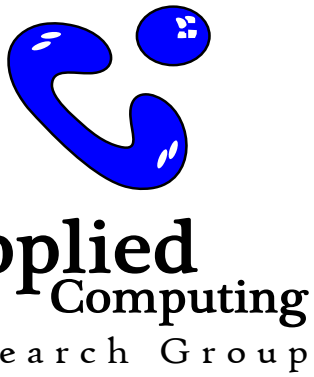
Modelagem e Simulação de uma Solução de Integração para Identificação de Gargalos de Desempenho Baseadas em Formalismo Matemático

Uma Abordagem Orientada à Cadeias de Markov

Márcia Maria Horn
Universidade Regional do Noroeste do Estado do Rio Grande do Sul

Dissertação de Mestrado

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



Fevereiro, 2016

First published in February 2014 by
Applied Computing Research Group - GCA
Department of Exact Sciences and Engineering
Rua Lulu Ilgenfritz, 480 - São Geraldo
Ijuí, 98700-000, Brazil.

Copyright © MMXII Applied Computing Research Group
http://www.gca.unijui.edu.br
gca@unijui.edu.br

In keeping with the traditional purpose of furthering science, education and research, it is the policy of the publisher, whenever possible, to permit non-commercial use and redistribution of the information contained in the documents whose copyright they own. You however are *not allowed* to take money for the distribution or use of these results except for a nominal charge for photocopying, sending copies, or whichever means you use redistribute them. The results in this document have been tested carefully, but they are not guaranteed for any particular purpose. The publisher or the holder of the copyright do not offer any warranties or representations, nor do they accept any liabilities with respect to them.

Financiamento: A pesquisa desenvolvida neste trabalho teve suporte da bolsa parcial de mestrado concedida pela UNIJUÍ.

Universidade Regional do Noroeste do Estado do Rio Grande do Sul

A Comissão Examinadora, abaixo assinada, _____ a dissertação intitulada "Modelagem e Simulação de uma Solução de Integração para Identificação de Gargalos de Desempenho Baseadas em Formalismo Matemático: Uma Abordagem Orientada à Cadeias de Markov", elaborada por Márcia Maria Horn, como requisito parcial para a obtenção do título de Mestre em Modelagem Matemática.

_____

Dr. Sandro Sawicki
UNIJUÍ
(Orientador)

_____

Dr. Rafael Zancan Frantz
UNIJUÍ
(Co-orientador)

_____

Dr. Daniel Welfer
UFSM

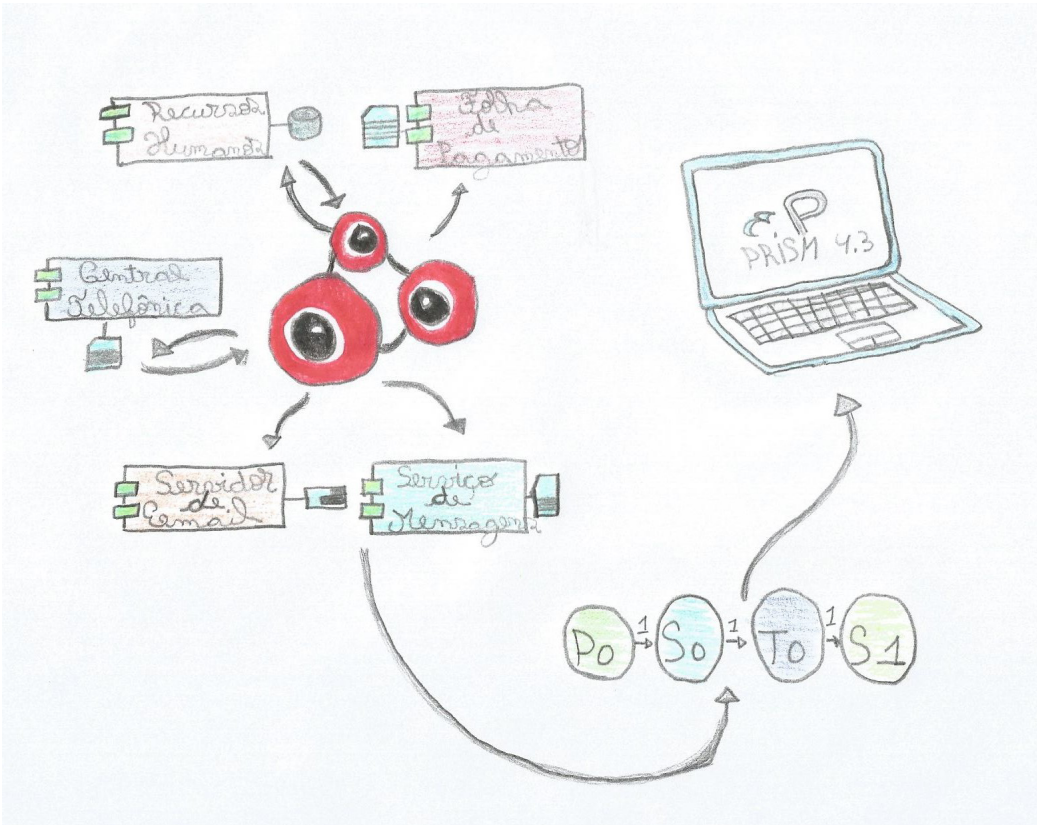
_____

Dra. Fabrícia C. Roos Frantz
UNIJUÍ

_____

Dr. Paulo Sérgio Sausen
UNIJUÍ

Ijuí, _____ de _____ de _____.



Modelo Conceitual Desenvolvido Através da tecnologia Guaraná para Representar uma Solução de Integração, técnicas matemáticas e simulação por Cristian, 9 anos de idade.

Dedico este trabalho a minha família em especial ao meu noivo Felipe Hanauer pelo apoio incondicional na realização desta pesquisa.

Conteúdo

Agradecimentos . . . . . **ix**

Resumo . . . . . **xi**

Abstract . . . . . **xiii**

I Apresentação da Dissertação

1 Introdução **3**
- 1.1 Contexto da Pesquisa 4
- 1.2 Motivação 12
- 1.3 Objetivos 13
  - 1.3.1 Geral 13
  - 1.3.2 Específicos 13
- 1.4 Metodologia 14
- 1.5 Resumo das Contribuições 15
- 1.6 Estrutura dessa Dissertação 16

II Revisão da Literatura

2 Referencial Teórico **19**
- 2.1 Tecnologia Guaraná 20
  - 2.1.1 Linguagem de domínio especifica Guaraná DSL 20
  - 2.1.2 Blocos de construção do Guaraná. 21
  - 2.1.3 Sintaxe Concreta. 22
  - 2.1.4 Modelo de Execução do Runtime System do Guaraná RT. 29

2.2 Classificação de Sistemas e Simulação de Modelos. 30
2.2.1 Modelo 32
2.2.2 Simulação 35
2.3 Cadeias de Markov 38
2.3.1 Definição 39
2.3.2 Propriedades das Cadeias de Markov de Tempo Discreto 40
2.4 Resumo do Capítulo 43

3 Trabalhos Relacionados 45
3.1 Verificação de um modelo de DTMC utilizando ferramentas de Simulação 46
3.2 Verificação do modelo de DTMC utilizando a ferramenta PRISM 48
3.3 Resumo do Capítulo 53

III Proposta e Experimentação

4 Desenvolvimento e Experimentação do Modelo Formal . 57
4.1 Problema de Integração 58
4.2 Ecossitema de Software 58
4.3 Solução de Integração 60
4.4 Equivalência dos elementos do Guaraná com Cadeias de Markov 62
4.5 Distribuição de Entrada 65
4.5.1 Poisson 65
4.5.2 Distribuição Exponencial Negativa 66
4.5.3 Distribuição de Erlang 66
4.6 Desenvolvimento do Modelo Formal 67
4.6.1 Variáveis a se Observar 67
4.6.2 Modelo de Simulação 69
4.7 Experimentos 74
4.7.1 Cenários submetidos 75
4.7.2 Ferramenta de Simulação 76
4.7.3 Modelo de Simulação na Ferramenta PRISM 78
4.7.4 Análise dos resultados experimentais 82
4.8 Resumo do Capítulo 94

5 Verificação 95

5.1 Introdução 96
5.2 Validação e Verificação 96
5.3 Técnicas de Validação e Verificação de Modelos de Simulação . . 99
5.4 Verificação do Modelo Proposto 102
5.5 Resumo do Capítulo 108

IV Considerações Finais

6 Conclusões e Trabalhos Futuros 111

V Anexos

A Gráficos dos Cenários 119

B Verificações efetuadas nos Cenários 125

Bibliografia 135

Índice de figuras

1.1 Exemplo de solução de integração 8

2.1 Modelo de execução do Guaraná baseado em tarefas (de Frantz et al. [22]) 29
2.2 Processos de uma solução de integração (de Frantz et al. [23]) 30
2.3 Abordagens para estudar um sistema (de Law e Kelton [38]) 32
2.4 Taxonomia de modelos (de Law e Kelton [38]) 33
2.5 Trajetória de um sistema dinâmico de variável contínua (SDVC). Fonte: Carvalho [10] 34
2.6 Trajetória de um sistema dinâmico de eventos discretos (SDED) 35
2.7 Fases de um estudo de simulação tradicional.(Adaptado de van der Aalst [58]) 38
2.8 Diagrama de transição de dois estados 42

4.1 Solução de Integração para o Problema de Gerenciamento de Chamadas Telefônicas. Fonte: Frantz [24]. 59
4.2 Modelo Conceitual Desenvolvido Através da Linguagem Guaraná para Representar uma Solução de Integração para o Problema de Gerenciamento de Chamadas Telefônicas. Fonte: Frantz [24]. 60
4.3 Modelo genérico de uma representação em Cadeias de Markov (a) com a equivalência no Guaraná. 64
4.4 Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov correspondente ao Modelo Conceitual. 70
4.5 Diagrama de transição entre dois estados da matriz de uma cadeia de Markov. Fonte: Dimuro et al. [17] 72
4.6 Diagrama de Transição Parcial do Modelo Conceitual 74
4.7 Comando PRISM 77
4.8 Uma captura de tela da interface da ferramenta PRISM 79
4.9 Média de Mensagens Acumuladas em cada *Slot* de cada cenário. . . . 83

4.10 Média de Mensagens Acumuladas no cenário de 10.000 e 100.000 mensagens. 84
4.11 Diagrama *Box-Plot* para identificação de *Outliers*. 85
4.12 *Outliers* referente aos *slots* S_0 e S_4 em cada cenário 86
4.13 Cenário com 10.000 mensagens no *slot* S_0 e S_4. 87
4.14 Cenário com 20.000 mensagens no *slot* S_0 e S_4. 88
4.15 Cenário com 30.000 mensagens no *slot* S_0 e S_4. 89
4.16 Cenário com 40.000 mensagens no *slot* S_0 e S_4. 89
4.17 Cenário com 50.000 mensagens no *slot* S_0 e S_4. 90
4.18 Cenário com 60.000 mensagens no *slot* S_0 e S_4. 91
4.19 Cenário com 70.000 mensagens no *slot* S_0 e S_4. 91
4.20 Cenário com 80.000 mensagens no *slot* S_0 e S_4. 92
4.21 Cenário com 90.000 mensagens no *slot* S_0 e S_4. 93
4.22 Cenário com 100.000 mensagens no *slot* S_0 e S_4. 93

5.1 Versão simplificada do processo de desenvolvimento do modelo. Fonte: Adaptado Sargent [55] 97

A.1 Média de Mensagens Acumuladas com entrada de 10.000 mensagens 119
A.2 Média de Mensagens Acumuladas com entrada de 20.000 mensagens 120
A.3 Média de Mensagens Acumuladas com entrada de 30.000 mensagens 120
A.4 Média de Mensagens Acumuladas com entrada de 40.000 mensagens 121
A.5 Média de Mensagens Acumuladas com entrada de 50.000 mensagens 121
A.6 Média de Mensagens Acumuladas com entrada de 60.000 mensagens 122
A.7 Média de Mensagens Acumuladas com entrada de 70.000 mensagens 122
A.8 Média de Mensagens Acumuladas com entrada de 80.000 mensagens 123
A.9 Média de Mensagens Acumuladas com entrada de 90.000 mensagens 123
A.10 Média de Mensagens Acumuladas com entrada de 100.000 mensagens 124

Índice de tabelas

2.1 Simbologia da sintaxe concreta (de Frantz et al. [23]) 23
2.2 Tarefas da categoria *Routers* (de Frantz et al. [23]) 25
2.3 Tarefas da categoria *Modifiers* (de Frantz et al. [23]) 26
2.4 Tarefas da categoria *Transformers* (de Frantz et al. [23]) 27
2.5 Tarefas da categoria *Stream Dealers* (de Frantz et al. [23]) 27
2.6 Tarefas da categoria *Mappers* (de Frantz et al. [23]) 28
2.7 Tarefas da categoria *Communicators* (de Frantz et al. [23]) 28

4.1 Principais variáveis utilizadas como medidas de desempenho do sistema [51] 68
4.2 Fórmulas que tratam as variáveis randômicas do modelo M/M/1 [51] 69
4.3 Média de Mensagens em cada slot de cada cenário 83

5.1 Dados Obtidos em Cada Simulação 103
5.2 Cenário com entrada de 10.000 mensagens no sistema 106

B.1 Cenário com entrada de 20.000 mensagens no sistema 125
B.2 Cenário com entrada de 30.000 mensagens no sistema 126
B.3 Cenário com entrada de 40.000 mensagens no sistema 127
B.4 Cenário com entrada de 50.000 mensagens no sistema 128
B.5 Cenário com entrada de 60.000 mensagens no sistema 129
B.6 Cenário com entrada de 70.000 mensagens no sistema 130
B.7 Cenário com entrada de 80.000 mensagens no sistema 131
B.8 Cenário com entrada de 90.000 mensagens no sistema 132
B.9 Cenário com entrada de 100.000 mensagens no sistema 133

Agradecimentos

Sempre é mais valioso ter o respeito,
que a admiração das pessoas.

Jean-Jacques Rousseau, filósofo, teórico político, escritor suíço.
(1712-1778)

Além de dedicar todo este trabalho a minha família, não posso deixar de agradecer ao meu noivo Felipe. Sou muito grato pela compreensão, mas principalmente pelo apoio que me deram durante o período em que estive envolvido nesta pesquisa.

Agradeço também ao meu professor orientador Dr. Sandro Sawicki. Sempre me ajudou muito para que o desafio fosse vencido com êxito.

O meu professor co-orientador professor Dr. Rafael Zancan Frantz. Amplo conhecedor do assunto, auxiliou muito para viabilizar esta pesquisa.

A professora Dra. Fabrícia Carneiro Roos Frantz. Sempre muito dedicada e preocupada em proporcionar condições para que o trabalho progredisse.

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

Aos demais professores do Mestrado.

Aos meus colegas.

Obrigada!

Resumo

O início de todas as coisas é pequeno.

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

É de senso comum que, para auxiliar seus processos de negócios, as empresas utilizam heterogêneas aplicações de *software*, em sua maioria, compostas de sistemas legados, pacotes de *softwares* adquiridos de terceiros ou sistemas desenvolvidos por sua própria equipe de programadores como solução para um problema específico. Neste aspecto, percebe-se que no momento que surge a demanda pela comunicação e interação entre tais aplicações, as soluções tornam-se adaptações, acarretando mais complexidade ao problema, insegurança, além de elevados custos de implantação e manutenção. Neste cenário, a área de integração de aplicações empresárias tornou-se fundamental para a gestão da informação, pois proporciona metodologias e ferramentas para projetar e implementar soluções de integração, sem afetar a estrutura de dados e aplicações existentes. Seu desenvolvimento seguem processos da área da engenharia de *software*, tipicamente incluem as fazes de especificação, projeto, implementação, teste e evolução. A tecnologia Guaraná é ferramenta que proporciona o desenvolvimento de soluções de integração possibilitando projetar soluções em um alto nível de abstração, utilizando uma sintaxe concreta gráfica e muito intuitiva. Este trabalho propõe analisar o comportamento e identificar gargalos de desempenho em soluções de integração de aplicações por meio do desenvolvimento de um modelo formal de simulação baseado em Cadeias de Markov. Utiliza-se, como estudo de caso, um sistema de gerenciamento de telefonemas pessoais implementado na Unijuí e desenvolvido pela tecnologia Guaraná. Os resultados experimentais demonstraram que é possível avaliar a qualidade de uma solução de integração ainda na fase de projeto, sem a necessidade de implementa-la. O modelo de simulação proposto foi validado por meio de técnicas de verificação formal.

Abstract

The beginnings of all things are small.

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

It is common sense that in order to assist their business processes, companies use heterogeneous software applications, mostly composed of legacy systems, third-party purchased software packages or systems developed by its own programmers teams as a solution for a specific problem. In this aspect it is realized that in the moment that communication and interaction between such applications come become necessary, solutions become adaptations, bringing more complexity to the problem, insecurity, and high deployment and maintenance costs. In this scenario, the area of enterprise application integration has become essential to the information management, it provides methodologies and tools to design and implement integration solutions without affecting the data structure and current applications. Its development follows engineering area softwareâs processes, typically it includes the specification phase, the design, the implementation, testing and evolution. The Guaraná technology is a tool that provides the development of integration solutions allowing design solutions at a high level of abstraction, using a very intuitive and concrete graphical syntax. This paper aims at analyzing the behavior and identifying bottlenecks performance in application integration solutions through the development of a formal model of simulation based on Markov chains. As a case study, one personal phone management system implemented in Unijuí and developed by Guaraná technology is used. The experimental results show that it is possible to evaluate the quality of an integration solution still in the design stage without the need to implement it. The proposed simulation model was validated using formal verification techniques.

Parte I

Apresentação da Dissertação

Capítulo 1

Introdução

Escuta e serás sábio.

O começo da sabedoria é o silêncio.

Pitágoras, Matemático, filósofo e político (575 a.C. - 490 a.C)

O presente trabalho tem como eixo central o estudo de uma solução de integração. Propõe-se a analisar e identificar gargalos de desempenho em solução de integração de aplicações empresariais, modeladas na tecnologia Guaraná, antes da sua implementação e execução, por meio de desenvolvimento de um modelo matemático em Cadeias de Markov, simulado e verificado em um modelo de simulação, na ferramenta PRISM. Neste contexto, o capítulo aborda, na Seção §1.1, o contexto desta pesquisa, seguido pela Seção §1.2 que discute a motivação deste trabalho. A Seção §1.3 descreve o objetivo geral e os objetivos específicos deste trabalho. A Seção §1.4 apresenta a metodologia seguida para alcançar os objetivos propostos. Na Seção §1.5 são descritas as contribuições resultantes deste trabalho. E por último, na Seção §1.6 é apresentada a estrutura desta dissertação.

1.1 Contexto da Pesquisa

Com o grande volume de informações sendo trocadas e disponibilizadas nos dias atuais, nas empresas, o seu gerenciamento, de forma correta, segura e íntegra, torna-se cada vez mais necessário. Sistemas de informação ganham grande importância nesse contexto, pois são capazes de atender a essas necessidades. Segundo Laudon e Laudon [37] um sistema de informação pode ser definido como um conjunto de componentes inter-relacionados que coletam (ou recuperam), processam, armazenam e distribuem informações destinadas a apoiar a tomada de decisões, a coordenação e o controle de uma organização. Além de dar apoio à tomada de decisões, à coordenação e ao controle, esses sistemas também auxiliam os gerentes e trabalhadores a analisar problemas, visualizar assuntos complexos e criar novos produtos.

Nesse contexto, o sistema de informação armazena um número muito grande de informações em pouco espaço físico, podendo fornecer uma grande quantidade de informações em pouco tempo. Isso facilita a busca dessas informações, que ocorrem a todo instante, auxiliando no trabalho das empresas. Segundo Batista [5], o objetivo de usar os sistemas de informação é criar um ambiente empresarial em que as informações sejam confiáveis e possam fluir na estrutura organizacional. Dessa forma, o sistema de informação garante que a tomada de decisões numa organização empresarial possa ocorrer em tempo real.

Além da definição apresentada por Laudon e Laudon [37], há na literatura outros autores que definem sistemas de informação abordando outras características desses sistemas. Para O'Brien et al. [43], sistema de informação é um conjunto organizado de pessoas, *hardware*, *software*, redes de comunicações e recursos de dados que coleta, transforma e dissemina informações em uma organização. A combinação de recursos humanos e computacionais, a criação de novos produtos e serviços e a visualização de questões complexas, abrange a administração do fluxo de informações geradas e distribuídas dentro de uma organização.

As empresas, para suprir as necessidades de gerenciamento em seus processos de negócios, possuem um sistema empresarial que pode ser definido como um programa ou *software* que auxilia a empresa em seu negócio, como no fluxo de informações, gerenciamento de relatórios e funcionamento geral. Cada empresa tem o *software* desenvolvido especialmente para suprir nas necessidades específicas. Por exemplo, uma concessionária de automóveis possui um sistema que fornece suporte ao gerenciamento de carros, um

hotel terá um sistema com suporte ao gerenciamento de reservas e diárias, e assim por diante.

As aplicações empresariais realizam funções lógicas com o auxílio de um *software*, como processamento de ordem, agendamento de produção, gerenciamento de energia e gerenciamento de informação do cliente. Normalmente são serviços hospedados em servidores para uma grande quantidade de usuários. Nesse sentido, para a realização desta pesquisa considera-se um sistema de informação como um *software* desenvolvido para apoiar os processos de negócios empresariais.

O desenvolvimento de *software* sofreu ao longo dos anos grandes modificações. Inicialmente era projetado sob medida para atender um problema específico, tendo uma distribuição relativamente limitada. Atualmente o *software* está presente em nosso cotidiano, segundo Pressman [52], a definição de *software* consiste em: (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados; (2) estrutura de dados que possibilitam aos programas manipular informações adequadamente; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Encontram-se, atualmente, vários processos de desenvolvimento de *software*. O processo de como fazer um *software* é detalhado na engenharia de *software*. A engenharia de *software* trata de aspectos relacionados ao estabelecimento de processos, métodos, técnicas, ferramentas e ambientes de suporte ao desenvolvimento de *software*.

De acordo com Pressman [52] a engenharia de *software* inclui um conjunto de atividades, tais como, um planejamento e estimativa de projeto, análise de requisitos de *software* e de sistemas, projeto da estrutura de dados, arquitetura de programas e algoritmo de processamento, codificação, teste e manutenção. Existem vários processos de desenvolvimento de *software* atualmente, porém algumas etapas são fundamentais a todos eles, tais como:

- **Especificação:** Define a funcionalidade do *software*, quais serviços serão fornecidos, e as restrições sobre sua operação e desenvolvimento. Nessa fase destina-se a tomada de decisões ao planejamento de como será conduzido o desenvolvimento, analisando a viabilidade da execução do *software*. Nessa etapa, são definidos os requisitos que o *software* deverá conter de acordo com as necessidades do cliente.

- **Projeto:** Envolve o desenvolvimento de sistemas em vários níveis de abstração. Desenvolve-se, nessa fase, os modelos conceituais das aplicações, sendo realizada uma descrição detalhada da estrutura do software

a ser implementado. Na fase de projeto, pretende-se descobrir os possíveis gargalos de desempenho que o sistema possa conter, antes de implementá-lo, por meio da simulação computacional.

- **Implementação:** É o processo de converter a especificação do sistema em um sistema executável. O *software* que atenda a especificação deve ser produzido mediante o uso de uma ou mais linguagens de programação. Nessa fase, o sistema é codificado ocorrendo à tradução da descrição computacional obtida a partir da fase de projeto em código executável.

- **Validação de *software*:** O *software* deve ser validado para garantir que ele atenda as especificações e faça o que o cliente deseja. Esta etapa envolve processos de verificação, revisão e teste do *software*.

- **Evolução:** Nesta fase se dá a descoberta de novos problemas e requisitos. Durante sua vida o *software* enfrenta diversas modificações (manutenção), quando essas modificações ocorrem é provável que novos defeitos sejam introduzidos. *Software* deve evoluir para atender aos novos requisitos que naturalmente poderá surgir.

Nas organizações empresariais, as aplicações, que neste contexto de pesquisa, caracterizam-se como *softwares* de aplicação são utilizadas para obter informações de forma eficaz. Tornando-se imprescindível para as empresas que desejam alcançar um alto nível de serviço e ter uma base sólida de dados que auxilie na tomada das decisões em seus processos de negócios.

Neste contexto, as organizações empresariais buscam incrementar, qualificar e dar suporte para os seus processos de negócios através da utilização de aplicações que compõem o seu ecossistema de *software*, Messerschmitt e Szyperski [39]. Um ecossistema de *software* é composto por um conjunto de aplicações, habitualmente heterogêneo, ou seja, podem ter sido desenvolvidas utilizando diferentes tecnologias, linguagens de programação, modelos de dados e sistemas operacionais. Esse ecossistema de *software* geralmente envolve várias aplicações complementares de fornecedores diferentes ou construídos na própria empresa, necessitam operar suas funcionalidades de forma integrada [39].

Geralmente as aplicações não foram desenvolvidas para operarem suas funções de forma integrada. A integração de aplicações é necessária quando uma aplicação isolada não é mais capaz de gerir ou suportar certos processos de negócio. Segundo Hohpe e Woolf [28], a integração de aplicações fornece

metodologias e ferramentas para desenvolver e implementar soluções de integração. A integração possibilita reutilizar aplicações já existentes, com o objetivo orquestrar essas aplicações para mantê-las sincronizadas ou proporcionar novas funcionalidades que possam ser construídas a partir daquelas já existentes [21].

A área de Integração de Aplicações Empresariais **EAI** (do inglês *Enterprise Application Integration*) utiliza técnicas e ferramentas computacionais para que as empresas possam integrar dados e funcionalidades oferecidas por distintas aplicações. Hohpe e Woolf [28] atribuem a uma solução de integração a tarefa de fazer com que as aplicações legadas sejam reutilizadas e todas as suas funcionalidades disponíveis. O objetivo de uma solução de integração é manter em sincronia uma série de dados e de funcionalidades das aplicações ou desenvolver novas funcionalidades em relação às existentes, de tal forma que as aplicações não sejam alteradas na solução de integração [28]. Na Figura §1.1 é apresentado um exemplo de solução de integração que engloba um conjunto de aplicações formando um ecossistema de *software*.

Para integrar aplicações, os engenheiros de *software* geralmente seguem um determinado estilo de integração. Os diferentes estilos de integração existentes levam em consideração alguns critérios, tais como acoplamento das aplicações, simplicidade na integração, tecnologia utilizada pelas aplicações e formato dos dados. Hohpe e Woolf [28] consideram dois estilos de integração: compartilhamento de dados e compartilhamento de funcionalidades.

O compartilhamento de dados ocorre quando uma aplicação deseja compartilhar informações presentes em seu ecossistema, este compartilhamento pode acontecer através de transferência de arquivo ou utilizando um banco de dados. Atualmente é muito comum as empresas possuírem vários *softwares* trabalhando para realizar seus processos de negócios. Porém, muitas vezes estes *softwares* são implementados com linguagens e para plataformas diferentes, em épocas diferentes e com tecnologias diferentes. Com isso fica muito difícil fazer uma comunicação de dados entre esses *softwares*.

Para resolver esse problema, é preciso uma forma de comunicação comum entre os diferentes tipos de *softwares*, uma linguagem comum a todas as aplicações. Uma forma universal de armazenamento de dados, suportado por qualquer plataforma e qualquer linguagem de programação é o Arquivo. Por isso, a forma de integração de aplicações com arquivos é uma das mais usadas, pela sua facilidade e agilidade. Arquivos XML é um exemplo de formato de arquivo, o qual foi criado, entre outros objetivos, para facilitar a comunicação de variados tipos de dados entre várias

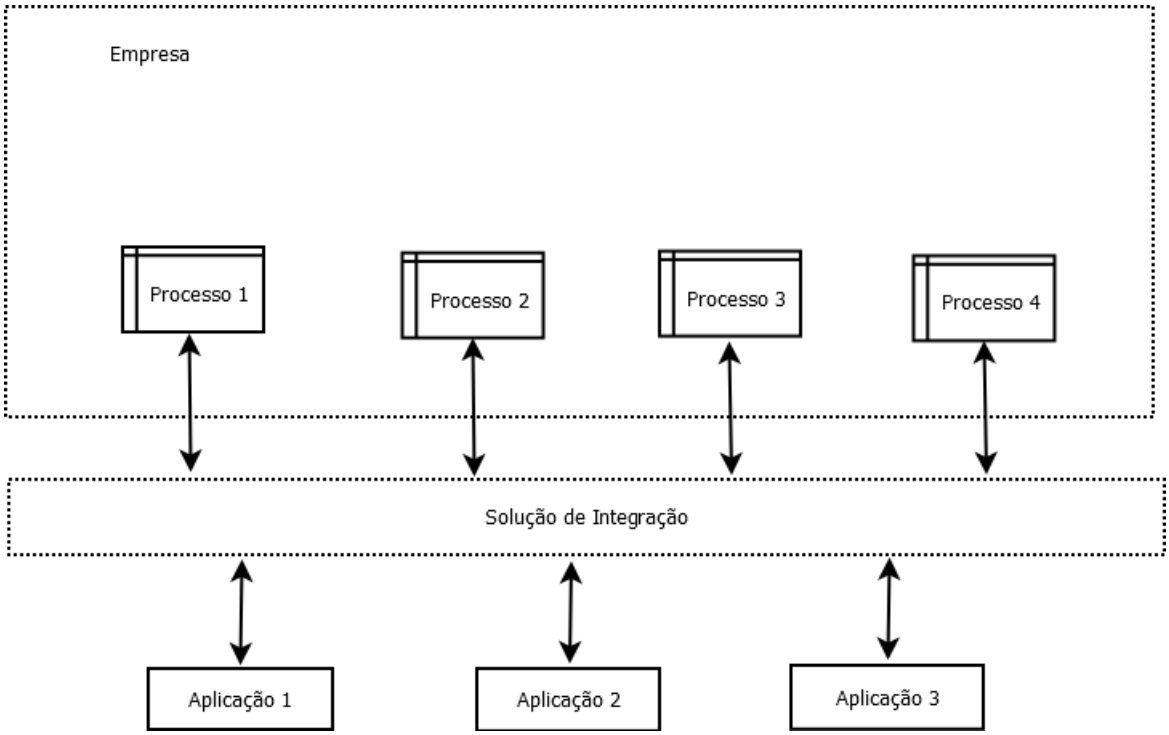


Figura 1.1: *Exemplo de solução de integração*

aplicações, e, portanto, pode ser usado para a integração de aplicações. Porém, um fator negativo é o custo computacional para persistir novos dados ou para ler dados desses arquivos. É necessário que periodicamente a solução de integração execute esse processo [28].

Geralmente as aplicações da organização usam um único local fixo para guardar e consumir dados, o banco de dados. Bancos de dados são suportados pela maioria das linguagens de programação e plataformas e utilizam uma linguagem universal para se comunicar com as aplicações, o SQL (Linguagem de Consulta Estruturada).

Entre as vantagens do banco de dados em relação ao arquivo, pode-se incluir a segurança, a consistência dos dados, a disponibilidade e a formatação dos dados. O próprio banco de dados possui sua formatação dos dados bem definida, nesse caso para as aplicações não há a necessidade de preocupação com a formatação de dados. Para tanto, destaca-se dois pontos negativos, a criação e manutenção de um único esquema de dados devido as

constantes alterações e as novas funcionalidades solicitadas pelas novas exigências das empresas e, a perda de desempenho provocado pelos processos simultâneos [28].

O compartilhamento de funcionalidades é utilizado quando existem funcionalidades presentes em uma aplicação que precisam ser acessadas pelas demais aplicações. Esse compartilhamento pode acontecer através de chamada de procedimento remota ou através de um sistema baseado em mensagens. Nem sempre a troca de dados entre as aplicações é suficiente. Muitas vezes é preciso que as aplicações compartilhem também funcionalidades, trocando informações diretamente umas com as outras.

Chamadas de Procedimento Remotas permitem que a aplicação possa reutilizar funcionalidades existentes em outras aplicações. Esse procedimento é útil quando há poucos dados a serem trocados, visto que uma comunicação remota utiliza a infraestrutura de rede e portanto podem ocorrer problemas de desempenho. Hohpe e Woolf [28] citam como problemas, a possibilidade de ter inúmeras solicitações para a mesma funcionalidade, podendo ocasionar uma sobrecarga, resultando em falhas na aplicação e no tempo de chamada e resposta para esta circunstância, debilitando o seu desempenho.

Por fim, o sistema de mensagens possui algumas características do estilo de arquivos, com a vantagem, porém, de permitir a transferência de pequenos pacotes de dados, de forma assíncrona, com facilidade e segurança. Essa transferência feita de forma assíncrona, significa que é possível transferir dados ou executar operações, mesmo que as aplicações de destino não estejam em funcionamento. Nesse caso, a mensagem permanece no canal até que possa ser entregue.

O sistema de mensagens possui a vantagem de permitir não só a troca de dados entre as aplicações, mas também o compartilhamento de suas funcionalidades e a transferência de alta velocidade e de forma frequente. Os estilos de integração baseados em arquivos e banco de dados não permitem compartilhar funcionalidades. Outra desvantagem é que as mensagens enviadas em sequência podem ficar desordenadas, caso alguma mensagem não seja entregue e necessite que o canal tente reenviá-la.

Realizar uma integração entre aplicações não é uma tarefa fácil de realizar, pois os sistemas são complexos e heterogêneos. Para realizar essa tarefa, padrões de integração devem ser usados como um modelo. Os padrões de integração são soluções prontas e testadas para um problema no contexto das integrações. Reúne conhecimento de especialistas em integração prontos

para serem usados. Ajudam a lidar com complexidade e desafios e formam uma maneira de modelar integrações com qualidade. Os padrões de integração fundamentais são:

- **Filtros e Canais (*Pipes and Filters*):** resolvem o problema do complexo processamento de mensagens, mantendo a independência e flexibilidade. Utiliza o conceito de dividir uma tarefa em tarefas menores e independentes (Filtros) que são conectadas por canais (*Pipes*);

- **Extremidade (*Endpoint*):** problemas de como uma aplicação deve se conectar a um canal para enviar e receber mensagens são resolvidas por esse padrão. É utilizado um cliente mensageiro dentro da aplicação que isola o sistema de mensagens do resto da aplicação;

- **Mensagem (*Message*):** como uma aplicação deve trocar informações com outras é a preocupação desse padrão. Utiliza-se uma mensagem para empacotar as informações e ser enviada para outra aplicação.

- **Canal (*Channel*):** as aplicações deve se comunicar de forma desacoplada e confiável. Este objetivo pode ser alcançado utilizando canais para conectar as aplicações;

- **Roteador (*Router*):** como uma mensagem deve ser direcionada a diferentes filtros é a tarefa desse padrão. Um roteador encapsula o critério de decisão para republicar uma mensagem para outro canal;

- **Tradutor (*Translator*):** aplicações usam diferentes formatos de dados diferentes para se comunicar. Um tradutor pode ser utilizado para converter formatos de dados em outro;

- **Barramento de controle (*Control Bus*):** para administrar efetivamente um sistema de mensagens distribuído deve-se utilizar barramentos de controle que gerenciem a solução de integração.

Algumas tecnologias utilizam padrões de integração como Camel [29], Spring Integration [20], Mule [19] e Guaraná [23]. São tecnologias com suporte para projetos e desenvolvimento de soluções de integração. Essas tecnologias possuem uma linguagem de domínio específico baseadas

em mensagens e permitem desenvolver soluções de integração em uma plataforma.

Com a evolução dos processos de negócios, atualmente há uma grande quantidade de organizações empresariais que necessitam realizar integração entre suas aplicações, visando reutilizá-las, dando suporte às novas demandas que surgiram. Portanto, uma solução de integração engloba um conjunto de aplicações, ou para orquestrá-las de forma a mantê-las sincronizadas ou para proporcionar novas funcionalidades já oferecidas por essas aplicações.

O sucesso da execução correta e eficiente da solução de integração torna-se essencial. Portanto, para aprimorar a qualidade das soluções construídas, é necessária uma análise do comportamento da solução de integração, para encontrar os possíveis gargalos de desempenho, ainda na fase de projeto. Para isso, é necessário ter um modelo conceitual de uma solução de integração, o que auxiliaria na redução de custos, riscos e tempo gasto.

A simulação é um campo de pesquisa que lida com a experimentação de modelos que permitam fazer previsões sobre o comportamento e o desempenho dos sistemas de reais. Baseia-se na utilização de técnicas matemáticas para estudar o comportamento e reações de um sistema por meio de modelos, imitando o funcionamento de operações de sistemas reais, numa sucessão de eventos que ocorrem ao longo do tempo. Esse estudo concentra-se em modelos estocásticos, dinâmicos e discretos, por caracterizar um sistema de uma solução de integração.

Caracteriza-se como sistemas de modelos estocásticos se uma ou mais de suas variáveis de entrada for aleatória. Utiliza-se probabilidade para modelar sistemas reais, não sendo possível determinar o comportamento do sistema. As soluções de integração apresentam características idênticas ao modelo estocástico. Os modelos dinâmicos caracterizam-se por representar sistemas de alteram seu estado ao longo do tempo. Nesse caso, tem-se o número de mensagens a ser processado em um determinado tempo de execução, e as operações executadas nas mensagens. Os modelos discretos caracterizam-se por ser orientados a eventos. São utilizados para modelar sistemas que mudam seu estado em pontos específicos de tempo, a partir da ocorrência de eventos. Por consumir tempo específico de processamento, as soluções de integração assemelham-se aos modelos discretos.

Caracterizando uma solução de integração conceitual como um modelo de eventos discretos, permite prever o comportamento futuro do sistema, através de técnicas de simulação. Existem diversas ferramentas já consolidadas para dar suporte à simulação de eventos discretos, tais como, SimEvents

[13], ProModel [53], PRISM [35], Pipe [18] e Arena [15].Para a realização da pesquisa dessa dissertação, utilizou-se a tecnologia Guaraná, a qual possibilita projetar, implementar e executar soluções de integração. Além disso, a tecnologia foi criada por pesquisadores que atuam no Grupo de Pesquisa em Computação Aplicada (GCA), grupo pelo qual a pesquisa foi desenvolvida. Em função desta proximidade que permitiu o acesso a documentação, destacou-se a tecnologia Guaraná com o objetivo de contribuir para o aperfeiçoamento e evolução, no contexto de identificação de gargalos de desempenho.

Para verificação e análise de desempenho do modelo solução de integração que pretende-se estudar, dentre as inúmeras técnicas matemáticas, optou-se pela decisão formal de Cadeias de Markov, por adotar modelos probabilísticos. Esse modelo pode ser especificado na linguagem de modelagem, na ferramenta de simulação PRISM, para a verificação e análise do modelo, sendo possível a implementação no mecanismo interno, para simulação de eventos discretos. A ferramenta PRISM é um software de verificação de modelos probabilísticos, livre e de código aberto, utilizado para análises de sistemas, modelagem formal e sistemas de integração.

Nesse sentido, propõe-se, neste trabalho, analisar o comportamento e identificar gargalos de desempenho em soluções de integração de aplicações empresariais modeladas na tecnologia Guaraná, antes de sua implementação e execução, por meio da criação/desenvolvimento de um modelo matemático em Cadeias de Markov, simulado e verificado em um modelo de simulação na ferramenta PRISM. Como estudo de caso, utilizou um sistema de gerenciamento de telefonemas pessoais implementado na Unijuí.

1.2 Motivação

A tecnologia de informação oferece suporte às organizações empresariais para suprir as suas necessidades de gerenciamento, oferecendo agilidade e qualidade em seus processos de negócios. Este suporte é fornecido por aplicações que compõe o chamado ecossistema de *software* [39] que, de uma maneira geral, não apresenta características necessárias à integração, dificultando a sua reutilização. Neste sentido, a Integração de Aplicações Empresariais (*Enterprise Application Integration* - EAI) concentra-se na concepção e implementação de soluções de integração. A tecnologia Guaraná é uma das ferramentas que fornecem este suporte, pois é uma proposta para construir soluções de integração acessível, apresenta como solução uma construção gráfica. O Guaraná propõe diferentes caixas de ferramentas para diferentes contextos de integração.

A análise do comportamento e a identificação de gargalos de desempenho em soluções de integração de aplicações geralmente envolve sua construção para posterior execução e teste frente à cenários críticos projetados pelos engenheiros de *software*. Por demandar que as soluções sejam construídas, tal abordagem traz consigo custos (tempo, recurso) e riscos (*bugs*) inerentes que costumam ser elevados. A presente pesquisa é motivada pela possibilidade de análise do comportamento e identificação de gargalos de desempenho ainda na fase de projeto, a partir dos modelos conceituais das soluções.

Como conclusão formula-se a seguinte hipótese:

Soluções de integração de aplicações podem ser classificadas como sistemas estocásticos, dinâmicos e discretos. Portanto, é possível utilizar modelos computacionais e matemáticos, e técnicas de simulação de eventos discretos para analisar o comportamento e identificar gargalos de performance que podem surgir em soluções de integração de aplicações quando submetidas à cenários críticos de funcionamento, tendo como base para a análise seus modelos conceituais.

1.3 Objetivos

A análise do comportamento e a identificação de gargalos de performance ainda na fase de projeto possibilitará projetar uma solução de integração otimizada. Neste sentido, esta pesquisa tem os seguintes objetivos:

1.3.1 Geral

Analisar o comportamento e identificar gargalos de desempenho em soluções de integração de aplicações a partir de seus modelos conceituais por meio da criação de um modelo matemático e de simulação para representar formalmente uma solução de integração utilizando Cadeias de Markov.

1.3.2 Específicos

- Realizar uma análise de tempos através de modelos de simulação com vistas a identificar gargalos de desempenho.
- Adiantar para a fase de projeto a identificação de gargalos de desempenho.
- Prever o comportamento de soluções de integração de aplicações em cenários críticos de funcionamento.

- Analisar o comportamento das mensagens nos *slots* durante a execução do modelo de solução de integração.
- Identificar como a solução de integração se comporta com a entrada de uma demanda elevada de mensagens.
- Verificar a equivalência do modelo conceitual com o modelo matemático proposto por meio de uma técnica de verificação.

1.4 Metodologia

A metodologia utilizada baseia-se no *framework* de referência fornecido pelo *Unified Process* (UP). Esta escolha é motivada pela experiência que o grupo de pesquisadores possui em utilizá-lo em pesquisas, e porque o seu ciclo de vida iterativo e incremental é apropriado ao desenvolvimento de projetos que visam alcançar um grande dinamismo, incorporando, a todo o momento, o *feedback* de outros grupos de trabalho e/ou projetos relacionados, mas mantendo sob controle os riscos que podem ocorrer no dia-a-dia. Considerando-se este *framework*, o projeto foi dividido nas seguintes fases:

INÍCIO: Trata-se de compreender o que será feito, identificar os principais pontos do projeto, compreender o problema de pesquisa desta dissertação e decidir sobre o processo a ser seguido. O trabalho foi realizado através de reuniões do tipo *brainstorming* com os professores e colegas mestrandos do Grupo de Pesquisa em Computação Aplicada (GCA), presencialmente para que cada pesquisador pudesse contribuir com suas ideias e conhecimento, não somente para que esta dissertação tivesse êxito.

ELABORAÇÃO: Parte dos objetivos desta fase consiste em identificar e descrever os principais blocos de trabalho e suas atividades. Este trabalho também foi desenvolvido através das reuniões do tipo *modelstorming* e os resultados foram à elaboração do plano de atividades e o cronograma. Durante o desenvolvimento desta dissertação, foi necessário refinar os blocos de trabalhos, priorizar algumas atividades e incrementar novos pontos não considerados no início do trabalho.

CONSTRUÇÃO: Esta fase dividiu as atividades dos blocos de trabalho em pequenas etapas que foram abordadas de forma iterativa, buscando ciclos rápidos de análise detalhada do caso de estudo, bibliografia, proposta de solução do problema e verificação, para que fosse possível obter, rapidamente, *feedbacks* úteis. A forma de abordar, com êxito, esta fase foi

mediante uma reunião inicial do tipo *task planning game*, reuniões periódicas para apresentação de resultados e seminários de pesquisa no qual eram apresentados os progressos alcançados, sejam eles resultados ou conclusões sobre a bibliografia estudada. As reuniões e seminários foram realizados sempre de forma presencial. Durante esta fase, participou-se de eventos científicos e tecnológicos relacionados com os temas desta dissertação, com o objetivo de apresentar os resultados e obter *feedback*.

TRANSIÇÃO: O principal objetivo desta fase é a transferência de resultados obtidos para o grupo de pesquisa e a apresentação desta dissertação. Os resultados obtidos serão utilizados para melhorar soluções de integração projetadas na Tecnologia Guaraná, como também publicados em conferências para consolidar e ampliar a rede de colaborações acadêmicas internacionais.

1.5 Resumo das Contribuições

Esta dissertação é parte integrante do projeto de pesquisa sobre a análise do comportamento e a identificação de gargalos de desempenho ainda na fase de projeto no contexto de integração de aplicações empresariais, e tem como objetivo uma solução de integração otimizada.

Neste sentido, a principal contribuição deste trabalho está relacionada na identificação de uma técnica e ferramenta de simulação para analisar o comportamento e identificar gargalos de desempenho em soluções de integração de aplicações empresariais. No desenvolvimento de um modelo de simulação equivalente ao modelo conceitual projetado na tecnologia Guaraná. E, verificação do modelo de simulação utilizando técnicas de verificação formal existentes na literatura. No entanto, os resultados parciais deste trabalho foram publicados e apresentados nos seguintes eventos:

- Um trabalho sobre Modelagem Matemática e Computacional para Análise do Comportamento de Soluções de Integração de Aplicações Através da Criação de Modelos de Simulação com Cadeias de Markov foi apresentado no III Seminário de Formação Científica e Tecnológica, promovido pelo Grupo de Pesquisa em Computação Aplicada (SFCT) no dia 19 de Maio de 2014, Ijuí, RS, Brasil. Este evento tem como objetivo criar um espaço local de debate que possa contribuir positivamente sobre o trabalho que vem sendo desenvolvido pelos membros do grupo.
- Outro trabalho sobre Modelagem Matemática e Computacional de uma Solução de Integração Utilizando Cadeias de Markov foi apresentado na XX Jornada de Pesquisa da UNIJUÍ, realizado no período de 28 de setembro a 2 de outubro de 2015, Ijuí, RS, Brasil.

1.6 Estrutura dessa Dissertação

Essa dissertação está organizada da seguinte maneira:

Parte I: Prefácio. Compreende essa introdução.

Parte II: Revisão da Literatura. Proporciona para o leitor uma revisão da literatura técnica e científica relacionadas à pesquisa desenvolvida nessa dissertação. No Capítulo §2, apresenta-se o referencial teórico. No Capítulo §3, são introduzidos os trabalhos relacionados identificados ao longo dessa pesquisa.

Parte III: Trabalho Desenvolvido. É apresentado o trabalho desenvolvido com base nos objetivos propostos para esta dissertação. No Capítulo §4, é apresentada a contribuição central da pesquisa desenvolvida. E no Capítulo §5, apresenta-se o trabalho de validação da proposta.

Parte IV: Considerações Finais. As conclusões e trabalhos futuros, a partir da pesquisa desenvolvida nessa dissertação são apresentadas no Capítulo §6.

Parte II

Revisão da Literatura

Capítulo 2

Referencial Teórico

A mente que se abre a uma nova ideia
jamais voltará ao seu tamanho original.

Albert Einstein, Físico, teórico alemão (1879 - 1955)

Para analisar o comportamento e identificar gargalos de desempenho em soluções de integração de aplicações empresariais, busca-se compreender os conceitos envolvidos, tais como: a tecnologia Guaraná, os sistemas em eventos discretos, a simulação e o formalismo matemático Cadeias de Markov. Nesse sentido, neste capítulo é apresentado na Seção Seção §2.1 a os conceitos sobre a tecnologia Guaraná. Na Seção §2.2 apresenta os principais conceitos sobre a classificação de sistemas e modelos de simulação. A Seção §2.3 é apresentada as propriedades sobre Cadeias de Markov de Tempo Discreto. E por fim, na Seção §2.4 o resumo do capítulo.

2.1 Tecnologia Guaraná

A tecnologia Guaraná é uma ferramenta que oferece suporte para o desenvolvimento de soluções de integração, possibilitando projetar soluções em um alto nível de abstração, utilizando uma sintaxe concreta gráfica, permitindo que os engenheiros de *software* tenham a visão de todo o conjunto de processos que compõe uma solução de integração.

Um fator importante desta tecnologia é que os modelos obtidos, implementados na tecnologia Guaraná, são independentes de plataforma, de modo que os engenheiros não dependem de conhecimentos e habilidades em uma tecnologia de integração de baixo nível, ao projetar suas soluções. As implementações podem ser reutilizadas para gerar soluções de EAI, automaticamente, executáveis para diferentes tecnologias de destino. Com a tecnologia Guaraná, pode-se tanto projetar suas soluções EAI quanto usá-la como uma ferramenta de implementação, execução e monitoramento de suas soluções EAI. A tecnologia Guaraná faz uso de tarefas baseadas nos Padrões de Integração Empresariais (EIP Patterns).

Na seção §2.1.1 é apresentado uma definição de *Domain-Specific Language*. Na seção §2.1.2 é descrito o modelo de execução do Guaraná DSL.

2.1.1 Linguagem de domínio especifica Guaraná DSL

A linguagem de domínio específico (em inglês ***Domain-Specific Language - DSL***) é uma linguagem utilizada para solucionar problemas específicos num domínio particular, através de uma linguagem própria, aplicadas ao desenvolvimento de *software*. O uso de uma DSL é recomendado quando um domínio focado (ou problema específico) tem um agrupamento simplificado de abstrações e notações correlacionadas [40].

As DSLs podem ser usadas por usuários com pouco conhecimento em linguagens de programação, já que elevam o nível de abstração e a facilidade de uso, pois os modelos usam uma forma visual através de blocos construtores e não através de código fonte. Com a simplicidade das DSLs, é possível uma comunicação eficaz com os clientes, facilidade de manutenção, sintaxe legível e de fácil compreensão, tornando o tempo de desenvolvimento menor.

A tecnologia Guaraná possui uma Linguagem de Domínio Específico com o objetivo de projetar soluções de Integração de Aplicações Empresariais

(EAI), com alto nível de abstração e custo razoável, proporcionando aos engenheiros de software, o uso de ferramentas para criar e implementar soluções de integração. Ela permite a manutenção do foco no domínio do problema por meio de um suporte gráfico e conceitos de modelagem muito intuitivos. Essa linguagem é chamada de Guaraná DSL. Guaraná DSL é uma linguagem de modelagem baseada nos padrões de integração conceitual [23].

Os modelos das soluções de integração são projetados usando a tecnologia Guaraná DSL. Os engenheiros de *software*, para configurar uma solução de integração, necessitam configurar uma série de adaptadores para se comunicar com as aplicações que estão sendo interligados, não necessitando, para isso, ter conhecimento sobre tecnologias de integração. A transformação de modelos DSL Guaraná em código executável pode ser obtida usando uma engenharia orientada por modelos.

A tecnologia Guaraná DSL é uma proposta para construir soluções de integração acessíveis. A mesma apresenta como solução uma construção gráfica. O Guaraná propõe diferentes caixas de ferramentas para diferentes contextos de integração.

2.1.2 Blocos de construção do Guaraná.

A implementação de uma linguagem de domínio específico, permite que os engenheiros de *software* possam manter seu foco no domínio do problema por meio de um suporte gráfico e conceitos de modelagem muito intuitivos. Segundo Frantz et al. [23] uma solução apresenta um conjunto de processos que integram um conjunto de aplicações. Guaraná DSL é uma linguagem de modelagem baseada nos padrões de integração conceitual [23], que são suportados por construtores como:

- **Mensagem**: uma abstração de uma informação que é modificada e compartilhada através de uma solução de integração. É composta por um cabeçalho, corpo da mensagem e um ou mais anexos. O cabeçalho inclui propriedades personalizadas e frequentemente as seguintes propriedades pré-definidas: identificador de mensagens, identificador de correlação, tamanho sequência, número de sequência, de endereço do remetente, data de validade e prioridade da mensagem. O corpo contém os dados de carga útil, cujo tipo é definido pelo parâmetro do modelo na classe mensagem. Anexos permitem que mensagens possam transportar peças extras de dados associados com a carga útil, por exemplo, uma imagem ou uma mensagem de e-mail.

- **Tarefa**: é uma operação atômica que pode ser executada em uma mensagem, como uma divisão, agregação, tradução, filtro, combinação, replicação, entre outros. Uma tarefa pode ter uma ou mais entradas assim como uma ou mais saídas aonde às mensagens chegam, são processadas, e enviadas.

- ***Slot***: é um canal conectando a entrada de uma tarefa com a saída de outra tarefa onde as mensagens são processadas de modo assíncrono pelas tarefas. Um *slot* pode seguir políticas para distribuir as mensagens entre as tarefas, como saídas baseadas em prioridade, ou um FIFO.

- **Porta**: as portas podem ser consideradas uma abstração dos detalhes necessários para a comunicação de componentes dentro de um ecossistema de *software*. De um modo geral, uma porta possibilita a escrita, leitura, solicitação e resposta de operações de comunicação com os componentes de integração.

- **Processo**: é a lógica da integração que executa a transformação, roteamento, modificação e operações relacionadas ao tempo das mensagens. O processo é composto de portas que possibilitam a comunicação com os componentes de integração, *slots* e conjunto de tarefas para especificar a lógica de integração.

Conceitualmente, uma solução de integração agrega um ou mais processos integração através dos quais flui mensagem que e são processados de forma assíncrona. O fluxo de integração é implementado como uma tubulação e arquitetura de filtro, em que os canais são implementados por entalhes e os filtros são implementadas pelas Tarefas. Cada tarefa percebe um padrão de integração [28] e a sua execução depende da disponibilidade de mensagens nos *slots* ligados às suas entradas. Os *slots* são construtores chaves para acionar de forma assíncrona uma solução de integração, assim as mensagens são armazenadas nos *slots*, até que possam ser processadas pela próxima tarefa no fluxo de integração.

2.1.3 Sintaxe Concreta.

A tecnologia Guaraná possibilita projetar soluções de integração utilizando sintaxe gráfica. Na Tabela §2.1 é apresenta a sintaxe concreta dos principais construtores que é utilizada para representar os conceitos da tecnologia Guaraná. As funcionalidade e estrutura de uma solução de EAI são

completamente definidas usando a linguagem de blocos de construção. Através destes símbolos é possível expressar todas as classes propostas: *Application*, *Process*, *EntryPort*, *ExitPort*, *IntegrationLink*, *ApplicationLink*, *Slot* e *Task*. Uma solução de integração modelada a partir desta linguagem é formada por um conjunto de aplicações e tarefas que cooperam para integrar ás diferentes aplicações. As tarefas do Guaraná são baseadas nas *Enterprise Integration Patterns* (Padrões EIP) por Hohpe e Woolf [28]. Esta linguagem disponibiliza um conjunto de ferramentas, denominadas *Task*, que são utilizadas para modelar diferentes integrações de aplicações empresariais.

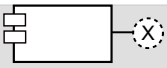
Notação	Conceito	Notação	Conceito
	*Application*		*IntegrationLink*
	Process		*ApplicationLink*
	EntryPort		*Slot*
	ExitPort		*Task*

Tabela 2.1: *Simbologia da sintaxe concreta (de Frantz et al. [23])*

Para a solução de integração proposta são utilizadas as tarefas (*Task*) descritas nas tabelas. O símbolo utilizado para representar tarefas (*Task*) possui entradas e saídas que podem ser observadas como saliências na lateral do ícone. Estas entradas e saídas são utilizadas para conectar tarefas entre si, através de *slots*. Dentro da solução de integração desenvolvida, são transmitidas mensagens, as quais contêm informações que são enviadas pelas aplicações. No Guaraná - DSL as tarefas são classificadas de acordo com a sua semântica em *Router*, *Modifier*, *Transformer*, *Stream Dealer*, *Mapper* e *Communicator*. Esses grupos de tarefas responsáveis pela comunicação entre os processos.

- ***Router*** (apresentado na Tabela §2.2) é uma tarefa que não modifica a mensagem em seu processo, mas direciona a mensagem durante o processo. Outras tarefas desse grupo são o filtro de mensagem que não satisfazem uma condição e a replicação de uma mensagem.

- ***Modifier*** (apresentado na Tabela §2.3) é um grupo de tarefas que adiciona ou remove dados de uma mensagem, porém o esquema da

mensagem não pode ser alterado. Melhorar uma mensagem adicionando dados ou movendo informações do corpo para o cabeçalho podem ser incluídos neste grupo.

- ***Transformer*** (apresentado na Tabela §2.4) compreende o grupo de tarefas que traduzem uma ou mais mensagem em uma nova mensagem com um novo esquema. Um exemplo de tarefas é a divisão de mensagens em várias ou a união de várias em uma.

- ***Timer*** tem a finalidade de realizar tarefas de ações relacionadas a tempo, por exemplo, atrasando o envio de mensagens ou a produção de mensagens em intervalos regulares.

- ***StreamDealers*** (apresentado na Tabela §2.5) são tarefas que trabalham com os fluxos de dados e responsáveis por ações como compactar/descompactar, codificar/decodificar e encriptar/decriptar dados.

- ***Mapper*** (apresentado na Tabela §2.6) é um grupo de tarefas que alteram as representações das mensagens em processos. Por exemplo o mapeamento de um fluxo de dados em um arquivo XML.

- ***Communicator*** (apresentado na Tabela §2.7) faz o encapsulamento de um adaptador. As tarefas deste grupo possuem dois objetivos: possibilitam adaptadores a serem exportados para um registro para que possam ser acessados remotamente; o comunicador pode ser configurado para apurar periodicamente um processo ou aplicação usando um adaptador.

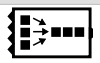
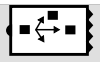
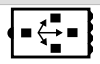
Notação	Tarefa	Descrição
	Correlator	Analisa mensagens de entrada e produz como saída um conjunto de dados correlacionados.
	*Merger*	Direciona as mensagens que chegam de diferentes *slots* de entrada em um único *slot* de saída
	Resequencer	Reordena as mensagens em sequências por meio de uma ordem pré-estabelecida.
	Filter	Filtra as mensagens indesejadas.
	Idempotent Transfer	Remove mensagens duplicadas.
	Dispatcher	Envia uma mensagem a exatamente um *slot* de saída.
	*Distributor*	Distribui mensagens para dois ou mais *slots*.
	*Replicator*	Realiza cópias da mensagem de acordo com o número de *slots* de saída.
	Semantic Validator	Valida a semântica de uma mensagem.

Tabela 2.2: *Tarefas da categoria Routers (de Frantz et al. [23])*

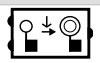
Notação	Tarefa	Descrição
	Slimmer	Remove o conteúdo de uma mensagem de acordo com uma política estática.
	Context Based Slimmer	Remove o conteúdo de uma mensagem base de acordo com uma política dinâmica que é fornecida por uma mensagem contexto.
	Content Enricher	Adiciona conteúdos estáticos na mensagem.
	Context Based Content Enricher	Adiciona conteúdos dinâmicos de uma mensagem contexto em uma mensagem base.
	Header Enricher	Adiciona conteúdos estáticos para o cabeçalho de uma mensagem.
	*Context Based Header Enricher*	Adiciona conteúdo dinâmico de uma mensagem contexto para o cabeçalho de uma mensagem base.
	Header Promoter	Promove uma parte do corpo de uma mensagem para o cabeçalho.
	Header Demoter	Retira uma parte do cabeçalho de uma mensagem e coloca no corpo da mensagem.

Tabela 2.3: *Tarefas da categoria Modifiers (de Frantz et al. [23])*

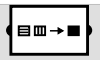
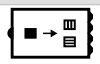
Notação	Tarefa	Descrição
	Translator	Traduz o conteúdo da mensagem para outro formato.
	Splitter	Divide uma mensagem que contém elementos repetidos em várias mensagens.
	*Aggregator*	Constrói uma nova mensagem a partir de várias mensagens produzidas anteriormente através do *splitter*.
	*Chopper*	Quebra uma mensagem em duas ou mais mensagens.
	Assembler	Constrói uma nova mensagem a partir de duas ou mais mensagens.
	Cross Builder	Constrói uma nova mensagem contendo o produto cartesiano de todas as mensagens de entrada.

Tabela 2.4: *Tarefas da categoria Transformers (de Frantz et al. [23])*

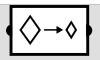
Notação	Tarefa	Descrição
	*Zipper*	Compacta uma mensagem.
	Unzipper	Descompacta uma mensagem.
	Encrypter	Criptografa uma mensagem.
	Decrypter	Descriptografa uma mensagem
	Encoder	Codifica uma mensagem.
	Decoder	Decodifica uma mensagem.

Tabela 2.5: *Tarefas da categoria Stream Dealers (de Frantz et al. [23])*

Notação	Tarefa	Descrição
	XML2Stream	Mapeia um fluxo de bytes para uma mensagem XML.
	Stream2XML	Mapeia uma mensagem XML para um fluxo de bytes.

Tabela 2.6: *Tarefas da categoria Mappers (de Frantz et al. [23])*

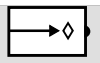
Notação	Tarefa	Descrição
	*In Communicator*	Usado em portas para ler mensagens.
	Out Communicator	Usado em portas para escrever mensagens.

Tabela 2.7: *Tarefas da categoria Communicators (de Frantz et al. [23])*

2.1.4 Modelo de Execução do Runtime System do Guaraná RT.

O motor de execução da tecnologia Guaraná é apresenta na Figura §2.1. São apresentados conceitos altamente técnicos. Alguns desses conceitos serão apresentados nesta dissertação. Prioriza-se o estudo dos conceitos fundamentais para a realização da pesquisa da dissertação, tais como: *work unit*, *work queue* e *thread*.

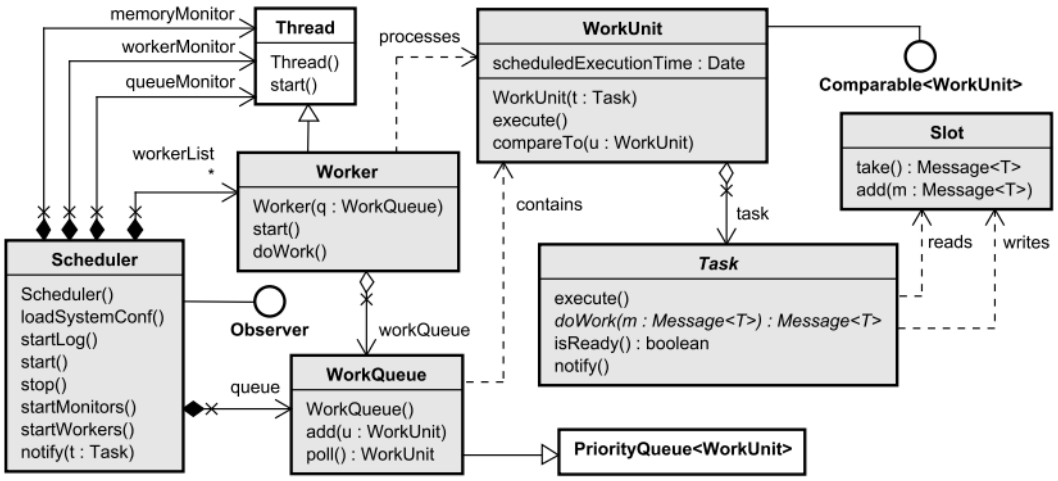


Figura 2.1: *Modelo de execução do Guaraná baseado em tarefas (de Frantz et al. [22])*

O modelo de execução da tecnologia Guaraná é baseado em tarefas, diferenciando-se das demais tecnologias de integração. As *threads* são atribuídas ao motor de execução de tarefas, para que não apresente problema de prioridade. Isso permite que as tarefas sejam executadas assim que tiver mensagens disponíveis em seus *slots* de entrada, sendo assim considerada pronta para executar. Para tanto, esse processo não requer a disponibilidade de todas as mensagens para iniciar o processo, pois as mensagens que necessitam ser processadas em conjunto são correlacionadas no interior da tarefa específica.

As unidades de trabalho (*work units*) a serem processadas são armazenadas em uma fila de prioridades, chamada de fila de unidades de trabalho

(*work queue*). Uma unidade de trabalho possui uma referência para uma tarefa e um tempo de execução programada. Antes deste tempo a unidade de trabalho não pode ser executada.

Uma tarefa pode executar assim que possível, pois o tempo de execução programada pela *work units* é definida com o tempo atual do sistema. No entanto, se o tempo de execução for definido para um tempo futuro, significa que a execução desta tarefa será atrasada até passar esse tempo. A *thread* no motor de execução é por responsável por retirar uma unidade de tempo (tarefa) da fila e executar. A tecnologia Guaraná permite que em sua solução de integração sejam implementadas mais de uma *thread*. Isso é útil para implementar tarefas que precisam serem executadas periodicamente e utilizam esse tipo de característica.

Para entender melhor o funcionamento do motor de execução é apresentado na figura §2.2 um processo de uma solução de integração. Na figura é possível observar uma orquestração de processos que coordenam as aplicações envolvidas na solução de integração, contendo portas de entradas (*Entry Port*) e portas de saídas (*Exit Port*). Essas portas são criadas para ler as mensagens de um processo ou de uma aplicação. As mensagens aguardam nos slots para serem executadas pelas tarefas (*Task*). No entanto para que a tarefa possa ser executada é necessário que tenha uma mensagem disponível no seu *slot* de entrada. As tarefas aptas são armazenadas em uma fila de prioridades*work queue* para serem executadas pelo motor de execução.

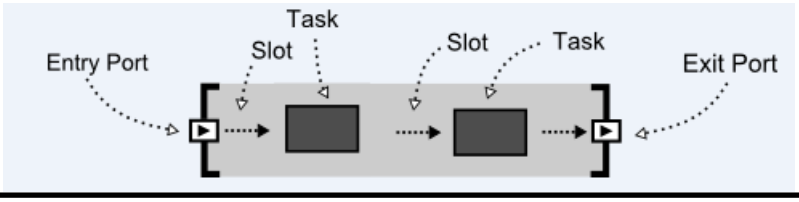


Figura 2.2: *Processos de uma solução de integração (de Frantz et al. [23])*

2.2 Classificação de Sistemas e Simulação de Modelos.

Essa seção fornece conceitos fundamentais envolvidos na proposta de pesquisa, tais como: sistema, modelos e simulação. Para este estudo, a simulação

refere-se à simulação de um modelo do sistema. O termo sistema pode ser utilizado para uma grande variedade de contextos. Para Sommerville [56], um sistema é uma coleção significativa de componentes inter-relacionados, que trabalham em conjunto para atingir algum objetivo, consistindo em uma combinação de informações atuando juntas para abordar um objetivo específico. Os próprios elementos e suas inter-relações determinam como o sistema trabalha. Todas as áreas do conhecimento possuem sistemas, características e leis, independentemente da área onde se encontram.

O sistema pode incluir *software* e *hardware* e ser administrado por pessoas, consiste em um conjunto de elementos, materiais e imateriais, entre os quais se pode definir uma relação que operam com uma estrutura organizada [14].Os componentes de um sistema são dependentes de outros componentes, sendo que suas propriedades e o comportamento dos componentes de sistema são fortemente interligados. A especificação de um sistema compreende várias informações, devendo cada uma delas ser expressa numa linguagem ou técnica apropriada.

Na Figura §2.3 estão representadas as diversas maneiras de se estudar um sistema. Inicialmente há duas possibilidades, uma por experimento com o sistema real e outra, por experimento com um modelo do sistema. Para estudar um sistema real, tanto por meio de experimentos quanto por medições diretamente com o sistema, há uma demanda de um elevado custo, risco e tempo gasto para esse estudo. Por outro lado, há a possibilidade de realizar experiências por meio de um modelo do sistema.

Os experimentos realizados com um modelo do sistema subdividem em duas partes: uma delas em um modelo físico, e outra em um modelo matemático. Busca-se, portanto, um estudo através da modelagem matemática, que procura descrever um sistema real a partir de um conjunto de restrições definidas pelo desenvolvedor do sistema, podendo ser manipulados e analisados por meio de simulação.

As soluções analíticas de acordo com Law e Kelton [38], utilizam técnicas tradicionais de matemática e de estatística para realizar a análise e obter informações exatas sobre o sistema modelado. No caso de um modelo ser muito complexo, com as soluções analíticas torna-se inviável a solução, pois os cálculos são muito complexos e extensos. Assim, as soluções analíticas não permitem prever o comportamento do sistema real. Neste caso, procura-se utilizar a simulação, que constitui em uma importante técnica para modelar sistemas complexos, inclusive seu aspecto estocástico (ou probabilístico) e dinâmico, sem tornar o modelo difícil de ser estudado.

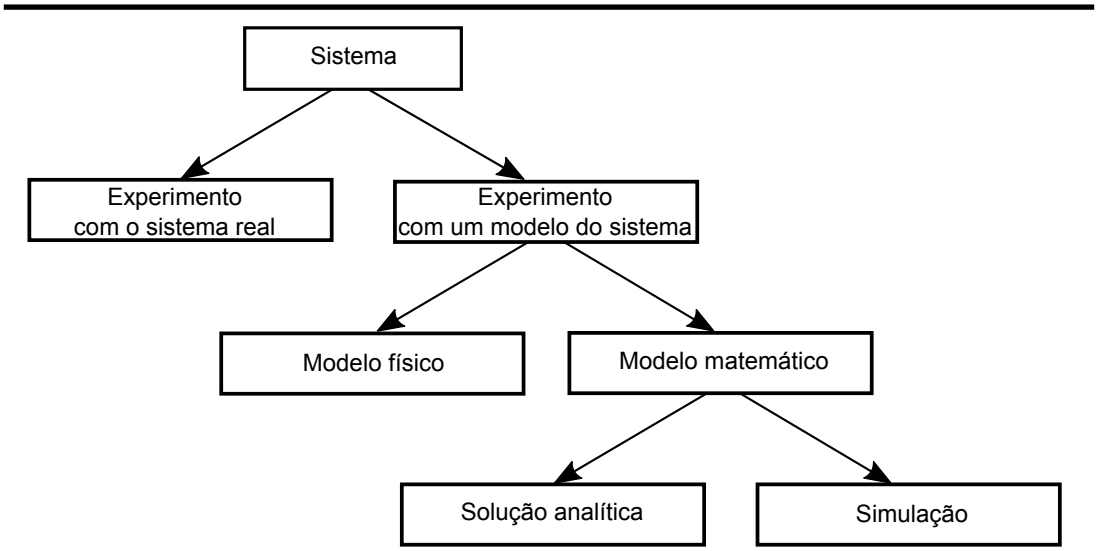


Figura 2.3: *Abordagens para estudar um sistema (de Law e Kelton [38])*

2.2.1 Modelo

Um modelo representa as relações dos componentes de um sistema. É uma interpretação que ajuda a explicar e entender uma abstração simplificada de um sistema em estudo, devendo apresentar detalhamento suficiente para ser utilizado como uma representação válida. O modelo é uma representação externa e explícita de uma parte da realidade vista pela pessoa que deseja usar aquele modelo para entender, mudar, gerenciar e controlar parte daquela realidade que se aproxima da realidade [48].

O modelo descreve o comportamento do sistema. Os sistemas podem ser classificados de diversas maneiras, como por exemplo, subdividir os sistemas em modelos Determinísticos e Estocásticos; Estático e Dinâmico; Discreto e Contínuo. Os modelos são simulados para estudar o comportamento e reações de um determinado sistema, como podemos observar na Figura §2.4, que ilustra o modelo proposto por Law e Kelton [38].

A classificação de modelos na taxonomia de Law e Kelton [38], faz uso tanto de modelos determinísticos como estocásticos. Caso o modelo de simulação não contenha variáveis probabilísticas, são considerados modelos determinísticos. Os modelos determinísticos são utilizados quando não há

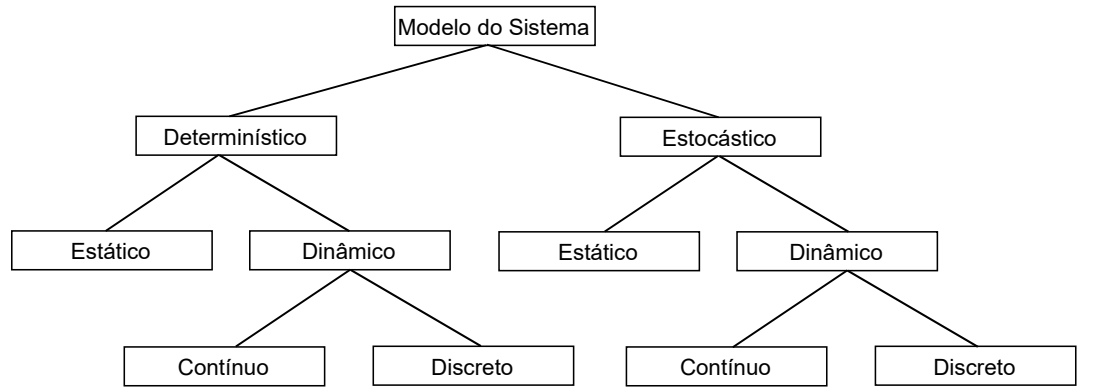


Figura 2.4: *Taxonomia de modelos (de Law e Kelton [38])*

dúvidas quanto à forma do resultado, e os dados utilizados são perfeitamente conhecidos e determinados. Um sistema é determinístico se, para um conjunto conhecido de entrada de dados, o resultado seja um conjunto único de saída, sendo possível definir o estado a ser alcançado.

No caso de um modelo apresentar uma ou mais variáveis de entrada que levam a serem aleatórias as saídas, são chamados de modelos estocásticos. Os modelos estocásticos são determinados por modelos probabilísticos, pois não é conhecido o resultado esperado, mas é conhecida a respectiva probabilidade. Ou seja, um sistema é dito estocástico se uma ou mais de suas variáveis forem aleatórias. Nesse caso, o vetor de estado do problema é descrito por um processo estocástico e uma descrição probabilística é necessária para caracterizar o comportamento do sistema [14]. A saída de simulação estocástica deve ser tratada como estimativa estatística das características reais de um sistema. Normalmente, modelos estocásticos são mais complexos e melhor representam sistemas reais do que os modelos determinísticos.

Os sistemas de modelos determinísticos e estocásticos são subdivididos e classificados em relação ao tempo, em modelos estáticos e modelos dinâmicos. Os modelos estáticos representam um sistema em um determinado momento. Um sistema é caracterizado estático por não apresentar alteração ao longo do tempo. Nos sistemas estáticos, a saída é independente dos valores passados da entrada e o comportamento dos sistemas estáticos é descrito através de métodos numéricos, problemas de otimização, teste de algoritmos, equações algébricas e Modelos de Monte Carlo. Por outro lado, os modelos dinâmicos representam sistemas em que há mudança ao

longo do tempo. Um sistema é caracterizado dinâmico por apresentar variações. Assim, nos sistemas dinâmicos, a saída depende dos valores passados da entrada. O comportamento dos sistemas dinâmicos é, de modo geral, descrito através de equações diferenciais [14].A maioria das simulações emprega modelos dinâmicos, porque a maioria dos sistemas muda com o tempo.

Os modelos dinâmicos ainda podem ser classificados em discreto e contínuo, de acordo com o comportamento em relação ao tempo [38]. Esse comportamento pode ser observado na Figura §2.5 e a Figura §2.6. OOs sistemas de eventos contínuos são aqueles sistemas nos quais as variáveis de estado mudam continuamente no tempo. A cada unidade de tempo, o sistema poderá assumir um novo valor, representando o comportamento de um sistema em qualquer instante. A seguir a Figura §2.5 representa um sistema em um instante qualquer, a partir de um instante inicial.

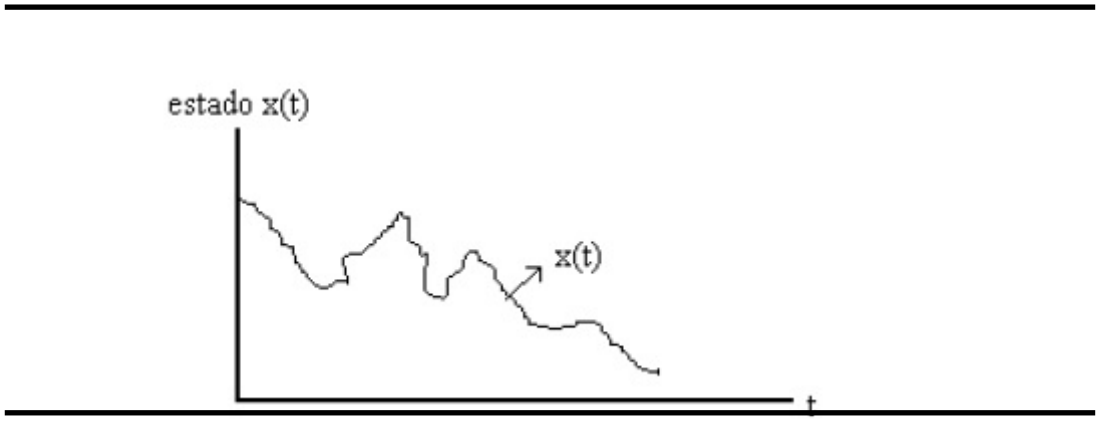


Figura 2.5: *Trajetória de um sistema dinâmico de variável contínua (SDVC). Fonte: Carvalho [10]*

Já nos sistemas de eventos discretos (SED) o tempo é definido em certos instantes. Cada estímulo do evento causa uma transição ou mudança de estado no sistema. Cardoso e Valette [7] definem sistemas a eventos discretos como sistemas que são modelados de tal forma que as variáveis de estado alteram bruscamente em instantes determinados e cujos valores das variáveis nos próximos estados podem ser calculados diretamente a partir dos valores precedentes, sem ter que considerar o tempo entre estes dois instantes.

De acordo com a Figura §2.6, na qual variáveis mudam de estado instantaneamente em pontos específicos de tempo, pode-se observar que a cada evento que ocorre no sistema, há uma mudança de estado. Os eventos ocorrem num determinado instante, como por exemplo, a chegada de pessoas numa fila de supermercado.

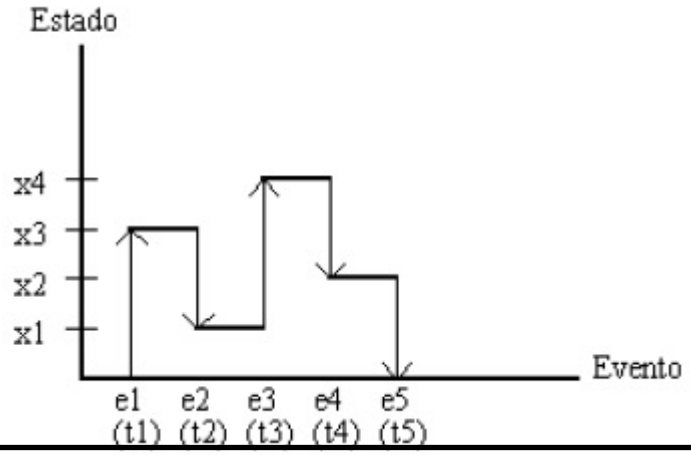


Figura 2.6: *Trajetória de um sistema dinâmico de eventos discretos (SDED)*

A decisão de usar um sistema de modelo discreto ou um sistema de modelo contínuo para um determinado problema, depende dos objetivos específicos do estudo. Dessa forma, o estudo dessa dissertação tem por objetivo a análise do comportamento e a identificação de gargalos de desempenho ainda na fase de projeto de uma solução de integração, a partir dos seus modelos conceituais, sem que as mesmas tenham de ser implementadas, desenvolvendo um modelo de simulação.

2.2.2 Simulação

A simulação computacional de sistemas consiste na utilização de técnicas matemáticas para estudar o comportamento e reações de um sistema através de modelos, imitando o funcionamento de operações de sistemas reais, numa sucessão de eventos que ocorrem ao longo do tempo. A simulação consiste em validar/verificar um modelo, estudando o seu comportamento, e considera as informações sob um ponto de vista sistêmico e dinâmico.

A simulação é uma das mais poderosas técnicas disponíveis para a solução e resolução de problemas muito complexos nas organizações. É uma técnica de solução de um problema pela análise de um modelo que descreve o comportamento do sistema, usando um computador digital [50]. A simulação leva em conta não somente médias ou padrões determinísticos, mas também a distribuição estatística associada ao fenômeno estudado, aproximando o modelo ao mundo real de forma estocástica.

A simulação proporciona vantagens e desvantagens. As vantagens evidenciam-se quando se compara com a implementação direta do sistema, levando-se em conta custos e tempo de execução, visto que a simulação promove uma maior eficiência no funcionamento do sistema. Ou seja, o mesmo é otimizado por apresentar um custo adequado, sendo que os usuários estarão satisfeitos com o ambiente ou com o serviço oferecido. Law e Kelton [38] apresentam as seguintes vantagens para a utilização da modelagem e simulação computacional em estudo de sistemas:

a) Sistemas complexos que contenham elementos estocásticos que não conseguem ser tratados adequadamente por técnicas analíticas podem ser, na maioria das vezes, estudados via simulação. Pois o modelo de simulação é, em geral, mais fácil de compreender do que um conjunto de complicadas equações matemáticas;

b) Fornece um controle melhor sobre as condições experimentais do que seria possível na experimentação no sistema real;

c) Interações e experimentação. Tornando possível a experimentação de forma interativa;

d) Modelos de Simulação com maior liberdade na sua construção, tornando-os mais realistas. O modelo pode ser relativamente simples na sua implementação, mas aumenta a complexidade aos poucos, identificando de maneira mais clara as peculiaridades do problema;

e) Permite simular longos períodos em um tempo reduzido;

f) Em geral, é mais econômico do que implementar e testar um sistema real.

Embora as inúmeras vantagens, o processo de simular apresenta algumas dificuldades. Como desvantagens para a utilização de simulação computacional em estudo de sistemas, Law e Kelton [38] apresentam:

a) A simulação é dependente da validade do modelo desenvolvido. Se o modelo criado não representa confiabilidade o sistema ou se os dados de entrada não são confiáveis, de nada adianta fazer um estudo detalhado dos dados de saída e encontrar uma solução para o problema;

b) A técnica da simulação não é por si só otimizante, a simulação testa apenas as alternativas dadas pelo usuário na hora da implementação na ferramenta;

c) Um estudo e análise da simulação podem se tornar demorado e consumir recursos financeiros elevados e de tempo;

Segundo Arcaute et al. (2001) *apud* Carvalho [10],a simulação de um sistema contínuo, usando um simulador a eventos discretos, é possível. Porém, simplificações se aplicam ao modelo, especialmente no manuseio de recursos e entidades. Os resultados são limitados se comparados aos de um sistema discreto. A simulação é uma poderosa ferramenta para a tomada de decisões no desenvolvimento de sistemas mais eficientes. O seu uso durante o desenvolvimento é uma técnica preditiva e preventiva, no qual respostas para questões situacionais irão auxiliar em tomadas de decisões e podem evitar cobranças futuras quanto a questões não levantadas ou analisadas no momento da implantação de determinado processo Oliveira [44].

Para ter sucesso no processo de simulação van der Aalst [58] apresenta algumas fases de um estudo de simulação tradicional. Na Figura §2.7 é apresentado as fases de um típico projeto de simulação. Essas fases devem ser vistas como um refinamento. O processo de simulação inicia-se com uma definição do problema, descrevendo os objetivos e fixa o âmbito do estudo de simulação. Após a definição do problema, a próxima fase é a modelagem. Nesta fase, o modelo conceitual é criado. O modelo conceitual define classes de objetos e as relações entre esses objetos. Além disso, o processo de modelagem pode trazer à tona novas questões para o estudo de simulação para responder. Para tanto, a definição do problema deve ser ajustado.

Após a fase de modelagem conceitual, inicia-se a fase de realização. Neste caso, o modelo conceitual é implementado em um modelo executável. O modelo pode ser executável, simulado em um computador. Como criar este modelo depende fortemente da ferramenta de simulação utilizada. Linguagens de simulação exigem um design simples e implementação fácil.

Um modelo executável não é necessariamente correto, por isso tem que ser verificado. A verificação do modelo é necessária para examinar se o modelo contém erros qualitativos ou quantitativos, como erros de programação ou ajustes errados de parâmetros.

A partir do modelo validado, as simulações podem ser efetuadas. As simulações devem ser conduzidas de tal maneira que os resultados obtidos sejam confiáveis. Nesta etapa, serão tomadas as decisões sobre o número de simulações. Os resultados das simulações precisam ser interpretados para permitir um *feedback* para a definição do problema. Além disso, os resultados terão de ser interpretados afim de responder às perguntas da definição do problema.

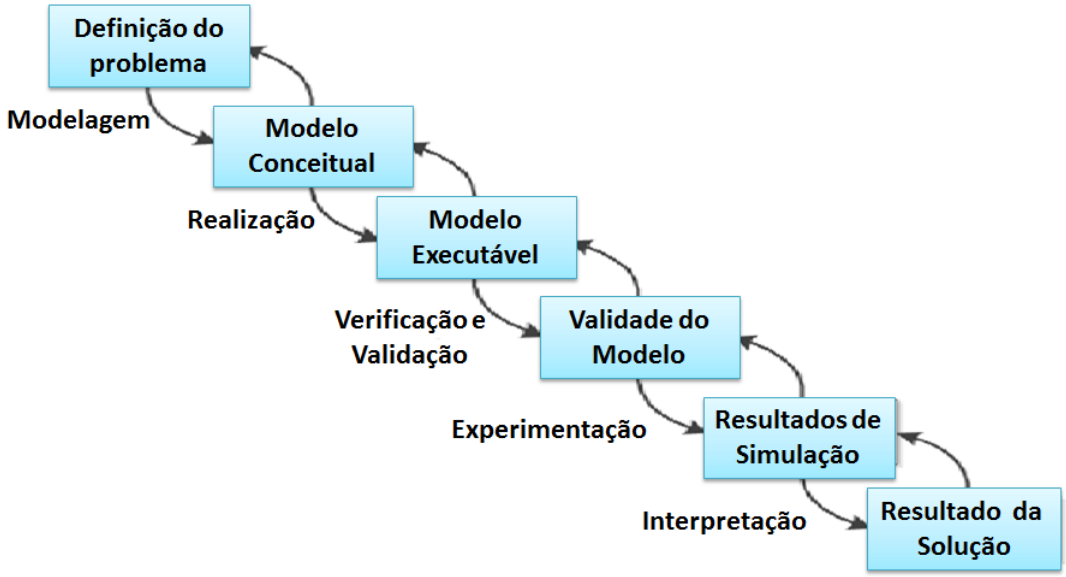


Figura 2.7: *Fases de um estudo de simulação tradicional.(Adaptado de van der Aalst [58])*

Para a realização do estudo desta dissertação, o modelo de simulação será implementado utilizando estratégias de simulação. Utiliza-se a simulação de um modelo estocástico de eventos discretos, que corresponde ao estudo de sistemas, empregando modelos nos quais as variáveis de estado mudam apenas em instantes discretos de tempo. A simulação é um campo de pesquisa que lida com a experimentação de modelos que permitam fazer previsões sobre o comportamento e o desempenho dos sistemas reais. O modelo matemático de simulação utiliza formalismo matemático das Cadeias de Markov, que é apresentado em seguida.

2.3 Cadeias de Markov

Esta seção apresenta alguns conceitos básicos sobre cadeias de Markov que serão utilizadas para o estudo nessa dissertação. Para que a teoria proposta por este trabalho seja apresentado de forma clara, precisamos conhecer uma série de conceitos relacionados ao estudo de processos estocásticos, em especial as Cadeias de Markov de tempo discreto. Um modelo

de um processo estocástico descreve atividades que terminam em eventos. Os eventos geram a transição de um estado a outro. Um tempo discreto no sistema induz o modelo a Cadeias de Markov [8].

2.3.1 Definição

Cadeia de Markov são um formalismo de modelagem de sistemas que descrevem o sistema como um processo estocástico. Desse ponto de vista, o sistema modelado é caracterizado pelos seus estados e a forma pela qual eles se alternam. Cadeias de Markov são processos estocásticos com estados discretos. O parâmetro, que em geral é o tempo, pode ser discreto ou contínuo. As cadeias de Markov são caracterizadas por seu estado futuro depender apenas do seu estado atual, sendo que os estados passados não influenciam no estado futuro.

Os modelos empregados nos testes probabilísticos descrevem os possíveis usos de um sistema e suas interações com o usuário através do emprego de um modelo de estados discretos. As cadeias de Markov tornam-se a maneira mais simples e direta de representar esses sistemas por ser um dos mais antigos formalismos de descrição de modelos de estados discretos. Representa o sistema como um conjunto de estados juntamente com as transições entre estes estados. As Cadeias de Markov são processos estocásticos com a chamada propriedade Markoviana.

Um processo Markoviano é dito ser uma Cadeia de Markov quando as variáveis randômicas X_t possuem uma relação de dependência particular com o tempo, estas variáveis estão definidas em um espaço de estados discretos. As transições entre os estados são modeladas por um processo estocástico de tempo contínuo ou discreto definidos. Para Pinheiro [49] um Processo Estocástico é definido como uma coleção de variáveis randômicas $(X(t))$ indexadas por um parâmetro t pertencente a um conjunto T. Frequentemente T é tomado para ser o conjunto dos inteiros não negativos (porém, outros conjuntos são perfeitamente possíveis) e $X(t)$ representa uma característica mensurável de interesse no tempo t. Exemplificando, $X(t)$ pode representar o nível de estoque de um produto no fim da semana t. Formalmente, a notação $\{X_t; t \epsilon T\}$ define processo estocástico sobre um dado espaço de probabilidade.

As cadeias de Markov são utilizadas devido à sua simplicidade e ao fato de possibilitar inferências de raciocínios e inspeções de desempenho para realidades variadas através de técnicas de modelagem. Uma cadeia de Markov pode ser vista como um sistema de transição de estados onde o modelador define quais estados são acessíveis a partir de outros estados e configura uma taxa ou probabilidade das transições do sistema.

No contexto de Cadeias de Markov, é possível representar um sistema usando tempo contínuo (Cadeias de Markov de tempo contínuo *Continuous Time Markov Chains*, - CTMC) ou tempo discreto (Cadeias de Markov de tempo discreto *Discrete Time Markov Chains*, - DTMC). Uma cadeia de Markov em tempo discreto é uma coleção de variáveis aleatórias de um processo de Markov que assumem valores dentro de um espaço de estados enumerável (finita ou infinitamente) [45], onde temos a probabilidades de transições ocorrem de um estado para o outro. Cadeias de Markov de tempo contínuo fornecem um paradigma diferente para modelagem de sistemas. Nas CTMCs as transições entre estados podem ocorrer a qualquer momento, ou seja, a qualquer instante de tempo, ao contrário das cadeias de Markov de tempo discreto, em que as transições ocorrem sempre em instantes de tempo discretos e conhecidos [6].

Neste sentido, para o estudo dessa dissertação optou-se na utilização de cadeias de Markov de tempo discreto, pois as Cadeias de Markov em tempos discretos representam um modelo de um processo estocástico que descreve atividades que terminam em eventos, esses eventos geram as transições de estados. Uma Cadeia de Markov é representada por uma maquina de estados, é uma sequência de variáveis aleatórias X_t que representa o estado em um determinado tempo. Ou seja a variável X_2 representa o estado do sistema no tempo 2. O domínio de X_t é o próprio espaço de estados.

2.3.2 Propriedades das Cadeias de Markov de Tempo Discreto

Uma propriedade importante das cadeias de Markov é a falta de memória, onde os estados anteriores são irrelevantes, também é conhecido como um processo Markoviano. Em termos de probabilidades, uma cadeia de Markov a tempo discreto com espaço de estados S é um processo estocástico $\{X_n\}\, n \in T$, onde $T = \{0, 1, 2, ...\}$, tal que se verificam as seguintes propriedades:

- Para qualquer $i \in S$ tem-se:

$$P(X_0 = i) = P_i \tag{2.1}$$

- Para quaisquer $i, j \in S$, e $n \in T$:

$$P(X_{n+1} = j | X_n = i) = P_{ij} \tag{2.2}$$

- Para quaisquer $n \in T$ e $i_0, i_1, ..., i_{n-1}, i, j \in S$, vale a condição:

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1} ..., X_0 = i_0) = P(X_{n+1} = j | X_n = i) \tag{2.3}$$

No caso $X_n = i$ diz-se que o processo no instante n está no estado i. Em especial, a terceira propriedade nos diz que dado o presente (X_n), o futuro (X_{n+1}) e o passado ($X_0, X_1, ..., X_{n-1}$) são independentes.

A equação §2.3 representa a probabilidade dos estados, estabelecendo um conjunto de estados discretos e que o próximo estado só depende do estado atual e não do passado. Ou seja, os estados anteriores são irrelevantes para a predição dos estados seguintes desde que o estado atual seja conhecido.

As Cadeias de Markov envolvem uma matriz, denominada matriz de transição, cujos elementos são as probabilidades de transição de um estado para outro. A transição de um estado a outro é chamado de passo. A cada passo a probabilidade de se alcançar um próximo estado independe das probabilidades dos estados anteriores. A matriz de transição pode ser representada por uma matriz **n** por **n** onde **n** é o número de estados da cadeia e cada posição ij da matriz representa a probabilidade de se transitar do estado i para o estado j.

Para cadeias de Markov homogêneas, $P(X_{n+1} = j|X_n = i) = P_{ij}$. Seja $\{X_n\}n \in T$ uma cadeia de Markov homogênea com espaço de estados discretos $S = \{1, 2, 3, ...\}$. Nesse caso, tem-se $P_{ij} = P(X_{n+1} = j|X_n = i), i, j \geq 0$ independente de n. Uma maneira intuitiva de apresentar e operacionalizar as probabilidades de transição entre os estados é dada por meio de uma matriz estocástica, conhecida por matriz de transição. Uma matriz estocástica da cadeia de Markov de transição pode ser representada de acordo com a matriz P, onde cada entrada na matriz P deve ser positiva e as entradas de cada fila na matriz devem somar 1. P pode ser escrita como:

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1S} \\ P_{21} & P_{22} & \dots & P_{2S} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ P_{S1} & P_{S2} & \dots & P_{SS} \end{bmatrix} \tag{2.4}$$

Ou seja, a matriz de transição de uma cadeia de Markov com k estados, deve se ter:

$$P_{1j} + P_{2j} + ... + P_{sj} = 1, j = 1, 2, ..., s \tag{2.5}$$

Graficamente uma cadeia de Markov de tempo discreto é representada por um diagrama de estados e transições, que é um grafo finito direcionado, onde os estados são representados pelos nós e as transições entre os estados são representadas por setas valoradas, representada na Figura §2.8:

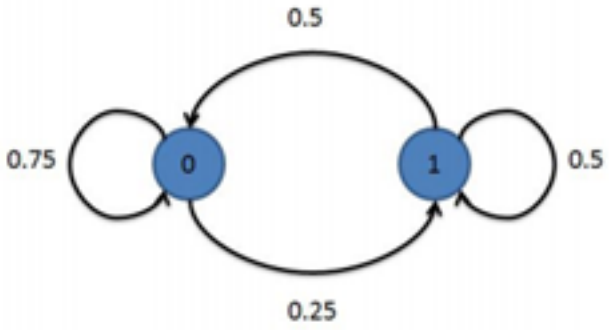


Figura 2.8: *Diagrama de transição de dois estados*

Observa-se Figura §2.8 que o somatório das probabilidades de transição de um estado i para todos os outros estados da cadeia de Markov é sempre igual a 1.O passo 0 tende a seguir ao passo 1 com a probabilidade de 0.25 e ficar no passo 0 com probabilidade 0.75, o passo 1 tende a seguir ao passo 0 com a probabilidade de 0.5 e com a probabilidade 0.5 de permanecer nesse passo.

Em geral não tem como definir quantos estados possui um sistema de uma Cadeia de Markov, o que se pode fazer é especificar as probabilidades para cada um dos estados possíveis. Para tanto cada matriz de transição possui um vetor coluna de probabilidade que representa os s estados. Um vetor de probabilidades de estado de uma cadeia de Markov de tempo discreto é chamado de estacionário se as transições descritas pela matriz P não modificam essas probabilidades de estado, ou seja, $vj = \sum v_i p_{ij}$ [6].

$$P = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_S \end{bmatrix} \tag{2.6}$$

O estado v_1 representa a probabilidade de o sistema estar no primeiro estado, v_2 representa a probabilidade de estar no segundo estado e vs representa a probabilidade do sistema estar no s-ésimo estado.

Um vetor de estado de uma Cadeia de Markov torna-se estacionário quando se conhece o vetor inicial e com ele faz-se um número significativo de iterações, os vetores de estado convergem para um vetor fixo.

A escolha da utilização das Cadeias de Markov de Tempo Discreto para este estudo justifica-se, pois é um caso particular de processos estocásticos e, portanto utilizado para sistemas de eventos discretos, possibilitando identificar e analisar os gargalos de desempenho na formação de filas na modelagem de um sistema. Apesar de amplamente utilizadas, as Cadeias de Markov possuem problemas fundamentais, sendo o principal deles, quando um modelo possui um elevado número de estados tornando-o de difícil analisar, sendo impraticável a solução dos sistemas de equações analíticos. Para tanto o estudo, baseia-se em técnicas de simulação para obter os dados da Cadeia de Markov de tempo discreto.

2.4 Resumo do Capítulo

Neste capítulo foram elencados os principais conceitos sobre as temáticas que envolvem a pesquisa. De acordo com a abordagem, os modelos das soluções de integração são projetados utilizando a tecnologia Guaraná DSL. Os engenheiros de software, para configurar uma solução de integração, necessitam configurar uma série de adaptadores para se comunicar com as aplicações que estão sendo interligados, não necessitando, para isso, ter conhecimento sobre tecnologias de integração. A transformação de modelos DSL Guaraná em código executável pode ser obtida usando uma engenharia orientada por modelos. Além disso, foram especificados os tipos de modelos que caracterizam uma solução de integração. Nesse contexto, abordou-se a simulação como um campo de pesquisa que lida com a experimentação de modelos que permitam fazer previsões sobre o comportamento e o desempenho dos sistemas de reais. Por último, foi apresentada a técnica matemática de Cadeias de Markov. Essa técnica adota modelos probabilísticos na sua teoria, na qual auxilia a análise de desempenho do modelo solução de integração que pretende-se estudar.

Capítulo 3

Trabalhos Relacionados

Ensinar não é transferir conhecimento,
mas criar as possibilidades para a sua própria produção ou a sua construção.

Paulo Freire, Pedagogo e filósofo (1921-1997)

Neste capítulo é apresentado alguns trabalhos que propõem a utilização de técnicas matemáticas e ferramentas de simulação a fim de validar os dados obtidos ao longo da pesquisa. A partir do estudo sobre o formalismo de Cadeias de Markov de Tempo Discreto (DTMC) com a utilização de ferramentas de simulação para verificar os modelos descritos, buscou-se relacionar alguns trabalhos que utilizam a técnica matemática e verificam com o auxilio de ferramentas de simulação. Neste sentido, existem diferentes abordagens e inúmeras obras na literatura que contêm a utilização de técnicas matemáticas juntamente com o processo de simulação. No entanto não há na literatura trabalhos que relacionam integração de aplicações empresariais, utilizando técnicas matemáticas e simulação para verificação de modelos. As seções subdividem-se em duas partes: a primeira parte está descritos trabalhos que utilizam a técnica matemática com DTMC com distintas ferramentas de simulação, Seção §3.1. E, a segunda parte está descrito os trabalhos relacionados que utilizam DTMC com a ferramenta de simulação PRISM, Seção §3.2. Por fim, a Seção §3.3 apresenta o resumo de capítulo.

3.1 Verificação de um modelo de DTMC utilizando ferramentas de Simulação

Ahluwalia e Singhal [2] propõem modelar a performance da arquitetura de comunicação de uma máquina massivamente paralela. Para garantir o foco nas análises, os autores restringiram-se a uma máquina SIMD de grão fino, a *Thinking Machine Corporations Connection Machine* CM-2. De acordo com os autores, essa máquina apresenta um paralelismo massivo e pode suportar várias formas de comunicação entre processadores. Os autores desenvolveram um modelo da DTMC da arquitetura de rede do CM-2 (*Connection Machine*) para computar o tempo de atraso introduzido pelos algoritmos de roteamento e pela arquitetura da rede.

Para conseguir modelar o desempenho da máquina, os autores desenvolveram um modelo da Cadeia de Markov de um único nó de roteador, já que a modelagem de um sistema inteiro de roteamento seria muito mais complexa. Como o estado de um *buffer* do roteador muda a cada ciclo, os autores modelaram um modelo de DTMC, e calcularam a matriz de probabilidades de transição e as probabilidades dos estados estáveis da Cadeia de Markov.

Os autores realizaram várias simulações em SIMPAS, com vários valores de entrada dos modelos desenvolvidos. Com os resultados obtidos, puderam concluir que as análises apresentadas estimam o desempenho da rede de roteadores com alta precisão. Devido a limitações de implementação, foi possível simular apenas redes de até 8 graus. Porém, devido ao fato de que o erro encontrado desapareceria com uma rede maior que 8 graus, a alta precisão permaneceria.

Gomes e Wanke [25] propõem aplicar o conceito de Cadeias de Markov na gestão de peças de reposição. Os autores ressaltaram que em uma cadeia de Markov aplicada à gestão de estoques, os estados possíveis denotam as diferentes posições de estoques que podem ocorrer ao longo do tempo. Nesse estudo, os autores buscaram modelar o consumo e a política de estoque de peças de reposição através de cadeias de Markov. Através da propriedade de convergência das Cadeias de Markov, é possível inferir a distribuição de probabilidades da posição de estoque, e a partir destes valores, determinar indicadores como estoque médio, probabilidade de falta e probabilidade de ressuprimento.

Os autores destacaram como principal objetivo da pesquisa a aplicação dos principais conceitos de Cadeias de Markov na gestão de estoques de

peças de reposição descartáveis. Almejaram determinar um procedimento heurístico (não otimizante) de apoio à decisão e escolha dos principais parâmetros da política de estoques (S, s) - nível máximo de estoque e ponto de pedido - a partir do histograma de possíveis estados para a posição final de estoque.

Os autores constataram que o método de Cadeias de Markov é uma alternativa computacionalmente mais eficiente que a simulação em planilhas no Excel. Tanto via simulação computacional quanto via Cadeias de Markov, testou-se um nível máximo de nove unidades em estoque e um ponto de pedido de seis unidades, com demanda média de duas unidades por período de tempo, obedecendo à distribuição de Poisson. Os resultados sugerem que ambas as distribuições se equivalem. Entretanto, os ganhos em relação a um menor esforço computacional e tempo de execução são substanciais na modelagem por Cadeias de Markov.

James et al. [30] apresentaram dois algoritmos de estimativa de parâmetro de amostras rápidas das cadeias de Markov homogêneas, observado em um ruído branco de Gaussian. Os algoritmos que os autores propuseram são obtidos através da discretização das equações diferenciais estocásticas robustas envolvidas na estimativa de modelos ocultos de Markov, através do algoritmo EM (*Expectation-Maximization*). O primeiro algoritmo é baseado na discretização robusta dos filtros de tempo contínuo, descobertos por Elliott, para estimar quantidades usadas no algoritmo EM. O segundo é baseado na discretização de facilitadores de tempo contínuo, produzindo as equações de reestimação de Baum-Welch.

Os autores destacaram que a escolha de discretização resulta em equações que são idênticas às obtidas por derivação dos resultados em tempo discreto. Os algoritmos são obtidos pela discretização robusta de equações diferenciais estocásticas envolvidas na estimativa de tempo contínuo dos Modelos Ocultos de Markov (HMM) através do algoritmo EM. O algoritmo EM é baseado em filtro e tem requisitos de memória negligenciáveis, ou seja, independe do número de observações.

Sandmann [54] investiga a importância de uma amostragem ideal para estimar probabilidades de estado em DTMC, tanto ao longo do infinito quanto em estado estacionário. A importância de uma amostragem é uma técnica de redução de variância para uma eficiente simulação através de uma mudança de medida. Em particular, ele pode ser aplicado para a simulação de eventos raros (eventos que ocorrem com uma probabilidade extremamente pequena) de Cadeias de Markov. Tais eventos raros são importantes para determinar substancialmente o desempenho e a confiabilidade do

sistema. Assim, para estimar eventos raros, é necessário modelagem de simulação eficiente e análise. Uma técnica comum na aplicação da importância de uma amostragem para o processo Markoviano em redes de filas é mudar a chegada de taxas de serviço, o que corresponde à evolução das taxas de transição dos de tempo contínuos da cadeia de Markov, ou de transição probabilidades subjacentes das DTMC.

Uma das principais vantagens da importância de uma amostragem é que sempre existe um estimador de variância zero, enquanto a maioria das técnicas de redução de variância só pode alcançar uma quantidade limitada de redução de variância. No entanto, o estimador ótimo explicitamente depende de uma quantidade desconhecida para ser estimada e, portanto, não é normalmente conhecida com antecedência. Não haveria necessidade de estimativa por simulação, se a quantidade é conhecida. Como uma grande desvantagem da importância de uma amostragem, é que não garante redução de variância, podendo aumentar a variância, inclusive, até o infinito. Assim, a aplicação adequada é necessária. O autor ressalta que os resultados mostram que uma amostragem ótima para as cadeias Markov não pode ser realizada através da simulação de uma cadeia de Markov alternativa, uma vez que as probabilidades de transição em cada etapa devem depender do processo já simulado.

3.2 Verificação do modelo de DTMC utilizando a ferramenta PRISM

Abrahám et al. [1] apresentaram um algoritmo de verificação de modelos, utilizando DTMC. O algoritmo tem como base a detecção e captação de componentes fortemente ligados, oferece contraexemplos, que podem ser refinados interativamente pelo usuário. Durante a verificação do algoritmo, todas as informações para as fases da recursividade são armazenadas. Isso permite que um usuário que precisa obter um contraexemplo para uma determinada propriedade tenha acessibilidade para formular um contraexemplo abstrato. No final, este produz, de maneira sucinta, contraexemplos que são construídos de acordo com a estrutura do DTMC.

Nesse trabalho, os autores investigaram o problema de um algoritmo de verificação de um modelo de árvore lógica de computação probabilística (PCTL). Introduziram um método que calcula o resultado, verificando o modelo para uma DTMC e uma propriedade de acessibilidade ilimitada por abstração recursiva. Durante este cálculo, é armazenada informação suficiente para criar contraexemplos em níveis diferentes de abstração com

contraexemplos concretos. Essa abordagem mostrou-se eficiente e competitiva com as verificações do modelo padrão (que não fornecem informações de diagnóstico), aplicando a nossa implementação do protótipo existem alguns *benchmarks* amplamente utilizados.

Clément et al. [12] propõem um estudo a partir da observação de uniformidade dos modelos PP (Protocolos de População), que permite aplicar um contador de abstração. A contribuição é destacar duas receitas para a verificação da propriedade de estabilização corretas em modelos com tamanhos PP de população finita. Para uma população de k agentes britânicos, $\pi = \{\pi_1, ..., \pi_k\}$ denota o conjunto de agentes. Cada agente no sistema é modelado como uma máquina de estado finito, que representa o programa do agente. Esses programas são uniformes: cada agente executa a mesma máquina de estado finito, de tal forma a não depender do número de agentes do sistema. O modelo fortemente anônimo: aos agentes não é possível armazenar um identificador exclusivo. Quando dois agentes se encontram, eles podem interagir e realizar uma mudança em seus estados. A rede de comunicação subjacente é um gráfico completo, cada par de agentes podem se encontrar. O estado inicial de cada agente é definido pelo protocolo. O objetivo de um PP é calcular por cada agente, o mesmo valor, que é será a saída do protocolo. O valor a ser calculado é o valor de uma função do estado inicial do protocolo.

Para a verificação os autores utilizaram o software PRISM, podendo lidar de forma muito eficiente com o modelo de verificação dessa propriedade sobre as abstrações do modelo PP. PRISM é uma ferramenta para modelagem formal e análise por modelo de verificação dos sistemas que exibem comportamento aleatório ou probabilístico. Modelos são descritos utilizando a linguagem PRISM, uma linguagem simples, baseada no estado. Um modelo probabilístico de verificação como PRISM permite calcular o tempo médio esperado de convergência, ou seja, o número de interações que ocorrem antes da convergência. Para esclarecer o modelo tem-se o tempo médio de convergência para dois problemas clássicos: (1) transmissão de epidemia: uma informação transmitida por alguns agentes devem chegar a todos os agentes; e (2) eleição do líder: a partir de qualquer número de líderes, o protocolo converge para um estado com no máximo um líder. Para ajustar essas *receitas*, os autores utilizaram teoremas já existentes que envolvem DTMC.

Basagiannis et al. [4] introduziram a verificação de um modelo probabilístico como uma abordagem de uma ferramenta assistida, viável para quantificar sistematicamente ameaças de segurança de Negação de Serviço (DoS). Ameaças DoS são expressas como propriedades de acessibilidade probabilísticos, que são automaticamente verificados através de uma DTMC,

que representa os participantes de protocolo e do modelo invasor. A análise de verificação do modelo probabilístico é realizada no software PRISM. A abordagem da verificação do modelo probabilístico é baseada em transições rotuladas entre estados modelos, com informações sobre a probabilidade de virem a ocorrer.

O modelo DTMC que corresponde a uma descrição da linguagem PRISM é construída como a composição paralela de seus módulos calculando o espaço de estados alcançáveis do modelo e descartando quaisquer estados inacessíveis. Em cada estado, há um conjunto de comandos (pertencentes a qualquer um dos módulos), que são ativados. A escolha entre qual comando a ser executado é probabilística, sendo cada comando ativado, selecionado com igual probabilidade. O cálculo subjacente em PRISM envolve uma combinação de (i) algoritmos Gráfico-teórico, para análise qualitativa acessibilidade e verificação modelo probabilístico; (ii) o cálculo numérico (agentes de resolução iterativos), para o modelo probabilístico quantitativa verificar que no caso de uma DTMC implica a solução de sistemas de equações lineares.

Os autores desenvolveram um modelo probabilístico para a base protocolo de troca HIP (identidade do protocolo de acolhimento) que utiliza um mecanismo de quebra-cabeça do cliente para proteção contra DoS. Os resultados obtidos fornecem informações valiosas sobre a gravidade da ameaça DoS demonstrada em diferentes casos de implementação de protocolo. Os autores acreditam que o método proposto possa ser utilizado no desenho de novos protocolos de segurança e implementações do protocolo.

Kwon e Agha [36] descreveram uma nova forma de modelagem de sensores de redes com DTMC e expressaram as propriedades agregadas da rede, tanto nos seus estados transitórios quanto no seu estado de equilíbrio. Fornecendo uma maneira de estimar uma DTMC, sugerindo um teste estatístico para assegurar com alta probabilidade de que a rede de sensores obedece à propriedade de Markov, e que a estimativa é suficientemente precisa, os autores mostraram um novo método para o controle daquelas propriedades que podem ser aplicadas ao modelo. Para isso, usaram os resultados experimentais para ilustrar e validar o método com a simulação no PRISM.

Os autores modelaram a dinâmica de transição entre estados como uma DTMC. Isso é razoável para muitas aplicações rodando em rede de sensores sem fio (WSN) como o seu comportamento corresponde a um sistema de transição probabilística: Levam a um estado de suspensão para economizar energia com certa probabilidade e, a fim de minimizar colisões, enviam pacotes com base na escolha probabilística. Escolheram um modelo de DTMC

em vez de um modelo de Cadeia Markov Tempo Contínuo, porque a interpretação do tempo é discreto em sua lógica de avaliação de desempenho.

Os autores concluíram que os resultados dos experimentos sugerem uma estimativa e controle dos métodos de avaliação de desempenho com base no modelo, que pode ser útil para estabelecer algumas propriedades agregadas de certas redes de sensores de grande escala. Dada a natureza de aplicações em redes de sensores, muitas serão adequadamente modeladas como DTMC. Podem-se modelar subgrupos da rede de sensores como DTMC.

Kumar e Vasudevan [32] introduziram o SHARPE (Análise Estatística de Alto Nível e Desempenho Rigoroso de Estimativa), uma metodologia sistemática rigorosa para verificar a correção de design em RTL (Registro do Nível de Transferência) na presença de variações. Descrevem SHARPE no contexto da computação invariante de atrasos estatísticos, na presença de variações de entrada. O código-fonte RTL utiliza técnicas de análise de programas estáticas para calcular probabilidades. Modelaram os módulos probabilísticos RTL em uma DTMC, que são então verificados formalmente para invariantes probabilísticas usando PRISM, um verificador de modelo probabilístico.

As etapas do SHARPE são as seguintes: São determinados correlações do sinal, usando análise estática de código-fonte RTL; Em seguida, é convertido em um módulo RTL de um modelo de DTMC, com as probabilidades de transição derivados de distribuições de probabilidade de entrada, bem como a análise estática; Essa DTMC é convertida para um modelo de recompensa, onde os estados são anotados por uma função de custo, que neste caso é o tempo de atraso. Obtiveram macro modelos de atraso a partir de nível de entrada que podemos usar para atribuir recompensas para o modelo RTL-DTMC. A verificação do modelo probabilístico é feito no PRISM, que permite calcular as invariantes probabilísticas em relação ao tempo.

Os autores concluíram apresentando uma técnica de análise estática no código-fonte RTL para calcular automaticamente as dependências do sinal interno e correlações de sinal para a criação de um modelo DTMC. SHARPE é uma nova metodologia para a geração de invariantes probabilísticas nos níveis mais altos de design do sistema. A técnica é ilustrada na descrição RTL do caminho de dados de OR1200, um processador de código aberto incorporado. Demonstraram a escalabilidade melhorada do SHARPE aplicando raciocínio de composição para a verificação de modelo probabilístico. Essa técnica pode ser usada para modelar diferentes fontes de variação.

Kumar e Vasudevan [33] estenderam SHARPE para executar a verificação de temporização no RTL no contexto de variações de processo. Através

da obtenção de um conjunto de variações de processos conscientes conseguese modelos de atraso RTL e, correspondentemente, modificar as etapas já existentes de SHARPE. É ilustrado SHARPE na descrição RTL do caminho de dados de OR1200, um processador de código aberto incorporado. Também é aplicado o SHARPE para outros projetos RTL intensivos de dados, tais como componentes não-triviais de sistemas de comunicação e alguns desenhos de referência. Estenderam o SHARPE por considerar as variações de processo como a principal fonte de estatísticas. Nesse contexto, as interpretações das etapas na metodologia SHARPE de uma maneira diferente. Construíram modelos macro para o atraso variável de operadores RTL, por representá-los como variáveis aleatórias gaussianas. São usadas as variáveis de atraso de Gauss como as variáveis do estado DTMC. Cada estado de uma DTMC corresponde a uma instanciação do *chip* que é obtida tomando um conjunto único de amostras das distribuições de atraso de Gauss. É definido um modelo de recompensa por marcação de todos os estados, (ou seja, *chips*), onde a restrição de tempo não é violada. Finalmente, é usado o PRISM para calcular a fracção de aparas que satisfazem a especificação de tempo, ou seja, o rendimento.

As principais contribuições dos autores com este trabalho são os seguintes: Introduziram com SHARPE, a noção de verificação da concepção de variação com reconhecimento em RTL. No entanto, aplicaram SHARPE para análise de tempo probabilística formal na RTL. SHARPE é igualmente competente na realização de análise estatística na RTL de sincronismo em relação a variações de padrão de entrada, bem como no que diz respeito a variações do processo. Com SHARPE, apresentaram uma ferramenta de CAD (Desenho Projetado pelo Computador) que pode auxiliar na exploração RTL design, para melhorar o pior caso de projetos, fornecendo uma análise abrangente de tempo estatístico. Os autores descrevem SHARPE como uma solução integrada para SSTA (Analise Estatística do Tempo Estático) formais no nível RT, considerando tanto as variações de entrada ou variações de processo como fonte de estatísticas da tempestividade.

Norman et al. [41] apresentaram uma implementação do controle para a probabilística de π-cálculo, sendo este um processo algébrico que suporta a modelagem da simultaneidade, mobilidade e comportamento probabilístico de um modelo discreto. Técnicas de verificação formal para este cálculo têm aplicações em vários domínios, incluindo protocolos de rede móvel *ad-hoc* e protocolos de segurança aleatória. Apesar disso, nenhuma implementação de verificação automatizada existe. Os autores mostram um procedimento automático para a construção do processo de decisão de Markov que representa um processo de π-cálculo probabilístico. Este pode, então,

ser verificado utilizando verificadores de modelos probabilísticos existentes, tais como PRISM.

Os autores demonstraram a viabilidade da implementação de modelo de verificação probabilístico de π-cálculo. A variante utilizada no cálculo para que as técnicas fossem aplicáveis, provou ser suficientemente expressiva para os domínios de aplicação adequadas e ainda passíveis de análise com ampliações e adaptações de ferramentas de verificação existentes. Além disso, mostraram, através da sua aplicação, a eficiência da abordagem.

Paulevé et al. [47] propõem uma técnica para ajustamento de características temporais dentro de cálculos estocásticos, apresentando a construção do fator de absorção de estocasticidade no cálculo estocástico clássico, com taxas exponenciais. Os autores também apresentaram ferramentas para manipular esse fator de absorção de estocasticidade e sua ligação com os intervalos programados para disparar transições.

Os autores conseguiram obter a absorção de estocasticidade trocando a distribuição estocástica por ações disparadas pela distribuição Erlang. Apresentaram uma tradução dos cálculos estocásticos da distribuição de Erlang em uma distribuição exponencial. Com isso, os autores conseguiram simular e analisar modelos ajustados usando grandes painéis de ferramentas padrões. Os modelos são expressos em termos de Cadeias de Markov, que podem então ser eficientemente simulados e verificados no modelo. O modelo-controle probabilístico dos cálculos estocásticos com as distribuições Erlang. foram obtidos usando PRISM, com uma construção fatores de absorção de estocasticidade dentro de modelos PRISM, obtidos de uma tradução dos cálculos estocásticos.

Foi concluído pelos autores que enquanto os modelos ajustados ainda são eficientes, o modelo-controle pode sofrer uma explosão de estado do espaço. Também concluíram que sua abordagem para a modelagem de um sistema biológico tem sua utilidade, já que pode ser usado para a modelagem de processos computacionais em geral.

3.3 Resumo do Capítulo

Nesse capítulo foram apresentados alguns trabalhos encontrados na literatura que utilizam DTMC. Dentre as abordagens, nota-se uma incidência maior na utilização da ferramenta de simulação PRISM. A ferramenta PRISM é uma ferramenta para modelagem formal e análise por modelo de verificação dos sistemas que exibem comportamento aleatório ou probabilístico.

Modelos são descritos utilizando a linguagem PRISM, uma linguagem simples, baseada em estados. Essa pesquisa diferencia-se de todos os trabalhos apresentados por não apresentar evidencias de simulação de eventos discretos visando á análise de modelos conceituas de soluções de integração empresariais.

Parte III

Proposta e Experimentação

Capítulo 4

Desenvolvimento e Experimentação do Modelo Formal

A Matemática apresenta invenções tão sutis que poderão servir não só para satisfazer os curiosos, como também para auxiliar as artes e poupar trabalho aos homens.

René Descartes, filósofo, cientista e matemático francês (1596-1650)

Para que seja possível a análise do comportamento e a descoberta de gargalos de desempenho ainda na fase de projeto das soluções de integração, sem a necessidade de implementação, precisa-se ter um conhecimento detalhado sobre o modelo de estudo. Nesse sentido, neste capítulo é apresentado o estudo de caso desta dissertação, um modelo conceitual para a solução de integração de um problema real projetado na tecnologia Guaraná. Com a análise do modelo conceitual da solução de integração desenvolvida na tecnologia Guaraná, obtém-se o Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov. Para tanto, é necessário mostrar à equivalência dos elementos de uma solução de integração projetada na tecnologia Guaraná com os elementos que compõem as Cadeias de Markov. Para demostrar a equivalência, utiliza-se uma ferramenta de simulação para a verificação do modelo, essa ferramenta possui extensões probabilísticas da lógica temporal que visa detecção de gargalos e análise de desempenho.

A partir das variáveis observadas, é possível a identificação dos gargalos de desempenho da solução de integração. A análise é realizada a partir

dos resultados das simulações. Esses resultados são obtidos através da Ferramenta de Simulação PRISM. Na Seção §4.1 é apresentado o problema de integração. A descrição do ecossistema de software da solução de integração é detalhada na Seção §4.2. Na Seção §4.3 é realizada uma descrição detalhada do modelo conceitual considerando o fluxo de trabalho das mensagens na solução. Na Seção §4.4 é apresentado a equivalência dos elementos da tecnologia Guaraná com Cadeias de Markov. Na Seção §4.5 são apresentadas as distribuições de entrada. O estudo de caso é apresentado na Seção §4.6. Na Seção §4.7 são apresentados resultados experimentos obtidos através da ferramenta de simulação. E, na Seção §4.8 o Resumo do Capítulo.

4.1 Problema de Integração

Trata-se de um sistema de gerenciamento de telefonemas pessoais utilizado na Unijuí, proposto por Frantz [24]. No momento em que o funcionário realiza uma ligação com o telefone da instituição, faz-se necessária a verificação de sua finalidade, ou seja, se é uma ligação pessoal ou destinada ao trabalho. Caso seja uma ligação pessoal, necessita ser efetuado o débito da ligação na folha de pagamento do funcionário. O problema em questão ocorre na Universidade Regional do Noroeste do Estado de Rio Grande do Sul (UNIJUÍ). O objetivo é automatizar o faturamento de telefonemas pessoais realizados pelos funcionários com o sistema de telefonia da Universidade.

4.2 Ecossitema de Software

É apresentado na Figura §4.1, o problema de integração da Solução de Gerenciamento da Central Telefônica da universidade UNIJUÍ, proposto por Frantz.

O ecossistema de *software* que compõe a solução de integração envolve cinco aplicações distintas: Central Telefônica, Recursos Humanos, Servidor de E-mail, Serviço de Mensagens e Folha de Pagamento. Cada aplicação é executada em uma plataforma diferente; Recursos Humanos e Folha de Pagamento são sistemas legados desenvolvidos pela Universidade e as outras aplicações foram adquiridas. Além disso, o Servidor de E-mail provê interfaces POP3 e SMTP, e as outras aplicações foram projetadas sem levar em conta a possibilidade de integração.

A Central Telefônica registra todas as chamadas que cada funcionário realiza dos telefones da Universidade. Isso é possível por que cada funcionário

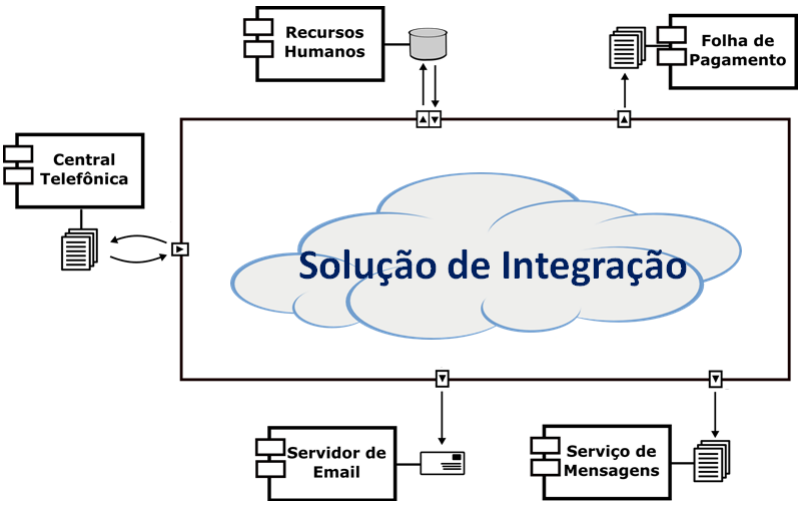


Figura 4.1: *Solução de Integração para o Problema de Gerenciamento de Chamadas Telefônicas. Fonte: Frantz [24].*

possui um código de acesso pessoal, que precisa ser digitado antes de discar o número que se deseja ligar. Este código é utilizado para correlacionar as ligações efetuadas pelo funcionário com as informações da aplicação de Recursos Humanos e Folha de Pagamento. A cada mês, uma análise é realizada para encontrar quais chamadas possuem um custo e não possuem relação com as atividades do trabalho; tais atividades são debitadas dos funcionários, utilizando a Folha de Pagamento. A aplicação de Recursos Humanos fornece informações pessoais sobre os funcionários e todo mês a Folha de Pagamento calcula o salário de todos os funcionários, incluindo as deduções. O Servidor de E-mail e o Serviço de Mensagens executam, respectivamente, o e-mail da Universidade e serviços do sistema de mensagens, que são utilizados com a finalidade de notificação.

Todas as chamadas registradas pela Central Telefônica, exceto as chamadas gratuitas, são transformadas em débito no sistema de Folha de Pagamento. Após a concretização do desconto, o funcionário é notificado por e-mail ou por mensagem de texto sobre as chamadas realizadas para que estejam cientes da dedução na folha de pagamento. Entretanto, ela deve ser submetida a um processo de filtragem, para impedir que estes sistemas recebam uma mensagem vazia ou com informações incompletas do destinatário.

4.3 Solução de Integração

Dentro da solução de integração desenvolvida, são transmitidas mensagens, as quais contêm informações que são enviadas pelas aplicações. A solução de integração projetada na linguagem da tecnologia Guaraná é composta de um processo de orquestração que coordena, de forma exógena, as aplicações envolvidas na solução. Algumas portas usam arquivos de texto para comunicar-se indiretamente com as aplicações da Central Telefônica, Folha de Pagamento e Serviço de Mensagens. Para obter informações do banco de dados da aplicação de Recursos Humanos, utiliza-se uma porta de solicitação.

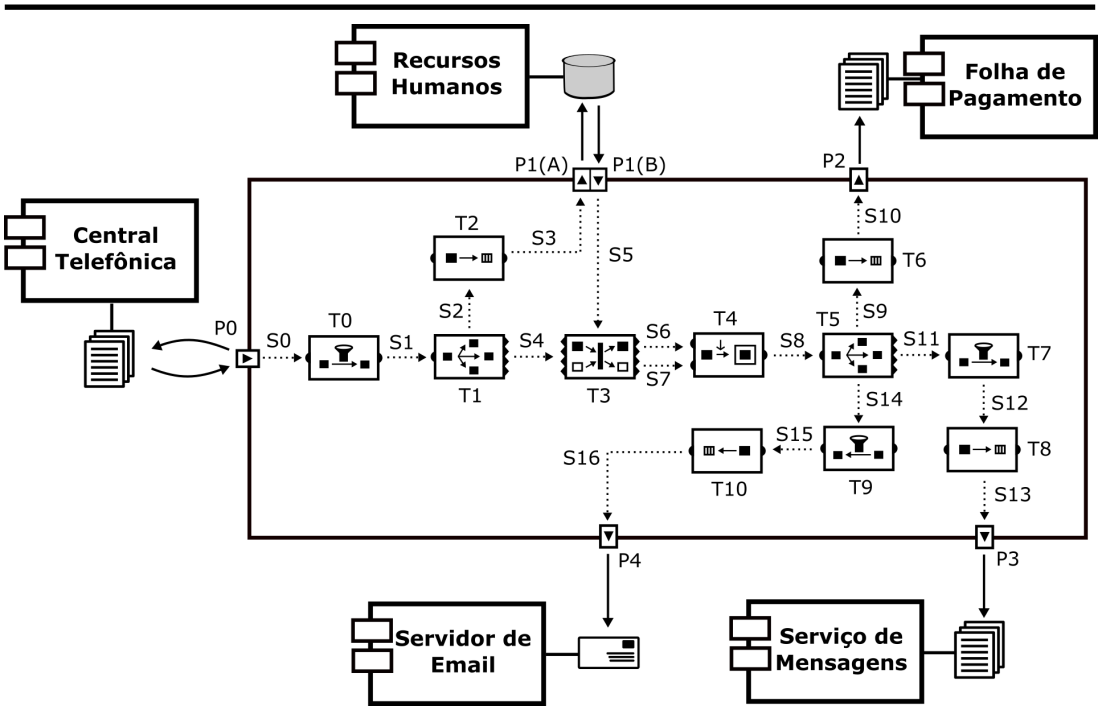


Figura 4.2: *Modelo Conceitual Desenvolvido Através da Linguagem Guaraná para Representar uma Solução de Integração para o Problema de Gerenciamento de Chamadas Telefônicas. Fonte: Frantz [24].*

Baseado na Figura §4.2 percebe-se que o fluxo de trabalho inicia na entrada da porta P0 que, periodicamente, pesquisa a Central de Chamadas, com

o intuito de encontrar novas ligações telefônicas. Cada telefonema resulta em uma mensagem, que é adicionada pela porta P0 no *slot* S0, possibilitando que a tarefa T0 (*Filter*) possa ser executada. Sempre que uma tarefa estiver executando as mensagens que chegam, aguardam nos *slots* formando uma fila, e serão selecionadas para o processamento pela disciplina FIFO.

A tarefa T0 filtra e descarta todas as chamadas que resultam nas mensagens de ligação gratuita e encaminha somente aquelas que têm custo para o *slot* S1, para serem executadas pela tarefa T1 (*Replicator*). Essa tarefa T1 (*Replicator*) realiza uma cópia da mensagem, que é encaminhada ao *slot* S2 para ser processada pela tarefa T2 (*Translator*), que traduz o conteúdo da mensagem para o formato da aplicação Recursos Humanos, e encaminha a outra cópia ao *slot* S3, para consultar a aplicação, por meio da porta de saída P1(A). As mensagens retornam por meio da porta de solicitação P1(B) e envia ao *slot* S5. Outra cópia é encaminhada para o *slot* S4, que aguarda uma mensagem correlacionada pelo *slot* S5 para ser processada pela tarefa T3 (*Correlator*).

A tarefa T3, ao executar, analisa as mensagens de entrada e produz, como saída, um conjunto de dados correlacionados, contendo no *slot* S6 informações encontradas nos Recursos Humanos e, no *slot* S7, as informações da Central Telefônica. Quando há mensagens correlacionadas, estas são conduzidas para a tarefa T4 (*Context-based Content Enricher*) que adiciona ao corpo da mensagem as informações que retornaram da aplicação Recursos Humanas com as informações oriundas da Central Telefônica. Após a mensagem transitar na tarefa T4, a mensagem é encaminhada para o *slot* S8, para que a tarefa T5 (*Replicator*) possa ser executada. A tarefa T5 (*Replicator*) realiza uma cópia da mensagem enriquecida e tem como finalidade de destino as aplicações da porta P3, referente à Folha de Pagamento, porta P4, referente à Serviço de Mensagens e porta P5, referente à Servidor de Email. Uma cópia é encaminhada para o *slot* S9, que tem como destino a tarefa T6 (Translator) que, ao executar, traduz o conteúdo da mensagem para o formato da aplicação Folha de Pagamento.

Após a mensagem ser processada pela tarefa T6, a mensagem é encaminhada para o *slot* S10 que destina à porta P2, informando as ordens de débito na folha de pagamento. Outra cópia é encaminhada para o *slot* S11, que tem como destino a tarefa T7 (*Filter*), que ao processar a mensagem, analisa e descarta a mensagem que não tiver a informação do número de telefone do funcionário. Após a mensagem ser filtrada, é encaminhada ao *slot* S12 que tem como destino a tarefa T8 (*Translator*), que ao executar, traduz o conteúdo da mensagem para o formato da aplicação Serviço de Mensagem e encaminha a mensagem ao *slot* S13, que tem como

destino a porta P3, que envia uma mensagem de texto para o telefone celular do funcionário, informando a cobrança. Outra cópia é encaminhada para o *slot* S14 que tem como destino a tarefa T9 (*Filter*) que, ao processar a mensagem, analisa e descarta a mensagem que não tiver a informação do endereço de email do funcionário. Após a mensagem ser filtrada, é encaminhada ao *slot* S15, que tem como destino a tarefa T10 (*Translator*), que traduz o conteúdo da mensagem para o formato da aplicação Servidor de Email e encaminha a mensagem ao *slot* S15, que tem como destino a porta P4, a qual envia um demonstrativo de gastos para o email do funcionário.

4.4 Equivalência dos elementos do Guaraná com Cadeias de Markov

Esta seção aborda a equivalência dos elementos de uma solução de integração projetada na tecnologia Guaraná - DSL com os elementos que compõem Cadeias de Markov. A caracterização de uma solução de integração como um sistema que apresenta um comportamento estocástico possibilita demonstrar a equivalência da solução com Cadeias de Markov. Para demostrar a equivalência, utiliza-se uma ferramenta de simulação para a verificação do modelo, a qual possui extensões probabilísticas da lógica temporal que visa detecção de gargalos e análise de desempenho.

Um processo estocástico representa, nesse contexto, a evolução do sistema de variáveis aleatórias no tempo. Em outras palavras, um processo com grau de incerteza, que evolui através de um conjunto de possíveis indicações, a partir de um dos possíveis estados.

Simulação do modelo estocástico, no geral, e o modelo de verificação, em particular, são importantes componentes no projeto e análise de sistemas de softwares críticos, com características de tempo probabilísticas, não determinísticas e reais. Os verificadores do modelo são capazes de analisar se um sistema satisfaz um estado/transição do modelo e uma especificação lógica temporal. Além da análise do sistema convencional, a verificação por modelo estocástico calcula a probabilidade de determinados eventos durante a execução do sistema, incluindo considerações de relacionamento temporais, Kwiatkowska et al. [35].

Dentre vários modelos probabilísticos existentes, Cadeias de Markov de Tempo Discreto permite especificar a probabilidade de fazer uma transição de um estado para outro. Os Processos de Decisão de Markov permitem tanto a modelagem de probabilidade como a não determinística, sendo de especial interesse para a solução de integração de modelagem formal. Em

modelos de Markov, estados futuros são influenciados apenas pelo estado atual e não existe influência de estados passados. Simulação baseada em Markov e modelo de verificação permitem análises de propriedades de sistemas, tais como baseada em passos, transitórios e propriedades de estado estacionário, Parker [46].

Cadeias de Markov de Tempo Discreto pode ser descrito como um modelo de transição de estado, aumentada com probabilidades. Formalmente, é descrito como um conjunto finito de estados (S), um estado inicial s_0 pertencente a S, uma matriz de probabilidade de transição (P) de SxS$\rightarrow$[0-1] onde a soma das transições de um estado devem ser 100%, uma função de rotulagem atribuindo aos estados um conjunto de proposições atômicas (L : S $\rightarrow 2^{AP}$). O tempo é comumente visto nos modelos Cadeia de Markov de Tempo Discreto como um intervalo de tempo discreto, homogêneo, e probabilidades de transição independentes de tempo. Uma execução de um modelo Cadeia de Markov de Tempo Discreto é representada por um caminho, com comprimento igual às transições do mesmo. Os cálculos de probabilidade do caminho permitem analisar o comportamento do sistema e a razão em propriedades, tais como acessibilidade probabilística. Propriedades quantitativas e qualitativas, incluindo acessibilidade e persistência repetidas, podem ser de interesse no contexto de uma análise de comportamento do sistema, e conduziu com a verificação do modelo probabilístico de Cadeia de Markov de Tempo Discreto. Modelagem de estruturas retributivas podem também ser usadas para representar benefícios ou características de custo. Transição retributiva (instantâneas) e retributiva de estado (acumulativas) são modeladaspelas funções retributivas rs : S $\rightarrow$ R $\geq$ 0 e rt : S$\times$S$\rightarrow$ R$\geq$0 respectivamente. Uma estrutura retributiva pode ser usada para medir o número de intervalos de tempo gasto em um estado ou a chance de que o sistema está em um estado específico depois de um certo número de intervalos de tempo, [34] [46].

Os processos de decisão de Markov, como a Cadeia de Markov de Tempo Discreto, modela um sistema como um conjunto discreto de estados e transições entre estados que ocorrem em intervalos de tempo discretos. Além disso, a decisão de Markov extende os modelos do processo DTMC através da permissão para escolhas não determinantes. Este tipo de modelagem é especialmente adequado para concorrência, ambientes desconhecidos e aplicação de cenários não específicos. Desde que os modelos da Cadeia de Markov de Tempo Discreto são totalmente probabilísticos, eles são incapazes de resolver alguns aspectos do sistema, como a escolha não determinante, [46].

Formalmente, um processo de decisão Markov é uma tupla (S, $_{sinit}$, Passos, L) onde S é um conjunto finito de estados, $S_{sinit} \epsilon$ S, e S é o estado inicial.

Os Passos: S$\rightarrow 2^{Act \times Dist(S)}$ é a função de probabilidade de transição, agindo com um conjunto de ações, Dist(S) o conjunto de distribuições de probabilidade discretas sobre S, e (L : S $\rightarrow 2^{AP}$) uma rotulagem com proporções atômicas. Um caminho em um processo de decisão Markov é uma sequência de estados e pares de ações/distribuições, isto é, $s_0(a_0,\mu_0)s_1(a_1,\mu_1)s_2$, representando a execução de um sistema. Resolvendo caminhos não determinantes (transformação do modelo em um DTMC) e escolhas probabilísticas, e então calcule uma medida de probabilidade sobre caminhos, [34] [46].

A Figura §4.3 (a) ilustra um modelo genérico dos elementos das Cadeias de Markov. Este processo baseia-se em modelos de Markov, que têm uma estrutura semelhante à do modelo conceptual de uma solução de integração concebido com a tecnologia Guaraná.

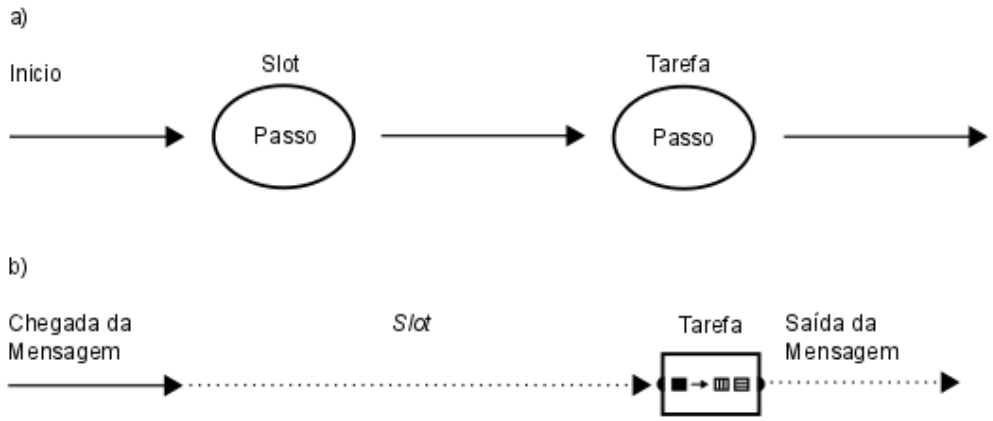


Figura 4.3: *Modelo genérico de uma representação em Cadeias de Markov (a) com a equivalência no Guaraná.*

Na Figura §4.3 (a), a partir do momento em que há uma mensagem na fila, o processo inicia, as mensagens chegam ao inicio do processo, e aguardam no sistema. Da mesma forma, a Figura §4.3 (b) ilustra um trecho de uma solução de integração desenvolvido com a tecnologia Guaraná, em que a chegada de mensagem representa inicio do processo. Em Cadeia de Markov, os Passos representam os *slots* e as Tarefas da tecnologia Guaraná.

4.5 Distribuição de Entrada

Para o caso de estudo apresentado nesta dissertação, observa-se o comportamento de uma solução de integração projetada na tecnologia Guaraná. Nessa solução de integração desenvolvida, são transmitidas mensagens, as quais contêm informações que são enviadas pelas aplicações. As mensagens chegam ao sistema de forma aleatória, apresentando um comportamento estocástico, ou seja, as chegadas ocorrem no tempo e no espaço de acordo com as leis da probabilidade; assim, é preciso conhecer qual a distribuição de probabilidade que descreve os tempos entre as chegadas dos clientes. Nesta Seção são apresentadas algumas distribuições de probabilidade que podem ser usadas para modelar os dados, posteriormente serão apresentados e analisados.

4.5.1 Poisson

A distribuição de Poisson é uma das distribuições mais usadas para analisar as variáveis discretas. Poisson representa a distribuição de probabilidades de uma variável aleatória que registra o número de ocorrências sobre um intervalo de tempo ou espaço específico.

O processo de Poisson tem propriedades importantes como: o número de eventos ocorrendo em um intervalo de tempo ou espaço é independente do número de eventos ocorridos no segmento anterior, pois o processo de Poisson não tem memória; a taxa média do processo l deve permanecer constante durante o período de tempo e espaço considerados; quanto menor o intervalo de tempo e espaço, menor a probabilidade de ocorrer mais de um evento naquele segmento. A probabilidade de ocorrência de dois ou mais eventos se aproxima de zero, quando o tamanho do intervalo se aproxima de zero.

A probabilidade de ocorrência de uma variável aleatória discreta x, tomando os seguintes valores: 0,1,2,...,n, quando regida por uma distribuição de Poisson, é dada por:

$$P_X = \frac{e^{-\lambda}\lambda^X}{X!} \tag{4.1}$$

Onde, P(x) é a probabilidade de x ocorrências em um intervalo e λ é o índice ou taxa de chegada dos clientes. E, a média e a variância da distribuição de Poisson são, respectivamente:

$$\mu = \sigma^2 = \lambda \Delta T \tag{4.2}$$

Da expressão acima, pode-se observar que no processo de Poisson, o número esperado de chegadas em um intervalo de tempo fixo é proporcional à taxa de chegadas e ao comprimento do intervalo de tempo, que é a razão pela qual o parâmetro l é referido como a taxa do processo de chegada.

4.5.2 Distribuição Exponencial Negativa

A distribuição exponencial é uma das distribuições contínuas amplamente utilizadas. A distribuição exponencial é frequentemente utilizada para modelar o tempo decorrido entre os eventos. Na teoria das probabilidades e estatísticas, a distribuição exponencial (distribuição exponencial negativa) é a distribuição de probabilidade que descreve o tempo entre eventos em um processo de Poisson, isto é, um processo em que os eventos ocorrem continuamente e de forma independente a uma taxa média constante. Definição: Dizemos que uma variável aleatória X não negativo tem a distribuição exponencial com parâmetro $\lambda > 0$ se:

$$P(X > t) = e^{-\lambda t}, \in \subset \mathbb{R}_+. \tag{4.3}$$

A densidade desta distribuição é dada por:

$$f(t) = \lambda\, e^{-\lambda t}, \forall t \in \subset \mathbb{R}_+. \tag{4.4}$$

em que f(t) é a função da densidade, sendo λ o ritmo de chegada e x o tempo. A média e variância de X são respectivamente dadas por:

$$E(X) = \int_0^\infty tf(f)dt = \frac{1}{\lambda}, var(X) = \int_0^\infty t^2 f(t)dt - E(X)^2 = \frac{1}{\lambda^2} \tag{4.5}$$

4.5.3 Distribuição de Erlang

Se X_1, X_2, X_3....X_K são variáveis independentes, identicamente distribuídas exponencialmente, a sua soma tem a distribuição de Erlang. A média (β) de cada distribuição exponencial e o número de variáveis aleatórias (k) são os parâmetros da distribuição. A média da distribuição exponencial é um numero real positivo e k é o valor inteiro positivo.

$$\{ f(x) = \frac{\beta^{-k} x^{k-1} e - \frac{x}{\beta}}{(k-1)!} para x > 0 \qquad caso contrário \qquad 0 \tag{4.6}$$

$$Média : k\beta, Variância : k\beta^2 \tag{4.7}$$

A distribuição de Erlang é usada em situações em que uma atividade ocorre em sucessivas etapas e cada fase é ajustada por uma distribuição exponencial. Para valores grandes de k, a distribuição de Erlang se aproxima da distribuição normal, sendo usada para representar o tempo requerido para completar uma tarefa.

4.6 Desenvolvimento do Modelo Formal

Nesta Seção é apresentado, partir do modelo conceitual projetado na tecnologia Guaraná, o Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov e, posteriormente, o modelo matemático. Para tanto, é necessário que as variáveis que serão observadas para a identificação dos gargalos de desempenho da solução de integração sejam apresentadas.

4.6.1 Variáveis a se Observar

Outra característica comum entre a solução de integração apresentada pelo Guaraná e um sistema de eventos discretos é a relação entre os seus elementos e estrutura de funcionamento. Na modelagem de sistemas de evento discreto, como uma solução de integração, a identificação e relação com as variáveis ao desempenho e a forma com que elas interagem entre si e com os outros elementos, auxilia a compreensão do sistema.

Na tecnologia Guaraná, as entidades são mensagens, ou seja, objetos que permitem a interação com o sistema. Um processo é uma sequência de atividades. Uma atividade ocorre entre dois eventos e, durante um evento, o estado do sistema pode mudar. A ocorrência de um evento na tecnologia Guaraná pode ser caracterizada como a chegada de uma mensagem a ser processado por uma tarefa. Cada vez que uma tarefa processa uma mensagem, o sistema muda seu estado. Uma tarefa ocorre entre dois eventos, para que possa ser caracterizada como uma atividade e, consequentemente, uma processo pode ser caracterizado como uma sequência de tarefas. Em geral, também é possível definir o tamanho de filas de espera, no entanto, em uma solução de integração projetada com a tecnologia Guaraná, o tamanho da fila, caracterizada como um *slot* pode conter um número infinito de mensagens

O modelo de filas M/M/1 apresentado por Kendall [31] é um modelo baseado na teoria das filas que é amplamente utilizada para eventos discretos e de simulação. Esse modelo também permite a construção e compreensão das ideias básicas e métodos de filas teoria. Segundo Prado [51] no modelo M/M/1 tanto as chegadas quanto o atendimento são marcovianos (o que é o mesmo que dizer que seguem a distribuição de Poisson ou a exponencial negativa) e em que temos um único atendente, sendo a população finita.

Prado [51] afirma que o processo de atendimento também é quantificado por uma importante variável randômica. A letra grega μ significa

Sistema	TS = Tempo médio de permanência no sistema NS = Número médio de clientes no sistema
Processo de chegada	λ = Ritmo médio de chegada IC = Intervalo médio entre chegadas
Fila	TF = Tempo médio de permanência na fila NF = Número médio de clientes na fila
Processo de atendimento	TA = Tempo médio de atendimento ou de serviço μ = Ritmo médio de atendimento de cada atendente

Tabela 4.1: *Principais variáveis utilizadas como medidas de desempenho do sistema [51]*

ritmo médio de atendimento, e TA tempo ou duração média do serviço ou atendimento.

A representação de um sistema de filas requer conhecimento sobre algumas variáveis randômicas (aleatórias) usadas, sendo as principais as seguintes:

- λ = ritmo médio de chegada
- IC = intervalo médio entre chegadas = 1/ λ
- μ = ritmo médio de atendimento
- TA = tempo ou duração média do serviço ou atendimento = 1/ μ

A Tabela §4.1 descreve as principais variáveis utilizadas como medidas de desempenho referentes ao sistema, processo de chegada, fila e processo de atendimento.

A Tabela §4.2 são descritas as fórmulas que tratam as principais variáveis randômicas do modelo M/M/1. Quando conhecida a distribuição de probabilidade que descreve o número de clientes no sistema podemos calcular os parâmetros da fila: número médio de clientes na fila, número médio de clientes no sistema, tempo médio durante o qual o cliente fica na fila, tempo médio durante o qual o cliente fica no sistema. Também podemos calcular a probabilidade de haver n clientes no sistema, ou seja a distribuição de probabilidades do número de clientes no sistema.

Nome	Descrição	Fórmula
NF	Número médio de clientes na fila	NF = $\frac{\lambda^2}{\mu(\mu-\lambda)}$
NS	Número médio de clientes no sistema	NS = $\frac{\lambda}{\mu-\lambda}$
TF	Tempo médio durante o qual o cliente fica na fila	TF = $\frac{\lambda}{\lambda(\mu-\lambda)}$
TS	Tempo médio durante o qual o cliente fica no sistema	TS = $\frac{1}{\mu(\mu-\lambda)}$
P_n	Probabilidade de haver n clientes no sistema	$P_n = (1-\frac{\lambda}{\mu})(\frac{\lambda}{\mu})^n$

Tabela 4.2: *Fórmulas que tratam as variáveis randômicas do modelo M/M/1 [51]*

4.6.2 Modelo de Simulação

A Figura §4.4 é ilustrado o esquema de funcionamento do processo de decisão de Markov, e visa auxiliar o entendimento do método utilizado no modelo que representa uma solução de integração. A partir do momento em que há uma mensagem na fila, o processo inicia pelo estado P_0. Esse estado representa que uma mensagem foi encontrada e retirada da fila. Em seguida, no estado T_0, são escolhidas apenas as mensagens que contém custo da ligação telefônica. A porta P_1 envia uma mensagem para o Sistema de Recursos Humanos, a fim de obter informações do funcionário.

A mensagem segue para o sistema de Folha de Pagamento, estado P_3, que desconta o valor da ligação realizada pelo funcionário. Após a concretização do desconto, o funcionário é notificado pelos estados P_4 e P_5, que representam, respectivamente, um sistema mensagens SMS e um servidor de *e-mail*. Entretanto, ela deve ser submetida a um processo de filtragem para impedir que estes sistemas recebam uma mensagem vazia ou com informações incompletas do destinatário.

O processo de decisão Markov representado como uma tupla M = (S, sinit, P, L) o exemplo presente na Figura §4.4 pode ser descrito como:

S conjunto finito de estados. Portas = P_0, P_1, P_2, P_3, P_4, *Slot* = S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16} e Tarefas = T_1, T_2,T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10} que representam um conjunto finito de estados, ou espaço de estado.

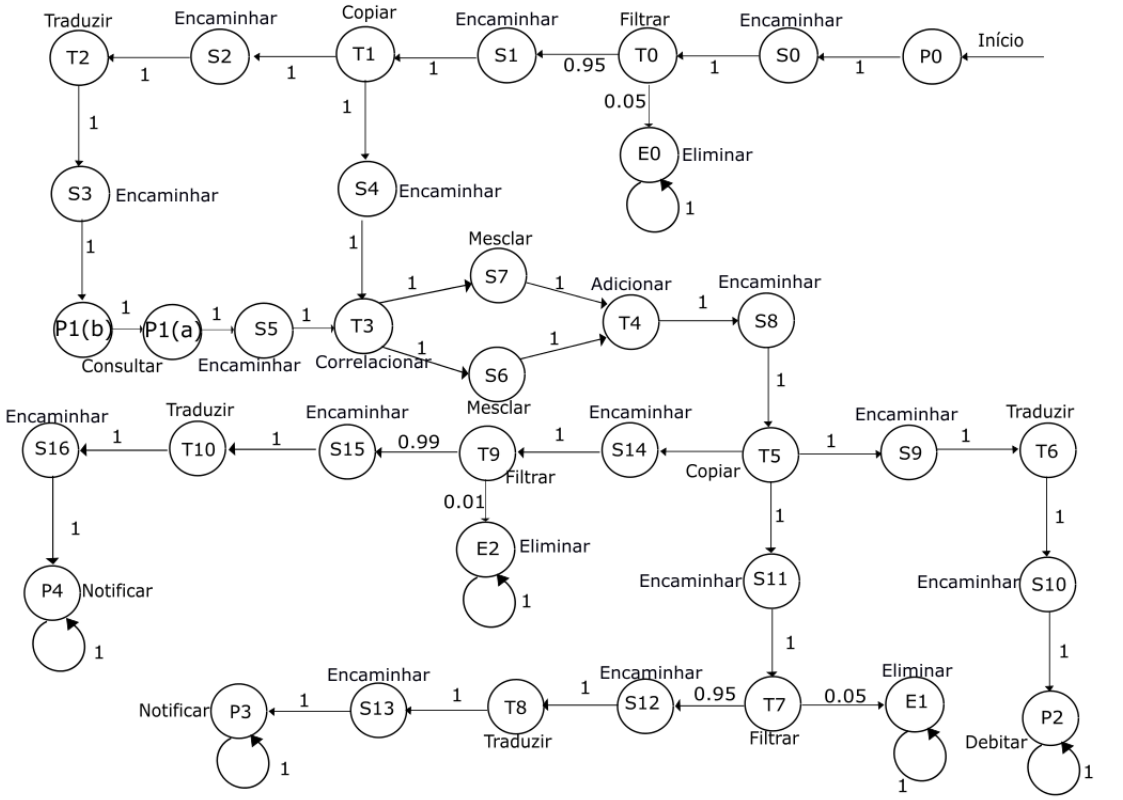


Figura 4.4: *Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov correspondente ao Modelo Conceitual.*

S_{sinit}, = s_0, o estado inicial.

P = SxS $\rightarrow$ $[0, 1]amatrizdeprobabilidadedetransio$.

L = A rotulagem com proposições atômicas ($L : S \rightarrow 2^{AP}$), que são simplesmente associadas aos estados no exemplo atual. E AP = [inicio, filtrar, copiar, traduzir, consultar, comparar, mesclar, debitar e notificar, encaminhar, eliminar], o conjunto de proposições atômicas.

4.6.2.1 Modelo Matemático

Cadeias de Markov de tempo discreto, as variáveis aleatórias X_0, X_1,..., X_n,... são discretas, o conjunto T é discreto ($T = 0, 1, ..., n, ...$), dada por:

$$P(X_t = x_j | X_{t_1} = x_{i_1}, X_{t_2} = x_{i_2}, ..., X_{t_n} = x_{i_n}) \tag{4.8}$$

E, processo estocástico X_t é uma Cadeia de Markov se a distribuição de X_t for independente de todos os estados prévios em que se encontrou a cadeia, com exceção do estado imediatamente anterior, isto é,

$$P(X_t = x_j | X_{t-1} = x_{i_{t-1}}, ..., X_2 = x_{i_2}, X_1 = x_{i_1}) = P(X_t = x_j | X_{t-1} = x_{i_{t-1}}) \quad (4.9)$$

Um modelo de Markov onde o espaço de estados I é discreto é denominado de Cadeia de Markov e é completamente descrito por sua matriz de transição de estados. Esta matriz e dinâmica, pois permite que as probabilidades de transição se modifiquem em função do tempo t, onde t e discreto.

Considere uma cadeia de Markov com N estados $x_n \in I$ e sejam $x_i, x_j \in I$. Denota-se $x_i(t)$ para significar que o processo está no estado x_i no tempo t.

Definição 1: Se p_{ij} e a probabilidade de transição do estado $x_i(t)$ para o estado $x_j(t+1)$, então a matriz $N \times N$, dada por

$$P_{ij} = Pr(X_{n+1} = j | X_n = i) \quad (4.10)$$

ou seja,

$$P = [p_{ij}] \quad (4.11)$$

denomina-se matriz de transição de estados da cadeia de Markov.

Observa-se que, na Definição 1, a soma das linhas da matriz P deve ser sempre igual a 1. A matriz de transição também pode ser dada por um diagrama de transições de estados. A Figura §4.5 mostra o diagrama de transições de estados para uma cadeia de Markov com apenas 2 estados.

Proposição 1: Para t arbitrário, tem-se que:

i. A probabilidade de transição do estado $x_i(t)$ para o estado $x_j(t+n)$ (em n passos) é dada por $P^n_{i,j}$;

ii. A matriz de transição de n passos, denotada por P_n, é calculada como a potência n da matriz de transição P, isto é,

$$P^n = P^n. \quad (4.12)$$

De acordo com a Figura §4.5, o diagrama de transição representa a transição entre dois estados, portanto a matriz de transição é uma matriz quadrada, e é representada pela matriz P:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{2x2} \quad (4.13)$$

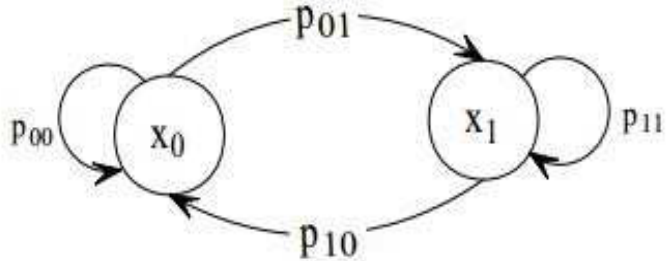


Figura 4.5: *Diagrama de transição entre dois estados da matriz de uma cadeia de Markov. Fonte: Dimuro et al. [17]*

Para simular um processo de Markov, considerando um estado inicial x_0, pode-se escolher um estado sucessor de acordo com as probabilidades p_0j, para $j = 1, ..., N$, determinando um novo estado x_1. Repete-se o processo para gerar o próximo estado, e assim sucessivamente. Devido à natureza probabilística do modelo, cada vez que esta simulação for repetida, é provável que uma sequencia diferente de estados seja obtida como resultado. Portanto, a única forma de analisar o processo e manter o registro das probabilidades de estar em um estado.

Definição 2: Seja $S_i(t)$ a probabilidade de que um processo de Markov esteja em um estado x_i no tempo t. Então o vetor

$$s(t) = \begin{pmatrix} S_1(t) \\ S_2(t) \\ \vdots \\ S_n(t) \end{pmatrix} \tag{4.14}$$

é denominado de vetor de distribuição de probabilidades de estado da cadeia de Markov no tempo t.

Seja $s^T(0)$ a distribuição inicial do processo (s^T é o vetor transposto de s) . A evolução do vetor de distribuição é governada pela matriz de transição em t passos.

Proposição 2: Para qualquer tempo t, tem-se que

$$s^T(t) = s^T(0)P^t, \tag{4.15}$$

onde P^t é calculada e s^T e o vetor transposto de s.

4.6.2.2 Modelo de Verificação de Cadeias de Markov de Tempo Discreto

Pela definição apresentada por Kwiatkowska et al. [34], uma Cadeias de Markov de Tempo Discreto D é uma tupla $(S; \bar{s}; P; L)$, onde:

- S é um conjunto finito de estados, representado pelas portas, *slots* e tarefas;

- $\bar{s} \in S$, é o estado inicial;

- $P : SxS \rightarrow [0, 1]$ é a matriz de probabilidade de transição onde $\sum s' \in SP(s, s') = 1$ para todos $s \in S$;

- $L : S \rightarrow 2^{AP}$ é uma função de marcação que atribui a cada estado $s \in S$ o conjunto $L(s)$ de proposições atômicas que são válidos no estado.

Cada elemento $P(S; s_0)$ da matriz de transição obtém-se a probabilidade de fazer uma transição do estado s para o estado s_0. Note-se que as probabilidades de transições que emanam de um único estado devem somar um. Finalizando estados, ou seja, aqueles a partir do qual o sistema não pode mover-se para outro estado, pode ser modelado pela adição de uma única transição voltar para o mesmo estado, com probabilidade 1.

De acordo com o Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov correspondente ao Modelo Conceitual descrito na seção §4.5, obtém-se o diagrama de transição parcial descrito na Figura §4.6, que descreverá como é o comportamento da matriz de transição completa. A Figura §4.6 mostra um exemplo simples de uma DTMC $D_1 = (S_1; \bar{s}_1; P_1; L_1)$. Na notação gráfica, os estados são desenhados como círculos e as transições como flechas, rotulado com suas probabilidades associadas. O estado inicial está indicado por uma seta de adicional de entrada. O DTMC D_1 tem quatro estados: $\{P_0, S_0, T_0, S_1\}$, com o estado inicial $\bar{s} = S_0$.

A matriz de transição parcial de probabilidade que representa a Figura

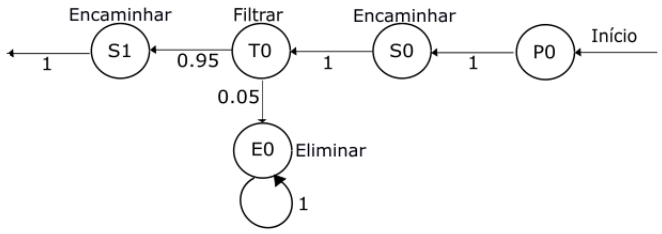


Figura 4.6: *Diagrama de Transição Parcial do Modelo Conceitual*

§4.6 P_1, é dada por:

$$s(t) = \begin{bmatrix} & P_0 & S_0 & T_0 & E_0 & S_1 \\ P_0 & 1 & 0 & 0 & 0 & 0 \\ S_0 & 0 & 1 & 0 & 0 & 0 \\ T_0 & 0 & 0 & 0,95 & 0,05 & 0 \\ E_0 & 0 & 0 & 0 & 1 & 0 \\ S_1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

As proposições atômicas usadas para rotular os estados são tomadas a partir do conjunto AP = porta P_0, *slot* S_0, filtro T_0, eliminadas E_0 e *slot* S_1. Neste modelo da DTMC é um processo simples que envia mensagens em uma solução de integração. Após um passo de tempo, ele entra no estado P_0 a partir do qual, com probabilidade 1 é enviado a mensagem para o outro passo S_0, que segue para um outro passo de tempo, com probabilidade 1 é enviado a mensagem para o passo seguinte T_0, e com probabilidade 0,05 a tarefa filtro T_0 elimina as mensagens gratuitas, e com probabilidade de 0,95 as mensagens seguem no fluxo.

4.7 Experimentos

Os experimentos e os cenários em que as simulações são submetidas encontram-se descritas nessa seção. E com os resultados obtidos através da Ferramenta de Simulação constituem-se os dados para a análise dos resultados experimentais. Para realizar as simulações é utilizado um modelo de

estados e transições onde cada transição ocorre uma determinada probabilidade. A ferramenta utilizada para a realização das simulações é o PRISM *Model Checker*, que é utilizada para fazer a modelagem e verificação formal de modelos probabilísticos.

4.7.1 Cenários submetidos

Para a realização das simulações, são definidos dez cenários, com entrada de mensagem no sistema de simulação de 10.000, 20.000, 30.000, 40.000, 50.000, 60.000, 70.000, 80.000, 90.000 e 100.000. Visando aumentar a eficiência dos resultados obtidos, cada cenário foi submetido a 25 repetições. Afinal, em estatística, um experimento é repetido um grande número de vezes com os mesmos dados, seguindo a Lei dos Grandes Números Grinstead e Snell [26]. Empiricamente, para os experimentos de simulação que é executado, a média populacional costuma ser obtida com, aproximadamente, 25 repetições. O número de repetições é devido o modelo de simulação ser um modelo probabilístico. Após efetuar as 25 simulações analisou-se a média aritmética, variância e desvio padrão por meio dos seguintes métodos Triola et al. [57]:

- A **Média Aritmética** é a medida de posição mais utilizada. É obtida dividindo-se a soma das observações pelo número delas. Tem como facilidade o seu cálculo e como desvantagem de ser muito afetada por valores extremos (valores *oultiers*). Tem a seguinte forma:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4.17}$$

  Em que:

  X_i = valores da variável X

  n = número de dados

- **Análise de Variância:** visa, fundamentalmente, verificar se existe uma diferença significativa entre as médias e se os fatores exercem influência em alguma variável dependente. A análise de variância é utilizada quando se quer decidir se as diferenças amostrais observadas são reais (causadas por diferenças significativas nas populações observadas) ou casuais (decorrentes da mera variabilidade amostral). Portanto, essa análise parte do pressuposto que o acaso só produz pequenos desvios, sendo as grandes diferenças geradas por causas reais. A análise de

variância é a média dos desvios quadráticos de cada valor em relação à média. A variância amostral é dada por:

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1} \tag{4.18}$$

ou

$$S^2 = \frac{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}{n-1} \tag{4.19}$$

- **Desvio padrão:** mostra o quanto de variação ou "dispersão"existe em relação à média (ou valor esperado). Um baixo desvio padrão indica que os dados tendem a estar próximos da média; um desvio padrão alto indica que os dados estão espalhados por uma gama de valores. O desvio padrão é a raiz quadrada da variância.

$$S = \sqrt{S^2} \tag{4.20}$$

4.7.2 Ferramenta de Simulação

Esta seção apresenta uma visão geral da funcionalidade da ferramenta PRISM em modelos estocásticos de simulação. A ferramenta PRISM[35] oferece suporte para vários tipos de modelos probabilísticos, tais como: Cadeias de Markov de Tempo Discreto (DTMC), Cadeias de Markov de Tempo Contínuo (CTMC), Processos de decisão de Markov (PDM), *Probabilistic Timed Automata* (PTA). Para tanto, nesse trabalho, o estudo baseia-se na simulação de Cadeias de Markov de Tempo Discreto que modela um sistema como um conjunto discreto de estados e, transições entre estados, que ocorrem em intervalos discretos de tempo. A escolha pela ferramenta de simulação PRISM se dá, por ser um simulador de eventos discretos, com suporte para a verificação de modelos.

A ferramenta PRISM é um verificador de modelos probabilísticos, uma ferramenta utilizada para fazer analises formais para a modelagem e análise de sistemas que exibem comportamento probabilístico. A ferramenta baseia-se na construção de um modelo matemático preciso de um sistema que está a ser analisado. Aceita modelos probabilísticos descritos na sua tecnologia de modelagem, uma tecnologia baseada em estados simples, de alto nível. A ferramenta PRISM primeiro analisa a descrição do modelo e constrói uma representação interna do modelo probabilístico, em seguida, o

programa calcula o espaço de estados alcançáveis do modelo e descartando quaisquer estados inacessíveis. Isso representa o conjunto de todas as configurações viáveis que podem surgir no sistema modelado. Por fim, o programa analisa a especificação descrita pelos algoritmos por meio de um modelo de indução sobre a sintaxe.

Os componentes fundamentais da linguagem PRISM são módulos variáveis. Um modelo é composto de um número de módulos que podem interagir uns com outros módulos. Um módulo contém um número de variáveis. Os valores destas variáveis, em determinado momento, constituem o estado do módulo. O estado global de todo o modelo é determinada pelo estado local de todos os módulos. O comportamento de cada módulo é descrito por um conjunto de comandos.

A Cadeia de Markov é sintetizada a partir da composição paralela de todos os módulos [3]. Cada estado dessa cadeia é determinado por um estado de valoração do conjunto de variáveis globais e de módulo do modelo. Cada módulo é composto por uma série de comandos. Cada comando define um estado e as transições que partem desse estado. A Figura §4.7 apresenta um exemplo do comando PRISM:

```
[ação] <condição> -> <expressão> : <atualização de variáveis>;
```

Figura 4.7: *Comando PRISM*

Cada expressão pode envolver várias constantes reais resultando em número real p tal que $0 \leq p \leq 1$, que representa a probabilidade do modelo fazer aquela transição. Cada atualização de varáveis representa uma transição de estado modelo como um todo. Um comando pode apresentar diversos pares de <expressão>: <atualização de variáveis>, separados pelo operador + representando várias transições a partir do estado representado pelo mesmo.

Neste trabalho é utilizado um modelo de Cadeias de Markov de Tempo Discreto, e utiliza-se a ferramenta PRISM para verificar o modelo. A ferramenta PRISM suporta diferentes tipos de modelos probabilísticos, em particular o modelo de Cadeias de Markov de Tempo Discreto que permite que as transições do modelo sejam feitas por meio de escolhas probabilísticas, e podermos identificar os diferentes estados de execução dos componentes de maneira discreta.

Para detalhar o relacionamento entre os modelos PRISM e a síntese da Cadeia de Markov, são introduzidas características importantes da síntese da linguagem PRISM, [3] [27], [35], apud Nunes [42]:

i. Cada transição no módulo PRISM tem uma, e somente uma, transição correspondente na cadeia de Markov sintetizada. Essa relação não é inversa.

ii. Linguagem PRISM introduz a abstração de módulos PRISM na cadeia de Markov. Esses módulos são executados em paralelo.

iii. Módulos PRISM podem sincronizar transições usando rótulos de ação. Quando um módulo alcança uma transição rotulada com o mesmo rótulo de ação de outra transição em outro módulo, o primeiro módulo fica bloqueado esperando pelos outros módulos em execução alcançarem a transição deste rótulo. Neste caso, esses módulos são sincronizados nesse rótulo.

iv. Cada transição de módulos diferentes rotulados com a mesma ação são sincronizados em uma transição única na cadeia de Markov com o objetivo de representar a transição de sincronismo.

v. Cada módulo controla internamente os seus estados através da atribuição de valores em um ou mais variáveis de estado. Na cadeia de Markov sincronizada, cada variável de estado tem um estado global do modelo determinado pelos valores de todas as variáveis de todos os modelos. Então, se uma única variável de estado de um único modelo modifica seu valor, então isso representa um novo estado da cadeia de Markov.

A Figura §4.8 mostra uma captura de tela da interface do PRISM, referente ao modelo de verificação, ilustrando os resultados quando submetidos e um cenário. Esses valores podem ser representados num gráfico. A ferramenta também possui um editor de texto embutido para a linguagem de PRISM. A tabela resultante de valores pode ser visto diretamente, exportados para uso em uma aplicação externa, tais como uma folha de cálculo, ou representados como um gráfico.

4.7.3 Modelo de Simulação na Ferramenta PRISM

Essa seção apresenta o modelo de simulação implementado na ferramenta PRISM. A partir do modelo conceitual, projetado na tecnologia Guaraná,

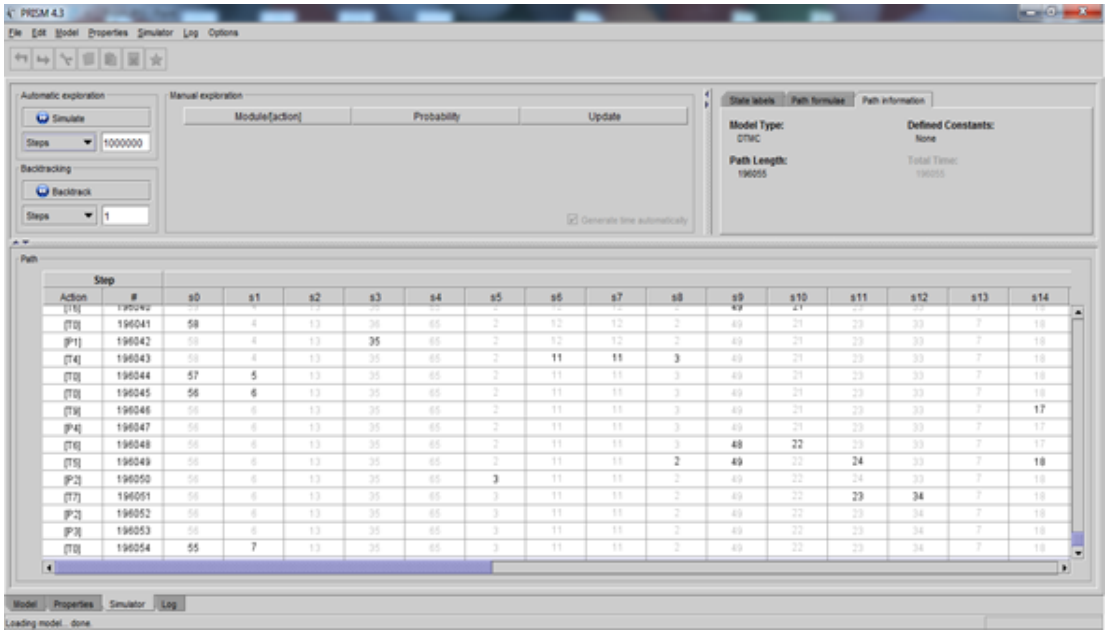


Figura 4.8: *Uma captura de tela da interface da ferramenta PRISM*

efetuou-se a transcrição para o diagrama de transição entre estados da matriz de uma Cadeia de Markov correspondente ao Modelo Conceitual, e posteriormente à construção do modelo de simulação na ferramenta PRISM.

Inicialmente, define-se qual é o modelo matemático que irá ser desenvolvido na ferramenta, linha §1 , em seguida, como esta sendo modelado uma quantidade de mensagens para serem processadas pelo modelo de simulação, atribui-se um valor para o tamanho da fila, linha §2. Na linha §3 é implementado a quantidade de mensagens que entram na solução de integração, ou seja, a partir do momento em de 10.000 mensagens entrarem no sistema, ele automaticamente, irá parar de executar, para que possa ser feita a análise de quantidade de mensagens acumuladas em cada *slot*. Os valores que entram no sistema são modificados de acordo com o cenário que se deseja estudar.

Atribui-se um nome para a solução estudada, linha §4. Os slots que se encontram na solução de integração, projetada na tecnologia Guaraná são declarados nas linhas §5 à §21. Nas linhas §22 à §25 são declaradas as portas (a de solicitação, e as de saídas), pois quando a simulação estagnar é possível haver mensagens aguardando nas portas para serem processadas. A aplicação de recursos humanos está declarada na linha §26. Nessa aplicação poderá

conter mensagens aguardando para serem processadas, e processando, devido à solicitação de informações. Para verificar se realmente a quantidade de mensagens desejadas entraram no sistema, atribui-se um contador, linha §27.

Para a análise dos resultados, e a comprovação de que o modelo de simulação esteja implementado corretamente, são declaradas todas as portas contidas na solução de integração, linhas §28 à §32, a fim de saber a quantidade exata de mensagens que saíram do sistema por cada porta. Assim, também é implementado um contador para cada filtro, linhas §33 à §35, primeiramente para saber a quantidade exata de mensagens que foram processadas pela tarefa filtro, e quantas dessas mensagens foram filtradas, linhas §36 à §41. A quantidade de mensagens filtradas correspondem a 5%, 5% e 1% respectivamente. As mensagens filtradas correspondem as mensagem gratuitas, ou seja, que não geram cobrança na folha de pagamento do funcionário.

Um modelo na linguagem PRISM é composto de um número de determinado de módulos que interagem uns com os outros, as linhas §42 à §62, é onde ocorre o processamento do sistema. Um módulo contém um número de variáveis já declaradas. Os valores destas variáveis, em determinado momento constituem o estado do módulo. O estado de todo o modelo é determinada pelo estado de todos os módulos. O comportamento de cada módulo é descrita por um conjunto de comandos. No momento em que todos os módulos são descritos, finaliza-se o modelo de simulação, linha §63.

```
1  dtmc
2  const int s_max = 1000000;
3  const int m_max = 10000;
4  module unijui_solution
5  s0 : [0..s_max] init 0;
6  s1 : [0..s_max] init 0;
7  s2 : [0..s_max] init 0;
8  s3 : [0..s_max] init 0;
9  s4 : [0..s_max] init 0;
10 s5 : [0..s_max] init 0;
11 s6 : [0..s_max] init 0;
12 s7 : [0..s_max] init 0;
13 s8 : [0..s_max] init 0;
14 s9 : [0..s_max] init 0;
15 s10 : [0..s_max] init 0;
16 s11 : [0..s_max] init 0;
17 s12 : [0..s_max] init 0;
18 s13 : [0..s_max] init 0;
19 s14 : [0..s_max] init 0;
20 s15 : [0..s_max] init 0;
```

```
21 s16 : [0..s_max] init 0;
22 sp1 : [0..s_max] init 0;
23 sp2 : [0..s_max] init 0;
24 sp3 : [0..s_max] init 0;
25 sp4 : [0..s_max] init 0;
26 srh : [0..s_max] init 0;
27 m : [0..m_max];
28 p0 : [0..m_max] init 0;
29 p1 : [0..m_max] init 0;
30 p2 : [0..m_max] init 0;
31 p3 : [0..m_max] init 0;
32 p4 : [0..m_max] init 0;
33 f1 : [0..m_max] init 0;
34 f2 : [0..m_max] init 0;
35 f3 : [0..m_max] init 0;
36 x1 : [0..s_max];
37 x2 : [0..s_max];
38 x3 : [0..s_max];
39 e0 : [0..s_max] init 0;
40 e1 : [0..s_max] init 0;
41 e2 : [0..s_max] init 0;
42 [P0]    true & m < m_max -> (s0'=min(s0+1,s_max))&(p0'=p0+1)&(m'=m+1)
       ;
43 [T0]    true & m < m_max & s0>0 -> 0.05 :(x1'=x1+1)&(s0'=s0-1)&(e0'=
       min(e0+1,s_max)) + 0.95 :(f1'=f1+1)&(s0'=s0-1)&(s1'=min(s1+1,
       s_max));
44 [T1]    true & m < m_max & s1>0 -> (s1'=s1-1)&(s2'=min(s2+1,s_max))&
       (s4'=min(s4+1,s_max));
45 [T2]    true & m < m_max & s2>0 -> (s2'=s2-1)&(s3'=min(s3+1,s_max));
46 [P1]    true & m < m_max & s3>0 -> (s3'=s3-1)&(sp1'=min(sp1+1,s_max))
       ;
47 [RH]    true & m < m_max & sp1>0 -> (sp1'=sp1-1)&(srh'=min(srh+1,
       s_max)) &(p1'=p1+1);
48 [P2]    true & m < m_max & srh>0 -> (srh'=srh-1)&(s5'=min(s5+1,s_max)
       );
49 [T3]    true & m < m_max & s5>0 & s4>0 -> (s5'=s5-1)&(s4'=s4-1)& (s6
       '=min(s6+1,s_max))&(s7'=min(s7+1,s_max));
50 [T4]    true & m < m_max & s6>0 & s7>0 -> (s6'=s6-1)&(s7'=s7-1)& (s8
       '=min(s8+1,s_max));
51 [T5]    true & m < m_max & s8>0 -> (s8'=s8-1) &(s9'=min(s9+1,s_max))
       &(s11'=min(s11+1,s_max))&(s14'=min(s14+1,s_max));
52 [T6]    true & m < m_max & s9>0 -> (s9'=s9-1)&(s10'=min(s10+1,s_max))
       ;
53 [P2]    true & m < m_max & s10>0 -> (s10'=s10-1)&(sp2'=min(sp2+1,
       s_max));
54 [P2]    true & m < m_max & sp2>0 -> (sp2'=sp2-1)&(p2'=p2+1);
55 [T7]    true & m < m_max & s11>0 -> 0.05 : (x2'=x2+1)&(s11'=s11-1) &(
       e1'=min(e1+1,s_max))+ 0.95:(f2'=f2+1)&(s11'=s11-1)&(s12'=min(s12
       +1,s_max));
```

```
56 [T8]    true & m < m_max & s12>0 -> (s12'=s12-1)&(s13'=min(s13+1,
       s_max));
57 [P3]    true & m < m_max & s13>0 -> (s13'=s13-1)&(sp3'=min(sp3+1,
       s_max));
58 [P3]    true & m < m_max & sp3>0 -> (sp3'=sp3-1)&(p3'=p3+1);
59 [T9]    true & m < m_max & s14>0 -> 0.01 : (x3'=x3+1)&(s14'=s14-1)& (
       e2'=min(e2+1,s_max))+ 0.99:(f3'=f3+1)&(s14'=s14-1)&(s15'=min(s15
       +1,s_max));
60 [T10]   true & m < m_max & s15>0 -> (s15'=s15-1)&(s16'=min(s16+1,
       s_max));
61 [P4]    true & m < m_max & s16>0 -> (s16'=s16-1)&(sp4'=min(sp4+1,
       s_max));
62 [P4]    true & m < m_max & sp4>0 -> (sp4'=sp4-1)&(p4'=p4+1);
63 Endmodule
```

4.7.4 Análise dos resultados experimentais

Nesta Seção é apresentado os resultados de simulação. O modelo de simulação equivale ao modelo da solução de integração projetado na tecnologia Guaraná. Para a simulação, o modelo conceitual foi implementado na ferramenta PRISM. O objetivo da simulação é avaliar o comportamento e prever possíveis problemas de desempenho que a solução possa apresentar. Os resultados são obtidos através das simulações em diferentes cenários. Os experimentos de simulação que são executados, são obtidos com 25 repetições, [26]. O número de repetições é devido o modelo de simulação ser um modelo probabilístico

Após a realização das simulações de todos os cenários é apresentado a Tabela §4.3 com um conjunto de resultados das médias obtidos com a simulação na ferramenta PRISM para este estudo de caso.

Slots	10 Mil	20 Mil	30 Mil	40 Mil	50 Mil	60 Mil	70 Mil	80 Mil	90 Mil	100 Mil
S_0	65	73,80	104,50	147,15	133,28	135,45	221,92	215,56	252,12	374,72
S_1	18,72	13,28	18,96	16,88	20,88	21,66	21,24	15,32	11,08	14,28
S_2	16,88	29,76	15,44	14,16	11	18,54	18,76	16,56	14	17,76
S_3	12,04	20,84	14,28	19,76	13,4	20,87	17,36	21,48	13,4	14,92
S_4	81,48	103,52	97	84,92	89,60	115,48	96,48	91,08	76,60	89,04
S_5	16,48	15,68	22,2	16,56	22,08	27,45	18,48	15,8	15,48	16,96
S_6	12,56	19,72	20,36	14,32	12,48	20,25	14,36	10,16	19,64	10,96
S_7	12,56	19,72	20,36	14,32	12,48	20,25	14,36	10,16	19,64	10,96
S_8	20,08	17,24	21,08	11,28	16,8	17,75	23,64	17,12	19,4	18,12
S_9	12,84	15,2	15,8	18,28	15,32	20,91	18,8	11,84	12,96	25,84
S_{10}	20,64	13,4	19,68	19,96	14,08	17,25	16,44	15,96	33,08	11,16
S_{11}	22,32	14,12	26,2	22	21,4	26,70	21,04	19,36	23,28	13,76
S_{12}	10	8,6	6,64	7,88	5,48	7,54	8,36	8,6	8,72	9,96
S_{13}	6,56	10,64	6,6	6,72	9,4	8,45	10,12	4,8	3,92	12,04
S_{14}	9,88	6,72	18,16	13,48	12,8	16,66	12,16	8,72	11,32	7,96
S_{15}	21,68	17,2	12,48	13,8	12,2	16,75	12,6	10,64	20,92	17,36
S_{16}	11,6	14,44	11,8	17,6	20,56	16,12	23,8	19,4	24,8	14,08

Tabela 4.3: *Média de Mensagens em cada slot de cada cenário*

São presentados os gráficos de cada cenário de simulação no anexo §??. Para obter um comparativo dos *slots* com todos os cenários de simulação, opta-se em organizar o Gráfico §4.9 com todos os cenários.

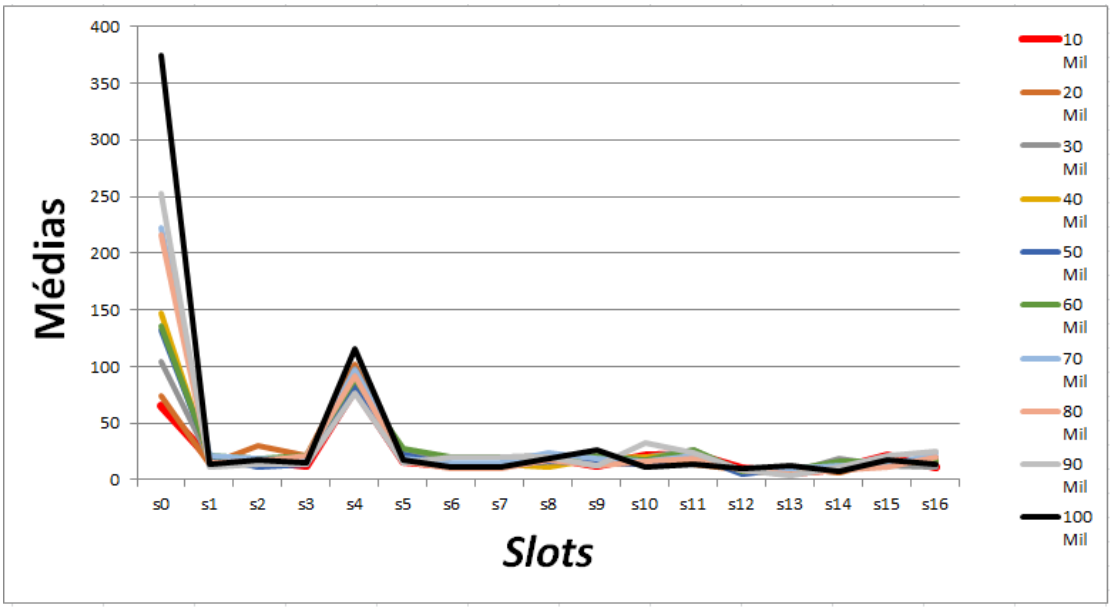


Figura 4.9: *Média de Mensagens Acumuladas em cada Slot de cada cenário.*

Analisando o Gráfico §4.9, pode-se verificar que os gargalos de desempenho do sistema de simulação encontram-se no *slot* s0 e s4, nos demais *slots* há poucas mensagens acumuladas. O Gráfico §4.10 apresenta a comparação de acúmulos de mensagens, nos cenários de 10.000 e 100.000 mensagens. De acordo com os resultados pode-se afirmar que o acúmulo de mensagens no S_0 é resultante da demanda excessiva de mensagens que entram no sistema de simulação. Ou seja, quanto mais mensagens entrar no sistema, maior será o acúmulo no *slot* S_0. Isso também ocorre devido a ferramenta de simulação não trabalhar em relação ao tempo, e sim, em passos de acordo com a Cadeia de Markov de Tempo Discreto. E, quanto mais mensagens entrarem no sistema, haverá também um acúmulo no *slot* S_4. Entretanto esse acúmulo não é na mesma proporção que o *slot* S_0. Isso ocorre devido as mensagens que estão acumuladas no S_0, esperando para serem processadas ocasionando uma quantidade menor de mensagens circulando pelo sistema.

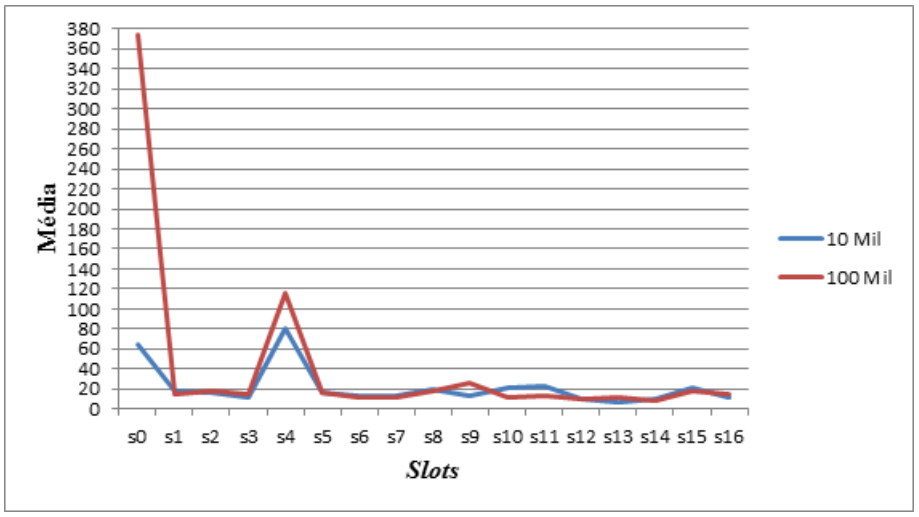


Figura 4.10: *Média de Mensagens Acumuladas no cenário de 10.000 e 100.000 mensagens.*

Todo processo de simulação de eventos discretos está sujeito a obter valores não usuais conhecidos como *outliers*. Para identificar se há discrepância de valores é necessário que os *outliers* sejam calculados para a análise, bem como destacar em cada gráfico os valores que estão fora dos limites superiores e inferiores do desvio padrão. Os *outliers* são calculados com o auxilio do Excel, e analisados através do gráfico *Box-Plot*. Observa-se na Figura §4.11 que os gráficos *Box-Plot* são baseados no valor mínimo, primeiro quartil (Q1), mediana (segundo quartil Q2), terceiro quartil (Q3) e valor máximo. São necessários os resultados dos *slots* de cada cenário para organizar

um gráfico *Box-Plot*.

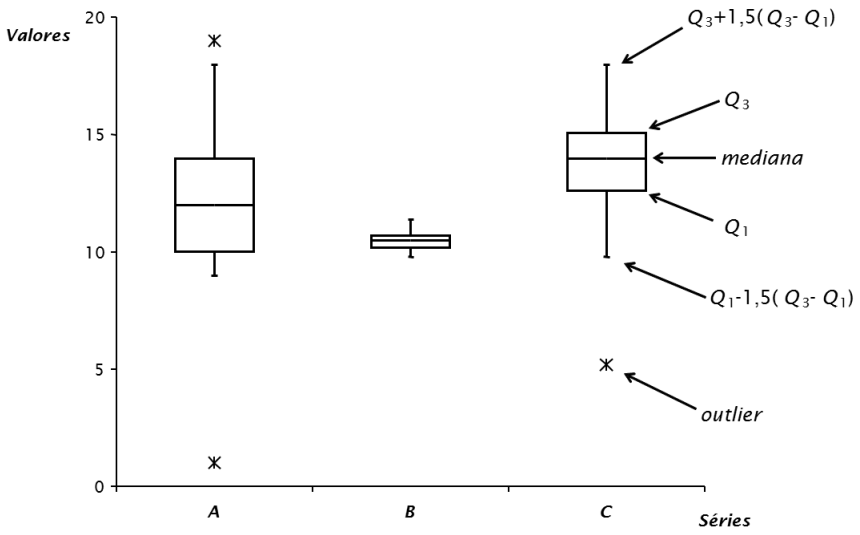


Figura 4.11: *Diagrama Box-Plot para identificação de Outliers.*

Considerando-se uma simulação em um sistema de filas de um modelo M/M/1, a ocorrência de um *outliers*, sobre a variável número médio de clientes na fila é devido a quantidade de clientes e o tempo de atendimento. Observa-se que se o atendimento de um cliente demorar mais que a média, a quantidade de clientes esperando na fila aumenta. Esse cliente passa a ser considerado um *outliers*. Por se tratar de uma simulação de eventos discretos, considera-se que a ocorrência de *outliers* possa ser influenciado pela taxa média de mensagens que chegam ao sistema de simulação, ou uma determinada tarefa possa vir a consumir tempo específico de processamento, devido ao tamanho de uma mensagem no sistema.

De acordo com a Figura §4.12(a) constata-se que não há *outliers* no *slot* S_0, portanto todos os valores das simulações são aceitáveis. No *slot* S_4 há três (12%) *outliers* superiores. Na Figura§4.12(b) podemos observar que no *slot* S_0 há quatro(16%) *outliers* superiores, e no *slot* S_4 há três *outliers* inferiores e cinco superiores. Constata-se que no cenário de 20.000 mensagens há 32% dos valores obtidos com a simulação no *slot* S_4 são *outliers*. A Figura§4.12(c) há dois *outliers* interiores no *slot* S_0 e, no *slot* S_4 há três *outliers* inferiores e dois superiores, que correspondem a 20% dos valores da simulação. Na Figura§4.12(d) todos os valores das simulações no *slot* S_0 são aceitáveis, e há quatro *outliers* inferiores e dois superiores no *slot* S_4, que correspondem a 28% dos valores das simulações. Pode-se observar que na Figura§4.12(e) há apenas *outliers* no *slot* S_0, dois inferiores e três superiores.

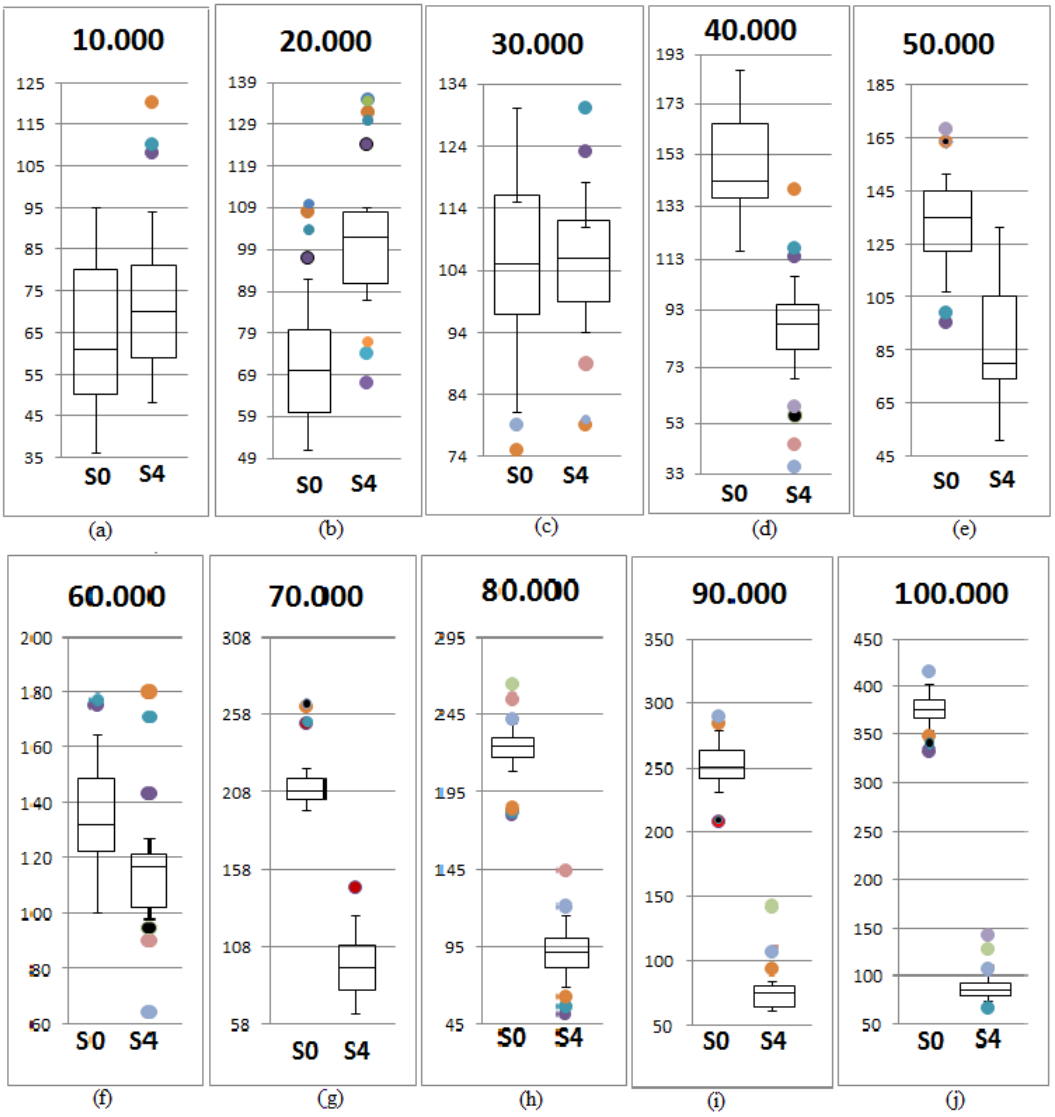


Figura 4.12: *Outliers referente aos slots* S_0 *e* S_4 *em cada cenário .*

Na Figura§4.12(f) e na Figura§4.12(g), os *outliers* (dois e quatro, respectivamente) superiores no *slot* S_0 encontram-se próximos. No *slot* S_4, há três *outliers* inferiores e superiores e, há um *outliers* superiores, respectivamente. Nas Figura§4.12(h) e Figura§4.12(i) os *outliers* (três em ambos) inferiores no *slot* S_0 encontram-se próximos; e, três e dois *outliers* superiores, respectivamente. No *slot* S_4 da Figura§4.12(h) três *outliers* inferiores

muito próximos, e dois superiores. Na Figura§4.12(j) há no *slot* S_0 dois *outliers* inferiores, e três *outliers* superiores muito próximos. No *slot* S_4 há um *outliers* interior e três superiores.

De acordo com as simulações, pode-se observar que há acúmulos consideráveis de mensagens nos *slots* S_0 e S_4 em todos os cenários. A seguir, será analisado a média, a variação dos valores e desvio padrão dos *slots* S_0 e S_4 em cada cenário:

- Cenário com entrada de 10.000 mensagens no sistema: No Gráfico §A.1(a) pode-se observar o comportamento, do *slot* S_0 no cenário em que entram 10.000 mensagens no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 36 á 95 mensagens. Sendo 65 a média de mensagens acumuladas, e o desvio padrão de 17,18. De acordo com o desvio padrão destacado do gráfico com as barras verticais, percebe-se que 36% dos valores obtidos com as simulações estão fora da margem estipulada pelo desvio padrão. Observa-se que neste caso os valores não estão em torno da média.Mas não são valores discrepantes, pois não há *outliers*. No Gráfico §A.1(b) pode-se observar o comportamento do *slot* S_4. Obtendo a média de mensagens acumuladas de 81,48 e desvio padrão de 16,96, portanto 24% dos valores das simulações estão fora da margem estipulada pelo desvio padrão. Desconsiderando os *outliers* a média reduz para 77,23 e o desvio padrão 12,91.

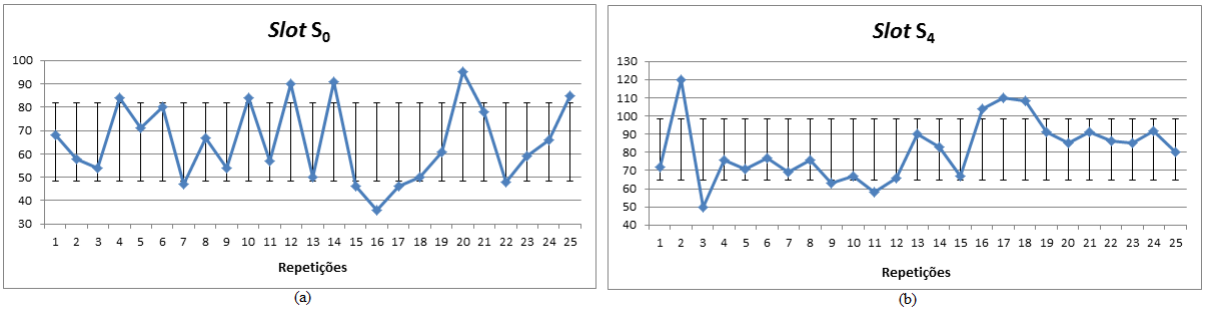


Figura 4.13: *Cenário com 10.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 20.000 mensagens no sistema: No Gráfico §A.2(a). pode-se observar o comportamento, do *slot* S_0 no cenário de

20.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo de mensagens entre 41 á 110 e obtém-se a média 73,8 com desvio padrão de 17,63. Desconsiderando *outliers*, obtemos a média diminui para 67,9 e desvio padrão 11,8.No Gráfico §A.2(b) pode-se observar o comportamento do *slot* S_4. Obtendo a média de mensagens acumuladas de 103,52 e desvio padrão de 19,25. Desconsiderando os *outliers*, a média reduz para 98,69 e o desvio padrão 6,85. Neste caso percebe-se que os *outliers*, tanto inferiores quanto superiores são discrepantes em relação à média, quando eliminados reduziram o desvio padrão drasticamente

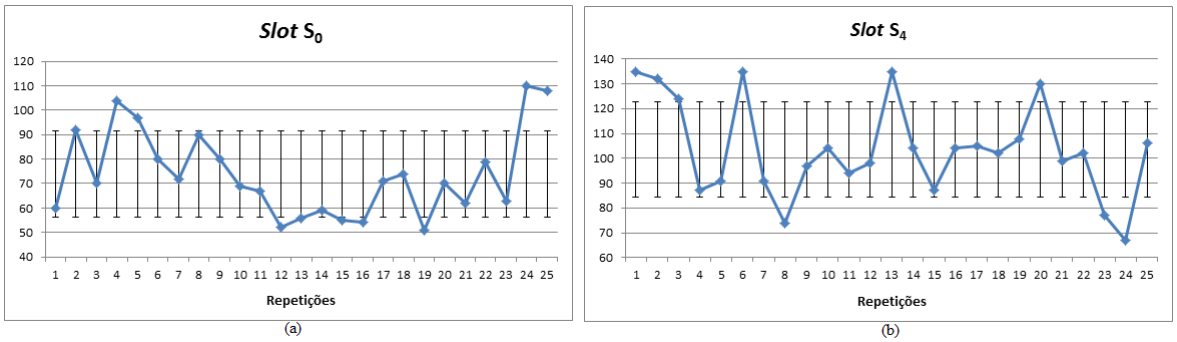


Figura 4.14: *Cenário com 20.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 30.000 mensagens no sistema: No Gráfico §A.3(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 30.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo de mensagens entre 75 á 130 mensagens, e obtém-se de média 104,5, com desvio padrão de 15,12. Desconsiderando *outliers*, a média eleva-se para 106,91, ou seja, os *outliers* são valores inferiores, quando descartados a média eleva-se, e desvio padrão reduz para 13,2. No Gráfico §A.3(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 37 e 177, obtendo como média de mensagens acumuladas de 97, e desvio padrão de 27,54. Desconsiderando os *outliers*, a média aumenta para 106,4 e o desvio padrão reduz para 6,66. Neste caso observa-se a redução significativa do desvio padrão do *slot* S_4 quando os *outliers* são descartados, os demais valores das simulações permanecem entre o limite superior e inferior do desvio padrão.

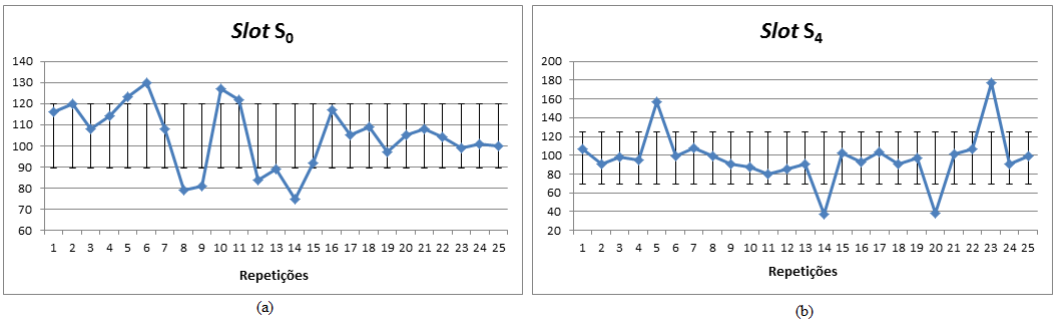


Figura 4.15: *Cenário com 30.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 40.000 mensagens no sistema: No Gráfico §A.4(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 40.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo de mensagens entre 116 á 185 e média 147,15, com desvio padrão de 19,47. Como não há *outliers*, os valores não são considerados discrepantes. No Gráfico §A.4(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 35 e 139 Obtendo a média de mensagens acumuladas de 84,92 e desvio padrão de 22,47. Desconsiderando os *outliers*, a média aumenta significativamente para 106,4. Ou seja, os *outliers* são os valores inferiores e desvio padrão reduz para 11,06.

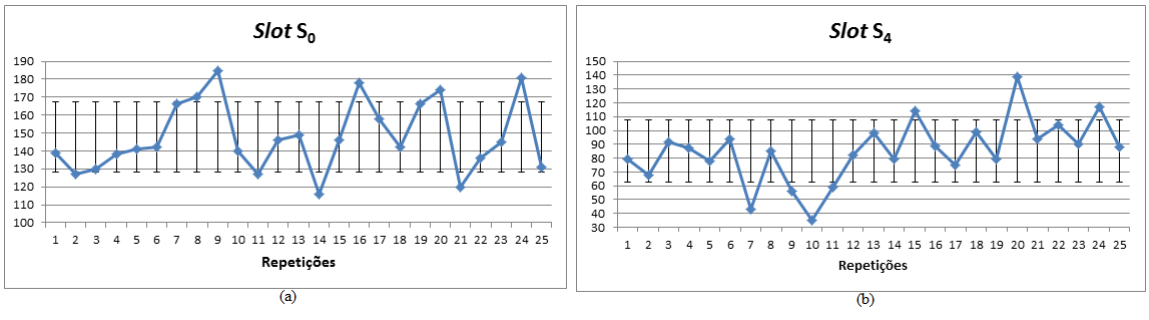


Figura 4.16: *Cenário com 40.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 50.000 mensagens no sistema: No Gráfico §A.5(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 50.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 95 á 166 mensagens e média 133,12 com desvio padrão de 20,39. Desconsiderando *outliers*, obtemos a média de para 131,75 e desvio padrão 14,52. No Gráfico §A.5(b) pode-se observar o comportamento do slot S_4. Os valores oscilam entre 51 e 131 Obtendo a média de mensagens acumuladas de 89,6 e desvio padrão de 22,22. No *slot* S_4 não há *outliers*, ou seja, não valores das simulações discrepantes.

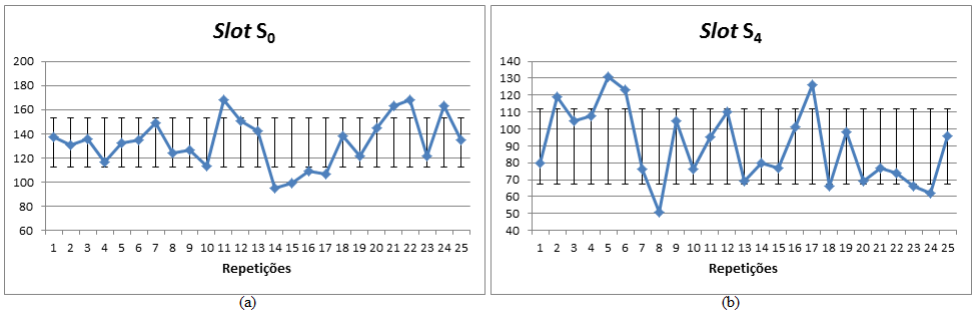


Figura 4.17: *Cenário com 50.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 60.000 mensagens no sistema: No Gráfico§A.6(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 60.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 100 á 177 mensagens e média 135,44 com desvio padrão de 20,04. Desconsiderando *outliers*, obtemos a média de para 131,91 e desvio padrão 17,2, os valores dos *outliers* são os dois maiores resultados das simulações. No Gráfico §A.6(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 64 e 180 Obtendo a média de mensagens acumuladas de 115,48 e desvio padrão de 24,12. Desconsiderando os *outliers*, a média diminiu para 112,84 e o desvio padrão 9,23, o que indica que os valores das simulações descartando os *outliers* então entre o limite superior e inferior indicado pelo desvio padrão inicial.

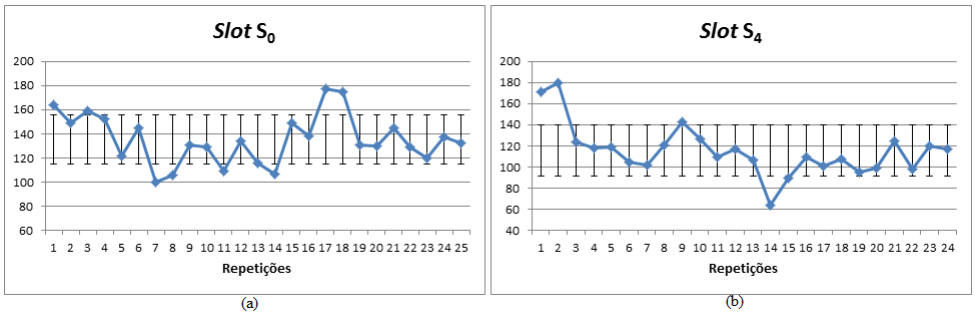


Figura 4.18: *Cenário com 60.000 mensagens no slot S_0 e S_4.*

- Cenário com entrada de 70.000 mensagens no sistema: No Gráfico§A.7(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 70.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 196 á 265 mensagens e média 215,52 com desvio padrão de 20,21. Desconsiderando *outliers*, obtemos a média de 207,43 e desvio padrão 7,29, o que indica que os valores que então no gráfico acima e abaixo do indicados do desvio padrão são considerados *outliers*. No Gráfico §A.7(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 64 e 146 Obtendo a média de mensagens acumuladas de 96,48 e desvio padrão de 21,61. Desconsiderando os *outliers*, a média diminiu para 94,41 e o desvio padrão 19,40, sendo apenas o maior valor descartado.

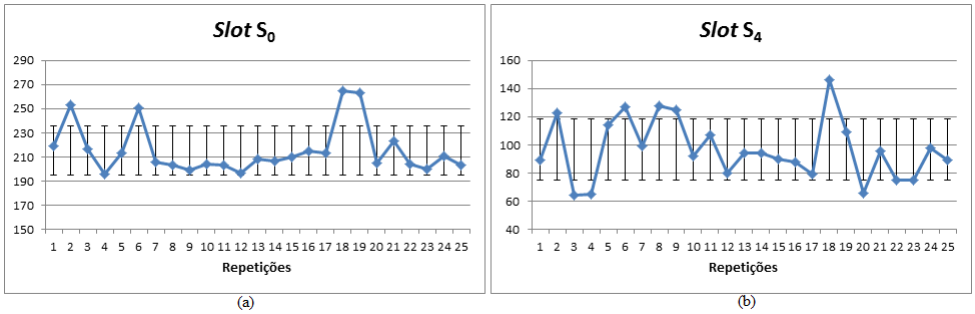


Figura 4.19: *Cenário com 70.000 mensagens no slot S_0 e S_4.*

- Cenário com entrada de 80.000 mensagens no sistema: No Gráfico §A.8(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 80.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 179 á 254 mensagens e média 221,92 com desvio padrão de 19,51. Desconsiderando *outliers*, obtemos a média para 223,48 e desvio padrão 7,65, o que indica que os valores que então no gráfico acima e abaixo do indicados do desvio padrão são considerados *outliers*. No Gráfico§A.8(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 50 e 144. Obtendo a média de mensagens acumuladas de 91,08 e desvio padrão de 20,65. Desconsiderando os *outliers*, a média aumenta para 92,25 e o desvio padrão diminui para 12,15.

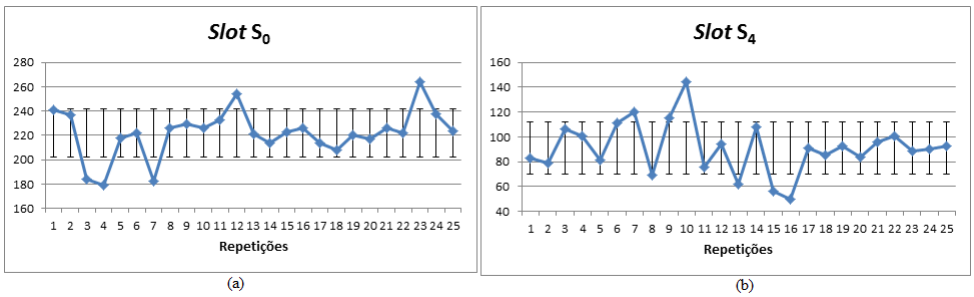


Figura 4.20: *Cenário com 80.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 90.000 mensagens no sistema: No Gráfico§A.9(a) . pode-se observar o comportamento, do *slot* S_0 no cenário de 90.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 208 á 289 mensagens e média 252,12, com desvio padrão de 19,89. Desconsiderando *outliers*, obtemos a média de 253,05 e desvio padrão 12,95. No Gráfico§A.9(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 44 e 142. Obtendo a média de mensagens acumuladas de 76,6 e desvio padrão de 20,59. Desconsiderando os *outliers*, a média diminiu para 69,62 e o desvio padrão 11,17, o que indica que há valores superiores discrepantes.

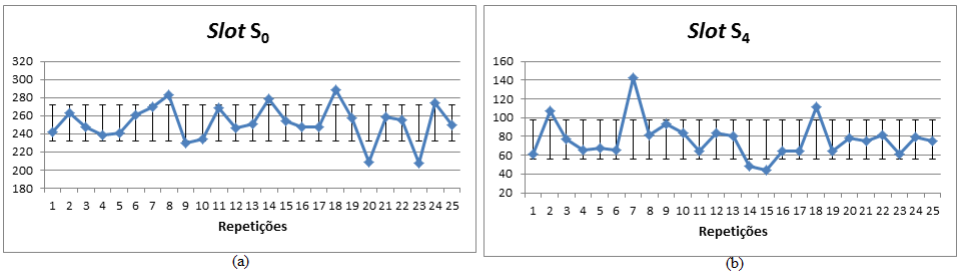


Figura 4.21: *Cenário com 90.000 mensagens no slot* S_0 *e* S_4.

- Cenário com entrada de 100.000 mensagens no sistema: No Gráfico§A.10(a) pode-se observar o comportamento, do *slot* S_0 no cenário de 100.000 mensagem no sistema de simulação. Com as simulações obtém-se uma variação de acúmulo entre 332 á 415 mensagens e média 374,76, com desvio padrão de 19,73. Desconsiderando *outliers*, obtemos a média 377,86 e desvio padrão 13,62. No Gráfico§A.10(b) pode-se observar o comportamento do *slot* S_4. Os valores oscilam entre 47 e 143. Obtendo a média de mensagens acumuladas de 89,04 e desvio padrão de 19,31. Desconsiderando os *outliers*, a média diminiu para 85,53 e o desvio padrão de 8,26. O que indica que os valores que então no gráfico acima e abaixo do indicados do desvio padrão são considerados *outliers*.

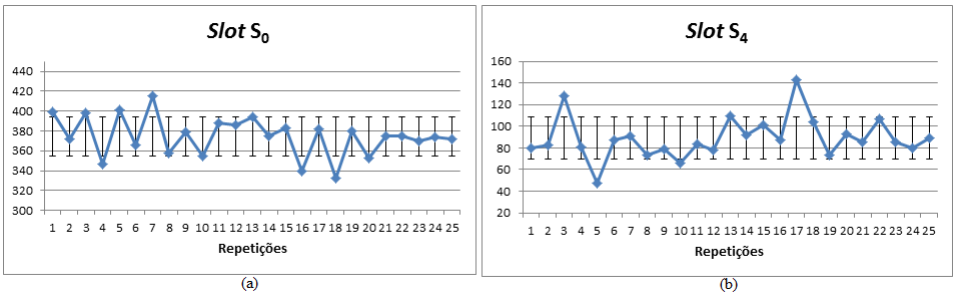


Figura 4.22: *Cenário com 100.000 mensagens no slot* S_0 *e* S_4.

4.8 Resumo do Capítulo

Esse capítulo apresentou a equivalência dos elementos da tecnologia Guaraná com os elementos presentes nas Cadeias de Markov. Essa equivalência proporcionou o estudo detalhado do modelo conceitual. Para obter os resultados, utilizou-se a ferramenta de simulação PRISM. Os resultados obtidos demostraram que há um acúmulo de mensagens no *slot* S_0 e S_4. No *slot* S_0 o acúmulo é resultante das entradas das mensagens no sistema, quanto maior o número de mensagens que entram no sistema modelado, maior é o acúmulo. Para tanto, no *slot* S_4 o acúmulo é resultante da tarefa *correlator* que antecede a esse *slot*. Ou seja, para que uma mensagem possa ser processada pela tarefa *correlator*, tem que ter duas mensagens correlacionadas em seus *slots* de entrada. A descrição da solução de integração em um modelo formal, modelo estocástico de verificação baseado em simulação de eventos discretos, permitiu uma rigorosa exploração, atribuindo a verificação e análise de informações confiáveis.

Capítulo 5

Verificação

A dúvida é o princípio da sabedoria.

Aristóteles, Filósofo Grego (367a.C. - 347a.C)

Este capítulo aborda as técnicas de verificação e validação de modelos de simulação. São discutidas questões relativas a essas técnicas baseadas em modelos de simulação discretos. As abordagens principais das técnicas envolvidas nesse processo procuram garantir que o programa computacional gerado no PRISM, resultante do modelo em Cadeias de Markov, esteja isento de erros. Na Seção §5.1 é apresentada uma introdução sobre as técnicas de verificação e validação. Na Seção §5.2 os conceitos de Verificação e Validação de Modelos de Simulação. Na Seção §5.3 é apresentada algumas Técnicas de Verificação e Validação de Modelos de Simulação. Na Seção §5.4 é apresentada a verificação do modelo proposto. A Seção §5.5 apresenta o resumo do capítulo.

5.1 Introdução

Os modelos são usados para prever ou comparar o desempenho futuro de um novo sistema, um sistema modificado ou um sistema existente sob novas condições. Quando os modelos são utilizados para fins de comparação, geralmente a comparação é feita para um modelo conceitual que representa um sistema existente, a concepção de como um sistema novo ou modificado irá funcionar. Em qualquer desses casos, é necessário que o modelo tenha uma precisão suficiente. Precisão suficiente significa que o modelo pode ser utilizado como um substituto para o sistema real, para efeitos de análise e experimentação Carson et al. [9].

Uma das tarefas mais árduas no modelamento e na simulação de um sistema é determinar se o modelo proposto retrata com fidelidade o sistema investigado. Para tanto, existem duas atividades importantes para legitimar um modelo: a verificação e a validação. O processo de validação e verificação, na literatura, muitas vezes é encontrado como VV e são fundamentais para o estudo de um modelo de simulação.

Considere a versão simplificada do processo de desenvolvimento do modelo na Figura §5.1. O problema é o sistema real ou proposta de estudo; o modelo conceitual é a representação gráfica (simulação) do problema desenvolvido para um estudo particular; e o modelo computacional é o modelo conceitual implementado em uma linguagem para simulação, e inferências sobre o sistema real são obtidas através da realização de experimentos no modelo de simulação em fase de experimentação.

O objetivo de verificação e a validação é obter um modelo que é preciso quando utilizado para prever o desempenho do sistema do mundo real que ele representa, ou para predizer a diferença de desempenho entre os dois cenários ou entre as duas configurações modelo Carson et al. [9].

5.2 Validação e Verificação

Para relacionar a verificação e validação de modelo para esta versão simplificada do processo de desenvolvimento do modelo (ver Figura §5.1), a validação do modelo conceitual é definida como a determinação de que as teorias e pressupostos subjacentes ao modelo conceitual estão corretos e que a representação do modelo do sistema real é "razoável"para a finalidade do

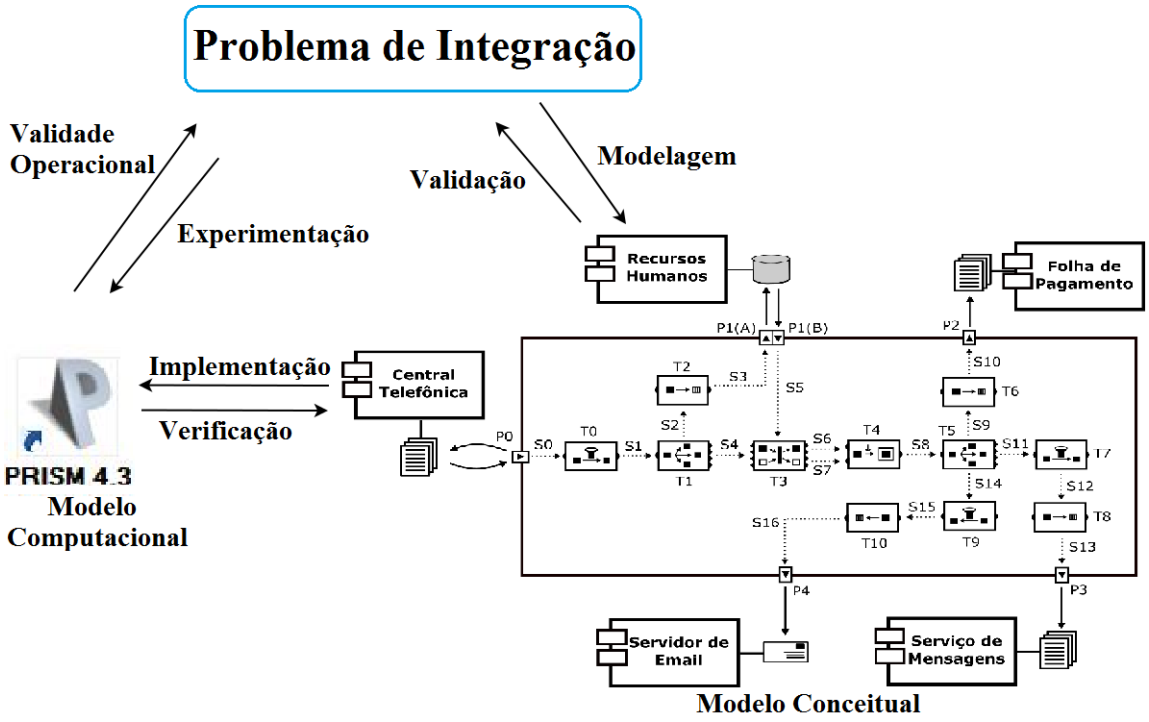


Figura 5.1: *Versão simplificada do processo de desenvolvimento do modelo. Fonte: Adaptado Sargent [55]*

modelo. A verificação do modelo computacional é definida para assegurar que a implementação do modelo conceitual está correta. A validação operacional é definida como a determinação de que o modelo de comportamento de saída tenha uma precisão satisfatória para finalidade de aplicabilidade do modelo pretendido. A validade de dados é definida para assegurar que os dados necessários para a construção do modelo, a avaliação do modelo e testes, e realização de experimentos modelo para resolver o problema são adequados e corretos.

É utilizado um processo interativo para desenvolver um modelo de simulação válido [55]. Um modelo conceitual é desenvolvido seguido de validação do modelo conceitual. Esse processo é repetido até que o modelo conceitual tenha um comportamento satisfatório. Em seguida, o modelo de simulação é desenvolvido, a partir do modelo conceitual, seguido de verificação computacional do modelo. Esse processo é repetido até que o

modelo computacional seja satisfatório. Em seguida, a validade operacional seja conduzida no modelo computacional. Verificação e validação devem ser realizadas novamente quando qualquer mudança de modelo é feita. Este processo é repetido até que um modelo de simulação válido é obtido. Diversas versões de um modelo normalmente são desenvolvidos antes de obter um modelo de simulação válido.

A validação ocorre quando o desenvolvedor do modelo e pessoas conhecedoras do sistema real ou projeto novo/modificado trabalham em conjunto para analisar e avaliar como um modelo funciona. Referindo-se aos processos e técnicas utilizadas em conjunto para assegurar que o modelo representa o sistema real (ou sistema real proposto) a um nível suficiente de precisão [9]. A validação do modelo consiste em comprovar o modelo no seu domínio de aplicabilidade, com o comportamento satisfatório e consistente de precisão com os objetivos do estudo.

No caso da verificação, ela ocorre quando o desenvolvedor modelo exerce um modelo aparentemente correto para a finalidade específica de encontrar e corrigir erros de modelagem. Refere-se aos processos e técnicas que o desenvolvedor modelo usa para assegurar que o seu modelo é correto e corresponde a quaisquer especificações e presunções [9].

A verificação do modelo consiste em comprovar que o modelo conceitual é transformado de uma forma para outra, como se pretende, com suficiente precisão. A Verificação do modelo lida com a construção correta do modelo. A precisão de transformar uma formulação do problema em uma especificação do modelo ou verificado de converter uma representação do modelo em um fluxograma em um código executável é avaliada em verificação do modelo.

Não há como validar e verificar um modelo em 100%, ou garantir que ele seja 100% válido. O que é possível conseguir é aumentar a confiança do modelo, ou acreditar que ele represente satisfatoriamente o sistema, Chwif e Medina [11].

A pesquisa propõe a realização de uma análise criteriosa sobre os resultados da simulação, para fazer a verificação do modelo de simulação. A validação não poderá ser feita por não haver dados reais para comparar com o modelo de simulação. A seguir são apresentadas técnicas de validação e verificação de modelos de simulação.

5.3 Técnicas de Validação e Verificação de Modelos de Simulação

Sargent [55] descreve algumas técnicas e testes comumente utilizados na verificação e validação do modelo de simulação. A maioria das técnicas descritas por ele é encontrada na literatura, embora algumas dessas técnicas possam estar descritas de diversas maneiras. Uma combinação de técnicas geralmente é utilizada. Essas técnicas são usadas para verificar e validar os submodelos e o modelo conceitual. A seguir são descritas algumas técnicas de validação:

- **Animação:** Apresentar graficamente o comportamento operacional do modelo, descrevendo como o modelo se move através do tempo. Geralmente, apenas um intervalo de tempo relativamente curto, pode ser observado, podendo resultar que nem todos os comportamentos vão sendo observado.
- **Comparação com outros Modelos:** Os resultados obtidos do modelo de simulação que pretende-se validar, são comparados com os resultados de modelos válidos. Por exemplo, casos simples de um modelo de simulação são comparados com resultados conhecidos de modelos analíticos, e o modelo de simulação é comparado a outros modelos de simulação já validados.
- **Validade do Evento:** A ocorrência de eventos no modelo de simulação são comparados com eventos do sistema real para determinar se eles são semelhantes.

Outras técnicas destinadas à verificação de modelos de simulação sugeridas por Sargent [55] são similares as que de Freitas Filho [16] descreve a seguir. De acordo com de Freitas Filho [16] essas técnicas baseiam-se em dois conjuntos de elementos:

1. Métodos típicos de correção de programas (*debugging*) presentes na literatura que trata do desenvolvimento de programas de computadores;

2. Procedimentos especialmente relacionados ao desenvolvimento de modelos de simulação. Freitas apresenta alguns métodos para a verificação de modelos de simulação, tais como:

- **Uso de Modelos Determinísticos:** Um problema comum que se apresenta durante a verificação dos modelos de simulação, de acordo com [16], é a variabilidade das respostas obtidas do programa devido ao comportamento das variáveis aleatórias. Tais variáveis, necessárias para representar os processos estocásticos que ocorrem em um sistema do mundo real, podem dificultar o trabalho de verificação. A análise e a busca de erros em programas determinísticos são mais fáceis do que naqueles que contêm valores aleatórios. Uma técnica bastante útil consiste em atribuir valore determinísticos a essas variáveis. Com isso, fica mais simples a previsão de respostas em um sistema e, como consequência, a verificação e busca de erros em vários módulos do programa.

- **Variação sobre os Dados de Entrada:** Outra técnica de análise bastante comum é a variação dos dados de entrada no modelo e verificar se as respostas obtidas são adequadas e consistentes com as variações realizadas. Se os dados de entrada sofrem variações, o esperado é que os resultados sofram variações condizentes. Caso haja um aumento de valores que entram no sistema, o esperado que ocorra um aumento de entidades nas filas, devido a um maior congestionamento. Essa técnica é descrita por Sargent [55] (Validade Interna).

- **Uso de Rotinas de Rastreamento:** As rotinas de rastreamento ou acompanhamento são módulos existentes na maioria das linguagens de simulação, que permitem uma visualização da lista dos eventos ocorridos, ordenados pelo tempo de sua ocorrência os módulos de comando encontram-se ordenados ao longo do tempo. A cada ponto discreto de tempo, um conjunto de módulos é ativado. Ao lado dos eventos, a lista também apresenta as modificações ocorridas nos conteúdos nas inúmeras variáveis e atributos delas dependentes. Geralmente, tais rotinas podem ser ativadas e desativadas pelo usuário, uma vez que sua ativação provoca uma carga adicional ao processamento. Esse tipo de ferramenta é extremamente útil para a verificação de erros, especialmente quanto à lógica da modelagem.

- **Aplicação de Testes da Continuidade:** Esses testes são aplicados sobre várias rodadas de simulação do mesmo modelo. A cada rodada aplica-se uma pequena alteração sobre os parâmetros de entrada. Para cada um dos parâmetros, uma pequena alteração geralmente provoca uma mudança nos resultados. Qualquer mudança mais radical nos resultados deve ser investigada, pois, quase sempre, são provocadas por erros no programa. O emprego de distribuição de probabilidade como a exponencial pode provocar grandes variações sobre os resultados. Assim,

há dúvidas quanto às diferenças observadas, se teriam elas sido causadas pelas distribuições de parâmetros ou pelo uso das distribuições de probabilidade. Nesse caso, o emprego de parâmetros probabilísticos ou de controle sobre uma função geradora de números aleatórios si simulador pode ser a solução. Sargent [55] descreve como a Análise da Variabilidade da Sensibilidade do Parâmetro.

- **Emprego de Testes de Degenerescência ou Robustez:**Esses testes consistem em verificar o funcionamento do modelo quando, sobre os parâmetros, aplicam-se valores extremos (maiores e menores) do que os permitidos. Exemplifica-se esse teste com o modelo de simulação de um roteador de tráfego de pacotes em uma rede de computadores. Uma das características desses equipamentos é a presença de *buffers*, os quais permitem que, enquanto um pacote é processado, um outro que chegar ao roteador aguarde e não seja descartado. Um problema é saber o tamanho ideal do *buffer*. Caso o *buffer* seja grande a probabilidade de descartes de pacote é pequena. Com um buffer muito grande, o pacote poderá permanecer muito tempo no roteador antes de ser processado. Um modelo de simulação é uma excelente ferramenta para testar possíveis soluções a esse problema. Ele permite que mesmo em situações extremas, seja possível observar qual o comportamento do sistema real a que se refere.

- **Execução de Casos Simplificados:** Uma forma especialmente sugerida para a verificação de modelos é a execução de casos simplificados. Um exemplo seria uma rede de computadores, trabalhando com poucos pacotes, um número reduzido de fontes geradoras, de nós intermediários e de nós de destino, facilitando a análise e comparações. No entanto, considere, que a utilização de modelos reduzidos para casos simplificados não garanta que modelos mais complexos funcionam de modo correto. Para maior segurança, embora fazendo o uso de modelos simplificados, garanta que os testes efetuados possuam suficiente complexidade e aderência ao sistema real.

- **Verificação de Consistência:** Os testes de consistência servem para verificar se o modelo produz resultados similares quando aos parâmetros de entrada, são atribuídos valores equivalentes em diferentes execuções. Se os modelos apresentam diferenças significantes nos seus resultados, o programa contém erros na modelagem. De outra forma, devem ser apresentadas justificativas consistentes para explicar as diferenças. Quando empregadas distribuições de probabilidades, as similaridades entre os resultados serão verificadas para experimentos de razoável

duração.

- **Independência de Sementes Geradoras de Números Aleatórios:** Em programas de simulação, os valores aleatórios de entrada, não devem influenciar o comportamento geral dos resultados. Diferenças entre diversas rodadas de simulação são esperadas, mas tais diferenças, considerando-se as médias de diversas amostras (rodadas) devem ser pequenas, caracterizando que estas são devidas aos processos aleatórios envolvidos no sistema real e bem representados no modelo.

- **Rotinas de Verificação:** de Freitas Filho [16] salienta que as rotinas de verificação permitem ao modelador averiguar a ocorrência de determinados tipos de erros, se existirem. Caso as probabilidades de um conjunto de eventos devem somar um, o programa poderá verificar isso emitindo avisos ao usuário no caso da ocorrência de valores diferentes do esperado. Outro exemplo que Freitas destaca, consiste na contabilidade de entidades geradoras e servidas. No caso da rede de computadores, por exemplo, o modelo contaria o número de pacotes enviados pelas fontes, bem como o número de pacotes recebidos nos diversos destinos. A diferença entre o número de pacotes enviados e o número de pacotes recebidos, acrescidos daqueles perdidos nas diversas rotas, deve ser igual à zero. Ou seja, valor de entrada em um sistema, descontando os valores que permanecem no sistema com os valores de saída é igual à zero. Alguma coisa diferente de zero, indica erros de programação ou modelagem. Descrita por Sargent [55] como Teste de Condição Extrema.

5.4 Verificação do Modelo Proposto

A verificação do modelo proposto segue as técnicas de simulação sugeridas por Sargent [55] e de Freitas Filho [16]. Para comprovar a legitimidade do modelo, são analisados uma simulação de cada cenário. Neste caso, de acordo com o modelo conceitual §4.2, os resultados das simulações são descritas em tabelas de acordo com a Tabela §5.1, obtendo os seguintes dados:

Step: Quantidade de passos executados no cenário.

Srh: Sistema da aplicação dos Recursos Humanos

S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}: *Slots*

Sp1: Mensagens que estão executando na porta de solicitação.

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
194886	118	20	9	2	22	10	0	0

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
9	38	10	36	18	7	23	37	10

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
1	11	17	9	0	10000	10000	9345	9267

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
8784	9140	9377	8826	9196	505	464	107

Tabela 5.1: *Dados Obtidos em Cada Simulação*

Sp2: Mensagens que estão executando na porta de saída da Folha de Pagamento.

Sp3: Mensagens que executando na porta de saída para o Serviço de Mensagens.

Sp4: Mensagens que executando na porta de saída para o Servidor de Email.

M: Contador.

P_0: Mensagens que entraram na Porta de Entrada.

P_1: Porta de Solicitação.

P_2: Mensagens que saíram pela porta P_2, e encaminhada para o sistema de Folha de Pagamento.

P_3: Mensagens que saíram pela porta P_3, e encaminhada para o sistema de Serviço de Mensagens.

P_4: Mensagens que saíram pela porta P_4, e encaminhada para o sistema de Servidor de Email.

F_1: Mensagens filtradas pelo filtro da tarefa T_0.

F_2: Mensagens filtradas pelo filtro da tarefa T_7.

F_3: Mensagens filtradas pelo filtro da tarefa T_9.

X_1: Quantidade de mensagens que passaram pelo filtro (T_0).

X_2: Quantidade de mensagens que passaram pelo filtro (T_7).

X_3: Quantidade de mensagens que passaram pelo filtro (T_9).

Em cada cenário foram efetuadas 25 repetições. De acordo com de Freitas Filho [16] as diferenças entre as diversas rodadas de repetições são esperadas. Tais diferenças, considerando as médias dessas repetições devem ser pequenas. Sargent [55] salienta que uma grande quantidade de variabilidade entre as repetições torna questionáveis os resultados do modelo. As repetições são organizadas em tabelas e analisadas, e posteriormente é realizada a média das mensagens acumuladas nas filas em cada *slot*. A análise constatou que não há discrepâncias de resultados.

Outra análise é feita de acordo com de Freitas Filho [16] e Sargent [55].Em cada cenário, analisa-se uma simulação detalhadamente. De acordo com os autores, subtraindo a quantidade de mensagens que entraram na solução de integração, das mensagens que estão executando no sistema e as mensagens que saíram por cada uma das portas o resultado teria de ser igual á zero.

Para comprovar que o modelo é válido, considera-se que o modelo de simulação estudado, apresenta: (1) três portas de saída; (2) tarefa que permite replicar as mensagens; e (3) cada tarefa esteja executando cada vez que tiver filas nos *slots*. Nestas condições para cada porta de saída é estruturada uma forma genérica para verificar cada cenário.

Para cada mensagem que chega a tarefa T_0 (*Replicator*) é criada uma cópia que segue no fluxo do sistema. Uma cópia é encaminhada para o S_2 e outra cópia encaminhada ao S_4. A cópia 1 (C_1) possui a mesma quantidade de mensagens que se encontram na cópia 2 (C_2). Considerando que no *slot* S_6 (cópia oriunda do fluxo que segue do S_2) e S_7 (cópia oriunda do fluxo que segue do S_4) contém a mesma quantidade de mensagens, portanto:

$$\Sigma\ C_1 = S_2 + S_3 + SP1 + SRH + S_5 + S_6 \tag{5.1}$$

$$\Sigma\ C_2 = S_4 + S_7 \tag{5.2}$$

$$\frac{\sum C_1 + \sum C_2}{2} \tag{5.3}$$

- Para a Porta de saída do Sistema de Folha de Pagamento, obtemos:

Considerando que FL_1, seja o caminho percorrido pelas mensagens com destino ao Sistema da Folha de Pagamentos.

$$\sum FL_1 = S_0 + F_1 + S_1 + S_8 + S_9 + S_{10} + SP2$$

$$V_1 = P_0 - \left(\sum FL_1 + \frac{\sum C_1 + \sum C_2}{2} + P_2 \right) \tag{5.4}$$

- Para a porta de saída do Sistema de Mensagens:

Considerando que FL_2, seja o caminho percorrido pelas mensagens com destino ao Sistema de Mensagens.

$$\sum FL_2 = S_0 + F_1 + S_1 + S_8 + S_{11} + F_2 + S_{12} + S_{13} + SP_3$$

$$V_2 = P_0 - \left(\sum FL_2 + \frac{\sum C_1 + \sum C_2}{2} + P_3 \right) \tag{5.5}$$

- Para a porta de saída do Sistema de Email:

Considerando que FL_3, seja o caminho percorrido pelas mensagens com destino ao Sistema de Email.
$\sum FL_3 = S_0 + F_1 + S_1 + S_8 + S_{14} + F_3 + S_{15} + S_{16} + SP_4$

$$V_3 = P_0 - \left(\sum FL_3 + \frac{\sum C_1 + \sum C_2}{2} + P_4 \right) \tag{5.6}$$

Na Tabela §5.2 é apresentado os resultados de uma simulação com 10.000 mensagens no sistema. Verificando as equação no cenário de 10.000 mensagens de entrada, obtemos:

$$\sum C_1 = 9 + 2 + 1 + 10 + 0 + 0$$

$$\sum C_1 = 22$$

$$\sum C_2 = 22 + 0$$

$$\sum C_2 = 22$$

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
194886	118	20	9	2	22	10	0	0

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
9	38	10	36	18	7	23	37	10

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
1	11	17	9	0	10000	10000	9345	9267

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
8784	9140	9377	8826	9196	505	464	107

Tabela 5.2: *Cenário com entrada de 10.000 mensagens no sistema*

- Para a Porta de saída do sistema de folha de pagamento (V1):

$$FL_1 = S_0 + F_1 + S_1 + S_8 + S_9 + S_{10} + SP_2$$

$$\sum F_1 = 118 + 505 + 20 + 9 + 38 + 10 + 11$$

$$\sum F_1 = 711$$

$$V_1 = 10.000 - \left(711 + \frac{22+22}{2} + 9267\right)$$

$$V_1 = 0$$

- Para a porta de saída do sistema de mensagens (V2):

$$\sum F_2 = 118 + 505 + 20 + 9 + 36 + 464 + 18 + 7 + 17$$

$$\sum F_2 = 1194$$

$$V_2 = 10.000 - \left(1194 + \frac{22+22}{2} + 8784\right)$$

$$V_2 = 0$$

- Para a porta de saída do sistema de email (V3):

$$\sum F_3 = 118 + 505 + 20 + 9 + 23 + 107 + 37 + 10 + 9$$

$$\sum F_3 = 838$$

$$V_3 = 10.000 - \left(838 + \frac{22+22}{2} + 9140\right)$$

$$V_3 = 0$$

]

$$V_1 = V_2 = V_3 = 0$$

Após a realização dos testes de verificação em cada cenário de simulação (apresentado no anexo §**??**), constata-se que a implementação do modelo de simulação na ferramenta PRISM está de acordo. Isso porque a validade operacional determinou que o comportamento de saída do modelo de simulação tem a precisão necessária para o fim pretendido do modelo sobre o domínio da aplicabilidade do modelo pretendido.

Quando uma linguagem de simulação é utilizada, a verificação está preocupada em garantir que a linguagem de simulação utilizada esteja isenta de erros, e ter sido adequadamente implementada e o modelo tenha sido corretamente programado Sargent [55].

Sargent [55] destaca que a verificação e a validação de modelos, são fundamentais para o desenvolvimento de um modelo de simulação. Para tanto, não há um conjunto de testes específicos que podem ser aplicados para determinar se um modelo está correto. Além disso, não há um algoritmo para determinar quais as técnicas e/ou procedimentos a serem usados. Todas as técnicas de verificação discutidas na Seção §5.3 podem ser aplicados para verificar um modelo de simulação. Cada projeto, ou modelo de simulação apresenta um único desafio. E a técnica que se deseja usar para a verificação depende por quem esta desenvolvendo o modelo.

5.5 Resumo do Capítulo

Esse capítulo apresentou uma descrição detalhada de algumas técnicas de verificação para modelo de simulação. A validação consiste em confrontar o modelo matemático com os resultados de execução de um sistema real, e a verificação consiste em examinar se o modelo matemático de simulação foi construído de forma correta e equivale ao modelo conceitual. Como não há dados reais do problema proposto, efetuou-se a verificação. Aplicando algumas técnicas no modelo proposto, verificou-se que o modelo de simulação apresentado está em conformidade com o modelo conceitual, conseguiu-se comprovar a legitimidade do modelo de simulação.

Parte IV

Considerações Finais

Capítulo 6

Conclusões e Trabalhos Futuros

"O aprender se concentra em dois pilares:
a própria pessoa, como agente, e a escola,
como lugar de crescimento profissional permanente"

António Nóvoa, Professor universitário português (1954)

As empresas estão sempre a procura de como aperfeiçoar seus processos de negócios. Assim, não é de hoje que a tecnologia de *software* desempenha um papel importante ao fornecer aplicações de software para apoiar as atividades diárias de uma empresa. Ao longo dos anos, uma empresa pode ter adquirido ou desenvolvido internamente várias aplicações, que não são comumente prontas para compartilhar dados e funcionalidade, ou seja, elas foram construídas sem preocupações de integração. A Integração de Aplicação Empresarial é um campo de pesquisa que lida com o desenvolvimento de metodologias e ferramentas para projetar e implementar soluções de integração. O sucesso de tais processos de negócios é altamente dependente da execução correta e eficiente das soluções de integração. A utilização de técnicas matemáticas e simulação podem ser usadas para analisar o comportamento de soluções de integração com vistas a melhorar a sua qualidade antes de sua construção e implantação.

Em sequência, apresentou-se o estudo de caso de uma solução de integração, que é analisado nesta dissertação. Uma solução de integração permite reutilizar aplicações legadas e integrar aplicações heterogêneas, podendo ser caracterizada como sistemas discretos pelo fato de todos os componentes envolvidos em uma solução de integração consumirem um tempo de execução específico, apenas quando ocorre um evento.

Levando-se em consideração o estudo de caso, torna-se imprescindível conhecer os conceitos envolvidos nessa abordagem. Contudo, essa dissertação apresentou um levantamento bibliográfico envolvendo a tecnologia Guaraná, sistemas, modelos estocásticos, dinâmicos, discretos, simulação e Cadeias de Markov. Buscou-se entender o funcionamento da tecnologia Guaraná, a interação que a solução de integração apresenta. Portanto, verificou-se que é possível utilizar modelos computacionais e matemáticos, técnicas de simulação de eventos discretos para analisar o comportamento e identificar gargalos de desempenho que podem surgir em soluções de integração de aplicações, quando submetidas à cenários críticos de funcionamento.

Pelo estudo realizado e apresentado, tornou-se possível perceber que há alguns trabalhos na literatura que utilizaram a simulação de eventos discretos, técnicas e ferramentas para analisar sistemas para prever o comportamento e para encontrar os possíveis gargalos de desempenho. No entanto, percebe-se que não há evidência de simulação de eventos discretos visando a análise de modelos conceituais de soluções de integração de aplicações empresariais, que utilizam técnicas matemáticas e simulações. E, um número reduzido de trabalhos apresenta a ferramenta de simulação PRISM. Foram referenciados trabalhos relacionados que utilizam a técnica matemática de Cadeias de Markov de Tempo Discreto e simulação utilizando a ferramenta PRISM. Notou-se, no entanto, uma incidência maior nos trabalhos relacionados com a técnica matemática Cadeias de Markov de Tempo Discreto em processos estocásticos, utilizando várias técnicas de simulação. No entanto, a simulação na ferramenta PRISM permite a simulação em eventos discretos, apresentando uma linguagem simples, baseado em estados, condizente com a técnica matemática de Cadeias de Markov de Tempo Discreto. Além do que, a ferramenta de simulação demonstra os dados obtidos através do comportamento aleatório ou probabilístico do sistema modelado.

Com a análise do modelo conceitual da solução de integração desenvolvida na tecnologia Guaraná, obtém-se o Diagrama de Transição entre Estados da Matriz de uma Cadeia de Markov. Percebeu-se que há equivalência dos elementos de uma solução de integração projetada na tecnologia Guaraná com os elementos que compõem as Cadeias de Markov. Para demostrar a equivalência, utilizou-se uma ferramenta de simulação para a verificação do modelo, essa ferramenta possui extensões probabilísticas da lógica temporal que visa detecção de gargalos e análise de desempenho.

Outra característica comum entre a solução de integração apresentada pela tecnologia Guaraná e um sistema de eventos discretos é a relação entre os seus elementos e estrutura de funcionamento. Na modelagem de

sistemas de evento discreto, como uma solução de integração, a identificação e relação com as variáveis de desempenho e a forma com que elas interagem entre si e com os outros elementos, auxilia a compreensão do sistema. Contudo, ilustra-se o esquema de funcionamento do processo de decisão de Markov, percebem-se os métodos utilizados no modelo que representa uma solução de integração. A partir do momento em que há uma mensagem na fila, e inicia-se o processo.

Nesse sentido, observou-se que, nas simulações em eventos discretos, as variáveis são equacionadas em um modelo, cujas mudanças de estado ocorrem em pontos discretos de tempo. A escolha da utilização das Cadeias de Markov para este estudo se dá, pois é um caso particular de processos estocásticos e, portanto, utilizado para sistemas de eventos discretos, possibilitando estudar e analisar a formação de filas. Os eventos geram as transições de estados representada por uma matriz de transição. Para a simulação propôs-se a utilização da ferramenta PRISM, que é uma aplicação utilizada para a modelagem formal de sistemas.

Com essa abordagem de simulação, buscou-se descobrir os gargalos de desempenho ainda na fase de projeto reduzindo, consequentemente, o custo, o risco e tempo na implementação. Assim, as soluções de integração são vistas como sistemas de evento discreto e seus modelos conceituais são traduzidos em modelos formais, que são simulados utilizando técnicas e ferramentas conhecidas para a simulação de eventos discretos.

A partir das tecnologias de integração disponíveis para as empresas a projetar e implementar soluções de integração, a tecnologia Guaraná foi usada para ilustrar a caracterização de soluções de integração como sistemas de eventos discretos. Várias técnicas e ferramentas podem ser utilizadas para apoiar a realização de uma simulação de eventos discretos. Explorou-se o uso de simulação em um modelo de Cadeias de Markov de Tempo discreto e, a técnica de verificação baseou-se em analisar o modelo conceitual da solução de integração. Processos de decisão de Markov foram consideradas o tipo mais adequado de modelos probabilísticos para executar a verificação de modelo formal e análise de desempenho. Esses tipos de modelos probabilísticos têm amplo apoio em ferramentas modernas probabilísticas de modelo verificação de software, como PRISM. Assim, também foi fornecido um debate aprofundado sobre este tipo de modelo e software PRISM.

A partir do modelo de simulação, obteve-se um indicativo do comportamento futuro da aplicação, com relação à variabilidade dos parâmetros de entrada do sistema, como a quantidade de mensagens acumuladas em cada

slot, quando submetidos aos diferentes cenários. Esse conhecimento habilita uma tomada de decisão, evitando que as mensagens fiquem acumuladas nos *slots* gerando gargalos de desempenho. É válido ressaltar que, de acordo com a revisão bibliográfica que embasou essa dissertação, não se obtiveram indícios de outra proposta defendida sob uma postura estocástica e focada em integração de aplicações empresariais.

Utiliza-se o modelo de simulação proposto para prever e comparar o desempenho futuro do sistema. Geralmente a comparação é feita para um modelo conceitual, neste estudo, projetado na linguagem Guaraná, que representa o sistema existente, a concepção de como um sistema novo ou modificado irá funcionar. Para tanto, uma das tarefas mais árduas no modelamento e na simulação de um sistema é determinar se o modelo proposto retrata com fidelidade o sistema investigado.

Diante dos resultados obtidos, pode-se dizer que a ferramenta PRISM permitiu construir modelos probabilísticos equivalentes com os modelos de solução de integração desenhados pela ferramenta Guaraná. Percebe-se que a ferramenta é de simples entendimento, auxiliando na modelagem da solução de integração. Integrar aplicações pode trazer grandes benefícios para as empresas. No caso de estudo, apresentado no presente trabalho, a integração das aplicações permite que diversos processos sejam realizados de forma automática, tornando o processo da central telefônica mais simples e rápido, reduzindo os custos e tempo gasto, que incorre devido às atividades relacionadas com a construção e execução.

Demonstra-se com os resultados obtidos, que há um acúmulo de mensagens no *slot* S_0 e S_4. No *slot* S_0 o acúmulo é resultante das entradas das mensagens no sistema, sendo que quanto maior o número de mensagens que entram no sistema modelado, maior é o acúmulo. No entanto, no *slot* S_4, o acúmulo é resultante da tarefa *correlator* que precede a esse *slot*. Neste caso para que a tarefa *correlator* possa processar as mensagens, necessita ter em seus*slot* de entrada, mensagens correlacionadas, oriundas das replicações da tarefa *replicator*. A disponibilidade da integração solução descrita em um modelo formal, modelo estocástico de verificação baseado em simulação de eventos discretos, permitiu uma rigorosa exploração, atribuindo a verificação e análise de informações confiáveis.

Essa dissertação destaca-se por verificar a confiabilidade do modelo de simulação. Constatou-se durante os testes de verificação, que o código na ferramenta de simulação não estava implementado de forma correta, nos *slots* que antecedem a tarefa filtro estavam acumulando uma quantidade muito

maior de mensagens do que o esperado. Após vários testes de simulação e estudos comprovou-se que a forma como a implementação havia sido feita permitia que as mensagens que deveriam ser filtradas retornassem ao *slot* anterior da tarefa filtro e repetissem o processo, gerando um acúmulo exagerado de mensagens. Nesse caso, comprova-se a importância de realizar testes de verificação. Neste caso, atesta-se que quando uma linguagem de simulação é utilizada, a verificação está principalmente preocupada em garantir que a implementação da linguagem de simulação utilizada esteja isenta de erros, e tenha sido adequadamente implementado e o modelo tenha sido corretamente programado.

Após estudos, modificou-se o código, efetuando-se novamente as simulações, repetindo e testando os diferentes cenários propostos. Analisando-se os resultados obtidos, comprova-se a legitimidade do modelo de simulação. Para suprir a precisão de confiabilidade necessária, aplica-se algumas técnicas de verificação no modelo de simulação. Nesse caso, garante-se que linguagem de simulação utilizada esteja isento de erros.

Em suma, conseguiu-se elaborar uma proposta de simulação de eventos discretos, capaz de representar um sistema real de uma solução de integração projetada na tecnologia Guaraná. A comprovação foi obtida através de técnicas de verificação, e do desenvolvimento de uma equação para representa-la a partir do modelo conceitual estudado.

Para trabalhos futuros, recomenda-se estender a aplicação da técnica de simulação computacional na ferramenta PRISM para a técnica matemática Cadeias de Markov de Tempo Contínuo (*Continuous Time Markov Chain* - CTMC). A especificação do comportamento de CTMC é feita de forma semelhante a uma Cadeia de Markov de Tempo Discreto. A principal diferença é que as atualizações em comandos são rotuladas com taxas (de valor positivo), ao invés de probabilidades. A notação utilizada em comandos, para associar as taxas de transições, é idêntica ao utilizado para atribuir probabilidades.

A contribuição na pesquisa utilizando o estudo de caso proposto por essa dissertação e estender para as CTMC, é que auxilia na análise do comportamento das mensagens em relação ao tempo, a fim de detectar em quais os *slots* necessitam de um tempo maior de processamento, evitando os gargalos de desempenho, para que o sistema flua com maior eficiência. Nesse caso, consegue-se observar: (a) taxa média de saída do estado i para o estado j, (b) o tempo gasto em cada estado, antes de qualquer transição ocorrer, a distribuição é com taxa exponencial.

Em uma solução de integração projetada na tecnologia Guaraná, utilizado a ferramenta de simulação PRISM, as mensagens entram no sistema de simu-

lação, contabilizando o tempo gasto em cada *slot* e tarefa, e ao sair do sistema, uma nova mensagem entra. Ou seja, em todo o processamento, haverá apenas uma mensagem no sistema, quando essa sair, outra mensagem que está aguardando na fila ingressa no sistema e faz todo o processo.

Outra linha de atuação que se mostrou interessante, a partir desse trabalho, se relaciona a estudos para a implementação da solução de integração. Com a implementação, obtêm-se os dados reais, e com esses dados é possível realizar a validação do modelo de simulação, e obter um modelo analítico a fim de comparação com o modelo de simulação.

Parte V

Anexos

Apêndice A

Gráficos dos Cenários

Nesta Seção é apresentado os gráficos obtidos dos resultados das repetições das simulações referente a Seção §4.7.4.

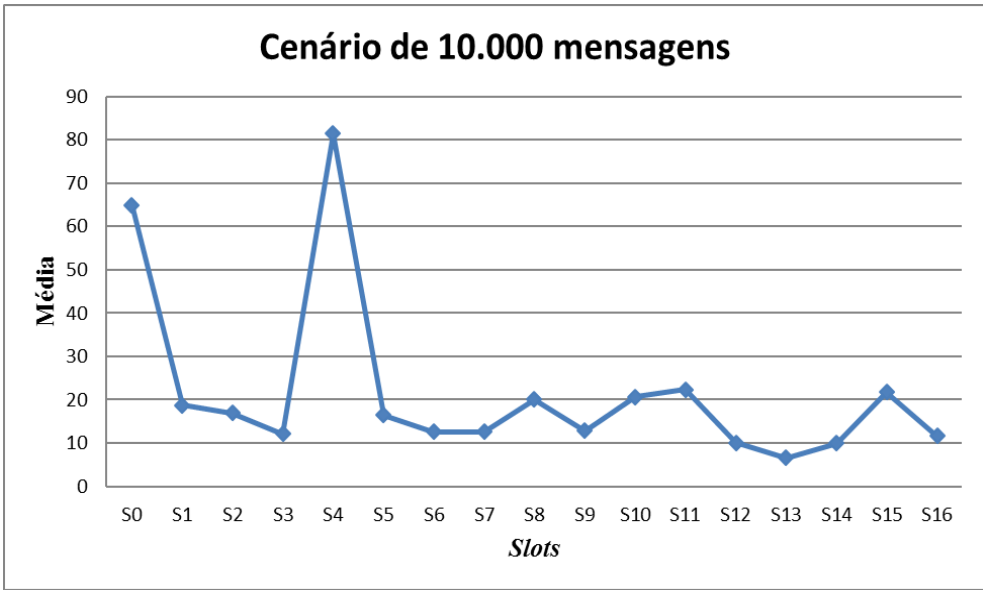


Figura A.1: *Média de Mensagens Acumuladas com entrada de 10.000 mensagens .*

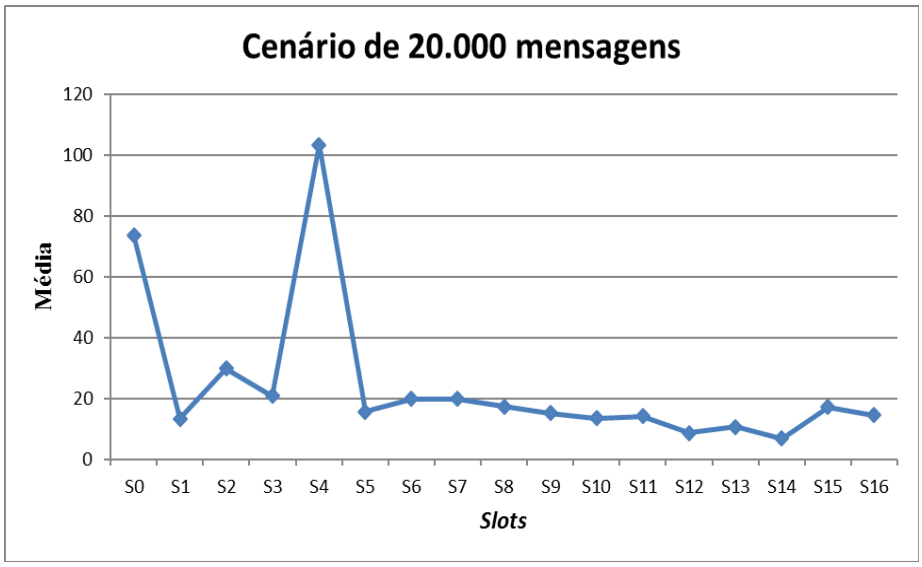


Figura A.2: *Média de Mensagens Acumuladas com entrada de 20.000 mensagens .*

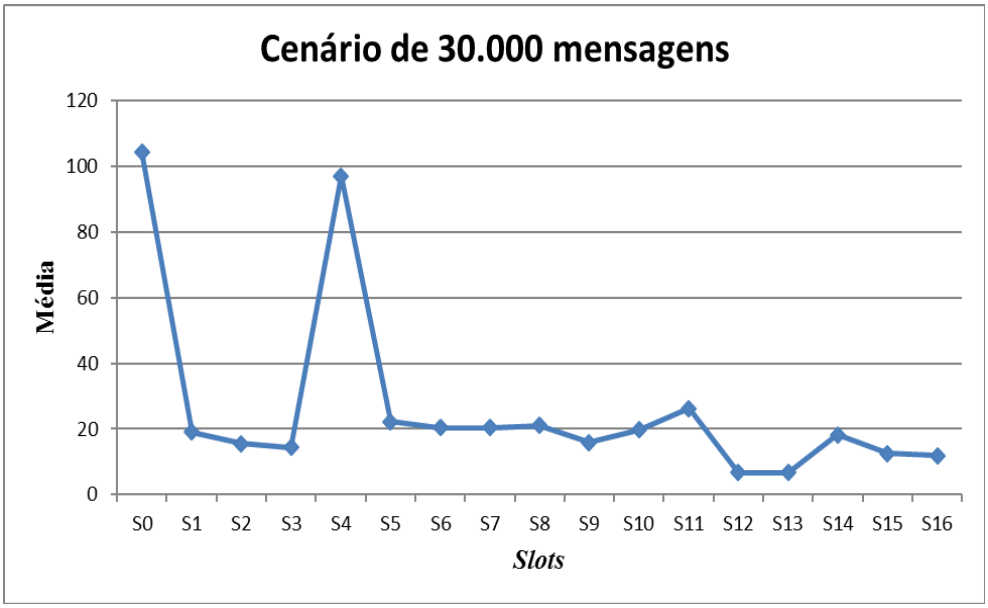


Figura A.3: *Média de Mensagens Acumuladas com entrada de 30.000 mensagens .*

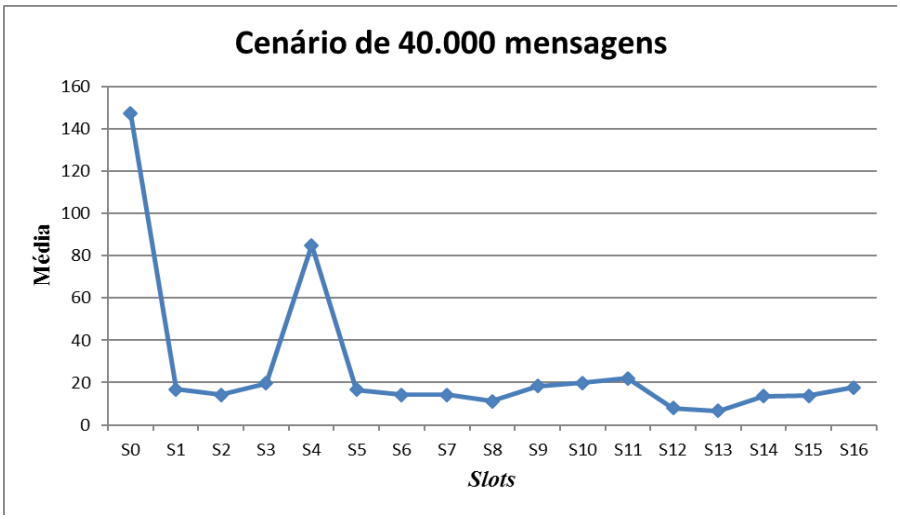


Figura A.4: *Média de Mensagens Acumuladas com entrada de 40.000 mensagens .*

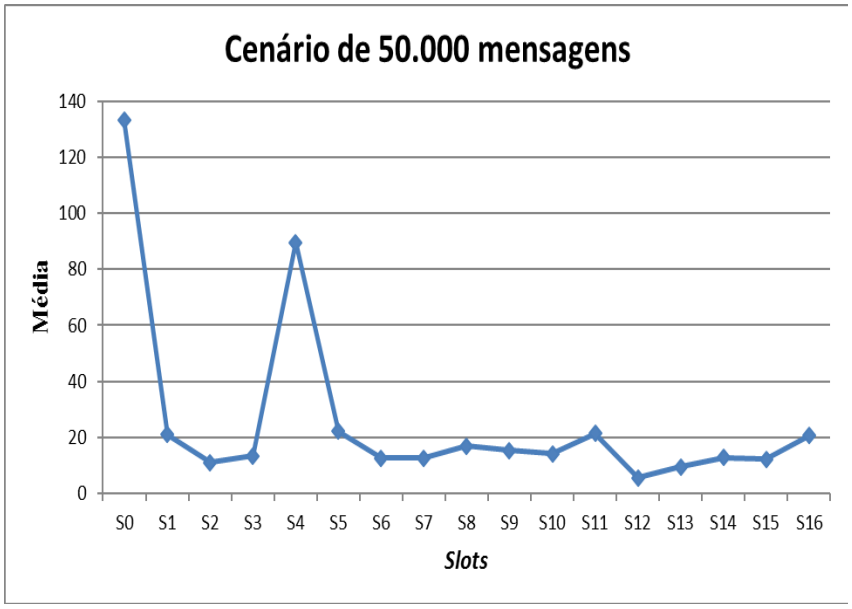


Figura A.5: *Média de Mensagens Acumuladas com entrada de 50.000 mensagens .*

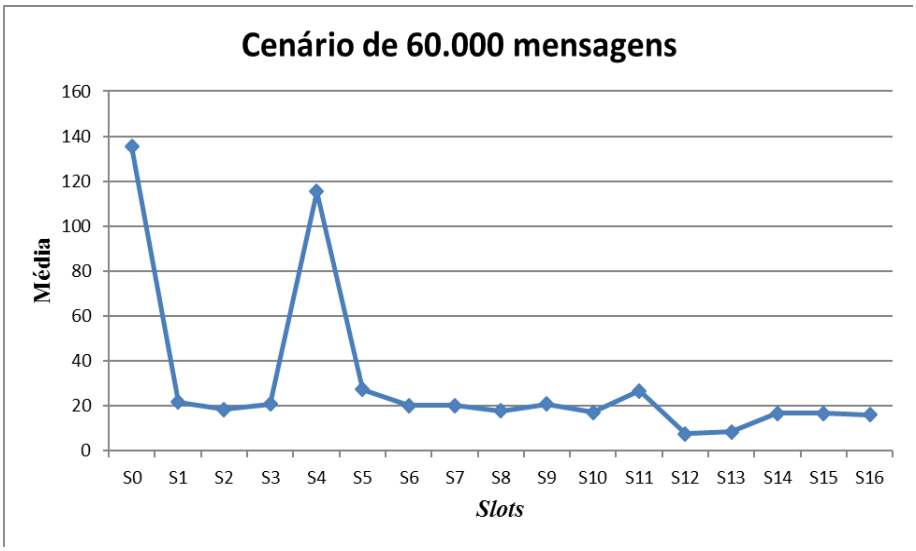


Figura A.6: *Média de Mensagens Acumuladas com entrada de 60.000 mensagens .*

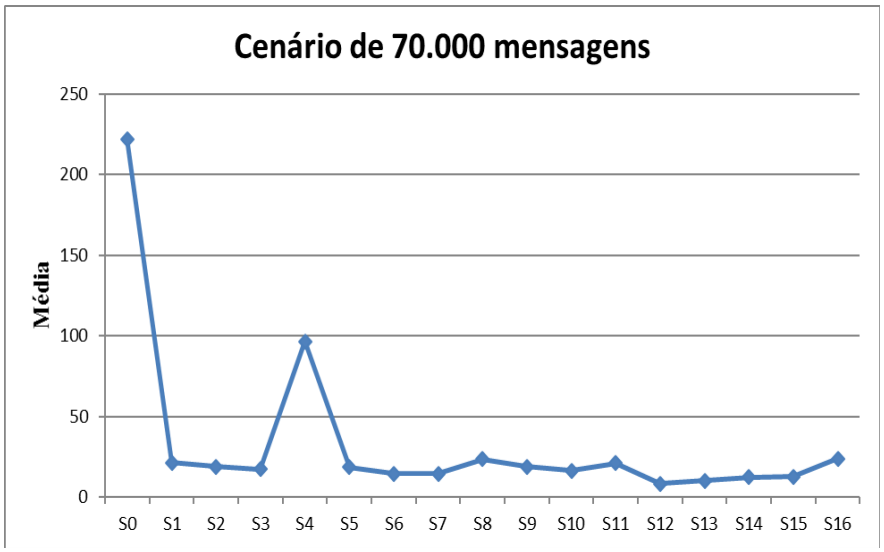


Figura A.7: *Média de Mensagens Acumuladas com entrada de 70.000 mensagens .*

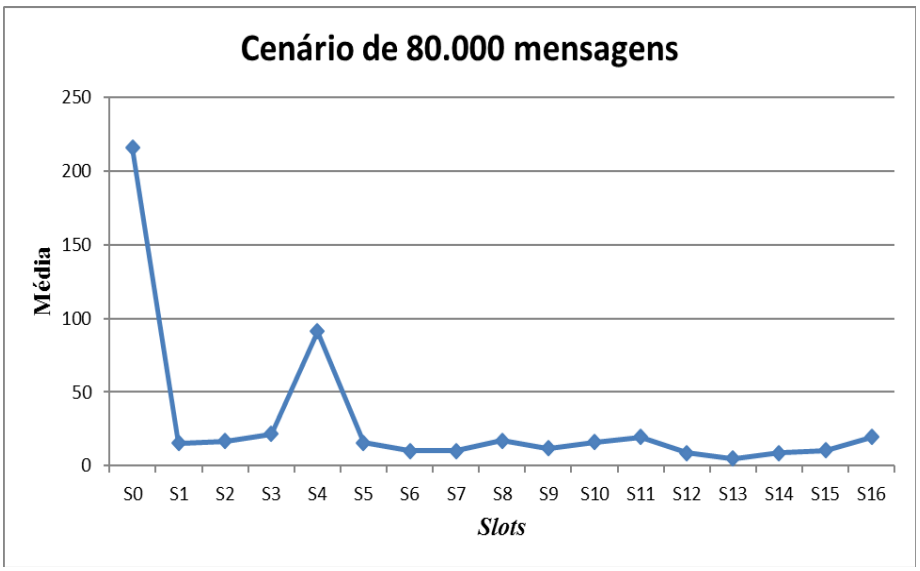


Figura A.8: *Média de Mensagens Acumuladas com entrada de 80.000 mensagens .*

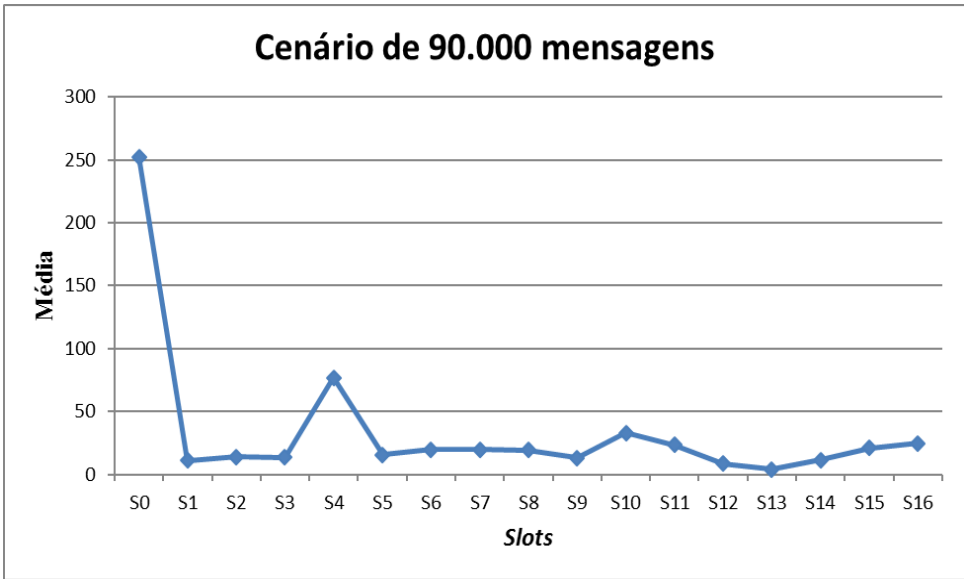


Figura A.9: *Média de Mensagens Acumuladas com entrada de 90.000 mensagens .*

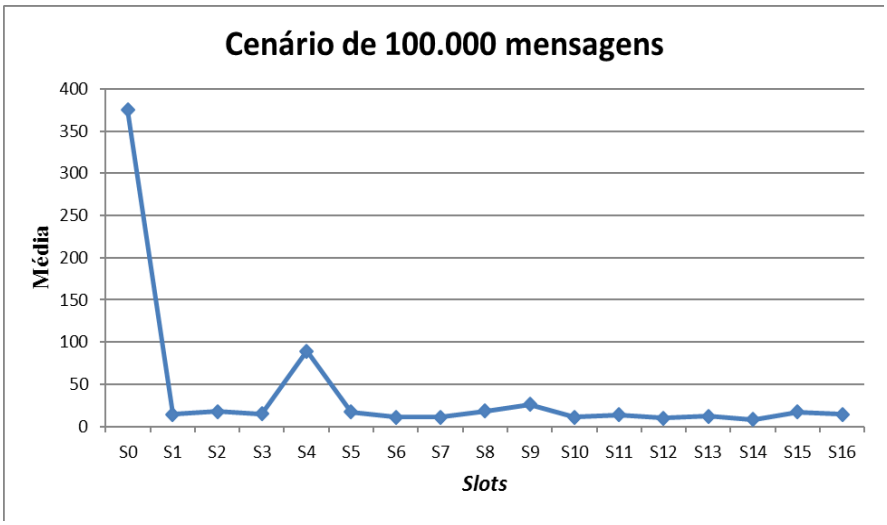


Figura A.10: *Média de Mensagens Acumuladas com entrada de 100.000 mensagens .*

Apêndice B

Verificações efetuadas nos Cenários

Nesta Seção são apresentadas as verificações referentes à Seção §5.4. Estão descritas as verificações de cada cenário, para a comprovação da legitimidade da implementação do modelo proposto na ferramenta de simulação.

Na Tabela §B.1 é apresentado os resultados de uma simulação com 20.000 mensagens no sistema. Verificando as equação no cenário de 20.000 mensagens de entrada, obtemos:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
392254	54	9	107	13	174	26	19	19

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
3	0	113	4	21	15	3	31	9

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
1	27	18	49	27	20000	20000	18796	18581

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
17728	18448	18926	17782	18537	1020	935	181

Tabela B.1: *Cenário com entrada de 20.000 mensagens no sistema*

$$V_1 = 20.000 - \left(1226 + \frac{193+193}{2} + 18581\right)$$

$$V_1 = 0$$

$$V_2 = 20.000 - (2079 + \frac{193+193}{2} + 17728)$$

$$V_2 = 0$$

$$V_3 = 20.000 - (1359 + \frac{193+193}{2} + 18448)$$

$$V_3 = 0$$

$$V_1 = V_2 = V_3 = 0$$

Na Tabela §B.2 é apresentado os resultados de uma simulação com 30.000 mensagens no sistema.Cenário com entrada de 30.000 mensagens no sistema, obtemos:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
589840	109	2	39	3	108	7	40	40

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
27	19	28	28	1	7	23	12	2

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
37	2	2	1	22	30000	30000	28242	28097

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
26692	27806	28323	26702	27821	1568	1416	302

Tabela B.2: *Cenário com entrada de 30.000 mensagens no sistema*

$$V_1 = 30.000 - (1755 + \frac{148+148}{2} + 28097)$$

$$V_1 = 0$$

$$V_2 = 30.000 - (3160 + \frac{148+148}{2} + 26692)$$

$$V_2 = 0$$

$$V_3 = 30.000 - (2046 + \frac{148+148}{2} + 27806)$$

$$V_3 = 0$$

$$V_1 = V_2 = V_3 = 0$$

Na Tabela §B.3 é apresentado os resultados de uma simulação com 40.000 mensagens no sistema.Cenário com entrada de 40.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
787937	137	10	17	19	67	18	38	38

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
7	9	30	2	5	3	0	22	0

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
11	2	33	22	2	40000	40000	37682	37576

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
35668	37188	37739	35709	37232	2124	1906	385

Tabela B.3: *Cenário com entrada de 40.000 mensagens no sistema*

$$V_1 = 40.000 - (2319 + \frac{105+105}{2} + 37576)$$

$V_1 = 0$

$V_2 = 40.000 - \left(4227 + \frac{105+105}{2} + 35668\right)$

$V_2 = 0$

$V_3 = 40.000 - \left(2278 + \frac{105+105}{2} + 37188\right)$

$V_3 = 0$

$V_1 = V_2 = V_3 = 0$

Na Tabela §B.4 é apresentado os resultados de uma simulação com 50.000 mensagens no sistema.Cenário com entrada de 50.000 mensagens no sistema,obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
990243	145	33	4	1	39	7	20	20

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
37	3	2	41	7	2	8	26	35

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
18	27	3	14	9	50000	50000	47374	47269

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
44940	46721	47430	44952	46796	2425	2308	497

Tabela B.4: *Cenário com entrada de 50.000 mensagens no sistema*

$V_1 = 50.000 - \left(2672 + \frac{59+59}{2} + 47269\right)$

$V_1 = 0$

$$V_2 = 50.000 - \left(5001 + \frac{59+59}{2} + 44940\right)$$

$$V_2 = 0$$

$$V_3 = 50.000 - \left(3220 + \frac{59+59}{2} + 46721\right)$$

$$V_3 = 0$$

$$V_1 = V_2 = V_3 = 0$$

Na Tabela §B.5 é apresentado os resultados de uma simulação com 60.000 mensagens no sistema.Cenário com entrada de 60.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1183635	248	13	9	60	171	16	2	2

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
28	16	20	39	9	1	8	23	46

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
53	9	9	1	33	60000	60000	56606	56482

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
53580	55887	56741	53599	55957	3011	2889	562

Tabela B.5: *Cenário com entrada de 60.000 mensagens no sistema*

$$V_1 = 60.000 - \left(3345 + \frac{173+173}{2} + 56482\right)$$

$$V_1 = 0$$

$$V_2 = 60.000 - \left(6247 + \frac{173+173}{2} + 53580\right)$$

$V_2 = 0$

$V_3 = 60.000 - (3840 + \frac{173+173}{2} + 55887)$

$V_3 = 0$

$V_1 = V_2 = V_3 = 0$

Na Tabela §B.6 é apresentado os resultados de uma simulação com 70.000 mensagens no sistema.Cenário com entrada de 70.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1386490	81	3	29	1	89	1	42	42

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
10	2	16	22	10	19	25	22	2

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
6	27	8	1	52	70000	70000	66343	66193

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
62788	65534	66382	62825	65559	3537	3391	654

Tabela B.6: *Cenário com entrada de 70.000 mensagens no sistema*

$V_1 = 70.000 - (3676 + \frac{131+131}{2} + 66193)$

$V_1 = 0$

$V_2 = 70.000 - (7081 + \frac{131+131}{2} + 62788)$

$$V_2 = 0$$

$$V_3 = 70.000 - \left(4335 + \frac{131+131}{2} + 65534\right)$$

$$V_3 = 0$$

$$V_1 = V_2 = V_3 = 0$$

Na Tabela §B.7 é apresentado os resultados de uma simulação com 80.000 mensagens no sistema.Cenário com entrada de 80.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1583011	359	22	0	11	30	2	8	8

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
39	31	52	27	52	18	24	2	30

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
7	27	14	45	10	80000	80000	75698	75529

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
71771	74815	75738	71855	74892	3903	3757	723

Tabela B.7: *Cenário com entrada de 80.000 mensagens no sistema*

$$V_1 = 80.000 - \left(4433 + \frac{38+38}{2} + 75529\right)$$

$$V_1 = 0$$

$$V_2 = 80.000 - \left(8191 + \frac{38+38}{2} + 71771\right)$$

$V_2 = 0$

$V_3 = 80.000 - (5147 + \frac{38+38}{2} + 74815)$

$V_3 = 0$

$V_1 = V_2 = V_3 = 0$

Na Tabela §B.8 é apresentado os resultados de uma simulação com 90.000 mensagens no sistema.Cenário com entrada de 90.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1782492	142	20	9	5	61	25	7	7

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
14	1	34	0	6	1	1	19	123

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
5	47	6	1	17	90000	90000	85220	85075

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
80864	84205	85259	80877	84348	4599	4280	808

Tabela B.8: *Cenário com entrada de 90.000 mensagens no sistema*

$V_1 = 90.000 - (4857 + \frac{68+68}{2} + 85075)$

$V_1 = 0$

$V_2 = 90.000 - (9068 + \frac{68+68}{2} + 80864)$

$V_2 = 0$

$V_3 = 90.000 - (5727 + \frac{68+68}{2} + 84205)$

$V_3 = 0$

$V_1 = V_2 = V_3 = 0$

Na Tabela §B.9 é apresentado os resultados de uma simulação com 100.000 mensagens no sistema.Cenário com entrada de 100.000 mensagens no sistema, obtém-se:

Step	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1978322	415	18	3	15	80	0	3	3

S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
24	14	13	15	10	4	1	14	6

sp1	sp2	sp3	sp4	SRH	M	P_0	P_1	P_2
29	2	3	2	33	100000	100000	94564	94475

P_3	P_4	X_1	X_2	X_3	F_1	F_2	F_3
89750	93521	94629	89767	93543	4956	4722	960

Tabela B.9: *Cenário com entrada de 100.000 mensagens no sistema*

$V_1 = 100.000 - (5442 + \frac{83+83}{2} + 94475)$

$V_1 = 0$

$V_2 = 100.000 - (10167 + \frac{83+83}{2} + 89750)$

$$V_2 = 0$$

$$V_3 = 100.000 - \left(6396 + \frac{83+83}{2} + 93521\right)$$

$$V_3 = 0$$

$$V_1 = V_2 = V_3 = 0$$

Bibliografia

[1] E. Abrahám, N. Jansen, R. Wimmer, J.-P. Katoen e B. Becker. *Dtmc model checking by scc reduction*. Em *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, páginas 37–46. IEEE, 2010.

[2] A. K. Ahluwalia e M. Singhal. *Performance analysis of the communication architecture of the connection machine*. *Parallel and Distributed Systems, IEEE Transactions on*, 3(6):728–738, 1992.

[3] R. Alur e T. A. Henzinger. *Reactive modules*. *Formal Methods in System Design*, 15(1):7–48, 1999.

[4] S. Basagiannis, P. Katsaros, A. Pombortsis e N. Alexiou. *A probabilistic attacker model for quantitative verification of dos security threats*. Em *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, páginas 12–19. IEEE, 2008.

[5] E. d. O. Batista. *Sistema de informação: o uso consciente da tecnologia para o gerenciamento*. Saraiva, 2004.

[6] G. Bolch, S. Greiner, H. de Meer e K. S. Trivedi. *Queueing networks and markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[7] J. Cardoso e R. Valette. *Redes de petri*. Editora da UFSC, 1997.

[8] E. A. Carrión. *Teoria das filas como ferramenta para análise de desempenho de sistemas de atendimento: estudo do caso de um servidor da UECE*. Dissertação de Mestrado, Universidade Estadual do Ceará, 2007.

[9] J. S. Carson et al.. *Model verification and validation*. Em *Simulation Conference, 2002. Proceedings of the Winter*, volume 1, páginas 52–58. IEEE, 2002.

[10] D. D. T. d. Carvalho. *Metodologia de análise do desempenho da usina de concentração da samarco mineração sa baseada em simulação das operações*. 2003.

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

[12] J. Clément, C. Delporte-Gallet, H. Fauconnier e M. Sighireanu. *Guidelines for the verification of population protocols*. Em *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, páginas 215–224. IEEE, 2011.

[13] M. I. Clune, P. J. Mosterman e C. G. Cassandras. *Discrete event and hybrid system simulation with simevents*. Em *Proceedings of the 8th international workshop on discrete event systems*, páginas 386–387, 2006.

[14] E. Costa e A. Lima. *Sistemas dinâmicos a eventos discretos: Fundamentos básicos para a moderna automação industrial, ed. UFBA, Salvador, BA*, 2005.

[15] P. J. de Freitas Filho. *Introdução à modelagem e simulação de sistemas: com aplicações em arena*. Visual Books, 2001.

[16] P. J. de Freitas Filho. *Introdução à modelagem e simulação de sistemas: com aplicações em arena*. Visual Books, 2001.

[17] G. P. Dimuro, R. H. Reiser, A. C. Costa e P. Souza. *Modelos de markov e aplicações. VI Oficina de Inteligência Artificial, Pelotas: Educat*, páginas 37–59, 2002.

[18] N. J. Dingle, W. J. Knottenbelt e T. Suto. *Pipe2: a tool for the performance evaluation of generalised stochastic petri nets. ACM SIGMETRICS Performance Evaluation Review*, 36(4):34–39, 2009.

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

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

[21] R. Frantz, S. Sawicki, F. Roos-Frantz, R. Corchuelo, V. Basto-Fernandes e I. Hernández. *Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem*. 2014.

[22] 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.

[23] R. Z. Frantz, A. M. Reina 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.

[24] R. Z. Frantz. *Enterprise application integration: an easy-to-maintain model-driven engineering approach*. Tese Doutoral, Universidad de Sevilla, 2012.

[25] A. V. P. Gomes e P. Wanke. *Modelagem da gestão de estoques de peças de reposição através de cadeias de markov*. *Gestão & Produção*, 15 (1):57–72, 2008.

[26] C. M. Grinstead e J. L. Snell. *Introduction to probability*. American Mathematical Soc., 1997.

[27] E. M. Hahn. *Parametric markov model analysis*. Tese Doutoral, Saarland University, 2008.

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

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

[30] M. R. James, V. Krishnamurthy e F. Le Gland. *Time discretization of continuous-time filters and smoothers for hmm parameter estimation*. *Information Theory, IEEE Transactions on*, 42(2):593–605, 1996.

[31] D. G. Kendall. *Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain*. *The Annals of Mathematical Statistics*, páginas 338–354, 1953.

[32] J. A. Kumar e S. Vasudevan. *Variation-conscious formal timing verification in rtl*. Em *VLSI Design (VLSI Design), 2011 24th International Conference on*, páginas 58–63. IEEE, 2011.

[33] J. A. Kumar e S. Vasudevan. *Formal probabilistic timing verification in rtl*. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(5):788–801, 2013.

[34] M. Kwiatkowska, G. Norman e D. Parker. *Stochastic model checking*. Em *Formal methods for performance evaluation*, páginas 220–270. Springer, 2007.

[35] M. Kwiatkowska, G. Norman e D. Parker. *Prism 4.0: Verification of probabilistic real-time systems*. Em *Computer aided verification*, páginas 585–591. Springer, 2011.

[36] Y. Kwon e G. Agha. *Scalable modeling and performance evaluation of wireless sensor networks*. Em *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*, páginas 49–58. IEEE, 2006.

[37] K. C. Laudon e J. P. Laudon. *Sistemas de informação gerenciais: administrando a empresa digital*. Pearson Prentice Hall, 2011.

[38] A. M. Law e W. D. Kelton. *Simulation modeling and analysis, mcgraw-hill. New York*, 1991.

[39] D. G. Messerschmitt e C. Szyperski. *Software ecosystem: understanding an indispensable technology and industry*. *MIT Press Books*, 1, 2005.

[40] J. d. MORAES et al.. *Usando arquétipos e linguagem específica de domínio no desenvolvimento de aplicações ubíquas para o cuidado de saúde pervasivo*. Em *CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE*, volume 13, 2012.

[41] G. Norman, C. Palamidessi, D. Parker e P. Wu. *Model checking the probabilistic pi-calculus*. Em *Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the*, páginas 169–178. IEEE, 2007.

[42] V. U. C. Nunes. *Gerência de variabilidade em modelos de confiabilidade para linha de produtos de software*. 2013.

[43] J. A. O'Brien, C. K. Moreira e C. K. Moreira. *Sistemas de informação e as decisões gerenciais na era da internet*. Saraiva, 2004.

[44] G. B. Oliveira. *Simulação computacional: Análise de um sistema de manufatura em fase de desenvolvimento*. Dissertação de Mestrado, Universidade Federal de Engenharia de Itajubá, 2007.

[45] P. Orientadores. *Reconhecimento de voz e de locutor em ambientes ruidosos: comparação das técnicas mfcc e zcpa*. 2007.

[46] D. Parker. *Lectures - probabilistic model checking*. University of Oxford, Department of Computer Science, 2011.

[47] L. Paulevé, M. Magnin e O. Roux. *Tuning temporal features within the stochastic π-calculus*. *Software Engineering, IEEE Transactions on*, 37(6): 858–871, 2011.

[48] M. Pidd. *Modelagem empresarial: ferramentas para tomada de decisão*. Bookman, 1998.

[49] G. Pinheiro. *Teoria de filas e sistemas de comunicação. Apostila de Aula. Departamento de Engenharia Eletrônica e Telecomunicações, Universidade Estadual do Rio de Janeiro–UERJ*, 2013.

[50] D. Prado. *Teoria das filas e da simulação. Belo Horizonte, MG: Editora de Desenvolvimento Gerencial*, 2, 1999.

[51] D. Prado. *Teoria das filas e da simulação. Belo Horizonte, Nova Lima: Editora FALCONI*, 5, 2014.

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

[53] H. G. Reyes e L. E. C. Barrón. *Simulación y análisis de sistemas con promodel*. Pearson Educación, 2006.

[54] W. Sandmann. *On optimal importance sampling for discrete-time markov chains*. Em *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, páginas 230–239. IEEE, 2005.

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

[56] I. Sommerville. *Engenharia de software*. Pearson Addison Wesley São Paulo, 2003.

[57] M. F. Triola et al.. *Introdução à estatística*, volume 9. Ltc Rio de Janeiro, 2005.

[58] W. M. van der Aalst. *Business process simulation survival guide*. Em *Handbook on Business Process Management 1*, páginas 337–370. Springer, 2015.

This document was typeset on May 8, 2016 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.