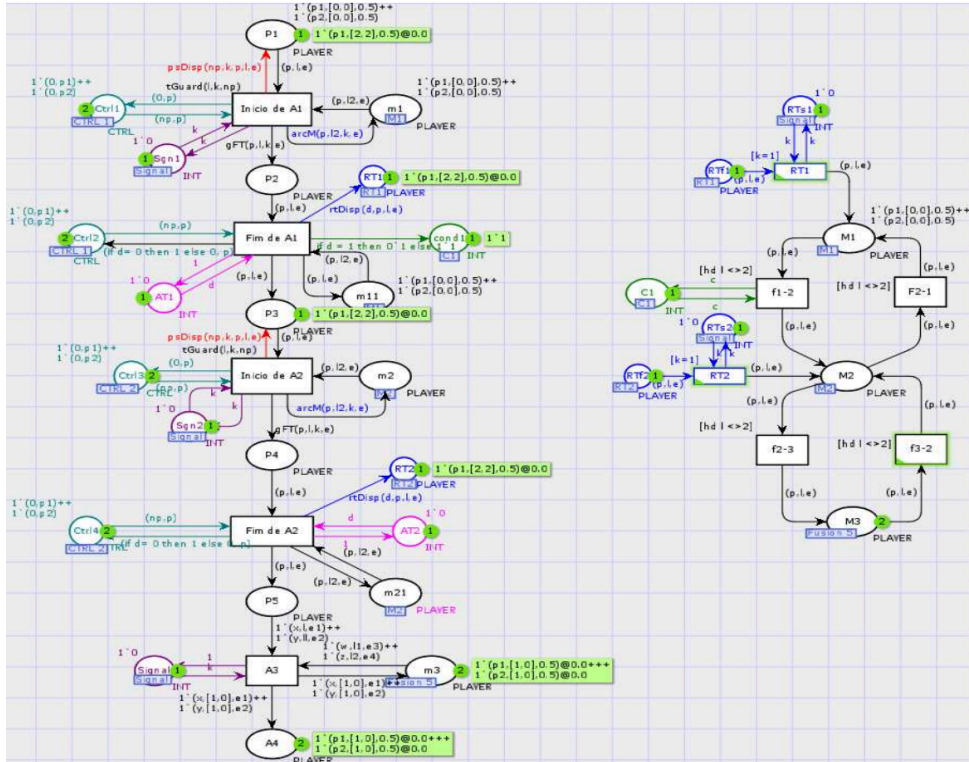


Figura 65 – Estado da rede ao final da *quest*.

5.2 Modelo de rede de Petri colorida Multiplayer

O jogo escolhido para representar um exemplo de cenário *multiplayer* tanto com o modelo de rede de Petri colorida quanto 3D, foi um nível do jogo *Silent Hill II*. Apesar de que este jogo não contém modo de jogo *multiplayer* em sua versão atual, continua sendo um exemplo válido uma vez em que ele contém atividades que vários jogadores poderiam executar simultaneamente (entrar no apartamento por exemplo) e atividades em que as ações de um jogador interferem diretamente nas ações dos seus companheiros de equipe (encontrar o mapa por exemplo que corresponde a uma atividade que todos os jogadores vão tentar realizar mas que será concluída somente por um dos jogadores; de fato, uma

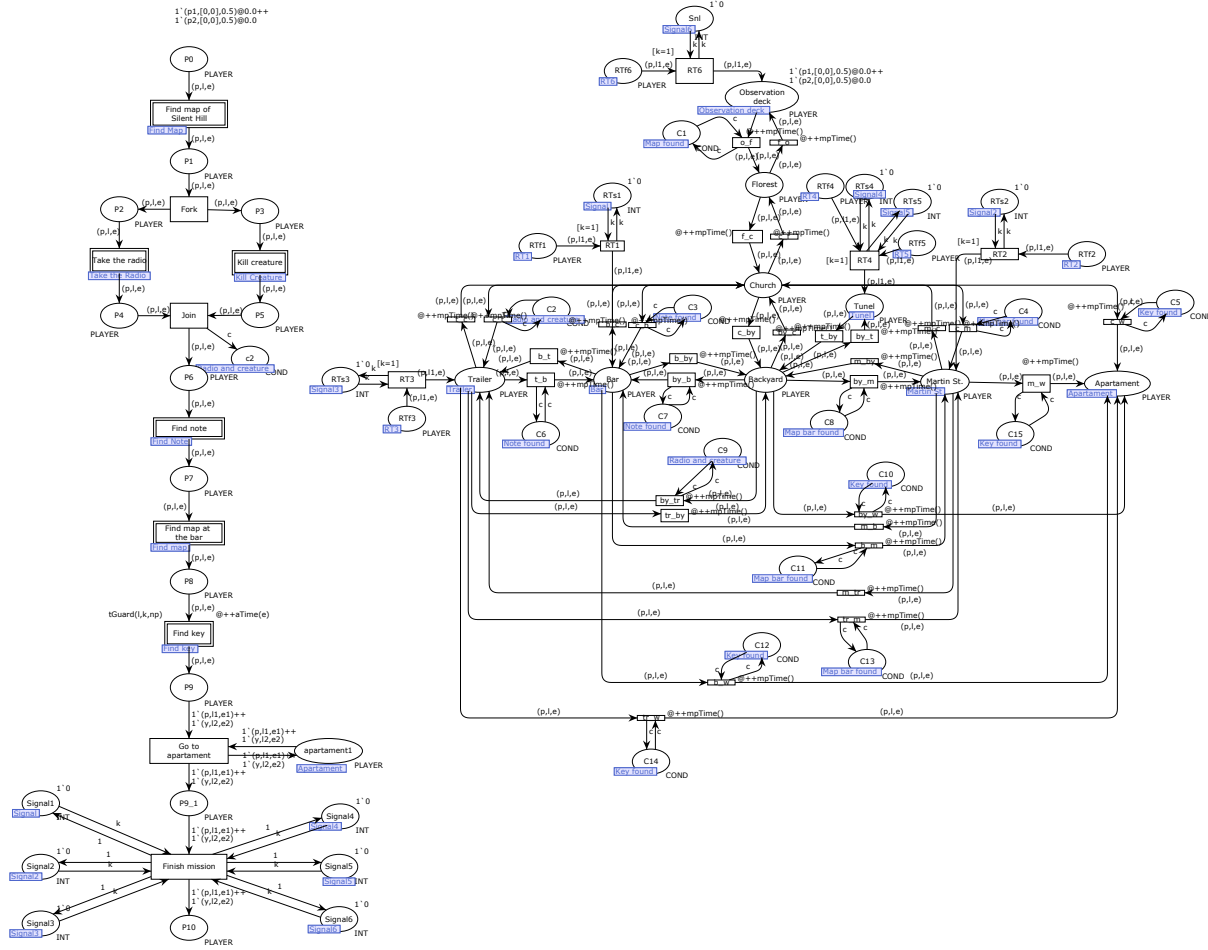


Figura 66 – Modelo de Petri Colorida global hierárquico Multiplayer do jogo *Silent Hill II*.

vez encontrado, não faria sentido os outros jogadores continuarem tentando encontrar o mapa).

A Figura 66 representa o modelo *multiplayer* global hierárquico feito com rede de Petri colorida. Nele, as atividades “*Find map of Silent Hill*”, “*Take the radio*”, “*Kill creature*”, “*Find note*”, “*Find map at the bar*” e “*Find key*” são dadas por transições hierárquicas (a transição fica associada a uma outra rede colorida) e também contém a lógica de pseudo-disparo apresentada por (BARRETO; JULIA, 2020a). Estas atividades foram representadas de forma hierárquica para deixar o modelo mais organizado já que com a lógica de pseudo-disparo é necessário acrescentar lugares e transições adicionais (em particular para representar o início e a conclusão do pseudo disparo correspondente à atividade executada pelos jogadores) que poderão tornar um modelo único ilegível.

Ao lado esquerdo da Figura 66 é ilustrado o Modelo Lógico Multiplayer e do outro lado o Modelo Topológico Multiplayer. Para o funcionamento adequado do Modelo Lógico Multiplayer é necessário a implementação de algumas funções. Além disso, para que a simulação seja iniciada com dois jogadores, a ficha inicial é configurada da seguinte forma:

2 1'(p2,[0,0],0.5)@0.0

Esta configuração inicial gera duas fichas, pois no exemplo utilizado são apenas dois jogadores. Apesar disso, caso o número de jogadores seja maior do que 2, isso pode ser feito adicionando mais inscrições de ficha na declaração. A inscrição 1'(p1,[0,0],0.5) representa 1 jogador com identificador $p1$, com a lista de atributos $[0,0]$ e com o nível de experiência inicial em 0.5. Vale ressaltar que este valor do nível de experiência do jogador é um parâmetro que pode ser definido de acordo com os critérios e regras de cada jogo.

Um aspecto importante que está presente nos jogos multiplayer é o compartilhamento das atividades. Ele pode ser cooperativo quando é necessário ter dois ou mais jogadores para executar uma atividade, ou exclusivo quando uma atividade só pode ser executada apenas uma única vez no jogo. A interação entre os jogadores faz com que atividades de compartilhamento exclusivo gerem incertezas no modelo. Para lidar com esse problema, o Modelo Lógico Multiplayer baseado nas WorkFlow nets Possibilísticas (REZENDE; JULIA; CARDOSO, 2016) foi apresentado na Seção 4.1. Uma forma de representar o modelo de simulação correspondente ao Modelo Lógico Multiplayer é criar uma representação gráfica para a região de incerteza que permite determinar o início e fim das atividades compartilhadas, executar os disparos certos e incertos (pseudo-disparos), e aplicar a confirmação do disparo certo e/ou cancelamento dos pseudo-disparos quando necessário.

Ao considerar o modelo da Figura 67, as atividades *Go to apartment* e *Finish Mission* correspondem ao caso de um compartilhamento cooperativo, pois é necessário que todos os jogadores as realizem juntos. Já todas as outras atividades correspondem a um compartilhamento exclusivo, ou seja, podem ser executadas uma única vez. Para cada uma dessas atividades de compartilhamento exclusivo é necessário definir no modelo uma estrutura que especifique seu início e fim. Por exemplo, na atividade hierárquica *Find map of Silent Hill* ilustrada pela Figura 68, a transição *Find map of Silent Hill* determina o início da atividade e a transição *Map Found* o fim. Logo, quando uma ficha se encontra no lugar P0_1, significa que a atividade está sendo executada pelo jogador.

Esta estrutura de início e fim que demarca uma atividade compartilhada no jogo permite que os jogadores iniciem a atividade juntos. Ao iniciarem uma atividade compartilhada, o disparo da transição correspondente será incerto para todos os jogadores, pois não se sabe ainda quem terminará de fato a atividade. Vale ressaltar que todas as outras atividade hierárquicas seguem o mesmo padrão, facilitando o entendimento e o desenvolvimento dos modelos em rede de Petri colorida.

Ao observar a atividade incerta ilustrada na Figura 68, é possível perceber o uso da função $tGuard(l,k,np)$ associada à transição de início da atividade. Ela faz parte do mecanismo para controlar os disparos, verificando por meio dos atributos se a ficha tem ou não permissão para disparar a transição. Por exemplo, uma ficha gerada por pseudo-disparo só poderá disparar uma determinada transição caso a confirmação ou o

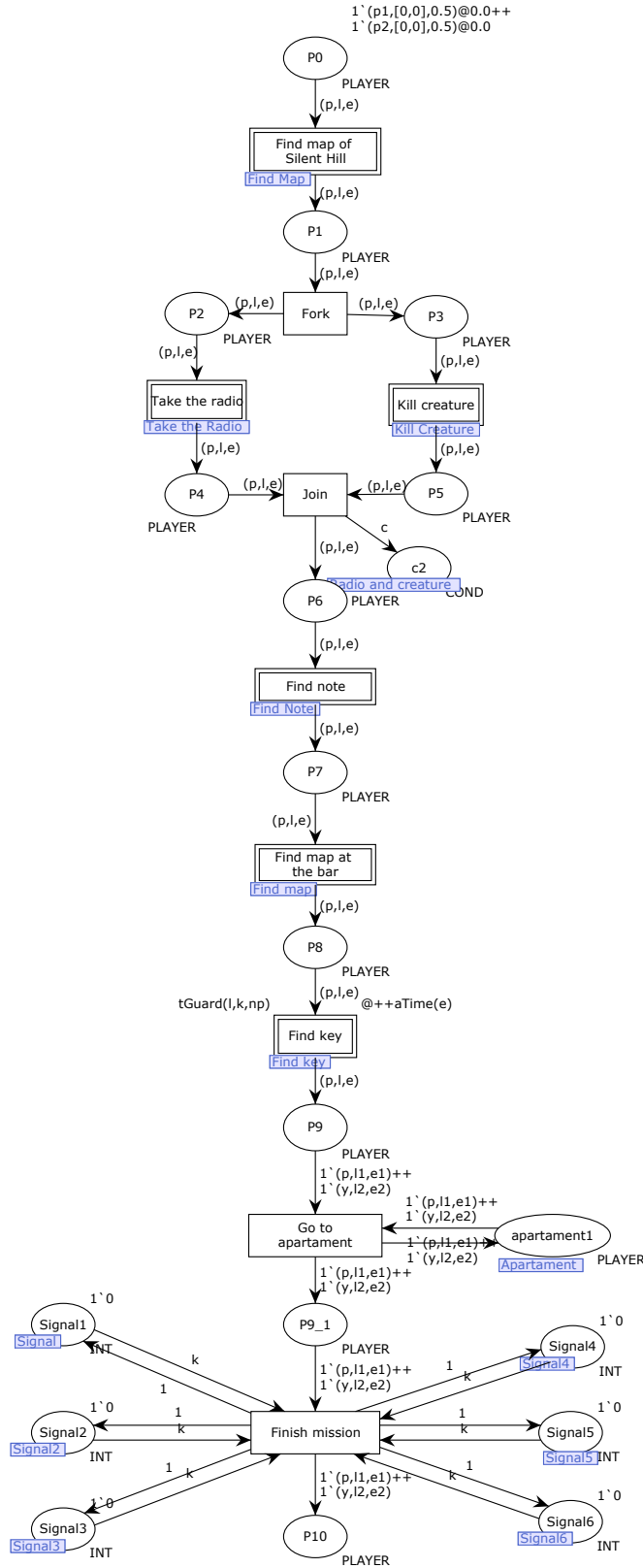


Figura 67 – Modelo Lógico Multiplayer do primeiro nível do jogo Silent Hill II implementado no CPN Tools.

cancelamento do pseudo-disparo for acionado.

As variáveis np , k e d , servem para controlar os disparos das transições no modelo,

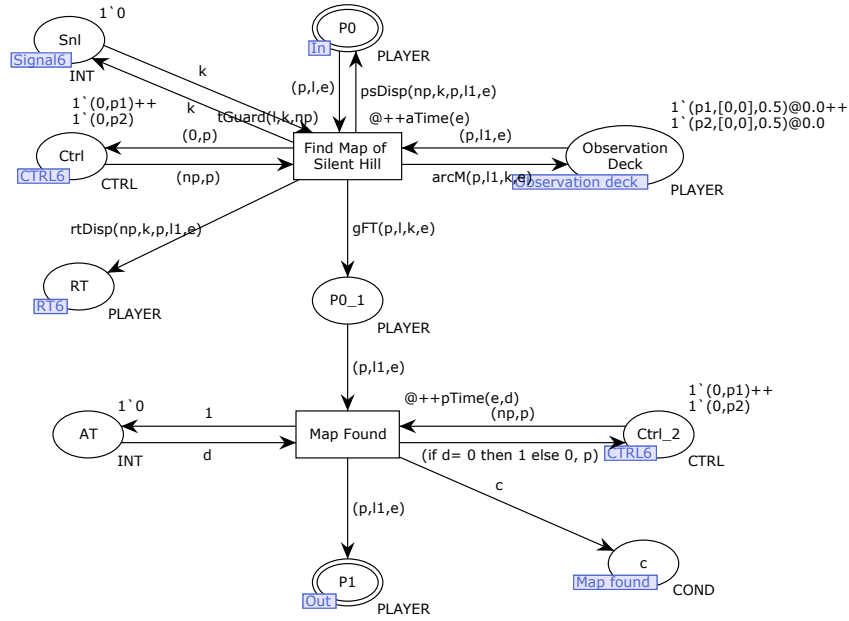


Figura 68 – Região incerta associada a um pseudo-disparo e representação da atividade “Find Map”

usando para isso a seguinte função com seus parâmetros $psDisp(np, kp, l, e)$. Essa função é associada aos arcos de saída das transições que fazem parte do comportamento possibilístico para os seus respectivos lugares precedentes. A função $psDisp(np, k, p, l, e)$ é definida da seguinte forma:

```

1 fun psDisp(np:INT, k:INT, p:P, l:L, e:REAL) =
2   if np = 0 andalso k = 0
3   then 1'(p, [2,2], e)
4   else 0'(p, l, e);

```

A função $psDisp$ recebe como parâmetro np , k , o identificador do jogador (p), a lista de atributos do jogador (l) e seu nível de experiência (e). Essa função é formada com a estrutura *IF THEN ELSE*, verificando se a atividade foi finalizada e se a atividade foi encerrada. Se a condição for satisfeita, então um pseudo-diparo acontece. O pseudo-disparo deposita uma ficha no lugar de saída da transição (executado no modelo pela função $gFT(p, l, k, e)$, apresentada a seguir), mas não retira as instâncias do lugar de entrada. Assim, a função $psDisp$ altera a lista de atributos da ficha para o valor 2, indicando que a ficha é gerada através de um *pseudo-disparo*.

```

1 fun gFT (p:P, l:L, k:INT, e:REAL) =
2   if hd l = 2 andalso tl l = [2]
3   then 0'(p, l, e)
4   else 1'(p, l, e);

```

Quando se trata de um modelo para simulação de cenários de jogos é importante considerar o tempo estimado para a execução das atividades. Cada atividade do jogo possui um tempo de execução. Na abordagem proposta nesta pesquisa para o modelo de

simulação de cenários multiplayer, o tempo de jogo será medido de acordo com o nível de experiência do jogador. Para tanto, a função exponencial será usada para gerar um tempo aleatório de execução para cada atividade. A função exponencial é uma das funções nativas do CPN Tools.

```

1 fun aTime (e:REAL) = exponential(e);
2
3 fun pTime (e:REAL, d:INT) =
4     if d = 1
5     then 0.0
6     else exponential (e) ;

```

A função $aTime(e)$ tem como parâmetro o nível de experiência do jogador e é associada as transições atividades do modelo. A função $pTime(e,d)$ também gera o tempo de execução aleatório baseado na função exponencial, porém é associada as transições da região de incerteza que representam a finalização da atividade. No caso da transição atividade ser pseudo-disparada (não corresponde então a um disparo certo e indica que a atividade correspondente será finalizado pelo outro jogador) o tempo de execução daquela atividade não é somado ao tempo de jogo do jogador.

Na transição *Finish Mission*, foram adicionados alguns lugares para receber o sinal de pseudo-disparo. A partir deste sinal, fica confirmado que o pseudo-disparo foi executado e também permite que as fichas remanescentes da atividade cancelada sejam consumidas e não permaneçam no lugar que indica a execução da atividade cancelada.

A utilização de funções, variáveis e tipos de dados específicos faz com que seja possível simular o funcionamento do Modelo Lógico Multiplayer Temporizado. O uso dos pseudo-disparos torna possível alterar os cenários do jogo (no caso multiplayer, os cenários de cada jogador são definidos somente em termos de possibilidade) sem mudar a sua estrutura. Os lugares especiais que são adicionados ao modelo trazem pouca complexidade e são eficientes para atualizar as variáveis locais e globais que são necessárias para efetuar os disparos das transições atividades sensibilizadas.

Uma das grandes vantagens do modelo apresentado por (BARRETO; JULIA, 2020a) é que toda a sua lógica de disparo é controlada por variáveis e condições do tipo *IF-THEN-ELSE*. Isso faz com que o modelo possa ser aplicado em outras ferramentas que tenham a lógica de *IF-THEN-ELSE* implementada, assim como é o caso do software SIMIO.

Um dos objetivos da criação de modelos de simulação apresentados neste trabalho é garantir que os estados de um jogo sejam sempre alcançáveis. No caso de um modelo lógico, um erro acontecerá quando uma atividade do jogo não puder, em nenhuma situação, ser executada pelo jogador. A verificação desse tipo de situação pode ser feita através da análise da propriedade de vivacidade, por exemplo. Se não há transições mortas no modelo, ou seja, a rede é livre de deadlock, é possível afirmar então que todas as atividades previstas no modelo do nível analisado poderão ser executadas eventualmente pelos jogadores.

do CPN Tools e no modelo proposto a sua implementação é dada como $mpTime()$.

```
1 fun mpTime () = uniform(a,b);
```

Ao considerar a Figura 69, é possível perceber a inscrição (marcação) na área significativa *Observation Deck* ($1'(p1, [0, 0], 0.5)@0.0 + + e 1'(p2, [0, 0], 0.5)@0.0$) que indica a área onde se encontram os dois jogadores no início do jogo. Além disso, função $mpTime()$ está associada às transições com os valores de $a = 0.1$ e $b = 1.0$. O disparo da transição representará então o tempo de movimentação do jogador para passar de uma área para a outra.

Como já explicado anteriormente, as transições de prefixo *RT* ficarão habilitadas ao final da simulação para realizar o cancelamento dos pseudo-disparos. A Figura 70 ilustra a implementação dos lugares e transição adicionais no Modelo Topológico Multiplayer para a realização do cancelamento.

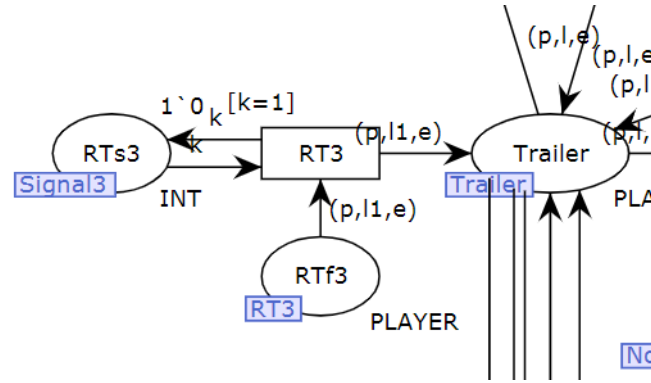


Figura 70 – Lugares e transições adicionais no Modelo Topológico Multiplayer para mecanismo do pseudo-disparo.

As demais atividades são configuradas da mesma forma, facilitando o entendimento e aplicação de novas técnicas de modelagem. Os modelos completos poderão ser encontrados nos anexos.

Ao simular e replicar o modelo da rede de Petri colorida, foi obtido os seguintes resultados representados na Figura 71.

As simulações finalizaram entre 127 e 330 passos com todas as transições do modelo disparadas em cada uma das replicações executadas. Além disso o tempo médio gasto nas simulações foi de aproximadamente 77,8 unidades de tempo de simulação.

5.3 Modelo 3D Multiplayer

Nesta seção será apresentado o Modelo de Simulação 3D baseado no Modelo de rede de Petri colorida apresentado na Seção 5.2. Também será ilustrado a partir de exemplos e imagens como aconteceu a criação do modelo SIMIO 3D para manter a equivalência de comportamento com o modelo CPN de simulação.

Simulation no.: 1 Steps.....: 206 Model time....: 65.899879449 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds	Simulation no.: 6 Steps.....: 188 Model time....: 66.5279704186 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds
Simulation no.: 2 Steps.....: 282 Model time....: 85.2871220586 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds	Simulation no.: 7 Steps.....: 127 Model time....: 58.5511880496 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds
Simulation no.: 3 Steps.....: 225 Model time....: 80.3555095992 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds	Simulation no.: 8 Steps.....: 330 Model time....: 96.6135465554 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds
Simulation no.: 4 Steps.....: 312 Model time....: 92.8214759086 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds	Simulation no.: 9 Steps.....: 266 Model time....: 76.2141657814 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds
Simulation no.: 5 Steps.....: 240 Model time....: 74.4811623946 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds	Simulation no.: 10 Steps.....: 282 Model time....: 80.9849122861 Stop reason...: No more enabled transitions! Time to run simulation: 0 seconds

Figura 71 – Replicações no modelo de rede de Petri colorida.

A Figura 72 ilustra o cenário completo do primeiro nível do jogo *Silent Hill II* criado no SIMIO. É possível perceber a representação dos jogadores no mapa, localizados na área *Floresta*. Assim como no modelo criado usando uma CPN, os jogadores podem transitar livremente entre as áreas desde que tenham concluído as restrições das áreas significativas. Além disso, a representação das atividades são feitas utilizando os processos de cada *Server* que representa um lugar específico. Desta forma, as atividades ficam diretamente associadas às áreas as quais pertencem.

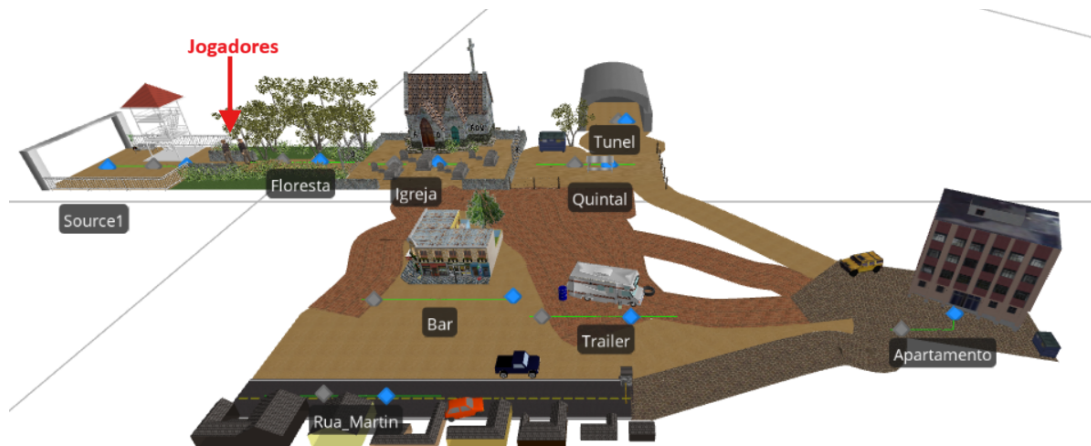


Figura 72 – Modelo 3D *multiplatform* feito utilizando SIMIO

5.3.1 Conversão do modelo CPN para o modelo 3D SIMIO

A conversão do modelo CPN para o modelo 3D feito no SIMIO é baseado nas sugestões descritas em (MOTA et al., 2017) e são apresentadas à seguir. A principal diferença ao modelo 3D SIMIO singleplayer será a representação da noção de pseudo-disparo para representar de modo realista a interação de dois ou mais jogadores num cenário de jogo que eles compartilham. O restante do modelo é similar ao que foi apresentado na Seção 4.2.

Para iniciar a simulação do Modelo Multiplayer 3D no SIMIO com mais de um jogador, é necessário alterar o atributo *Maximum Arrivals* para 2 no objeto *Source*, conforme ilustrado na Figura 73. Isso fará com que dois jogadores possam iniciar a simulação. Como dito anteriormente, o objeto *Source* irá criar as entidades na simulação. Portanto é necessário que ele seja alterado para conseguir criar mais de um jogador por simulação.

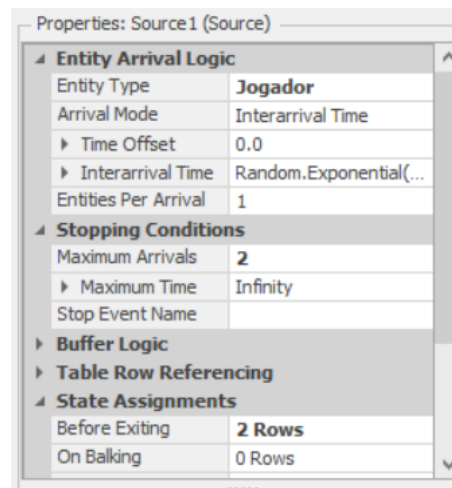


Figura 73 – Atributo *Maximum Arrivals* do objeto *Source*.

Assim como no Modelo Singleplayer, as atividades são realizadas nos locais do mapa como ilustrado na Seção 4.2 e as restrições das atividades são configuradas como apresentado na Figura 44. O que difere do Modelo Singleplayer para o Modelo Multiplayer, além de iniciar com mais de um jogador, é o mecanismo de pseudo-disparo na simulação.

Para configurar o pseudo-disparo no SIMIO é necessário entender a sequência de funções e atribuições que são feitas durante a execução da atividade para serem implementadas corretamente nos processos do SIMIO. Tendo a Figura 68 como exemplo, é possível verificar que a seguinte sequência de atribuições e funções ilustrada na Figura 74 é realizada a cada pseudo-disparo.

Esta sequência se dá da seguinte forma:

1. Verifica se a variável d é igual a 0;
2. Adiciona +1 ao valor da variável d ;
3. Executa a função *psDisp*;

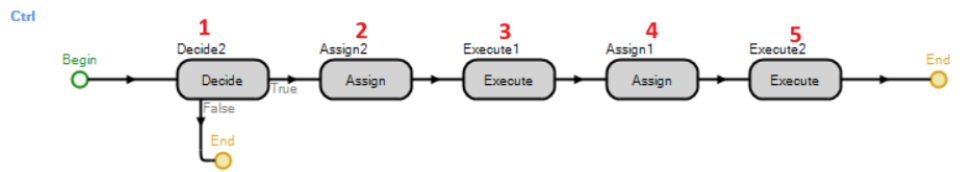


Figura 74 – Processo *Ctrl* responsável pela sequência de disparo das funções do pseudo-disparo.

4. Atribui o valor 1 à variável *np*;

5. Executa a função *rtDisp*;

O processo *Ctrl* é composto por 5 *steps* do SIMIO sendo eles: 1 *step* do tipo *Decide*, 2 *steps* do tipo *Assign* e 2 *steps* do tipo *Execute*. O *step Decide* realiza verificações de condições ou probabilidades dependendo do argumento inserido em seus campos de configuração. Os *steps Assign* alteram e/ou definem valores de variáveis na simulação. Já os *steps Execute* disparam um processo previamente definido de acordo com as definições configuradas.

Ao verificar se a variável *d* é igual a 0, é certificado de que ainda não ocorreu o pseudo-disparo para aquela atividade. Logo em seguida, é adicionado +1 ao valor da variável *d* para sinalizar que está acontecendo o pseudo-disparo. Após isso é disparada a função *psDisp*, gerando uma nova ficha ao lugar anterior à transição atual (na Figura 68 este lugar corresponde ao lugar *P0*). É atribuído o valor 1 para a variável *np*, confirmando o pseudo-disparo e, por último, é executada a função *rtDisp*, que produz a segunda ficha que registra qual dos jogadores termina a atividade primeiro. Tal sequência é representada no SIMIO pelo processo da Figura 74.

Para realizar tal verificação foi utilizado o *step Decide*. A atribuição dos valores às variáveis foi feito pelo *step Assign* e o disparo das funções foi realizado pelo *step Execute*.

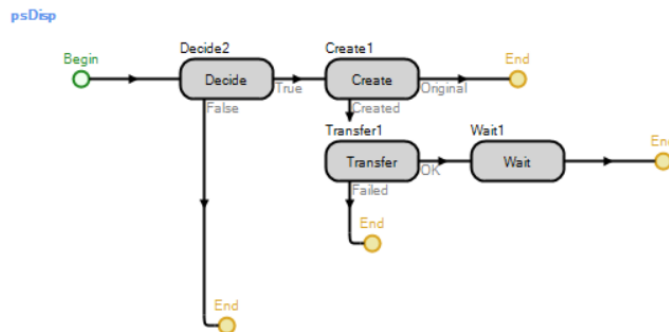


Figura 75 – Processo *psDisp*.

A função *psDisp*, ilustrada na Figura 75 reproduz o comportamento programado da mesma função do modelo CPN. Primeiro, é verificado se as variáveis *np* e *k* são iguais a 0 por meio do *step Decide*. Caso seja verdadeiro, será criada uma cópia do jogador (objeto)

atual com o *step Create*, este objeto será transferido para o local de escolha com o *step Transfer* e irá esperar até que ambos os jogadores cheguem à atividade final com o *step Wait*.

A função *rtDisp* representa pelo processo da Figura 76, similar à função *psDisp* também começa com uma verificação, porém desta vez da variável *d* utilizando o *step Decide*. Caso a variável seja igual a 0 é criado uma cópia do jogador atual (objeto) com o *step Create* que é transferida para a área desejada com o *step Transfer*. Diferente da função *psDisp*, aqui é feito uma atribuição a uma variável de controle para certificar que o pseudo-disparo já ocorreu para aquela atividade. Não há necessidade no modelo hierárquico do CPN de considerar tal variável de controle, pois cada atividade irá ser executada em uma simulação separada que fica associada ao modelo hierárquico associado à atividade considerada no CPN. Porém se for realizada uma simulação num modelo único no SIMIO (sem a decomposição hierárquica que existe no modelo CPN), é necessário a alteração do valor desta variável de controle da atividade pois as outras variáveis (*np*, *d*, *k*) serão reutilizadas nas outras atividades. Por fim, o objeto (cópia do jogador) que foi criado fica aguardando o fim das atividades por meio do *step Wait*.

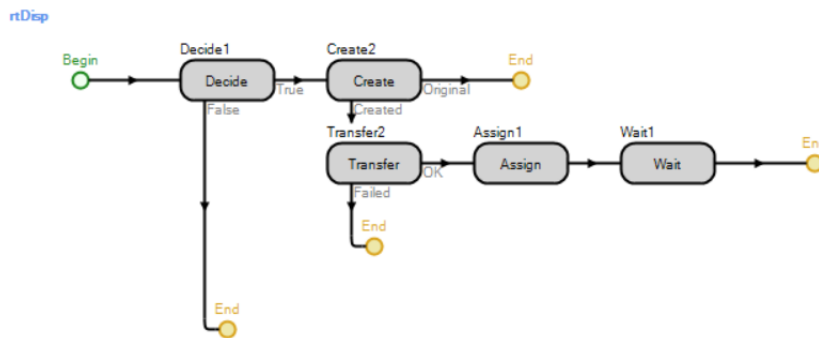


Figura 76 – Processo *rtDisp*.

Seguindo esta sequência de disparos e atribuições de valores, o mecanismo de pseudo-disparo é reproduzido de forma equivalente nos modelos criados no SIMIO. Ao final de todas as atividades de jogo consideradas, o cancelamento dos pseudo-disparos é realizado, destruindo todos os jogadores entidades criados durante a execução da simulação com o *step Destroy*. A Figura 77 ilustra a aplicação deste comportamento aplicado a área significativa *Apartamento*, que se encontra no final do modelo de simulação, assim como é liberado o mecanismo de pseudo-disparo no modelo CPN. A partir disso, o comportamento realizado no modelo de simulação hierárquico multiplayer SIMIO fica equivalente ao modelo de simulação hierárquico multiplayer CPN.

Como exemplo, a Figura 78 ilustra o pseudo-disparo da primeira atividade realizada na área significativa *Plataforma de Observação*. Para visualizar melhor os objetos gerados a partir do pseudo-disparo, foram criados *Servers* adicionais para cada atividade que contém o mecanismo de pseudo-disparo. Desta forma, as cópias dos jogadores ficarão separadas do mapa, facilitando o entendimento do processo pois os jogadores verdadeiros estarão

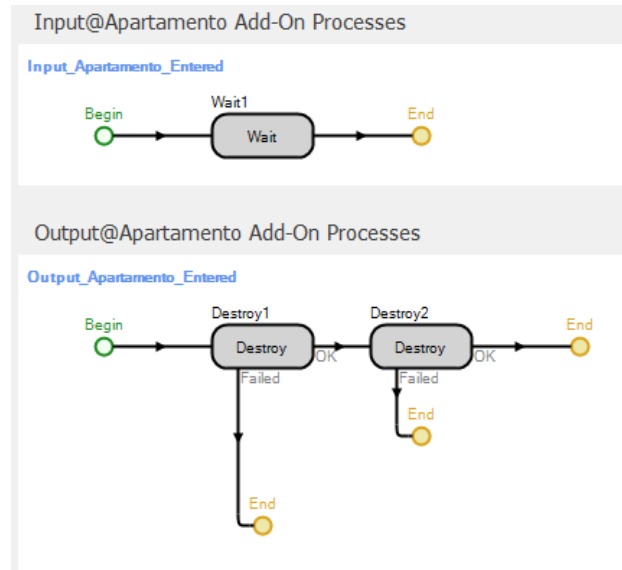


Figura 77 – Processo de cancelamento dos pseudo-disparos na área significativa *Apartamento*.

no mapa, enquanto as cópias geradas a partir dos pseudo-disparos ficarão separadas nos locais específicos de cada atividade, assim como acontece no modelo hierárquico CPN, no qual existem lugares específicos para registrar as fichas provindas do pseudo-disparo.

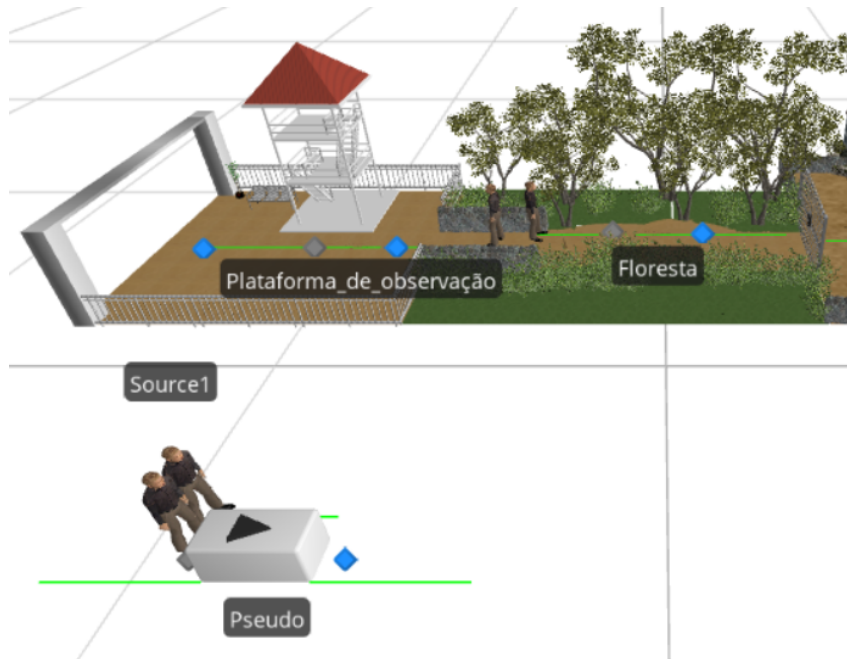


Figura 78 – Pseudo-disparo da atividade *Encontrar mapa* no SIMIO.

Para a segunda atividade que é realizada na área significativa *Tunel*, a mesma sequência de atribuições e chamadas de funções (processos ilustrados nas Figuras 74, 75 e 76) acontece resultando em algo similar ao o que aconteceu na primeira atividade *Find Map*. A Figura 79 ilustra o pseudo-disparo ocorrido nas atividades realizadas a área significativa *Tunel*.



Figura 79 – Pseudo-disparo na área significativa *Tunel* no SIMIO.

O mesmo acontece para todas as outras atividades. O mecanismo de pseudo-disparo (registro do jogador que completou primeiro a atividade) será acionado gerando duas cópias somente quando um dos jogadores efetivamente finalizar a atividade que está sendo executada. Por fim, na área *Apartamento* é realizado o cancelamento dos pseudo-disparos dos jogadores que não completaram as atividades. A Figura 80 ilustra o momento em que este processo acontece.

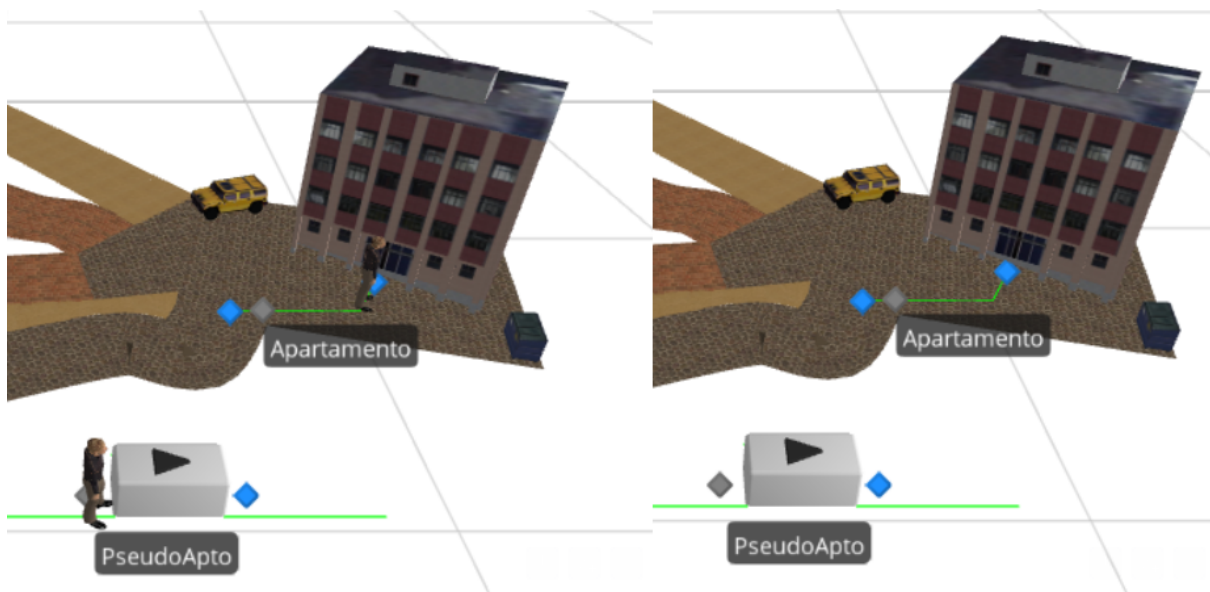


Figura 80 – Cancelamento do pseudo-disparo.

Ao lado esquerdo da Figura 80 é ilustrado o momento antes do último jogador entrar no apartamento para a simulação ser finalizada. É possível notar que há um jogador (objeto) no *Server pseudoApto*, representando a ficha do pseudo-disparo. Assim que o jogador entra na área *Apartamento* junto com o outro jogador, o processo ilustrado na Figura 77 é executado, destruindo os jogadores dos pseudo-disparos, assim como as fichas são consumidas no modelo CPN. Desta maneira a simulação é finalizada de forma equivalente.

5.3.2 Resultados das simulações do modelo 3D

Ao simular as atividades separadas por modelos no SIMIO, foi possível chegar ao seguinte resultado, representado na Figura 81:

Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Encontrar Mapa de Silent Hill	Global	I_	UserSpecified	Time0,0	Average (Sec...	35,9260
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	35,9260
Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Atividades paralelas	Global	I_	UserSpecified	Time0,0	Average (Sec...	24,6296
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	24,6296
Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Encontrar Bilhete	Global	I_	UserSpecified	Time0,0	Average (Sec...	1,5635
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	1,5635
Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Encontrar Mapa	Global	I_	UserSpecified	Time0,0	Average (Sec...	7,9600
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	7,9600
Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Encontrar Chave	Global	I_	UserSpecified	Time0,0	Average (Sec...	1,4961
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	1,4961
Object Type ▲	Object Name ▲	Data Source ▲	Category ▲	Data Item ▲	Statistic ▲ ▼	Average Total
Global Ir para o Apartamento	Global	I_	UserSpecified	Time0,0	Average (Sec...	23,5900
					Occurrences	1,0000
					Percent	100,0000
					Total (Seconds)	23,5900

Figura 81 – Resultado das simulações das atividades hierárquicas multiplayer no SIMIO.

A soma total dos tempos da simulação foi de 95,1652 unidades de tempo, ficando próximo da média alcançada pelo modelo em rede de Petri colorida. Cada resultado da Figura 81 representa a simulação da atividade descrita na linha da tabela.

Tais resultados correspondem a um tipo de validação da nossa abordagem para mostrar que o modelo SIMIO é equivalente ao modelo CPN, tanto qualitativamente quanto quantitativamente.

5.4 Considerações Finais

Neste capítulo foram apresentados os modelos de jogo *multiplayer* em rede de Petri colorida e em 3D. Com o modelo em rede de Petri colorida foi possível representar o comportamento esperado dos jogadores em um jogo, assim como as ações possíveis que aconteçam a partir de decisões individuais tomadas pelos jogadores durante a execução dos cenários de jogo simulados. Também vale ressaltar que foi possível criar uma representação das áreas do mapa, representando o modelo topológico do jogo. Além disso, a comunicação entre os modelos lógico e topológico permite indicar as atividades que devem ser executadas em cada área de jogo.

Já com o modelo 3D, foi possível representar o funcionamento do jogo como um todo, podendo ter um entendimento maior do que acontece nas atividades a partir de um modelo visual. Além disso, foi possível criar a representação dos pseudos-disparos que são utilizados no modelo em rede de Petri colorida, assim como os comportamentos esperados e inesperados das ações individuais dos jogadores durante a execução das simulações. Tudo isso fez com que o modelo se tornasse rico em detalhes, informações e confiável, uma vez que ele é baseado em uma rede de Petri colorida, além de poder ser analisado e entendido por pessoas que não tem conhecimento específico de modelos baseados em rede de Petri. Tal representação do modelo de jogo poderá então ser considerada para um tipo de avaliação do requisito de jogabilidade, já que uma representação 3D simulada poderá ser vista como uma versão de um protótipo de jogo do ponto de vista do usuário final (o jogador).

Além disso, após as simulações dos modelos em rede de Petri colorida e 3D, é possível observar que ambos atingem resultados semelhantes, principalmente em questão ao tempo gasto para realizar as atividades, confirmando a equivalência dos dois modelos.

Conclusão

Este capítulo apresenta as conclusões a respeito deste trabalho de pesquisa. A seção 6.1 mostra as principais contribuições. Na seção 6.2 são apresentados os trabalhos futuros. E, por fim, a seção 6.3 lista as contribuições bibliográficas obtidas por meio deste trabalho de pesquisa.

6.1 Principais Contribuições

Esta pesquisa apresenta métodos e possibilidades de modelagem e análise de cenários de vídeo games *singleplayer* e *multiplayer* utilizando redes de Petri coloridas e ambientes 2D/3D.

Como visto neste trabalho o desenvolvimento de jogos *singleplayer* e *multiplayer* não é uma tarefa fácil. Realizar o levantamento de requisitos e modelar os cenários de jogos auxilia no processo de *game design* e *level design* melhorando o entendimento principalmente dos usuários finais. Deve-se tomar cuidado com as tecnologias utilizadas para realizar o levantamento dos requisitos, pois apesar de retornarem dados importantes para as análises dos cenários, podem dificultar o entendimento do que está acontecendo durante a simulação.

Também foram apresentadas as vantagens e as desvantagens de utilizar as ferramentas de modelagem. As redes de Petri coloridas trazem o formalismo matemático, garantindo que o modelo simulado está correto e não haverá erros estruturais nele. Apesar disso, sua implementação e entendimento deste tipo de modelagem é dificultada principalmente para os usuários finais pois geralmente não têm o conhecimento necessário de métodos formais para compreender a complexidade dos modelos apresentados. Já ao utilizar softwares de simulação 3D como o SIMIO, os usuários finais conseguem ter um entendimento maior do que está acontecendo durante a simulação, porém é necessário tomar um cuidado maior durante a modelagem para não cometer erros estruturais, principalmente se não tiver um modelo de apoio fortemente estruturado como uma rede de Petri colorida.

Para ilustrar como os modelos de rede de Petri colorida e SIMIO podem ser equivalentes, foi utilizado o modelo de simulação *singleplayer*. Nele apenas um jogador realiza as atividades nas áreas significativas do Modelo Topológico, sendo que as atividades são feitas pelo Modelo Lógico. A comunicação entre os modelos deve ser feita com maestria para não haver perda de informação durante a simulação. A partir do modelo *singleplayer* é possível criar novas possibilidades de modelagem, já que o modelo é simplificado.

Foi também ilustrado tanto nas redes de Petri coloridas quanto no SIMIO como um pseudo-disparo pode ser implementado para simular o comportamento de um jogo *multiplayer*. Como o modelo de jogo *multiplayer* contém dois ou mais jogadores, é necessário um mecanismo de controle de execução de atividade para saber qual dos jogadores finaliza a atividade primeiro para poder liberar automaticamente a necessidade de completar a atividade proposta pelos outros jogadores. Além disso, foi apresentado os resultados de simulação tanto utilizando redes de Petri colorida quanto SIMIO e os resultados se mostraram compatíveis, validando o desenvolvimento do modelo SIMIO, assim como a sua compatibilidade com as redes de Petri coloridas.

6.2 Trabalhos Futuros

Como trabalho futuro, deve-se investigar a possibilidade de melhorar o entendimento de pessoas que não são familiarizadas com modelos formais utilizando jogos que já existem para deixar o aprendizado mais interativo. Pode-se utilizar, por exemplo, jogos como *Minecraft* para construir modelos de simulação, assim como é apresentado em (GROBELNA; MAZURKIEWICZ; JANUS, 2022). Desta forma, será possível ter um entendimento melhor sobre métodos formais e redes de Petri, estando em um ambiente mais descontraído.

Outra possibilidade é gerar *event logs* de simulação de jogos já existentes e comparar com os *event logs* reais dos jogos. Com a comparação é possível verificar se a simulação apresentada é realmente compatível com a realidade ou não. Além disso, análises mais profundas da sequência de atividades podem ser feitas utilizando mineração de processos para gerar automaticamente modelos de processos de jogos e obter informações mais aprofundadas sobre o que pode ser feito para melhorar a jogabilidade do jogo.

6.3 Produções Bibliográficas

Durante o desenvolvimento deste trabalho houve a publicação de um artigo (NEDOPETALSKI et al., 2024) no evento *ITNG 2024: 21st International Conference on Information Technology-New Generations*. O artigo em questão expõe o cenário *singleplayer* apresentado neste trabalho, mostrando a equivalência entre os modelos CPN e SIMIO.

Referências

- AALST, W. M. P. Van der. **Timed coloured Petri nets and their application to logistics**. Tese (Doutorado) — Technische Universiteit Eindhoven Eindhoven, 1992.
- AALST, W. M. Van der. The application of petri nets to workflow management. **Journal of circuits, systems, and computers**, World Scientific, v. 8, n. 01, p. 21–66, 1998.
- AALST, W. M. van der; STAHL, C. **Modeling Business Processes: A Petri Net-Oriented Approach**. The MIT Press, 2011. ISBN 9780262322546. Disponível em: <<https://doi.org/10.7551/mitpress/8811.001.0001>>.
- AALST, W. V. D.; HEE, K. M. V. **Workflow management: models, methods, and systems**. Massachusetts, London, England: MIT press, 2004.
- ABRAHÃO, S. et al. Early usability in model-driven game development. In: SPRINGER. **Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17**. [S.l.], 2016. p. 713–722.
- ANG, C. S.; RAO, G. S. V. R. K. Designing interactivity in computer games: a uml approach. **Int. J. Intell. Games Simulation**, v. 3, n. 2, p. 62–69, 2004.
- ARAÚJO, M.; ROQUE, L. Modeling games with petri nets. **Breaking New Ground: Innovation in Games, Play, Practice and Theory. DIGRA2009**. Londres, Royaume Uni, 2009.
- BARRETO, F. M.; FREITAS, J. C. J. de; JULIA, S. A timed petri net model to specify scenarios of video games. In: LATIFI, S. (Ed.). **Information Technology - New Generations**. Cham: Springer International Publishing, 2018. p. 467–473. ISBN 978-3-319-77028-4.
- BARRETO, F. M. et al. A straightforward introduction to formal methods using coloured petri nets. In: **16th international conference on enterprise information systems**. [S.l.: s.n.], 2014. (ICEIS 2014, v. 2), p. 145–152.
- BARRETO, F. M.; JULIA, S. Modeling and analysis of video games based on workflow nets and state graphs. In: **Proceedings of 24th Annual International Conference on Computer Science and Software Engineering**. Riverton, NJ, USA: IBM Corp., 2014. (CASCON '14), p. 106–119. Disponível em: <<http://dl.acm.org/citation.cfm?id=2735522.2735535>>.

- _____. A possibilistic simulation model for multiplayer game scenarios using cpn tools. In: **Proceedings of the 34th Brazilian Symposium on Software Engineering**. [S.l.: s.n.], 2020. p. 114–119.
- _____. Uma abordagem baseada em redes de petri para modelagem, análise e simulação de cenários de vídeo games singleplayer e multiplayer. Universidade Federal de Uberlândia, 2020.
- BERARD, B. et al. The expressive power of time petri nets. **Theoretical Computer Science**, Elsevier, v. 474, p. 1–20, 2013.
- BERTHOMIEU, B.; MENASCHE, M. An enumerative approach for analyzing time petri nets. In: CITESEER. **Proceedings IFIP**. [S.l.]: Elsevier Science Publishers, 1983. p. 41–46.
- BOWDEN, F. D. A brief survey and synthesis of the roles of time in petri nets. **Mathematical and Computer Modelling**, Elsevier, v. 31, n. 10-12, p. 55–68, 2000.
- CALLELE, D.; NEUFELD, E.; SCHNEIDER, K. Requirements engineering and the creative process in the video game industry. In: IEEE. **Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on**. [S.l.], 2005. p. 240–250.
- CARDOSO, J.; VALETTE, R. **Redes de petri**. Florianópolis, Santa Catarina, Brazil: Editora da UFSC Florianópolis, 1997.
- CELKO, J. Time token design methodology. **Software: Practice and Experience**, Wiley Online Library, v. 12, n. 10, p. 889–895, 1982.
- CERONE, A.; MAGGIOLO-SCHETTINI, A. Time-based expressivity of time petri nets for system specification. **Theoretical Computer Science**, Elsevier, v. 216, n. 1-2, p. 1–53, 1999.
- CHANDLER, B. A.; DAVIDRAJUH, R. Modeling and analyzing the strategy game “factorio” using modular petri nets and the general-purpose petri net simulator. **Electronics**, MDPI, v. 13, n. 7, p. 1377, 2024.
- CHUECA, J. et al. The consolidation of game software engineering: A systematic literature review of software engineering for industry-scale computer games. **Information and Software Technology**, Elsevier, p. 107330, 2023.
- COLLÉ, F.; CHAMPAGNAT, R.; PRIGENT, A. Scenario analysis based on linear logic. In: **Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology**. New York, NY, USA: ACM, 2005. ISBN 1-59593-110-4. Disponível em: <<http://doi.acm.org/10.1145/1178477.1178583>>.
- CRAWFORD, C. **The art of computer game design**. [S.l.]: Osborne/McGraw-Hill, 1982.
- DAVID, R.; ALLA, H. **Discrete, continuous, and hybrid Petri nets**. [S.l.]: Springer, 2005. v. 1.

- DOMINGO, Á. et al. Evaluating the benefits of model-driven development: Empirical evaluation paper. In: SPRINGER. **Advanced Information Systems Engineering: 32nd International Conference, CAiSE 2020, Grenoble, France, June 8–12, 2020, Proceedings 32**. [S.l.], 2020. p. 353–367.
- DOTOLI, M. et al. A timed petri nets model for intermodal freight transport terminals. **IFAC Proceedings Volumes**, v. 47, n. 2, p. 176 – 181, 2014. ISSN 1474-6670. 12th IFAC International Workshop on Discrete Event Systems (2014). Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1474667015373997>>.
- DOTOLI, M.; FANTI, M. P. An urban traffic network model via coloured timed petri nets. **Control Engineering Practice**, Elsevier, v. 14, n. 10, p. 1213–1229, 2006.
- FREITAS, J. C. J. d. Modelagem e simulação de sistemas de gerenciamento de processos de negócios baseadas em workflow net temporais com mecanismos de alocação de recursos híbridos fuzzy. Universidade Federal de Uberlândia, 2017.
- GAL, V. et al. Writing for video games. **Proceedings Laval Virtual (IVRC)**, Citeseer, 2002.
- GROBELNA, I.; MAZURKIEWICZ, M.; JANUS, D. Help students learn interpreted petri nets with minecraft. **Informatics in Education**, Vilnius University Institute of Data Science and Digital Technologies, v. 22, n. 2, p. 257–276, 2022.
- HORROCKS, I. **Constructing the User Interface with Statecharts**. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201342782.
- JENSEN, K. Coloured petri nets and the invariant-method. **Theoretical computer science**, Elsevier, v. 14, n. 3, p. 317–336, 1981.
- JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri nets: modelling and validation of concurrent systems**. Verlag Berlin Heidelberg: Springer Science & Business Media, 2009.
- KANODE CHRISTOPHER M. E HADDAD, H. M. Software engineering challenges in game development. In: LATIFI, S. (Ed.). **ITNG**. [S.l.]: IEEE Computer Society, 2009. p. 260–265. ISBN 978-0-7695-3596-8.
- KELTON, W. D.; SMITH, J. S.; STURROCK, D. T. Simio and simulation: Modeling, analysis, applications third edition. 2010.
- KESSING, J.; TUTENEL, T.; BIDARRA, R. Designing semantic game worlds. In: **Proceedings of the The third workshop on Procedural Content Generation in Games**. [S.l.: s.n.], 2012. p. 1–9.
- KRISTENSEN, L. M.; CHRISTENSEN, S.; JENSEN, K. The practitioner’s guide to coloured petri nets. **International Journal on software Tools for technology Transfer**, Springer-Verlag, v. 2, p. 98–132, 1998.
- LAHAYE, S.; KOMENDA, J.; BOIMOND, J.-L. Modeling of timed petri nets using deterministic (max,+) automata. **IFAC Proceedings Volumes**, Elsevier, v. 47, n. 2, p. 471–476, 2014.

- LEWIS, C.; WHITEHEAD, J. The whats and the whys of games and software engineering. In: **Proceedings of the 1st International Workshop on Games and Software Engineering**. New York, NY, USA: ACM, 2011. (GAS '11), p. 1–4. ISBN 978-1-4503-0578-5. Disponível em: <<http://doi.acm.org/10.1145/1984674.1984676>>.
- LUHOVA, T. et al. The canvas-oriented formalization of the game design processes. In: IEEE. **2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)**. [S.l.], 2019. p. 1254–1259.
- MARÍN-LORA, C.; CHOVER, M.; SOTOCA, J. M. A game logic specification proposal for 2d video games. In: SPRINGER. **Trends and Innovations in Information Systems and Technologies: Volume 1 8**. [S.l.], 2020. p. 494–504.
- MERLIN, P. A study of the recoverability of computer systems. **Ph. D. Thesis, Computer Science Dept., University of California**, 1974.
- MOTA, I. F. D. L. et al. **Robust Modelling and Simulation**. [S.l.]: Springer, 2017.
- MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541–580, April 1989.
- NATKIN, S.; VEGA, L.; GRÜNVOGEL, S. A new methodology for spatiotemporal game design. In: Mehdi, Q. and Gough, N.,(Eds.). **Proceedings of CGAIDE 2004, 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education**. [S.l.: s.n.], 2004. p. 109–113.
- NEDOPETALSKI, F. et al. 3d video game simulation based on colored petri net and simio. In: LATIFI, S. (Ed.). **ITNG 2024: 21st International Conference on Information Technology-New Generations**. Cham: Springer Nature Switzerland, 2024. p. 521–530. ISBN 978-3-031-56599-1.
- OLIVEIRA, G. W. de; JULIA, S.; PASSOS, L. M. S. Game modeling using workflow nets. In: SMC. IEEE, 2011. p. 838–843. ISBN 978-1-4577-0652-3. Disponível em: <<http://dblp.uni-trier.de/db/conf/smc/smc2011.html#OliveiraJP11>>.
- PEGDEN, C. D. Simio: a new simulation system based on intelligent objects. In: IEEE. **2007 Winter Simulation Conference**. [S.l.], 2007. p. 2293–2300.
- PIEVA, L.; BERNARDINO, M. Trends and research in digital game project management: A systematic literature review. **Journal on Interactive Systems**, v. 14, n. 1, p. 533–545, 2023.
- RADFORD, P. Petri Net Theory and the Modeling of Systems. **The Computer Journal**, v. 25, n. 1, p. 129–129, 02 1982. ISSN 0010-4620. Disponível em: <<https://doi.org/10.1093/comjnl/25.1.129>>.
- RAMCHANDANI, C. **Analysis of asynchronous concurrent systems by Petri nets**. Massachusetts, Cambridge, 1974.
- REUTER, C.; GÖBEL, S.; STEINMETZ, R. Approximating balance in collaborative multiplayer games. In: **Proceedings of the 9th European Conference on Games Based Learning**. [S.l.: s.n.], 2015. p. 449–455.

- REZENDE, L. P. D.; JULIA, S.; CARDOSO, J. Possibilistic workflow nets for dealing with cancellation regions in business processes. 2016.
- REZIN, R. et al. Model checking in multiplayer games development. In: IEEE. **2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)**. [S.l.], 2018. p. 826–833.
- ROLLINGS, A.; MORRIS, D. **Game Architecture and Design: A New Edition [Taschenbuch]**. [S.l.]: New Riders, 2003.
- RUCKER, R. v. B. **Software engineering and computer games**. [S.l.]: Pearson Education, 2003.
- THOMAS, P.; YESSAD, A.; LABAT, J.-M. Petri nets and ontologies: Tools for the "learning player" assessment in serious games. In: **ICALT**. [S.l.]: IEEE Computer Society, 2011. p. 415–419.
- TSAI, J. J. P.; YANG, S. J.; CHANG, Y.-H. Timing constraint petri nets and their application to schedulability analysis of real-time system specifications. **IEEE transactions on Software Engineering**, IEEE, v. 21, n. 1, p. 32–49, 1995.
- WANG, P.; FANG, W.; GUO, B. A measure of mental workload during multitasking: Using performance-based timed petri nets. **International Journal of Industrial Ergonomics**, Elsevier, v. 75, p. 102877, 2020.
- WU, N.; ZHOU, M. Real-time deadlock-free scheduling for semiconductor track systems based on colored timed petri nets. **OR Spectrum**, Springer, v. 29, n. 3, p. 421–443, 2007.
- ZUBEREK, W. M. Timed petri nets definitions, properties, and applications. **Microelectronics Reliability**, Elsevier Science, v. 31, n. 4, p. 627–644, 1991.