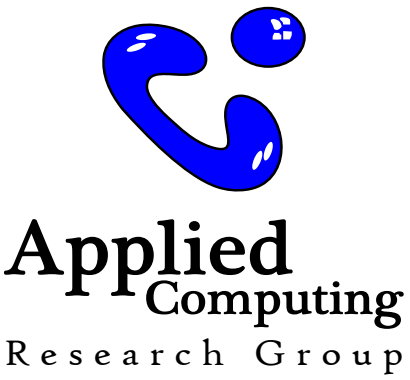
Modelagem Computacional e Simulação do Comportamento de uma Solução de Integração no Contexto Acadêmico da Unijuí Utilizando Redes de Petri Coloridas e Temporizadas

Adriana Rosélia Kraisig

Universidade Regional do Noroeste do Estado do Rio Grande do Sul

Dissertação de Mestrado

Orientador:
Dr. Rafael Zancan Frantz
Coorientador:
Dr. Sandro Sawicki



Março, 2017

First published in March 2017 by
Applied Computing Research Group - GCA
Department of Exact Sciences and Engineering
Rua Lulu Ilgenfritz, 480 - São Geraldo
Ijuí, 98700-000, Brazil.

Financiamento: Bolsa de mestrado concedida pela PROSUP/CAPES.

Universidade Regional do Noroeste do Estado do Rio Grande do Sul

A Comissão Examinadora, abaixo assinada, _____ a dissertação intitulada "Modelagem Computacional e Simulação do Comportamento de uma Solução de Integração no Contexto Acadêmico da Unijuí Utilizando Redes de Petri Coloridas e Temporizadas: ", elaborada por Adriana Rosélia Kraisig, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática.

_____

Dr. Rafael Zancan Frantz
UNIJUÍ
(Orientador)

_____

Dr. Sandro Sawicki
UNIJUÍ
(Co-orientador)

_____

Dra. Inmaculada Hernández Salmerón
Universidade de Sevilha
Espanha

_____

Dr. Paulo Sérgio Sausen
UNIJUÍ

_____

Dra. Fabricia Carneiro Roos Frantz
UNIJUÍ

Ijuí, _____ de _____ de _____.

Integração de Aplicações por Diovana, 12 anos de idade.

Dedico este trabalho à minha família.

Conteúdo

Índice de figuras

Índice de tabelas

Agradecimentos

Concentre-se naquilo que você é bom,
delegue todo o resto.

Steve Jobs

A Deus, que todos os dias de minha vida me deu forças para nunca desistir. Deus é o dono de tudo. Devo a ele a oportunidade que tive de chegar aonde cheguei. Muitas pessoas têm essa capacidade, mas não têm essa oportunidade. Ele a deu para mim, não sei por quê. Sei que não posso desperdiçá-la, por (Ayrton Senna).

A minha família, pai Auri A. Kraisig, mãe Rosane M. G Kraisig, avô Anildo J. Kraisig, avó Olivia Q. Kraisig, irmã Ângela R. Kraisig e ao meu namorado Dênis J. V. Serafini, minha gratidão pelo apoio e incentivo incondicional em todas as etapas da pesquisa.

Ao meu orientador, professor Dr. Rafael Zancan Frantz, pelo apoio, dedicação, competência e especial atenção nas revisões e sugestões, fatores fundamentais para a conclusão deste trabalho.

Aos Professores do grupo GCA Professor Dr. Sandro Sawicki e a Professora Dra. Fabricia Carneiro Roos Frantz que destinaram parte de seu precioso tempo para participarem dessa pesquisa.

Ao grupo de professores do Mestrado em Modelagem Matemática que de alguma forma contribuíram para minha formação.

Aos professores coordenadores do Mestrado em Modelagem Matemática, Dra. Airam Sausen e Dr. Paulo Sausen.

Aos meus colegas do grupo GCA, pelo companheirismo e apoio nos momentos difíceis.

Aos meus colegas de mestrado, pelo convívio, amizade e estudo.

À Geni, pela atenção, disposição e incentivo.

Ao Ivan, pela disposição, conhecimento e ajuda.

Aos meus alunos, que ouviram as prévias das minhas apresentações.

A PROSUP/CAPES, pelo aporte financeiro que recebi, tornando possível a realização deste sonho, tão significativo para mim.

Resumo

O início de todas as coisas é pequeno.

Marcus T. Cicero, Filósofo Romano (106 AC - 43 AC)

Frequentemente, as empresas adquirem ou desenvolvem aplicações para apoiar a tomada de decisões e aperfeiçoar seus processos de negócio. Estas aplicações compõe o ecossistema de *software*, que geralmente é heterogêneo e ainda são desenvolvidas sem levar em conta sua possível integração, dificultando assim a sua reutilização. A área de *Enterprise Application Integration* (EAI) proporciona metodologias, técnicas e ferramentas para as empresas desenvolverem soluções de integração. O problema abordado nessa dissertação consiste em identificar os possíveis gargalos de desempenho na solução de integração que trata do processo e rematrículas da Universidade Unijuí, para que estes possam ser minimizados antes da implementação da solução. O aparecimento destes possíveis gargalos é um problema, porque se um modelo conceitual for implementado com gargalos, poderá gerar falhas, que aumentam os custos, tempo e riscos da solução. Nesse contexto, propõe-se identificar possíveis gargalos de desempenho, utilizando o modelo conceitual, por meio do qual é desenvolvido um modelo formal de simulação, utilizando o formalismo matemático das Redes de Petri Coloridas e Temporizadas. É por meio da simulação, que busca-se conhecer o comportamento do sistema, visando identificar tarefas que possam representar gargalos de desempenho. A partir da simulação, foi possível analisar duas variáveis: tempo médio de permanência das mensagens nos *slots* e tamanho máximo e médio dos *slots*. Os resultados da simulação das duas variáveis foram interpretados e analisados, identificando-se a ocorrência de gargalos de desempenho.

Palavras-chaves: Simulação, Redes de Petri Coloridas, Redes de Petri Temporizadas, Gargalos de Desempenho, Integração de Aplicações Empresariais.

Abstract

The beginnings of all things are small.

Marcus T. Cicero, Roman philosopher (106 BC - 43 BC)

Companies often acquire or develop applications to support decision-making and improve their business processes. These applications compose the software ecosystem, which is usually heterogeneous and its applications are frequently developed without taking into account integration, thus handicapping their reuse. The Enterprise Application Integration (EAI) area provides methodologies, techniques, and tools for companies to develop integration solutions. The problem addressed in this dissertation is to identify the possible performance bottlenecks in the integration solution which deals with the UNIJUÍ University process of re-enrollment, in order they can be minimized before the solution implementation. The occurrence of these possible bottlenecks is a problem, because if a conceptual model is implemented with bottlenecks, it can generate failures, which increase the costs, time and risks of the solution. In this respect, it is proposed to identify possible performance bottlenecks, using the conceptual model, whereby a formal simulation model is developed using the mathematical formalism from Timed and Colored Petri Nets. It is through the simulation that it seeks to know the behavior of the system, aiming to identify tasks that may represent performance bottlenecks. It was possible to analyze two variables from the simulation: messages average time of stay in the slots and maximum and medium size of the slots. The simulation results of the two variables were interpreted and analyzed, identifying the occurrence of performance bottlenecks.

Key-words: Simulation, Colored Petri Nets, Timed Petri Nets, Performance Bottlenecks, Enterprise Application Integration.

Capítulo 1

Introdução

Os sonhos não determinam o lugar
onde iremos chegar, mas produzem
a força necessária para tirar-nos
do lugar onde estamos.

Augusto Cury

Nesta dissertação é proposta a identificação de possíveis gargalos de desempenho, por meio do desenvolvimento de um modelo formal de simulação, utilizando o formalismo matemático das Redes de Petri Coloridas e Temporizadas. Com a simulação, foi possível observar o comportamento do sistema, sendo possível analisar duas variáveis: tempo médio de permanência das mensagens nos *slots* e tamanho máximo e médio dos *slots*, sob diferentes cenários. Dessa forma, este trabalho está organizado em seis seções. A Seção §1.1 apresenta o contexto no qual esta inserida a presente pesquisa. A Seção §1.2 trata da motivação para o desenvolvimento da pesquisa aborda, inicialmente, o problema de pesquisa e, posteriormente, a hipótese de solução do referido problema. A Seção §1.3 apresenta os objetivos geral e específicos da pesquisa. A Seção §1.4 apresenta a metodologia de desenvolvimento da pesquisa. A Seção §1.5 apresenta, de forma resumida, as contribuições desta pesquisa, no que diz respeito, à modelagem de soluções de integração e os trabalhos publicados. Por fim, a Seção §1.6 apresenta a estrutura organizacional da dissertação.

1.1 Contexto da Pesquisa

A utilização de sistemas de informação tem se tornado cada vez mais frequente nas práticas de negócios em empresas [41]. Nesse sentido, um sistema de informação pode ser definido como todo conjunto de dados e informações que são organizados de forma integrada, com o objetivo de atender a demanda e antecipar as necessidades dos usuários. Ademais, sistemas de informação para apoio à decisão são sistemas que coletam, organizam, distribuem e disponibilizam a informação utilizada nesse processo, com a finalidade de fornecer suporte aos processos de negócio da empresa [30]. Esta dissertação considera que um sistema de informação é um *software*, elaborado para apoiar as empresas em seus processos de negócio.

Segundo Pressman [51], *software* de aplicação define programas que solucionam uma necessidade específica dos processos de negócio, mediante aplicações que processam dados comerciais, tendo como intuito, facilitar as tomadas de decisão.

A engenharia de *software* é a área responsável pelo processo de criação do *software*. Um *software* pode ser desenvolvido seguindo cinco etapas, sendo elas: especificação, projeto, implementação, testes e evolução. A primeira etapa é responsável por definir o que se espera como resultado. Na segunda etapa são criados os modelos conceituais do *software*, sendo feita a descrição da estrutura do *software*, dos dados e das *interfaces* entre os componentes do sistema. A terceira etapa torna executável tudo o que foi realizado nas etapas anteriores. A quarta etapa detecta erros e verifica o *software* desenvolvido. A última etapa consiste em atender as necessidades mutáveis do cliente [58].

As empresas, usualmente, adquirem ou desenvolvem aplicações, visando eficácia em suas tomadas de decisão. Este conjunto de aplicações compõem o ecossistema de *software* das empresas, que normalmente é heterogêneo. Um ecossistema de *software* é composto por um conjunto de diferentes aplicações, que podem ser adquiridas de diferentes fornecedores ou desenvolvidas na própria empresa. Dessa forma, um ecossistema de *software* consiste em um conjunto de soluções de *software* que suportam e automatizam atividades e transações de usuários que estão associados a um ecossistema social ou de negócio [3].

Geralmente, as aplicações que compõem o ecossistema de *software* das empresas são heterogêneas e são desenvolvidas sem levar em consideração a

possibilidade de reutilização. Uma maneira de promover a comunicação entre essas aplicações é a utilização de ferramentas e tecnologias projetadas para a integração das aplicações. A integração de aplicações apresenta como vantagem a utilização da base tecnológica dos sistemas já em funcionamento, evitando assim um investimento de alto custo em novas plataformas [24].

A área para Integração de Aplicações Empresariais, do inglês *Enterprise Application Integration* (EAI), está relacionada à elaboração de soluções que resolvam problemas pontuais de integração dos processos de negócio. A EAI busca desenvolver metodologias, técnicas e ferramentas para criar soluções de integração que permitam reutilizar as aplicações do ecossistema de *software* por meio da sua integração [44]. O objetivo de uma solução de integração é manter em sincronia os dados e as funcionalidades das aplicações ou desenvolver novas funcionalidades a partir daquelas já existentes de tal forma que as aplicações não sejam alteradas pela solução [25]. Uma típica solução de integração, orquestra um conjunto de aplicações do ecossistema da empresa permitindo que seus dados e funcionalidades sejam compartilhados. Dentre as várias tecnologias disponíveis, que possibilitam projetar modelos conceituais de soluções de integração, estão: Camel [33], Mule [17], Spring Integration [20] e Guaraná [25].

Integrar aplicações não é uma tarefa trivial e o desenvolvimento da solução pode envolver custos, tempo e riscos. Portanto, busca-se, com esta pesquisa, analisar o comportamento da solução de integração, para identificar possíveis gargalos de desempenho, utilizando o modelo conceitual, através do desenvolvimento de um modelo formal de simulação aliado a técnicas de simulação de eventos discretos. Para o desenvolvimento do modelo de simulação, foi utilizado as Redes de Petri Coloridas e Temporizadas. As Redes de Petri Coloridas quando comparadas com as Redes de Petri tradicionais, caracterizam-se por diminuir o tamanho do modelo e permitem que seus *tokens* sejam individualizados, enquanto que as Redes de Petri Temporizadas associam tempo a componentes da rede.

Modelos discretos são orientados a eventos e são usados para modelar sistemas que mudam seu estado em momentos distintos no tempo, a partir da ocorrência de eventos. Soluções de integração podem ser caraterizadas como sistemas discretos, porque quando ocorre um evento todos os componentes envolvidos na solução consomem um tempo determinado de execução. Assim, a ocorrência de um evento altera o estado da solução.

A simulação é um método que pode utilizar um modelo matemático para possibilitar o estudo e a análise do comportamento do sistema sem que seja

necessário realizar alterações no sistema real podendo, assim, prever um comportamento futuro [56]. É através da simulação que busca-se conhecer o comportamento do sistema, visando melhorar o seu desempenho ainda na fase de projeto.

Para o desenvolvimento desta dissertação, optou-se por utilizar a tecnologia Guaraná, a qual permite projetar, implementar e executar soluções de integração, fatores estes que auxiliaram no desenvolvimento desta dissertação. Esta tecnologia foi criada por pesquisadores que atuam no Grupo de Pesquisa em Computação Aplicada (GCA), no qual a pesquisa foi realizada. A presente pesquisa visa contribuir para o aperfeiçoamento e evolução, no contexto de identificação de possíveis gargalos de desempenho em uma solução de integração.

Neste contexto, para tornar possível a análise do comportamento e a identificação de possíveis gargalos de desempenho na solução de integração que trata do processo de rematrículas da Universidade Unijuí, foi necessário conhecer um conjunto de temas fundamentais que serão introduzidos neste capítulo. Nessa perspectiva, é fundamental apresentar a motivação, os objetivos geral e específicos, a metodologia, as contribuições e a estrutura, que norteiam a presente dissertação.

1.2 Motivação

A análise do comportamento e a identificação de possíveis gargalos de desempenho em soluções de integração de aplicações geralmente envolve sua implementação para posterior execução e testes. Como as soluções são construídas, há custos (tempo e recursos) e riscos de falhas que costumam ser elevados e que podem comprometer o correto funcionamento das soluções de integração em situações em que tenham que processar um grande volume de informação [56]. Nesse sentido, a presente pesquisa é motivada pela possibilidade de analisar o comportamento e identificação de possíveis gargalos de desempenho, ainda na fase de projeto, levando em consideração os modelos conceituais das soluções e o formalismo matemático Redes de Petri Coloridas e Temporizadas.

Como conclusão, formulou-se a seguinte hipótese:

> *A busca por soluções de integração confiáveis e de qualidade prevê uma imprescindível análise do seu comportamento. Portanto, é possível utilizar modelos matemáticos aliados a técnicas de simulação de*

eventos discretos para analisar o comportamento e identificar possíveis gargalos de desempenho que podem surgir nas soluções de integração quando submetidas a diferentes cenários, tendo como base seus modelos conceituais. A simulação é realizada antes da implementação, com isso, pode-se detectar onde ocorrem os gargalos e possibilitar alterações para melhorar a solução.

1.3 Objetivos

Prever o comportamento de soluções de integração representa um passo importante para a diminuição dos custos (tempo, recurso) e riscos (falhas). Neste sentido, esta dissertação tem os seguintes objetivos:

1.3.1 Geral

Desenvolver um modelo de simulação com base no formalismo matemático das Redes de Petri Coloridas e Temporizadas, para analisar o comportamento e identificar possíveis gargalos de desempenho de uma solução de integração, responsável pelo processo de rematrículas da Universidade UNIJUÍ, ainda na fase de projeto.

1.3.2 Específicos

- Demonstrar a equivalência entre os elementos da solução de integração projetada na tecnologia Guaraná com os elementos de Redes de Petri Coloridas e Temporizadas;
- Elaborar um modelo de simulação, utilizando Redes de Petri Coloridas e Temporizadas equivalente ao modelo conceitual do Guaraná;
- Realizar experimentos, por meio de simulações com o *software* CPN Tools;
- Analisar o tempo médio de permanência das mensagens nos *slots*, a qual permite identificar em que *slots* as mensagens permaneceram por mais tempo;
- Analisar o tamanho máximo e médio dos *slots*, que permite identificar os *slots* em que estão ocorrendo acúmulos de mensagens;

- Analisar os resultados experimentais para identificar possíveis gargalos de desempenho e avaliar o comportamento da solução utilizando diferentes cenários;
- Verificar o modelo de simulação proposto, utilizando técnicas de verificação propostas na literatura.

1.4 Metodologia

A metodologia utilizada nesta pesquisa foi dividida em seis etapas.

Na primeira etapa, foram realizadas as leituras dos referenciais teóricos, as quais foram discutidas entre os integrantes do grupo de pesquisa GCA, permitindo o compartilhamento de ideias, juntamente com a definição das etapas e prioridades a serem seguidas durante o desenvolvimento da pesquisa.

Na segunda etapa, foram analisados os diferentes tipos de Redes de Petri, identificando quais poderiam ser aplicados ao contexto da Integração de Aplicações Empresariais.

Na terceira etapa, foi organizada uma tabela de equivalência entre os elementos da solução de integração projetada na tecnologia Guaraná com os elementos das Redes de Petri. A partir disso, foi possível elaborar um modelo de simulação, utilizando o formalismo matemático das Redes de Petri, representando satisfatoriamente a solução de integração. Em seguida, foram definidos os cenários de simulação e as variáveis a serem analisadas. A partir da definição dos cenários e variáveis foi necessária a elaboração de um novo modelo de simulação, o qual é representado pelo formalismo matemático das Redes de Petri Coloridas e Temporizadas. Elucida-se que este novo modelo foi necessário porque o modelo de simulação original, apesar de representar satisfatoriamente a equivalência dos elementos, não possibilita realizar as análises inerentes as questões de pesquisa. As simulações foram realizadas por meio da ferramenta *CPN Tools* que melhor se aplica ao tipo de Rede de Petri escolhida. A opção pela ferramenta deve-se ao fato dela ser capaz de simular Redes de Petri Coloridas, permitindo através do uso de monitores analisar o tempo médio de permanência das mensagens nos *slots* e o tamanho máximo e médio dos *slots*, sob diferentes cenários.

Na quarta etapa, foi realizada a verificação do modelo de simulação, por meio de técnicas de verificação formais encontradas na literatura.

Na quinta etapa, efetivou-se as simulações, as quais foram realizadas por meio da ferramenta CPN Tools que melhor se aplica ao tipo de Rede de Petri escolhida, ou seja, Rede de Petri Colorida e Temporizada. A simulação

proporcionou a análise da variável tempo médio de permanência e da variável tamanho máximo e médio dos *slots*, permitindo a identificação dos gargalos de desempenho.

Por fim, na sexta etapa, foi realizada a transcrição dos resultados alcançados para esta dissertação. Quanto aos resultados, estes podem ser utilizados para melhorar soluções de integração projetadas na tecnologia Guaraná, como também para serem publicados em conferências, para consolidar e ampliar a rede de colaborações acadêmicas nacionais e internacionais.

1.5 Resumo das Contribuições

Esta dissertação é parte integrante do projeto de pesquisa "Simulação para Predição do Comportamento de Soluções de Integração de Aplicações Empresariais", do grupo de pesquisa GCA. Busca-se, com esta pesquisa, contribuir para melhorar a qualidade das soluções desenvolvidas pelos engenheiros de *software*. As principais contribuições desta pesquisa são listadas a seguir:

- Seleção de uma técnica e ferramenta de simulação para analisar o comportamento e identificar possíveis gargalos de desempenho em soluções de integração de aplicações empresariais.
- Desenvolvimento de um modelo de simulação formal, equivalente ao modelo conceitual que trata das rematrículas das Universidade UNIJUÍ projetado na tecnologia Guaraná.
- Identicação de gargalos de desempenho, tendo como base o modelo conceitual, que possibilitou o desenvolvimento de um modelo de simulação com o auxílio do formalismo matemático Redes de Petri Coloridas e Temporizadas.
- Verificação do modelo de simulação, utilizando técnicas de verificação formais existentes na literatura.
- O estudo das ferramentas de simulação possibilitou analisar os *softwares* PIPE2 e Oris2, no que tange as suas diferentes características. Esta análise teve como intuito verificar qual dos *softwares* melhor se adapta ao problema de pesquisa. Resultando em uma publicação no IV Seminário de Formação Científica e Tecnológica (SFCT), intitulada "Comparação das Ferramentas PIPE2 e Oris2 quanto à Simulação", que foi apresentada na cidade de Santa Rosa, no dia 17 de junho de 2016 [38].

- O estudo de diferentes ferramentas de simulação gerou informações relevantes a respeito do uso desses *softwares*, no que tange a simulação e as semelhanças e diferenças entre propriedades inerentes a elas, estudadas em um *framework* de comparação. Os resultados obtidos foram publicados com o título de "*Framework* de Comparação entre Ferramentas de Simulação"na XXI Jornada de Pesquisa da UNIJUÍ, que foi apresentada na cidade de Ijuí, incorporada ao Salão do Conhecimento, realizado entre os dias 26 a 30 de setembro de 2016 [40].

- Durante o desenvolvimento da pesquisa, foi possível desenvolver um modelo de simulação equivalente ao modelo conceitual, utilizando o formalismo matemático das Redes de Petri. Este modelo de simulação trata do processo de rematrículas da Universidade UNIJUÍ. O resultado desta pesquisa foi publicado na 8ª edição da *Conference on ENTERprise Information System*, intitulado de *"Mathematical Model for Simulation an Application Integration Solution in the Academic Context of Unijuí University"*, que ocorreu nos dias 05, 06 e 07 de outubro de 2016, em Porto, Portugal [39]. Além da participação e a apresentação nestes dias no evento, também tive a oportunidade de fazer uma visita com a missão de pesquisa no Instituto Politécnico de Leiria (IPL). Na ocasião fui recepcionada pelo professor Dr.Vitor Manuel Basto Fernandes, que também é professor colaborador do Programa de Pós-Graduação em Modelagem Matemática e membro do Grupo de Computação Aplicada (GCA) da UNIJUÍ.

1.6 Estrutura desta Dissertação

Esta dissertação está organizada da seguinte maneira:

Capítulo I: Introdução. Compreende a presente introdução.

Capítulo II: Revisão da Literatura. Apresenta o referencial teórico. Primeiramente, abordam-se os conceitos referentes à integração de aplicações empresariais, em seguida, apresenta-se a tecnologia Guaraná, posteriormente discute-se a simulação de sistemas de eventos discretos, após, são feitas as definições de distribuição de probabilidade e de Redes de Petri, em especial as Coloridas e Temporizadas, evidenciando seus elementos e suas principais características. Por fim, apresenta os trabalhos relacionados ao estudo e desenvolvimento desta pesquisa.

Capítulo III: Modelagem da Solução de Integração. Apresenta o desenvolvimento do trabalho que é realizado por meio do modelo conceitual, tornando possível produzir um modelo de simulação utilizando Redes de Petri Coloridas e Temporizadas, identificando-se os gargalos de desempenho. O capítulo aborda a equivalência dos elementos da linguagem de domínio específico da plataforma Guaraná, referenciada por Guaraná DSL com os de Redes de Petri e o modelo de simulação desenvolvido.

Capítulo IV: Experimento e Análise dos Resultados. É descrito o processo de experimentação adotado, bem como a descrição dos cenários, variáveis e a descrição da ferramenta de simulação. Ainda neste capítulo são apresentados os resultados e discussões. Finalizando, com a discussão dos conceitos de verificação e validação. É neste momento que realiza-se a verificação do modelo de simulação desenvolvido.

Capítulo V: Considerações Finais. São apresentadas as conclusões a partir da pesquisa desenvolvida nesta dissertação juntamente com as possibilidades de trabalhos futuros.

Capítulo 2

Revisão da Literatura

Nunca consideres o estudo como uma
obrigação mas sim como uma oportunidade
para saberes mais.

Albert Einstein

Para tornar possível a análise do comportamento e a identificação de possíveis gargalos de desempenho em soluções de integração de aplicações empresariais, é necessário conhecer um conjunto de temas fundamentais que serão introduzidos neste capítulo. A Seção §2.1 trata da integração de aplicações empresariais, estilos de integração, topologias e plataformas de integração. A Seção §2.2 apresenta a plataforma Guaraná. A Seção §2.3 aborda os conceitos e a fundamentação teórica sobre sistema, modelo e simulação de eventos discretos. A Seção §2.4 refere-se à distribuição de probabilidade. A Seção §2.5 trata das Redes de Petri Coloridas e Temporizadas. A Seção §2.6 apresenta os trabalhos relacionados. Por fim, a Seção §2.7 apresenta o resumo do capítulo.

2.1 Integração de Aplicações Empresariais

O que demanda a modificação dos processos de negócio atuais ou o surgimento de novos nas empresas é a constante evolução da dinâmica de negócios das mesmas. Para que estes processos de negócio sejam implantados, geralmente é necessário o suporte computacional. No decorrer dos anos,

as empresas vêm adquirindo pacotes de aplicativos de terceiros ou desenvolvendo aplicativos sob medida para dar suporte aos seus processos de negócio. Isso tem resultado em um ecossistema de *software* heterogêneo no que diz respeito a tecnologias de desenvolvimento e modelos de dados. Verifica-se que estes aplicativos geralmente não foram projetados com a preocupação de integração, ou seja, eles não fornecem uma *interface* que permita uma troca de dados ou compartilhamento de funcionalidades entre as aplicações. Como resultado, a integração nem sempre é uma tarefa trivial, na maioria dos casos por meio dos recursos que pertencem às aplicações, tais como seus bancos de dados, arquivos de dados, filas de mensagens e *interfaces* de usuário. Desafios recorrentes, são para fazer as aplicações interoperarem para manter suas informações sincronizadas, oferecer novos pontos de vista de informação, ou para criar novas funcionalidades. Com isso, tornam-se necessárias as soluções de integração, as quais são implantadas no ecossistema de *software* como um novo aplicativo que oferece a seus usuários uma visão de alto nível das aplicações integradas com as quais um processo de negócio interage [25].

As soluções de integração podem ser classificadas de acordo com os objetivos da integração de funcionalidades ou dados. No primeiro grupo, existem dois tipos de integração, a saber: *Enterprise Application Integration* (EAI) e *Business-to-Business Integration*. O último grupo inclui outros três tipos de integração, a saber: *Enterprise Information Integration*, *Extract, Transform and Load*, e *Mashup*. A *Enterprise Application Integration* (EAI) busca integrar aplicações que geralmente pertencem a mesma empresa e encontram-se no mesmo ecossistema de *software*.

A *Enterprise Application Integration* (EAI) se concentra em fornecer metodologias e ferramentas para integrar as muitas aplicações do ecossistema de *software* das empresas. Nos últimos anos, com o surgimento da computação em nuvem, o ecossistema de *software* de uma empresa passou a incluir também serviços que estão na nuvem. Neste sentido, a EAI visa manter uma série de dados das aplicações em sincronia ou desenvolve novas funcionalidades em cima deles, de forma que as aplicações não tenham que ser alteradas e que não sejam afetadas pela solução de integração. Nota-se que a demanda da empresa é de compartilhar dados e processos sem ter que fazer grandes mudanças para as aplicações ou estruturas de dados. Sendo assim, somente a criação de um método que realiza integrações pode ser funcional, tendo na EAI um custo efetivo. *Business-to-Business Integration* é semelhante à *Enterprise Application Integration*, porém o primeiro visa integrar aplicações de ecossistemas de *softwares* de diferentes empresas, enquanto que a segunda visa integrar aplicações de ecossistemas de *softwares* dentro da própria empresa [32].

Neste contexto, surgem duas questões importantes: em primeiro lugar, as aplicações no ecossistema de *software* de uma empresa não estão abertas ao mundo exterior; em segundo lugar, a infraestrutura para a comunicação das aplicações em diferentes ecossistemas, envolve o uso público de redes em que a segurança e a confiabilidade da qualidade do serviço são as principais preocupações. A *Enterprise Information Integration* permite desenvolver soluções de integração que fornecem uma vista homogênea e *on-line* dos dados manipulados. Sobre este ponto de vista, os engenheiros de *software* podem executar operações usando uma linguagem declarativa que lhes permite consultar e modificar dados nos aplicativos integrados. Uma diferença importante entre *Enterprise Application Integration* e *Enterprise Information Integration* é que não existem fluxos de dados para manter as aplicações sincronizadas neste último, mas um vasto número de dados. A semelhança entre elas é de que a *Enterprise Application Integration*, proporciona a integração com aplicações de um mesmo ecossistema de *software*. *Extract, Transform and Load* (ETL) são semelhantes a *Enterprise Information Integration*, a diferença é que é *off-line*. Em ETL, os dados das distintas fontes são carregados para uma única base de dados, a qual posteriormente será utilizada para consultas de informações. Nesse sentido, este tipo de integração requer uma base de dados persistente em que os mesmos são extraídos das aplicações integradas podendo ser armazenados, para posterior processamento. *Mashups* diferem dos tipos anteriores de integração, pois, estão focados na integração exclusivamente de dados a partir de serviços *web* [32].

A presente pesquisa centra-se na *Enterprise Application Integration* (EAI), pois sabe-se que a tarefa de integrar uma solução é muito difícil. Uma solução de integração, pode processar milhares ou até mesmo milhões de mensagens trocadas entre vários aplicativos, que podem ter seu estado alterado por todas as mensagens. Além disso, pode haver gargalos de desempenho não apenas na integração da solução, mas também nas aplicações a serem integradas, devido à comunicação com a solução de integração. Para tornar as coisas ainda mais difíceis, as partes envolvidas em uma solução de integração comunicam-se de forma assíncrona, podendo ser distribuídas dentro do ecossistema de *software*, tendo a possibilidade de falhas [56].

O restante da seção está organizada da seguinte forma: a Seção §2.1.1 apresenta os estilos de integração; a Seção §2.1.2 introduz as diferentes topologias para soluções de integração; a Seção §2.1.3 apresenta as distintas plataformas utilizadas para implementar, executar e monitorar soluções de integração.

2.1.1 Estilos de Integração

Apesar de existirem diversos critérios sobre a abordagem dos estilos de integração, existem quatro estilos de integração de aplicativos, cuja classificação foi proposta por Hohpe e Woolf [32] que devem ser abordados, a saber: Transferência de Arquivos, em que os arquivos de dados são produzidos por uma aplicação e consumidos por outras; Banco de Dados Compartilhado, em que as aplicações gravam dados em um banco de dados para que sejam compartilhados com outras aplicações; *Remote Procedure Invocation*, no qual as operações de uma aplicação são expostas, a fim de que possam ser remotamente chamadas por outras aplicações, para realizar determinadas tarefas ou para compartilhar dados e o *Messaging*, através do qual ocorrem trocas de mensagens. As aplicações compartilham dados e invocam operações de um sistema de mensageria, ao qual estão conectadas [32].

Transferência de Arquivos

Para resolver problemas de incompatibilidade e fazer com que as aplicações trabalhem de forma assíncrona, torna-se necessário um mecanismo de transferência de dados. Este mecanismo pode ser utilizado em diversas plataformas e linguagens, dessa maneira, torna-se possível ser compatível com todas as aplicações existentes na empresa. Salienta-se que nem todos esses arquivos terão o mesmo formato, neste sentido, poderá ser necessário, realizar transformações de formato para adaptar os dados de uma aplicação a outra. A transferência de arquivos ocorre quando cada aplicativo armazena dados que deseja compartilhar com outros aplicativos, além disso o aplicativo também pode consumir arquivos produzidos por outros [32].

Apesar da transferência de arquivos ser considerada, muitas vezes, simples pelo fato de não ter necessidade de criar ferramentas ou plataformas de integração, torna-se necessário ter o cuidado com os nomes das aplicações, pois esses devem concordar entre si, os nomes das aplicações devem ser únicos, sendo imprescindível que as aplicações criem estratégias para que uma não tente ler um arquivo, antes da outra tê-lo escrito. Pode ser necessário, que uma aplicação fique responsável pela transferência dos arquivos de um disco para outro. Outro fato relevante é que as atualizações não ocorrem com grande frequência, o que pode ocasionar falta de sincronização [32].

A transferência de arquivos pode ser definida através da Figura §2.1, que apresenta a exportação de dados de uma aplicação para um arquivo e a outra

aplicação que apresenta a importação desses dados para serem processados. O maior problema na transferência de arquivos está em gerenciar todos os arquivos que são produzidos e garantir que todos eles serão lidos e nenhum irá se perder, pois, existem custos com o processamento de um arquivo, e estes podem ser bem onerosos, o que torna especialmente difícil produzir arquivos rapidamente [32].

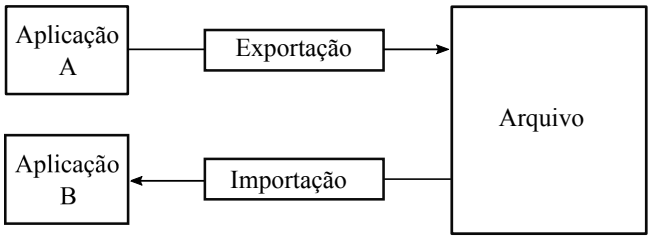


Figura 2.1: *Transferência de Arquivos. (Hohpe e Woolf [32])*

Banco de Dados Compartilhado

Para que a empresa tenha eficiência em seu trabalho, é necessário que as informações sejam compartilhadas de maneira rápida e consistente. Nesse sentido, para que o gerenciamento dos arquivos ocorra de forma satisfatória, é necessário criar uma central de armazenamento de dados que conecte todas as aplicações de modo que qualquer aplicação tenha acesso aos dados sempre que for necessário. Desse modo, as aplicações serão conectadas a essa base de dados única e, portanto, poderão compartilhar dados umas com as outras por meio dessa base, armazenando os dados em somente um banco de dados compartilhado [32].

Conforme apresenta a Figura §2.2, é possível definir o banco de dados compartilhado, pois, ao compartilhar uma mesma base de dados, todas as aplicações passam a lidar com dados que são compatíveis e consistentes. Se as atualizações passam a ocorrer simultaneamente em um determinado banco de dados, criam-se sistemas de gerenciamento de transações, em que estes manipulam os dados de forma eficiente, de modo que o tempo decorrente entre as atualizações seja pequeno, tornando fácil encontrar e corrigir possíveis falhas [32].

Um banco de dados compartilhado tem como principal tarefa facilitar o gerenciamento de dados, pois eles sempre estarão no mesmo formato de

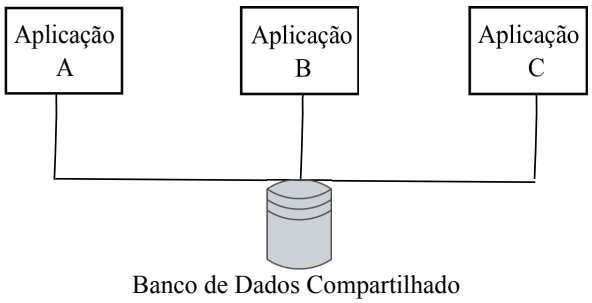


Figura 2.2: *Banco de Dados Compartilhado. (Hohpe e Woolf [32])*

arquivo. Pelo fato de todas as aplicações utilizarem o mesmo banco de dados, o criador da aplicação se vê obrigado a corrigir problemas de forma bastante rápida, para que as demais aplicações não sejam afetadas. Percebe-se que também é possível encontrar problemas com a utilização do banco de dados compartilhado, dentre estes, um dos maiores é encontrar um *design* adequado, que atenda às possíveis aplicações. Existem também outros tipos de dificuldades como, por exemplo, conflitos internos, que podem ocorrer entre os que formam a equipe de trabalho, com relação às aplicações que, muitas vezes, podem ser elaboradas em formatos diferentes [32].

Chamada de Procedimento Remoto

Um dos mais poderosos mecanismos estruturantes no *design* do aplicativo é o do encapsulamento, no qual os módulos escondem seus dados através de uma interface chamada função. Desse modo, eles podem interceptar as alterações nos dados para realizar as mais diversas ações. Quando os dados são alterados, o banco de dados compartilhado fornece uma grande estrutura de dados não encapsulado, o que torna muito mais difícil a alteração dos dados. No *Remote Procedure Invocation*, ocorre o princípio de encapsulamento dos dados, para que seja possível realizar a integração de aplicações. No momento em que uma aplicação necessita de algum dado, que está em outro aplicativo, a mesma faz um pedido para que a aplicação lhe passe estes dados. Quando torna-se necessário que um aplicativo modifique dados de outro, é preciso chamar o aplicativo de origem. As aplicações têm como obrigação manter a integridade dos seus dados, além disso cada uma delas tem a capacidade de alterar seus dados sem afetar as outras aplicações [32].

A Figura §2.3 define Chamada de Procedimento Remoto como estilo que permite o compartilhamento de funcionalidades. Com este estilo, uma

aplicação expõe uma interface, que é um conjunto de funções, às demais aplicações do ecossistema, que podem então por meio dessa interface chamar funcionalidades daquela aplicação ou obter dados [32].

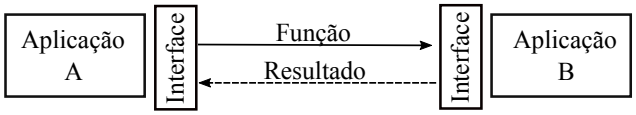


Figura 2.3: *Chamada de Procedimento Remoto. (Hohpe e Woolf [32])*

Existem diferentes maneiras de tornar mais fácil a compreensão dos dados pelas diferentes aplicações. As aplicações podem fornecer inúmeras interfaces utilizando os mesmos dados, o que permite realizar alterações conforme se torne necessário, proporcionando uma capacidade maior de suportar diferentes tipos de dados. Apesar do encapsulamento de dados ajudar a reduzir o acoplamento de aplicações, os programadores, muitas vezes, projetam a integração imaginando utilizar somente uma aplicação, o que pode vir a causar problemas futuros.

Messaging

O *Messaging*, pode ser definido através da Figura §2.4, pois é utilizado para transferir dados e compartilhar funcionalidades de modo imediato, confiável e assíncrono. Esta ferramenta utiliza formatos personalizáveis, que faz com que cada aplicação se conecte a um sistema de mensagens comum, fazendo trocas de dados. As trocas de mensagens entre as aplicações ocorre de forma assíncrona, ou seja, torna-se uma solução para os problemas de distribuição de sistemas [32]. É importante lembrar que o *Messaging* pode transmitir dados ou funcionalidades, não sendo necessário preocupar-se com seu formato.

Este estilo geralmente é a melhor abordagem para integração de aplicações empresariais, porém não se deve pensar que ele não tem problemas. Quanto maior for a frequência de mensagens, maior é a redução dos problemas de inconsistência que geram problemas na transferência de arquivos, mas nunca serão totalmente excluídos. Neste contexto, a capacidade de transformar as mensagens, permite que as aplicações sejam dissociadas umas das outras [32].

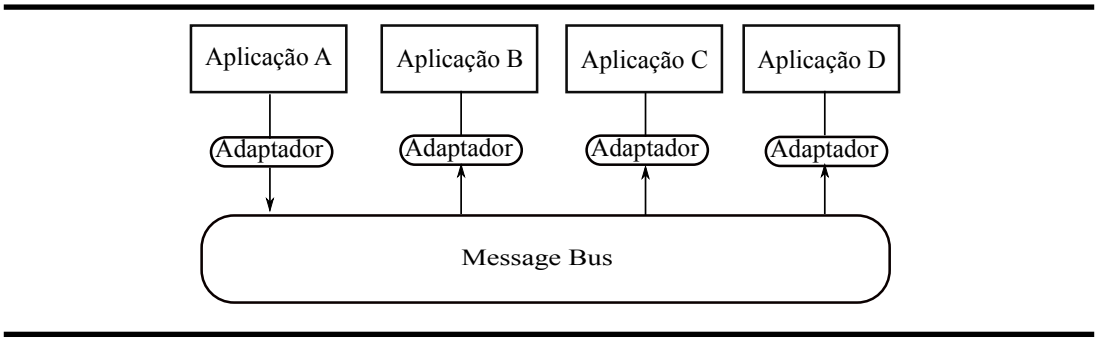


Figura 2.4: *Messaging. (Hohpe e Woolf [32])*

2.1.2 Topologias

De acordo com Soeiro [57], existem três tipos de topologias que indicam como as aplicações se relacionam em uma solução de integração, ou seja, como estão conectadas umas com as outras. Estas três topologias são classificadas em: *Point-to-Point*, *Hub-and-Spoke* e *Bus*:

Point-to-Point

É a topologia mais simples e mais utilizada pelas empresas para construir suas soluções de integração. A Figura §2.5 apresenta essa topologia, que cria um canal de comunicação direto entre duas aplicações, podendo ser síncrono ou assíncrono. Com isso, permite-se a troca de mensagens, ou seja, uma comunicação direta entre duas aplicações.

A Figura §2.5 apresenta cinco aplicações de um ecossistema de *software* e as flechas representam conexões *point-to-point* entre essas aplicações. Assim, o número de comunicações necessárias para n aplicações, é encontrada utilizando-se a equação §2.1:

$$Total\ de\ comunicacoes = \frac{(n(n-1))}{2} \tag{2.1}$$

No exemplo da Figura §2.5, existem 5 aplicações a serem comunicadas, neste sentido, através da fórmula, percebe-se que são necessárias 10 comunicações para interligar todas. Entretanto, a comunicação entre 10 aplicações, por exemplo passa a demandar 45 comunicações para interligar

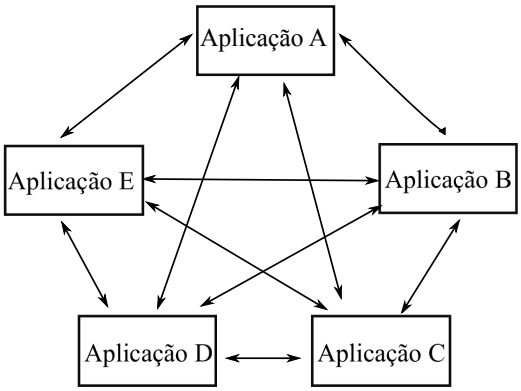


Figura 2.5: *Topologia Point-to-Point. (Martins [48])*

todas. Em razão desse crescimento exponencial, a *point-to-point* é uma topologia recomendada quando o número de aplicações envolvidas é baixo.

Nessa topologia, as decisões são tomadas de forma bilateral, mas o processamento é executado de forma local. As decisões são assim porque há necessidade de ambas as aplicações concordarem quanto ao mecanismo de funcionamento da integração. Essa topologia de integração faz com que as aplicações fiquem fortemente acopladas. Em caso de substituição ou de mudança drástica em uma das aplicações, o mecanismo de integração terá que ser refeito. A comunicação entre as aplicações pode ser feita seguindo o estilo baseado em arquivos, banco de dados ou RPC [57].

O crescimento exponencial ocorre quando o número de aplicações aumenta e, consequentemente, o número de conexões cresce de forma exponencial, tornando-se difícil de administrar, sendo uma arquitetura pouco escalável. Outra desvantagem é o fato desta topologia ser pouco flexível, ou seja, apresenta pouca capacidade de adaptação à mudança. Isto deve-se ao fato da ocorrência de uma pequena mudança num dos pontos, ser propagada para vários pontos distintos, tornando-se difícil de prever os efeitos que as mudanças podem ocasionar no conjunto dos sistemas integrados. Por fim, importa ainda destacar a dificuldade de monitorar as interfaces numa arquitetura deste tipo, devido ao fato da informação não passar por um ponto de controle externo onde possa ser auditada [57].

Hub-and-Spoke

Esta topologia surgiu com a intenção de resolver os problemas das arquiteturas, a nível de escalabilidade e manutenção [57]. Esta topologia é

denominada de Arquitetura EAI por muitos autores, por ter sido a primeira a utilizar os princípios EAI, embora estes também se encontrem em outras topologias [44].

A topologia *Hub-and-Spoke* centraliza a comunicação num canal de comunicação assíncrono e comum *(hub)* para troca de informação, como apresenta a Figura §2.6. Sendo assim, uma aplicação manda uma mensagem para outra aplicação, através do *hub*, que é o responsável por encaminhar estas mensagens, através do *spoke*, que executa o papel de distribuidor das mensagens. Nesta topologia, o que fluí entre as aplicações e, portanto, passa pelo *hub* são as mensagens. Nesse caso, as mensagens em um *hub* devem levar no seu cabeçalho o endereço de destino, ou seja, a aplicação para a qual devem ser enviadas.

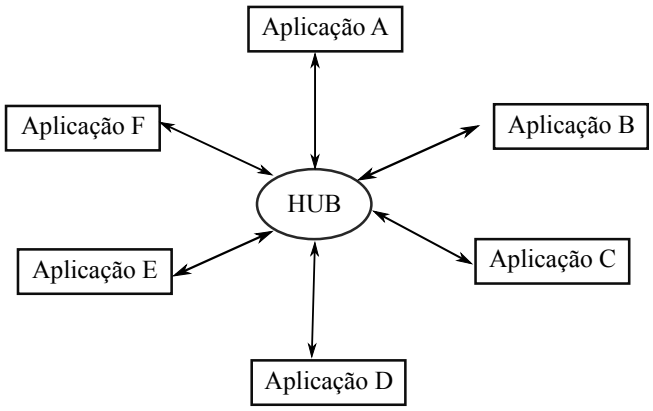


Figura 2.6: *Topologia Hub-and-Spoke. (Martins [48])*

Nesta topologia, o número de ligações cresce linearmente com o número de aplicações integradas. Percebe-se, na Figura §2.6, que para um conjunto de 6 aplicações a integrar, são necessárias apenas 6 comunicações. O que ocorre é que a aplicação manda uma mensagem que deve ter um formato canônico, ou seja, comum ao *hub*, que distribui para as outras aplicações. O que fluí entre a aplicação e o *hub* é a mensagem.

O *hub* é o responsável pela validação, transformação e reencaminhamento das mensagens, isto se deve ao fato das regras já terem sido configuradas anteriormente. A configuração é centralizada, facilitando a comunicação do sistema. Outro fator positivo é de que esta topologia é centralizada no *hub*, fazendo com que o *hub* seja um ponto único de falha e revele problemas de escalabilidade à medida que o volume de mensagens aumentam. As

desvantagens são de que as aplicações são consideradas fracas, pois não conseguem conhecer as aplicações as quais se integram, pois comunicam-se através do *hub*. Outro fato é de que o *hub* é geralmente dispendioso e proprietário, o que dificulta a ligação a novas aplicações [57].

Enterprise Service Bus

A arquitetura *Enterprise Service Bus* (ESB) procura aproveitar as vantagens da *Hub-and-Spoke*, enquanto tenta resolver os seus problemas. Entre as vantagens que se mantêm nesta topologia, pode-se contabilizar a gestão centralizada, o baixo número de ligações necessárias (que se mantém igual ao número de aplicações integradas). A estas vantagens acrescentam-se a escalabilidade desta arquitetura pelo fato de ser distribuída, o que também permite eliminar um ponto único de falha, como é o caso dos *Brokers*. Também é importante destacar a possibilidade de disponibilizar funcionalidades remotas através de serviços que outras aplicações podem facilmente utilizar, seguindo o paradigma SOA *(Service Oriented Architecture)*. Uma topologia SOA tem como princípio a disponibilização das funcionalidades das aplicações existentes sobre a forma de serviços reutilizáveis, utilizado tecnologias *standards* como *Web Services*. Porém, a topologia *Hub-and-Spoke* também apresenta alguns pontos negativos, como: utilização obrigatória de um componente adicional, o *bus*, que tem que estar acessível a partir de todas as máquinas, nos quais são alojados serviços de integração; existência de algum *overhead* associado à comunicação de mensagens através do *bus*, devido às transformações que as mensagens sofrem para poderem serem enviadas e recebidas nos *endpoints*; ao criar serviços, é necessário o acordo entre as entidades que os vão utilizar para difinir a estrutura das mensagens, uma vez que, diferentes versões da mesma mensagem anulam o *loose coupling* [57].

Conforme apresenta a Figura §2.7, a comunicação que ocorre no *Enterprise Service Bus* é de que são difundidas mensagens para um ou mais destinos que as reconhecem subscrevendo a sua recepção. O canal de comunicação é denominado *bus* permitindo uma comunicação assíncrona.

Ao contrário do *Hub-and-Spoke*, esta é uma arquitetura distribuída, baseada em serviços de integração, que correm dentro de *Service Containers*. Segundo Soeiro [57], estes *containers* são aplicações leves, instaladas em diferentes máquinas cuja função é alojar um número arbitrário de:

- *Middleware* de integração: componente vital da arquitetura que define a topologia da mesma;

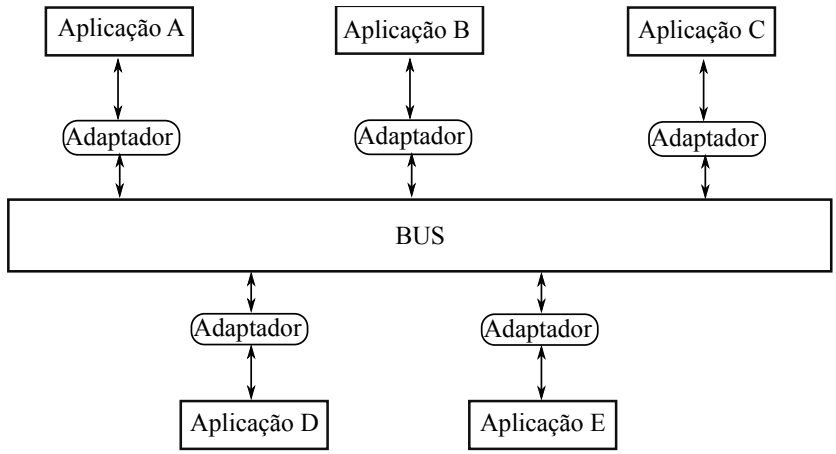


Figura 2.7: *Topologia Enterprise Service Bus. (Martins [48])*

- Adaptadores: componentes de *software* responsáveis pela ligação entre as aplicações externas (ou fontes de dados) e o *middleware* utilizado (e.g. adaptador SQL para estabelecer ligação a bases de dados ou adaptador HL7 capaz de enviar e receber mensagens utilizando este *standard*);
- Armazenamento de mensagens: as mensagens recebidas e enviadas devem ser armazenadas em um formato *standard* (e.g. XML) para poderem ser consultadas ou reenviadas mais tarde;
- Transformação de mensagens: possibilidade de transformar as mensagens com base em regras, normalmente recorrendo a tecnologias *standard* (e.g. XSLT);
- Reencaminhamento de mensagens: possibilidade de definir regras para reencaminhar automaticamente as mensagens, com base em configurações prévias ou no conteúdo da mensagem.

Pode-se destacar que, ao contrário do *hub*, que tem apenas o papel de distribuidor, o *bus*, também, proporciona serviços adicionais como: *Orchestration language*, *Message routing*, *Message transformation*, *Monitoring system*, *Toolkit of adapters*, *Distributed deployment*, *Transaction support* e *Usually based on open standards*.

As infraestruturas mais modernas seguem essa topologia. A *Enterprise Service Bus* (ESB) é uma tecnologia para implementar o *bus*, o qual trans-

porta mensagens que estão conectadas a ele. A plataforma de integração oferece vários recursos para implementar o ESB.

A Tabela §2.1 apresenta a comparação das topologias. Nesse sentido, para uma melhor compreensão, é possível compará-las e analisar suas semelhanças e diferenças no que tange as suas características mais importantes [57]:

	Point-to-Point	**Hub-and-Spoke**	**Enterprise Service Bus**
Número Máximo de Ligações para n Aplicações	$\frac{n(n-1)}{2}$	n	n
Loose Coupling	Não	Sim	Sim
Processamento	Dividido por duas aplicações	Centralizado (no Hub)	Distribuido
Gestão Centralizada	Não	Sim	Sim
Escalabilidade	Muito baixa	Baixa	Elevada
Ponto Único de Falha	Não	Sim	Não
Middleware Adicional	Não	Não	MOM
Overhead Computacional	Nenhum (ligações diretas)	Conversão entre formatos	Fila de mensagens (MOM) e conversão entre formatos
Investimento Inicial	Só o estritamente necessário	Elevado	Elevado
Criação de Interfaces	Interfaces são programadas	Interfaces são configuradas	Interfaces são configuradas

Tabela 2.1: *Comparação das Características das Topologias. (Soeiro [57])*

2.1.3 Plataformas de Integração

Existem diversas plataformas para o desenvolvimento e implementação de soluções de integração disponíveis no mercado. A plataforma é uma interface projetada para o desenvolvimento e implementação de soluções de integração, tendo como base o processo de criação desenvolvido por peritos, que deverá dispor de capacidades de gestão e geração de modelos de instrução para dar suporte às decisões e efetuar a integração das aplicações [15]. As plataformas para desenvolvimento de soluções de integração, proporcionam um conjunto de ferramentas que, geralmente, inclui uma linguagem de domínio específico (DSL), uma *application programming interfaces* (API) de programação, um motor de execução e ferramentas de

monitoramento. Para a tarefa de modelagem, as plataformas costumam oferecer linguagens de domínio específico que dão suporte a criação de modelos conceituais para as soluções de integração. A *application programming interfaces* permite aos desenvolvedores implementar os modelos conceituais em código executável. O motor de execução é responsável pela execução das soluções de integração implementadas. As ferramentas de monitoramento permitem acompanhar o funcionamento de uma solução de integração, seja no seu desempenho, na quantidade de mensagens recebidas e na quantidade de mensagens enviadas, dentre outras funções.

Ademais, o DSL foi desenvolvido especificamente para um domínio, tendo como objetivo principal desenvolver a semântica específica para aquele determinado domínio. Geralmente, o DSL é pequeno, de fácil entendimento, melhora a comunicação entre um grupo de especialistas daquele domínio, expressa soluções para o problema e tem uma expressividade limitada do domínio. O DSL possui um nível de abstração elevado e geralmente é utilizado para aumentar a comunicação e a produtividade entre as aplicações.

Uma API é um conjunto de padrões de programação que permite a construção de aplicativos e a sua utilização. A API, geralmente, se comunica com diversos códigos interligando várias funções em uma mesma aplicação. A criação de uma API é difícil e onerosa, um *software* que possui API sem dúvida exige muito mais trabalho do que um *software* sem API. Porém, a utilização da API permite criar sistemas melhores, facilitando o entendimento dos mesmos.

O motor de execução da plataforma de integração possui o *scheduler*, que é o coração do motor de execução. O *scheduler* é responsável por executar as tarefas das soluções implementadas, que ocorre através de notificações. Assim, uma tarefa, dentro do motor de execução está pronta para ser executada quando tiver recebido uma mensagem ou mais em todas as suas entradas. A mensagem de entrada é processada pela tarefa, gerando zero ou mais mensagens de saída.

As ferramentas de monitoramento permitem o gerenciamento de conteúdos, análise de engajamentos e integração dos resultados aos processos de negócios da empresa. Geralmente, as plataformas de monitoramento integram aplicações já existentes dentro da empresa com outras que vão sendo adquiridas no decorrer do tempo, dependendo das necessidades que venham a surgir. As ferramentas de monitoramento tem por objetivo a melhora do gerenciamento e execução dos processos de negócio.

Além disso, pode-se destacar que existem distintas plataformas para implementar soluções de integração e que as mais modernas estão voltadas à

implementação de soluções de integração, seguindo a topologia *bus*. Dentre as plataformas open-source mais conhecidas para a integração de aplicações empresariais, estão Camel [33], Guaraná [25], Mule [17] e Spring [20].

Nesse sentido, a presente pesquisa propõe-se a utilizar a linguagem de domínio específico da plataforma Guaraná, referenciada por Guaraná DSL pode ser usado como um vocabulário comum e, ainda assim, simples para se comunicar no campo da *Enterprise Application Integration*. Guaraná DSL fornece um conjunto de ferramentas de tarefas de uso geral, que contém uma coleção de tarefas que fornece as bases para muitos outros *toolkits* de tarefas para fins especiais para diferentes contextos de integração. Oferece também uma notação gráfica que pode ser usada para representar os modelos de soluções de integração com alto nível de abstração independente de plataforma. Guaraná DSL define um conjunto de construtores para as abstrações fundamentais envolvidos nas soluções de integração que podem ser usados por engenheiros de *software*, com um vocabulário acessível e de fácil compreensão [56].

2.2 Plataforma Guaraná

A tecnologia Guaraná é utilizada para projetar soluções de integração de aplicações empresariais. Dispõe como recursos uma linguagem de domínio específico e um motor de execução (do inglês *runtime system*), que permite a implementação e execução da solução de integração.

2.2.1 Linguagem de Domínio Específico

As variadas tecnologias utilizadas para o desenvolvimento dos modelos de solução de integração adotam um DSL, que propõe uma linguagem própria para a modelagem de soluções de integração, dentro de um determinado contexto, de modo a simplificar códigos complexos, facilitando a compreensão da sua estrutura e funcionamento. A linguagem DSL tem excelência dentro do domínio que opera, porém perde eficiência se submetida a situações fora do seu domínio [23].

A tecnologia Guaraná proporciona uma linguagem de domínio específico, que permite projetar soluções de integração a um alto nível de abstração utilizando uma sintaxe concreta gráfica e conceitos de modelagem intuitivos. Esta linguagem de modelagem é baseada nos padrões de integração documentados por Hohpe e Woolf [32].

2.2.2 Notação Gráfica

A estrutura do Guaraná é fundada nos padrões de integração de aplicações, documentados por Hohpe e Woolf [32]. Nessa estrutura, a representação de tarefas possibilita à tecnologia Guaraná projetar todos os processos de integração e suas portas de comunicação. A Tabela §2.2 apresenta a notação gráfica utilizada para representar os conceitos da tecnologia Guaraná.

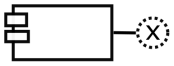
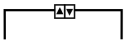
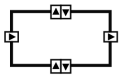
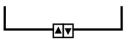
Notação	Conceito	Notação	Conceito
	Aplicação		Porta de Solicitação
	Processo de Integração		Porta de Resposta
	Porta de Entrada		Tarefa
	Porta de Saída		Slot

Tabela 2.2: *Notação Gráfica da Tecnologia Guaraná DSL. (Frantz [26])*

A tecnologia Guaraná DSL é composta por cinco elementos: aplicação, processo de integração, portas, *slots* e tarefas. Na Tabela §2.2, é possível observar como cada componente é representado graficamente.

Os processos são blocos os quais agrupam as tarefas. Os processos possuem portas, as quais são responsáveis por fazer a comunicação com as aplicações. Estas portas podem ser: (a) de entrada, (b) saída, (c) solicitação ou resposta.

Os *slots* equivalem as unidades de armazenamento temporário. Eles viabilizam o processamento assíncrono das mensagens que estão inseridas em um processo, possibilitando independência entre as tarefas. Neste contexto, os *slots* interligam as tarefas, assim cada *slot* recebe a mensagem já processada da tarefa anterior e a deixa disponível para ser processada pela tarefa seguinte. Uma vez que uma mensagem é processada e despachada para os *slots* seguintes, a tarefa está pronta para processar outra mensagem [23].

As portas possibilitam o envio e recebimento de mensagens. Assim, a porta de entrada envia uma mensagem para um *slot* e este a disponibiliza para uma tarefa. Já a porta de saída lê sempre uma mensagem a partir de um *slot* e a deixa disponível para o elemento seguinte no fluxo. Portas de solicitação e resposta permitem a comunicação entre o processo e as aplicações integradas.

Uma mensagem compõe-se de cabeçalho e corpo. O cabeçalho contém propriedades já pré-definidas, como identificador de mensagens, identificador de correlação e prioridade da mensagem. A estrutura das mensagens depende das soluções de integração em que estão envolvidas. Nesse sentido, a mensagem é considerada uma abstração de uma parte da informação que é trocada e transformada [59].

O Guaraná DSL permite a modelagem de diferentes tarefas, estas, que são representadas graficamente por um ícone, o qual está associado à função que essa tarefa desempenha. As tarefas que compõem os processos são o seu principal elemento, as mesmas são responsáveis pelo processamento e modificação das mensagens. Desse modo, uma tarefa é responsável por ler uma mensagem do *slot* de entrada, processar e escrever no *slot* seguinte [7]. De acordo com Frantz et al. [25], as tarefas são classificadas de acordo com a sua semântica em: *router*, *modifier*, *transformer*, *timer*, *stream dealer*, *mapper* e *communicator*:

- *Router*: Direcionam uma mensagem de entrada para zero, um ou mais destinos. Dentre as tarefas deste grupo, destacam-se: o *filter* é utilizado para retirar mensagens do fluxo, de acordo com seus critérios. O *replicator* é responsável por fazer cópias de uma mensagem de entrada quando a solução requer enviá-la para dois ou mais destinos.

- *Modifier*: Tem a função de modificar o conteúdo original de uma mensagem de entrada. Essa alteração ocorre quando são acrescentadas informações ao conteúdo, dessa forma o conteúdo da mensagem é traduzindo para outro formato ou até mesmo é responsável por produzir novas mensagens. Neste contexto, a mensagem que chega não é a mesma mensagem que vai continuar no fluxo, após passar por uma tarefa modificadora. O *context-basead slimmer* recebe uma mensagem de entrada e busca mais informações em um recurso externo baseada no conteúdo da mensagem original e acrescenta à ela.

- *Transformer*: Tem a função de modificar a estrutura de uma mensagem ou ainda de construir mensagens novas a partir de outras. O *translator* tem a função de traduzir mensagens de um formato para outro, uma

vez que é necessário quando se integra aplicativos que geralmente trabalham com diferentes formatos de mensagens.

- *Timer*: Executam ações relacionadas ao tempo. O *delayer* atrasa uma mensagem por um determinado período de tempo. O *ticker* produz uma mensagem em intervalos regulares.
- *Stream Dealer*: Tarefas que trabalham com um fluxo de bytes e ajudam a comprimir/descomprimir, criptografar/descriptografar ou codificar/decodificar mensagens. O *zipper* comprime uma mensagem. O *unzipper* descomprime uma mensagem.
- *Mapper*: Tarefas mapeadoras mudam o formato das mensagens que processam, por exemplo, a partir de um fluxo de bytes em um documento XML. O *XML2Stream* mapeia um fluxo de bytes para uma mensagem XML. O *stream2XML* mapeia uma mensagem XML para um fluxo de bytes.
- *Communicator*: Tarefas do tipo comunicadoras são utilizadas nas portas para interagir com componentes de comunicação, geralmente, conhecidos também como adaptadores. O *In Communicator* é usado em portas de entrada para ler mensagens. O *out communicator* é usado em portas de saída para gravar mensagens.

2.3 Simulação de Eventos Discretos

A simulação é o campo de pesquisa que lida com a experimentação de modelos que permitem fazer previsões sobre o comportamento e o desempenho de sistemas reais. Salienta-se, que existem diferentes modelos de simulação, nesta pesquisa, será abordado o modelo discreto, este tipo de modelo utiliza-se de eventos orientados, que são utilizados para modelar sistemas que mudam seu estado em momentos distintos no tempo a partir da ocorrência de eventos [56].

A seguir, a Seção §2.3.1 trata dos conceitos de sistema, modelo e simulação, a Seção §2.3.2 apresenta as vantagens e desvantagens da simulação e a Seção §2.3.3 traz a classificação dos sistemas de simulação.

2.3.1 Sistema, Modelo e Simulação

Um sistema é um agrupamento de partes que operam juntas, visando um objetivo em comum [21]. Cassandras e Panayiotou [9] abordam um sistema como uma combinação de componentes que atuam conjuntamente

para a realização de uma função que não é possível realizar com qualquer componente individual.

A Figura §2.8 apresenta as diferentes formas de estudo de um sistema. Inicialmente, existem duas possibilidades: experimento com o sistema real ou experimento com um modelo do sistema.

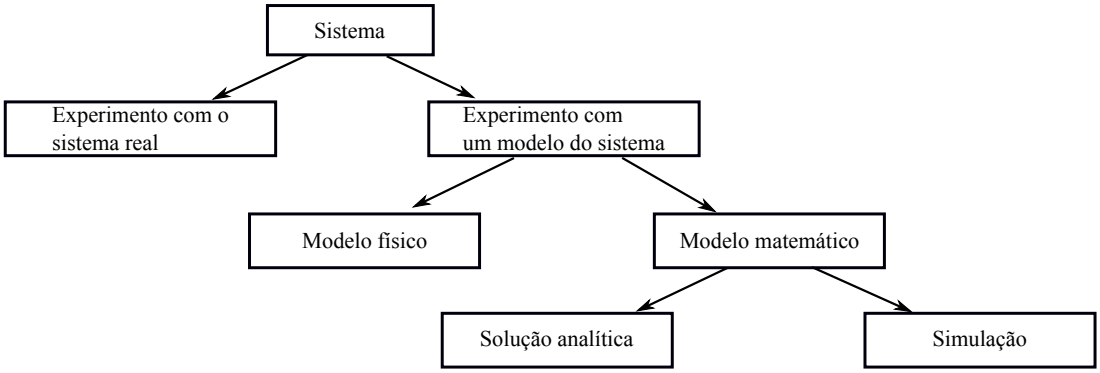


Figura 2.8: *Abordagens de Estudo de um Sistema. (Law e Kelton [42])*

Os sistemas reais ocorrem por meio de experimentos e medições no sistema. Isso ocasiona uma demanda maior de tempo, custos elevados e ocorrências indesejadas no sistema. Já a outra opção é a realização de experiências mediante o modelo do sistema [59].

Sob esta perspectiva, um modelo pode ser definido como a representação das relações dos componentes de um sistema, sendo considerada como uma abstração, no sentido em que tende a se aproximar do verdadeiro comportamento do sistema [10].

Os modelos do sistema podem ser físicos ou matemáticos. Modelos físicos são empregados para projetar protótipos em escala. O modelo matemático, segundo Chwif [10], pode ser reduzido a um conjunto de equações que, ao serem resolvidas, permitem obter a solução esperada. Embora estes modelos forneçam soluções precisas, se o sistema a ser modelado for extremamente complexo, as soluções podem se tornar complicadas e, em muitos casos, têm de ser utilizadas hipóteses simplificadoras para que se resolva o modelo analiticamente. Isto pode levar à perda da validade do modelo, visto que o mesmo não conseguiu representar, satisfatoriamente, a realidade [10].

Ainda de acordo com Chwif [10], é possível dizer que existe o "bom" modelo ou "mau" modelo. Um "bom" modelo, de uma maneira geral, é aquele que permite atingir os objetivos da simulação, com o mínimo de custo. Segundo Gladwin e Tumay [27], o "bom" modelo deve possuir certas características, das quais, duas estão dispostas abaixo:

- É válido - no sentido de representar satisfatoriamente a realidade;
- É mínimo - no sentido de incluir somente elementos que influenciam no problema a ser solucionado.

Nesse contexto, constata-se que é mediante a avaliação do desempenho que pode-se resolver problemas relacionados a sistemas computacionais. Esses problemas estão agrupados nos seguintes tópicos: a comparação de sistemas, a identificação de gargalos, a caracterização de cargas de trabalho, a configuração de sistemas e a previsão de desempenho [6].

A literatura aborda a avaliação de desempenho sob três perspectivas. Todas possuem vantagens e desvantagens. Para escolher a abordagem mais apropriada para um caso específico, é fundamental considerar os recursos disponíveis para a análise do problema e o nível desejado de exatidão dos resultados. Essas abordagens são: modelagem analítica, simulação e medição [4].

A modelagem analítica utiliza modelos matemáticos, ou seja, conjunto de fórmulas e/ou algoritmos para descrever e analisar numericamente as medidas de desempenho de um sistema. Este tipo de modelagem oferece soluções precisas, porém, se o sistema a ser modelado for muito complexo, as soluções podem também tornarem-se complicadas [4].

Os modelos de simulação, por sua vez, são implementados com o auxílio de um computador. Este tipo de modelo utiliza-se de uma linguagem de programação para ser representado. Estes modelos são executados, ao invés de solucionados como no caso dos analíticos. A simulação envolve a geração de uma situação abstrata do sistema baseada no modelo de simulação, e a partir desta situação abstrata a inferência de como o sistema real funcionaria, permitindo fazer previsões sobre o desempenho dos sistemas reais [23]. Quanto às desvantagens, elas podem ser difíceis de se construir e podem levar a resultados equivocados, quando comparados aos analíticos. Com relação as suas vantagens, são modelos excelentes para representar sistemas que possuem um número muito grande de variáveis e com dinâmica muito complexa.

A medição é realizada sobre o sistema real, mediante a utilização de códigos-fonte instrumentados, podendo, ser *softwares* de medição, *hardwares* de medição ou uma mescla de *softwares* de medição e *hardwares* de medição. Quanto às desvantagens, ela apresenta um custo muito alto de implementação e não pode ser utilizada em todas as fases do sistema. A sua vantagem é a de ser considerada potencialmente precisa [4].

Segundo dos Santos [16], simulação é a imitação, durante determinado período de tempo, da operação de um sistema ou de um processo do mundo real. A simulação envolve a geração de uma situação abstrata do sistema baseada no modelo de simulação, e a partir desta situação abstrata a inferência de como o sistema real funcionaria, permitindo fazer previsões sobre o comportamento e o desempenho dos sistemas reais.

O comportamento do sistema é estudado pelo desenvolvimento de um modelo de simulação. Este modelo normalmente toma a forma de um conjunto de restrições relacionadas à operação do sistema. Estas restrições podem ser expressas através de relações matemáticas, lógicas e simbólicas entre as entidades ou objetos de interesse do sistema. Uma vez construído e validado, um modelo pode ser usado para investigar uma série de questões do tipo e se... sobre o sistema do mundo real e possíveis gargalos [10]. O comportamento de um sistema de simulação tem como base um modelo de simulação. Após ser construído o modelo de simulação, é possível que ele seja validado mediante a simulação. A partir da validação do modelo, torna-se possível a identificação de gargalos de desempenho, visando diminuir custos com a implementação, riscos e tempo.

Para a construção de um modelo de simulação, é necessário partir dos objetivos da simulação. Torna-se, pois, importante considerar as medidas de desempenho, ou seja, as variáveis de saída de interesse do modelo de simulação. Também deve-se distinguir, ao analisar um modelo de simulação, três elementos básicos. A entidade, atributo, e atividade. Qualquer objeto envolvido com o modelo é chamado entidade. A propriedade desta entidade é chamada atributo. Qualquer processo que gera uma alteração no modelo é chamada de atividade. Por exemplo, em um modelo de simulação de eventos discretos, o banco de entidades representaria os clientes que chegam no sistema; os atributos seriam o seu equilíbrio e crédito pessoal disponível, e o serviço fornecido por um departamento específico deste banco, seria uma atividade. Nesse sentido, um modelo de simulação de eventos discretos caracteriza-se por reproduzir as atividades envolvidas no sistema, visando prever seu comportamento e desempenho, levando em consideração que cada evento ocorre em um determinado momento particular no

tempo provocando uma mudança de estado no sistema. É possível formular modelos de simulação de eventos discretos sob três maneiras: (a) tendo como entrada a memória descritiva e a descrição do processo utilizado pelas entidades do sistema; (b) tomando como entrada a descrição completa das atividades das entidades envolvidas no sistema, e (c) ao definir as alterações que podem ocorrer nos estados em cada momento do evento. O modelo de simulação seria implementado utilizando estratégias de simulação [56].

A Figura §2.9 apresenta o processo de simulação, entendido como um método científico, em que, primeiramente, devem-se formular hipóteses, seguidas do preparo do modelo, na sequência é possível testar estas hipóteses através do modelo e analisar se elas foram validadas levando em consideração os resultados obtidos. Destaca-se que a simulação requer uma linguagem formal do problema analisado, pois a visualização gráfica do modelo conceitual proposto é importante para uma melhor compreensão do problema estudado, bem como apontar possíveis gargalos.

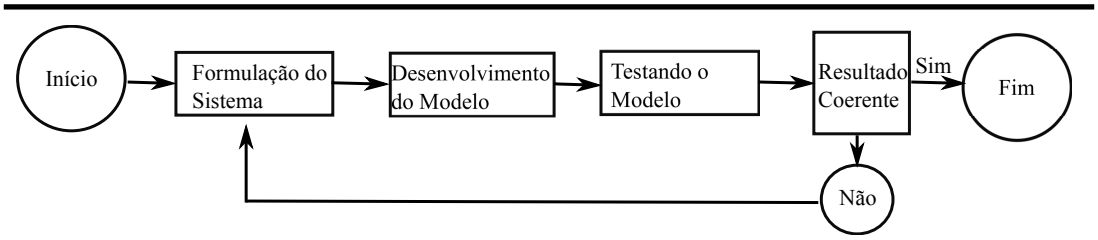


Figura 2.9: *Método Científico Aplicado à Simulação.*

Neste sentido, a elaboração de um modelo de simulação, segundo Paul e Balmer [50], apresenta três grandes etapas, conforme ilustra a Figura §2.10:

Figura 2.10: *Etapas de Elaboração de um Modelo. (Adaptada de Wiesner [59])*

Na primeira etapa, deve-se entender claramente o sistema a ser simulado, seus objetivos e as possibilidades de estudo de um sistema, como a decisão a respeito da abrangência do modelo, o nível de detalhes e todas as

hipóteses estabelecidas. Após estas decisões, é realizada a formulação do modelo conceitual. Nesta etapa, também devem ser coletados os dados de entrada [59].

Por sua vez, na segunda etapa, o modelo conceitual é convertido em um modelo de simulação equivalente, o qual é testado para averiguar se o modelo de simulação é uma representação precisa da realidade, ou seja, se o mesmo está dentro dos objetivos da simulação [59].

Na terceira e última etapa, após a verificação do modelo de simulação, está tudo pronto para a realização dos experimentos, dando origem ao modelo experimental. Nesse sentido, são efetuados vários testes do modelo e os resultados da simulação são analisados, podendo ser satisfatórios ou não [59].

Salienta-se que através da simulação, é possível identificar possíveis gargalos de desempenho, ainda na fase de projeto, diminuindo custos e tempo que poderiam ser desperdiçados. Os resultados obtidos com a simulação, trazem uma melhor precisão na análise de gargalos e fornecem informações úteis para possíveis melhorias. Dessa forma, a Seção seguinte trata das vantagens e desvantagens da simulação.

2.3.2 Vantagens e Desvantagens da Simulação

A simulação possui vantagens e desvantagens. Inúmeros autores abordam o assunto, tendo opiniões diversas. Com relação às vantagens, Gold [28] diz que a simulação surge como uma solução quando utiliza-se de um modelo executável dos requisitos em conjunto as técnicas matemáticas, resultando na diminuição dos custos e do tempo. Para Chwif [10], além da diminuição dos custos, a experimentação de um sistema real envolve riscos, tanto materiais quanto humanos, o que não ocorre com um sistema simulado. Segundo dos Santos [16], as novas políticas, procedimentos operacionais, regras de negócio, fluxos de informação podem ser estudadas sem se alterar o mundo real; novos equipamentos, *layouts*, sistemas de transporte podem ser testados sem se comprometer recursos na sua aquisição; hipóteses sobre como e porque certos fenômenos ocorrem podem ser testados visando verificar sua praticabilidade; o tempo pode ser comprimido ou expandido permitindo acelerar ou retardar o fenômeno. De acordo com Sakurada e Miyake [54], as linguagens de simulação oferecem abertura para gerar modelos de simulação para os mais variados tipos de sistema, oferecendo vantagens como requerimento de tempo relativamente menor para construção do modelo e maior facilidade de utilização apoiada em menus e gráficos amigáveis.

Quanto às desvantagens, pode-se dizer que a construção do modelo de simulação requer um treinamento especial. É uma arte que é aprendida com tempo e experiência. Segundo dos Santos [16], a simulação não dá resultados exatos. Para Sakurada e Miyake [54], as desvantagens também estão associadas à necessidade de conhecimentos bastante específicos de programação para a construção de modelos mais complexos. Além disso, os simuladores revelam desvantagens como menor flexibilidade para representar detalhes do sistema real e restrições para controlar a realização de experimentos sob condições muito específicas.

2.3.3 Classificação dos Modelos de Simulação

É imprescindível, para modelar e simular sistemas, saber identificar se o sistema que está sendo analisado, representa um modelo determinístico ou estocástico, estático ou dinâmico, discreto ou contínuo. A Figura §2.11 apresenta a classificação dos modelos descritos:

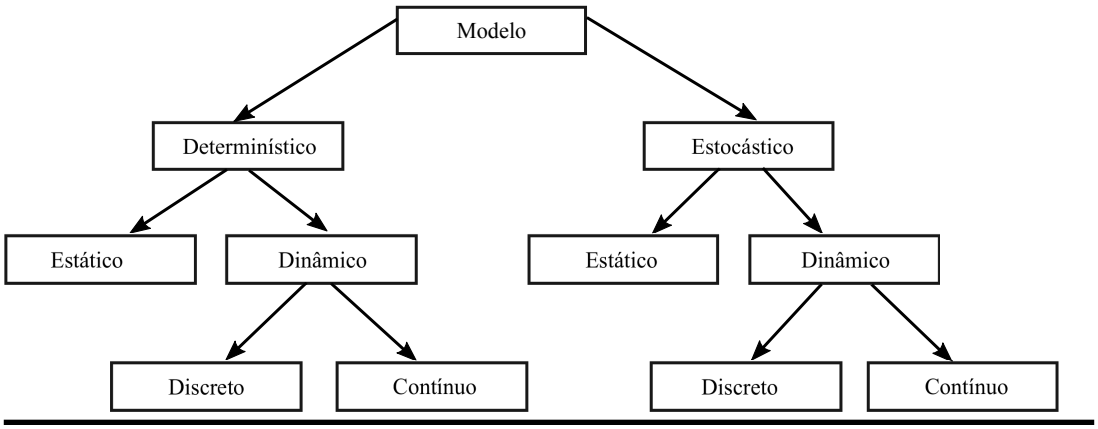


Figura 2.11: *Classificação dos Modelos dos Sistemas. (Law e Kelton [42])*

Estes modelos de sistemas foram propostos por Law e Kelton [42], que classificaram os modelos em determinísticos ou estocásticos. Considera-se um modelo determinístico, aquele que trabalha somente com variáveis do tipo não probabilísticas, sendo que o resultado da simulação é sempre o mesmo, não importando quantas vezes se "rode" o modelo [10].

Em contrapartida, um modelo estocástico ou probabilístico não obedece a um padrão determinístico de entradas e saídas, mas aleatório, geralmente caracterizado por uma distribuição probabilística que melhor representa o

fenômeno real estudado [54]. Em termos de comparação, geralmente, os modelos estocásticos são mais complexos, mas representam melhor um sistema real do que os modelos determinísticos.

Quanto ao modelo de simulação estático, Law e Kelton [42] dizem que é uma representação de um sistema em um determinado momento. Para este tipo de simulação, a variável tempo não é considerada um fator determinante, não influenciando no modelo.

Para Law e Kelton [42], a simulação de um modelo dinâmico, considera a aleatoriedade e a interdependência das variáveis. Este modelo é influenciado pelo tempo, pois representa um sistema que evolui no decorrer do tempo. É devido a isso que este modelo possibilita uma melhora na capacidade da simulação em prever o comportamento real do sistema.

Ainda, quanto à classificação, o modelo dinâmico pode ser contínuo ou discreto. Segundo Law e Kelton [42], nos sistemas contínuos, as variáveis de estado mudam continuamente no tempo. Constata-se que, nesse modelo, as variáveis que descrevem o sistema têm seus valores alterados de forma contínua em incrementos de tempos iguais.

Com relação ao modelo de simulação discreto, o mesmo mantém as variáveis de estado inalteradas no decorrer do tempo e alteram seus valores somente em momentos específicos. Assim, segundo Sawicki et al. [56], os modelos discretos são voltados a eventos e assim usados para modelar sistemas que mudam seu estado em momentos distintos no tempo a partir da ocorrência de eventos. As soluções de integração podem ser caracterizadas como sistemas discretos para a razão que todos os componentes envolvidos em uma solução de integração consumam um tempo de execução específico quando ocorre um evento. Assim, de qualquer modo pode mudar o estado da solução de integração. Como um sistema discreto, o modelo conceitual projetado para uma solução de integração pode ser traduzido em um modelo de simulação e simulado, tendo a vantagem de técnicas e ferramentas bem estabelecidas para a simulação de eventos discretos.

Neste contexto, o modelo de simulação desenvolvido pode ser simulado, utilizando técnicas e ferramentas para a simulação, reduzindo custos, riscos e tempo. A simulação de sistemas de eventos discretos é uma ferramenta que pode ser utilizada, pois consegue avaliar sistemas complexos, levando em consideração seu comportamento dinâmico. Além disso, consegue prever o futuro comportamento do sistema, pois permite identificar e reproduzir as variáveis relacionadas ao desempenho e sua forma de interagir com os outros elementos, tornando-se possível a construção de hipóteses

através da observação dos modelos. Outro fator a considerar é o tempo, que pode ser controlado, a fim de analisar o comportamento do sistema [56].

2.4 Distribuição de Probabilidade

A distribuição de probabilidade é utilizada para representar sistemas aleatórios. Assim, os modelos de simulação devem ser capazes de imitar durante um determinado período de tempo, o comportamento estocástico do sistema real.

Segundo Martins [47], distribuição de probabilidade (ou modelo de probabilidade) de uma variável aleatória é um modelo matemático que se idealiza para estudar o fenômeno aleatório em causa.

A distribuição de probabilidade caracteriza-se por associar uma probabilidade a um dado resultado numérico de um experimento, isso significa, que uma distribuição de probabilidade proporciona a probabilidade de cada valor de uma variável aleatória. O uso de métodos estatísticos é importante, pois permite que se analise uma determinada amostra que é uma parte do que está sendo analisado e que a partir disso se possa tirar algumas conclusões [47].

Para Martins [47], existem dois tipos de distribuições de probabilidade que correspondem a diferentes tipos de dados ou variáveis aleatórias: a distribuição discreta e a distribuição contínua.

A distribuição é dita discreta quando a variável que está sendo analisada só pode assumir valores particulares e que sejam finitos. Exemplo de uma distribuição discreta são os valores inteiros 0 e 1, etc.

Já a distribuição é dita contínua quando a variável que está sendo analisada é expressa por um número infinito de valores. Exemplo de uma distribuição contínua é a temperatura, que é um elemento medido numa escala contínua, ou seja, uma variável aleatória.

Segundo Correa [12], existem características numéricas que são muito importantes em uma distribuição de probabilidade de uma variável aleatória discreta. São os parâmetros das distribuições, a saber:

- Esperança matemática (ou simplesmente média), $\bar{x}$ é um número real, e também uma média aritmética, representada pela equação matemática §2.2:

$$\bar{x} = \frac{\sum(f_i * x_i)}{n}. \tag{2.2}$$

- Variância, s^2 é a medida que dá o grau de dispersão (ou de concentração) de probabilidade em torno da média. O fato de conhecer a média de uma distribuição de probabilidades já ajuda bastante, porém precisa-se de uma medida que dê o grau de dispersão de probabilidade em torno dessa média, representada pela equação matemática §2.3:

$$s^2 = \sum_{i=1}^{n} \frac{f_i * x_i^2}{n} - (\bar{x}). \tag{2.3}$$

- Desvio Padrão, S é a raiz quadrada da variância, representada pela equação matemática §2.4.

$$S = \sqrt{S^2}. \tag{2.4}$$

Dentre as distribuições de probabilidade discretas, pode-se citar: Bernoulli, Binomial, Poisson, Uniforme Discreta, Geométrica, Hipergeométrica, Multinomial, entre outras.

Quanto às distribuições de probabilidade contínuas, pode-se citar: Uniforme, Normal, Exponencial, Log-Normal, Gama, Qui-Quadrado, entre outras.

Na modelagem de sistemas de Redes de Petri, as distribuições de probabilidade são usadas para representar o comportamento aleatório do tempo de processamento de cada mensagem e na tarefa filtro. A distribuição de probabilidade comumente utilizada para representar o tempo de processamento de cada mensagem e a tarefa filtro é a Distribuição Uniforme Discreta.

A Distribuição Uniforme Discreta é a mais simples de todas as distribuições discretas de probabilidade. É aquela, na qual, a variável aleatória assume todos os seus valores com a mesma probabilidade [19].

Definição: A variável aleatória X, diz-se que tem Distribuição Uniforme Discreta em, N, pontos quando tem função probabilidade §2.5:

$$f(x_i) = P(X = x_i) = \frac{1}{N}, para\ todos\ os\ valores\ possiveis\ x_i, i = 1, ..., n. \tag{2.5}$$

2.5 Redes de Petri

Propostas por Carl Adam Petri em 1962, as Redes de Petri (RdP) constituem-se em um formalismo matemático que permite modelar o comportamento de sistemas, descrevendo as relações existentes entre condições

e eventos e ainda permitem estudar propriedades tais como paralelismo, sincronização e compartilhamento de recursos [2].

Para Francês [22], Rede de Petri é uma ferramenta de modelagem que permite a representação de sistemas, utilizando como alicerce uma forte base matemática. Esta ferramenta possui a particularidade de permitir modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos.

As Redes de Petri são grafos bipartidos, pois apresentam dois tipos de nós, chamados de lugares e transições, principais componentes da sua representação gráfica. Os lugares equivalem às variáveis de estado do sistema e são representados por círculos ou elipses. As transições equivalem às ações realizadas pelo mesmo e são representadas por barras ou retângulos.

Os grafos das Redes de Petri são dirigidos sempre de lugares para transições e/ou transições para lugares. Estes lugares e transições são ligados por arcos. Assim, um arco liga lugar à transição, e transição ao lugar. Os arcos são representados por setas que indicam o sentido do fluxo do sistema modelado. Outro fato a considerar é de que os arcos de uma Rede de Petri jamais realizam conexões diretas entre transições ou lugares [1].

A Figura §2.12 apresenta os elementos básicos de uma Rede de Petri. Quando os lugares estão conectados a uma transição no sentido lugar-transição, denominam-se lugar de entrada, porém, quando os lugares estão conectados a uma transição no sentido transição-lugar, denominam-se lugar de saída.

Figura 2.12: *Elementos Básicos de uma Rede de Petri. (Francês [22])*

A Figura §2.12 retrata uma Rede de Petri básica, que contém os elementos principais de uma Rede de Petri, em que os lugares são representados por P0 (lugar 1) e P1 (lugar 2), a transição T0 e os arcos interligando os lugares a transições e as transições a lugares.

Uma Rede de Petri também possui *tokens*, que se encontram nos lugares. Quando as transições são ativadas consomem *tokens* dos lugares que as

alimentam e os geram nos lugares alimentados por elas. Os *tokens* são representados por pontos pretos dentro dos lugares e funcionam como indicadores de fluxo. Quanto à quantidade de *tokens* nos lugares e sua posição, os mesmos oscilam de acordo com o funcionamento da rede.

Neste contexto, são as transições que podem destruir e criar os *tokens* contidos nos lugares. Os arcos ligados a cada transição indicam exatamente sobre que lugares atuam. Um arco com origem em um lugar e término em uma transição (a partir daqui designado por arco de entrada), indica que essa transição ao ser disparada subtrai um *token* desse lugar. De forma simétrica, um arco com origem em uma transição e fim em um lugar (daqui em diante designado por arco de saída), indica que essa transição adiciona, quando do seu disparo, um *token* a esse lugar. Os arcos indicam o sentido do movimento dos *tokens* de um lugar para outro, atravessando a transição. Com isso, uma transição só pode disparar se cada lugar de entrada tiver, pelo menos, um *token*, de forma a ser retirado no disparo da transição, pelo respectivo arco. Quando tal sucede diz-se que a transição está habilitada. O disparo de uma transição é instantâneo, ou seja, as ações são efetuadas ao mesmo tempo [1].

Há vários tipos de Redes de Petri usadas para modelar sistemas. Segundo Roos-Frantz et al. [53], uma Rede de Petri estocástica é formada pelos elementos (P, T, I, O, M_0, $\bigwedge$), onde P é o conjunto de lugares, T é o conjunto de transições, I é o conjunto de arcos de entrada, O é o conjunto de arcos de saída, M_0 é a marcação inicial, $\bigwedge$ é o conjunto de taxas da transição. A taxa de transição é um atraso no disparo para determinada transição por uma variável aleatória X, com uma função densidade de probabilidade exponencial negativa, a função $F_{Xi}(x)= 1\text{-}e^{-\lambda_i x}$ onde $-\lambda_i$ é a taxa de disparo da transição t_i. A taxa de disparo é dependente da marcação e a média de atraso no disparo da transição t_i na marcação m_j é $[\lambda_i(M_j)]^{-1}$.

O conceito de Redes de Petri é baseado na ideia de que é o menor número de elementos de estados e eventos que mudam as condições dos elementos do estado. Com isso, a ideia central é que um elemento recebe dados de seu ambiente, processa e, assim, produz novos dados, que são liberados no ambiente. Ao fazê-lo, o componente não precisa saber a origem ou destino dos seus dados.

A Figura §2.13 apresenta uma Rede de Petri para modelar as condições representadas por três variáveis de estado (lugares), e por três eventos (transições): amanhecer, entardecer e anoitecer. Para representar a situação atual, ou seja, em que condições encontra-se o sistema modelado, utiliza-se uma marca grafada (um ponto) no lugar que corresponde a essa situação, por exemplo, a condição atual é manhã [45].

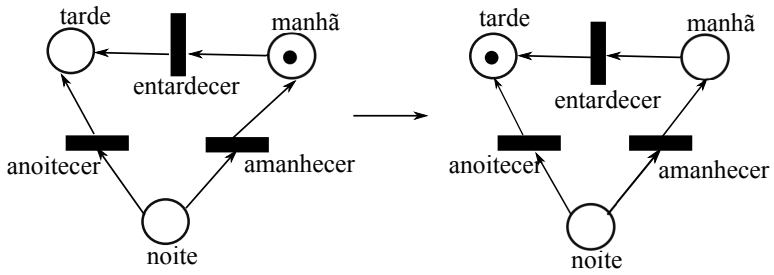


Figura 2.13: *Condições de Lugares e Transições. (Maciel et al. [45])*

Nesse modelo, tem-se a condição atual representada pela marca no lugar manhã. Nesta condição, o único evento que pode ocorrer é entardecer que é representado pela transição de mesmo nome. Na ocorrência desse evento, tem-se uma nova situação, ou seja, tarde.

Quanto à matemática nas Redes de Petri, afirma Bressan [5] que a notação matricial é uma das formas de representar uma Rede de Petri. A partir disso, pode-se analisar a Figura §2.14:

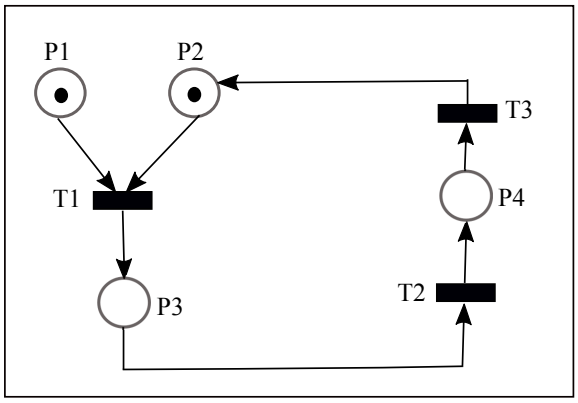


Figura 2.14: *Exemplo de Rede de Petri para Notação Matricial. (Bressan [5])*

A notação matricial pode ser feita usando três tipos de matrizes, sendo:

- Matriz de Entrada (E): ETxP representa a quantidade de arcos de entrada em cada transição;

- Matriz de Saída (S): STxP representa a quantidade de arcos de saída em cada transição;
- Matriz de Incidência (I): ITxP = STxP (Matriz de Saída) ETxP (Matriz de Entrada).

A partir da Figura §2.14 gerou-se as matrizes que estão ilustradas na Figura §2.15:

	P1	P2	P3	P4
T1	1	1	0	0
T2	0	0	1	0
T3	0	0	0	1

Matriz de Entrada

	P1	P2	P3	P4
T1	0	0	1	0
T2	0	0	0	1
T3	0	1	0	0

Matriz de Saída

	P1	P2	P3	P4
T1	-1	-1	1	0
T2	0	0	-1	1
T3	0	1	0	-1

Matriz de Incidência

Figura 2.15: *Notação Matricial da Rede de Petri. (Bressan [5])*

Segundo da Penha et al. [13], é através da notação matricial que torna-se possível modelar uma Rede de Petri graficamente, pode-se extrair a quantidade de lugares, transições e arcos. No entanto, somente estes elementos não são sufucientes para um modelo completo. Ainda é necessário os *tokens* e os pesos.

Os *tokens* podem ser representados por meio de uma marcação inicial. Assim, é possível informar o valor V_i. Sendo V_i= (1, 1, 0, 0). Quanto ao peso de cada arco, na Figura §2.14, é igual a 1.

Ainda, quanto a matemática inserida nas Redes de Petri, segundo Maciel et al. [45], as mesmas possuem uma equação fundamental, a qual, descreve o comportamento das redes, proporcionando a análise de propriedades comportamentais e estruturais. Dessa forma, a Equação Fundamental das Redes de Petri é dada por §2.6:

$$M'(p) = M_0(p) + C.\bar{s}, \forall_{pi} \in P \tag{2.6}$$

De acordo com Francês [22], as Redes de Petri podem ter como enfoque três fundamentações, que levam a diferentes modelos formais, sendo que cada uma delas tem suas peculiaridades e utiliza diferentes teorias matemáticas como suporte. A primeira utiliza a teoria *bag*, a segunda baseia-se nos conceitos da álgebra matricial e a última busca aporte teórico na estrutura definida por relações.

Segundo Maciel et al. [45], que utiliza a teoria *bag*, a Rede de Petri é composta por cinco partes: o conjunto de lugares P, o conjunto de transições T, o *bag* de entrada I, o *bag* de saída O e a capacidade associada a cada lugar K. Para cada transição existe uma função de entrada, que é um mapeamento de uma transição tj em um *bag* de lugares $I(t_j)$. De forma semelhante, as funções de saída mapeiam uma transição tj em uma *bag* de lugares $O(t_j)$. Para denotar conjuntos, utiliza-se { } e para *bags* "[ ]".

Exemplo, de acordo com Maciel et al. [45], a rede dos períodos do dia:

A rede dos períodos do dia, conforme apresenta a Figura §2.13 é formada por três lugares e três transições, que representam respectivamente as variáveis de estados e as ações realizadas pelo sistema. As pré-condições para a realização das ações são representadas pelos *bags* $I(t_i)$, ou seja, para que o amanhecer possa acontecer, é necessário que a condição noite seja verdadeira, dado que I(amanhecer)={noite}. Os lugares que são pós-condições das transições (t_i) são representados pelos *bags* $O(t_i)$. A transição amanhecer tem como lugar de saída O(amanhecer)={manhã} [45].

$(R_{periodos\ do\ dia})$= (P, T, I, O, K)

onde

o conjunto de lugares P é

P={manhã, tarde, noite},

o conjunto de transições T é,

T={amanhecer, entardecer, anoitecer},

o conjunto de *bags* de entrada I é

I={I(amanhecer)= [noite], I(entardecer)= [manhã], I(anoitecer)= [tarde],},

o conjunto de *bags* de saída O é

O={O(amanhecer)= [manhã], O(entardecer)= [tarde],

O(anoitecer)= [noite],},

e o conjunto de capacidades dos lugares K é

K={K_{manha}=1, K_{tarde}=1, K_{noite}=1}.

A segunda baseia-se nos conceitos da álgebra matricial, nesta definição a Rede de Petri R é uma quíntupla R= (P, T, I, O, K), onde P é um conjunto finito de lugares T é um conjunto finito de transições. I: PxT é a matriz de pré-condições. O: PxT é a matriz de pós-condições. K é o vetor das capacidades associadas aos lugares. Se o conjunto de lugares ou o conjunto de transições é vazio, a rede é dita degenerada conforme Maciel et al. [45].

Exemplo:

Aqui, apresenta-se a estrutura da rede do exemplo períodos do dia, segundo a representação matricial da Figura §2.16. O conjunto de lugares e transições são representados respectivamente por P e T. As matrizes I e O representam as pré-condições e as pós-condições, respectivamente, de todas as transições da rede. A coluna amanhecer da matriz I mostra que o lugar N noite é pré-condição de amanhecer. De forma semelhante, observamos na coluna amanhecer da matriz O que o lugar manhã é pós-condição da transição amanhecer. Estas duas matrizes representam a estrutura do modelo [45].

I=

amanhecer	entardecer	anoitecer	
0	1	0	manhã
0	0	1	tarde
1	0	0	noite

O=

amanhecer	entardecer	anoitecer	
1	0	0	manhã
0	1	0	tarde
0	0	1	noite

Figura 2.16: *Redes de Petri Matriciais. (Maciel et al. [45])*

A última busca aporte teórico na estrutura definida por relações, nesta definição a Rede de Petri R também é uma quíntupla R= (P, T, A, V, K), composta

pelo conjunto de lugares P, o conjunto de transições T, o conjunto dos arcos interligam lugares às transições ou transições aos lugares, a valoração ou peso dos arcos representada por V e o conjunto das capacidades dos lugares K [45].

Exemplo de acordo com Maciel et al. [45]:

A estrutura da Rede de Petri do exemplo períodos do dia é

($R_{periodos\ do\ dia}$)= (P, T, I, O, K), onde

o conjunto P lugares é:

P={manhã, tarde, noite},

o conjunto de transições T é

T={amanhecer, entardecer, anoitecer},

o conjunto de arcos A é

A={(manhã, entardecer), (entardecer, tarde),

(tarde, anoitecer), (anoitecer, noite)

(noite, amanhecer), (amanhecer, manhã)},

o conjunto de valorização dos arcos V é

V={1, 1, 1, 1, 1, 1}

e o conjunto de capacidades dos lugares K é

K={K_{manha}=1, K_{tarde}=1, K_{noite}=1}.

As Redes de Petri são consideradas processos estocásticos, ou seja, qualquer tipo de evolução temporal que seja analisável em termos de probabilidade. Uma maneira de classificar as Redes de Petri é quanto ao seu grau de abstração, pois elas podem ser separadas em Redes de Petri de baixo nível e Redes de Petri de alto nível [46].

As Redes de Petri de baixo nível são aquelas cujo significado de suas marcas não são diferenciáveis a não ser pela estrutura da rede à qual estão associadas. Destaca-se que na década de setenta este foi o tipo de rede mais utilizada, por permitir uma compactação razoável dos modelos. Já as Redes de Petri de alto nível, são aquelas cujas marcas incorporam alguma semântica, viabilizando sua diferenciação. Nesse sentido, considera-se semântica como sendo a atribuição de valores ou cores às marcas, ou a adoção de noções de tipos de dados abstratos [46].

Existem extensões que podem abranger as Redes de Petri de baixo nível e as de alto nível, as quais buscam incluir hierarquias e aspectos temporais. Dentre elas merecem destaque: as extensões coloridas que tem por objetivo reduzir o tamanho do modelo, permitindo que seus *tokens* sejam individualizados; as extensões temporizadas que congregam aspectos temporais determinísticos aos modelos e as extensões hierárquicas que visam representar modelos de sistemas complexos de forma mais compreensível [46].

2.5.1 Redes de Petri Coloridas

As Redes de Petri surgiram no início da década de 60 e rapidamente tornaram-se populares nas empresas e academicamente. Este sucesso é resultado de sua aplicabilidade na modelagem e análise de sistemas. Durante a década de 70, começaram a ser apontadas algumas limitações das Redes de Petri. Primeiramente, verificou-se que, por não suportarem o conceito de dados, os modelos gerados eram extremamente grandes e complexos. Outro ponto negativo é de que não havia suporte a características hierárquicas, ou seja, era impossível modelar um sistema através de vários módulos, conectados entre si através de interfaces. Na década seguinte, as pesquisas realizadas sobre as Redes de Petri apresentaram ao mundo novas extensões conhecidas como redes de alto nível, que resolviam os problemas citados. Dentre os novos membros da família das redes de alto nível, destacavam-se as Redes de Petri Coloridas [14].

O objetivo das Redes de Petri Coloridas é reduzir o tamanho do modelo, permitindo que os *tokens* sejam individualizados, através de cores atribuídas a eles, assim, diferentes processos ou recursos podem ser representados em uma mesma rede. As cores não significam apenas cores ou padrões. Elas podem representar tipos de dados complexos, usando a nomenclatura de colorida apenas para referenciar a possibilidade de distinção entre os *tokens* [46].

Conforme Maciel et al. [45], uma Rede de Petri Colorida é definida como RdPC= (R, $\sum$, C, G, E, In), onde R= (P, T, A), P é um conjunto de lugares, T é um conjunto de transições, A é um conjunto de arcos. Os elementos de A são arcos que conectam transições a lugares ou lugares a transições (A $\subseteq$ (PxT) $\bigcup$ (TxP)). $\sum$ é o conjunto finito não-vazio de tipo, denominado cores. C é a função de coloração C : P $\longrightarrow$ Cor, G é a função de guarda G : T $\longrightarrow$ exp, onde exp é uma expressão tal que $\forall t_i \in T$ | Tipo(G(t))= bool e o Tipo(Var(G(t))) $\in \sum$, E é uma função associada aos arcos E : A $\longrightarrow$ exp, onde $\forall$a C A | Tipo (E(a))= C e o tipo (Var(E(a))) $\in \sum$ e In é a expressão de inicialização, onde $\forall p_i \in P$ | Tipo(In(p))= C(p) e Var(In(p))= $\emptyset$.

Quanto ao tipo de ligação em uma Rede de Petri Colorida, para uma transição t $\in$ T com Var(t) = $v_1, ..., v_n$ define-se o tipo de ligação por $BT(t) = Tipo(v_1)$ x ...x $Tipo(v_n)$ [45].

No que diz respeito ao conjunto de ligações: define-se o conjunto de ligações de uma transição por $B(t) = (c_1..., c_n)$, $\in$ BT(t) ou mesmo por $B(t) = G(t) < v_1 = c_1, ..., v_n = c_n >$ [45].

Com relação à marcação, a distribuição de marcas é uma função M: P $\longrightarrow$ $\sum$ e uma marcação de uma rede colorida é uma distribuição de marcas [45].

A marcação inicial das redes coloridas é obtida através da avaliação da expressão de inicialização para cada lugar da rede, ou seja, $M_0(p) = IN(p) <> \forall$ p $\in$ P [45].

A distribuição de ligações é uma função Y: T $\longrightarrow$ B. Denomina-se elemento de Y o par (t,b), onde b é uma ligação, tal que b $\in$ Y(t). Denomina-se por passo (*step*) uma distribuição de ligações não-vazia [45].

Um passo (Y= (t,b)) está habilitado quando o número de marcas de cores correspondente à ligação do passo é maior ou igual à avaliação da expressão do arco que interliga os lugares de entrada à transição t [45].

Quanto ao passo habilitado, um passo se diz habilitando se, e somente se, $\sum_{t,b \in Y}$ E(p,t) <b> $\leqslant M(p), \forall$ p $\in$ P [45].

Um passo habilitado pode ocorrer e a ocorrência provoca uma alteração da marcação da rede, pela remoção das marcas dos lugares de entrada e adição de marcas aos lugares de saída [45].

Quanto à ocorrência de um passo, seja M_0 uma marcação e Y um passo habilitado para essa marcação. A ocorrência do passo Y altera a marcação da rede para uma marcação M_1 = M_0 - $\sum_{t,b \in Y}$ E(p,t) <b> + $\sum_{t,b \in Y}$ E(p,t) <b>, onde E(p,t) e T(t,p) correspondem às expressões dos arcos de entrada e saída da transição (em t) do passo, respectivamente [45].

Diz-se que uma uma marcação M'' é diretamente acessível de M' se a partir de M' e pela ocorrência do passo Y alcança-se M'', ou seja, M' [Y > M''] [45].

Um conceito importante para o estudo das Redes de Petri Coloridas é segundo Jensen [34], o multiconjunto. Este conceito possibilita o armazenamento de marcas individualizadas em um mesmo lugar. Um multiconjunto é um conjunto que apresenta uma particularidade: pode haver múltiplas ocorrências de uma mesma marca. Como exemplo, seja um conjunto A = a,

b, c, e que se deseje adicionar um elemento c ao conjunto A. Pela regra da união de conjuntos, tem-se: A = a, b, c, mas em se tratando de multiconjuntos, ter-se-ia A = a, b, c, c. Como definição formal, um multiconjunto é $\sum$ (s pertence S) m(s')S, no qual: S é um conjunto não vazio e; m(s) é o número de ocorrências do elemento s no multiconjunto m. A Figura §2.17 apresenta a propriedade de adição de multiconjuntos. O primeiro conjunto possui marcas do tipo a, c, e e, enquanto o segundo conjunto possui marcas do tipo a, b, c, e. O conjunto resultante é o somatório de cada tipo de marcas presentes em cada um dos conjuntos iniciais.

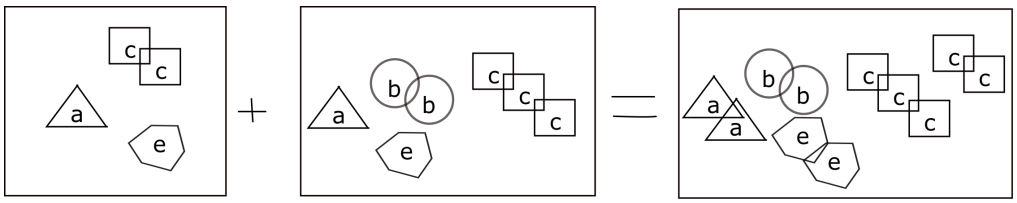


Figura 2.17: *Propriedade de Adição de Multiconjuntos. (Jensen [34])*

Com o formalismo exposto, pode-se concluir que as fichas expressam informações complexas, aumentando o poder de descrição das Redes de Petri Coloridas, permitindo a obtenção de modelos mais compactos. Esta versatilidade possibilita ao projetista distribuir a complexidade do modelo de um sistema entre as inscrições, declarações à estrutura da rede. Portanto, uma Rede de Petri Colorida compõe-se de três partes distintas: estrutura, declarações e inscrições [34].

A estrutura é formada por lugares, transições e arcos direcionados. As declarações compreendem a especificação dos conjuntos de cores (domínios), declarações de variáveis e operações (funções) usadas nas inscrições. Já as inscrições variam de acordo com o componente da rede e podem ser de quatro tipos: (i) cores dos lugares, determinam a cor associada ao lugar e um lugar só pode comportar fichas cujos valores respeitem sua cor; (ii) guardas, conhecidas como expressões booleanas que restringem a ocorrência das transições; (iii) expressões dos arcos, utilizadas para manipular a informação contida nas fichas e; (iv) inicializações, que estão associadas aos lugares e estabelecem a marcação inicial da rede [34].

Como as Redes de Petri, as Redes de Petri Coloridas também são grafos direcionados e bipartidos. Porém, ao invés de pesos inteiros, são associadas aos arcos inscrições que determinam quantas e quais fichas devem ser

removidas ou adicionadas aos lugares associados, na ocorrência de uma transição. Inscrições, denominadas guardas, podem ser também associadas às transições. Guardas restringem a ocorrência de transições a determinadas condições. O estado inicial de uma Rede de Petri Colorida também é determinado por inscrições associadas aos lugares. Cada inscrição é, em geral, uma expressão construída a partir de constantes, variáveis e operadores previamente definidos. Uma Rede de Petri Colorida também possui um conjunto de declarações para indicar a natureza dos elementos citados nas diversas inscrições, à semelhança de uma área de declarações de uma linguagem de programação qualquer [43].

As inscrições e declarações de uma Rede de Petri, em geral, podem ser escritas em praticamente qualquer linguagem com sintaxe e semântica bem definidas. Já para as Redes de Petri Coloridas têm sido utilizadas em associação com uma linguagem denominada CPN-ML derivada da linguagem *Standard* ML. A abreviação ML, tem sua origem do inglês *Marking Language*, cuja sintaxe é semelhante a utilizada por linguagens de programação convencionais [43].

Com relação à teoria das Redes de Petri Coloridas, é utilizada a expressão conjunto de cores, que substituem os tipos de dados, assim, cada valor é denominado cor, que pode ser (inteiro, real, lista, etc.). Assim, a cada lugar na estrutura interna, é associado um conjunto de cores, que indica o tipo de fichas que o lugar pode conter [43].

Quanto às variáveis de transição, estas tratam-se de um conjunto de variáveis presentes nas inscrições dos arcos e na guarda da referida transição. Já uma ligação é a substituição de cada variável da transição por uma cor (valor). No que tange às cores, estas devem pertencerem aos conjuntos de cores apropriados e que impliquem a avaliação da guarda como verdadeira [43].

No tocante a cada marcação, a ocorrência de uma transição sob uma determinada ligação é habilitada, se todos os seus lugares de entrada tiverem fichas suficientes para satisfazer às expressões dos arcos. Cada expressão deve ser devidamente avaliada segundo as substituições determinadas pela ligação, a fim de determinar quantas e quais fichas são requeridas nos lugares de entrada. Caso a transição ocorra, então são retiradas fichas dos lugares de entrada e depositadas novas fichas nos lugares de saída. A quantidade de fichas é determinada também pela avaliação das expressões dos arcos, segundo as substituições implicadas pela ligação [43].

É importante salientar que este tipo de Rede de Petri é muito utilizada, pois, é capaz de modelar sistemas grandes e complexos, devido a quantidade

de recursos que dispõe, possibilitando uma redução nos tamanhos dos modelos. Sendo assim, para uma melhor compreensão, torna-se importante observar a Figura §2.18, que apresenta um exemplo de Rede de Petri Colorida, em que os arcos são rotulados com cores (a, b e c). Para que uma transição desta rede esteja habilitada, é necessário que os lugares de entrada desta transição tenham marcas do tipo (cor) associados ao arco que interliga estes lugares a transição. A transição t_0 não se encontra habilitada, pois o lugar P_0 não possui uma marca da cor a, contudo a transição t_1 está habilitada, pois o lugar P_1 possui uma marca da cor a e o lugar P_2 possui marcas da cores a, b, o que satisfaz as condições rotuladas nos arcos que interligam estes lugares à transição t_1. O disparo desta transição retira as marcas das cores associadas aos arcos, dos lugares de entrada e adiciona aos lugares de saída marcas de cor igual à cor associada ao arco que interliga a transição aos lugares de saída. Neste caso, uma marca de cor c é depositada no lugar P_4 [45].

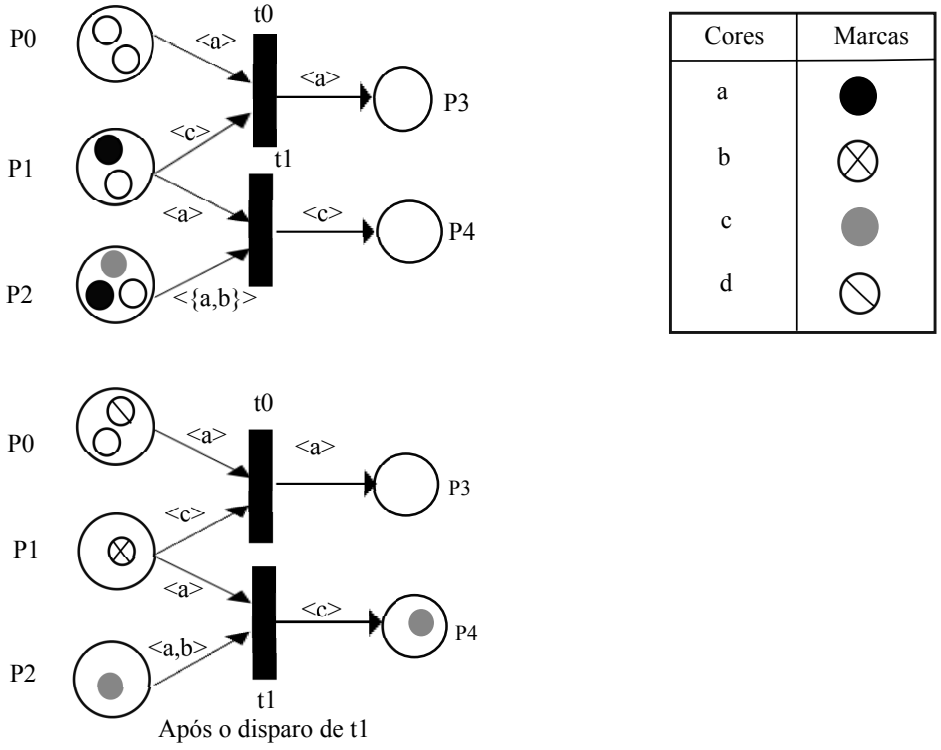


Figura 2.18: *Redes de Petri Coloridas. (Maciel et al. [45])*

Na rede apresentada acima, as marcas são identificadas por cores, contudo as marcas podem ser estruturas mais complexas.

2.5.2 Redes de Petri Temporizadas

Para este estudo, além das Redes de Petri Coloridas, é possível a representação de sistemas dinâmicos a eventos discretos, com atividades concorrentes e assíncronas. Entretanto, para que seja possível especificar sistemas de tempo real, avaliar o desempenho de sistemas dinâmicos de um modo geral e examinar questões referentes ao seu escalonamento, por exemplo, é necessário que se considerem informações relativas ao tempo em que ocorrem os eventos no sistema considerado.

Segundo Marranghello [46], a associação do tempo a componentes da rede pode se realizar de várias maneiras, sendo elas:

- O tempo associado a lugares. Os *tokens* (após o disparo) estarão disponíveis a disparar uma nova transição após um certo tempo que está associado ao lugar;
- O tempo associado às marcas. O tempo indica quando a marca ficará disponível para o disparo;
- O tempo associado às transições. Utiliza tempos associados às transições (subentende-se que está se referindo às Redes de Petri Temporizadas com tempos associados às transições).

Este tipo de Rede de Petri pode ser determinística ou estocástica. As determinísticas surgiram na primeira metade da década de setenta e indicam tempos absolutos relativos à execução dos eventos correspondentes. As estocásticas, por sua vez, permitem considerar incertezas nos instantes de execução de eventos do sistema associando a eles funções de probabilidade para a determinação de sua execução [46].

Salienta-se que, nesta pesquisa, utilizou-se o tipo de Rede de Petri Temporizada Determinística. Segundo Marranghello [46], uma Rede de Petri Temporizada é considerada uma septupla TPN = (L, T, F, C, P, M_0, I_e), onde os primeiros seis termos correspondem a uma rede lugar/transição normal:

R= (L, T, F), que corresponde a uma rede base

C : L $\rightarrow \aleph + \smile\propto$, função capacidade

P : F $\smile$ ℵ + $\smile$, função peso

M_0 : L $\smile$ ℵ, marcação inicial

E o sétimo termo é uma função, I_e: T $\smile$ ρ x (ρ $\smile$ $\propto$), que associa um intervalo fechado [α, β] a cada transição, chamado de intervalo de disparo estático. O limite inferior, α, do intervalo é o tempo de disparo inicial (TDI) estático e o limite superior, β, do intervalo é o tempo de disparo final (TDF) estático. ρ é o conjunto dos números racionais. O TDI é indicador do tempo mínimo que a transição deve esperar para executar e o TDF corresponde ao tempo máximo de espera para o disparo da transição. Dessa forma, as transições estariam associadas a um intervalo de disparo estático de [0, $\propto$] [46].

Para uma melhor compreensão, é importante a observação de um exemplo de Rede de Petri Temporizada, como apresenta a Figura §2.19, que, trata de uma transição com disparo atômico, as marcas permanecem nos lugares de entrada das transições até o tempo associado à transição ter sido consumido e então as marcas são removidas dos lugares de entrada e imediatamente armazenadas nos lugares de saída [45].

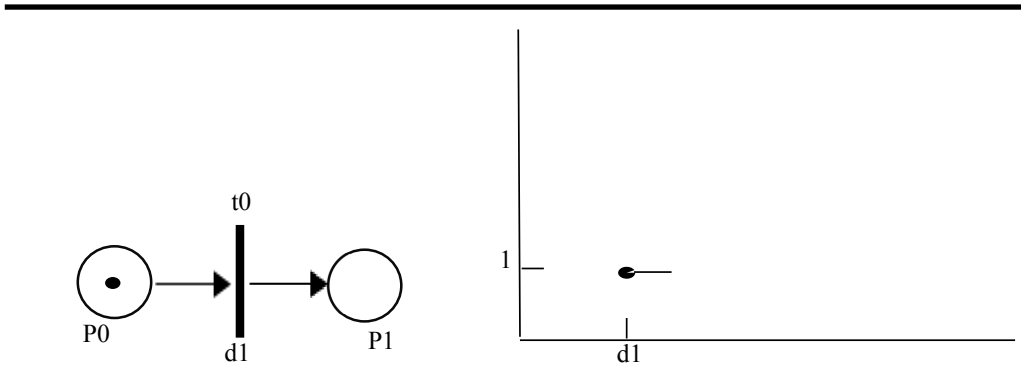


Figura 2.19: *Redes de Petri Temporizadas. (Maciel et al. [45])*

Nas Redes Temporizadas com disparo atômico, uma transição ao ser habilitada, um contador com o tempo associado à transição é iniciado. O contador é decrementado a uma taxa constante. Quando o contador atinge o valor zero, a transição dispara [45].

Constata-se através do conceito de Redes de Petri e das suas diferentes extensões como as já relacionadas, Coloridas e Temporizadas, que pode-se

relacionar importantes áreas que utilizam-se destes conceitos, como: automação de escritórios e de manufatura, avaliação de desempenho, bancos de dados, circuitos integrados, protocolos de comunicação, sistemas distribuídos e sistemas de produção, evidenciando assim, a sua real importância e necessidade.

2.6 Trabalhos Relacionados

A utilização de formalismos matemáticos, como as Redes de Petri na modelagem e simulação de sistemas de eventos discretos, vêm aumentando em diversas áreas do conhecimento. Esta ocorrência deve-se a sua possibilidade de aplicação na modelagem, sendo que sua notação gráfica possibilita analisar o desempenho de sistemas, como no caso desta pesquisa, que trata de sistemas de eventos discretos. Outro fato que vem a colaborar é que a semântica de uma Rede de Petri é matematicamente definida, proporcionando formas de estudo de sistemas, como no caso, a identificação de gargalos de desempenho. A seguir, serão abordados alguns trabalhos que contribuiram para o desenvolvimento desta pesquisa. Estes trabalhos estão relacionados ao uso das Redes de Petri para encontrar gargalos de desempenho por meio da simulação de sistemas de eventos discretos. Relatam as características dos sistemas, como os mesmos foram modelados e apresentam os resultados obtidos com a simulação.

O trabalho apresentado por Yamada et al. [60] sugere efetuar simulações das etapas de funcionamento de uma indústria de cana de açúcar, utilizando, para isso, as Redes de Petri. O sistema da indústria abrange ações de recebimento, descarregamento e a movimentação de matéria prima até o processo de extração. Com a simulação, visa-se analisar propostas de otimização e identificação de gargalos de desempenho sem interferir no funcionamento do sistema real. O processo de recebimento de cana de açúcar e alimentação da central de moagem possuem características discretas, e o tempo de cada execução interfere no desempenho do sistema. A construção do modelo parte do estudo das atividades do processo. Essas atividades constituem-se da identificação de uma entrada que representa a chegada da matéria-prima transportada por caminhões, os quatro caminhos possíveis que representa os quatro tipos de descarga, encaminhamento da cana de açúcar para a central de moagem que representa a saída do sistema e o veículo é encaminhado ao setor de pesagem concluindo a operação.

Com a simulação do modelo desenvolvido por Yamada et al. [60], verificou-se a presença de paralelismo de atividades, o qual acarretaria

uma sobrecarga na alimentação da central de moagem em situações de chegadas sucessivas de matéria-prima. Nesse sentido, sentiu-se a necessidade do sincronismo de atividades, já que os recursos de descarregamento são compartilhados entre os setores. Torna-se fundamental uma alimentação constante de matéria-prima para o beneficiamento. O modelo não considera suas características determinísticas e estocásticas, embora tenha denotado as variáveis do sistema. O autor ainda sugere a utilização das Redes de Petri Temporizadas para diferenciar cada etapa do processo e consolidar a validação do modelo.

O trabalho apresentado por Roos-Frantz et al. [53] parte de um estudo de caso, fundamentado num problema real de integração que trata de um projeto para automatizar o registro de novos usuários em um repositório único do Conselho do Condado de Huelva. Propõe automatizar o processo de registro de novos usuários de um sistema através da integração de seis aplicações independentes, implementadas em plataformas diferentes e em sua maioria desenvolvidas sem a preocupação com integrações futuras. Nesse sentido, o estudo parte de um modelo conceitual desenvolvido na plataforma Guaraná, a partir, do qual foi desenvolvido um modelo de simulação modelado por Redes de Petri Estocásticas.

Segundo Roos-Frantz et al. [53], consideram que a abordagem sobre simulação pode melhorar a qualidade de soluções de integração de aplicações empresariais desenvolvidas com o Guaraná. E ainda, que a partir da simulação pode-se identificar informações relacionadas ao desempenho da solução.

O trabalho apresentado por Ramos e de Oliveira [52] refere-se a uma abordagem de modelagem utilizando Redes de Petri Coloridas para especificação e verificação formal de um modelo de sistema tutor inteligente (STI), que utiliza a aprendizagem baseada em problemas (PBL) como estratégia pedagógica. A especificação e a verificação formal permitem verificar se as funcionalidades planejadas do modelo pedagógico são realizadas, antes da etapa de implementação do sistema. Experimentos indicam consistência geral e benefícios da proposta. O trabalho foi realizado utilizando a ferramenta de simulação CPN Tools.

Segundo Ramos e de Oliveira [52], a modelagem apresenta como benefícios a possibilidade de especificar, verificar e simular as principais funcionalidades do sistema, a viabilidade de ter as Redes de Petri funcionando em paralelo com o STI, realizar simulações para buscar identificar novas alternativas que o sistema pode oferecer, tais como: ajuda, orientação, verificação

do funcionamento do modelo e análise de desempenho do sistema. Dentre as principais contribuições, destaca-se a identificação do estado mais apropriado para oferecer ajuda ou cooperação, evitando que o aprendiz permaneça muito tempo em um caminho errado. A PBL foi a estratégia pedagógica escolhida porque é uma abordagem que tem sido muito discutida.

O trabalho realizado por Miyagi et al. [49] apresenta a modelagem de sistemas de saúde com abordagem das Redes de Petri interpretadas, aplicada aos serviços do ambulatório do Hospital das Clínicas (HC) de São Paulo. A modelagem tem como objetivo estudar o fluxo de pacientes que procuram atendimento do ambulatório, pacientes não urgentes e, principalmente, pacientes que procuram atendimento pela primeira vez, com intuito de auxiliar a tomada de decisão pela gerência do sistema e analisar a evolução do sistema e o comportamento dos setores. O modelo foi desenvolvido pela metodologia PFS (*Production Flow Schema*) e MFG (*Mark Flow Graph*), interpretações de Redes de Petri, que evidenciam os recursos envolvidos e a movimentação dos componentes do sistema (pessoas, equipamentos, informações) no qual os detalhes do sistema vão sendo gradativamente inseridos no modelo para uma melhor compreensão, considerando a evolução discreta do sistema. Foram colhidos dados na unidade de pronto atendimento do HC, identificação do fluxo de pacientes, e com base nas informações obtidas pelos funcionários do HC, o modelo foi sendo construído de acordo com as possíveis sequências de serviços que os pacientes são submetidos, dando ênfase para pacientes que procuram o atendimento pela primeira vez.

A simulação do modelo proposto por Miyagi et al. [49] considerou sugestões de alteração do fluxo de pacientes, identificou a possibilidade da eliminação de alguns processos que sobrecarregavam setores do sistema e sugeriu novos processos. Por fim, juntamente com uma equipe multidisciplinar, o trabalho de modelagem e simulação do sistema de pronto atendimento do HC evidenciou constatações de como a teoria de Redes de Petri exige um conhecimento superficial em relação a outras técnicas de simulação, como a metodologia PFS usada na modelagem, auxiliados pelo recurso da animação do fluxo de itens do sistema proposta pelo simulador que permite a análise em determinados setores num instante qualquer e possibilita avaliar a situação.

O trabalho de Junqueira e Miyagi [36], propõem a modelagem e simulação de sistemas produtivos distribuídos, utilizando o formalismo matemático das Redes de Petri. Esta técnica de modelagem é considerada pelos autores como sendo um recurso fundamental para o projeto, possibilitando a implementação e melhorias de desempenho no sistema produtivo. Este trabalho utiliza-se das simulações, com a utilização de computadores separados

fisicamente, porém integrados com o auxílio de uma rede de comunicação, com o intuíto de avaliar o comportamento de sistemas, visando melhorar os resultados já existentes.

Neste contexto, Junqueira e Miyagi [36], propõem a utilização de um procedimento para a modelagem de sistemas produtivos em ambientes distribuídos. O procedimento foi aplicado com êxito em estudos de caso, nos quais a eficácia deste procedimento foi confirmada. Este trabalho, propõe a utilização de algoritmo para gerenciamento de simulações distribuídas. Neste sentido, os autores apresentaram uma nova abordagem para modelagem de sistemas distribuídos, baseada no conceito de orientação a objeto, com auxílio do formalismo Redes de Petri, associado a um procedimento de refinamento progressivo.

O trabalho proposto por Carvalho et al. [8] propõe um modelo de análise de desempenho de trabalho multifuncional em linhas de produção em forma de U usando o formalismo matemático das Redes de Petri Temporizadas. A problemática abordada trata da relação de tempo de execução das tarefas nos postos de trabalhos, avaliando o tempo de cada operação que compõe a tarefa de determinado posto, a formação de estoques intermediários entre os postos de trabalho, e sua interferência no fluxo produtivo do sistema. Uma linha de produção é composta por diversos postos de trabalho e, partindo desta afirmação, é desenvolvido um modelo em Redes de Petri Temporizadas de um posto de trabalho genérico que representa as ações aplicadas neste posto, o recurso de serviço análogo ao operador e o estoque intermediário (estoque de produtos semiacabados). A partir deste, é realizada a modelagem de uma linha de produção em forma de U com trabalho multifuncional, contendo oito postos de trabalho, sendo determinados três operadores para a linha. Nesse sentido, as Redes de Petri demostraram que modelam facilmente as linhas de produção trabalho multifuncional.

O trabalho proposto por Carvalho et al. [8] apresenta a simulação do modelo de análise de linhas de produção em forma de U, considerando o tempo de execução de cada posto, sob um cenário que determina uma meta de construção de 67 peças em um intervalo de tempo de 60 minutos. A investigação da diferença do tempo de execução de cada processo pode resultar na formação dos estoques intermediários devido ao acúmulo de produtos semiacabados entre os postos de trabalho e levar a ociosidade do trabalho, e por fim comprometer a meta de produção. Como alternativa de solução, é determinada analiticamente carga máxima de cada estoque intermediário e a redistribuição dos operadores em nove setores. Com estas modificações, foram observadas melhoras nas metas de produção nos tempos determinados.

Em seu trabalho Facchin e Sellitto [18] apresentam um modelo em Redes de Petri para medição do inventário e tempo de construção de uma manufatura em uma indústria calçadista. O objetivo é possibilitar a medição, antecipadamente, do inventário e o tempo necessário para produção de uma determinada encomenda. Conhecendo as grandezas antes da liberação do plano, um gestor pode antever e eventualmente prevenir problemas, mudando o plano. Para construção do modelo, foi realizado um mapeamento dos processos, representando cada tarefa que os compõe, atribuídos os respectivos tempos de execução, coletadas por meio de medição mecânica, das transições e agrupadas em um modelo único. Para teste e refino do modelo, foi escolhido um plano já realizado, simulado em computador alimentado com a situação de carga inicial dos processos e com um plano de produção, foi informada a carga de cada processo, resultante das ordens anteriores, no momento de liberação do plano.

Os resultados obtidos por Facchin e Sellitto [18], nas simulações, foram diferentes dos dados reais, sendo observado, por exemplo, o distanciamento de resultados simulados e reais e simulações que representam planos de fabricação longos. Por outro lado, a evolução discreta do sistema modelado e a sintaxe gráfica possibilitam que análises sejam realizadas em todas as posições de manufatura, alternativas de gestão, podem ser simuladas e seus impactos avaliados. Verificou-se que modificações momentâneas na estrutura da manufatura não são captadas individualmente pelo modelo de simulação, devido à simplificação de algumas atividades, negligencia algumas contingências e salienta a necessidade de trata-las de modo mais detalhado para uma possível contribuição para a resolução de problemas menos estruturados. Ao fim, realizou-se uma discussão na qual se explora como os resultados do método podem ser úteis em decisões de gestão, envolvendo o inventário admitido, restrições da manufatura. A logística interna dearmazenagem e movimentação também pode ser apoiada pelo método.

O trabalho apresentado por Cargnin [7] está inserido na área de Integração de Aplicações Empresariais, que trata da integração das aplicações existentes no ecossistema de *software* das empresas por meio de uma solução de integração. A pesquisa propôs analisar o comportamento e a identificação de gargalos de performance de uma solução de integração ainda na fase de projeto. Nesse trabalho, foi desenvolvido um modelo matemático de simulação equivalente ao modelo conceitual da solução, utilizando como base as Redes de Petri.

Nesse trabalho, realizado por Cargnin [7], constatou-se que é possível representar modelos conceituais de soluções de integração de aplicações por

meio das Redes de Petri. Além disso, possibilitou o desenvolvimento de um modelo de simulação, o qual indentificou pontos de formação de filas, possíveis gargalos de desempenho e comportamentos da solução de integração ainda na fase de projeto. O modelo de simulação foi verificado por meio de técnicas de verificação validadas na literatura.

2.7 Resumo do Capítulo

Neste capítulo, abordou-se os conceitos fundamentais para o desenvolvimento desta dissertação. Nesse contexto, foi realizada a revisão da literatura, na qual aprofundou-se os estudos referentes a Integração de Aplicações Empresariais. A linguagem de domínio específico da tecnologia Guaraná, que possibilita projetar soluções de integração com um alto nível de abstração. Foram abordados os conceitos referentes à análise de um sistema por meio da simulação de eventos discretos. Em seguida, foi abordado o conceito básico inerente a distribuição de probabilidade. Por fim, discutiu-se o conceito de Rede de Petri, elementos básicos, teorias matemáticas que dão suporte às Redes de Petri e a abrangência das Redes de Petri utilizadas nesta pesquisa.

Capítulo 3

Modelagem da Solução de Integração

Nossa maior fraqueza é a desistência.
O caminho mais certeiro para o sucesso
é sempre tentar apenas uma vez mais.

Thomas Edison

Este capítulo apresenta o modelo formal de simulação, o qual é uma das contribuições desta pesquisa. A Seção §3.1 apresenta o caso de estudo, considerado na pesquisa. A Seção §3.2 denota a equivalência dos componentes do Guaraná com as Redes de Petri, mediante a tradução dos principais elementos do Guaraná em elementos das Redes de Petri. Na Seção §3.3 é proposto o modelo de simulação, por meio do formalismo matemático das Redes de Petri Coloridas e Temporizadas. Por fim, a Seção §3.4 apresenta o resumo do capítulo.

3.1 Caso de Estudo

A solução de integração tem o intuito de melhorar o atendimento aos alunos no processo de rematrículas da Universidade UNIJUÍ. O objetivo desta solução de integração é disponibilizar automaticamente informações referentes às possibilidades de rematrículas de cada aluno, através da geração de uma lista contendo todas as disciplinas que o aluno ainda não cursou e as que serão ofertadas no próximo semestre. A Seção §3.1.1 apresenta a descrição do ecossistema de *software* da solução de integração. A Seção §3.1.2 realiza uma descrição detalhada do modelo conceitual, considerando o fluxo de trabalho das mensagens na solução.

3.1.1 Ecossistema de Software

A integração de aplicações proposta envolve cinco aplicações: Portal, Cadastro de Alunos, Cadastro de Disciplinas, Sistema de Cobrança e Servidor de *e-mail*. Cada aplicação executa, em plataformas diferentes, o Portal, as aplicações Cadastro de Alunos, Cadastro de Disciplinas e o Sistema de Cobrança são aplicações desenvolvidas pela própria universidade. Estas aplicações não foram desenvolvidas com o objetivo de serem integradas, por isso, para que possam colaborar e dar suporte ao processo de rematrículas, devem ser integradas por meio de uma solução de integração. O Servidor de *e-mail* oferece interfaces POP3 e SMTP. O Portal disponibiliza diversas informações para os alunos, como, por exemplo, as disciplinas já cursadas, percentual do curso realizado. Também é onde ocorrem processos como as rematrículas, de forma que todos os semestres os alunos precisam solicitar, por meio do Portal as disciplinas que desejam cursar no semestre seguinte. O Cadastro de Alunos disponibiliza um banco de dados com informações como nome, RG de aluno e curso matriculado. Cadastro de Disciplinas disponibiliza um banco de dados que contém informações como, nome da disciplina, curso vinculado, semestre, turmas disponíveis e data. Quando um aluno for realizar sua matrícula ou rematrícula, através do Portal, é possível, através das informações adquiridas na aplicação Cadastro de Aluno, consultar na aplicação Cadastro de Disciplinas, quais disciplinas o aluno ainda não realizou e quais disciplinas serão ofertadas no próximo semestre. Com estas informações, será gerada uma lista de possíveis disciplinas a serem cursadas no próximo semestre. A partir da lista de disciplinas, é possível consultar o preço das disciplinas no Sistema de Cobrança e gerar os preços das disciplinas contidas na lista de disciplinas. Esta lista pode ser enviada para o aluno por *e-mail*, utilizando o Servidor de *e-mail*.

3.1.2 Modelo Conceitual de Integração

A Figura §3.1 apresenta o modelo conceitual, proposto por Haugg et al. [31], que representa a solução de integração das aplicações para este problema.

A solução toma informações do Portal por meio da porta assíncrona (1), que lê mensagens contendo as solicitações dos alunos. A tarefa (2) filtra as mensagens recebidas, aceitando somente os pedidos de rematrícula que foram realizadas dentro do período destinado às rematrículas. Então, as mensagens aceitas são encaminhadas para o Cadastro de Alunos através da porta

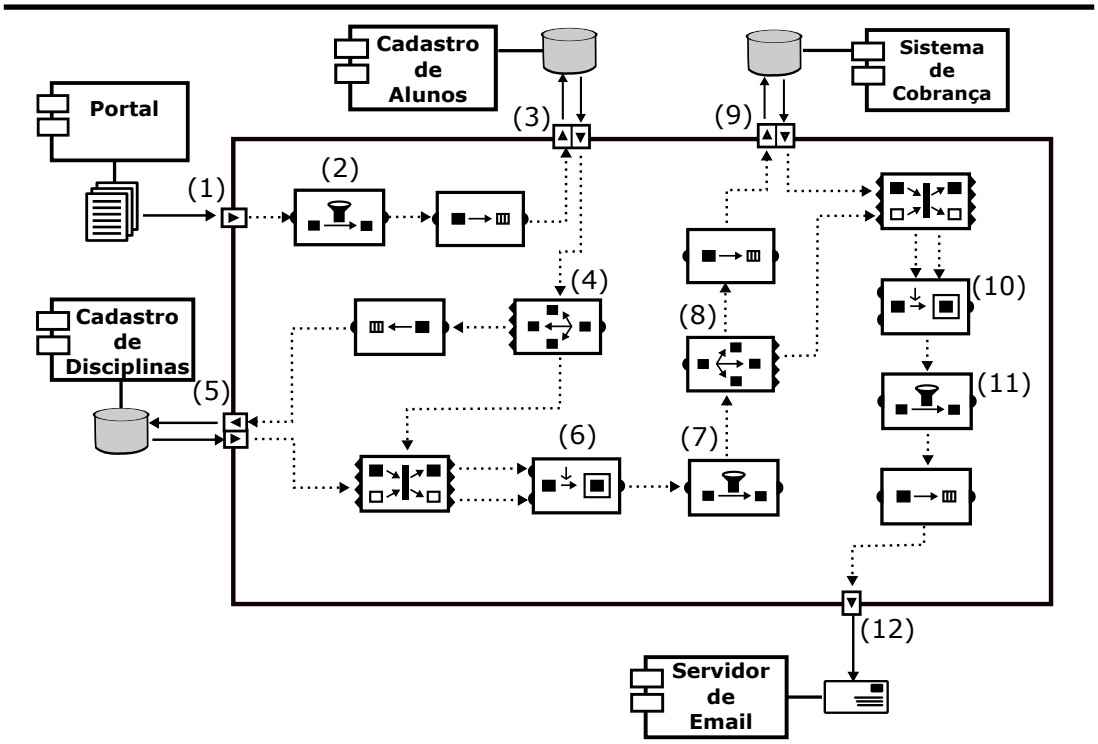


Figura 3.1: *Modelo Conceitual. (Haugg et al. [31])*

síncrona (3), para obter informações referentes ao aluno que realizou a solicitação. A tarefa (4) replica as mensagens, uma cópia é enviada para a porta síncrona (5). Na aplicação Cadastro de Disciplinas é adquirido a lista de disciplinas presentes no curso em que o aluno está matriculado. A tarefa (6) enriquece a segunda cópia realizada pela tarefa (4), correlacionando com as informações retornadas pelo Cadastro de Disciplina. A tarefa (7) tem a função de descartar as disciplinas que o aluno já cursou e também as disciplinas que não serão ofertadas no semestre. Em seguida, a tarefa (8) replica novamente a mensagem. Uma cópia é enviada para a porta assíncrona (9), onde, na aplicação Sistema de Cobrança, será obtido o preço das disciplinas. A tarefa (10) enriquece a segunda cópia realizada pela tarefa (8), correlacionando com as informações recebidas pelo Sistema de Cobrança. As mensagens são encaminhadas primeiramente para a tarefa (11), a qual realiza um filtro para evitar que as mensagens não sejam enviadas para os alunos sem estarem completas. Em seguida, as mensagens são encaminhadas para a porta assíncrona (12) que através da aplicação Servidor de *E-mail* encaminha para o aluno a lista de disciplinas com os preços.

3.2 Equivalência das Redes de Petri com o Guaraná

Pode-se caracterizar as soluções de integração como um sistema de eventos discretos. Neste contexto, é possível a tradução do modelo conceitual de uma solução de integração para o modelo de Redes de Petri. A Tabela §3.1 demonstra a equivalência dos elementos do Guaraná DSL (tarefas, *slots* e mensagens) com os elementos das Redes de Petri (transições, lugares e *tokens*).

Guaraná DSL	Redes de Petri
tarefa	transição
slot	lugar
mensagem	token

Tabela 3.1: *Equivalência dos Elementos. (Roos-Frantz et al. [53])*

O Guaraná possui como construtores: mensagens, tarefas, *slots*, portas e processo de integração. Enquanto que as Redes de Petri possuem: *tokens*, transições e lugares. Pode-se, a partir disso, representar uma solução de integração através do uso do formalismo matemático das Redes de Petri permitindo o desenvolvimento de um modelo de simulação, o qual pode ser simulado e analisado.

Nas Redes de Petri, o arco de entrada interliga um lugar a uma transição e um arco de saída interliga uma transição a um lugar. Quando a transição é disparada, os *tokens* que estão nos arcos de entrada são removidos e adicionados aos lugares interligados aos arcos de saída. Uma transição está habilitada, se a quantidade de *tokens* determinados pelos arcos de entrada existem nos respectivos lugares. A quantidade de *tokens* gerados pelos arcos de saída não é necessariamente a mesma que a removida pelos arcos de entrada. Quando uma transição dispara, a Rede de Petri muda seu estado [53]. A Figura §3.2 apresenta a equivalência da troca de estado em uma solução modelada com Guaraná e uma troca de estado em uma solução de Redes de Petri.

Em uma solução de integração, as mensagens que estão chegando aguardam no *slot*, enquanto a tarefa esta executando. As mensagens seguem uma

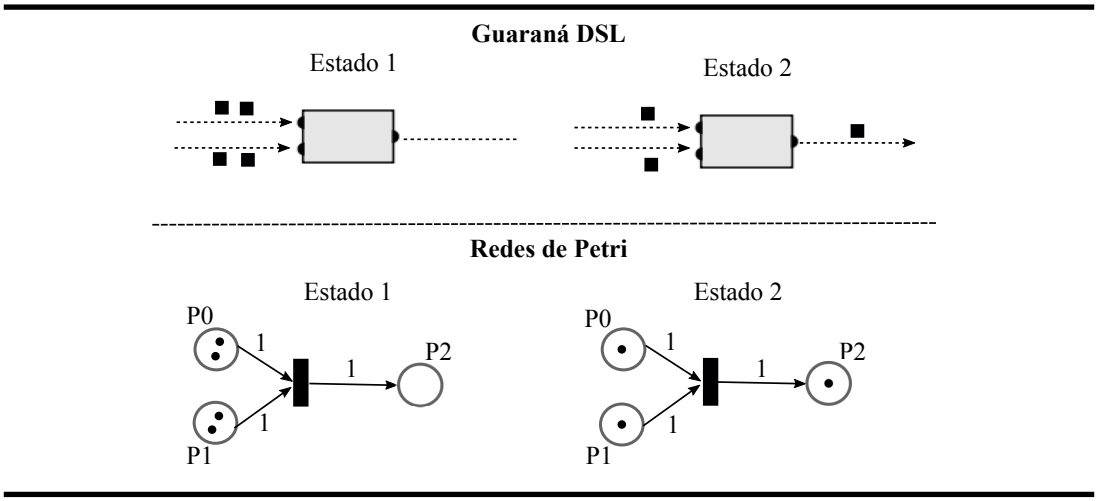


Figura 3.2: *Equivalência da Troca de Estados. (Roos-Frantz et al. [53])*

política de disciplina para serem executadas pelas tarefas. Depois de serem processadas, as mensagens seguem o fluxo da solução de integração.

A disciplina do *slot*, em uma solução de integração, define a ordem de processamento das mensagens. Neste aspecto, esta pesquisa levou em consideração o *slot* como sendo FIFO, isso significa que a primeira mensagem que entra é a primeira que sai.

Busca-se caracterizar a solução de integração que trata do processo de rematrículas da Universidade UNIJUÍ como um sistema de eventos discretos, utilizando para isso, o formalismo matemático das Redes de Petri Coloridas e Temporizadas. Nas Redes de Petri, os lugares junto com os arcos, exercem função semelhante às tarefas e *slots*. Os arcos, são responsáveis por realizar a ligação entre lugares e transições ou transições aos lugares, indicando o fluxo de mensagens.

Contudo, a respeito das tarefas, é importante evidenciar, a equivalência entre a notação gráfica do Guaraná DSL e seu grafo em Redes de Petri. A Tabela §3.2 apresenta o conjunto de tarefas conhecidas como modificadoras, as quais, se caracterizam por modificar o conteúdo original de uma mensagem de entrada [53].

A Tabela §3.3 apresenta o conjunto de tarefas roteadoras, as mesmas são responsáveis por direcionar uma mensagem de entrada para seu respectivo destino, correlacionar, multiplicar e ordenar mensagens [53].

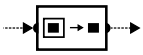
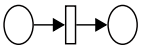
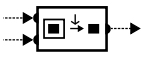
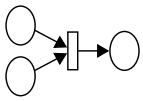
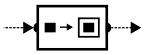
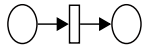
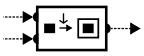
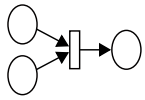
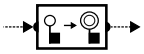
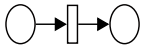
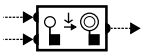
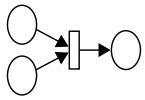
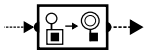
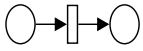
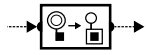
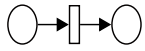
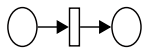
Tarefas	Guaraná	Redes de Petri
Slimer		
Context-based Slimmer		
Content Enricher		
Context-based Content Enricher		
Header Enricher		
Context-based Header Enricher		
Header Promoter		
Header Demoter		
Set Correlation ID		

Tabela 3.2: *Tarefas Modificadoras e Grafo em RdP. (Roos-Frantz et al. [53])*

A Tabela §3.4 apresenta o conjunto das tarefas transformadoras, que têm a função de modificar a estrutura de uma mensagem ou ainda construir mensagens novas [53].

E ainda, a Tabela §3.5 apresenta o conjunto das tarefas temporizadoras, as quais, podem interferir no tempo, antecipando ou atrasando uma execução [53].

A caracterização de uma solução de integração como um sistema de eventos discretos possibilita, fazer conjecturas baseadas nos resultados oriundos da simulação, permitindo, com isso, entender o real comportamento da solu-

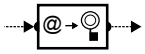
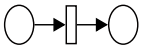
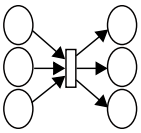
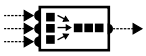
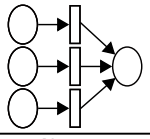
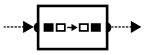
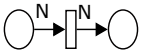
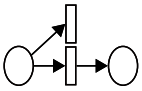
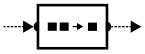
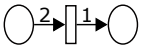
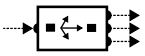
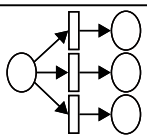
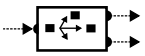
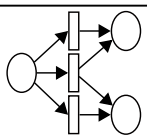
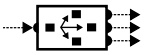
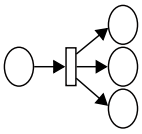
Tarefas	Guaraná	Redes de Petri
Set Return Address		
Correlator		
Merger		
Resequencer		
Filter		
Idempotent Transfer		
Dispacher		
Distributor		
Replicator		

Tabela 3.3: *Tarefas Roteadoras e Grafo em RdP. (Roos-Frantz et al. [53])*

ção de integração. O modelo conceitual da solução de integração, pode ser representado por uma Rede de Petri. As mensagens aguardam no *slot* (lugar) para serem executadas pelas tarefas (transições). O modelo de simulação é desenvolvido com o uso das Redes de Petri, originado da equivalência de elementos apresentada na Tabela §3.1.

Nesta pesquisa, propôs-se a tradução da solução de integração em Guaraná para um modelo de simulação em Redes de Petri Coloridas e

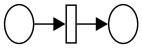
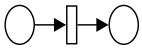
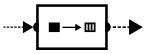
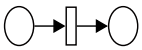
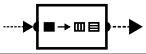
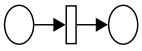
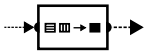
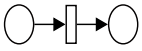
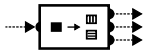
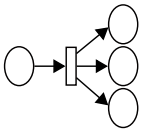
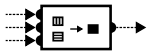
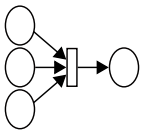
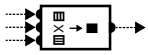
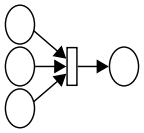
Tarefas	Guaraná	Redes de Petri
Semantic Validator		
Threader		
Translator		
Splitter		
Aggregator		
Chopper		
Assembler		
Cross Builder		

Tabela 3.4: *Tarefas Transformadoras e Grafo em RdP. (Roos-Frantz et al. [53])*

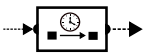
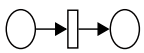
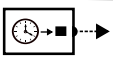
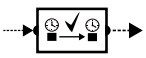
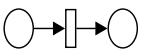
Tarefas	Guaraná	Redes de Petri
Delayer		
Ticker		
Expire Checker		

Tabela 3.5: *Tarefas Temporizadoras e Grafo em RdP. (Roos-Frantz et al. [53])*

Temporizadas. Dentre seus principais benefícios estão: (a) as Redes de Petri proporcionam análises do comportamento da solução; (b) possuem seus

construtores (lugares, transições e arcos) comuns a inúmeras ferramentas de simulação de Redes de Petri; e (c) é um dos formalismos matemáticos adequado para a tradução deste problema de pesquisa em um modelo formal de simulação [53].

3.3 Modelo de Simulação Proposto

Os elementos da solução de integração e suas características apresentam equivalência com os elementos e as características das Redes de Petri. Portanto, o comportamento de uma solução de integração pode ser analisado por meio das Redes de Petri, mediante à simulação.

Para obter um modelo de simulação equivalente ao modelo conceitual do processo de rematrículas da Universidade UNIJUÍ, considerou-se a equivalência dos elementos do Guaraná com os elementos das Redes de Petri. Além disso, observou-se as características e funcionalidades dos *slots* e das tarefas da solução.

Para traduzir a solução de integração em Guaraná para o modelo de simulação em Redes de Petri, foi necessário o uso da Tabela §3.6 que apresenta as tarefas que constituem o caso de estudo, seu grafo equivalente em Redes de Petri e a descrição do papel que desempenham. Essa tabela é usualmente conhecida como Pedra de Roseta, a qual converte cada componente do Guaraná em Rede de Petri equivalente.

A partir da utilização da Tabela §3.6 foi possível desenvolver o modelo de simulação equivalente a solução de integração, com a utilização do formalismo matemático das Redes de Petri. Este modelo de simulação é apresentado na Figura §3.3.

Este modelo de simulação é equivalente a solução de integração que trata do processo de rematrículas da Universidade UNIJUÍ. O fluxo de *tokens* inicia-se pela transição (P1), que toma informações do Portal, que identifica *tokens* que contém as solicitações dos alunos. A transição (T1-A e T1-B) filtra os *tokens* recebidos, aceitando somente os pedidos de rematrículas que foram realizados dentro do prazo. Os *tokens* aceitos seguem o fluxo até a transição seguinte, que é a do tradutor (T2), cuja função é transformar um *token* em outro, encaminhando-o a aplicação Cadastro de Alunos, representada pela transição (P2), de modo a obter informações referentes ao aluno. A transição (T3) replica os *tokens*, uma cópia é enviada a transição seguinte que é a do tradutor (T4), cuja função é transformar o *token* em outro, de forma que a

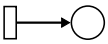
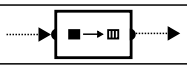
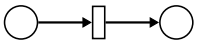
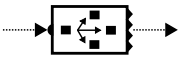
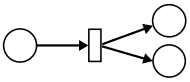
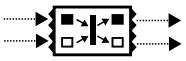
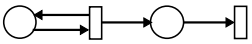
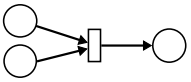
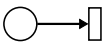
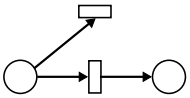
Conceito	Descrição	Notação em Guaraná	Notação em Redes de Petri
Porta de Entrada	Local por onde as mensagens entram		
Tradutor	Transforma uma mensagem em outra		
Replicador	Replica as mensagens para todas as duas saídas		
Correlacionador	Organiza as mensagens de acordo com a sua identidade		
Enriquecedor de Conteúdo	Organiza e completa uma mensagem		
Porta de Solicitação	Solicita uma mensagem de uma aplicação externa		
Porta de Saída	Local por onde as mensagens saem		
Filtro	Filtra as mensagens		

Tabela 3.6: *Tradução dos Elementos do Guaraná DSL em Redes de Petri*

aplicação entenda, encaminhando-o a aplicação Cadastro de Disciplinas, representada pela transição (P3), onde é adquirida a lista de disciplinas (já cursadas, que serão ofertadas, percentual do curso já realizado) e outra cópia é enviada a transição correlacionador (T5), a qual é responsável por correlacionar os *tokens* que retornam da aplicação Cadastro de Disciplinas com os *tokens* que o replicador (T3) envia. A transição (T6) enriquece a segunda cópia realizada pela transição (T3), correlacionando com as informações retornadas pelo Cadastro de Disciplina. A transição (T7-A e T7-B) tem a função de descartar as disciplinas que o aluno já cursou e também as disciplinas que não serão ofertadas no semestre. A transição (T8) replica novamente o *token*, que segue o fluxo para a transição tradutor (T9) (transforma uma mensagem em outra de forma que a aplicação entenda). Uma cópia é enviada para a aplicação Sistema de Cobrança, representada pela transição (P4), que fornece o custo das disciplinas e outra cópia é enviada a transição correlacionador (T10), a qual é responsável por correlacionar o *token* que retorna da aplicação Cadastro de Disciplinas com o *token* que o replicador envia. A transição

(T11) enriquece a segunda cópia realizada pela transição (T8), correlacionando com as informações recebidas pelo Sistema de Cobrança. Os *tokens* são encaminhados primeiramente para a transição (T12-A e T12-B), que filtra *tokens* incompletos. Os *tokens* seguem o fluxo para a transição tradutor (T13) (transforma uma mensagem em outra de forma que a aplicação entenda). Em seguida, os *tokens* são encaminhados para a aplicação Servidor de E-mail, representada pela transição (P5), que envia as informações para o aluno.

Após esta tradução, originou-se um novo modelo de simulação, o qual é representado pelo formalismo matemático das Redes de Petri Coloridas e Temporizadas. Elucida-se que este novo modelo é necessário porque o modelo de simulação original, apesar de representar fielmente a equivalência dos elementos, não possibilita realizar as análises inerentes as questões de pesquisa.

Este novo modelo de simulação considera mensagens com tamanhos diferentes. Nesse contexto, para diferenciar as mensagens, foi necessário criar uma função para cada tipo de mensagem, garantindo o seu intervalo de tempo de processamento.

A Figura §3.4 apresenta o modelo de simulação com Redes de Petri Coloridas e Temporizadas equivalente a solução de integração apresentada no caso de estudo.

A transição T_CTRL é o local em que os *tokens* são inseridos no sistema. A transição P1 (assíncrona) modela a porta de entrada, as transições P2 (síncrona), P3 (assíncrona) e P4 (assíncrona) representam as portas de solicitação para as aplicações Cadastro de Alunos, Cadastro de Disciplinas, Sistema de Cobrança, respectivamente. A transição P5 (assíncrona) modela a porta de saída a aplicação Servidor de *E-mail*. O modelo de simulação foi construído seguindo a organização e ligação das tarefas, a disposição das portas e a direção do fluxo de mensagens.

A tarefa filtro (TA, TAA e TAAA) tem a função de retirar mensagens do fluxo de acordo com o critério de filtragem definido, sendo que a função filtro é uma função booleana, a qual retorna um valor verdadeiro se o valor de a for menor ou igual a 19 e falso se for igual a 20. Isso quer dizer que 95% vai ser verdadeiro e 5% falso, funcionando como um filtro. Destaca-se que a transição TA retira do fluxo mensagens enviadas fora do prazo. A transição TAA retira do fluxo as mensagens que possuem as disciplinas que o aluno já cursou e também as disciplinas que não serão ofertadas no semestre. A transição TAAA filtra as mensagens incompletas. A tarefa tradutor (T2, T4, T9 e T13) tem a função de transformar a mensagem para um formato, que a aplicação

entenda, como as mensagens são equivalentes aos *tokens*, a tarefa é representada por um *slot* de entrada, uma transição e um *slot* de saída, que representam respectivamente entrada, processamento e saída.

A tarefa replicador (T3 e T8) é usada para replicar *tokens* para todas as suas saídas. A tarefa correlacionador (T5 e T10) é utilizada para localizar no fluxo mensagens correlacionadas, que devem ser processadas em conjunto. As transições só disparam quando houver *tokens* nos lugares de entrada, referentes às transições, e projeta conjuntamente *tokens* nos seus lugares de saída. A tarefa enriquecedor de conteúdo (T6 e T11) recebe mensagens correlacionadas e as combina em uma única, como representa o grafo equivalente em Redes de Petri, a transição T6 dispara quando houver ao menos um *token* nos lugares S9 e S10 e é colocado apenas um *token* no lugar S11, bem como a transição T11 dispara quando houver ao menos um *token* nos lugares S17 e S18, e é colocado apenas um *token* no lugar S19, representando a funcionalidade da tarefa.

3.4 Resumo do Capítulo

Neste capítulo, foi realizada a transcrição do modelo conceitual em um modelo formal de simulação, utilizando o formalismo matemático das Redes de Petri Coloridas e Temporizadas. Salienta-se, que a funcionalidade de cada tarefa é mantida ao efetivar a transcrição da tecnologia Guaraná para as Redes de Petri Coloridas e Temporizadas, pois as tarefas possuem uma semântica implementada que não pode ser traduzida. Foram descritas as funcionalidades das tarefas que compõem o modelo conceitual e seu grafo equivalente representado por meio do formalismo das Redes de Petri, o qual mantém essa funcionalidade. Por fim, é proposto o modelo formal de simulação por meio do formalismo matemático das Redes de Petri Coloridas e Temporizadas.

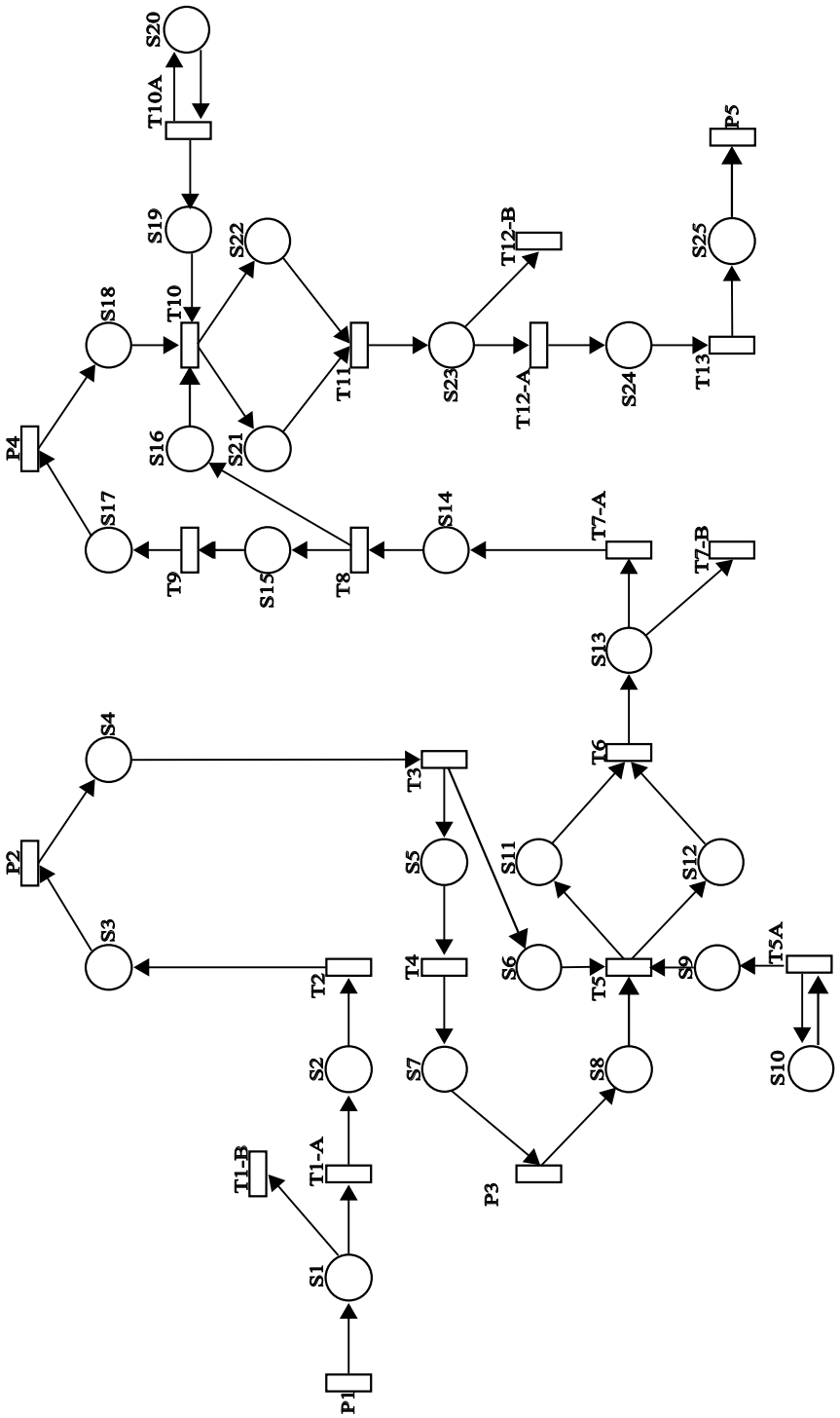


Figura 3.3: *Modelo de Simulação*

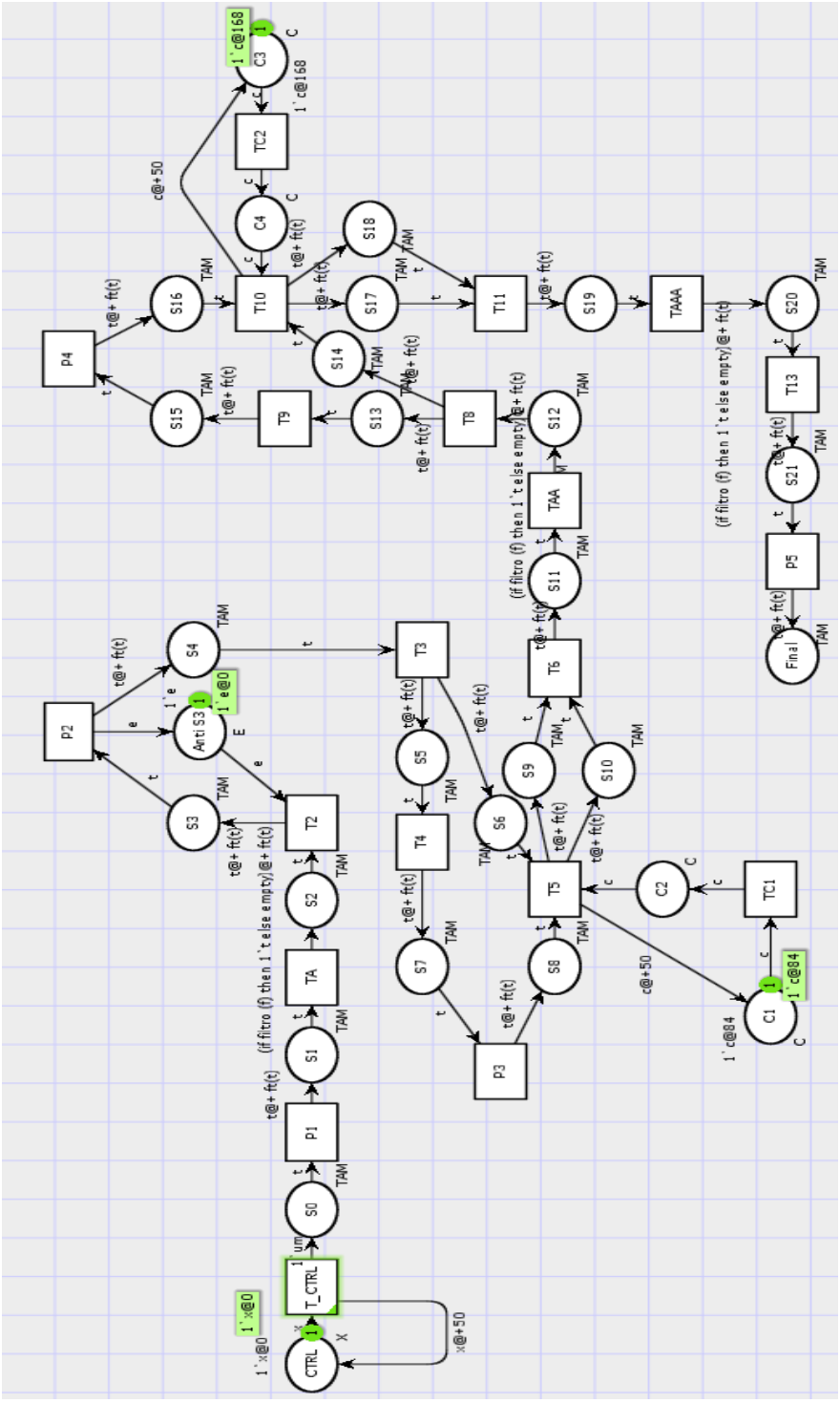


Figura 3.4: *Modelo Formal com Redes de Petri Coloridas e Temporizadas*

Capítulo 4

Experimento e Análise dos Resultados

O saber a gente aprende com os mestres e os livros. A sabedoria se aprende é com a vida e os humildes.

Cora Coralina

Este capítulo apresenta o experimento de simulação do caso de estudo. A Seção §4.1 apresenta a descrição do experimento e dos cenários utilizados para a simulação, as variáveis observadas e os resultados experimentais. A Seção §4.2 apresenta os resultados da simulação, bem como sua análise e interpretação. A Seção §4.3 descreve a verificação e a validação do modelo de simulação. Por fim, a Seção §4.4 expõe o resumo dos temas abordados.

4.1 Experimento

Segundo Wiesner [59], o desenvolvimento de um modelo de simulação obedece três grandes etapas: (a) formulação do modelo, nesta etapa busca-se compreender o problema e desenvolver o modelo conceitual; (b) implementação do modelo, nesta etapa desenvolve-se o modelo computacional, verificando-o e validando-o; e (c) análise dos resultados do modelo, esta etapa é responsável pela experimentação, interpretação, análise dos resultados e documentação.

Nesse sentido, uma das principais etapas, no estudo de um sistema por meio da simulação, é a experimentação, a qual permite analisar o desempenho da solução de integração, a partir dos resultados obtidos com a simulação, por meio das variáveis relacionadas ao desempenho da solução durante os experimentos nos diferentes cenários de funcionamento. Nesta pesquisa, levou-se em consideração as variáveis tempo médio de permanência das mensagens nos *slots* e o tamanho máximo e médio dos *slots*. Sob esta perspectiva, foram realizados os experimentos com o uso de seis cenários diferentes, a fim de identificar onde estão se formando os gargalos de desempenho.

4.1.1 Descrição do Experimento

O ambiente de execução da simulação foi um notebook Acer, Intel Core i5, 2.27GHz, com 2 núcleos físicos, 4GB, Disco Rígido 500 GB, Gravador de DVD-RW, Sistema Operacional Windows 64 bits.

Quanto ao desenvolvimento do modelo de simulação, primeiramente, foi criado o conjunto TAM para representar os diferentes tipos de mensagens, sendo: 1, 5, 10, 15 e 20 kB. Com a criação do conjunto TAM, foi criada a variável t, a qual é um elemento deste conjunto TAM. O conjunto TAM, é *timed*, pois representa o tempo de processamento de cada mensagem.

Para o tempo de processamento de cada mensagem, foram criadas 5 funções f(t), para diferenciar o tempo relativo ao tipo de mensagem. No caso:

- A mensagem de 1 kB leva aleatoriamente um tempo de 8 a 12 unidades de tempo;
- A mensagem de 5 kB leva aleatoriamente um tempo de 40 a 60 unidades de tempo;
- A mensagem de 10 kB leva aleatoriamente um tempo de 80 a 120 unidades de tempo;
- A mensagem de 15 kB leva aleatoriamente um tempo de 120 a 180 unidades de tempo;
- A mensagem de 20 kB leva aleatoriamente um tempo de 160 a 240 unidades de tempo;

Em seguida, foi desenvolvida a porta de entrada das mensagens, conforme a Figura §4.1.A transição T_CTRL tem a função de inserir os *tokens* no sistema. A porta de entrada tem a função de inserir novas mensagens na solução de integração. A tarefa porta de saída atua de forma contrária à porta de entrada, enviando mensagens para a aplicação integrada, seu grafo equivalente em Redes de Petri indica fluxo de *tokens* para fora do processo. A tarefa porta de solicitação e resposta (P2, P3 e P4) solicita informações a aplicações externas, as quais fornecem as respostas.

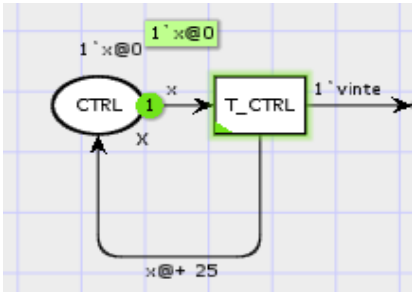


Figura 4.1: *Porta de Entrada Modelada no CPN Tools*

Para a representação de uma porta síncrona P2, foi necessário a criação de um Anti-S3. Este Anti-S3 permite que só exista 1 *token* de cada vez no lugar S3.

Com relação à tarefa filtro, foi necessário criar o conjunto F, que é constituído de números inteiros entre 1 e 20, incluindo o 1 e o 20. Com a criação do conjunto (F), foi feita a variável f, a qual é um elemento deste conjunto F. Posteriormente, foi criada a função filtro, a qual é uma função booleana que retorna um valor verdadeiro se o valor de a for menor ou igual a 19 e falso se for igual a 20. Isso quer dizer que 95% vai ser verdadeiro e 5% falso, funcionando como um filtro. A tarefa filtro (TA, TAA e TAAA) tem a função de retirar mensagens do fluxo da solução de integração de acordo com a filtragem definida.

A tarefa correlacionador é representada por uma transição, a T5. O lugar C1 envia um *token* para o lugar C2. Quando ocorre a existência de três *tokens* simultâneos, um *token* no lugar S6, um *token* no lugar S8 e um *token* no lugar C2, a transição T5 está habilitada. Do mesmo modo, ocorre com a transição T10. O lugar C3 envia um *token* para o lugar C4. Quando ocorre a existência de três *tokens* simultâneos, um *token* no lugar S14, um *token* no lugar S16 e um *token* no lugar C4, a transição T10 está habilitada.

A tarefa tradutor (T2, T4, T9 e T13) tem a função de transformar a mensagem a um formato que a aplicação seguinte entenda, como as mensagens são equivalentes aos *tokens*, a tarefa é representada por um *slot* de entrada, uma transição e um *slot* de saída que representam entrada, processamento e saída da tarefa, respectivamente.

A tarefa replicador é usada para replicar mensagens para todas as suas saídas. Esta tarefa é representada pelas transições T3 e T8.

A tarefa enriquecedor de conteúdo recebe mensagens correlacionadas e as combina em uma única, como representa o grafo equivalente em Redes de Petri, a transição T6 dispara quando houver ao menos um *token* nos lugares S9 e S10, e é colocado apenas um *token* no lugar S11, mesmo fato ocorre na transição T11 que dispara quando houver ao menos um *token* nos lugares S17 e S18, e é colocado apenas um *token* no lugar S19 representando a funcionalidade da tarefa.

A cada transição é incrementado o tempo associado ao *token*, sendo que esse incremento representa o somatório do tempo que a mensagem permanece no lugar anterior com o tempo que a mensagem demora para ser processada pela transição. O primeiro *token* chega com tempo 0 e a cada transição ele adiciona 200 unidades de tempo.

Para a obtenção dos dados, foram criados monitores, do tipo *Mark Size*. Cada monitor esta relacionado a um *slot* (lugar), o qual gera um log, que faz o registro da quantidade de *tokens* em cada *slot* (lugar) e também fornece o tempo de permanência de cada mensagem em cada *slot* (lugar). Dessa forma, permite responder as questões: tempo médio de permanência das mensagens nos *slots*; tamanho máximo e médio dos *slots*. Ressalta-se que a funcionalidade de cada tarefa é mantida ao realizar a transcrição das tarefas da tecnologia Guaraná para as Redes de Petri.

4.1.2 Descrição dos Cenários

O modelo de simulação proposto, foi desenvolvido para ser utilizado em 1 experimento, sendo que: consiste em executar uma simulação, considerando mensagens com tamanhos diferentes. Cada tamanho esta sendo representado por um *token* diferente. Os tamanhos a serem considerados são: 1, 5, 10, 15, 20 kB. Os *slots* são considerados FIFO. Há portas síncronas e assíncronas. A partir disso, analisa-se o que ocorre quanto ao tempo médio de permanência das mensagens nos *slots*, e o tamanho máximo e médio dos *slots*.

Os experimentos foram realizados em 6 cenários diferentes, sendo:

- Cenário 1 com 25 unidades de tempo: isto significa que, a cada 25 unidades de tempo a transição T_CTRL lança um *token* para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb);

- Cenário 2 com 50 unidades de tempo: é lançado 1 *token* a cada 50 unidades de tempo para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb);

- Cenário 3 com 100 unidades de tempo: é lançado 1 *token* a cada 100 unidades de tempo para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb);

- Cenário 4 com 200 unidades de tempo: é lançado 1 *token* a cada 200 unidades de tempo para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb);

- Cenário 5 com 400 unidades de tempo: é lançado 1 *token* a cada 400 unidades de tempo para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb);

- Cenário 6 com 800 unidades de tempo: é lançado 1 *token* a cada 800 unidades de tempo para dentro do sistema, além disso, assumimos que o tempo de processamento das tarefas são iguais e que existem 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb).

Os 6 cenários utilizados para a realização de cada experimento consideram uma carga de entrada de 10.000 *tokens* (sorteados aleatoriamente entre os valores possíveis). O incremento do tempo representa o intervalo em que acontecem as entradas de *tokens* no sistema. O critério de parada, em todos os cenários ocorre, ao terminar de injetar os 10.000 *tokens*.

Cada cenário corresponde a um intervalo de tempo de entrada diferente para as mensagens. Essa variação no tempo foi utilizada para analisar o desempenho da solução sob diferentes cargas de trabalho. Cabe ainda ressaltar, que são usados 5 tamanhos de *tokens* (1, 5, 10, 15 e 20 Kb), a opção por estes tamanhos levou em consideração a solução de integração, sendo um valor próximo da realidade.

Quanto a opção pelo número de repetições do experimento levou-se em consideração a especificação de Grinstead e Snell [29] que determina que experimentos como este devem apresentar cerca de 25 repetições. Em estatística, quando um experimento é repetido um grande número de vezes com os mesmos dados, seguindo a Lei dos Grandes Números, conforme o número de repetições se incrementa a média amostral das variáveis do experimento, dessa forma, os resultados se aproximam cada vez mais da média populacional, também conhecida como média teórica ou esperança matemática. As repetições buscam eliminar qualquer anormalidade nos resultados, uma vez que se trata de um processo estocástico. A ferramenta de simulação apresenta o tempo médio de permanência das mensagens em cada *slot* e o tamanho máximo e médio dos *slots*.

Empiricamente, para o tipo de experimento, a média populacional costuma ser obtida com, aproximadamente, 20 a 30 repetições [55]. Salienta-se que este número pode variar, isto porque, depende da "estabilidade"dos resultados. Assim, se ao executar cerca de 5 repetições e não houverem *outliers* ou se o número for insignificante, o número de repetições pode ser reduzido.

Este modelo de simulação foi implementado na ferramenta de simulação CPN Tools e executado 5 vezes para excluir qualquer discrepância nos dados. A opção pela quantidade de repetições ocorreu após a realização de testes com o intuito de verificar a quantidade de repetições necessárias para manter os resultados estáveis. Ou seja, não foram efetuadas as 25 repetições, pelo fato de não existirem *outliers* (diferenças significativas). Então, como os resultados são considerados estáveis, foi possível diminuir as repetições [29, 55].

Para o cálculo das médias dos experimentos, é possível seguir os seguintes critérios:

- Executar o experimento repetidas vezes, aproximadamente, 25 (quando existirem *outliers*), caso contrário pode-se efetuar menos repetições;
- Remover possíveis *outliers* inferiores e superiores utilizando um método adequado;
- Calcular a média dos valores, já descartados os *outliers*.

Para obter precisão estatística sobre os resultados da simulação, foi necessário repetir a execução do modelo várias vezes. Na modelagem de

sistemas estocásticos, entradas aleatórias geram saídas aleatórias, de maneira, que os resultados gerados pelos modelos estocásticos são diferentes em cada replicação, pois, as variáveis de entrada são aleatórias [59].

A replicação consiste na repetição da simulação do modelo, utilizando a mesma configuração, mesma duração e mesmos parâmetros de entrada, porém, com uma semente de geração dos números aleatórios diferentes. Isso significa que, apesar dos parâmetros de entrada serem os mesmos, os números aleatórios gerados são diferentes, pois cada replicação terá resultados diferentes [11].

4.1.3 Variáveis Observadas

Em cada cenário, foram coletados os dados para duas variáveis: tempo médio de permanência das mensagens nos *slots* e tamanho máximo e médio dos *slots*.

A variável tempo médio de permanência das mensagens nos *slots* representa o tempo que os diferentes tipos de mensagens permaneceram em cada *slot*. Assim, quanto mais tempo uma mensagem permanece em um determinado *slot* indica atraso, surgindo acúmulos que ocasionam gargalos.

A variável tamanho máximo e médio dos *slots* representa o número de mensagens que estão em cada *slot*. O acúmulo de mensagens nos *slots* representa a formação de filas nos *slots* da solução de integração. As filas são determinadas pela quantidade de mensagens que se agrupam nos *slots*. Sua análise possibilita uma visualização do estado geral do sistema, sendo a formação de filas muito elevadas em determinados *slots* certamente um indicador de que a tarefa esta tendo dificuldade de processar a demanda de mensagens exigidas [7]. Sendo assim, o tamanho do *slot* indica onde estão se acumulando as mensagens, sendo este mais um indicador de gargalos.

4.1.4 Apresentação da Ferramenta

A ferramenta de simulação utilizada para a análise do modelo foi o *CPN Tools* [35]. O *CPN Tools* é uma ferramenta de modelagem, análise e simulação de Redes de Petri Coloridas. A ferramenta foi desenvolvida na Universidade de Aarhus, Dinamarca, e é distribuída livremente para organizações não comerciais via *web*. Esta ferramenta tem sido utilizada em várias aplicações e projetos importantes, especialmente nas áreas de telecomunicações e sistemas de manufatura.

Na Figura §4.2 é ilustrada a *interface* gráfica do *CPN Tools* para a criação e edição de modelos de Redes de Petri. Para a criação de modelos, é fornecido um editor gráfico especial de Redes de Petri Coloridas. O editor permite que se desenhe uma Rede de Petri na tela do computador, bem como se escreva os atributos dos elementos da rede e declarações adicionais escritas na linguagem CPN ML. Os modelos são criados em uma tela usando componentes a partir de uma barra de ferramentas chamada *create*, nela é possível criar lugares, transições e arcos. A *interface* desta ferramenta oferece recursos como zoom, exportação, importação e edição por abas.

Quanto ao Tool Box do CPN Tools, este fornece as seguintes ferramentas:

- Ferramentas *auxiliary*: para a melhoria da legibilidade da rede;
- Ferramentas *create*: para edição de redes;
- Ferramentas *hierarchy*: para criação de redes hierárquicas;
- Ferramentas *monitoring*: para análise da simulação do comportamento das redes;
- Ferramentas *net*: para operações com toda a rede;
- Ferramentas *simulation*: para simulação do comportamento das redes;
- Ferramentas *state space*: para criação e análise do espaço de estados das redes;
- Ferramentas *style*: para modificações na aparência das redes;
- Ferramentas *view*: para escolha de escalas de visualização e destaque de grupos de elementos.

Quanto a análises dos modelos, a única forma de análise é a simulação de comportamento dos mesmos. O CPN *Tools* fornece simulação passo-a-passo para depuração do modelo, bem como simulação automática com um certo número de passos.

Através da simulação, é possível utilizar-se de uma outra ferramenta do *CPN Tools*, denominada monitor e, assim, analisar aspectos específicos de uma rede, tais como: quantas vezes uma determinada transição foi disparada; uma determinada marcação foi alcançada; tempo médio de realização de uma determinada tarefa, etc.

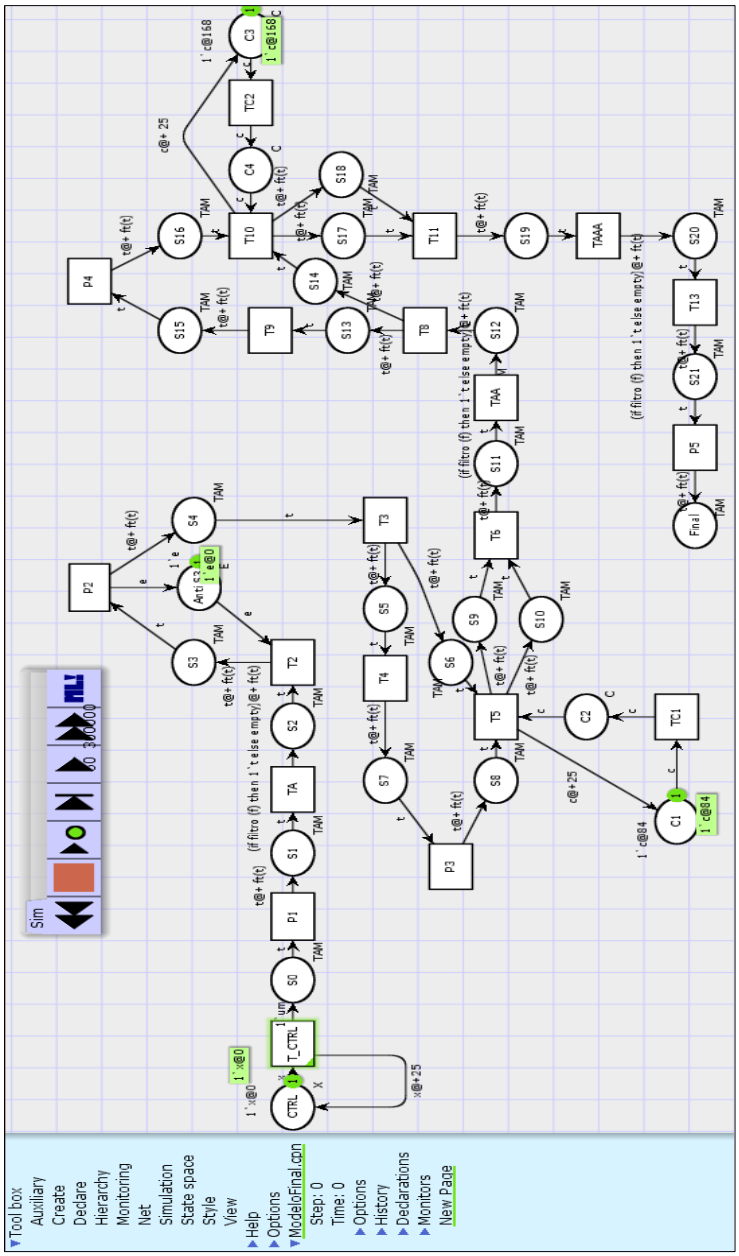


Figura 4.2: *Interface Gráfica do CPN Tools*

4.2 Resultados e Discussões

Inicialmente, apresentam-se os resultados da variável tempo médio de permanência das mensagens nos *slots* dos seis cenários, bem como os gráficos dessa variável para estimar as medidas de desempenho da solução de integração.

No cenário 1, conforme o gráfico da Figura §4.3 e Apendice §B Tabela §B.1, pode-se constatar que leva-se mais tempo para o processamento das mensagens no *slot* S2, porque há um intervalo de entrada menor, no caso, 25 unidades de tempo, ou seja, são injetadas mensagens a cada 25 unidades de tempo. Este tempo maior também deve-se à transição que representa a porta de solicitação P2 ser síncrona, isto significa que, enquanto o banco de dados esta processando uma requisição da aplicação externa, a porta síncrona não aceita a entrada de novas mensagens. Uma porta síncrona faz uma solicitação por vez à aplicação, ou seja, faz uma solicitação e aguarda a resposta desta solicitação para então fazer a seguinte. Se difere de uma porta assíncrona, que faz diversas solicitações à aplicação, sem que tenha que aguardar o retorno de uma solicitação para fazer próxima. Já o tempo maior encontrado nos *slots* S6 e S14, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Outro fato relevante é que as mensagens maiores levam mais tempo para serem processadas.

No cenário 2, conforme o gráfico da Figura §4.4 e Apendice §B Tabela §B.2, pode-se constatar que leva-se mais tempo para se processar as mensagens no *slot* S2, porque há um intervalo de entrada de 50 unidades de tempo, ou seja, são injetadas mensagens a cada 50 unidades de tempo. Este tempo maior também se deve, a transição que representa a porta de solicitação P2 ser síncrona. Já o tempo maior encontrado nos *slots* S6 e S14, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. As mensagens maiores levam mais tempo para serem processadas.

No cenário 3, conforme o gráfico da Figura §4.5 e Apendice §B Tabela §B.3, pode-se constatar que leva-se mais tempo para se processar as mensagens no *slot* S2, porque há um intervalo de entrada de 100 unidades de tempo, ou seja, são injetadas mensagens a cada 100 unidades de tempo. Este tempo maior também deve-se à transição que representa a porta de solicitação P2 ser síncrona. Já o tempo maior encontrado nos *slots* S6 e S14, deve-se à necessidade

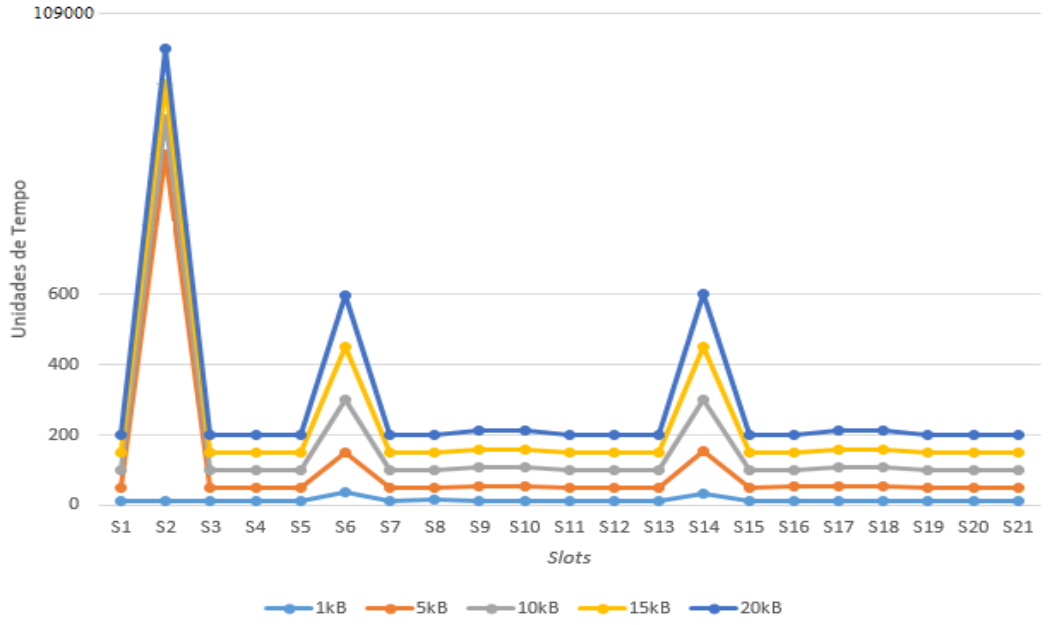


Figura 4.3: *Tempo médio de permanência das mensagens nos slots, cenário 1*

do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. As mensagens maiores levam mais tempo para serem processadas.

No cenário 4, conforme o gráfico da Figura §4.6 e Apendice §B Tabela §B.4, pode-se constatar que leva-se mais tempo para se processar as mensagens no *slot* S2, porque há um intervalo de entrada de 200 unidades de tempo, ou seja, são injetadas mensagens a cada 200 unidades de tempo, mas denota-se que este tempo de processamento está diminuindo em comparação aos cenários anteriores, pois, o intervalo de entrada das mensagens é maior. Este tempo maior também deve-se à transição que representa a porta de solicitação P2 ser síncrona. Já o tempo maior encontrado nos *slots* S6, S8, S14 e S16, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. As mensagens maiores levam mais tempo para serem processadas.

No cenário 5, conforme o gráfico da Figura §4.7 e Apendice §B Tabela §B.5, pode-se constatar que o tempo de processamento das mensagens no *slot* S2 está normal, em comparação aos outros cenários. Isto porque o

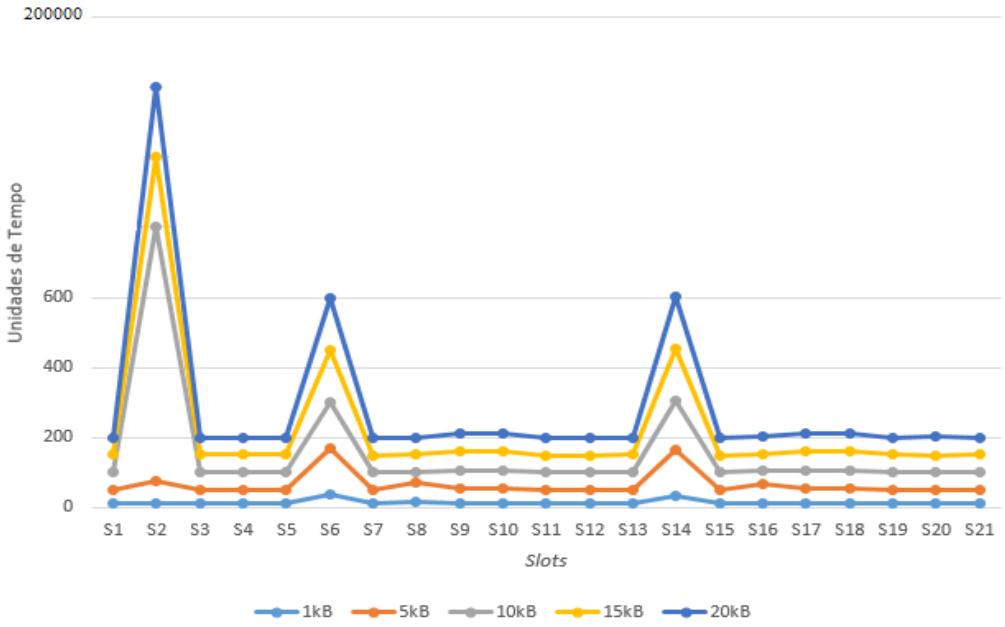


Figura 4.4: *Tempo médio de permanência das mensagens nos slots, cenário 2*

intervalo de entrada é de 400 unidades de tempo, ou seja, são injetadas mensagens a cada 400 unidades de tempo. Já o tempo de permanência maior é encontrado nos *slots* S6, S8, S14 e S16 deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque, deve aguardar as mensagens que possuem a mesma identidade para correlacionar. As mensagens maiores levam mais tempo para serem processadas.

No cenário 6, conforme o gráfico da Figura §4.8 e Apendice §B Tabela §B.6, pode-se constatar que o tempo de processamento das mensagens no *slot* S2 está normal, em comparação aos outros cenários. Isto porque o intervalo de entrada é de 800 unidades de tempo, ou seja, são injetadas mensagens a cada 800 unidades de tempo. Já o tempo de permanência maior é encontrado nos *slots* S6, S8, S14 e S16 deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. As mensagens maiores levam mais tempo para serem processadas. Outro fato relevante, observado em todos os cenários é que as mensagens maiores permanecem por mais tempo nos *slots* pois, foram criadas 5 funções f(t) para diferenciar o tempo relativo ao tipo de mensagem. A mensagem de 1 kB leva aleatoriamente um tempo de 8 a 12 unidades de tempo. A mensagem de 5 kB leva aleatoriamente um

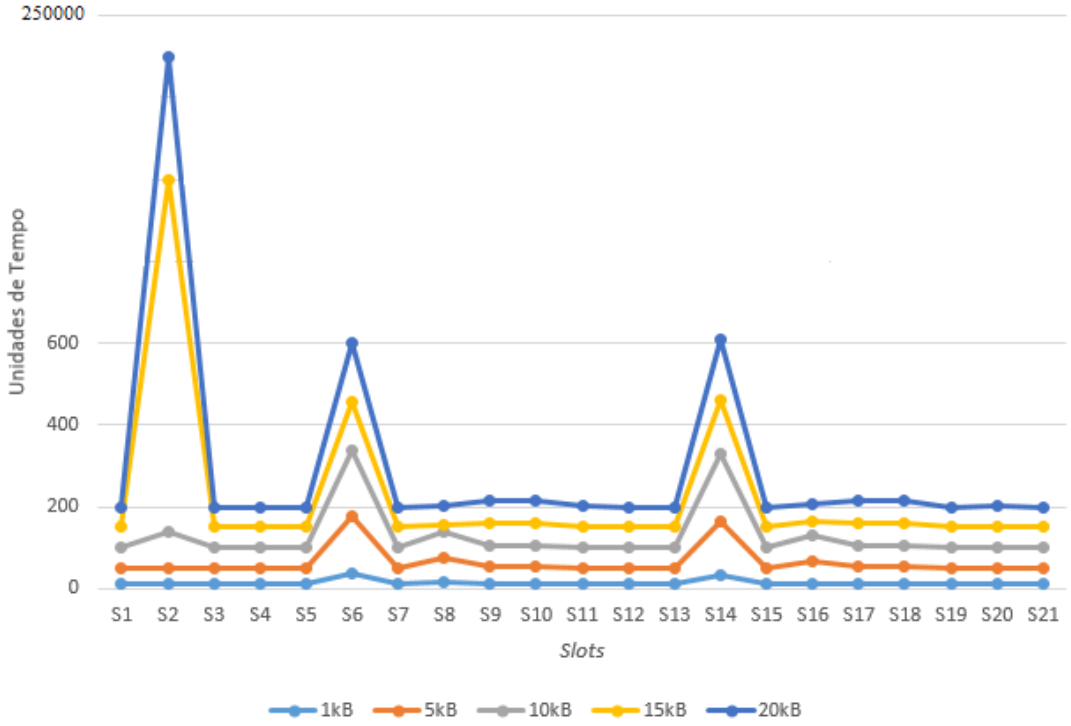


Figura 4.5: *Tempo médio de permanência das mensagens nos slots, cenário 3*

tempo de 40 a 60 unidades de tempo. A mensagem de 10 kB leva aleatoriamente um tempo de 80 a 120 unidades de tempo. A mensagem de 15 kB leva aleatoriamente um tempo de 120 a 180 unidades de tempo. A mensagem de 20 kB leva aleatoriamente um tempo de 160 a 240 unidades de tempo.

Além de estimar o tempo médio de permanência das mensagens nos *slots*, também foi possível analisar o desempenho da solução de integração quanto ao tamanho máximo e médio dos *slots*. Quando um *slot* assume valores além dos adequados, pode constituir um gargalo de desempenho. Os resultados obtidos da variável estão expressos em gráficos e permitem analisar o comportamento, identificar possíveis gargalos de desempenho da solução de integração, em seis diferentes cenários.

No cenário 1, conforme os gráficos das Figuras §4.9 e §4.10, pode-se observar que quando o intervalo de entrada é menor, no caso, 25 unidades de tempo, tendem a se acumular mais mensagens nos *slots* S1 e S2 respectivamente, pelo fato de serem injetadas as mensagens a cada 25 unidades de

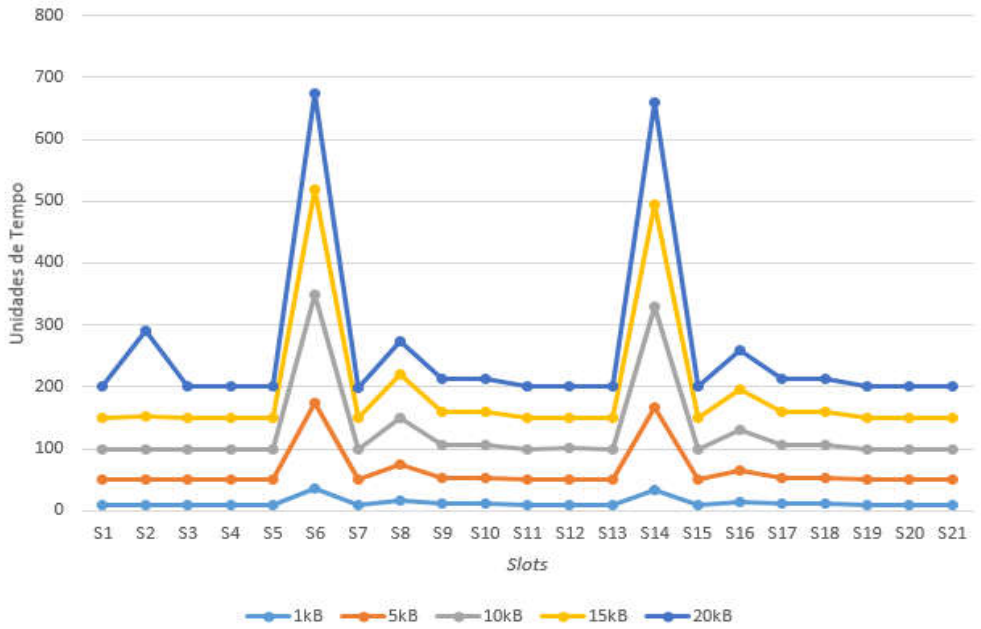


Figura 4.6: *Tempo médio de permanência das mensagens nos slots, cenário 4*

tempo. Este acúmulo também se deve, à porta P2 ser síncrona, isto significa que, enquanto o banco de dados está processando uma requisição da aplicação externa, a porta síncrona não aceita a entrada de novas mensagens. Já o acúmulo encontrado nos *slots* S6 e S14, conforme os gráficos das Figuras §4.11 e §4.12 respectivamente, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser verificados no Apendice §B Tabela §B.7.

No cenário 2, conforme os gráficos das Figuras §4.13 e §4.14 pode-se observar que quando o intervalo de entrada é menor, no caso, 50 unidades de tempo, tendem a se acumular mensagens nos *slots* S1 e S2 respectivamente, pelo fato de serem injetadas as mensagens a cada 50 unidades de tempo. Este acúmulo também se deve, à porta P2 ser síncrona. Em comparação com o gráfico do cenário 1, percebe-se que o acúmulo nos S1 e S2 estão diminuindo, isto ocorre porque está se injetando uma mensagem a cada 50 unidades de tempo. Já o acúmulo encontrado nos *slots* S6 e S14, conforme os gráficos das Figuras §4.15 e §4.16 respectivamente, deve-se à necessidade do uso do correlacionador, que acaba provocando um

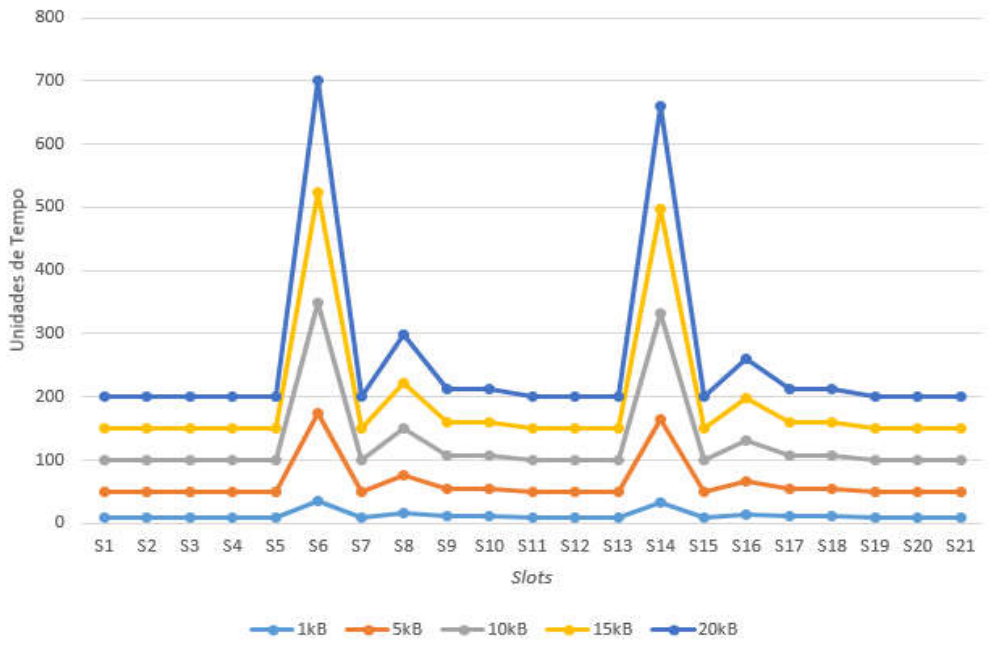


Figura 4.7: *Tempo médio de permanência das mensagens nos slots, cenário 5*

atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser verificados no Apendice §B Tabela §B.8.

No cenário 3, conforme os gráficos das Figuras §4.17 e §4.18 pode-se observar que quando o intervalo de entrada é 100 unidades de tempo, tendem a se acumular mensagens nos *slots* S1 e S2, pelo fato de serem injetadas as mensagens a cada 100 unidades de tempo. Este acúmulo também se deve, à porta P2 ser síncrona. Em comparação com os gráficos dos cenários 1 e 2, percebe-se que o acúmulo no *slot* S2 está diminuindo, isto ocorre, porque está se injetando uma mensagem a cada 100 unidades de tempo. Já o acúmulo encontrado nos *slots* S6 e S14, conforme os gráficos das Figuras §4.19 e §4.20 respectivamente, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser verificados no Apendice §B Tabela §B.9.

No cenário 4, pode-se observar que quando o intervalo de entrada é 200 unidades de tempo, tendem a se acumular mensagens de 10, 15 e 20 kB, nos

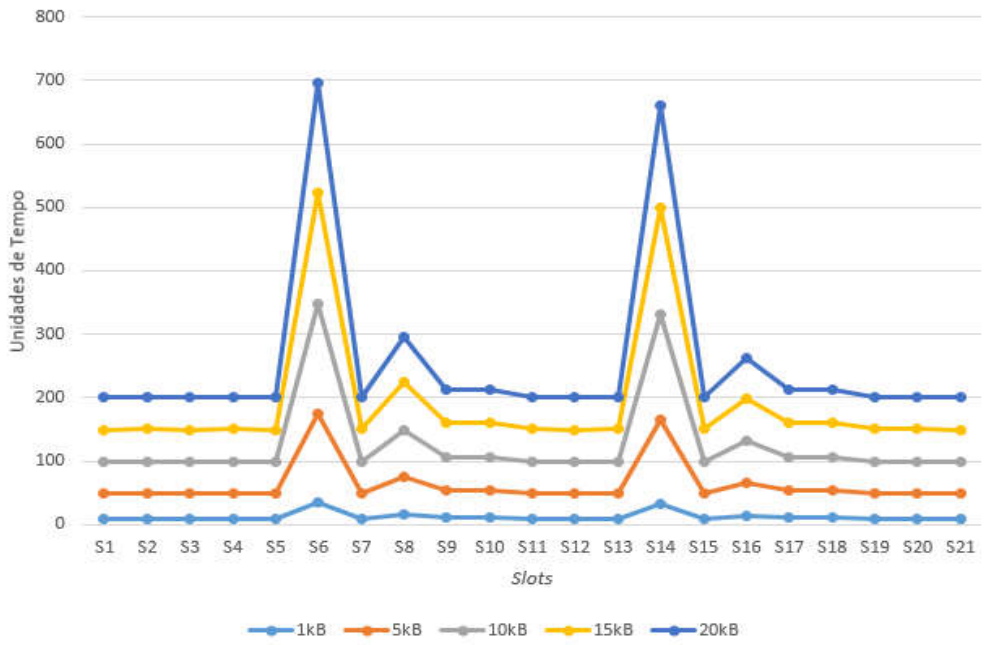


Figura 4.8: *Tempo médio de permanência das mensagens nos slots, cenário 6*

slots, pelo fato de serem injetadas as mensagens a cada 200 unidades de tempo, tendo uma taxa de entrada grande e um processamento mais lento para as mensagens maiores (10, 15 e 20 kB). Já o acúmulo encontrado nos *slots* S2, S6, S8, S14 e S16, conforme gráficos das Figuras §4.21, §4.22, §4.23, §4.24 e §4.25 respectivamente, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser verificados no Apendice §B Tabela §B.10.

No cenário 5, pode-se observar que quando o intervalo de entrada é 400 unidades de tempo, tendem a se acumular mensagens de 15 e 20 kB, nos *slots*, pelo fato de serem injetadas as mensagens a cada 400 unidades de tempo, tendo uma taxa de entrada maior e um processamento mais lento para as mensagens maiores (15 e 20 kB). Já o acúmulo encontrado nos *slots* S6 e S14, conforme os gráficos das Figuras §4.26 e §4.27 respectivamente, deve-se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser verificados no Apendice §B Tabela §B.11.

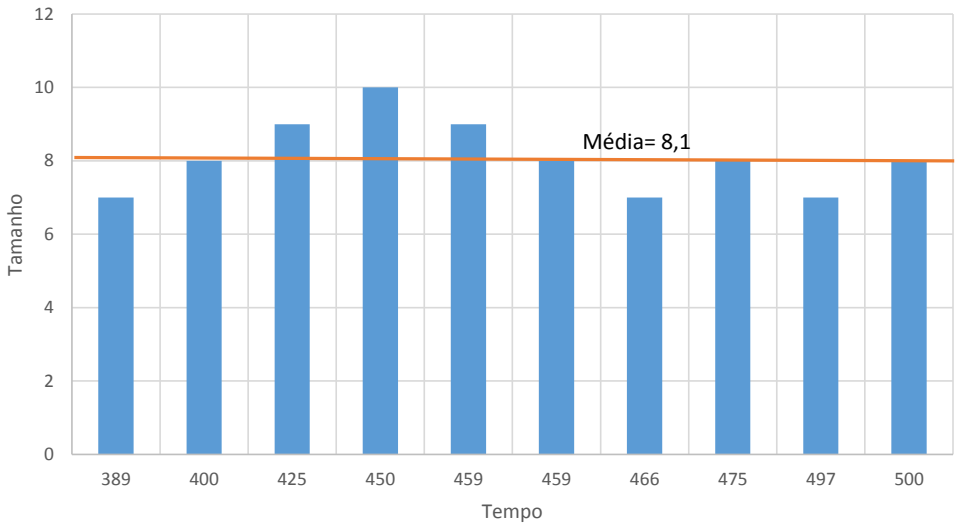


Figura 4.9: *Tamanho máximo e médio do slot S1 no cenário 1*

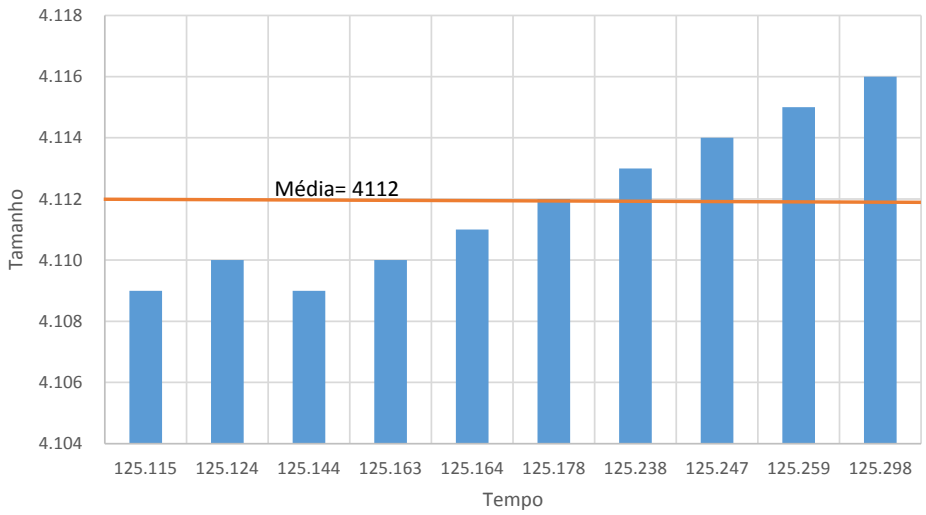


Figura 4.10: *Tamanho máximo e médio do slot S2 no cenário 1*

No cenário 6, pode-se observar que quando o intervalo de entrada é 800

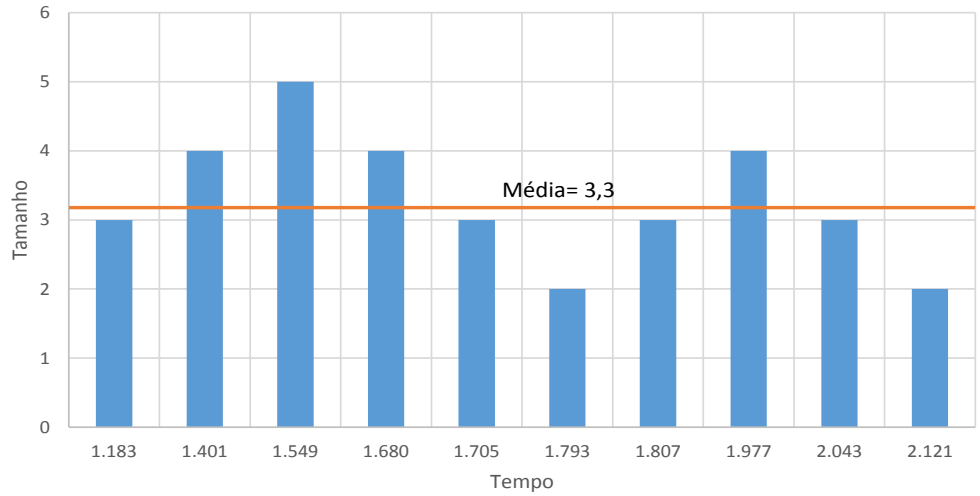


Figura 4.11: *Tamanho máximo e médio do slot S6 no cenário 1*

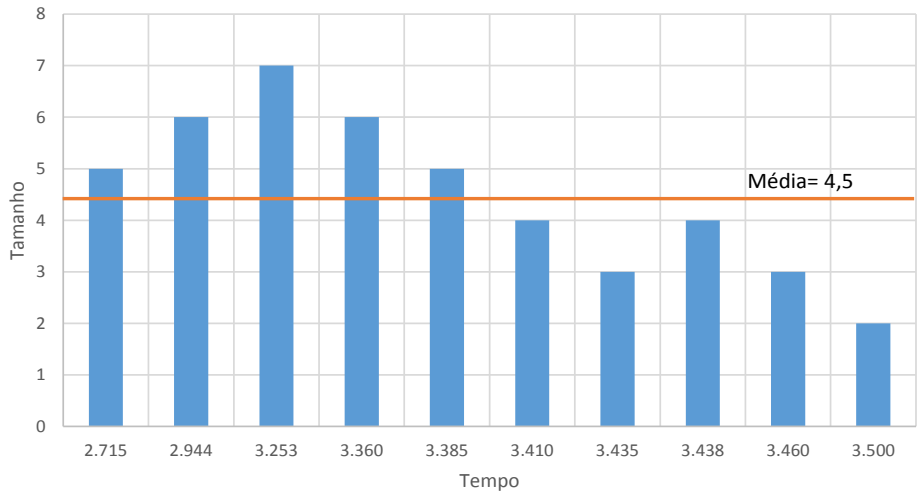


Figura 4.12: *Tamanho máximo e médio do slot S14 no cenário 1*

unidades de tempo, não tendem a se acumular mensagens nos *slots*, pelo fato de serem injetadas as mensagens a cada 800 unidades de tempo, tendo uma

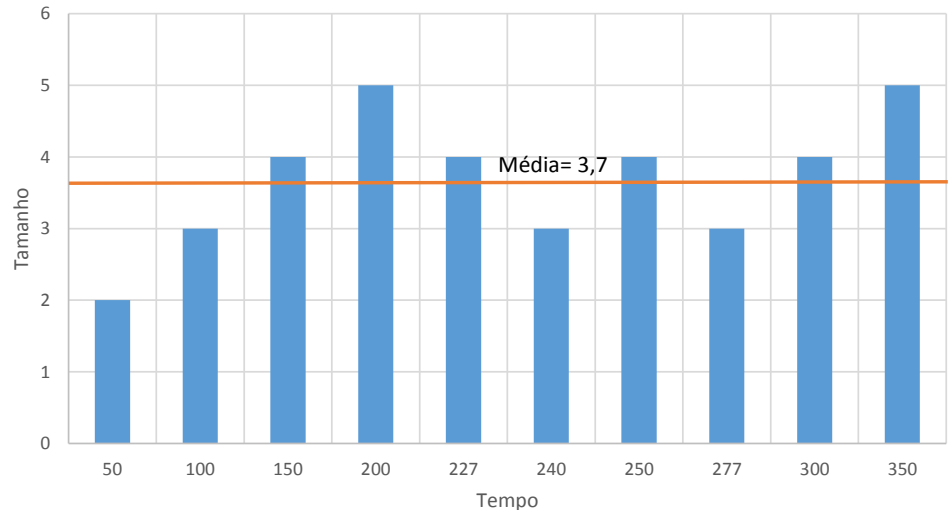


Figura 4.13: *Tamanho máximo e médio do slot S1 no cenário 2*

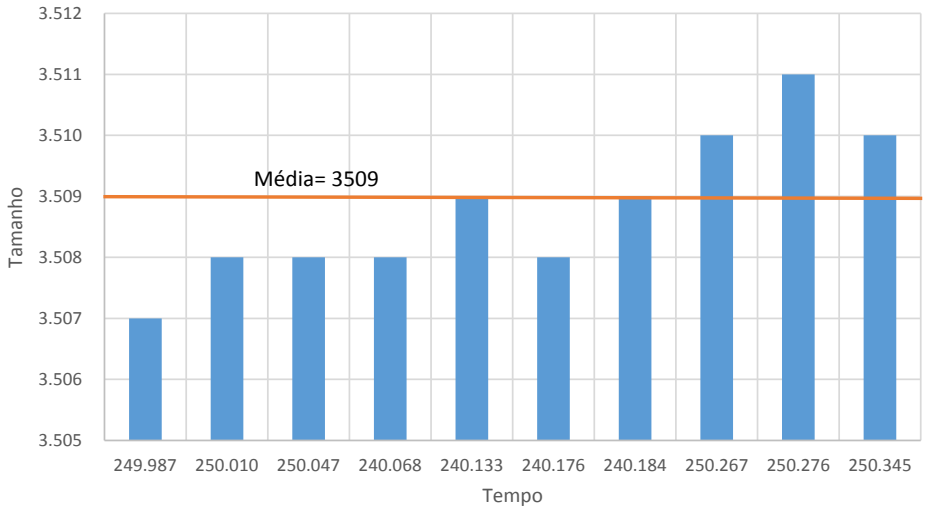


Figura 4.14: *Tamanho máximo e médio do slot S2 no cenário 2*

taxa de entrada muito grande em relação ao tempo de processamento. Na porta síncrona, as mensagens são processadas, antes da chegada da próxima mensagem. Já o acúmulo, não muito singnificativo encontrado nos *slots* S6 e S14, conforme os gráficos das Figuras §4.28 e §4.29 respectivamente, deve-

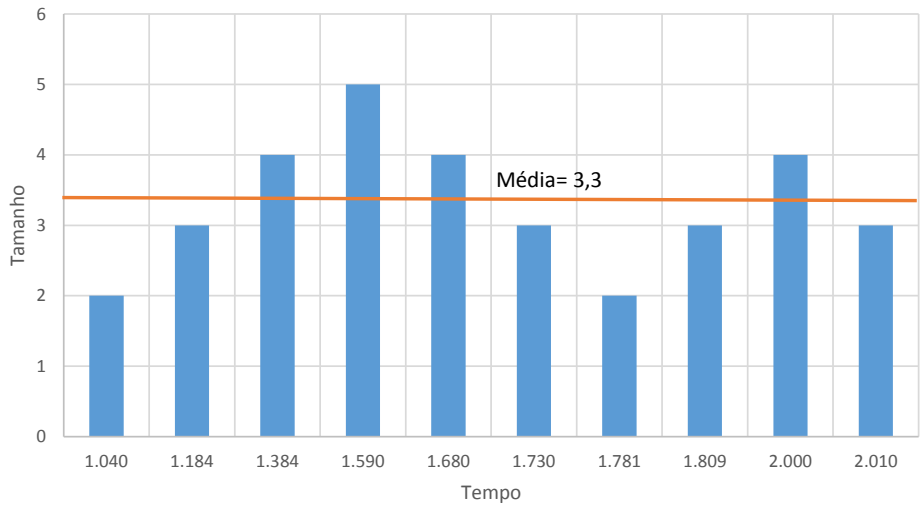


Figura 4.15: *Tamanho máximo e médio do slot S6 no cenário 2*

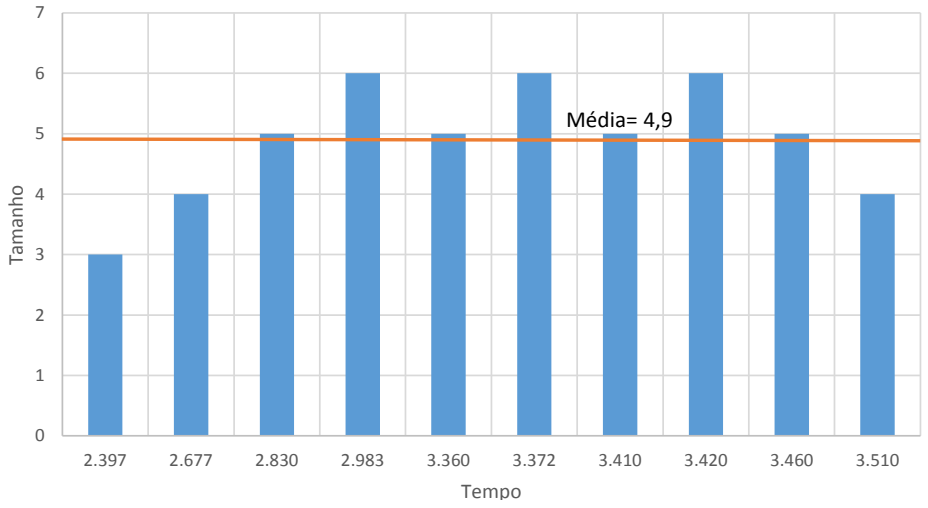


Figura 4.16: *Tamanho máximo e médio do slot S14 no cenário 2*

se à necessidade do uso do correlacionador, que acaba provocando um atraso, porque deve aguardar as mensagens que possuem a mesma identidade para correlacionar. Os resultados para esta variável também podem ser

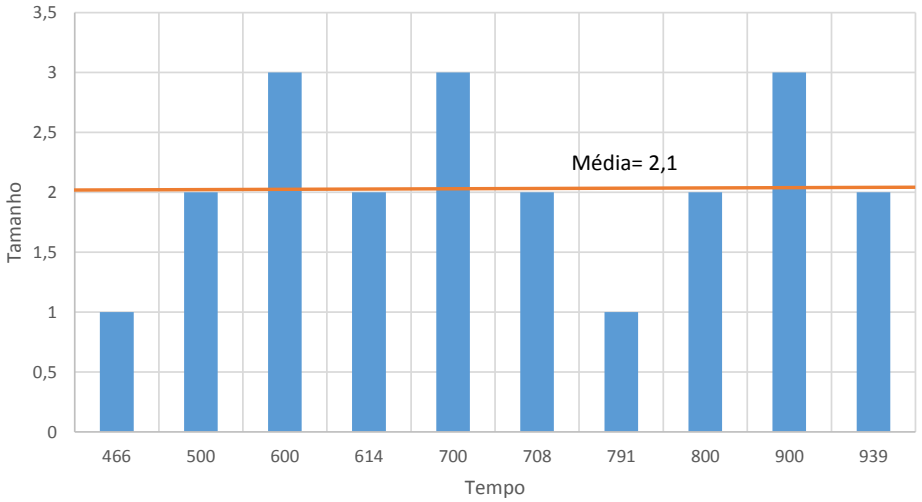


Figura 4.17: *Tamanho máximo e médio do slot S1 no cenário 3*

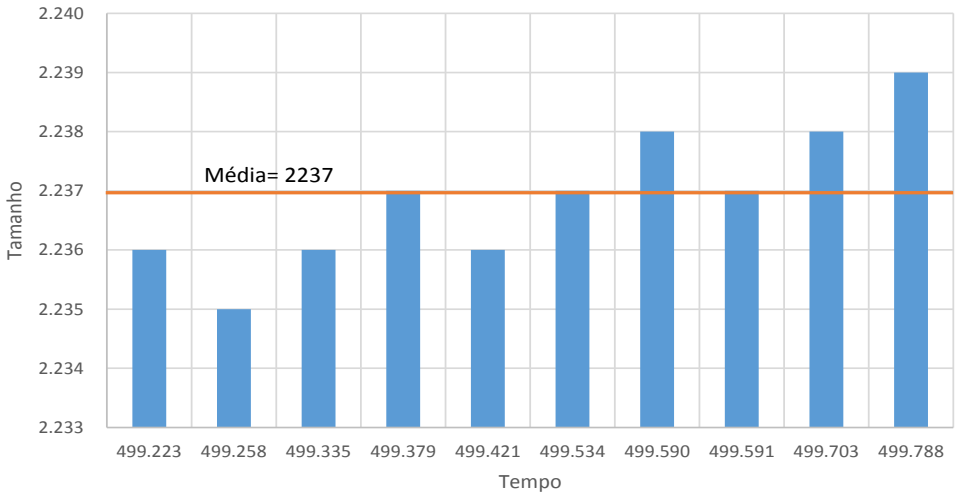


Figura 4.18: *Tamanho máximo e médio do slot S2 no cenário 3*

verificados no Apendice §B Tabela §B.12.

A partir da análise do comportamento da solução de integração, é possível destacar que nos cenários em que o intervalo de entrada dos *tokens* é

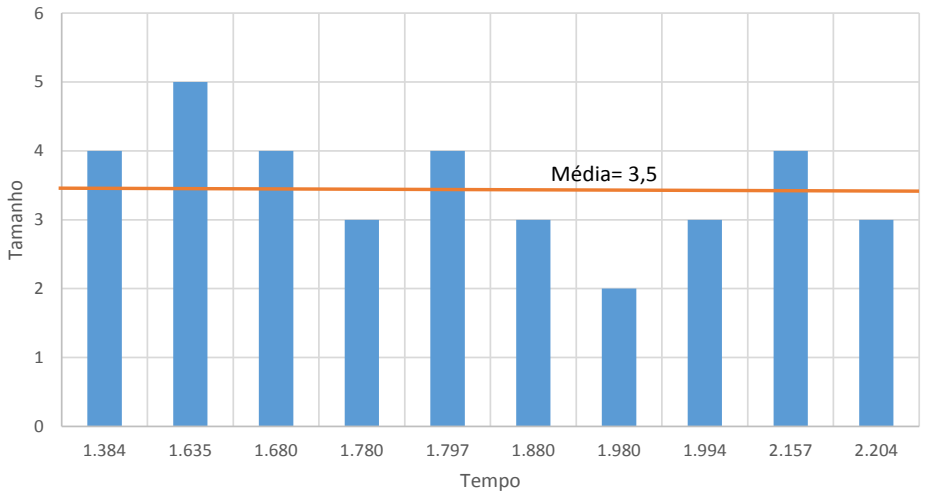


Figura 4.19: *Tamanho máximo e médio do slot S6 no cenário 3*

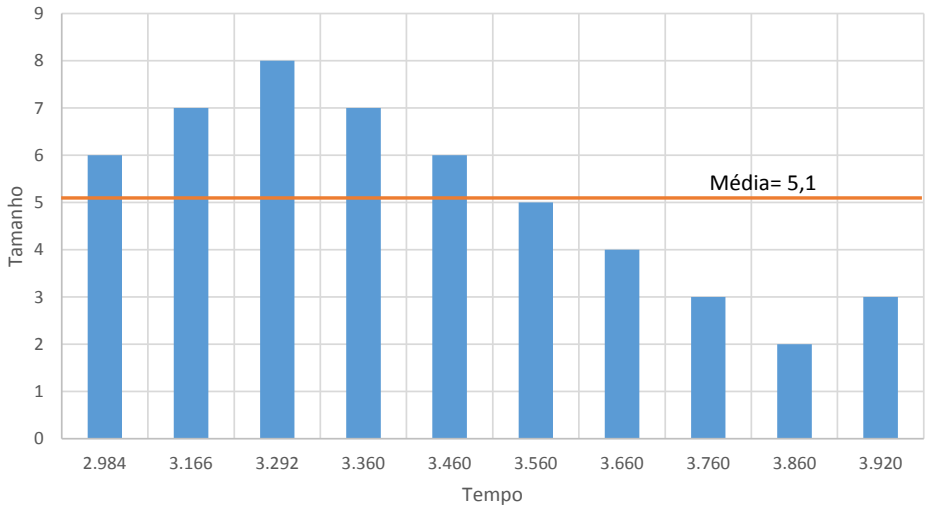


Figura 4.20: *Tamanho máximo e médio do slot S14 no cenário 3*

menor, ou seja, nos cenários com 25, 50, 100 e 200 unidades de tempo é perceptível que as mensagens tendem a se acumularem mais nos *slots* S1, S2, S6,

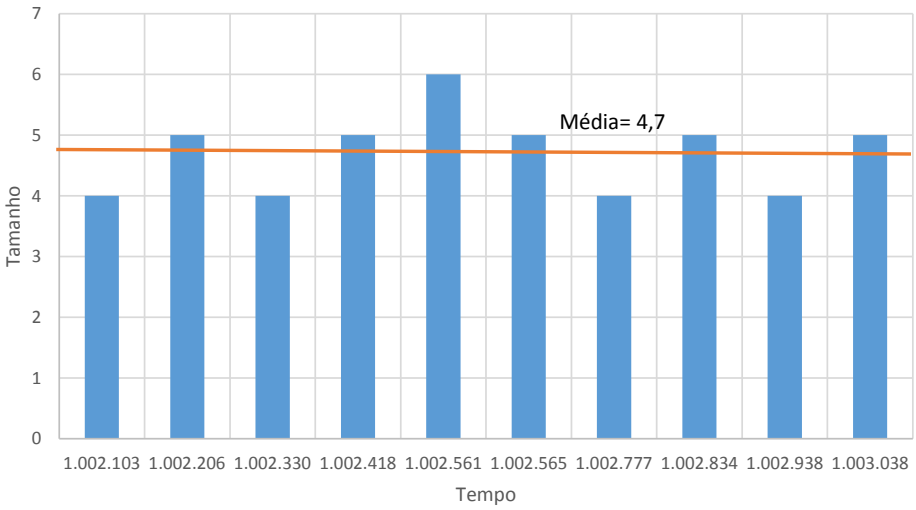


Figura 4.21: *Tamanho máximo e médio do slot S2 no cenário 4*

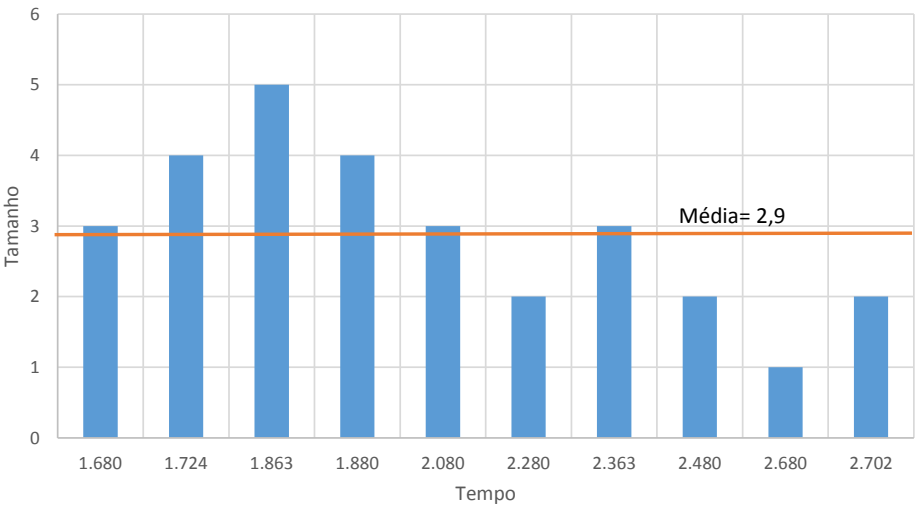


Figura 4.22: *Tamanho máximo e médio do slot S6 no cenário 4*

S14 e S16. Enquanto que nos cenários de 400 e 800 unidades de tempo, as mensagens tendem a diminuirem consideravelmente seu acúmulo nos

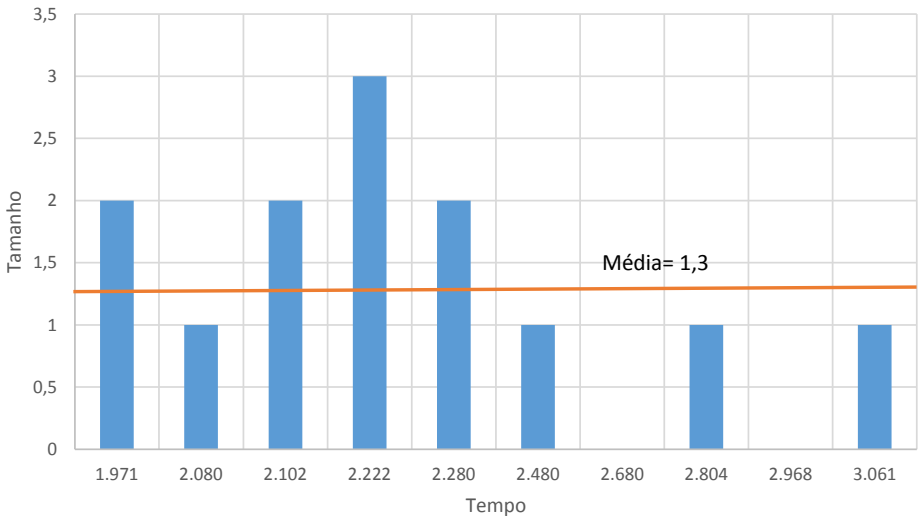


Figura 4.23: *Tamanho máximo e médio do slot S8 no cenário 4*

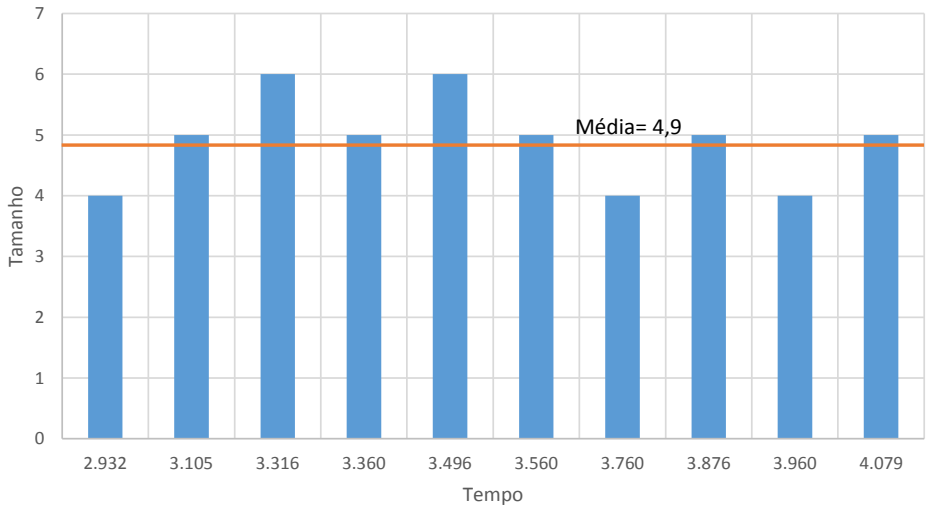


Figura 4.24: *Tamanho máximo e médio do slot S14 no cenário 4*

primeiros *slots*.

A formação de filas torna-se um gargalo quando apresenta um acúmulo desproporcional ao estado geral do sistema. Para uma melhor análise

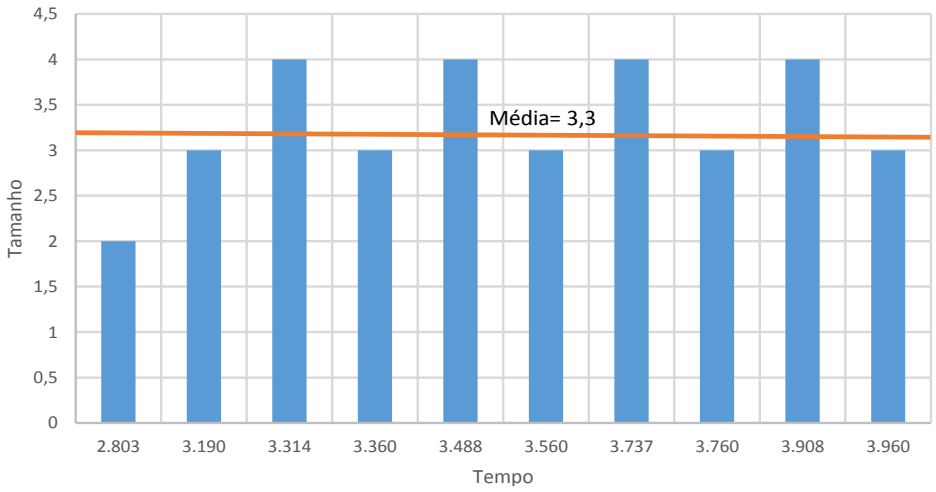


Figura 4.25: *Tamanho máximo e médio do slot S16 no cenário 4*

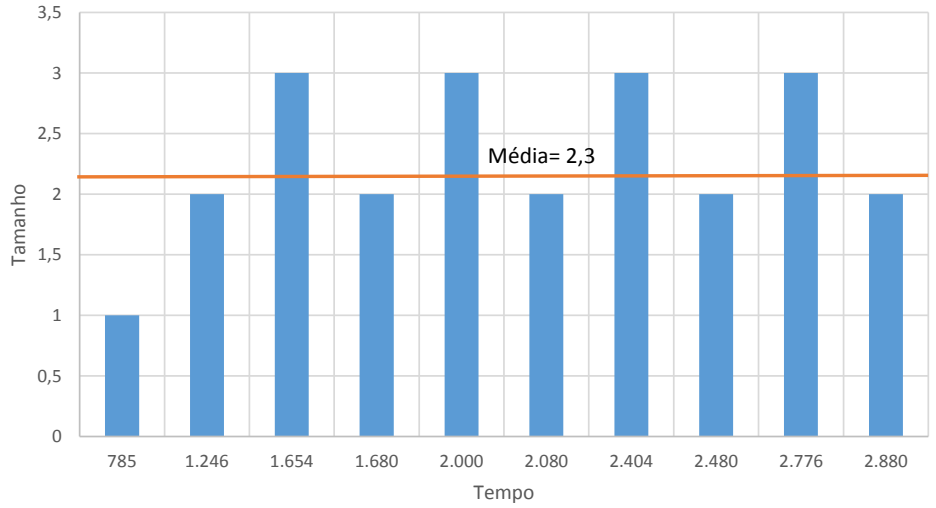


Figura 4.26: *Tamanho máximo e médio do slot S6 no cenário 5*

da formação das filas, foi utilizada a representação por gráficos dos lugares que sofreram acúmulos de mensagens pertencentes ao modelo de simulação e o respectivo número máximo e médio de *tokens* acumulados ao final do experimento. Foram organizados gráficos, dos diferentes cenários, considerando as mensagens de 20 kB em que foram constatados esses acúmulos.

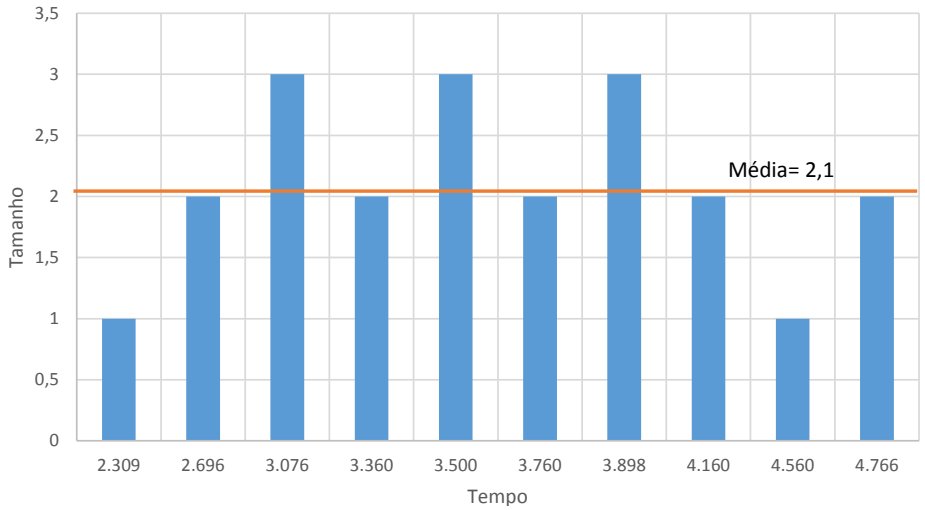


Figura 4.27: *Tamanho máximo e médio do slot S14 no cenário 5*

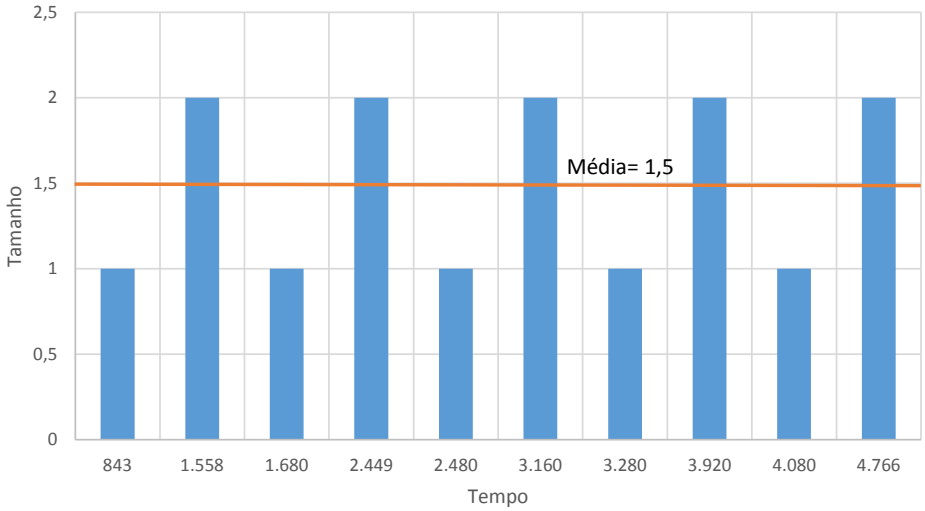


Figura 4.28: *Tamanho máximo e médio do slot S6 no cenário 6*

Destaca-se que, para estes cenários, também foram observados acúmulos de *tokens*, quando as mensagens são de outros tamanhos, porém o acúmulo é mais expressivo para as mensagens de 20 kB.

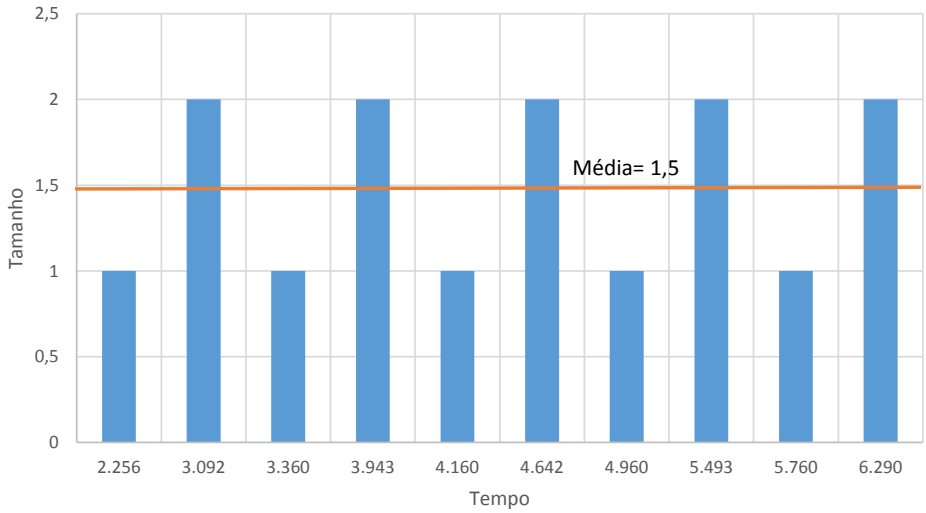


Figura 4.29: *Tamanho máximo e médio do slot S14 no cenário 6*

4.3 Verificação e Validação

O desenvolvimento de um modelo de simulação requer que o mesmo seja representativo do sistema real, seguro e livre de erros da implementação computacional. Assim, torna-se importante observar duas etapas: a etapa de verificação e validação do modelo respectivamente. Para realizar a verificação e a validação de um modelo são utilizadas técnicas formais descritas na literatura. A Seção §4.3.1 apresenta a definição de verificação e validação. Na Seção §4.3.2, são apresentadas as técnicas de verificação e validação encontradas na literatura. A Seção §4.3.3 apresenta a verificação do modelo de simulação formal proposto. Por fim, a Seção §4.4 apresenta o resumo do capítulo.

4.3.1 Definição de Verificação e Validação

O modelo de simulação imita o funcionamento do sistema real sendo utilizado para estudar o comportamento e prever o desempenho do sistema. Desse modo, é imprescindível estar seguro a respeito do desenvolvimento de um modelo de simulação, para que o mesmo seja implementado corretamente. Isso significa estar livre de erros de sintaxe e/ou lógica e ter acurácia

suficiente para ser utilizado como substituto do sistema, para a análise e experimentação. Neste contexto, o processo de validação e verificação de modelos de simulação são fundamentais para que um estudo de simulação seja bem-sucedido, sendo importante discutir cada um destes termos [59].

A verificação é realizada tendo como finalidade encontrar e corrigir erros de modelagem. Assim, são utilizadas técnicas para assegurar que o modelo está correto e corresponde aos seus pressupostos. Sob esta perspectiva, é possível dizer que a verificação consiste em certificar que o modelo está livre de erros de implementação computacional [59].

Um modelo é uma representação do mundo real ou pelo menos parte dele. Portanto, a validação de um modelo é realmente muito direta em princípio. Deve-se checar se o modelo comporta-se como o mundo real sob as mesmas condições. Se ele se comporta, então o modelo é válido, caso contrário, não é válido. Neste sentido, a validação refere-se às técnicas utilizadas para assegurar que o modelo, apesar dos pressupostos, representa, de fato, o sistema real [11].

4.3.2 Técnicas de Verificação e Validação

Para realizar a verificação do modelo de simulação, é imprescindível analisar as técnicas de verificação oriundas da literatura. As técnicas de verificação têm amparo e são abordadas pelos autores Kleijen [37] e Sargent [55].

As técnicas de verificação e validação, propostas por Kleijen [37], denotam uma pesquisa sobre como verificar e validar modelos de simulação mediante a aplicação de técnicas estatísticas, ou seja, o tipo de técnica aplicada depende da disponibilidade de dados sobre o sistema real. Para verificar o modelo, é necessário conhecimento especializado e, é por meio do conhecimento de especialistas no que tange o comportamento do sistema, que a verificação implicará. A Figura §4.30 apresenta o diagrama do processo de verificação e validação de modelos. A proposta de modelagem das soluções de integração tem por objetivo a identificação de gargalos de desempenho antes da implementação.

Um exemplo cotidiano de um sistema são as filas de espera em estabelecimentos comerciais, como em padarias. Sabe-se, em teoria, que o aumento da chegada de clientes por hora resultará no aumento do tempo de espera, havendo um crescimento da fila, a não ser que seja disponibilizado mais caixas para atender. O exemplo do sistema de atendimento da padaria permite um conhecimento qualitativo. Para obter conhecimento quantitativo,

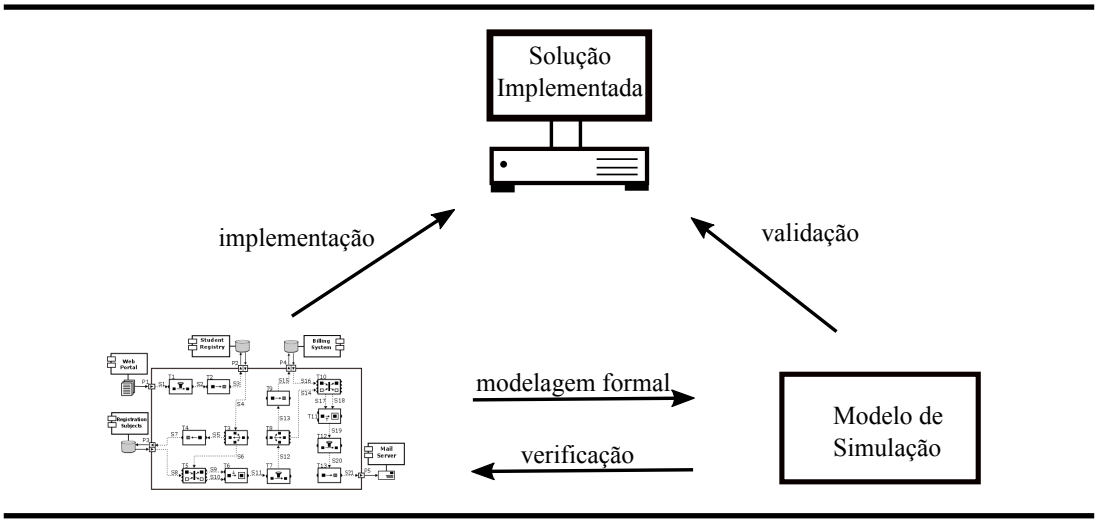


Figura 4.30: *Diagrama Verificação e Validação.*

um modelo de simulação é desenvolvido e seu efeito já é conhecido, porém, não a sua magnitude. A análise estatística ocorrerá diante de resultados quantitativos resultantes das simulações [7].

O conhecimento qualitativo é utilizado para representar as características que os especialistas esperam que ocorram no sistema simulado. O comportamento do sistema diante de algumas situações já é conhecido, e vai interferir nos resultados dos modelos de simulação que devem seguir na mesma linha. Com a utilização de dados numéricos provenientes das simulações, é possível aplicar uma análise quantitativa e estimar estatisticamente a dimensão das suas estruturas. Assim, os resultados experimentais devem acompanhar o comportamento que os especialistas esperam tanto no sistema como um todo, como de um determinado elemento [7].

Uma observação que é importante e pode ser realizada é quanto aos valores de entrada e saída do modelo, que contribuem para uma melhor verificação. Assim, se os valores de entrada e saída do modelo de simulação violam as características qualitativas, torna-se necessário questionar o modelo quanto a possíveis erros de modelagem ou se as condições ou cenários realmente estão adequados [7].

Sargent [55] propõe diferentes técnicas de verificação. Porém, é possível a utilização de apenas duas, pois estas estão de acordo com a situação de modelagem proposta: a técnica de Verificação do Modelo Computacional e a técnica de Validade de Eventos.

A técnica de Verificação do Modelo está associada ao modelo de simulação. O modelo de simulação é representado de acordo com o sistema que ele modela, utilizando para isso, uma representação específica. Neste caso, a verificação responsabiliza-se em garantir que o modelo de simulação seja implementado de forma correta na ferramenta. Nessa perspectiva, essa técnica propõe a comparação do número de elementos que entrou no sistema com a saída [7].

Com relação à técnica de Validade de Eventos, a mesma está relacionada à ocorrência de eventos no modelo conceitual quando comparados a ocorrência dos mesmos no modelo de simulação. Neste contexto, para que ocorra a verificação do modelo, é necessário que a taxa de ocorrência de determinado evento se aproxime quando realizados experimentos com o modelo de simulação [7].

Já, em consideração as técnicas de validação encontradas na literatura [6], levam em consideração: (a) as simplificações e pressupostos adotados; (b) valores dos parâmetros de entrada e as distribuições; e (c) resultados e conclusões. Assim, cada elemento desses pode ser submetido a um teste de validade, levando em consideração as seguintes técnicas:

- Teste de Turing: está técnica esta associada a dois resultados, um do sistema real e o outro do próprio modelo de simulação. Dessa forma, se o especialista não conseguir diferenciar os relatórios oriundos do modelo de simulação dos advindos do sistema real, o modelo pode ser considerado válido;

- Comparação com outros modelos: como o próprio nome já diz, essa técnica compara os vários resultados do modelo de simulação com outros modelos já validados;

- Medições obtidas em sistemas reais: esta técnica faz comparações de dois conjuntos de observações, um oriundo do modelo e outro do sistema real;

- Intuição de especialistas: esta técnica consiste em realizar reuniões entre pessoas conhecedoras do sistema. Todos os assuntos oriundos do desenvolvimento do modelo são discutidos. Geralmente, são validados três aspectos separadamente: (1) pressupostos; (2) entrada; e (3) resultados. É importante destacar que não é necessário esperar até o final do desenvolvimento do modelo, é possível validar antes, conforme o modelo avança.

Ainda, na literatura, é possível encontrar diversos trabalhos relacionados à verificação e validação de modelos de simulação, quando há dados do sistema real. Ao contrário destes, são mais escassos os trabalhos que relatam técnicas verificação e validação de modelos sem dados reais. A modelagem do modelo conceitual, proposto nesta pesquisa, não apresenta dados do sistema real [7]. Salienta-se que, nesta pesquisa, não ocorrerá a implementação do modelo, com isso, também não haverá dados reais para que se realize a validação.

4.3.3 Verificação do Modelo de Simulação

Para a verificação do modelo de simulação, utilizou-se a técnica de Kleijen [37], a qual refere-se à expectativa de comportamento do sistema, determinada por um especialista da área. Para um especialista, uma solução de integração é um sistema de mensagem, composto por unidades de armazenamento temporário denominado *slot*, no qual se espera que haja acúmulo de mensagens. A tarefa correlacionador (T5) é alimentada pelos *slots* S6 e S8. O *slot* S6 é alimentado pela tarefa replicador (T3), já S8 depende do processamento da tarefa T4 e da porta P3. Dessa maneira, espera-se que haja um maior acúmulo de mensagens em S6, o qual armazena mensagens para serem correlacionadas com as mensagens de S8 pela tarefa T5. A tarefa correlacionador (T10) é alimentada pelos *slots* S14 e S16. O *slot* S14 é alimentado pela tarefa replicador (T3), já S16 depende do processamento da tarefa T9 e da porta P4. Dessa forma, espera-se que haja acúmulo de mensagens em S14, o qual armazena mensagens para serem correlacionadas com as mensagens de S16 pela tarefa T10. A função das tarefas T5 e T10 é correlacionar as mensagens, neste caso, torna S6 e S14 supostos gargalos de desempenho. De acordo com o experimento realizado com o modelo de simulação, é possível afirmar que, ao final da simulação, foi gerado um acúmulo de *tokens* nos seus lugares. Pode-se observar que, nos lugares S6 e S14, ocorreram acúmulos de mensagens, em comparação com os demais lugares.

Seguindo a proposta de Kleijen [37], pode-se comparar as características esperadas com as encontradas nos experimentos, a qual constata que o acúmulo de mensagens esperado nos *slots* se repetiu nos lugares, confirmando que o modelo de simulação se comportou próximo de um sistema orientado a mensagem. O número mais expressivo de mensagens era esperado nos *slots* que alimentam a tarefa T5, neste caso, o *slot* S6, o que se constatou quando houve acúmulo nos lugares adjacentes a transição T5, neste caso, o lugar S6. De fato, o acúmulo no lugar S6 foi em média maior que os demais, comportamento que era esperado para o *slot* S6.

Um número expressivo de mensagens também era esperada nos *slots* que alimentam a tarefa T10, neste caso, o *slot* S14, o que se constatou quando houve acúmulo nos lugares adjacentes a transição T10, neste caso o lugar S14. De fato, o acúmulo no lugar S14 foi em média maior que os demais, comportamento que era esperado para o *slot* S14. Dessa forma, é evidente que o comportamento do experimento e o esperado pelos especialistas é muito próximo, demonstrando que há semelhança entre comportamentos.

Conforme Sargent [55], a proposta de verificação do modelo é aplicada ao próprio modelo de simulação. Levando isso em consideração, é proposto um experimento com 10.000 *tokens*, como apresenta a Figura §4.31. Este experimento foi executado uma vez, e sua intenção é descrever o fluxo de mensagens com intuito de verificar se o modelo está implementado corretamente na ferramenta.

O modelo foi divido em 4 setores:

- Antes da transição TA;
- Entre a transição TA e a TAA;
- Entre a transição TAA e a TAAA;
- Depois da transição TAAA.

Como os *tokens* que estão antes do TA não foram filtrados, basta contar a quantidade de *tokens*. Como os *tokens* que estão entre TA e TAA passaram por um filtro, é necessário contá-los e multiplicar o resultado pelo inverso de 0,95. Como os *tokens* que estão entre TAA e TAAA passaram por dois filtros, é necessário contá-los e multiplicar o resultado pelo inverso de 0,9095. Como os *tokens* que estão depois de TAAA passaram por três filtros, é necessário conta-los e multiplicar o resultado pelo inverso de 0,857375, originando a Tabela §4.1. Isso deve resultar em um valor aproximado a 10.000, porque os filtros são probabilísticos, por isso não há como garantir que a quantidade será exata.

Assim:

Somatório = 3,15+3,32+9976,96

Somatório = 9983,43

Uma observação importante é de que o valor do somatório não resulta exatamente em 10.000, porque tem-se uma função probabilística nos filtros.

Setores	Valores
Antes de TA	0
Entre TA e TAA	$3* \frac{1}{0,95} = 3,15$
Entre TAA e TAAA	$3* \frac{1}{0,9025} = 3,32$
Depois de TAAA	$8554* \frac{1}{0,857375} = 9976,96$

Tabela 4.1: *Verificação da Tarefa Filtro*

4.4 Resumo do Capítulo

Nesse capítulo, foi descrita a experimentação, por meio da simulação. O cenário proposto teve como carga de processamento 10.000 mensagens recebidas pelo sistema. O modelo foi implementado em uma ferramenta de simulação, sendo determinadas variáveis para análise do comportamento da solução de integração modelada. A ferramenta de simulação utilizada para a análise do modelo foi o *CPN Tools*, que é uma ferramenta de modelagem, análise e simulação de Redes de Petri Coloridas e Temporizadas.

Os resultados experimentais estão organizados em forma de gráficos, de acordo com cada cenário e variável que foi analisado. Ainda, abordou-se a definição de verificação e validação de modelos de simulação e também de algumas técnicas presentes na literatura. A possível aplicação de uma ou mais técnicas de verificação garante que o modelo está livre de erros da implementação computacional. Já, o uso de uma ou mais técnicas de validação procura garantir que o modelo tenha um comportamento semelhante ao do sistema modelado. Nesse sentido, a verificação do modelo de simulação formal proposto ocasionou segurança para a realização desta dissertação. Como a pesquisa visa analisar os resultados da simulação, ainda na fase de projeto, não realizou-se a validação.

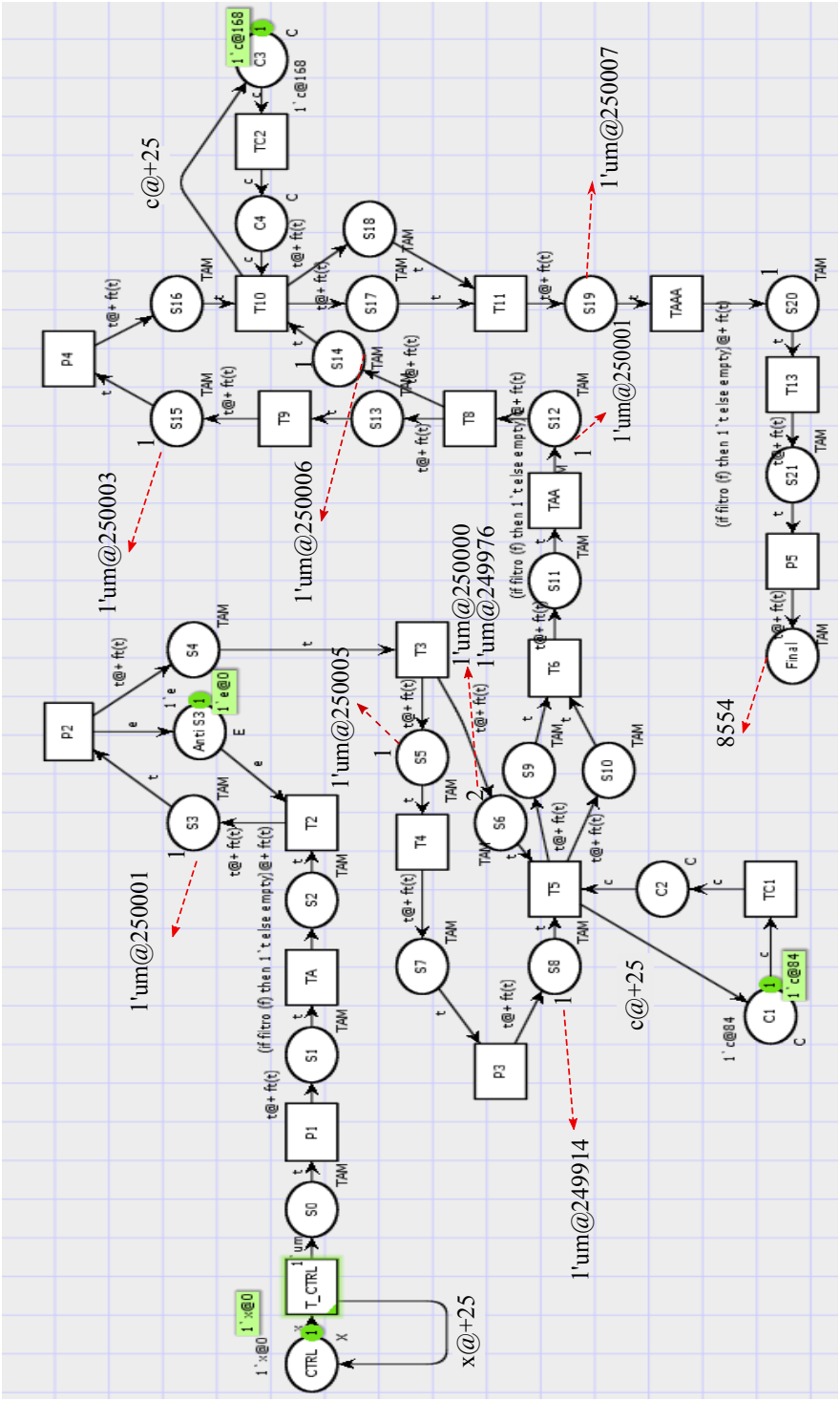


Figura 4.31: *Verificação do Modelo de Simulação*

Capítulo 5

Conclusão e Trabalhos Futuros

Todo mundo é um gênio, mas se você julgar
um peixe por sua capacidade de subir em uma
árvore, ele passará toda a sua vida acreditando
ser estúpido.

Albert Einstein

Frequentemente, as empresas adquirem ou desenvolvem novas aplicações para apoiar a tomada de decisões e aperfeiçoar seus processos de negócio. As empresas possuem o seu próprio ecossistema de *software*, o qual, geralmente, é heterogêneo, sendo composto por diferentes aplicações. Grande parte dessas aplicações foram projetadas não levando em consideração a possibilidade de serem reutilizadas. Nesse sentido, o objetivo de uma solução de integração é manter em sincronia os dados e as funcionalidades a partir daquelas já existentes, de tal forma que aplicações não sejam alteradas pela solução. Para fazer as aplicações colaborarem entre si, as empresas podem utilizar soluções de integração. O seu correto funcionamento ocasiona sucesso, agilidade e dinamismo nos processos de negócios das empresas. A análise do comportamento e a identificação de possíveis gargalos de desempenho em soluções de integração de aplicações geralmente envolve sua implementação para posterior execução e testes. Como as soluções são construídas, isso envolve custos (tempo, recursos) e riscos de falhas que costumam ser elevados e que podem comprometer o correto funcionamento das soluções de integração em situações em que tenham que processar uma grande demanda de informações.

A busca por soluções de integração confiáveis e de qualidade prevê uma imprescindível análise do seu comportamento. Conforme a revisão da literatura, uma solução de integração de aplicações empresariais pode ser caracterizada como um sistema, cujo o modelo é classificado como estocástico, dinâmico e discreto. A classificação da solução de integração utilizada nesta pesquisa obedece as características de um sistema de eventos discretos. Portanto, foi possível a utilização de um modelo matemático aliado a técnicas de simulação de eventos discretos para analisar o comportamento de uma solução de integração que trata do processo de rematrículas da Universidade Unijuí, confirmando-se a hipotése inicial. Porém, antes de realizar a simulação para a obtenção dos dados, foi necessário verificar o modelo. Neste sentido, foram aplicadas técnicas de verificação formais, visando assegurar que o modelo está livre de erros de implementação computacional, representando fielmente a solução de integração.

Após a verificação do modelo, realizou-se a simulação na ferramenta *CPN Tools*, a qual permite a modelagem, análise e simulação de sistemas de eventos discretos. Para realização da simulação, foi necessário partir de um modelo conceitual da solução de integração, ou seja, o caso de estudo. Após, realizou-se a transcrição dos elementos para as Redes de Petri Coloridas e Temporizadas e propôs-se um modelo formal de simulação, o qual foi submetido a experimentos para obtenção de dados. A partir da simulação, foi possível analisar duas variáveis: tempo médio de permanência das mensagens nos *slots* e tamanho máximo e médio dos *slots*. Os resultados da simulação para estas duas variáveis foram interpretados e analisados, identificando-se a ocorrência de possíveis gargalos de desempenho, em diferentes cenários.

Quanto a análise da variável tempo médio de permanência das mensagens nos *slots*, pode-se observar que quando incrementa-se o tempo de entrada das mensagens no sistema, consequentemente, fornece um intervalo maior de tempo para o processamento das mensagens nas transições. Além disso, as mensagens de tamanho maior levam mais tempo para serem processadas com relação as menores. A partir disso, concluí-se que a formação de gargalos é mais evidente quando o intervalo entrada é menor. Os gargalos estão localizados nos primeiros *slots* pelo fato da porta P2 ser síncrona e nos *slots* dos correlacionadores. Dessa forma, o tempo de espera torna-se um gargalo quando apresenta um acúmulo desproporcional de mensagens aguardando para serem processadas em comparação ao estado geral do sistema.

Já, com relação a análise da variável tamanho máximo e médio dos *slots*, pode-se observar que quando o intervalo de entrada das mensagens é menor, mais tendem a se acumularem as mensagens nos primeiros *slots* S2

(porta síncrona) e nos *slots* S6 e S14 (correlacionadores). Quando aumenta-se o intervalo de entrada das mensagens a tendência é a diminuição de mensagens acumuladas nos primeiros *slots*. E ainda, as mensagens maiores tendem a se acumularem mais que as menores (levam mais tempo para serem processadas). A partir disso, concluí-se que a possível formação de gargalos é mais evidente quando o intervalo entrada é menor, pois, há mais acúmulos de mensagens nos *slots* e que o tamanho da mensagem influência também na formação destes gargalos de desempenho, pois, leva-se mais tempo para processá-las. Novamente, os gargalos foram localizados nos primeiros *slots* pelo fato da porta P2 ser síncrona e nos *slots* dos correlacionadores (T5 e T10). Assim, a formação de filas torna-se um gargalo quando apresenta um acúmulo desproporcional ao estado geral do sistema.

O estudo destas variáveis permitiu a análise do comportamento da solução em diferentes cenários e sugerem a ocorrência de gargalos de desempenho na solução de integração. Para esta pesquisa, o uso da simulação ainda na fase de projeto mostra-se importante, pois a partir dela foi possível identificar em quais cenários e condições tendem a ocorrerem gargalos de desempenho. Sendo assim, pode-se afirmar que é imprescindível para melhorar a solução o uso da simulação antes da implementação.

Como trabalhos futuros sugere-se a análise de outras variáveis que poderiam auxiliar na identificação de gargalos de desempenho e/ou melhorar a solução de integração, como: prioridades nas transições, tempo médio de realização de uma determinada tarefa, utilização de funções aleatórias que permitam a descrição da intensidade de tráfico ou da taxa de mensagens enviadas em um sistema, determinar quantas vezes uma dada transição disparou, entre outras. Além disso, pode-se analisar esta solução de integração com a utilização de outras extensões do formalismo matemático das Redes de Petri, como por exemplo, as Redes de Petri estocásticas e as hierarquicas, pois deste modo, poderia ser observado o comportamento da solução de integração, quanto aos gargalos de desempenho nestes diferentes tipos de rede. E ainda, visto a grande variedade de ferramentas de simulação para Redes de Petri e as diferentes extensões destas para a modelagem de sistemas de eventos discretos, sugere-se como trabalho futuro o estudo de outras ferramentas de simulação.

Bibliografia

[1] J. P. Barros. *CpPNeTS: uma classe de Redes de Petri de alto-nível: implementação de um sistema de suporte à sua aplicação e análise*. Tese Doutoral, 1996.

[2] N. M. C. Barroso, J. M. Soares, G. C. Barroso, J. C. M. Mota e H. B. Neto. *Modelagem de conceitos e processos matemáticos por Redes de Petri Coloridas: o caso da integrabilidade de funções reais*. *Boletim de Educação Matemática*, 27(45):75–95, 2013.

[3] J. Bosch. *From software product lines to software ecosystems*. Em *Proceedings of the 13th International Software Product Line Conference*, páginas 111–119, 2009.

[4] K. R. Braghetto. *Técnicas de modelagem para a análise de desempenho de processos de negócio*. Tese Doutoral, Universidade de São Paulo, 2011.

[5] G. Bressan. *Modelagem e simulação de sistemas computacionais. Capítulo sobre Redes de Petri, LARC-PCS/EPUSP, www. larc. usp. br/conteudo/universo/pcs012/modsim05. pdf*, 2002.

[6] P. N. D. Bukh e R. Jain. *The art of computer systems performance analysis, techniques for experimental design, measurement, simulation and modeling*. *Informs*, 22(4):113–115, 1992.

[7] R. S. Cargnin. *Modelagem e simulação de uma solução de integração para identificação de gargalos de desempenho baseadas em formalismo matemático: uma abordagem orientada à Redes de Petri*. Dissertação de Mestrado, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2016.

[8] H. J. R. Carvalho, A. R. Yoshizawa, H. L. J. Pontes e A. J. V. Porto. *Análise de desempenho do trabalho multifuncional em linhas de produção, em forma de U pela modelagem e simulação usando Redes*

de Petri Temporizadas. Em *XXXVII Simpósio Brasileiro de Pesquisa Operacional*, páginas 109–121, 2005.

[9] C. G. Cassandras e C. G. Panayiotou. *Concurrent sample path analysis of discrete event systems*. *Discrete Event Dynamic Systems*, 9(2): 171–195, 1999.

[10] L. Chwif. *Redução de modelos de simulação de eventos discretos na sua concepção: uma abordagem causal*. Dissertação de Mestrado, Universidade de São Paulo, 1999.

[11] L. Chwif e A. Medina. *Modelagem e simulação de eventos discretos, 4ª edição: Teoria e aplicações*, volume 4. Elsevier Brasil, 2014.

[12] S. M. B. B. Correa. *Probabilidade e estatística*. *Belo Horizonte: PUC Minas Virtuais*, 2003.

[13] D. O. da Penha, H. C. Freitas e C. A. P. S. Martins. *Modelagem de sistemas computacionais usando Redes de Petri: aplicação em projeto, análise e avaliação*. *Anais do IV Escola Regional de Informática RJ/ES, Rio das Ostras, RJ, Brasil*, páginas 26–28, 2010.

[14] C. M. de Oliveira Rodrigues. *Mapeando estruturas LSC em Redes de Petri Coloridas*. Dissertação de Mestrado, Universidade de Pernambuco, 2006.

[15] P. Dias. *Desenvolvimento de objectos de aprendizagem para plataformas colaborativas*. Em *Actas do VII Congreso Iberoamericano de Informática Educativa. Universidad de Monterrey, Monterrey*, páginas 3–12, 2004.

[16] M. P. dos Santos. *Introdução à simulação discreta*. *Rio de Janeiro: UERJ*, 1999.

[17] D. Dossot, J. D'Emic e V. Romero. *Mule in action*. Manning, 2014.

[18] T. Facchin e M. A. Sellitto. *Medição do inventário em processo e tempo de atravessamento em manufatura por modelagem em redes de petri e diagrama de resultados*. *Revista Gestão & Produção, São Carlos*, 15 (2):307–321, 2008.

[19] F. Figueiredo. *Estatística Descritiva e Probabilidades*. Escolar Editora, 2007.

[20] M. Fisher, J. Partner, M. Bogoevici e I. Fuld. *Spring Integration in action*. Manning Publications Co., 2012.

[21] J. W. Forrester. *Principles of systems wright allen press. Cambridge MA*, 1968.

[22] C. R. L. Francês. *Introdução às Redes de Petri. Laboratório de Computação Aplicada, Universidade Federal do Pará*, 2003.

[23] R. Z. Frantz, R. Corchuelo e J. González. *Advances in a DSL for application integration*. Em *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, páginas 54–66, 2008.

[24] R. Z. Frantz, R. Corchuelo e F. Roos-Frantz. *On the design of a maintainable software development kit to implement integration solutions*. *Journal of Systems and Software*, 111:89–104, 2016.

[25] R. Z. Frantz, A. M. R. Quintero e R. Corchuelo. *A domain-specific language to design enterprise application integration solutions*. *International Journal of Cooperative Information Systems*, 20(02):143–176, 2011.

[26] R. Z. Frantz. *Enterprise application integration: An easy-to-maintain model-driven engineering approach.*, volume 1. Universidade de Sevilla, 2013.

[27] B. Gladwin e K. Tumay. *Modeling business processes with simulation tools*. Em *Proceedings of the 26th Conference on Winter Simulation*, páginas 114–121, 1994.

[28] R. Gold. *Petri Nets in software engineering*. Working Papers, 2004.

[29] C. M. Grinstead e J. L. Snell. *Introduction to probability*. American Mathematical Soc., 2012.

[30] E. M. P. Guimarães e Y. D. M. Évora. *Sistema de informação: instrumento para tomada de decisão no exercício da gerência. Ciência da Informação, Brasília*, 33(1):72–80, 2004.

[31] I. Haugg, R. Frantz, F. Roos-Frantz e S. Sawicki. *Modelagem de uma Solução de Integração para o Processo de Rematrícula da Universidade UNIJUÍ*. Em *Salão do Conhecimento da UNIJUÍ*, páginas 1–5, 2015.

[32] G. Hohpe e B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[33] C. Ibsen e J. Anstey. *Camel in action*. Manning Publications Co., 2010.

[34] K. Jensen. *Coloured Petri Nets: basic concepts, analysis methods and practical use*, volume 1. Springer Science & Business Media, 2013.

[35] K. Jensen e L. M. Kristensen. *Coloured petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media, 2009.

[36] F. Junqueira e P. E. Miyagi. *Modelagem e simulação distribuída de sistema produtivo baseados em Rede de Petri. Sba: Controle & Automação Sociedade Brasileira de Automatica*, 20(1):1–19, 2009.

[37] J. P. C. Kleijen. *Validation of models: statistical techniques and data availability*. *31st Conference on Winter Simulation: Simulation - a Bridge to the Future*, páginas 647–654, 1999.

[38] A. R. Kraisig e F. C. Welter. *Comparação das Ferramentas PIPE2 e Oris2 quanto à Simulação*. Em *IV Seminário de Formação Científica e Tecnológica*, páginas 1–7, 2016.

[39] A. R. Kraisig, F. C. Welter, I. G. Haugg, R. Cargnin, F. Roos-Frantz, S. Sawicki e R. Z. Frantz. *Mathematical Model for Simulating an Application Integration Solution in the Academic Context of Unijuí University*. *Procedia Computer Science*, 100:407–413, 2016.

[40] A. R. Kraisig, F. C. Welter e R. Z. Frantz. *Framework de Comparação entre Ferramentas de Simulação*. Em *Salão do Conhecimento da UNIJUÍ*, páginas 1–5, 2016.

[41] K. C. Laudon e J. P. Laudon. *Sistemas de informação gerenciais: administrando a empresa digital*. Pearson Prentice Hall, 2005.

[42] A. M. Law e D. W. Kelton. *Simulation modeling and analysis, McGraw-Hill*. 1991.

[43] J. Lied e M. A. Sellitto. *Aplicação da modelagem por Redes de Petri para avaliação ocupacional de trabalhadores*. *Revista Produção Online*, 9(3): 489–510, 2009.

[44] D. S. Linthicum. *Enterprise application integration*. Addison-Wesley Professional, 2000.

[45] P. R. M. Maciel, R. D. Lins e P. R. F. Cunha. *Introdução às Redes de Petri e aplicações*. UNICAMP-Instituto de Computação, 1996.

[46] N. Marranghello. *Redes de Petri: conceitos e aplicações. São Paulo: DCCE/IBILCE/UNESP*, 2005.

[47] M. E. G. Martins. *Distribuição de probabilidades. WikiCiências*, 2013.

[48] V. M. M. Martins. *Integração de sistemas de informação: Perspectivas, normas e abordagens*. Dissertação de Mestrado, Universidade do Minho Guimarães, 2006.

[49] M. M. Miyagi, P. E. Miyagi e M. Kisil. *Modelagem e análise de serviços de saúde baseados em Redes de Petri interpretadas. Production*, 11 (2):23–39, 2001.

[50] R. J. Paul e D. W. Balmer. *Simulation modelling*. Chartwell-Bratt, 1993.

[51] R. S. Pressman. *Engenharia de software*. McGraw Hill Brasil, 2011.

[52] E. S. Ramos e J. M. P. de Oliveira. *Especificação e verificação formal de um modelo de STI-PBL por Redes de Petri Coloridas. Revista Brasileira de Informática na Educação*, 17(03):53–66, 2010.

[53] F. Roos-Frantz, M. Binelo, R. Z. Frantz, S. Sawicki e V. B. Fernandes. *Using Petri Nets to enable the simulation of application integration solutions conceptual models.* Em *International Conference on Enterprise Information Systems*, páginas 87–96, 2015.

[54] N. Sakurada e D. I. Miyake. *Aplicação de simuladores de eventos discretos no processo de modelagem de sistemas de operações de serviços. Gestão & Produção*, 16(1):25–43, 2009.

[55] R. G. Sargent. *Verification and validation of simulation models. Journal of Simulation*, 7(1):12–24, 2013.

[56] S. Sawicki, R. Z. Frantz, V. M. B. Fernandes, F. Roos-Frantz, I. Yevseyeva e R. Corchuelo. *Characterising enterprise application integration solutions as discrete-event systems*. Em *Handbook of Research on Computational Simulation and Modeling in Engineering*, páginas 261–288. IGI Global, 2016.

[57] R. M. S. Soeiro. *Engenharia Informática e de Computadores*. Dissertação de Mestrado, Universidade Técnica de Lisboa, 2009.

[58] I. Sommerville, S. S. S. Melnikoff, R. Arakaki e E. de Andrade Barbosa. *Engenharia de software*, volume 6. Addison Wesley, 2003.

[59] A. K. Wiesner. *Modelagem e simulação de uma solução de integração para identificação de gargalos de desempenho baseadas em formalismo matemático: uma abordagem orientada à Teoria das Filas*. Dissertação de Mestrado, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2016.

[60] M. C. Yamada, A. J. V. Porto e R. Y. Inamasu. *Aplicação dos conceitos de modelagem e de Redes de Petri na análise do processo produtivo da indústria sucroalcooleira. Revista Pesquisa Agropecuária Brasileira. Brasília*, 37(6):809–820, 2002.

Apêndice

As Tabelas §B.1, §B.2, §B.3, §B.4, §B.5 e §B.6 apresentam o tempo médio de permanência das mensagens nos *slots* nos diferentes cenários.

	1kB	5kB	10kB	15kB	20kB
S1	10	49,88	100,12	150,08	199,69
S2	10,03	58962,88	91822,77	103024,5	108512,4
S3	10,01	49,97	99,6	150,43	200
S4	9,99	50	100,26	149,78	200,02
S5	9,99	49,92	99,86	149,57	199,76
S6	35,67	150,58	299,6	448,9	597,79
S7	9,98	49,99	99,78	149,82	199,15
S8	15,7	50,71	100,08	149,96	199,19
S9	10,83	53,55	106,98	159,51	213,5
S10	10,83	53,55	106,98	159,51	213,5
S11	9,99	50,01	100,16	149,63	199,85
S12	10,02	49,85	100,31	149,75	198,36
S13	10,01	49,94	100,07	150,31	199,21
S14	33,79	151,89	301,49	450,97	599,99
S15	10	49,97	100,43	150	200,3
S16	13,78	52,02	101,16	151,14	201,16
S17	10,81	53,46	106,64	159,91	213,57
S18	10,81	53,46	106,64	159,91	213,57
S19	10,01	49,96	100,08	150,52	198,86
S20	10,01	50,07	99,81	150,24	199,84
S21	9,97	50,1	100,07	150,49	199,21

Tabela B.1: *Tempo médio de permanência das mensagens nos slots do cenário 1*

As Tabelas §B.7, §B.8, §B.9, §B.10, §B.11, §B.12 apresentam o tamanho máximo e médio dos *slots* no primeiro cenário, com 25 unidades de tempo.

	1kB	5kB	10kB	15kB	20kB
S1	10	49,94	99,94	150,02	200,13
S2	10	73,97	118187,9	162400	184653,6
S3	10	50,06	100,13	150,14	200,32
S4	10,04	49,93	99,82	150,05	200,48
S5	9,97	49,92	100,15	149,93	200,07
S6	35,96	169,8	301,45	449,93	599,72
S7	10,01	49,98	100,34	149,78	199,63
S8	15,99	69,93	101,04	150,45	200,27
S9	10,81	53,51	106,88	160,49	213,06
S10	10,81	53,51	106,88	160,49	213,06
S11	9,99	50,08	100,03	149,69	199,95
S12	9,98	50,02	100,05	149,84	199,96
S13	9,99	50,04	100,01	149,99	199,87
S14	33,69	165,84	303,63	452,05	601,33
S15	10,01	49,95	100,04	149,86	199,57
S16	13,7	65,87	103,66	152,44	202,15
S17	10,81	53,49	106,86	160,08	213,12
S18	10,81	53,49	106,86	160,08	213,12
S19	9,99	50,04	100,36	150,27	200,54
S20	10,01	50,05	99,8	149,6	201
S21	9,97	49,9	99,87	150,28	199,4

Tabela B.2: *Tempo médio de permanência das mensagens nos slots do cenário 2*

	1kB	5kB	10kB	15kB	20kB
S1	10,01	50,06	99,95	150,17	199,51
S2	10	49,98	140,74	149250,1	235746,2
S3	10,01	50,08	99,71	150,03	200
S4	10,02	50,11	99,98	150,16	199,84
S5	9,98	49,99	99,89	150,22	199,7
S6	35,83	174,87	338,45	455,59	601,01
S7	9,97	50,06	99,97	149,97	199,37
S8	15,87	74,82	138,63	155,47	202,07
S9	10,79	53,49	106,7	160,05	213,27
S10	10,79	53,49	106,7	160,05	213,27
S11	10,02	49,98	100,24	150,07	200,38
S12	10	50,02	100,03	149,49	199,95
S13	9,99	49,97	100,06	149,73	199,75
S14	33,61	166,09	330,4	461,87	607,84
S15	10	49,99	99,95	150,07	200,25
S16	13,63	66,14	130,42	162,16	207,92
S17	10,81	53,49	106,72	160,18	213,53
S18	10,81	53,49	106,72	160,18	213,53
S19	9,99	49,93	100,02	150,14	199,67
S20	10,02	49,96	100,09	149,83	200,28
S21	10,01	50,1	99,89	149,85	199,74

Tabela B.3: *Tempo médio de permanência das mensagens nos slots do cenário 3*

	1kB	5kB	10kB	15kB	20kB
S1	10,03	49,96	100,21	150,35	200,38
S2	9,98	50,06	99,91	151,8	291,43
S3	10	50,09	100,06	150,08	200,18
S4	10,01	50,01	100,06	149,85	199,84
S5	10,01	50,08	99,95	150,09	200,02
S6	35,67	174,46	349,64	519,79	673,92
S7	9,99	49,99	100,07	149,7	199,61
S8	15,67	74,39	149,62	220,05	274,36
S9	10,81	53,48	106,92	159,91	212,91
S10	10,81	53,48	106,92	159,91	212,91
S11	10,02	50,07	99,97	150,36	200,22
S12	10	49,95	100,28	149,9	199,88
S13	9,99	49,97	100,03	150,36	200,19
S14	33,59	166,22	330,74	495,47	659,79
S15	9,99	50,07	100,07	149,93	200,43
S16	13,6	66,18	130,65	195,2	259,28
S17	10,81	53,47	106,93	160,32	213,31
S18	10,81	53,47	106,93	160,32	213,31
S19	10,03	49,9	100,18	150,25	199,99
S20	9,96	49,91	99,85	150,12	200,16
S21	10,01	50	99,85	150,1	199,8

Tabela B.4: *Tempo médio de permanência das mensagens nos slots do cenário 4*

	1kB	5kB	10kB	15kB	20kB
S1	10,03	50,05	99,99	149,75	199,91
S2	9,98	49,99	99,95	149,86	200,72
S3	10	50,1	99,8	149,85	199,69
S4	10,01	50,12	100,01	150,01	199,95
S5	10,01	50,01	99,87	150,17	200,2
S6	35,67	175,36	350,19	523,43	700,01
S7	9,99	50,12	99,97	150,1	199,97
S8	15,67	75,24	150,37	223,2	299,84
S9	10,81	53,49	106,83	160,26	213,17
S10	10,81	53,49	106,83	160,26	213,17
S11	10,02	49,98	99,99	149,78	199,85
S12	10	50,04	99,9	149,71	199,78
S13	9,99	49,96	100,03	150,02	200,26
S14	33,59	165,82	331,88	498,23	661
S15	9,99	49,85	100,23	150,15	199,77
S16	13,6	66,01	131,63	198,05	261
S17	10,81	53,46	106,75	160,06	213,51
S18	10,81	53,46	106,75	160,06	213,51
S19	10,03	50,02	100,16	149,87	200,01
S20	9,96	49,99	99,89	149,99	199,98
S21	10,01	49,94	99,9	149,82	200,11

Tabela B.5: *Tempo médio de permanência das mensagens nos slots do cenário 5*

	1kB	5kB	10kB	15kB	20kB
S1	10,03	49,97	99,9	149,75	200,1
S2	9,98	49,92	100,01	150,22	199,77
S3	10	50,05	100,01	149,58	200,14
S4	10,01	49,82	99,93	150	199,94
S5	10,01	50	99,8	149,81	199,84
S6	35,67	174,72	347,95	523,92	695,9
S7	9,99	49,98	99,84	150,2	199,9
S8	15,67	74,74	148,31	223,93	296,16
S9	10,81	53,41	106,91	160,37	213,54
S10	10,81	53,41	106,91	160,37	213,54
S11	10,02	50,04	100,03	150,24	199,74
S12	10	49,97	100,12	149,81	199,9
S13	9,99	50,02	100,1	149,98	199,87
S14	33,59	166,36	331,4	498,61	661,4
S15	9,99	49,99	100,07	150,06	200,01
S16	13,6	66,35	131,23	198,6	261,54
S17	10,81	53,55	106,93	160,07	213,42
S18	10,81	53,55	106,93	160,07	213,42
S19	10,03	50	100,12	150,01	199,89
S20	9,96	50,02	99,92	150,46	199,87
S21	10,01	50,04	100,08	149,96	200,13

Tabela B.6: *Tempo médio de permanência das mensagens nos slots do cenário 6*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	**Med**	**Max**	**Med**	**Max**	**Med**	**Max**	**Med**	**Max**	**Med**
S1	1	0,4	3	2	5	4	8	6	10	7,99
S2	1	0,38	4483	2236,82	6972	3481,93	7864	3924,42	8219	4109,15
S3	1	0,38	1	1	1	1	1	1	1	1
S4	1	0,38	2	1	2	1,01	2	0,99	2	1
S5	2	0,38	2	1	2	1	2	0,99	2	1
S6	2	1,36	5	3,01	5	3	5	2,98	5	2,98
S7	2	0,38	3	1	3	1	2	0,99	3	0,99
S8	2	0,6	3	1,01	3	1	3	0,99	3	0,99
S9	1	0,41	3	1,07	3	1,07	3	1,06	3	1,06
S10	1	0,41	3	1,07	3	1,07	3	1,06	3	1,06
S11	1	0,38	3	1	3	1	3	0,99	3	0,99
S12	1	0,36	3	0,94	3	0,96	3	0,94	3	0,93
S13	1	0,36	3	0,95	3	0,95	3	0,95	3	0,94
S14	3	1,22	7	2,88	5	2,87	5	2,84	7	2,82
S15	1	0,36	3	0,95	3	0,96	3	0,94	3	0,94
S16	2	0,5	5	0,99	3	0,96	4	0,95	6	0,94
S17	1	0,39	3	1,01	3	1,01	4	1	6	1
S18	1	0,39	3	1,01	3	1,01	4	1	6	1
S19	1	0,36	3	0,95	3	0,95	4	0,94	6	0,93
S20	1	0,34	3	0,9	3	0,91	4	0,9	5	0,88
S21	1	0,34	4	0,9	4	0,91	4	0,9	5	0,88

Tabela B.7: *Tamanho máximo e médio dos slots do cenário 1*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	Med	Max	Med	Max	Med	Max	Med	Max	Med
S1	1	0,2	2	1	3	2	4	3	5	4
S2	1	0,19	4	1,41	4483	2239,4	6177	3087,22	7008	3509,17
S3	1	0,19	1	0,95	1	1	1	1	1	1
S4	1	0,19	2	0,95	2	1	2	1	2	1
S5	1	0,19	2	0,95	2	1	2	1	2	1
S6	1	0,69	5	3,23	5	3,01	5	2,99	5	2,99
S7	1	0,19	3	0,95	3	1	2	1	3	0,99
S8	1	0,31	3	1,33	3	1,01	3	1	3	1
S9	1	0,21	2	1,02	3	1,07	3	1,07	3	1,06
S10	1	0,21	2	1,02	3	1,07	3	1,07	3	1,06
S11	1	0,19	2	0,95	3	1	3	0,99	3	0,99
S12	1	0,18	2	0,9	3	0,95	3	0,95	3	0,94
S13	1	0,18	3	0,9	3	0,95	3	0,95	3	0,94
S14	2	0,61	6	3	7	2,87	7	2,86	6	2,83
S15	1	0,18	3	0,9	3	0,95	3	0,95	3	0,94
S16	1	0,25	4	1,19	5	0,98	5	0,96	6	0,95
S17	1	0,2	2	0,97	3	1,01	4	1,01	5	1
S18	1	0,2	2	0,97	3	1,01	4	1,01	5	1
S19	1	0,18	2	0,9	3	0,95	4	0,95	5	0,94
S20	1	0,17	2	0,86	3	0,9	3	0,9	6	0,91
S21	1	0,17	3	0,86	3	0,9	4	0,9	6	0,9

Tabela B.8: *Tamanho máximo e médio dos slots do cenário 2*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	Med	Max	Med	Max	Med	Max	Med	Max	Med
S1	1	0,1	1	0,5	2	1	2	1,5	3	2
S2	1	0,09	1	0,48	4	1,34	2822	1415,63	4491	2236,99
S3	1	0,09	1	0,48	1	0,95	1	1	1	1
S4	1	0,1	2	0,48	2	0,95	2	1	2	1
S5	1	0,09	2	0,48	2	0,95	2	1	2	1
S6	1	0,34	3	1,66	5	3,21	5	3,03	5	3
S7	1	0,09	2	0,48	3	0,95	3	1	2	1
S8	1	0,15	2	0,71	4	1,32	3	1,04	3	1,01
S9	1	0,1	1	0,51	2	1,01	2	1,07	3	1,06
S10	1	0,1	1	0,51	2	1,01	2	1,07	3	1,06
S11	1	0,1	1	0,48	2	0,95	3	1	3	1
S12	1	0,09	1	0,45	2	0,9	3	0,94	3	0,95
S13	1	0,09	1	0,45	3	0,9	3	0,94	3	0,95
S14	1	0,3	3	1,5	6	2,98	7	2,91	8	2,9
S15	1	0,09	2	0,45	3	0,9	3	0,95	3	0,95
S16	1	0,12	2	0,6	4	1,18	5	1,02	5	0,99
S17	1	0,1	1	0,48	2	0,96	2	1,01	3	1,02
S18	1	0,1	1	0,48	2	0,96	2	1,01	3	1,02
S19	1	0,09	1	0,45	2	0,9	3	0,95	3	0,95
S20	1	0,09	1	0,43	2	0,86	3	0,9	3	0,9
S21	1	0,09	1	0,43	2	0,85	3	0,9	3	0,9

Tabela B.9: *Tamanho máximo e médio dos slots do cenário 3*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	Med	Max	Med	Max	Med	Max	Med	Max	Med
S1	1	0,05	1	0,25	1	0,5	1	0,75	2	1
S2	1	0,05	1	0,24	1	0,47	2	0,72	6	1,39
S3	1	0,05	1	0,24	1	0,48	1	0,71	1	0,95
S4	1	0,05	1	0,24	2	0,48	2	0,71	2	0,95
S5	1	0,05	1	0,24	2	0,47	2	0,71	2	0,95
S6	1	0,17	2	0,83	3	1,66	4	2,46	5	3,21
S7	1	0,05	1	0,24	2	0,48	2	0,71	3	0,95
S8	1	0,07	1	0,35	2	0,71	3	1,04	3	1,3
S9	1	0,05	1	0,25	1	0,51	1	0,76	2	1,01
S10	1	0,05	1	0,25	1	0,51	1	0,76	2	1,01
S11	1	0,05	1	0,24	1	0,47	2	0,71	2	0,95
S12	1	0,05	1	0,23	1	0,45	2	0,67	2	0,9
S13	1	0,04	1	0,23	2	0,45	2	0,68	2	0,9
S14	1	0,15	2	0,75	3	1,49	5	2,23	6	2,97
S15	1	0,04	1	0,23	2	0,45	2	0,67	3	0,9
S16	1	0,06	1	0,3	2	0,59	3	0,88	4	1,17
S17	1	0,05	1	0,24	1	0,48	1	0,72	2	0,96
S18	1	0,05	1	0,24	1	0,48	1	0,72	2	0,96
S19	1	0,05	1	0,23	1	0,45	2	0,68	2	0,9
S20	1	0,04	1	0,21	1	0,42	2	0,64	2	0,85
S21	1	0,04	1	0,21	2	0,42	2	0,64	2	0,85

Tabela B.10: *Tamanho máximo e médio dos slots do cenário 4*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	Med	Max	Med	Max	Med	Max	Med	Max	Med
S1	1	0,03	1	0,13	1	0,25	1	0,37	1	0,5
S2	1	0,02	1	0,12	1	0,24	1	0,36	1	0,48
S3	1	0,02	1	0,12	1	0,24	1	0,36	1	0,48
S4	1	0,02	1	0,12	1	0,24	1	0,36	2	0,48
S5	1	0,02	1	0,12	1	0,24	1	0,36	2	0,48
S6	1	0,08	1	0,42	2	0,83	2	1,25	3	1,67
S7	1	0,02	1	0,12	1	0,24	1	0,36	2	0,48
S8	1	0,04	1	0,18	1	0,36	2	0,53	2	0,72
S9	1	0,03	1	0,13	1	0,25	1	0,38	1	0,51
S10	1	0,03	1	0,13	1	0,25	1	0,38	1	0,51
S11	1	0,02	1	0,12	1	0,24	1	0,36	1	0,48
S12	1	0,02	1	0,11	1	0,23	1	0,34	1	0,45
S13	1	0,02	1	0,11	1	0,23	1	0,34	2	0,45
S14	1	0,08	1	0,38	2	0,75	3	1,13	3	1,49
S15	1	0,02	1	0,11	1	0,23	1	0,34	2	0,45
S16	1	0,03	1	0,15	1	0,3	2	0,45	2	0,59
S17	1	0,02	1	0,12	1	0,24	1	0,36	1	0,48
S18	1	0,02	1	0,12	1	0,24	1	0,36	1	0,48
S19	1	0,02	1	0,11	1	0,23	1	0,34	1	0,45
S20	1	0,02	1	0,11	1	0,22	1	0,32	1	0,43
S21	1	0,02	1	0,11	1	0,22	1	0,32	1	0,43

Tabela B.11: *Tamanho máximo e médio dos slots do cenário 5*

	1 Kb		5 kB		10 kB		15 kB		20 kB	
	Max	Med	Max	Med	Max	Med	Max	Med	Max	Med
S1	1	0,01	1	0,06	1	0,12	1	0,19	1	0,25
S2	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S3	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S4	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S5	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S6	1	0,04	1	0,21	1	0,41	1	0,62	2	0,83
S7	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S8	1	0,02	1	0,09	1	0,18	1	0,27	1	0,35
S9	1	0,01	1	0,06	1	0,13	1	0,19	1	0,25
S10	1	0,01	1	0,06	1	0,13	1	0,19	1	0,25
S11	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S12	1	0,01	1	0,06	1	0,11	1	0,17	1	0,23
S13	1	0,01	1	0,06	1	0,11	1	0,17	1	0,23
S14	1	0,04	1	0,19	1	0,37	2	0,56	2	0,75
S15	1	0,01	1	0,06	1	0,11	1	0,17	1	0,23
S16	1	0,02	1	0,07	1	0,15	1	0,22	1	0,3
S17	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S18	1	0,01	1	0,06	1	0,12	1	0,18	1	0,24
S19	1	0,01	1	0,06	1	0,11	1	0,17	1	0,23
S20	1	0,01	1	0,05	1	0,11	1	0,16	1	0,21
S21	1	0,01	1	0,05	1	0,11	1	0,16	1	0,21

Tabela B.12: *Tamanho máximo e médio dos slots do cenário 6*

This document was typeset on April 17, 2017 using class RG-BOOK α2.14 for LaTeX$_{2\varepsilon}$. As of the time of writing this document, this class is not publicly available. Only members of The Distributed Group (TDG) and the Applied Computing Research Group (GCA) are allowed to typeset their documents using this class.