UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Estela Ferreira Silva

Simulação 3D de cenários de Vídeo Games modelados por redes de Petri Coloridas no software SIMIO

Uberlândia, Brasil

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Estela Ferreira Silva

Simulação 3D de cenários de Vídeo Games modelados por redes de Petri Coloridas no software SIMIO

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Stéphane Julia

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2021

Estela Ferreira Silva

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 11 de junho de 2021:

Stéphane Julia
Orientador

Franciny Medeiros Barreto

Paulo Henrique Ribeiro Gabriel

Uberlândia, Brasil
2021

Aos meus pais, Marcelo Donizete da Silva e Maria de Lourdes Ferreira Silva

Agradecimentos

Agradeço aos meus pais, Lourdes e Marcelo, pelo apoio e amor incondicionais. Aos meus amigos, por serem essa inesgotável fonte de suporte e carinho. Ao meu orientador, Professor Doutor Stéphane Julia, por toda a paciência, confiança, auxílio e bons conselhos durante o desenvolvimento deste trabalho.

Resumo

Este trabalho apresenta uma abordagem para expressar a representação formal de cenários de *video games* modelados por redes de Petri Coloridas no ambiente gráfico 3D do *software* de simulação SIMIO. A importância da construção de protótipos na fase de pré-produção de *video games* se dá pelas vantagens em identificar falhas antes que o jogo seja implementado, garantindo deste modo que ele seja atrativo e não apresente erros em termos de jogabilidade. A fim de contribuir com essa fase de importância estratégica, o objetivo deste trabalho é apresentar uma plataforma mais intuitiva de prototipação que não se baseia puramente em notações matemáticas, através do uso da interface gráfica de simulação do *software* SIMIO para auxiliar nos desafios do desenvolvimento de jogos, tornando a etapa de modelagem dos cenários de *video games* mais clara pelo meio de uma representação visual e 3D. Os cenários de jogos baseados em redes de Petri Coloridas na ferramenta CPN Tool são modelados, analisados e simulados eficientemente na ferramenta CPN Tool graças ao formalismo matemático das redes de Petri; todavia isto requer um conhecimento da teoria e das notações formais empregadas. Com a possibilidade de migração destes modelos para o SIMIO, não se faz mais necessária a compreensão das abordagens formais usadas de forma que a simulação gráfica interativa do protótipo 3D no ambiente do SIMIO possa ser acompanhada e avaliada até mesmo pelos não especialistas da área. O primeiro nível do jogo Silent Hill II é utilizado para ilustrar a abordagem. Para representar o jogo é construído um Modelo Global Temporizado. Este modelo pode ser dividido em: um Modelo Lógico Temporizado (modelo das atividades do nível), um Modelo Topológico Temporizado (modelo do mapa topológico do nível) e um mecanismo de comunicação desenvolvido para estabelecer a relação entre esses dois modelos. Os modelos são simulados tanto no CPN Tools, para a versão baseada em redes de Petri Coloridas, quanto no SIMIO para a versão 3D. Regras de transformação do modelo de redes de Petri em modelos de simulação SIMIO são definidos e um estudo comparativo entre as duas abordagens mostra as vantagens e limitações de cada modelo. Com o propósito de estimar as durações mínimas, máximas e médias que o jogador gasta para completar as atividades do nível, os modelos são também submetidos à simulação automática por replicações tanto no CPN Tools quanto no SIMIO. Os resultados dessas simulações revelam que os tempos mínimos, máximos e médios de duração da execução dos modelos obtidos em ambas as ferramentas possuem valores semelhantes e valida os mecanismos de transformação apresentados para passar do modelo formal ao modelo 3D.

Palavras-chave: redes de Petri Coloridas, *video games*, simulação, CPN Tools, SIMIO.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PN	Petri Net
CP-net	Colored Petri Net
WF-net	WorkFlow net

Sumário

1 Introdução

De modo geral, pode-se dizer que um *video game* é um tipo especial de aplicação multimídia que se trata de um produto de entretenimento que requer participação ativa do usuário (CALLELE; NEUFELD; SCHNEIDER, 2005). O principal aspecto de um jogo, que parece separá-lo do desenvolvimento de um *software* tradicional, é a exigência para que ele seja divertido (LEWIS; WHITEHEAD, 2011). Segundo Kanode e Haddad (2009), no entanto, as aplicações de jogos diferem de *softwares* tradicionais não apenas pelo uso e integração de recursos multimídias, como também pela sua fase de pré-produção. A fase de pré-produção é um tipo de coleta de requisitos e criação de protótipos do jogo.

Do ponto de vista clássico da indústria de *video games*, o desenvolvimento de jogos é composto por duas etapas principais denominadas de *game design* e *level design*. Na etapa de *game design* são definidos os aspectos principais do universo do jogo e cada detalhe de como ele irá funcionar. É na etapa de *level design* que todos os diferentes componentes do jogo se unem. Nessa fase, define-se a descrição de como o jogador irá interagir com os objetos (itens) do jogo, a descrição das missões que o jogador irá executar, bem como o nível de dificuldade para a execução dessas atividades (GAL et al., 2002).

Um jogo possui um ou mais níveis (*levels*) e uma narrativa que move o jogador. Um nível de jogo consiste de uma topologia (espaço virtual), enigmas, ações principais e de um conjunto de itens/objetos que irão interagir para o objetivo ser alcançado. Cada nível tem que ser significativo e tem de manter a coerência com o tema e a meta central do jogo. Dessa forma, é preciso levar em consideração a lógica de sequenciamento das ações relativas aos objetos, o posicionamento desses objetos no universo do jogo, bem como as limitações do espaço virtual. Já uma narrativa (ou cenário) é uma sequência ordenada de eventos, onde um evento é um elemento atômico da história ou alguma coisa significativa que acontece (COLLÉ; CHAMPAGNAT; PRIGENT, 2005).

Em termos de *game design*, um cenário corresponde à descrição clássica de uma sequência de ações das principais fases do jogo e de como ocorre a navegação entre essas fases. Já no *level design*, os cenários correspondem ao posicionamento dos objetos em relação às missões do jogo. Isso induz uma sequência de ações parcialmente ordenadas que o jogador deve executar para chegar ao final de um determinado nível do jogo (GAL et al., 2002).

A ludologia (o estudo dos jogos em geral, particularmente dos *video games*) aponta a necessidade de criar modelos para especificar os mecanismos dos jogos, pois a falta deste conhecimento tem sido um dos grandes problemas do tradicional projeto de jogos (REYNO; CUBEL, 2009). Em particular, as maiores falhas de desenvolvimento de

jogos se encontram geralmente na etapa de levantamento de requisitos (LEWIS; WHITEHEAD, 2011). Portanto, é necessário encontrar novos conceitos e ferramentas para suprir os desafios do desenvolvimento de jogos (HORROCKS, 1999).

No contexto de *video games*, alguns trabalhos têm apresentado o uso de métodos formais para especificar o conjunto de requisitos de um jogo. Os modelos formais são rigorosamente definidos e não contém especificações ambíguas (LEWIS; WHITEHEAD, 2011). Dentro deste contexto, alguns trabalhos já consideram as redes de Petri como uma ferramenta eficiente para modelagem e análise de sistemas de jogos. 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).

Como apresentado em (MOTA et al., 2017), a forma de integração das redes de Petri com ambientes de modelagem e simulação tem sido recentemente estudada. Por um lado, ambientes de modelagem e simulação não têm sido bons o suficiente para auxiliar o analista a entender as relações de causa e efeito em sistemas que envolvem mecanismos de sincronização complexos. Por outro, o formalismo das redes de Petri é muito abstrato para ser apresentado aos tomadores de decisão que não necessariamente têm conhecimento das notações formais usadas nas linguagens de programação.

Com a presente pesquisa, é apresentada uma representação visual de cenários de *video games* que aborda os aspectos presentes na maioria dos jogos atuais. Para tanto, é considerada a teoria das redes de Petri, bem como ferramentas de modelagem e simulação que não requerem conhecimento de linguagens de programação particular para implementar as funcionalidades dos modelo simulados.

1.1 Objetivos

Inspirando-se nos modelos de cenários de *video games* utilizados em (BARRETO; JULIA, 2017; BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020), a presente pesquisa consiste em uma abordagem baseada em redes de Petri Coloridas para modelar e simular cenários de jogos em um ambiente gráfico visual 3D, validando-os através de uma análise qualitativa e quantitativa aplicada ao conceito de jogabilidade em *video games*. O objetivo deste trabalho é, portanto, auxiliar projetistas no processo de criação de *video games* tornando a fase de pré-produção de jogos (*game design* e *level design*) mais clara, em particular através da modelagem dos cenários de jogos.

O desempenho e corretude dos modelos de jogos em redes de Petri Coloridas são analisados utilizando-se a técnica de simulação automática da ferramenta CPN Tools. Finalmente, baseando-se na abordagem apresentada em (MOTA et al., 2017), realizou-se a migração dos modelos de jogos em redes de Petri Coloridas para o ambiente de

simulação 3D SIMIO, para a produção de protótipos de cenários de jogos, que poderão ser visualizados e avaliados até mesmo pelos não especialistas das abordagens formais usadas em Engenharia de Software.

1.1.1 Objetivos Específicos

Especificamente, objetiva-se com este trabalho:

1. selecionar um estudo de caso baseado nos modelos de cenários de *video games* apresentados em (BARRETO; JULIA, 2017; BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020), os quais empregam a ferramenta CPN Tools para analisar a performance e corretude do modelo de jogo;

2. realizar a migração do modelo de jogo baseado em redes de Petri Coloridas para o ambiente de simulação 3D do *software* SIMIO;

3. baseado no ambiente de simulação 3D do SIMIO, produzir um protótipo do modelo de jogo para a validação das etapas de *game design* e *level design*.

1.2 Justificativa

A fase de pré-produção de jogos possui uma importância estratégica, pois é nela onde ocorre a definição dos aspectos que vão caracterizar o jogo. É nessa etapa que temos uma visão geral de como o jogo se desenvolverá, onde ocorre a prevenção de erros de jogabilidade e se garante que a experiência do jogo se desenrolará em uma sucessão de metas dentro de um prazo razoável. E como já afirmou Kanode e Haddad (2009), a Engenharia de Software se torna indispensável para ajudar a suprir os desafios de desenvolvimento de jogos, melhorando a gerência dos projetos, diminuindo os riscos e garantindo o sucesso futuro da indústria de jogos.

Porém, esse ponto crítico no projeto do jogo envolve a modelagem dos possíveis cenários que o compõe para os analisar e validar. Esse procedimento requer uma escolha de metodologias consistentes e ferramentas que a suportem, algo que pode tornar o processo de verificação desafiante para aqueles que não estão familiarizados com técnicas de modelagem e métodos formais utilizados em Engenharia de Software.

O formalismo das redes de Petri Coloridas tem sido amplamente utilizado pela comunidade científica para realizar não apenas análise do comportamento de modelos, mas também como ferramenta de simulação para realizar tipos de análises quantitativas de sistemas. Em particular, suas características permitem uma melhor compreensão das relações causais presentes em Sistemas a Eventos Discretos (JENSEN; KRISTENSEN, 2009).

Por outro lado, a fim de reduzir os esforços necessários em projetos de simulação, os projetistas de sistemas propuseram novos paradigmas de desenvolvimento baseados em interfaces gráficas e em ferramentas automáticas de análise. Infelizmente, ao lidar com grandes projetos, os paradigmas de simulação baseados puramente em notações matemáticas dificultam o entendimento das relações causais dos sistemas (MOTA et al., 2017).

Por esse motivo, o presente trabalho propõe integrar o formalismo de modelagem das redes de Petri Coloridas com o *software* de simulação 3D SIMIO. Com isso, será estabelecida uma plataforma mais intuitiva de prototipação para os projetistas de *video games* nas fases iniciais de levantamento de requisitos de jogos (*game design* e *level design*), deixando a etapa de modelagem dos cenários de *video games* mais clara através de uma representação visual. O que torna isso possível é a ferramenta permitir que o projetista acompanhe a simulação em um ambiente gráfico visual 3D prático e de fácil entendimento, possibilitando que projetistas mais inexperientes ou pessoas com pouca familiarização com modelagem e simulação tenham uma melhor compreensão da evolução de cenários de jogos, sem exigir um conhecimento prévio do formalismo das redes de Petri (MOTA et al., 2017).

2 Referencial Teórico

Este capítulo apresenta os conceitos necessários para o entendimento desta pesquisa. Sendo assim, as seções 2.1, 2.2, 2.3, 2.4, 2.5 e 2.6 trazem os conceitos sobre as redes de Petri e suas propriedades, WorkFlow nets, grafos de estado, redes de Petri Coloridas, a ferramenta CPN Tools e o *software* SIMIO, respectivamente.

2.1 Redes de Petri

A teoria das redes de Petri foi proposta por Carl Adam Petri na tese (PETRI, 1962). 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).

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

De acordo com David e Alla (2010), os elementos básicos que permitem a definição das redes de Petri são:

- **Lugar**: um lugar é representado graficamente por um círculo. Esse elemento pode ser interpretado como uma condição, um estado de espera, um procedimento, a existência de um recurso, etc.;
- **Transição**: uma transição é representada graficamente por uma barra ou retângulo. Esse elemento é associado a um evento que ocorre no sistema;
- **Ficha** (*token*): a ficha é representada graficamente por um ponto preto 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.

Formalmente, as redes de Petri são definidas em (MURATA, 1989; CARDOSO; VALETTE, 1997) da seguinte forma:

Definição 1 (Rede de Petri). *Uma rede de Petri é uma quádrupla PN = (P, T, Pré, Pós), onde:*

- ***P**: é um conjunto finito de lugares;*
- ***T**: é um conjunto finito de transições;*
- ***Pré**: é uma relação que define os arcos que ligam os lugares às transições;*
- ***Pós**: é uma relação que define os arcos que ligam as transições aos lugares;*

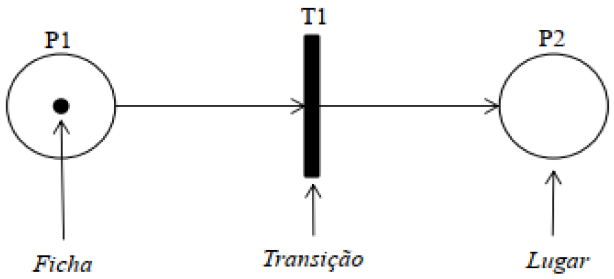


Figura 1 – Elementos básicos de uma rede de Petri. Figura retirada de (BARRETO, 2015).

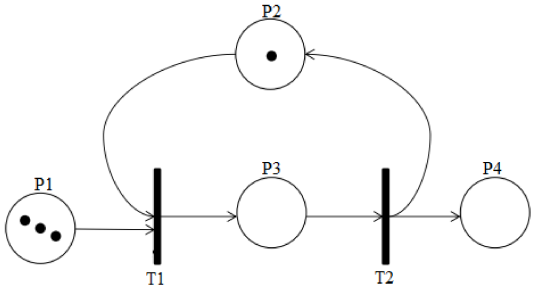
A Figura 1 apresenta um exemplo de uma rede de Petri e seus elementos básicos. Nela, o lugar *P1* é chamado lugar de entrada da transição *T1* pois existe um arco direcionado saindo de *P1* e chegando a *T1*. Já o lugar *P2* é lugar de saída da transição *T1* pois existe um arco direcionado de *T1* para *P2*.

Ainda na Figura 1, o lugar *P1* contém uma ficha. Em um dado momento em uma rede de Petri, um lugar p pode conter zero ou mais fichas. A marcação de uma rede de Petri, denotada por M, diz respeito à distribuição de fichas nos lugares. Durante a execução da rede, o número de fichas pode mudar de acordo com a ocorrência de certos eventos.

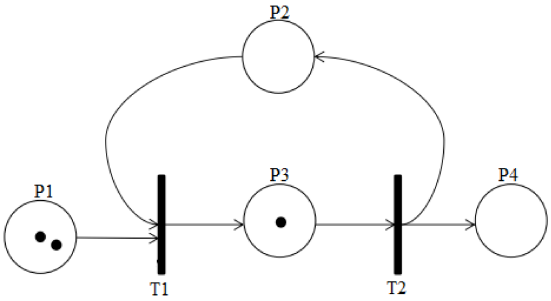
Em uma rede de Petri, a ocorrência de um evento no sistema é representada pelo chamado disparo da transição ao qual este evento está associado. As transições são os componentes ativos que mudam a marcação da rede de acordo com os disparos. De acordo com Murata (1989), o estado ou marcação em uma rede de Petri é alterado de acordo com as seguintes regras de disparo:

1. uma transição t é dita habilitada se cada lugar de entrada p de t contém pelo menos $w(p,t)$ fichas, onde $w(p,t)$ é o peso do arco de p para t;
2. uma transição habilitada pode ou não ser disparada, dependendo se o evento correspondente ocorre ou não;
3. um disparo de uma transição habilitada t remove $w(p,t)$ fichas de cada local de entrada p de t e adiciona $w(t,p)$ fichas a cada lugar de saída p de t, onde $w(t,p)$ é o peso do arco de t para p.

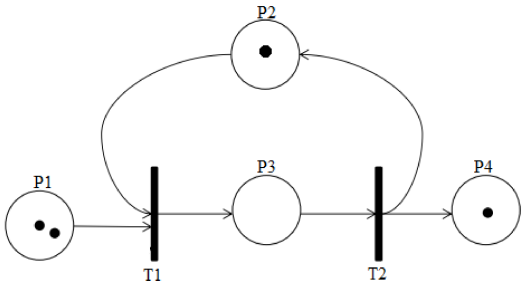
Retirar as fichas de um lugar indica que esta condição não é mais verdadeira após a ocorrência do evento. Por outro lado, depositar fichas em um lugar indica que a atividade associada ao lugar está sendo executada após a ocorrência do evento. Deste modo, no momento em que um evento ocorre, a rede passa de um estado discreto para outro, alterando sua marcação. Assim, o comportamento dinâmico do sistema é traduzido pelo comportamento da rede de Petri (DAVID; ALLA, 2010).



(a) Antes dos disparos.



(b) Após o primeiro disparo.



(c) Após o segundo disparo.

Figura 2 – Exemplo de disparos em uma rede de Petri. Figura adaptada de (BARRETO, 2015).

Um exemplo do funcionamento de uma rede de Petri pode ser visto na Figura 2. Na Figura 2a pode-se observar a marcação inicial dessa rede, onde a transição *T1* está habilitada já que todos os seus lugares de entrada, *P1* e *P2*, contém pelo menos uma ficha. Dessa forma, após o disparo de *T1* a Figura 2b é gerada, ou seja, é consumida uma ficha de cada lugar de entrada e gerada uma ficha no lugar de saída *P3*. Com isso a transição *T2* é habilitada e podemos ver na Figura 2c o resultado de seu disparo: é consumida a ficha de seu lugar de entrada e gerada uma ficha em cada lugar de saída.

A dinâmica de um sistema descrito por uma rede de Petri é dada pela evolução das marcações. De acordo com o conjunto de marcações acessíveis a partir da marcação inicial, são definidas algumas propriedades comportamentais (reagrupadas sob o nome genérico de boas propriedades (CARDOSO; VALETTE, 1997)). Tais propriedades são definidas por Murata (1989) e apresentadas a seguir:

- **Alcançabilidade**: essa propriedade indica a possibilidade de alcançar uma determinada marcação. Uma marcação é dita alcançável se existe uma sequência finita

de disparo de transições que conduza a ela a partir da marcação inicial. Essa propriedade garante que certos estados serão alcançados ou não;

- **Limitabilidade**: uma rede de Petri é dita limitada se o número de fichas em cada lugar da rede não exceder um inteiro positivo k. Neste caso, a rede é dita k-limitada. Se a rede for limitada ao inteiro 1, então diz-se que ela é binária;
- **Vivacidade**: uma rede de Petri é dita viva se todas as suas transições são vivas. Uma transição t é dita viva se para cada marcação alcançável da rede, existe uma sequência de disparo s que sensibiliza, ou habilita, t. Dessa forma, uma rede de Petri viva é uma rede onde todas as suas transições são disparadas. Essa propriedade garante que o sistema é livre de *deadlock*;
- **Reiniciabilidade**: uma rede de Petri é dita reiniciável se é sempre possível voltar para a marcação inicial através de uma sequência de disparos, seja qual for a marcação considerada.

2.2 WorkFlow nets

De acordo com Aalst (1998), uma rede de Petri que modela um processo de negócio é chamada de WorkFlow net (WF-net) e satisfaz as seguintes propriedades:

1. uma WF-net tem 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;
2. uma ficha no lugar i representa um caso que precisa ser tratado. Uma ficha no lugar o representa um caso que já foi tratado;
3. em uma WF-net, 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)

Um processo de negócio especifica quais tarefas precisam ser executadas e em qual ordem executá-las. Além disso, de acordo com Aalst (1998), processos de *workflow* são baseados em casos, isto é, cada parte do trabalho é executada para um caso específico. Modelar um processo de negócio em termos de uma WF-net é bem direto: as tarefas são modeladas por transições, condições são modeladas por lugares e os casos são modelados pelas fichas (AALST, 1998).

A ordem em que as tarefas são executadas varia de caso para caso. Por meio do roteiro de um caso ao longo de uma série de tarefas é possível determinar qual a ordem das tarefas que precisam ser executadas (AALST; HEE, 2004). Segundo Aalst e Hee (2004), quatro construções básicas são consideradas para o roteamento de tarefas: (1) a

sequencial, quando as tarefas precisam ser executadas uma após a outra; (2) a paralela, quando mais de uma tarefa pode ser executada simultaneamente ou em qualquer ordem; (3) a condicional, quando pode existir uma escolha entre duas ou mais tarefas; (4) a iterativa, quando é necessário repetir uma mesma tarefa ou uma sequência de tarefas várias vezes.

2.2.1 Propriedade Soundness

Soundness é um critério de corretude definido para as WorkFlow nets. Uma WorkFlow net é dita *sound* se, e somente se, os seguintes requisitos forem satisfeitos (AALST; HEE, 2004; AALST, 1998):

1. para cada ficha colocada no lugar de início i, uma e apenas uma ficha deve aparecer no lugar de término o;
2. quando uma ficha aparece no lugar o, todos os outros lugares estão vazios para o caso em questão;
3. para cada transição (tarefa), é possível evoluir da marcação inicial para uma marcação onde essa transição é sensibilizada, ou seja, em uma WorkFlow net não deve haver transição morta.

A propriedade *soundness* está relacionada com a dinâmica das WorkFlow nets. O primeiro requisito significa que todo caso será completado após um período de tempo. O segundo requisito significa que uma vez que um caso é completado, nenhuma referência a ele permanecerá no processo. Já o terceiro requisito afirma que todas as tarefas em uma WF-net podem ser, a princípio, executadas.

A propriedade *soundness* é um critério importante a ser satisfeito quando são considerados processos de negócios. No contexto das WF-nets, a prova desse critério está relacionada com um problema de análise qualitativa de um modelo. Em Aalst (1998), Aalst e Hofstede (2000) um método foi proposto para a verificação da propriedade *soundness* de uma WorkFlow net. Este método traduz a propriedade *soundness* através de duas propriedades bem conhecidas: vivacidade e limitabilidade (AALST; HEE, 2004).

Dado uma WorkFlow net PN = (P,T,F) deve-se decidir se PN é *sound*. Para tanto, define-se uma rede estendida $\overline{\text{PN}} = (\overline{\text{P}},\overline{\text{T}},\overline{\text{F}})$. $\overline{\text{PN}}$ é uma rede de Petri obtida adicionando-se uma transição extra t^*. A transição t^* conecta o lugar de término (o) ao lugar de início (i). A Figura 3 ilustra a relação entre PN e $\overline{\text{PN}}$.

Para uma WorkFlow-net PN arbitrária e sua correspondente rede de Petri estendida $\overline{\text{PN}}$, o seguinte teorema (AALST, 1997) pode ser provado:

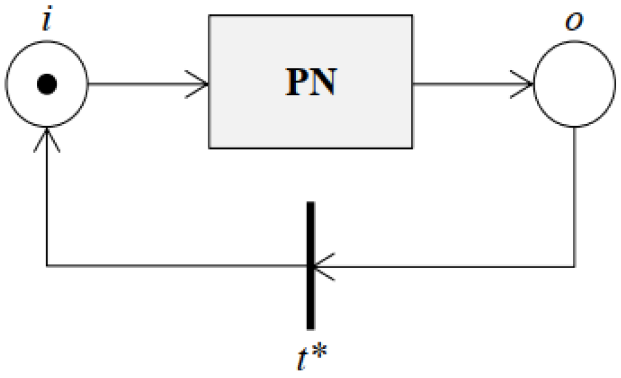


Figura 3 – Exemplo de uma rede de Petri estendida. Figura retirada de (BARRETO, 2015).

Teorema 1 (Soundness). *Uma WF-net é dita sound se, e somente se, $(\overline{PN},i)$ é viva (live) e limitada (boundedness).*

A prova deste teorema pode ser encontrada em Aalst (1996), Aalst (1997). Dessa forma, a verificação da propriedade *soundness*, em uma WF-net, resume-se em verificar se a rede de Petri estendida $\overline{\text{PN}}$ é viva e limitada. Isso significa que ferramentas de análise baseadas em redes de Petri podem ser utilizadas para definir o critério *soundness* da rede (AALST, 1996; AALST, 1997; AALST, 1998).

2.3 Grafos de Estado

Uma rede de Petri não marcada é um grafo de estado se, e somente se, cada transição tiver exatamente um arco de entrada e um arco de saída (DAVID; ALLA, 1994).

Uma rede de Petri marcada conhecida como grafo de estado, será equivalente a um grafo de estado no sentido clássico (ou diagrama de estados, representando um autômato que está em apenas um estado por vez) se, e somente se, contém exatamente uma ficha localizada em um dos lugares da rede. Em um grafo de estado, o peso de todos os arcos é igual a 1 (DAVID; ALLA, 2010).

2.4 Redes de Petri Coloridas

Redes de Petri são tradicionalmente divididas em redes de Petri de baixo nível, as quais possuem fichas simples (números naturais associados aos lugares) que geralmente indicam uma condição ou a existência de um recurso e redes de Petri de alto nível que permitem a construção de modelos compactos e parametrizados.

Como afirma Aalst e Stahl (2011), para a modelagem de sistemas complexos ou com um comportamento temporal, o modelo clássico das redes de Petri é insuficiente. Nesse sentido, muitas extensões do modelo básico da rede de Petri surgiram da necessidade de representar com precisão os sistemas complexos. Uma dessas extensões foi a rede de Petri Colorida (*Colored Petri Net* — CP-net).

A ideia por trás das redes de Petri Coloridas é reunir a capacidade de representação da sincronização e do paralelismo das redes de Petri com o poder expressivo das linguagens de programação com seus tipos de dados e o conceito de tempo. As CP-nets pertencem à classe das redes de Petri de alto nível e são caracterizadas pela combinação das redes de Petri e da linguagem de programação funcional CPN ML (MILNER et al., 1997).

Dessa forma, o formalismo das redes de Petri é adequado para descrever ações concorrentes e síncronas, enquanto que a linguagem de programação funcional pode ser usada para definir tipos de dados específicos e manipulá-los (JENSEN; KRISTENSEN, 2009).

Formalmente, as redes de Petri Coloridas são definidas em Jensen e Kristensen (2009) da seguinte forma:

Definição 2 (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:*

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

Os estados de uma CP-net são representados pela marcação dos lugares. Cada lugar tem um tipo associado que determina o tipo de dado que o lugar pode conter, ao mesmo tempo que os arcos de saída terão uma variável do mesmo tipo associada. Cada lugar conterá um número variado de fichas e durante o disparo de uma transição no modelo as variáveis dos arcos de entrada serão substituídas pelo valor da ficha. O número exato de fichas e seus valores de dados são indicados pela expressão do arco, localizada ao lado do arco.

Por sua vez, cada ficha tem um valor, denominado de cor no contexto das CP-nets, que pertence ao tipo de dado associado com o lugar (AALST; STAHL, 2011). Essas cores não representam apenas cores ou padrões, elas podem representar desde dados primitivos (número inteiros, por exemplo) até tipos de dados complexos (produto cartesiano de valores, por exemplo) (JENSEN; KRISTENSEN, 2009).

Em uma CP-net, as fichas geralmente representam objetos ou recursos em um sistema modelado (AALST, 1992). Esses objetos podem ter atributos, os quais não são facilmente representados por uma ficha simples de uma rede de Petri que não identifica as fichas de um mesmo lugar.

As redes de Petri Coloridas também permitem adicionar informações de tempo nos modelos a fim de investigar propriedades relativas a performance do sistema. Para isso, foi introduzido um relógio global que representa o tempo do modelo. Os valores do relógio podem ser discretos ou contínuos (KRISTENSEN; CHRISTENSEN; JENSEN, 1998).

Em um modelo CP-net temporizado, um valor de tempo é associado a ficha, conhecido como *timestamp* (JENSEN; KRISTENSEN, 2009). Para calcular o *timestamp* associado a uma ficha, é necessário usar uma inscrição de atraso associada à transição ou aos arcos de saída das transições. Quando uma inscrição/função de tempo é associada a uma transição, um tempo é adicionado em todas as fichas produzidas por essa transição. Por outro lado, quando uma inscrição/função de tempo é associada aos arcos de saída de uma transição, um tempo é adicionado apenas nas fichas criadas nesse arco (JENSEN; KRISTENSEN, 2009).

A hierarquia das redes de Petri Coloridas oferece um conceito conhecido como *fusion places* (lugares de fusão). Esse conceito permite especificar um conjunto de lugares que são considerados idênticos (KRISTENSEN; CHRISTENSEN; JENSEN, 1998). Isto significa que todos esses lugares representam um único lugar conceitual, ainda que sejam desenhados como vários lugares individuais. Esses lugares são chamados individualmente de *fusion places* e um conjunto de *fusion places* é chamado de *fusion set* (conjunto de fusão). Qualquer ação que acontecer a um lugar do conjunto de fusão, também acontece aos outros lugares do mesmo conjunto. Assim, se uma ficha for adicionada/consumida de um dos lugares, uma ficha idêntica também será adicionada/consumida em todos os outros lugares do conjunto de fusão. De acordo com Aalst e Stahl (2011), essa característica

facilita a modelagem de sistemas grandes e complexos, tais como sistemas de informação e de processos de negócio. A Figura 4 ilustra um exemplo de uma rede de Petri Colorida Temporizada com *fusion places*.

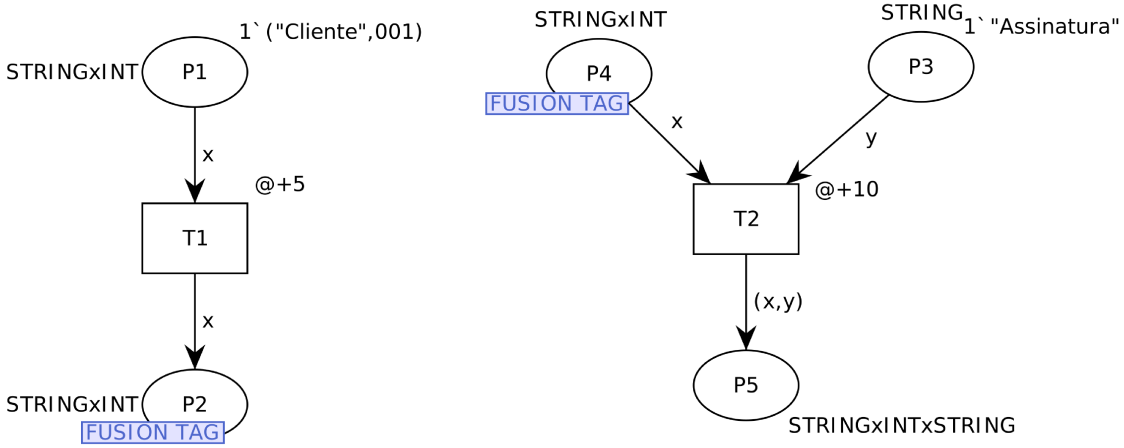


Figura 4 – Exemplo de uma CP-net Temporizada com *fusion places*.

2.5 CPN Tools

O CPN Tools é uma ferramenta adequada para edição, simulação e para análise do estado de espaço (*state space*) e análise do desempenho de modelos de redes de Petri Coloridas (KRISTENSEN; CHRISTENSEN; JENSEN, 1998). A Figura 5 apresenta a tela da interface da ferramenta.

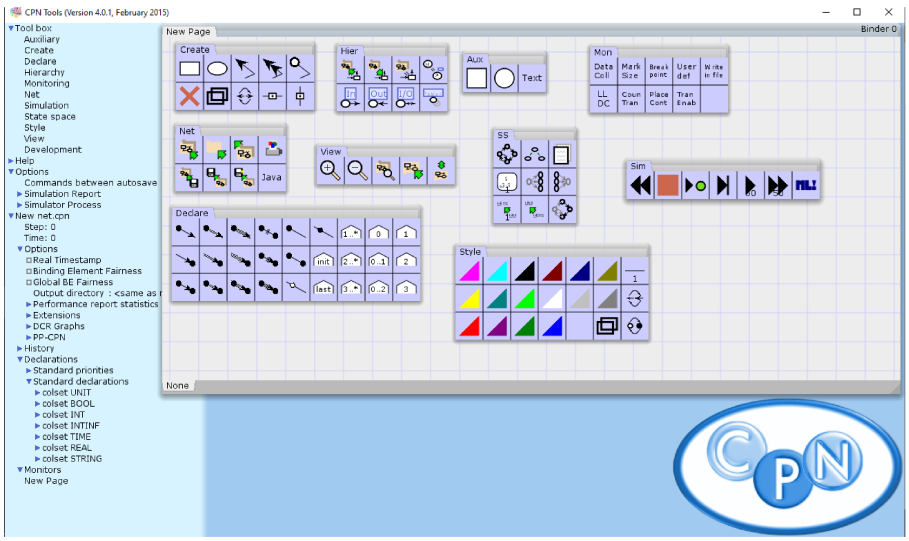


Figura 5 – Interface do CPN Tools.

O CPN Tools possui um ambiente que integra a área para criação gráfica da rede com a área para a declaração dos tipos de dados das variáveis e das funções, entre outros. Assim, o usuário trabalha diretamente com a representação gráfica do modelo de rede de

Petri. A Figura 6 ilustra um exemplo de uma CP-net construída na ferramenta, com os elementos básicos de uma rede de Petri Colorida indicados com comentários em vermelho. A ferramenta não aceita a inserção de caracteres especiais como os acentos, por exemplo.

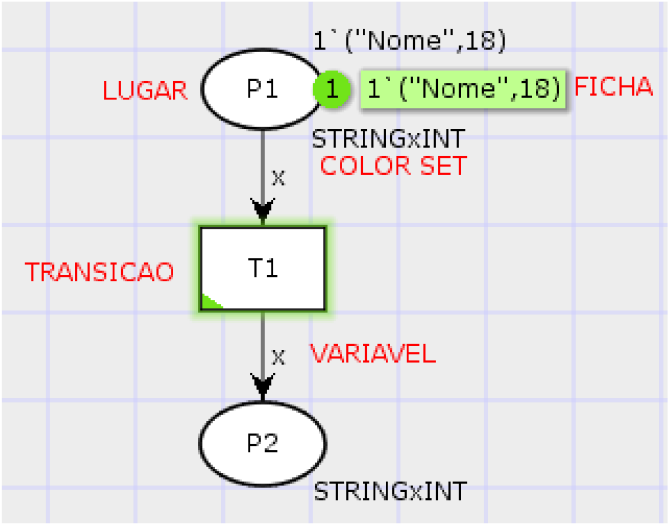
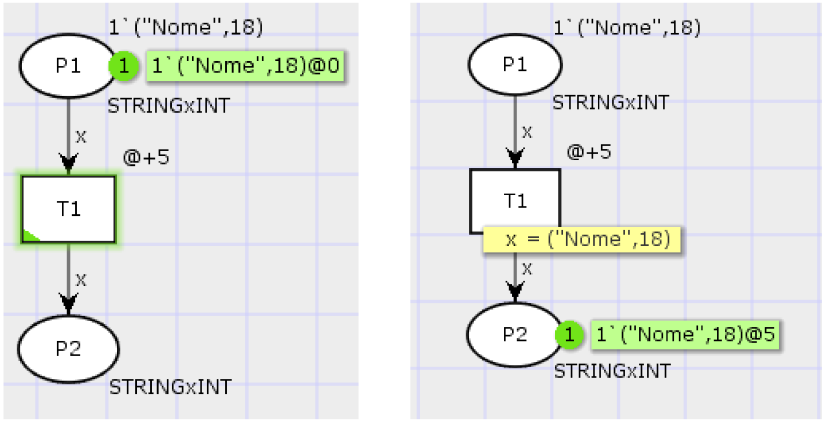


Figura 6 – Exemplo de uma rede de Petri Colorida.

O exemplo da Figura 6 contém dois lugares, *P1* e *P2*, que possuem o tipo de dado (*color set*) *STRINGxINT*, o qual também é atribuído para a variável x associada aos arcos da rede. No lugar *P1*, a inscrição *1‘(“Nome”,18)* corresponde a uma ficha (*1‘*) cujos atributos são dados pela string (*“Nome”*) e pelo valor inteiro (*18*).



(a) CP-net Temporizada antes do disparo.

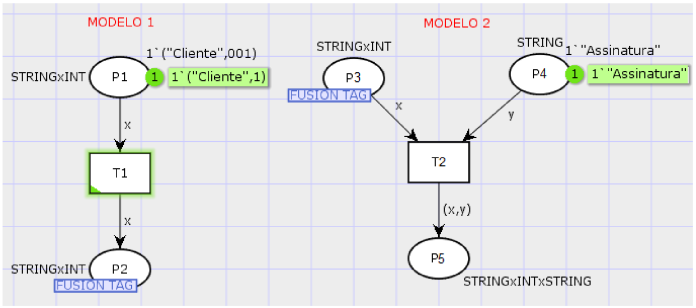
(b) CP-net Temporizada após o disparo de T1.

Figura 7 – Exemplo de funcionamento de uma rede de Petri Colorida Temporizada.

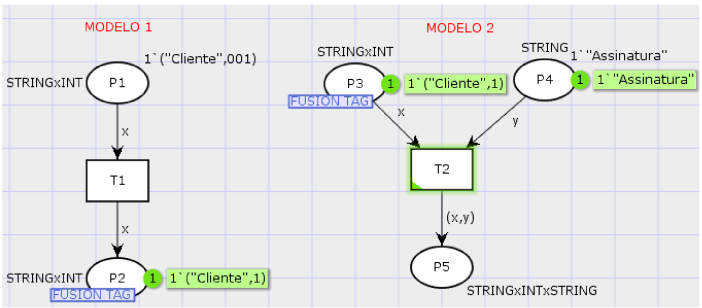
A fim de exemplificar o funcionamento de uma rede de Petri Colorida Temporizada no CPN Tools, considere a rede apresentada na Figura 7. Para adicionar a característica de tempo nas fichas é necessário que a declaração do *color set* seja feita de maneira adequada, adicionando a palavra reservada *timed* ao final da declaração.

Note que na Figura 7a a ficha presente no lugar *P1* apresenta uma declaração que indica que o tempo associado à ficha é @0 unidades de tempo. A inscrição de tempo @+5 na transição *T1* indica uma ação que demora 5 unidades de tempo para ser concluída. Dessa forma, toda vez que *T1* for disparada serão acrescentadas 5 unidades de tempo nas fichas produzidas por essa transição. A Figura 7b ilustra a rede após o disparo de *T1*, onde o tempo da ficha passa a ser @5 unidades de tempo.

O CPN Tools permite também estruturar os modelos das redes de Petri Coloridas em módulos, usando o recurso de hierarquia que a ferramenta oferece. O conceito de módulos em CP-nets é baseado em um mecanismo de estruturação hierárquica, com os já mencionados lugares individualmente chamados de *fusion places* e o *fusion set*, um conjunto de *fusion places*. A ideia básica da hierarquia por trás das CP-nets é permitir a construção de um amplo modelo combinando um número de pequenas redes de Petri Coloridas em um único modelo (JENSEN; KRISTENSEN, 2009). Não é necessária nenhuma declaração especial para a utilização dos *fusion places*, o CPN Tools trás essa funcionalidade no menu de opções.



(a)



(b)

Figura 8 – Exemplo da funcionalidade de *fusion places* disponível no CPN Tools.

Para exemplificar o funcionamento de *fusion places*, considere o exemplo da Figura 8 que apresenta dois modelos distintos. Os lugares *P1* e *P2* pertencem ao modelo 1 e os lugares de *P3* até *P5* pertencem ao modelo 2. O modelo 1 é a interface que coleta os dados do cliente, que são enviados em seguida para o modelo 2, onde é anexado às informações do clientes uma assinatura. O lugar *P2* e o lugar *P3* são marcados com uma *fusion tag*, representada por um retângulo azul, o que significa que esses lugares pertencem ao mesmo

fusion set. Quando *T1* é disparada e uma ficha é produzida em *P2*, automaticamente uma ficha também aparecerá no lugar *P3*, habilitando dessa forma *T2*.

Quando todos os membros de um *fusion set* pertencem a mesma parte ou página (no contexto de CPN Tools) da CP-net e tem apenas uma instância, os *fusion places* são nada mais que um mecanismo conveniente para evitar vários cruzamentos de arcos (JENSEN; KRISTENSEN, 2009).

2.6 SIMIO

O SIMIO é um *software* de modelagem e simulação para resolver problemas logísticos, empregado em diferentes áreas que vão desde gestão de negócios e operações até pesquisa operacional. É utilizado por empresas presentes em vários setores da indústria como, por exemplo, na saúde, manufatura, mineração, cadeias de abastecimento e transporte.

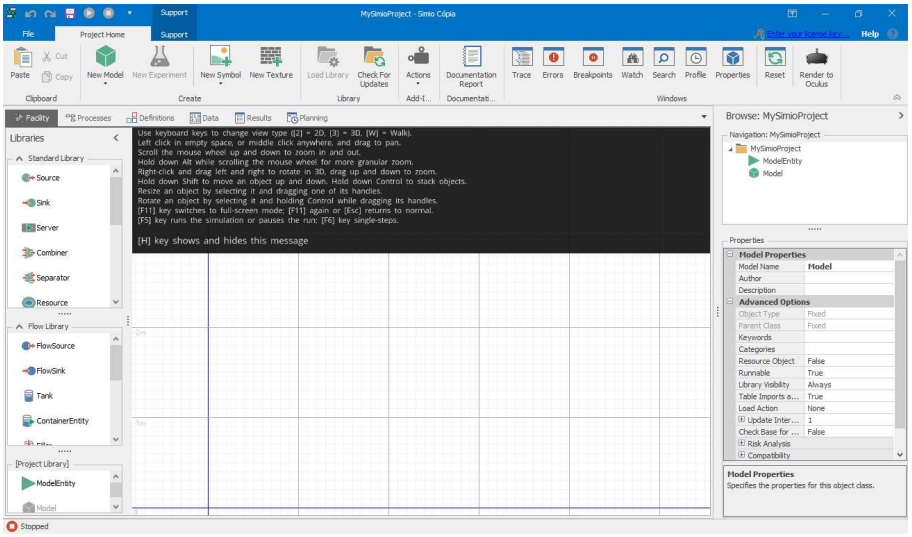
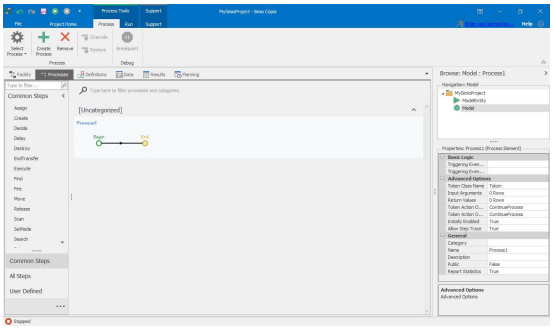


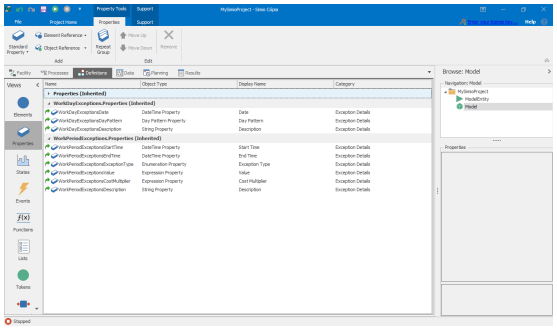
Figura 9 – Tela inicial do SIMIO — *Facility view* ou área de trabalho.

É um *software* jovem, que usa abordagens de modelagem avançadas indisponíveis em softwares tradicionais. Ele não requer um conhecimento prévio de uma linguagem de programação específica para definir as funcionalidades do modelo sendo desenvolvido, com processos e objetos sendo definidos graficamente. O SIMIO fornece um ambiente de modelagem 2D ou 3D multi-paradigma, combinando orientação a objetos e orientação a processos. Ou seja, todos os objetos utilizados no SIMIO são, na sua essência, objetos, mas toda a lógica inerente a eles é controlada através da utilização de processos que regem o seu comportamento. Estes processos são encapsulados nos objetos, mas novos processos podem ser utilizados para ampliar a lógica do seu comportamento (MOTA et al., 2017).

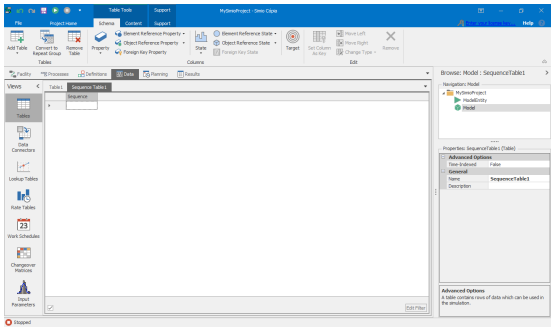
A interface gráfica, onde podem ser desenvolvidas os modelos, simulações e análise de dados, do SIMIO é amigável e composta por seis seções:



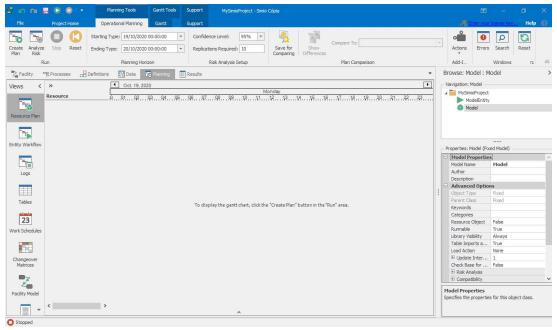
(a) Área de processos.



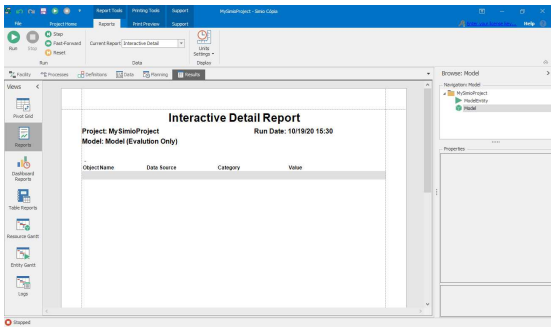
(b) Área de definições.



(c) Área de dados.



(d) Painel de controle.

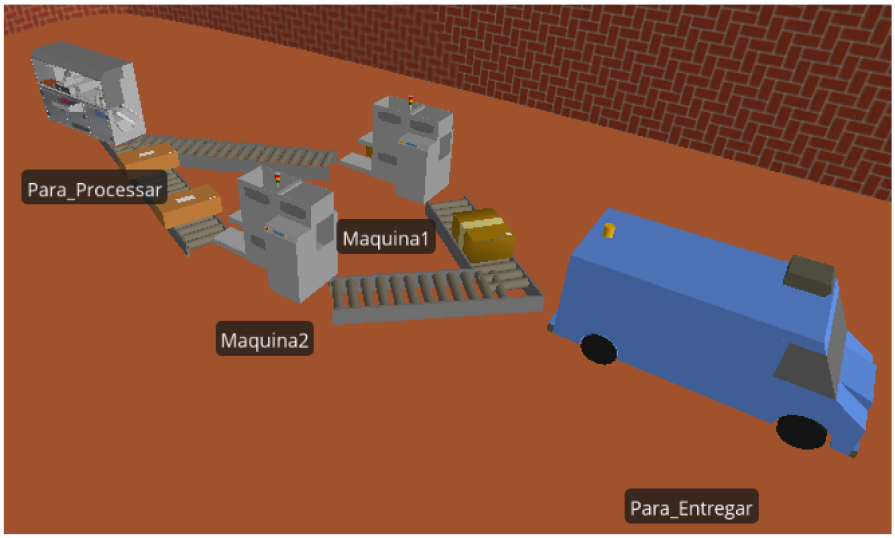


(e) Área de resultados.

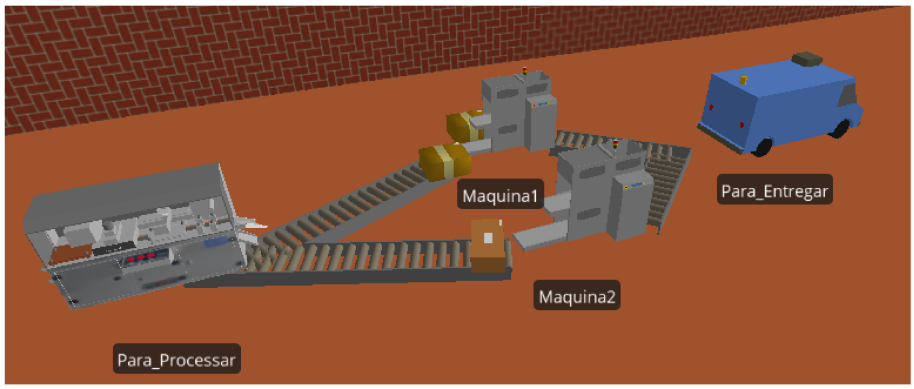
Figura 10 – Demais seções do SIMIO.

1. *facility view* ou área de trabalho (Figura 9): onde são desenvolvidos os modelos;
2. a área de processos (Figura 10a): utilizada para estender as funcionalidades dos objetos;
3. a área de definições (Figura 10b): onde são adicionadas variáveis, propriedades e outros elementos;
4. a área de dados (Figura 10c): usada para definir propriedades, como também para agendar, listar ou organizar informações através da utilização de tabelas;
5. o painel de controle (Figura 10d): serve para monitorar a evolução dos objetos no SIMIO;
6. e a área de resultados (Figura 10e): onde os resultados podem ser analisados.

A Figura 9 mostra a tela inicial do programa, que se abre estando na *facility view*. Os demais componentes citados acima podem ser acessado diretamente por essa tela, trocando-se a aba. As Figuras 10a, 10b, 10c, 10d e 10e mostram as demais seções. Todos elas possuem uma série de submenus, nos quais o usuário pode acessar diversas funcionalidades da ferramenta.



(a)



(b)

Figura 11 – Exemplo de execução de uma simulação 3D no SIMIO.

A Figura 11 apresenta *screenshots* da execução de uma simulação 3D de exemplo no ambiente do SIMIO. No cenário retratado há três máquinas ligadas por esteiras rolantes. A primeira, chamada de "Para_Processar", recebe dois tipos de pacotes (distinguíveis na simulação por apresentar diferentes aparências) e os encaminha para a próxima máquina correta para processamento. Os pacotes de tipo 1 vão para a "Maquina1" e os pacotes de tipo 2 vão para a "Maquina2". Ao final, todos os pacotes prontos seguem para a mesma saída no modelo, representada por um veículo chamado "Para_Entregar". As máquinas 1 e 2 processam um pacote por vez, os demais pacotes aguardam em uma fila.

Os tempos de processamento de cada máquina e chegada entre pacotes, a ordem de entrada e saída das filas e as funções de decisão que encaminham os pacotes corretos para

cada máquina são customizáveis dentro do *software*, da mesma forma como as aparências dos elementos e do ambiente da simulação também são.

2.6.1 Objetos no SIMIO

Um objeto é utilizado e criado na *facility view* e representa um componente físico do sistema, tal como uma pessoa, uma máquina ou um caminho, por exemplo. O comportamento de um objeto é controlado pela sua definição; logo se a definição mudar, todos os objetos derivados dessa definição também mudarão. Isso ocorre graças a propriedade de herança da orientação a objetos. A Figura 12 ilustra os tipos de objetos e a hierarquia presente no SIMIO.

Todos os objetos são derivados de uma classe geral que determina sua principal funcionalidade. Em particular, os diferentes objetos disponíveis na biblioteca inicial do SIMIO têm um comportamento específico derivado do comportamento da classe básica (MOTA et al., 2017).

Os objetos possuem três componentes principais:

1. modelo (comportamento);
2. interface (propriedades, estados, eventos);
3. representação externa (nós de entrada/saída, gráficos).

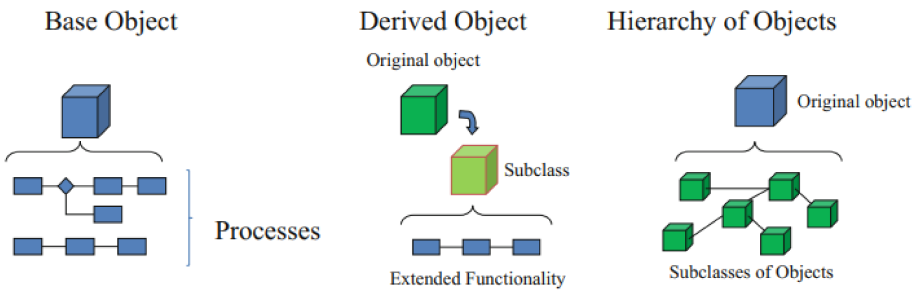


Figura 12 – Tipos de objetos e hierarquia. Figura retirada de (MOTA et al., 2017).

Qualquer modelo pode ser transformado em um objeto se uma interface e uma representação externa forem adicionadas. Um projeto pode ter um "modelo principal" e um ou mais "sub-modelos", que são usados como objetos.

É por padrão já definido que um modelo no SIMIO possuí uma "entidade modelo", que pode ser usada como uma entidade "burra" ou pode ser aprimorada ao se atribuir para ela estados, propriedades, atributos, representação externa ou uma lógica particular.

2.6.2 Migração das redes de Petri Coloridas para o software SIMIO

Na abordagem proposta em Mota et al. (2017), os seguintes elementos são apontados como os mais importante para se construir no SIMIO um modelo de simulação baseado em uma rede de Petri:

- ***Fixed Objects* (objetos fixos)**: um *fixed object* possui uma localização estática na área de trabalho (*facility view*). Os objetos têm nós ou portas associadas através das quais as entidades entram/saem de um objeto fixo. Entre eles podemos citar os objetos *Source*, *Server* e *Sink*;

- ***Links* (ligações)**: os *links* definem um caminho para as entidades e transporte entre dois nós, podendo ser unidirecionais ou bidirecionais. Estas conexões podem ser utilizadas para construir redes. Os *links* têm um peso que pode ser usado para selecionar um caminho, ou uma condição pode ser associada a cada *link*;

- ***Nodes* (nós)**: os nós definem o ponto inicial e final para um ou mais *links*. Eles também são utilizados para modelar a interseção de vários *links* de entrada/saída. Os nós podem definir os pontos de entrada e saída de um objeto fixo associado;

- ***Entities* (entidades)**: as entidades são os objetos que fluem através dos *links*, entrando e saindo dos objetos. Elas também podem pertencer a uma rede particular (determinada por nós e *links*) e seguir uma sequência de rotas específica. Dado que as entidades são objetos no SIMIO, pode ser atribuída a elas uma lógica que controla especificamente o seu comportamento. Geralmente, as entidades são geradas pelo objeto *Source*, mas também podem ser geradas por meio do uso de um processo interno ao objeto e sob condições específicas;

- ***Stations* (estações)**: as estações são elementos que se encontram na área de definições. O conceito de estação no SIMIO significa um local onde as entidades que fluem podem ser localizadas dentro do objeto. Nesse sentido (e por fazerem parte da anatomia de um objeto no SIMIO) podem ser usados para estender as funcionalidades de um objeto através da herança, a fim de desenvolver um novo objeto e ser usado de forma independente no modelo para armazenar entidades;

- ***Attributes* (atributos)**: o conceito de atributos é implementado no SIMIO por meio dos chamados *states* ou *states variables*. Estas variáveis no SIMIO correspondem a atributos já associados ao modelo global, como o objeto pai ou aos objetos que fazem parte do modelo global, que conceitualmente podem ser associados como variáveis locais em qualquer linguagem de programação. Sua principal característica é que os valores que adquirem mudam conforme a simulação prossegue;

- ***Properties* (propriedades)**: o conceito de propriedades é apresentado no SIMIO por meio das chamadas propriedades do objeto. Tais propriedades são parâmetros que definem a morfologia do objeto. O objetivo principal é que esses parâmetros sejam definidos durante o processo de desenvolvimento do modelo. Como essas propriedades são parâmetros que definem o comportamento do objeto, elas podem ser adicionadas a seu próprio critério, para que deste modo seja obtido um modelo mais adequado. Sua principal característica é que seus valores são definidos no início da simulação e não serão alterados durante a execução da simulação;

- ***Tokens* (fichas)**: o conceito de *token* é inerente ao SIMIO, mas tem diferente significado de seu uso na semântica das redes de Petri. No SIMIO um *token* serve para gerir as etapas dentro da área de processos (a execução lógica das etapas). Graficamente não é possível visualizar esses *tokens*, mas os diferentes processos que definem a função lógica de um objeto são executados através do fluxo virtual dos *tokens* nas etapas que compõem um processo.

Não há uma equivalência direta entre as redes de Petri comuns ou Coloridas e o SIMIO. Porém, com os principais componentes do ambiente SIMIO já apresentados, mostraremos agora como Mota et al. (2017) sugere que eles sejam utilizados na implementação dos elementos e características básicas de uma rede de Petri, de forma que o comportamento dinâmico originalmente estabelecido no modelo em rede de Petri Colorida seja mantido na construção do protótipo no SIMIO:

- ***Tokens* (fichas)**: as entidades que fluem pela rede no SIMIO são usadas para se conseguir uma correspondência com as fichas das redes de Petri. Elas podem não possuir atributos, equiparando-se a fichas simples de uma rede de Petri, mas ao definirmos atributos através do uso de *states* para essas entidades elas passam a corresponder a uma ficha de rede de Petri Colorida com atributos (cores). Os atributos podem ter diferentes tipos de valores tais como inteiros, *boolean* e *string* por exemplo;

- **Lugares**: os lugares correspondem às estações (*stations*) do SIMIO. A funcionalidade das estações no SIMIO é armazenar as entidades, que podem ser visualizadas através das filas na *facility view*. Elas fazem parte de quase todos objetos da biblioteca padrão do SIMIO (como em um objeto *Server*, por exemplo). Se necessário as estações também podem existir como objetos independentes;

- **Transições**: graças à arquitetura do SIMIO eventos e transições de uma CP-net podem ser implementados na lógica dos objetos. Tendo em vista que cada objeto possui por padrão um determinado número disponível de eventos, como o *Run Initialized*, *Entered*, *After Processing*, *Exited*, *Run Ending* entre outros, cabe ao modelador determinar quais processos devem ser executados (os quais podem ser criados pelo

próprio usuário com as etapas na área de processos) e quais condições devem ser conferidas em determinados eventos, de forma a seguir a lógica da CP-net original. Esses ajustes podem ser inseridos e configurados no submenu de propriedades que os objetos possuem. Assim, no momento em que uma entidade (representando uma ficha de uma rede de Petri) entrar em um objeto, ser processada ou tiver saindo de um objeto, por exemplo, esses eventos escolhidos irão gerenciar o seu comportamento guiando-se pela semântica da CP-net implementada;

- **Eventos condicionais**: nas redes de Petri Coloridas, eventos condicionados são modelados adequadamente através do uso de restrições que são implementadas utilizando-se os pesos de arcos, guardas associadas à transições e expressões de arco. No SIMIO tais condições podem ser modeladas utilizando-se as estações (*stations*) e a área de processos. As estações podem ser usadas como um elemento independente ou como estações pré-localizadas nos objetos do SIMIO, presentes na biblioteca padrão do SIMIO. Neste último caso, pode-se utilizar as funcionalidades dos objetos, tais como um processo de *seize* ou *delay*. Estas funcionalidades podem ser atribuídas a um objeto *Server*, por exemplo.

  Na área de processos, adicionamos às estações um ou mais fluxos de processos lógicos, definidos por um fluxograma composto de etapas (*steps*), as quais são executadas pelos *tokens* padrões do SIMIO. Conforme a restrição desejada, cabe ao modelador escolher as etapas apropriadas para serem adicionadas ao fluxograma. Por exemplo, para avaliação e decisão de restrições impostas por expressões de arcos em uma CP-net, pode se utilizar a etapa *Decide*. Esta etapa é equivalente a um *IF.THEN.ELSE* de qualquer linguagem de programação e em seu submenu de propriedades pode-se configurar os tipos de condições e as expressões que devem ser satisfeitas. Outro exemplo é a etapa *Search*, a qual realiza uma busca pelas entidades presentes na estação e, assim, pode ser é usada para verificar a existência de entidades que satisfaçam certas condições;

- **Sincronização**: há várias maneiras de se implementar no SIMIO a sincronização de diferentes processos que comportam atividades paralelas ou não, com dinâmicas diferentes. Pode-se usar objetos da biblioteca padrão do SIMIO, os quais possuem a dispor propriedades que especificam que o processo só poderá ser iniciado quando todos os recursos especificados estejam disponíveis, como em um objeto *Combiner*.

  Uma outra forma de se obter a sincronização é utilizando também as estações e a área de processos. Novamente as estações podem ser usadas como um elemento independente ou como objetos da biblioteca do SIMIO criados na *facility view*. A implementação do sincronismo consegue ser realizada através da área de processos, estabelecendo-se processos de condições com as já mencionadas etapas. A etapa *Decide* pode ser também aqui empregada, verificando se existem pelo menos uma

entidade (ficha) em cada um dos locais onde queremos modelar a sincronia. A Figura 13 traz um exemplo de um processo utilizando a etapa *Decide* e o seu fluxograma, onde a condição definida no submenu de propriedades confere se há pelo menos uma ficha no lugares que se deseja sincronizar, P1 e P2. Caso a condição seja satisfeita, é utilizada a etapa *Transfer* para que a transição habilitada seja disparada e a ficha (entidade) siga para o próximo lugar.

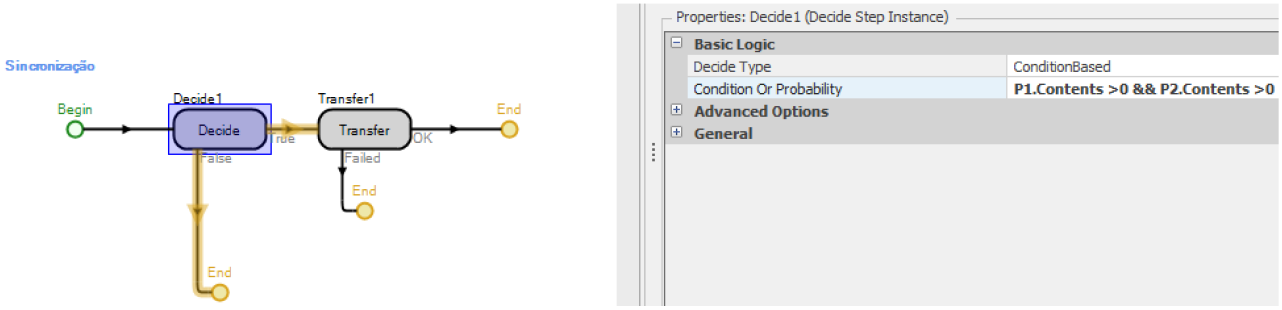


Figura 13 – Fluxograma do processo que utiliza a etapa *Decide* para sincronização.

- **Paralelismo**: a dinâmica da evolução de atividades paralelas é facilmente modelada no SIMIO utilizando-se os objetos fixos ou estações e as etapas presentes na área de processos. Para ilustrar uma das possíveis formas de construir paralelismo no SIMIO considere o seguinte exemplo: dois processos serão acionados a partir de um evento em comum, podemos então criar um processo que é disparado pela ocorrência deste acontecimento. A Figura 14 ilustra esse exemplo em uma CP-net, com a transição T1 sendo o evento específico que iniciará as atividades paralelas e os lugares P1 e P2 os destinos que precisam receber uma ficha cada simultaneamente.

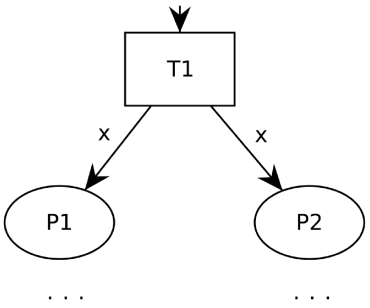


Figura 14 – Parte de uma CP-net com paralelismo.

Como será necessária uma entidade (ficha) além da original, a etapa *Create* é utilizada para criar um novo objeto, neste caso uma nova ficha. Assim, temos as duas fichas necessárias para P1 e P2. Por causa da lógica de roteamento de entidades no SIMIO, a etapa *Set Node* é empregada para indicar a direção em que as entidades criadas devem seguir para que os processos independentes avancem. Além do uso de etapas *Set Node*, a lógica do SIMIO permite que as entidades também possam ser roteadas atrás do peso de *links* ou por tabelas de sequências. Por fim, usamos a

etapa *Trasnfer* para enviar as entidades de seus pontos de origem até seus pontos de inserção no modelo SIMIO, sejam eles uma estação, um nó ou um objeto. O uso da etapa *Trasnfer* pode ser optativo para a entidade original se a saída do objeto estiver diretamente ligada ao próximo processo ou objeto. Já a entidade criada com a etapa *Create*, que não existia originalmente e, portanto, sua localização inicial é *FreeSpace* no momento em que é criada, a etapa *Trasnfer* deve ser usada para especificar que a entidade está sendo transferida do *FreeSpace* para um local específico no modelo (o processo paralelo de destino). A Figura 15 mostra o fluxograma deste processo.

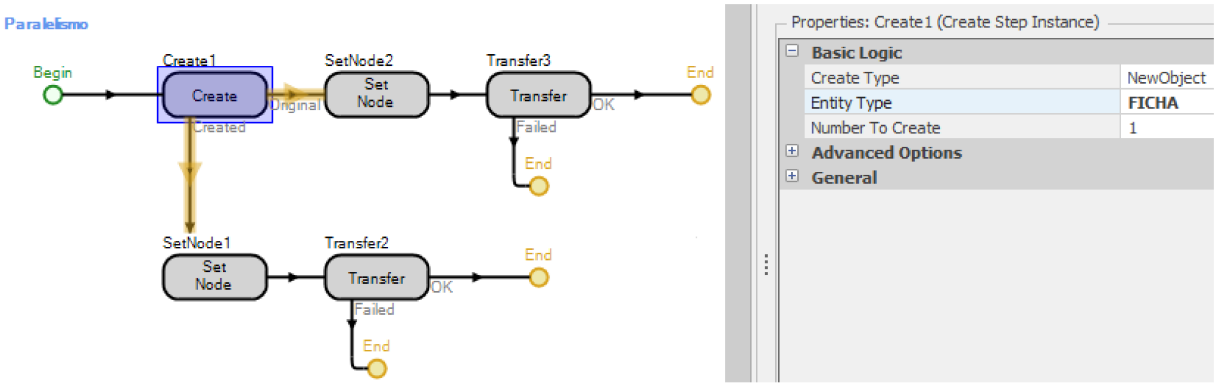


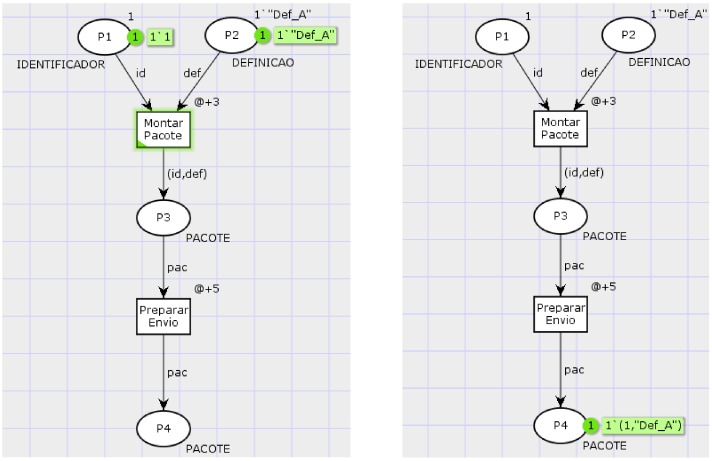
Figura 15 – Fluxograma do processo para construir paralelismo no SIMIO.

- **Temporização**: para modelar uma CP-net temporizada no SIMIO, o consumo de tempo pode ser calculado com as estações e os processos, utilizando a etapa de *Delay*. Uma vez que as condições de uma transição tenham sido satisfeitas, o tempo especificado para aquela transição será consumido. Além de estações e processos também podem ser utilizados os objetos encontrados na *facility view*, tais como o *Server*, *Path* ou *Vehicle* por exemplo, que possuem por padrão a propriedade da etapa *Delay*.

- **Processos e recursos**: em uma rede de Petri um processo pode ser modelado de várias formas, seja como uma única transição ou como sendo uma rede inteira, onde recursos são gerados, consumidos, ficam retidos e disponíveis ao decorrer da execução. No SIMIO conseguimos implementar essa característica utilizando-se *stations* e etapas de processos, como a *Set Node*, *Decide*, *Seize*, *Delay*, *Release* ou até mesmo utilizando-se objetos padrões que possuem a referida lógica em suas propriedades, como o *Server*. Para usar tais objetos, basta adicioná-los na *facility view* e especificar nas propriedades o(s) recurso(s) que é/são necessário(s) para realizar a atividade.

  A etapa *Seize* torna possível apreender os recursos, podendo ser elementos (fichas) ou objetos, que serão usados no processo. Com a etapa *Delay* o tempo que a operação com esses recursos irá levar é especificado. Ao final desse tempo a etapa *Release*

indica que o processo foi encerrado e, portanto, os recursos usados podem ser liberados. Como já visto anteriormente, as etapas *Set Node*, *Decide* e outras podem ser utilizadas para se sincronizar ou definir a sequência de ações que se deseja seguir.



(a) (b)

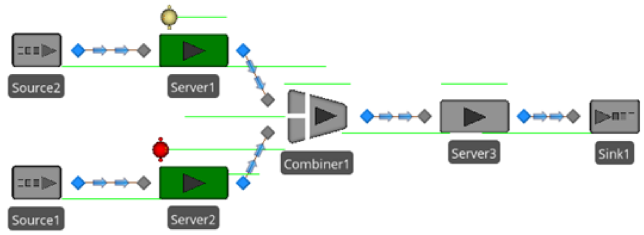
Figura 16 – Exemplo de uma CP-net.

A Figura 16 apresenta um modelo de rede de Petri Colorida que coleta informações nos lugares P1 e P2, um código identificador (do tipo inteiro) e uma descrição (do tipo *string*) respectivamente, produzindo em seguida um pacote (tipo de dado formado pelo produto cartesiano IDENTIFICADORxDEFINICAO) com esses dados. Após isso o pacote é preparado e segue para envio, deixando a rede. A transição *Montar Pacote* gasta 3 unidades de tempo e a transição *Preparar Pacote* consome 5 unidades de tempo. Desse modo o pacote enviado gasta 8 unidades de tempo no modelo e saí sendo composto por um identificador e um texto descritivo.

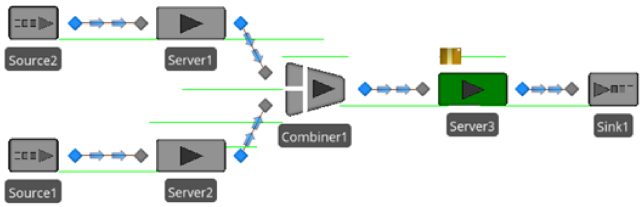
Para ilustrar uma forma de modelar tal rede no SIMIO, considere os *screenshots* apresentados na Figura 18 e na Figura 19. A primeira imagem traz a modelagem equivalente da CP-net em 2D, com a entidade que representa a ficha que carrega a informação do código identificador e a a entidade que representa a ficha com a descrição sendo distinguidas pelas cores bege e vermelho, respectivamente. A entidade que modela o pacote pronto é representada pela figura de uma caixa. As Figuras 17a e 17b mostram os valores dos atributos da entidade representando o pacote montado ao decorrer da simulação.

(a) (b)

Figura 17 – Informações da entidade.



(a)



(b)

Figura 18 – Modelo 2D no SIMIO.

A Figura 19 exibe uma captura de tela da simulação do modelo em 3D, onde foram escolhidos símbolos para representar os demais objetos e para construir o ambiente em que se sucede o processo.

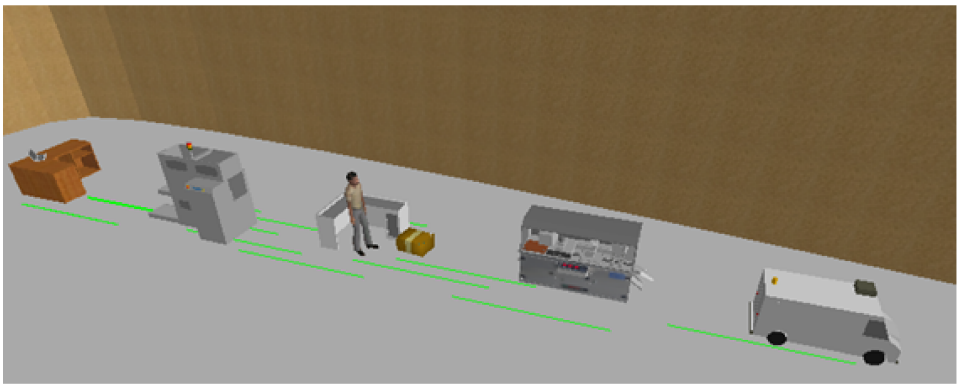


Figura 19 – Modelo 3D no SIMIO.

3 Trabalhos Correlatos

Para mostrar como os diferentes objetos em um jogo irão interagir de acordo com algumas ações que serão executadas pelo jogador, trabalhos como Rucker (2003) e Ang e Rao (2004) têm apresentado o uso de diagramas UML. Porém, de acordo com Oliveira, Julia e Passos (2011), diagramas UML não apresentam em uma maneira explícita os possíveis cenário que existem em uma missão ou nível de jogo.

Na abordagem de modelagem apresentada em Natkin, Vega e Grünvogel (2004), os autores abordam modelos para representar a lógica do jogo (sequência de ações) e o espaço virtual, utilizando para modelar a lógica do jogo o formalismo das redes de Petri. Nesta abordagem, uma rede de Petri denominada Rede de Transações determina o início e o fim de cada ação do jogador. Dessa forma, cada modelo representa um conjunto de atividades que podem ser executadas pelo jogador em determinado momento do jogo. Para modelar o espaço virtual, os autores utilizaram hipergrafos. Para unir as duas estruturas e representar o jogo num contexto global, Natkin, Vega e Grünvogel (2004) utilizaram um mecanismo denominado de conexões que tratam de substituir uma hiperaresta do hipergrafo pela árvore de alcançabilidade de uma rede de Petri, deste modo a cada vez que uma tarefa é realizada um lugar do mapa topológico é liberado e a hiperaresta substituída no modelo. No entanto, o uso de dois formalismos diferentes (redes de Petri e hipergrafos) torna o estudo global do modelo de jogo inviável. Dessa forma, para verificar a corretude do jogo, é necessário analisar os modelos separadamente de acordo com seus formalismos.

No trabalho de Araújo e Roque (2009), é ilustrado com um estudo de caso como as redes de Petri podem ser utilizadas com certas vantagens em relação a outras linguagens de modelagem. Os autores usaram diagramas baseados em redes de Petri para representar as possíveis ações dos jogadores em relação à objetivos de jogo. Assim, o fluxo do jogo pode ser mapeado e simulado pelas redes de Petri. Em Araújo e Roque (2009) é também apresentada uma modelagem baseada em uma abordagem hierárquica onde cada nível da hierarquia é composto por um conjunto específico de ações relacionadas.

Em Oliveira, Julia e Passos (2011) e Oliveira (2012) é apresentada uma nova abordagem baseada em um tipo particular de rede de Petri, denominado WorkFlow net, para especificar cenários existentes em um jogo. Nesta abordagem, os autores utilizaram uma WorkFlow net para representar o fluxo de atividades que deve ser executado pelo jogador a fim de alcançar um objetivo específico no jogo. Esse fluxo de atividades é associado à noção de *quest*, ou seja, uma missão que o jogador deve executar. De acordo com Oliveira, Julia e Passos (2011), em termos de modelo, cada *quest* é um subprocesso de uma WorkFlow net maior e a integração de diversas *quests* formam então uma rede de

quests e é por meio dela que o resultado global da análise do modelo do jogo é estabelecido. Como cada *quest* é um subprocesso, caso aconteça alguma mudança em uma *quest* já estudada, um novo estudo das boas propriedades será feito apenas para a *quest* que sofreu alteração. Além disso os autores realizaram uma análise qualitativa por meio de árvores de prova da lógica linear. De acordo com os autores, a tradução dos modelos em redes de Petri para árvores da lógica linear tem o objetivo de provar a propriedade *soundness* da rede que corresponde à consistência do cenário modelado do ponto de vista do jogo. Entretanto, nas abordagens apresentadas em Oliveira, Julia e Passos (2011) e Oliveira (2012) não foi apresentado um modelo para representar o mapa do mundo virtual do jogo onde as ações do jogador são executadas. Além disso, foi apresentada uma análise qualitativa usando um formalismo diferente das redes de Petri, sendo necessário realizar a tradução da rede de Petri para o formalismo da lógica linear.

Na abordagem apresentada por Lewis e Whitehead (2011), a lógica linear também foi utilizada para análise de jogos. Os autores explicam que o interesse de usar a lógica linear para análise de um cenário é a possibilidade de expressar ações e recursos do jogo sem reter informações inúteis. Na lógica linear, os personagens e recursos são representados por átomos, enquanto que as relações implicativas são usadas para projetar eventos. Esse método proporciona a verificação de propriedades de cenários de jogos que podem ser validadas antes da execução/implementação do jogo. Após expressar os cenários utilizando fragmentos da lógica linear, o modelo é traduzido numa rede de Petri. O modelo final obtido permite gerar então as possíveis narrativas para um dado cenário.

Em Lee e Cho (2012) é apresentado um mecanismo para gerar enredo de *quest* consistindo na representação de eventos baseados em redes de Petri. De acordo com os autores, um enredo de *quest* é uma sequência de eventos para alcançar um objetivo específico. Um evento consiste de ações que devem ser executadas. As rede de Petri são então utilizadas para representar um evento do jogo. Na rede, um lugar representa uma pré-condição para uma ação ou o armazenamento do resultado de uma ação, os arcos representam relacionamentos casuais entre uma ação e sua pré-condição, as transições representam as ações do jogador e as fichas representam o estado de uma ação em um evento. O gerador de enredo baseado em redes de Petri, proposto por Lee e Cho (2012), foi então experimentado em uma plataforma de jogo comercial. De acordo com os autores, além de ser aplicável, o método proposto proporciona um método baseado em roteiro mais formal. Além disso, as redes de Petri ajudaram a identificar elementos passivos (tais como as condições) e elementos ativos (tais como as ações) da história, o que facilitou a representação de eventos independentes, restrições e a sincronização de eventos.

No estudo realizado por Barreto e Julia (2017), é apresentada uma abordagem onde os cenários de *video games* são representados pela combinação de WorkFlow nets para representar as atividades que existem em um jogo e por redes de Petri do tipo Grafos

de Estado, representando as áreas do mundo virtual onde o jogador pode executar as atividades. A comunicação entre os dois modelos é dada por um mecanismo de comunicação síncrona baseado em redes de Petri Coloridas. Para verificar a corretude do modelo global obtido (modelo das atividades junto com o modelo do mapa), os autores usaram um algoritmo baseado na análise do espaço de estados (*state space*). Tal algoritmo pôde ser executado automaticamente por algumas funcionalidades existentes na ferramenta CPN Tools. Porém, a abordagem de Barreto e Julia (2017) considera apenas modelos não temporizados e, consequentemente, não pode ser usada para estimar o tempo de duração de um *video game*.

É também apresentado em Barreto, Freitas e Julia (2018) e em Barreto (2020) uma abordagem onde os cenários de *video games* são representados pela combinação de WorkFlow nets e por redes de Petri do tipo Grafos de Estado. Novamente a WorkFlow net é utilizada para representar as atividades que existem em um jogo e a noção de mapa que representa as áreas do mundo virtual onde o jogador pode executar as atividades é representada por um Grafo de Estado. Além disso, uma versão temporizada da WorkFlow net e uma versão temporizada do Gráfico de Estado é apresentada para produzir uma estimativa de tempo que corresponda à duração efetiva que um jogador precisará para completar um nível específico de um jogo. Ambos modelos são denominados, respectivamente, de Modelo Lógico Temporizado e de Modelo Topológico Temporizado. No Modelo Lógico Temporizado um tempo de duração é atribuído a cada atividade do modelo por meio de distribuições de tempo aleatórias exponenciais. Tal duração representa o tempo de execução que um jogador precisará para realizar uma atividade específica. Na versão temporizada do Modelo Topológico a passagem de uma área do mapa para outra também possui uma duração de tempo, associada a cada transição do modelo através do uso de distribuições de tempo aleatórias uniformes, as quais representam o tempo que um jogador levará para se mover de uma área para outra.

Em Barreto, Freitas e Julia (2018) e Barreto (2020) a ferramenta CPN Tools é utilizada para representar graficamente os modelos propostos. Para tanto, os modelos de jogo foram convertidos, praticamente sem perdas semânticas, em redes de Petri Coloridas. É levado em consideração que em um jogo o jogador só pode realizar uma tarefa se estiver na área apropriada, enquanto algumas áreas só podem ser acessadas após a realização de uma tarefa específica. Dessa forma o Modelo Lógico Temporizado e o Modelo Topológico Temporizado estão interligados e, juntos, formam o Modelo Global Temporizado do nível. Para representar a relação entre os dois modelos é implementado um mecanismo de comunicação síncrona usando o conceito de *fusion places*. A corretude do modelo global obtido é verificada utilizando algumas das funcionalidades automáticas existentes na ferramenta. O CPN Tools também é utilizado para mostrar, por meio de uma espécie de análise quantitativa, a influência de um modelo sobre o outro.

O CPN Tools, empregado nos trabalhos Barreto e Julia (2017), Barreto, Freitas e Julia (2018) e Barreto (2020), apesar de fornecer as ferramentas necessárias para a representação gráfica e hierárquica dos modelos, para a implementação das versões temporizadas e para realizar as análises e simulações, o *software* lida diretamente com uma combinação do modelo matemático das redes de Petri e com linguagens de programação funcional. Sendo assim, os autores destes trabalhos apresentam abordagens teóricas que requerem conhecimentos prévios dos formalismos empregados, até mesmo para a ferramenta de simulação utilizada.

Com o propósito de permitir que projetistas mais inexperientes ou pessoas com pouca familiarização com Engenharia de Software tenham uma melhor compreensão da evolução de simulações, sem exigir um conhecimento prévio do formalismo das redes de Petri ou de linguagens de programação, é apresentada em Mota et al. (2017) uma abordagem baseada em como realizar a migração de projetos modelados em redes de Petri para o *software* de simulação SIMIO. No entanto, os autores abordam apenas a modelagem de sistemas de análise de manufatura e cursos de modelagem e simulação que, apesar de abranger diferentes áreas que vão desde gestão de negócios e operações até pesquisa operacional, não chega a ser aplicado ao desenvolvimento de *video games*. Dessa forma e considerando a literatura correlata aqui apresentada, este trabalho propõe integrar a modelagem de cenários de jogos em redes de Petri Coloridas e a abordagem de mapeamento de equivalência entre os modelos em redes de Petri e o *software* SIMIO.

4 Modelagem de cenários de Video Games baseado em redes de Petri Coloridas

A primeira etapa do trabalho consistiu em um estudo da literatura, através da leitura das partes mais relevantes de livros como (CARDOSO; VALETTE, 1997; AALST; STAHL, 2011; JENSEN; KRISTENSEN, 2009), que proveram uma base para o entendimento dos conceitos básicos necessários para o desenvolvimento do trabalho, como a definição das redes de Petri Coloridas e trabalhos relacionados à modelagem de processos de negócios formalmente modelados usando redes de Petri Coloridas. Em particular, o livro (MOTA et al., 2017) apresentou os principais conceitos da integração do formalismo das redes de Petri Coloridas com o *software* de simulação 3D SIMIO. Após a compreensão desses tópicos e da familiarização com o funcionamento das ferramentas CPN Tools e SIMIO, necessárias para o desenvolvimento do projeto, tornou-se realizável o processo de modelagem, análise e simulação do estudo de caso de modelos de jogo selecionado.

Com o estudo de caso definido, este foi utilizado para a migração dos modelos do nível de jogo, implementados e analisados no CPN Tools com o formalismo das redes de Petri Coloridas, para a interface gráfica da ferramenta SIMIO, onde foi possível construir e simular protótipos de jogo representados visualmente em 2D e 3D. Logo após, foi feita a comparação dos resultados das duas abordagens utilizando, para a análise quantitativa, os valores mínimos, máximos e médios dos tempos de execução de cada modelo e avaliação das vantagens e limitações do uso de cada uma das ferramentas empregadas para a modelagem e simulação de cenários de *video games* baseados em CP-nets.

Este capítulo apresenta o processo de transição da lógica dos modelos de nível de jogo, construídos com o formalismo das redes de Petri Coloridas, para o *software* de simulação SIMIO. Tais modelos pertencem à abordagem apresentada em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) para modelagem de cenários de *video games*, empregando o formalismo das redes de Petri Coloridas com adição de tipos temporizados, onde é utilizado especificamente o primeiro nível do *video game* Silent Hill II para ilustrar a abordagem. Para este nível é construído um Modelo Lógico Temporizado, apresentado na seção 4.1, para representar a sequência e duração das atividades do nível de jogo, um Modelo Topológico Temporizado, apresentado na seção 4.2, para descrever as áreas do mundo virtual e os tempos de deslocamentos do jogador entre essas áreas, e um Modelo Global Temporizado, apresentado na seção 4.3, que é formado pela junção desses dois outros modelos.

4.1 Modelo Lógico Temporizado

Em (BARRETO; FREITAS; JULIA, 2018), a estrutura das atividades existentes em um jogo é associada com a estrutura de um processo de negócio e, assim, o Modelo Lógico Temporizado de um nível de jogo é definido como um tipo de WorkFlow net temporizada. Com o uso da ferramenta CPN Tools para representá-lo graficamente, o modelo é convertido em uma rede de Petri Colorida Temporizada, sem que ocorra perdas de semântica, onde no conjunto de cores há o *color set* temporizado *PLAYER*. Este *color set* é formado a partir do produto entre os *color sets* *P* (do tipo *player*) e *REAL* (do tipo primitivo de dado real); ele é associado a todos os lugares da rede. O conjunto de variáveis é composto pelas variáveis *p* (do tipo *P*) e *e* (do tipo *REAL*), que são associadas a todos os arcos do modelo.

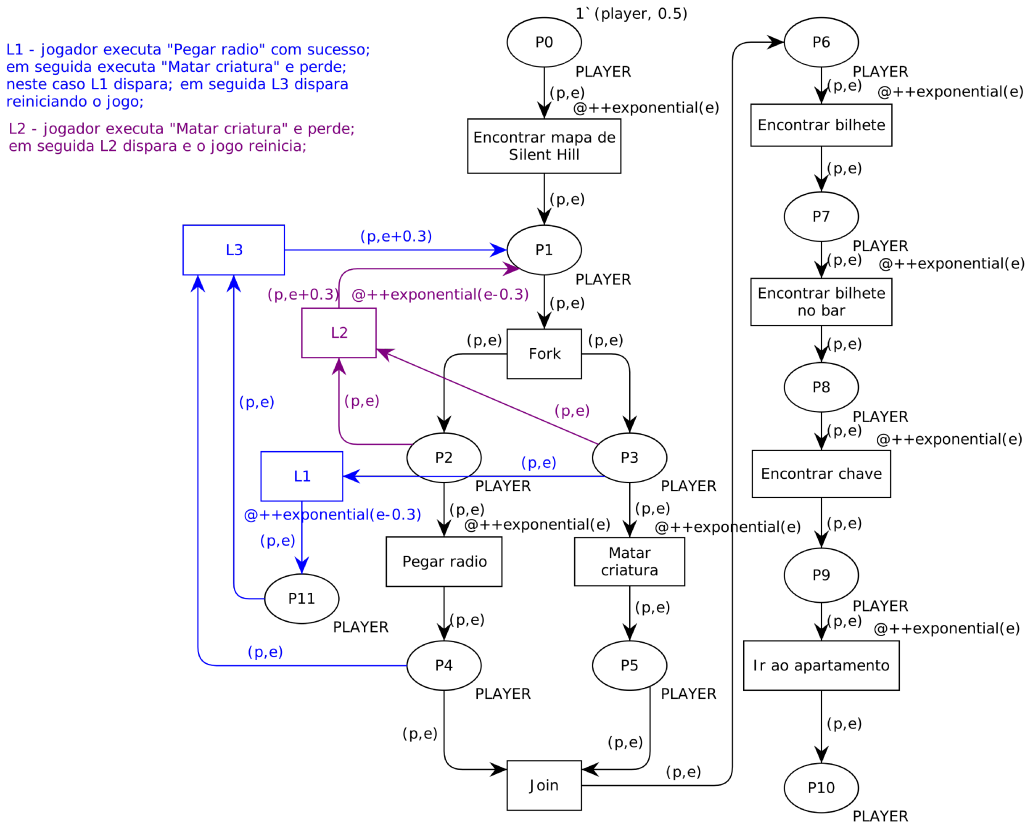


Figura 20 – Modelo Lógico Temporizado para o CPN Tools. Figura retirada de (BARRETO, 2020).

A Figura 20 apresenta a versão do Modelo Lógico Temporizado para o CPN Tools, no qual estão presentes as atividades (todas obrigatórias) que o jogador precisa realizar para terminar o primeiro nível do jogo, bem como uma duração anexada a cada uma dessas atividades que representa a quantidade de tempo (não nula) que leva para executá-las. Considerando que esse espaço de tempo gasto depende do nível de experiência do jogador, a função $exponential(e)$ produz um valor baseado na distribuição exponencial com média $1/e$, é associada a cada transição do modelo e corresponde a duração de cada atividade. O

parâmetro *e* representa o nível de experiência do jogador e é inicialmente definido com o valor 0.5, sendo somado a ele mais 0.3 unidades a cada vez que o jogador repetir o roteiro iterativo presente no modelo.

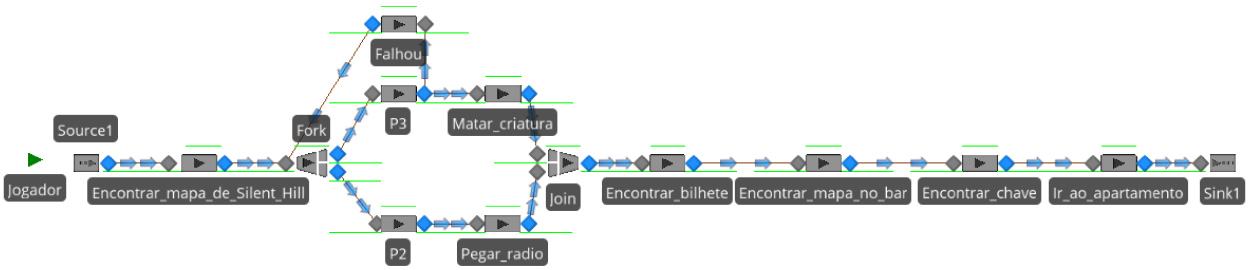
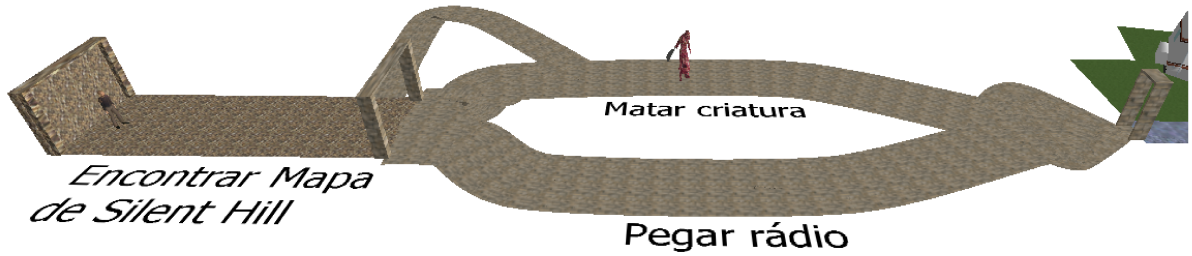


Figura 21 – Modelo Lógico Temporizado 2D no SIMIO.

Feita a migração da CP-net apresentada para o *software* SIMIO, temos inicialmente o modelo 2D apresentado na Figura 21 representando o Modelo Lógico Temporizado, construído de forma que não ocorram perdas semânticas. A versão 2D serve apenas como um "esqueleto" para o diferencial oferecido pelo SIMIO, o qual é poder exibir durante a simulação de uma versão 3D a evolução visual do cenário, de forma que ele possa ser compreendido até mesmo por aqueles que não são especialistas nas abordagens formais utilizadas.



(a) Primeira parte do Modelo Lógico Temporizado 3D no SIMIO.

(b) Segunda parte do Modelo Lógico Temporizado 3D no SIMIO.

Figura 22 – Modelo Lógico Temporizado 3D no SIMIO.

A fim de alcançar o proposto, são adicionados símbolos para alterar a aparência dos objetos e da entidade "Jogador" da versão 2D, bem como para criar os ambientes que

compõem a maquete construída para o modelo poder ganhar ares de um cenário de jogo. O *software* SIMIO possui uma série de símbolos, recursos de desenho e texturas que ficam à disposição do usuário, além de permitir novos símbolos serem desenhados, importados de arquivos *SketchUp* ou baixados da *3D Warehouse*. Novas texturas para a superfície de objetos podem também ser criadas a partir de imagens.

A Figura 22 traz o Modelo Lógico Temporizado 3D para o SIMIO, com o *screenshot* da maquete dividido em duas partes. Os cenários construídos neste projeto são meramente representativos e não buscam assemelhar-se com a aparência do nível correspondente presente no jogo Silent Hill II.

Para entender como a lógica da rede de Petri Colorida foi incorporada ao modelo do SIMIO, basta saber como foi feita a configuração dos objetos visíveis na Figura 21. Nela temos a entidade chamada de "Jogador" representando a ficha da rede de Petri, um objeto fixo *Source* ("Source1") para gerar somente uma entidade deste tipo dentro do modelo no momento de executá-lo e, no final, um objeto fixo *Sink* ("Sink1") para destruir essa entidade e registrar as estatísticas.

Entre "Source1" e "Sink1" existe para cada atividade, que deve ser realizada pelo jogador no nível, um objeto *Server* nomeado de acordo com a transição correspondente na CP-net equivalente e *paths*, para conectar os objetos e indicar a direção do fluxo das tarefas (através das setas azuis que podem ser vistas na Figura 21) que a entidade deve realizar pelo modelo. Em especial, para a transição *Fork*, que inicia as atividades paralelas e o roteiro iterativo do modelo, foi utilizado um objeto fixo *Separator* que recebeu o mesmo nome. Este objeto recebe a entidade "Jogador" e cria uma cópia dela. Logo a seguir envia uma dessas entidades para cada um de seus *Servers* de saída, "P2" e "P3". Também em especial, para a transição *Join*, que encerra a dinâmica de evolução das atividades paralelas, foi usado o objeto fixo *Combiner*, que devolve uma só entidade após receber uma de cada um de seus dois nós de entrada. Uma vez que uma entidade entra em um desses objetos, o seu comportamento é controlado pela semântica da CP-net incorporada a esses objetos, bem como a definição da próxima ação que especifica em que ponto do modelo SIMIO essa entidade deve então entrar.

Note que o modelo de rede de Petri Colorida para o CPN Tools da Figura 20 também possui lugares nomeados de *P2* e *P3*, porém na versão do SIMIO os *Servers* representando esses lugares são meramente ilustrativos, uma vez que a lógica associada à CP-net é adicionada ao protótipo do SIMIO após ser construída na área de processos, não dependendo necessariamente de "P2" e "P3". Os processos que constroem e imitam o comportamento das transições da CP-net poderiam ser, por exemplo, ativados no objeto *Separator* ou dentro dos *nodes* existentes antes que as entidades entrassem nos *Servers* que representam a execução das atividades. Da mesma forma, o *Server* "Falhou" que possui um significado puramente representativo é utilizado para indicar que o jogador falhou no

roteiro iterativo e terá que reiniciá-lo, voltando para a entrada do objeto "Fork".

A Figura 23 traz os eventos criados na área de processos para realizar a lógica das transições da CP-net no modelo do SIMIO. Para o caso das atividades *Matar criatura* e *Pegar radio*, que formam uma rota paralela e podem ser executadas em qualquer sequência, as quatro possíveis opções de desenrolamento apresentadas por (BARRETO; FREITAS; JULIA, 2018) são: (1) o jogador primeiro executa *Pegar radio*, em seguida *Matar criatura* com sucesso e segue para a próxima atividade do nível; (2) o jogador executa *Matar criatura* com sucesso, logo após executa *Pegar radio* e segue para a próxima atividade do nível; (3) o jogador executa *Matar criatura* e fracassa, consequentemente o jogo reinicia voltando para o início da rotina iterativa; (4) o jogador primeiro executa *Pegar radio*, mas fracassa em *Matar criatura* e consequentemente o jogo reinicia voltando para o início da rotina iterativa.

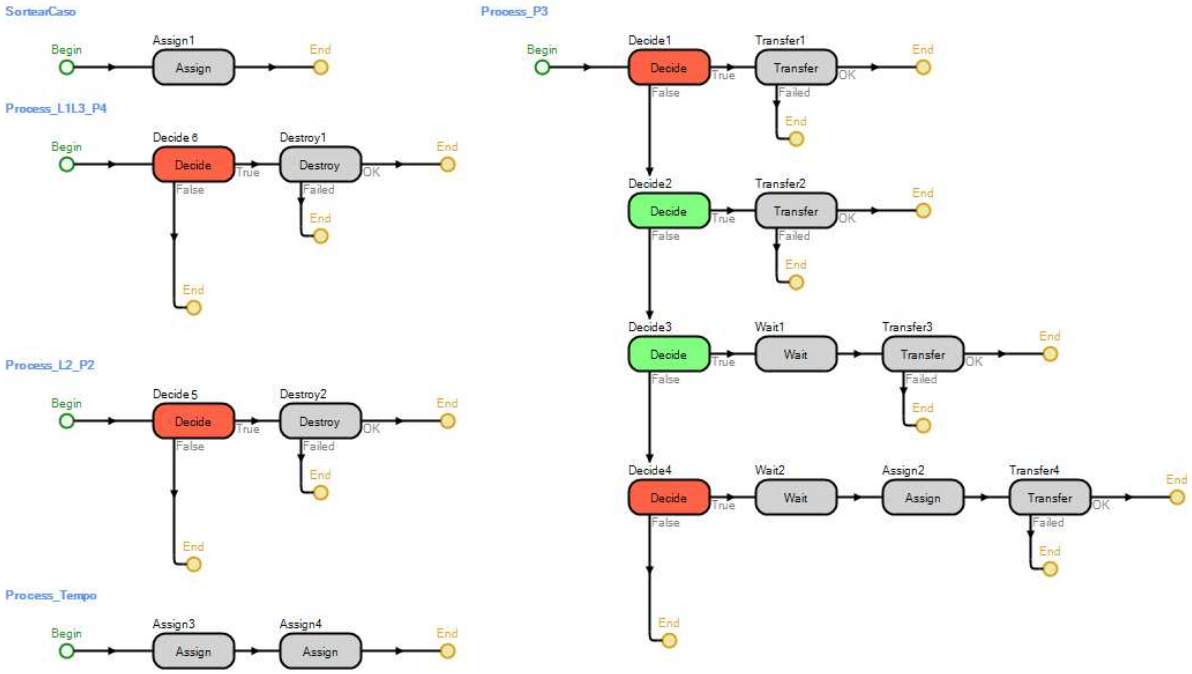


Figura 23 – Conteúdo da área de processo do Modelo Lógico Temporizado no SIMIO.

Estas opções são executadas na CP-net utilizando tanto as transições *Matar criatura* e *Pegar radio* como também as transições *L1*, *L2* e *L3*. No modelo do SIMIO o processo "SortearCaso", chamado para ser executado no objeto "Fork", realiza a escolha estocástica de qual acontecimento, representando as transições, será selecionado através de um sorteio entre essas opções, que são listadas em tabelas na área de dados. Feito isso, os processos "Process_L1L3_P4", "Process_L2_P2" e "Process_P3" são associados aos objetos *Servers* "Pegar_radio", "P2" e "P3", respectivamente.

O "Process_P3" em "P3" trata da entidade responsável por conseguir ou não executar com sucesso a atividade *Matar criatura*. As etapas *Decide* presentes no processo foram coloridas em verde para os sucessos em realizar a tarefa e em vermelho para os

fracassos, no caso de um sucesso as etapas *Transfer* são responsáveis por encaminhar "Jogador" para o *Server* "Matar_criatura" e no caso de fracasso para o *Server* "Falhou". As etapas *Wait* existentes são para as situações que a atividade *Pegar radio* deve ser realizada primeiro, logo a entidade em "P3" deve aguardar a segunda entidade finalizar o seu processamento no *Server* "Pegar_radio" antes de prosseguir.

O "Process_L2_P2" em "P2" é responsável por cuidar da entidade presente no *Server* "P2" em que o caso sorteado é o de executar a atividade *Matar criatura* primeiro e o jogador fracassa. Enquanto a entidade que está em "P3" é utilizada para representar o fracasso e continua no modelo retornando para a entrada do "Fork", a entidade em "P2" é destruída com a etapa *Destroy*, sumindo do modelo.

Já o "Process_L1L3_P4" no *Server* "Pegar_radio" lida com a entidade após concluído seu processamento no servidor na eventualidade da atividade *Pegar radio* ser a primeira a ser realizada, mas o jogador fracassar na tarefa *Matar criatura*. Novamente a etapa *Destroy* é utilizada para eliminar a entidade do modelo.

Por fim, o processo "Process_Tempo" é responsável por retornar o tempo de execução de cada atividade, usando a função *Random.Exponential(mean)* disponibilizada pelo SIMIO e ir somando tais valores para retornar no final o tempo total. A função exponencial(e) produz um valor baseado na distribuição exponencial com média $1/e$, como já apresentado a variável e representa o nível de experiência do jogador.

Todos os *Servers* responsáveis por representar a realização de uma tarefa do Modelo Lógico ("Encontrar_mapa_de_Silent_Hill", "Pegar_radio", "Matar_criatura", "Falhou", "Encontrar_bilhete", "Encontrar_mapa_no_bar", "Encontrar_chave" e "Ir_ao_apartamento") disparam "Process_Tempo" no momento em que a entidade entra no objeto em questão, o que corresponde à ocorrência do evento *Entered* do objeto. O processo além de atualizar o tempo global gasto nas atividades atualiza a variável (ou *state*, criado na área de definições) "Var_Delay", que será utilizada como tempo de processamento desses objetos. Dessa forma, a entidade "Jogador" permanece no *Server* apenas pelo tempo retornado pela função exponencial. Os demais objetos do protótipo que não requerem um intervalo de duração associado, mas que por serem objetos da biblioteca padrão do SIMIO possuem obrigatoriamente um tempo de *delay* ("Source1", "Fork", "P2", "P3" e "Join") tiveram seus tempos de processamento zerados.

4.2 Modelo Topológico Temporizado

As áreas que compõem o mundo virtual do nível de jogo são representadas em (BARRETO; FREITAS; JULIA, 2018) através de um Grafo de Estado, onde cada área é um lugar no grafo e as transições representam as fronteiras existentes entre áreas que se conectam. A fim de poder simular o tempo de deslocamento do jogador entre áreas, uma

função de tempo é adicionada a cada transição do Grafo de Estado. Para que o modelo seja representado graficamente na ferramenta CPN Tools, ele é convertido para a versão em rede de Petri Colorida Temporizada apresentada na Figura 24. Todos os lugares dessa rede pertencem ao tipo de dado *MAPA*, do *color set* temporizado *MAPA* que é composto pelo *color set P* (do valor *player*). O *color set REAL* (do tipo primitivo de dado real) também pertence ao conjunto de cores do modelo. O conjunto de variáveis é composto pelas variáveis p (do tipo P, que é associada a todos os arcos do modelo), a variável a (do tipo $REAL$) e a variável b (do tipo $REAL$).

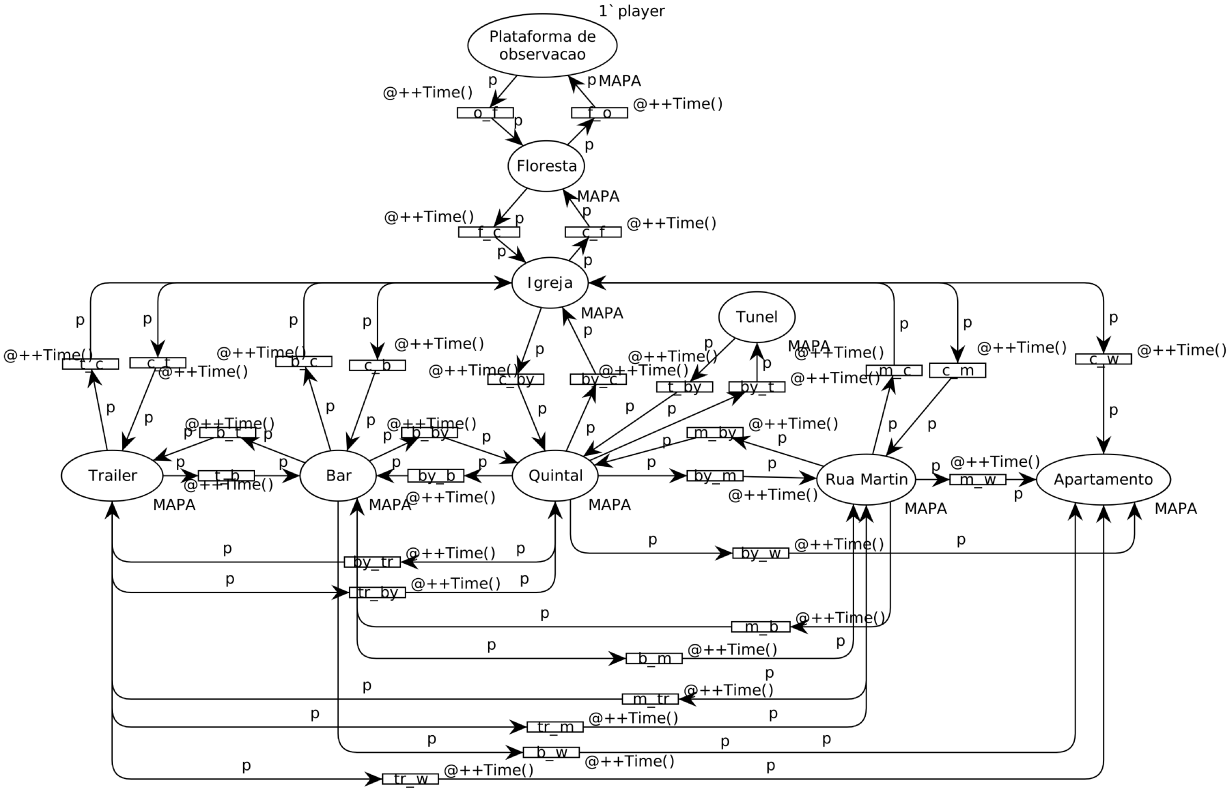


Figura 24 – Modelo Topológico Temporizado para o CPN Tools. Figura retirada de (BARRETO, 2020).

Na Figura 24 podemos ver também a função *Time()* associada às transições da rede para representar o tempo de movimentação do jogador pelo mapa, correspondendo a função de distribuição uniforme que usa as variáveis a e b como seus parâmetros mínimo e máximo, respectivamente. São escolhidos para esses parâmetros os valores a = 0.1 para a duração mínima e b = 1.0 para a duração máxima e, assim, o jogador pode gastar de 0.1 até 1.0 unidades de tempo para se locomover de uma área para outra.

O modelo 2D apresentado na Figura 25 corresponde ao Modelo Lógico Temporizado construído no SIMIO; é equivalente a versão de rede de Petri Colorida Temporizada do CPN Tools. A Figura 26 traz o Modelo Topológico Temporizado 3D para o SIMIO, no qual os ambientes da maquete que representam as áreas do mapa topológico foram construídos para visualmente retratar o local que as nomeiam, não levando em consideração

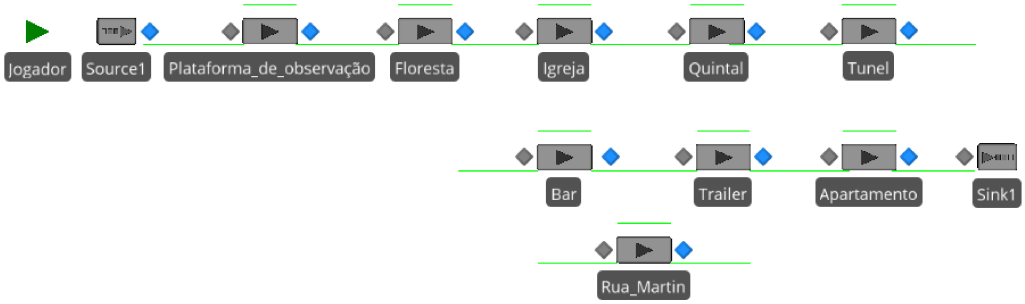


Figura 25 – Modelo Topológico Temporizado 2D no SIMIO.

atividades e tendo um único símbolo para representar o jogador conforme ele passa pelas áreas durante a execução da simulação.

Figura 26 – Modelo Topológico Temporizado 3D no SIMIO.1

Na construção desse protótipo, ao invés de utilizar *paths* para representar os possíveis caminhos entre as áreas, foi empregado um outro recurso disponibilizado pelo SIMIO: o método de roteamento *Select From List*, da propriedade *Routine Logic* responsável por definir o nó de destino para as entidades, a partir do nó de saída em que elas se encontram.

Para ilustrar, a Figura 27 traz o submenu de propriedades do nó de saída do *Server* "Igreja", exibindo-se em específico a configuração feita no atributo supracitado, onde o campo *Node List Name* contém a lista de possíveis nós de destino (neste caso a lista "NodeList_Igreja") e o campo *Selection Goal* indica, com o atributo *Random* selecionado, que o método de sorteio aleatório é o utilizado para se escolher o caminho no caso de haver vários candidatos possíveis na lista de nós de destino.

Todos os lugares da CP-net apresentada na Figura 24 que possuem fronteiras com pelo menos mais de um outro lugar, receberam no modelo do SIMIO uma lista exclusiva de nós de destino no nó de saída de seu objeto servidor equivalente. Dessa forma, todas as transições existentes na rede de Petri Colorida entre lugares adjacentes também existem virtualmente no modelo SIMIO, garantidas pela lógica de roteamento

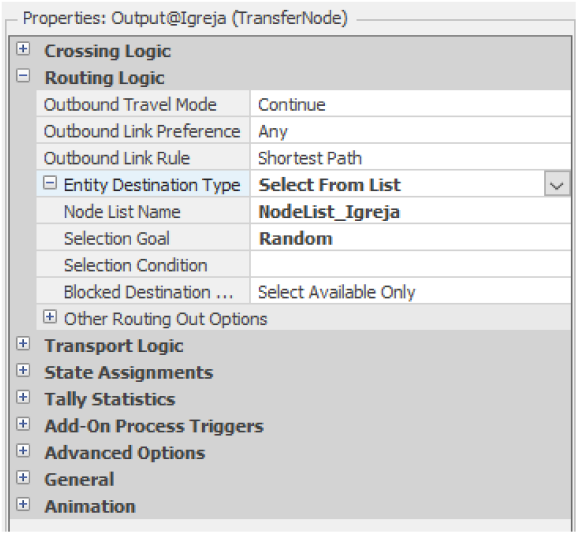


Figura 27 – Janela de propriedades do nó de saída do *Server* Igreja.

da ferramenta. Em especial, os *Servers* "Plataforma_de_observação" e "Apartamento" que possuem um único destino possível cada, não requerem uma lista de nós e foram configurados para despacharem a entidade diretamente para o objeto de destino, sendo eles, respectivamente, "Floresta" e "Sink1".

A Figura 28 apresenta a aba *Lists* na área de definições com todas as listas de nós utilizadas no modelo, com a "NodeList_Igreja" selecionada exibindo o seu conteúdo: os seis possíveis nós de entrada dos servidores que podem ser acessados a partir da área *Igreja*.

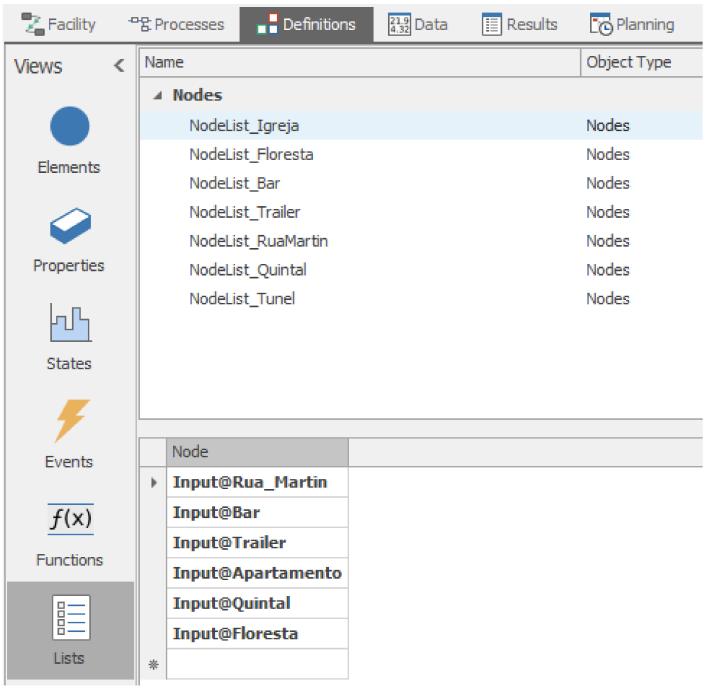


Figura 28 – *Nodes lists* na área de definições do Modelo Topológico Temporizado no SIMIO.

Além dos recursos já citados, é também utilizado no Modelo Topológico Temporizado construído no SIMIO o processo "Process_Tempo". A única diferença deste do processo já citado de mesmo nome existente no Modelo Lógico Temporizado do SIMIO é a função utilizada para se calcular o gasto de tempo. No Modelo Topológico, a função usada é a *Random.Uniform(min, max)* que recebe, como na versão da CP-net, os valores 0.1 para o parâmetro *min* e 1.0 para o parâmetro *max* e é empregada para retornar o tempo de deslocamento entre uma área e outra. Este valor é atribuído à variável "Var_Delay" e o processo é também responsável por atualizar o tempo total gasto em movimentação pelo jogador.

Todos os *Servers* presentes no modelo, com exceção de "Apartamento" que não leva a nenhuma outra área disparam "Process_Tempo" logo que a entidade entra no objeto em questão, dessa forma o tempo que o jogador irá gastar para ir deste lugar para o próximo é calculado neste momento. Como não é possível adicionar um tempo de deslocamento ao roteamento utilizando *Nodes Lists*, esse tempo é associado de forma ilustrativa ao tempo de processamento do servidor. Assim, o tempo total de deslocamento entre áreas gasto pelo jogador será igual ao tempo gasto pela entidade "Jogador" no sistema.

Para a realização deste trabalho foi utilizada a versão *SIMIO Personal Edition* do *software*, disponibilizada gratuitamente em <www.simio.com> e completamente funcional, apesar de contar com uma restrição no tamanho do modelo que se pode construir. Sem essa limitação, em uma outra versão do *software* que não possui um número limite de objetos que se possa adicionar ao modelo, uma alternativa ao uso da criação das possíveis rotas entre os *Servers* utilizando as *Nodes lists* seria utilizar o objeto *TimePath*, para representar os possíveis caminhos entre os servidores. O *TimePath* é um objeto que pode ser usado para definir um caminho entre duas localizações de nós, onde o tempo de deslocamento é especificado pelo usuário. A Figura 29 apresenta o modelo 2D no SIMIO de uma parte do mapa topológico do primeiro nível de Silent Hill II utilizando esta abordagem.

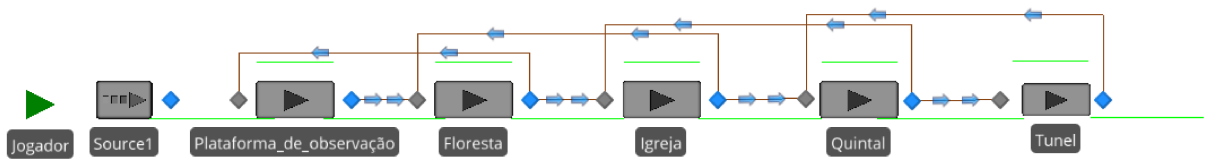


Figura 29 – Modelo 2D no SIMIO de parte do mapa topológico do primeiro nível de Silent Hill II utilizando *TimePaths*.

4.3 Modelo Global Temporizado

Em um jogo há atividades que o jogador só pode realizar se estiver em uma determinada área do mundo virtual, ao mesmo tempo em que algumas áreas só podem ser

acessadas após a realização de alguma tarefa específica. Dessa forma, o Modelo Global Temporizado presente em (BARRETO; FREITAS; JULIA, 2018) é apresentado por meio da relação existente entre o Modelo Lógico Temporizado e o Modelo Topológico Temporizado. Para especificar a comunicação entre os dois modelos é implementado em ambos um mecanismo de comunicação síncrona baseado em *fusion places*.

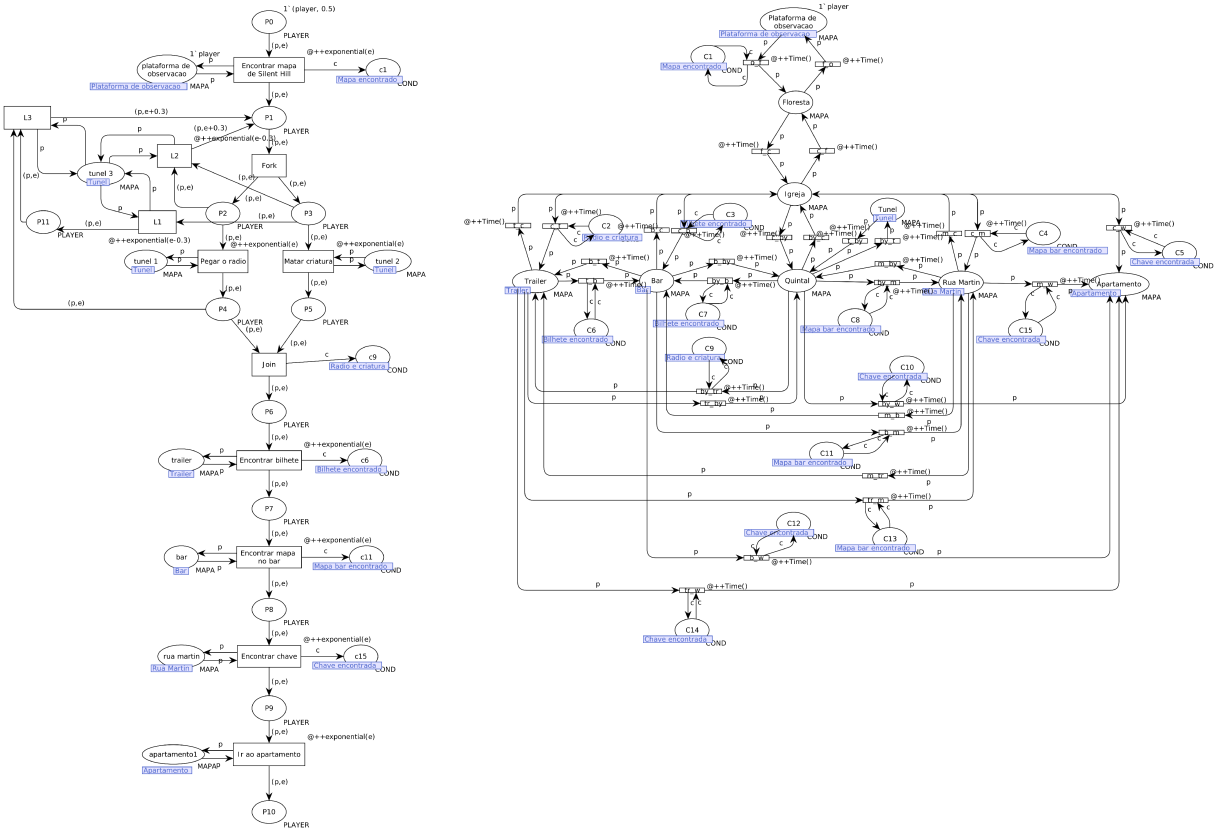


Figura 30 – Modelo Global Temporizado para o CPN Tools. Figura retirada de (BARRETO, 2020).

A Figura 30 apresenta o Modelo Global Temporizado para o CPN Tools, no qual podemos perceber que apesar dos modelos lógico e topológico serem graficamente independentes o uso do conceito de *fusion places* garante que os dois modelos trabalhem juntos. Também pode se identificar no modelo lugares representando requisitos que devem ser cumpridos para a liberação de áreas no mapa, associados ao *color set COND* e rotulados com *fusion tags* nomeadas de acordo com as condições (atividades) existentes. Cada uma dessas condições possuem um conjunto de fusão que é composto pelos lugares marcados pela mesma *fusion tag*. Os lugares pertencentes ao tipo de dado *MAPA* presentes no Modelo Lógico asseguram que atividades dependentes do jogador estar em uma determinada área no mundo virtual só poderão ser realizadas caso ele esteja na devida área no Modelo Topológico.

Após feita a migração da apresentada rede de Petri Colorida Temporizada para o *software* SIMIO, a Figura 31 traz o Modelo Global Temporizado 2D para o SIMIO. Note

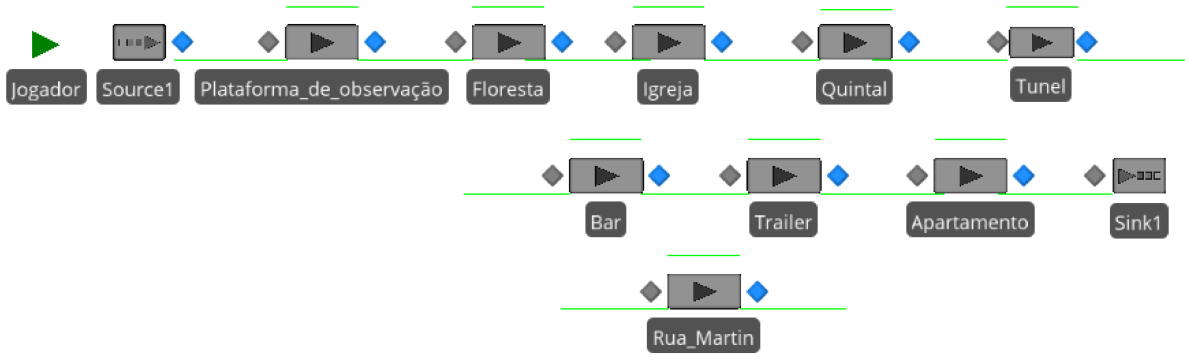


Figura 31 – Modelo Global Temporizado 2D no SIMIO.

que visualmente ele é idêntico ao Modelo Topológico Temporizado 2D para o SIMIO, apresentado na Figura 25, uma vez que a lógica da CP-net é retratada e incorporada no SIMIO em objetos e associada aos seus eventos não é exposta externamente. A Figura 32 traz o Modelo Global Temporizado 3D para o SIMIO, no qual a única diferença visualmente perceptível entre ele e o Modelo Topológico Temporizado 3D para o SIMIO, apresentado na Figura 26, é na área do túnel: no modelo topológico uma criatura é colocada nela para representar o monstro da atividade *Matar criatura*, enquanto no modelo global o túnel se apresenta inicialmente vazio. Somente durante a execução do modelo o túnel não ficará vazio, conforme o jogador passar por ele e executar as atividades da área o símbolo da entidade "Jogador" é alterada para representá-las.

Figura 32 – Modelo Global Temporizado 3D no SIMIO.1

As listas de nós de destino continuam sendo utilizadas para realizar o roteamento através do método *Select From List*, aplicado nos nós de saída dos objetos e para traçar as possíveis rotas disponíveis a partir de cada área. Como inicialmente há áreas bloqueadas para o jogador até que ele cumpra certas atividades, ao invés de se associar uma lista fixa em cada nó de saída de *Servers* que são interligados a outros *Servers*, um atributo do tipo *List Reference* é utilizado em seu lugar. Essa variável é atualizada sendo associada

a diferentes listas, que contém os lugares alcançáveis e já desbloqueados por ações do jogador no protótipo, ao decorrer da execução do modelo. Cada *Server* possui uma *List Reference* exclusiva, bem como um conjunto de listas de nós com as possíveis combinações de destinos para ele. Sempre que uma atividade é realizada, todas as *List Reference* de lugares afetados por desbloquearem uma nova passagem são também alteradas.

Nas áreas nas quais há atividades a serem realizadas, foram adicionados aos *Servers* que as representam os procedimentos necessários, tanto para representar a lógica associada à execução da tarefa, quanto para a possibilidade do jogador somente atravessar o lugar. Dessa forma, caso a entidade "Jogador" entre em uma área que possua uma atividade ainda não cumprida, uma escolha aleatória designada à variável de condição dessa atividade na propriedade *State Assignments*, localizada no submenu de propriedades do *Server*, irá selecionar se a entidade irá apenas passar pela área e seguir para a próxima (e, assim, apenas o tempo de deslocamento será calculado com o processo "Process_Tempo_Topologico") ou se ela irá permanecer na área e realizar a atividade, com o processo "Process_Tempo_Logico" retornando o tempo gasto para executá-la. Ocorre nesta última opção também a alteração da condição associada à atividade, que passa a ser assinalada como cumprida, consequente a atualização dos *List Reference* de lugares que desbloqueiam caminhos para novas áreas acessíveis.

Em particular, há dois casos nos quais os servidores não foram programados seguindo o raciocínio apresentado acima: (1) no *Server* "Apartamento", uma vez que a entidade entre no servidor só há a opção de se realizar a atividade *Ir ao apartamento* e encerrar a execução do modelo; (2) para melhor seguir e adaptar a lógica da CP-net, o *Server* "Tunel" também foi configurado de forma diferente das demais áreas.

Na Figura 33 é exibido os processos implementados na área de processos do modelo, note que o processo "SortearCaso" também existe no Modelo Lógico Temporizado para o SIMIO, apresentado na seção 4.1. A versão presente no Modelo Global Temporizado do SIMIO do processo "SortearCaso" segue, inicialmente, a mesma configuração do apresentado anteriormente: utiliza-se tabelas para listar as opções disponíveis para o jogador executar (cada uma representando um possível disparo de transição da CP-net equivalente) e o SIMIO escolhe aleatoriamente uma delas. Mas como neste modelo todo o processo de atividades da área *Tunel* é realizado no mesmo objeto servidor, foi incorporado ao "SortearCaso" a lógica para lidar com todas as possíveis sequências de eventos. Dentro do processo, *states variables* da área de definição foram utilizadas para se criar atributos de controle que são checados nos momentos necessários dentro dos *steps Decide* e até mesmo antes de declarações nos *steps Assign*. Os *steps Execute* são responsáveis por iniciarem a execução de outros processos, uma vez que a realização de cada atividade possível dentro do *Tunel* recebe um processo a parte. Em especial, o processo "Join" não representa uma atividade do jogador e sim a transição de mesmo nome na CP-net, onde

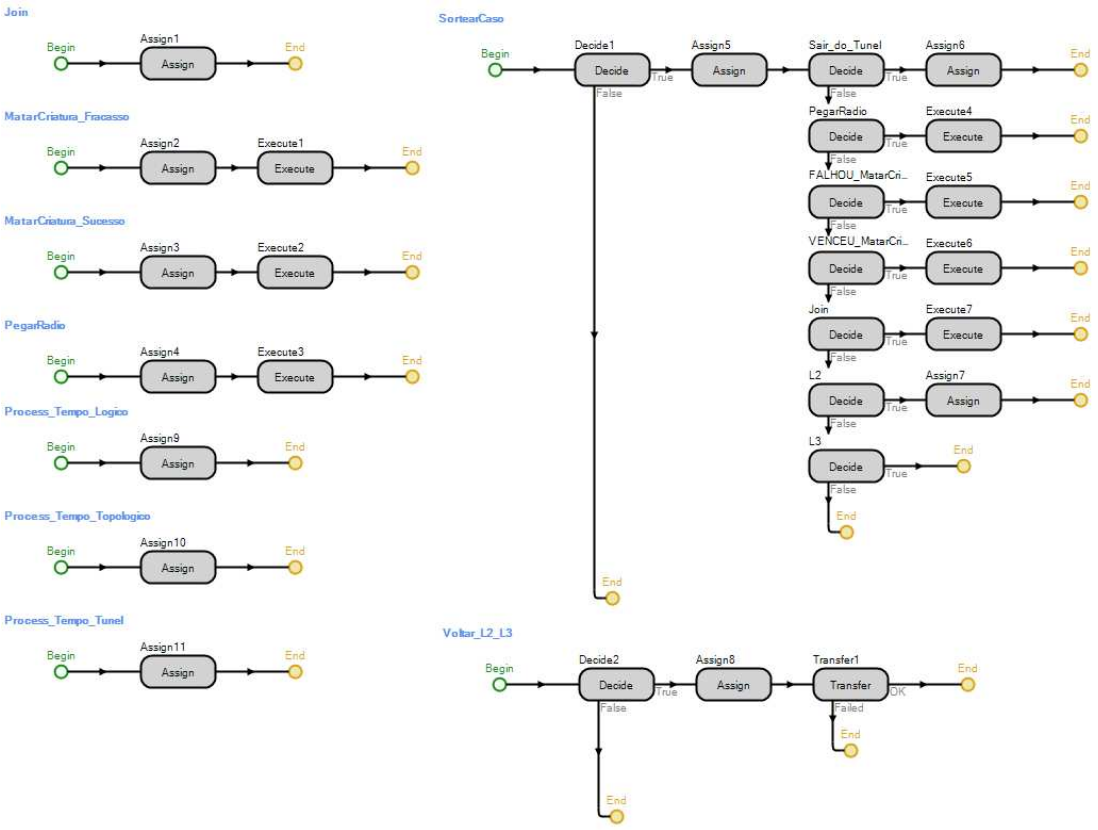


Figura 33 – Conteúdo da área de processo do Modelo Global Temporizado no SIMIO.

a condição *Radio e criatura* é dada como cumprida.

Os *steps Execute* que encerram os processos "MatarCriatura_Fracasso", "MatarCriatura_Sucesso" e "PegarRadio" tornam a chamar o processo "SortearCaso", para que um novo sorteio sobre o que será executado em seguida ocorra. É importante ressaltar que a cada nova chamada de "SortearCaso" as possíveis ações que entram no sorteio realizado pelo SIMIO mudam, da mesma forma como o disparo de uma transição na CP-net pode alterar as transições sensibilizadas. Neste momento, os atributos de controle e os recursos condicionais do SIMIO são empregados, dessa forma o processo "Join" só será executado caso as atividades *Matar criatura* e *Pegar radio* já tenham sido realizadas com sucesso. Deste mesmo modo, o processo "Voltar_L2_L3", chamado no nó de saída do servidor "Tunel" é responsável por direcionar a entidade "Jogador" para o nó de entrada do servidor em questão, só é executado caso o jogador falhe ao tentar realizar a atividade *Matar criatura* e esteja pronto para reiniciar o roteiro iterativo de atividades da área *Tunel*.

O processo "SortearCaso" é chamado em "Tunel" na ocorrência do evento *Entered*, correspondente a uma entidade entrar no servidor. Após esse processo encerrar sua execução, o "Process_Tempo_Tunel" é chamado no evento seguinte, *Before Processing*. Assim, quando o servidor entrar em sua fase de processamento o tempo de *delay* correspondente a ela será dado pela variável "Var_Delay_Tunel", estabelecida previamente em "Process_Tempo_Tunel", que será a duração total do tempo gasto com a(s) atividade(s)

realizada(s) pelo jogador em "SortearCaso", o que garante que, ao final da simulação, o tempo global da execução do modelo (tempo gasto para realizar atividades somado ao tempo consumido para se locomover entre as áreas) seja igual ao tempo que a entidade permaneceu no sistema.

5 Simulação

Na fase de pré-produção de jogos, busca-se garantir que o jogo seja atrativo e não apresente erros de jogabilidade. A abordagem presente em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) apesar de validar os modelos apresentados através de uma análise aplicada ao conceito de jogabilidade em *video games*, pouco pode garantir a respeito de seu apelo atrativo para os jogadores com os modelos baseados na teoria das redes de Petri exibidos na ferramenta CPN Tools.

Entretanto através da interface gráfica 3D do SIMIO, os modelos de cenários de jogos construídos neste ambiente ganham um novo ângulo sobre o qual a análise qualitativa pode ser realizada: durante uma execução interativa do modelo 3D é possível acompanhar a evolução visual do cenário conforme o jogador avança pelo protótipo do nível. Com essa representação gráfica passo a passo torna-se mais simples não somente a identificação de falhas e problemas nos protótipos de jogos, como também a percepção do quão atraente o jogo pode ser para futuros jogadores. Por meio dessa avaliação ainda nas etapas de *game design* e *level design*, verifica-se o bom funcionamento do jogo em termos de jogabilidade tanto ao garantir que o curso do jogo está correto, quanto ao conferir a relevância do cenário proposto ao analisar se ele é interessante.

A execução de tal simulação interativa no SIMIO ocorre conforme a entidade percorre os objetos e estações do modelo. Ao entrar em um objeto, os processos invocados nele e que regem o seu comportamento podem modificar não somente dados relacionados à lógica do modelo, como também a sua animação ao alterar o símbolo associado àquele objeto e/ou o símbolo associado à entidade em processamento.

Outro fator que contribui na análise de jogabilidade é o tempo médio de duração do jogo, uma vez que as atividades propostas no cenário devem ser interessantes para prender a atenção do jogador, mas não podem ser complexas demais a ponto de prendê-lo por muito tempo em determinada parte do jogo e, assim, comprometendo o seu entretenimento. Isto posto, o objetivo da análise quantitativa está relacionado com os tempos de execução dos modelos para o caso apresentado. Os tempos da execução do modelo global podem ser usados para estimar as durações mínima, máxima e média de um nível de jogo. Os modelos lógico e topológico podem ser simulados separadamente, gerando os tempos mínimo, máximo e médio estimados para executar todas as tarefas do nível e para percorrer todas as áreas do mapa.

Para garantir a consistência das atividades do nível de jogo e do mapa topológico construído, em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) é criado um modelo de análise através do Modelo Global estendido do jogo $\overline{\text{MG}}$, baseado no método

de verificação da propriedade *soundness*. Com as ferramentas de análise disponíveis no CPN Tools essa avaliação é feita através do cálculo do espaço de estado (*state space*) do modelo.

Com a corretude do modelo assegurada, para obter os tempos da simulação dos modelos da abordagem para a modelagem de cenários de *video games* usando o formalismo das redes de Petri Coloridas com adição de tipos temporizados, é utilizada uma outra funcionalidade de análise da ferramenta CPN Tools: a simulação automática por replicação. O relatório gerado por esta ferramenta fornece, além de outros dados, os tempos de execução de cada replicação do modelo em simulação. Todos os modelos são simulados automaticamente com 10 replicações e com 100 replicações.

Com a migração dos modelos de jogo baseados em redes de Petri Coloridas Temporizadas para os modelos do SIMIO apresentados anteriormente, a realização da simulação automática por replicações neste último *software* é executada através da adição de um *Experiment* ao modelo do projeto SIMIO.

As execuções feitas através de *Experiments* são realizadas automaticamente, sem animação e através de um conjunto de cenários, com cada cenário possuindo um número de replicações associado. Ao final da simulação, é fornecido pelo SIMIO uma série de informações que podem ser vistas e analisadas em diferentes configurações como, por exemplo, por meio de gráficos, tabelas, relatórios com detalhes das replicações ou relatórios com comparações entre cenários.

Para realizar a simulação dos modelos no SIMIO são criados um *Experiment* de dois cenários em cada modelo, um cenário com 10 replicações e um outro com 100 replicações. Uma *Response* é adicionada a cada *Experiment* para registrar no final de cada replicação de cada cenário o valor da variável "Tempo", responsável por registrar o tempo total gasto pelo jogador no modelo, bem como fornecer os valores mínimo, médio e máximo dessa variável ao final da execução do *Experiment*.

As seções a seguir apresentam os resultados das simulações presentes em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) e realizadas no CPN Tools para os modelos da abordagem apresentada, das simulações dos modelos convertidos para o SIMIO e uma comparação entre as execuções em cada ferramenta.

5.1 Simulação: Modelo Lógico Temporizado

Como já apresentado, no SIMIO um modelo pode ser executado através de um *Experiment* ou interativamente. O Modelo Lógico Temporizado 3D para o SIMIO da Figura 22 apresenta a maquete em que as atividades irão ocorrer, porém somente durante a sua simulação interativa é possível visualmente identificar as diferentes animações que

representam cada uma dessas atividades.

Os disparos das transições ao decorrer da execução do Modelo Lógico Temporizado no CPN Tools apresentado na Figura 20, apesar de também representarem a realização das atividades que devem ser realizadas pelo jogador, requerem um entendimento das abordagens formais usadas em Engenharia de Software. No SIMIO, a execução da versão 3D do modelo equivalente pode ser visualizada e avaliada até mesmo pelos não especialistas dessas abordagens.

(a) (b)

Figura 34 – Execução da atividade *Encontrar chave* no Modelo Lógico 3D no SIMIO.

Para ilustrar, considere a transição *Encontrar chave* da CP-net da Figura 20. Os *screenshots* da Figura 34 representam a execução dessa transição no modelo 3D do SIMIO, trazendo na Figura 34a o momento antes da atividade em questão ser realizada, no qual o jogador apenas está percorrendo o ambiente e na Figura 34b o momento em que ele para e encontra a chave.

Figura 35 – Execução da atividade *Matar criatura* no Modelo Lógico 3D no SIMIO.

Durante a simulação da versão 3D para simbolizar o fracasso na atividade *Matar criatura* o jogador percorre o caminho que retorna para a entrada da bifurcação na maquete da Figura 22, enquanto para representar o sucesso e a transição *Matar criatura* da CP-net da Figura 20 são associados ao objeto servidor “Matar_criatura” e a entidade

"Jogador" os símbolos que compõe o cenário do *screenshot* da Figura 35. A atividade *Pegar radio* é representada pelo símbolo mostrado no *screenshot* da Figura 36.

Figura 36 – Execução da atividade *Pegar radio* no Modelo Lógico 3D no SIMIO.

Já a realização de um *Experiment* traz os resultados de uma simulação automática por replicação, entre eles a variável que monitora o tempo consumido pelo jogador durante a execução do modelo. A Tabela 1 apresenta as unidades de tempo mínima, máxima e média gastas para executar as tarefas do nível simulado no Modelo Lógico Temporizado executado no CPN Tools em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) e do modelo equivalente executado no SIMIO. De forma geral é possível observar que os tempos obtidos em ambas as ferramentas são próximos, conservando uma média semelhante para a execução do modelo lógico. Pequenas diferenças são esperadas devido ao uso de números pseudoaleatórios para representar cada intervalo de tempo gasto pelo jogador.

Tabela 1 – Comparação dos Resultados da Simulação do Modelo Lógico Temporizado.

	10 Replicações		100 Replicações	
	CPN Tools	SIMIO	CPN Tools	SIMIO
Tempo máximo	23,41	18,53	32,14	27,77
Tempo mínimo	8,54	9,01	3,84	5
Tempo médio	13,7	12,65	13,55	13,98

5.2 Simulação: Modelo Topológico Temporizado

No Modelo Topológico Temporizado para o CPN Tools que pode ser visto na Figura 24, o deslocamento do jogador pelo mapa virtual do cenário se dá pela passagem da ficha que o representa pelas transições e lugares, os quais recebem nomes de acordo com a área retratada. Em uma simulação interativa desta abordagem, pouco apelo visual e sentido o esquema de transições e entrada e saída da ficha dos lugares da CP-net traz para quem não está familiarizado com as redes de Petri e com as abordagens utilizadas em modelagem e simulação.

Já em uma ferramenta que conta com o auxílio de um ambiente gráfico 3D, onde é possível transformar o Modelo Topológico Temporizado apresentado na versão do SIMIO

exibida na Figura 26, o jogador pode adquirir a aparência de um homem passeando por uma maquete na qual os ambientes retratam a área que desejam representar. Dessa forma, a passagem entre as áreas do mapa é facilmente compreendida por qualquer pessoa durante a execução do modelo.

Para demonstrar essa vantagem, a Figura 37 apresenta um *screenshot* da simulação do Modelo Topológico Temporizado 3D para o SIMIO, no qual o jogador está saindo da área da *Igreja* e retornando para a *Floresta*.

Figura 37 – Jogador percorrendo o mapa topológico do modelo 3D no SIMIO.

Uma vez que no modelo topológico é permitido ao jogador circular livremente entre as áreas quantas vezes quiser, a única constância na execução deste modelo é o lugar inicial ser a *Plataforma de observação* e ele encerrar quando o jogador chegar ao lugar final, o *Apartamento*.

Ao submeter o Modelo Topológico Temporizado a simulação automática com 10 replicações e 100 replicações, a Tabela 2 traz os resultados obtidos com a execução da simulação no CPN Tools realizada em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) e com a simulação do modelo traduzido no SIMIO. Observa-se que com mesmo toda a liberdade dada ao jogador para passear pelo mapa, o tempo médio estimado para percorrer todas as suas áreas mantém-se bem próximo em ambas as ferramentas. À medida que o número de replicações aumenta, os dados apresentados mostram que os tempos mínimos decaem tanto no CPN Tools, quanto no SIMIO, enquanto os tempos máximos crescem.

Tabela 2 – Comparação dos Resultados da Simulação do Modelo Topológico Temporizado.

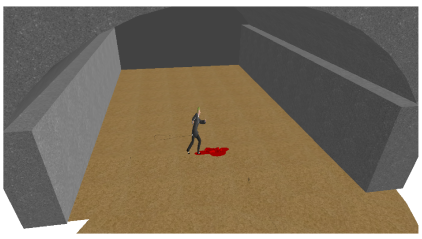
	10 Replicações		100 Replicações	
	CPN Tools	SIMIO	CPN Tools	SIMIO
Tempo máximo	10,5	11,45	14,9	24,08
Tempo mínimo	1,17	1,57	0,57	1,09
Tempo médio	5	4,02	4,96	5,32

5.3 Simulação: Modelo Global Temporizado

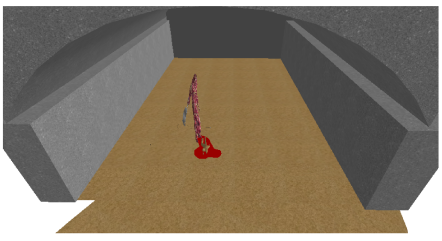
O Modelo Global Temporizado para o CPN Tools apresentado na Figura 30 é composto pelos modelos lógico e topológico com os *fusion places* incorporados a eles servindo como mecanismo de comunicação entre os dois. A execução deste modelo é necessária para estimar uma duração realista do jogo, uma vez que ele considera tanto o mapa virtual e as atividades que devem ser realizadas pelo jogador, como as restrições existentes para executar as atividades e para acessar os lugares.

Na ferramenta CPN Tools, a execução visual deste modelo dificilmente seria compreendida caso o protótipo fosse apresentado para pessoas sem um conhecimento prévio do formalismo das redes de Petri. Porém no SIMIO, com a maquete 3D construída para representar o mundo virtual do nível de jogo configurada para também executar as atividades do modelo lógico e lidar com as restrições mencionadas, a evolução da simulação do modelo é fácil de acompanhar.

(a) Pegar rádio.



(b) Sucesso em Matar Criatura.



(c) Fracasso em Matar Criatura.

Figura 38 – Execução das atividades na área *Tunel* no Modelo Global 3D no SIMIO.

No Modelo Global Temporizado 3D para o SIMIO exibido na Figura 32, além de acompanhar o jogador percorrer o mapa, o espectador da simulação pode também ver as atividades serem realizadas e novas áreas serem desbloqueadas para acesso. Dando

especial destaque para a área do *Tunel*, o símbolo da entidade "Jogador" é alterado a cada novo ingresso no local para representar qual atividade ele executará ou permanece o mesmo, caso ele esteja apenas passando pela área.

Na Figura 38 é apresentado *screenshots* de diferentes momentos da área *Tunel* durante a simulação do Modelo Global 3D. A Figura 38a traz a execução da atividade *Pegar radio*, a Figura 38b o sucesso na atividade *Matar Criatura* e a Figura 38c exibe o cenário que é apresentado caso o jogador fracasse em *Matar Criatura*.

A Tabela 3 apresenta os resultados da simulação do Modelo Global Temporizado no CPN Tools realizada em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020) e da simulação realizada no SIMIO com o modelo convertido equivalente. Percebe-se que a duração média do primeiro nível de Silent Hill II tem valores próximos em ambas as ferramentas, enquanto os tempos mínimos e máximos só se aproximam conforme o número de replicações aumenta. Com 100 replicações os tempos mínimos caem e os tempos máximos aumentam tanto no CPN Tools, quanto no SIMIO.

Tabela 3 – Comparação dos Resultados da Simulação do Modelo Global Temporizado.

	10 Replicações		100 Replicações	
	CPN Tools	SIMIO	CPN Tools	SIMIO
Tempo máximo	210,44	160,26	231,68	219,76
Tempo mínimo	59,13	28,35	29,8	20,97
Tempo médio	87,16	89,78	86,68	77,94

6 Conclusão

Neste trabalho foi apresentada uma abordagem para integrar protótipos de cenários de *video games* modelados com o formalismo das redes de Petri Coloridas com a interface gráfica 3D do *software* de simulação SIMIO. Para tal são selecionados cenários de jogos construídos em redes de Petri Coloridas com adição de tipos temporizados presentes em (BARRETO; FREITAS; JULIA, 2018; BARRETO, 2020), onde o primeiro nível do jogo Silent Hill II é utilizado para ilustrar o proposto.

Um nível de jogo é formado por um conjunto de atividades, as quais devem ser realizadas dentro de um espaço virtual, onde cada ação e movimentação do jogador possui uma duração não nula associada. Isto posto, foi construído um Modelo Lógico Temporizado para retratar a sequência de atividades e um Modelo Topológico Temporizado para representar as áreas do mapa virtual. Por meio de um mecanismo de comunicação desenvolvido para estabelecer a influência que um modelo tem sobre o outro é gerado o Modelo Global Temporizado. Na ferramenta CPN Tools os três modelos são construídos através do formalismo das redes de Petri.

Com o auxílio das abordagens de análise automática fornecidas pelo CPN Tools, a análise do espaço de estados (*state space*) é utilizada para realizar a análise qualitativa do protótipo e examinar a consistência do cenário implementado em termos de jogabilidade através da verificação da propriedade *soundness*. A outra abordagem de análise disponível é a de simulação, a qual pode ser realizada de forma interativa ou automática. Na simulação interativa é possível acompanhar a evolução do cenário e investigar diferentes detalhes do nível, o que requer um conhecimento da teoria das redes de Petri. Já a simulação automática pode ser realizada por replicações, onde um relatório com os dados de cada replicação é gerado ao final. A simulação automática por replicações é realizada com 10 replicações e 100 replicações sendo empregada como um método para a análise quantitativa, a fim de estabelecer os tempos mínimos, máximos e médios para realizar todas as atividades do nível, para o jogador acessar as diversas áreas do mapa virtual e para o jogador concluir o cenário.

Entretanto, apesar das redes de Petri Coloridas serem consideradas um recurso gráfico e matemático de representação formal eficiente para modelagem e análise de sistemas de jogos, o seu formalismo é muito abstrato para ser apresentado a quem não é especialista das abordagens usadas em Engenharia de Software.

A forma de integração das redes de Petri com ambientes de modelagem e simulação têm sido recentemente estudada pela comunidade científica, o que foi apresentado neste trabalho se baseia na abordagem apresentada em (MOTA et al., 2017) para migração da

lógica dos modelos em rede de Petri para o ambiente de modelagem 2D ou 3D multi-paradigma do SIMIO, o qual combina orientação a objetos e orientação a processos. Embora não haja uma equivalência direta entre as redes de Petri comuns ou Coloridas com o SIMIO, a implementação dos elementos e características básicas das redes de Petri nos componentes do modelo SIMIO é efetuada de forma que o comportamento dinâmico originalmente estabelecido no modelo em rede de Petri seja mantido no protótipo no SIMIO.

O SIMIO é um *software* de modelagem e simulação que têm sido adotado em diferentes áreas da indústria, sendo utilizado para resolver problemas logísticos como, por exemplo, pesquisa operacional ou cadeias de abastecimento. Com a migração dos modelos de cenários de jogos em rede de Petri Colorida para os protótipos equivalentes construídos no ambiente de simulação do SIMIO, a representação visual 3D desta ferramenta se mostrou uma plataforma intuitiva de prototipação.

Um modelo pode ser executado interativamente ou através de um *Experiment* no SIMIO. Por meio de uma execução interativa, a interface gráfica 3D do *software* permite acompanhar passo a passo a evolução visual do cenário, conforme o jogador avança pelo protótipo do nível de jogo. Com símbolos, formas geográficas, texturas e animações sendo utilizadas para criar a maquete que retrata o cenário de nível de jogo, para representar o jogador, cada ação realizada por ele e a sua passagem pelo modelo, a análise qualitativa do cenário em termos de jogabilidade torna-se muito mais simples. Além da identificação de falhas e problemas nos protótipos ainda na fase de pré-produção é mais fácil se certificar dos atrativos que o jogo deve ter para entreter os jogadores. Com tudo isso sendo realizado em um ambiente gráfico 3D, não se faz necessário nenhum conhecimento prévio do formalismo das redes de Petri, das notações formais presentes nas linguagens de programação ou demais abordagens formais.

Dessa maneira, a vantagem do que foi proposto neste trabalho é permitir que tanto projetistas, quanto os não especialistas em abordagens formais possam acompanhar e avaliar protótipos de jogos através de uma simulação gráfica interativa, auxiliando dessa forma na fase de pré-produção de jogos (*game design* e *level design*).

Com o uso de *Experiments*, o projeto no SIMIO pode ser submetido a simulação automática por replicações. Para realizar a análise quantitativa dos protótipos de cenários de jogos construídos no SIMIO, os três modelos são submetidos a simulação automática com 10 replicações e com 100 replicações. O relatório final gerado pelo *software* traz entre diversas informações os dados referentes à variável "Tempo", presente em todos os modelos e responsável por armazenar o tempo gasto pela entidade "Jogador" em cada replicação. Desta forma os tempos mínimos, máximos e médios estimados para executar as atividades do nível, para o jogador percorrer as áreas do mapa virtual e para ele concluir o modelo global nos protótipos do SIMIO são registrados.

Os tempos obtidos com as simulações dos modelos em redes de Petri Coloridas no CPN Tools e com as simulações dos modelos equivalentes no SIMIO são comparados e se mostram nos três casos bem próximos, até mesmo conforme o número de replicações aumenta. Certas diferenças são esperadas devido ao uso de funções de distribuição de tempo aleatório e números pseudoaleatórios para representar os intervalos de tempo consumidos pelo jogador. Em especial, os tempos médios dos três modelos apresentarem valores próximos em ambas as ferramentas é um indício da equivalência entre os protótipos.

A abordagem apresentada neste trabalho trata apenas de cenários de jogos com um jogador (*singleplayer*). Um trabalho futuro seria estender o proposto para cenários de jogos com mais de um jogador presente (*multiplayer*). Poderia ser interessante acompanhar através de uma simulação 3D no ambiente visual do SIMIO a orientação a objetos e a processos do *software* lidar com as influências diretas ou indiretas que a ação de um jogador tem sobre os demais jogadores e sobre o ambiente virtual do jogo.

Referências

AALST, W. van der. Timed coloured petri nets and their application to logistics. In: *Internarional Symposium on Physical Design*. [S.l.: s.n.], 1992. Citado na página 22.

AALST, W. van der. Structural characterizations of sound workflow nets. *Computing Science Reports*, v. 96, n. 23, p. 18–22, 1996. Citado na página 20.

AALST, W. van der. Verification of workflow nets. In: *Application and Theory of Petri Nets1997*. [S.l.]: Springer, 1997. p. 407–426. Citado 2 vezes nas páginas 19 e 20.

AALST, W. 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. Citado 3 vezes nas páginas 18, 19 e 20.

AALST, W. van der; HEE, K. M. van. *Workflow management: models, methods, andsystems*. [S.l.]: MIT press, 2004. Citado 2 vezes nas páginas 18 e 19.

AALST, W. van der; HOFSTEDE, A. H. ter. Verification of workflow task structures:a petri-net-baset approach. *Information Systems*, v. 25, n. 1, p. 43–69, 2000. ISSN 0306-4379. Citado na página 19.

AALST, W. van der; STAHL, C. *Modeling Business Processes: A Petri Net-Oriented Approach*. Cambridge, MA, England: MIT press, 2011. Citado 3 vezes nas páginas 21, 22 e 41.

ANG, C. S.; RAO, G. S. V. R. K. Designing interactivity in computer games: a uml approach. *International Journal of Intelligent Games and Simulation*, v. 3, n. 2, p. 62–69, 2004. Citado na página 37.

ARAÚJO, M.; ROQUE, L. Modeling games with petri nets. In: *DiGRA Conference*. [S.l.: s.n.], 2009. Citado na página 37.

BARRETO, F. M. *Modelagem e análise de vídeo games baseadas em WorkFlow net e Grafos de estado*. Dissertação (Mestrado) — Universidade Federal de Uberlândia, Uberlândia, 2015. Citado 3 vezes nas páginas 16, 17 e 20.

BARRETO, F. M. *Uma abordagem baseada em Redes de Petri para Modelagem, Análise e Simulação de Cenários de Vídeo Games Singleplayer e Multiplayer*. Tese (Doutorado) — Universidade Federal de Uberlândia, Uberlândia, 2020. Citado 14 vezes nas páginas 12, 13, 39, 40, 41, 42, 47, 51, 56, 57, 59, 60, 62 e 63.

BARRETO, F. M.; FREITAS, J. C. J. de; JULIA, S. A timed petri net model to specify scenarios of video games. In: *Information Technology - New Generations*. Las Vegas: Springer International Publishing AG, 2018. (Advances in Intelligent Systems and Computing, v. 738), p. 467–473. Citado 15 vezes nas páginas 12, 13, 39, 40, 41, 42, 45, 46, 51, 56, 57, 59, 60, 62 e 63.

BARRETO, F. M.; JULIA, S. Modeling of video games using workflow nets and state graphs. In: *Proceedings of the 31st Brazilian Symposium on Software Engineering -*

SBES'17. Fortaleza: [s.n.], 2017. p. 261–266. Citado 5 vezes nas páginas 12, 13, 38, 39 e 40.

CALLELE, D.; NEUFELD, E.; SCHNEIDER, K. Requirements engineering and the creative process in the video game industry. In: *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*. [S.l.]: IEEE, 2005. p. 240–250. Citado na página 11.

CARDOSO, J.; VALETTE, R. *Redes de Petri*. Florianópolis: Editora da UFSC, 1997. Citado 3 vezes nas páginas 15, 17 e 41.

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, ACE 05*. New York, NY, USA: ACM, 2005. Citado na página 11.

DAVID, R.; ALLA, H. Petri nets for modeling of dynamic systems: A survey. *Automatica*, v. 30, n. 2, p. 175–202, 1994. Citado na página 20.

DAVID, R.; ALLA, H. *Discrete, Continuous, and Hybrid Petri Nets*. France: Springer, 2010. Citado 3 vezes nas páginas 15, 17 e 20.

GAL, V. et al. Writing for videogames. *Proceedings Laval Virtual (IVRC)*, 2002. Citado na página 11.

HORROCKS, I. *Constructing the User Interface with Statecharts*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999. Citado na página 12.

JENSEN, K.; KRISTENSEN, L. M. *Coloured Petri nets: modelling and validation of concurrent systems*. [S.l.]: Springer Science & Business Media, 2009. Citado 6 vezes nas páginas 13, 21, 22, 25, 26 e 41.

KANODE, C. M.; HADDAD, H. M. Software engineering challenges in game development. In: LATIFI, S. (Ed.). *ITNG*. [S.l.]: IEEE Computer Society, 2009. p. 260–265. Citado 2 vezes nas páginas 11 e 13.

KRISTENSEN, L. M.; CHRISTENSEN, S.; JENSEN, K. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer (STTT)*, v. 2, n. 2, p. 98–132, 1998. Citado 2 vezes nas páginas 22 e 23.

LEE, Y.-S.; CHO, S.-B. Dynamic quest plot generation using petri net planning. In: *Proceedings of the Workshop at SIGGRAPH Asia*. [S.l.]: ACM, 2012. p. 47–52. Citado na página 38.

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, GAS '11*. New York, NY,USA: ACM, 2011. p. 1–4. Citado 3 vezes nas páginas 11, 12 e 38.

MILNER, R. et al. *The definition of standard ML: revised*. Cambridge, MA, USA: MIT press, 1997. Citado na página 21.

MOTA, I. F. D. L. et al. *Robust Modelling and Simulation - Integration of SIMIO with Coloured Petri Nets*. Cham, Switzerland: Springer, 2017. Citado 9 vezes nas páginas 12, 14, 26, 29, 30, 31, 40, 41 e 63.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, 1989. Citado 4 vezes nas páginas 12, 15, 16 e 17.

NATKIN, S.; VEGA, L.; GRÜNVOGEL, S. M. A new methodology for spatiotemporal game design. In: *Mehdi, Q. and Gough, N.,(Eds.). Proceedings of CGAIDE2004, 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education*. Wolverhampton, UK: The University of Wolverhampton, School of Computing and Information Technology, 2004. p. 109–113. Citado na página 37.

OLIVEIRA, G. W. de. *Modelagem e análise de video games usando as workflow nets e a lógica linear*. Dissertação (Mestrado) — Faculdade de Computação UFU, 2012. Citado 2 vezes nas páginas 37 e 38.

OLIVEIRA, G. W. de; JULIA, S.; PASSOS, L. M. S. Game modeling using workflownets. In: *2011 IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.: s.n.], 2011. p. 838–843. Citado 2 vezes nas páginas 37 e 38.

PETRI, C. A. *Kommunikation mit Automaten*. Tese (Doutorado) — Universität Hamburg, Germany, 1962. Citado na página 15.

REYNO, E. M.; CUBEL, J. Á. C. Automatic prototyping in model-driven game development. *Computers in Entertainment*, v. 7, n. 2, 2009. Citado na página 11.

RUCKER, R. v B. *Software engineering and computer games*. [S.l.]: Pearson Education, 2003. Citado na página 37.