

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 06.08.2001

Assinatura: 

Statecharts estocásticos e queuing
statecharts: novas abordagens para avaliação
de desempenho baseadas em especificação
statecharts

Carlos Renato Lisboa Francês

Orientador: *Prof. Dr. Marcos José Santana*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

USP – São Carlos
Agosto de 2001

A Comissão Julgadora:

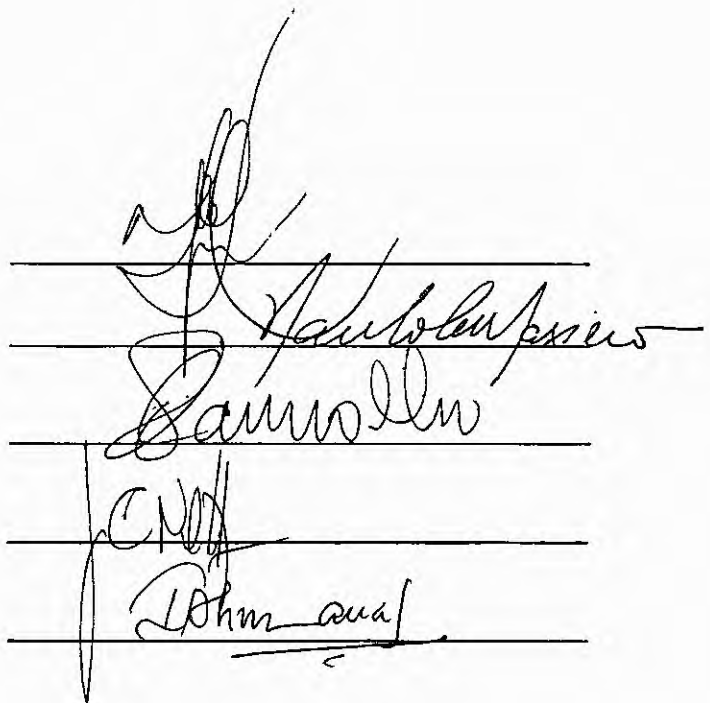
Prof. Dr. Marcos José Santana

Prof. Dr. Paulo Cesar Masiero

Prof. Dr. Solon Venâncio de Carvalho

Prof. Dr. João Cesar Netto

Prof. Dr. Vakulathil Abdurahiman



The image shows five horizontal lines, each with a handwritten signature written across it. The signatures are written in black ink and are somewhat stylized. The first signature is the most complex, with many loops. The second signature is more legible, appearing to be 'Paulo Cesar Masiero'. The third signature is also legible, appearing to be 'Solon Venâncio de Carvalho'. The fourth signature is very stylized and difficult to read. The fifth signature is also stylized, appearing to be 'Vakulathil Abdurahiman'.

"A imaginação é mais poderosa do que o conhecimento. Ela amplia a visão, dilata a mente, desafia o impossível. Sem a imaginação o pensamento estagna."

(Albert Einstein)

Dedicatória

À minha esposa, Regiane, por toda compreensão, apoio, amor e paciência que ela, com toda nobreza do mundo, soube me transmitir.

À minha mãe, Antônia, que, apesar de todas as adversidades da vida, ainda tem esperança que dias melhores virão.

Agradecimentos

Como diria o poeta Gonzaga Jr. (o Gonzaguinha)... "Toda pessoa sempre é as marcas das lições diárias de outras tantas pessoas"... e eu posso me considerar um privilegiado, por tudo o que eu aprendi com todas as pessoas que eu pude conviver...

Com o medo de incorrer no erro imperdoável de esquecer alguém, seguem os agradecimentos.

A Deus, por ter-me proporcionado incontáveis lições de humildade, amizade e solidariedade. Valores que, às vezes, na batalha do dia-a-dia, a gente esquece.

À Regiane, minha esposa, que se sacrificou junto comigo durante um longo período de três anos, nos quais eu estive ausente em muitos momentos importantes... mas que mesmo assim teve a capacidade de me proporcionar amor, carinho, compreensão e motivação, para que eu pudesse continuar em frente.

Aos meus pais, que se sacrificaram na tarefa de educar os filhos, num país onde essa tarefa essencial se transforma em um grande desafio.

Ao meu orientador, Marcos Santana, por ter-me proporcionado muito além da orientação acadêmica... pela sensibilidade para entender os meus problemas e limitações... pelos debates mais acalorados... Muito obrigado!

Ao professor Solon, um especialista em cadeias de Markov, que vislumbrou muitas das idéias desta tese, e que sempre as relatou com muita humildade, como se fosse uma rele opinião de um leigo... Muito obrigado por toda a ajuda!

À professora Regina Santana, pelas opiniões, discordâncias, dicas e conselhos, ao longo de todos esses anos de convívio.

À professora Maria Creusa, catedrática em processos estocásticos, que teve paciência para responder às minhas perguntas, por mais insipientes que elas fossem.

Ao professor Vijay, pela amizade, pelas dicas e por todos os trabalhos realizados em parceria.

Ao casal Laura e Jorge, pelos momentos alegres, pela amizade e por me tratarem como se eu estivesse em minha própria casa.

Ao amigo Ricardo (Kid Ricrs), por toda a ajuda no estudo de caso, pelas conversas e por todos os bandejões no RU.

Ao amigo Márcio Augusto, pelas brincadeiras e também pela possibilidade de sempre poder contar com ele nas horas difíceis.

À amiga Sarita, pela perspicácia na leitura e na depuração do código smpl.

À amiga Kalinka, por todas as batalhas (disciplinas, qualificações e TOEFL)...que dificuldade!

Aos amigos: Omar, Arion, Tomás, Mário, JC, Álvaro, Mara, Tatiana, Célia, Edmilson, Luciano, Aletéia, Paulo Horst, que fizeram com que os dias ficassem mais amenos, durante o longo período do mestrado e doutorado.

À Simone, pela revisão minuciosa dos capítulos e ao Paulo, por todas as dicas e cafezinhos.

À Andrezza, por ter realizado um trabalho extremamente didático sobre métodos analíticos, o qual foi de grande valia para a confecção deste trabalho.

Ào Ivan e Alex, grandes amigos, com os quais eu aprendo constantemente.

À Beth, Laura e Marília, que fazem o possível e impossível para que tudo transcorra na mais absoluta ordem.

À UNAMA e, em especial, à professora Núbia, que contribuiu (com o seu bom senso) de maneira decisiva para que eu pudesse chegar a este momento.

Ao Ricardo Ferreira, coordenador do curso da UNAMA, por não ter medido esforços para me ajudar em tudo o que foi possível.

À USP, pelo seu incrível senso de profissionalismo, que serve de exemplo de educação gratuita e de qualidade. A USP faz jus ao seu *slogan*: "Educação para o Brasil"!

Índice

LISTA DE FIGURAS	VIII
LISTA DE TABELAS	X
LISTA DE SIGLAS/ABREVIATURAS	XI
RESUMO	XII
ABSTRACT	XIII
1 INTRODUÇÃO	1
2 MODELAGEM E AVALIAÇÃO DE DESEMPENHO	9
2.1 CONSIDERAÇÕES INICIAIS	9
2.2 O PROCESSO DE MODELAGEM	10
2.3 REDES DE FILAS (RF).....	11
2.3.1 <i>Distribuições de Probabilidades</i>	12
2.3.2 <i>Disciplinas de Atendimento aos Clientes</i>	12
2.3.3 <i>Relação Fila/Servidor em Centros de Serviços</i>	13
2.3.4 <i>Representação Gráfica de Redes de Filas - Discussão</i>	15
2.4 REDES DE PETRI (RP).....	15
2.4.1 <i>Definição de RP</i>	16
2.4.2 <i>Redes de Petri Elementares</i>	18
2.4.3 <i>Redes de Petri Coloridas</i>	20
2.4.4 <i>Redes de Petri Estocásticas</i>	23
2.4.5 <i>Representação Gráfica de Redes de Petri - Discussão</i>	24
2.5 STATECHARTS (SC)	25
2.5.1 <i>Paralelismo e Hierarquia de Componentes</i>	27
2.5.2 <i>Entrada por Condição</i>	28
2.5.3 <i>Estados Parametrizados</i>	28
2.5.4 <i>Delay e Timeout</i>	29
2.5.5 <i>Representação Gráfica dos Statecharts - Discussão</i>	29
2.6 SOLUÇÕES PARA O MODELO.....	30
2.6.1 <i>Solução Analítica</i>	30
2.6.2 <i>Solução por Simulação</i>	31
2.7 CONSIDERAÇÕES FINAIS.....	32
3 PROBABILIDADE, ESTATÍSTICA E PROCESSOS ESTOCÁSTICOS EM AVALIAÇÃO DE DESEMPENHO	33
3.1 CONSIDERAÇÕES INICIAIS.....	33
3.2 COLETA E ANÁLISE DE DADOS	33
3.3 MEDIDAS ESTATÍSTICAS	34
3.3.1 <i>Medidas de Locação</i>	34

3.3.2	<i>Medidas de Dispersão</i>	35
3.4	PROBABILIDADE	36
3.4.1	<i>Conceitos Preliminares</i>	36
3.4.2	<i>Definições de Probabilidade</i>	36
3.4.3	<i>Probabilidades Condicionais e Eventos Independentes</i>	37
3.4.4	<i>Variáveis Aleatórias (v.a.) e Distribuições de Probabilidades</i>	38
3.4.5	<i>Distribuição de Probabilidades Discreta</i>	39
3.4.6	<i>Distribuição de Probabilidades Contínua</i>	40
3.4.7	<i>Distribuições de Probabilidades Discretas Clássicas</i>	41
3.4.8	<i>Distribuições de Probabilidades Contínuas Clássicas</i>	45
3.5	PROCESSOS ESTOCÁSTICOS E CADEIAS DE MARKOV	49
3.5.1	<i>Processos Estocásticos</i>	50
3.5.2	<i>Cadeias de Markov de Parâmetro Discreto</i>	50
3.5.3	<i>Cadeias de Markov de Parâmetro Contínuo</i>	53
3.6	INTRODUÇÃO À TEORIA DE FILAS	55
3.6.1	<i>Condição de Estabilidade</i>	56
3.6.2	<i>Clientes no Sistema e Clientes na Fila</i>	56
3.6.3	<i>Tempo de Resposta e Tempo de Fila</i>	57
3.6.4	<i>Leis Operacionais</i>	57
3.6.4.1	<i>Lei de Little</i>	58
3.6.4.2	<i>Lei do Fluxo Forçado</i>	59
3.6.4.3	<i>Lei de Utilização</i>	60
3.6.4.4	<i>Lei Geral do Tempo de Resposta</i>	60
3.7	CONSIDERAÇÕES FINAIS	61
4	STATECHARTS ESTOCÁSTICOS E QUEUING STATECHARTS	62
4.1	CONSIDERAÇÕES INICIAIS	62
4.2	SINTAXE DOS STATECHARTS	62
4.3	ADAPTAÇÕES NECESSÁRIAS À VISÃO ESTOCÁSTICA (STATECHARTS ESTOCÁSTICOS)	65
4.3.1	<i>Escolha por Condição Probabilística</i>	66
4.3.2	<i>A Semântica dos Statecharts: Passos, Configurações e Micro-configurações</i>	67
4.3.3	<i>Redefinição de Passos, Configurações e Micro-configurações para Statecharts Estocásticos</i>	71
4.3.4	<i>Templates e Eventos-Padrão para Sistemas de Filas</i>	75
4.4	QUEUING STATECHARTS	78
4.4.1	<i>Definição de Queuing Statecharts e seus Templates</i>	79
4.4.2	<i>Templates para as Distribuições Hiperexponencial e de Erlang</i>	83
4.5	CONSIDERAÇÕES FINAIS	85
5	SOLUÇÕES PARA MODELOS STATECHARTS	86
5.1	CONSIDERAÇÕES INICIAIS	86
5.2	SOLUÇÃO ANALÍTICA	87
5.2.1	<i>Redes de Jackson</i>	88
5.3	SOLUÇÃO POR SIMULAÇÃO	89
5.3.1	<i>Fases de uma Simulação</i>	90

5.3.2	<i>Orientações para Simulação</i>	92
5.3.3	<i>Software para Simulação</i>	93
5.3.4	<i>A Simulação Baseada em smpl</i>	94
5.4	SOLUÇÕES APLICADAS ÀS ESPECIFICAÇÕES STATECHARTS	95
5.4.1	<i>Solução de Jackson Aplicada aos Queuing Statecharts</i>	96
5.4.2	<i>Solução por Simulação Aplicada aos Statecharts Estocásticos</i>	99
5.4.3	<i>Limitações das Soluções Abordadas</i>	102
5.5	CONSIDERAÇÕES FINAIS	104
6	ESTUDO DE CASO: PROGRAMAÇÃO DISTRIBUÍDA BASEADA EM PVM	106
6.1	CONSIDERAÇÕES INICIAIS	106
6.2	DESCRIÇÃO DO EXPERIMENTO	107
6.2.1	<i>Objetivo do Estudo</i>	107
6.2.2	<i>Características de Hardware e de Software do Ambiente Utilizado</i>	108
6.2.3	<i>O Processo de Instrumentação do Sistema</i>	109
6.3	O PROCESSO DE MODELAGEM DO EXPERIMENTO	110
6.3.1	<i>Especificação do Modelo</i>	110
6.3.2	<i>Parametrização do Modelo</i>	113
6.3.3	<i>Solução do Modelo</i>	114
6.4	CONSIDERAÇÕES FINAIS	120
7	CONCLUSÕES	122
7.1	PONDERAÇÕES FINAIS DESTA TESE	122
7.2	CONTRIBUIÇÕES DESTA TESE	125
7.3	SUGESTÕES PARA TRABALHOS FUTUROS	129
	REFERÊNCIAS BIBLIOGRÁFICAS	132
	APÊNDICE A: INFERÊNCIA ESTATÍSTICA PARA O ESTUDO DE CASO	137
	APÊNDICE B: SMPL - PRINCIPAIS FUNÇÕES, CÓDIGOS-FONTE E INTERVALO DE CONFIANÇA	142
	APÊNDICE C: MÉTODOS DE SOLUÇÃO ANALÍTICA	150
	GLOSSÁRIO	156

Lista de Figuras

Figura 2.1. Etapas do Processo de Modelagem	10
Figura 2.2. Centro de Serviço	11
Figura 2.3. Centro de Serviço com Múltiplos Servidores	13
Figura 2.4. Tipos de Absorção de Centros de Serviço.....	14
Figura 2.5. Redes Abertas, Fechadas e Mistas	14
Figura 2.6. Elementos Básicos de uma Rede de Petri.....	15
Figura 2.7. Ano Letivo representado em Rede de Petri.....	16
Figura 2.8. Seqüenciamento	18
Figura 2.9. Distribuição	18
Figura 2.10. Junção	18
Figura 2.11. Escolha Não-Determinística.....	19
Figura 2.12. (a) Conflito Estrutural (b) Conflito Efetivo.....	19
Figura 2.13. Rede de Petri Colorida de um Cliente TCP/IP	21
Figura 2.14. Linha de Manufatura em Redes Ordinárias	22
Figura 2.15. Linha de Manufatura em Redes Coloridas.....	23
Figura 2.16. Estados W, P e F	25
Figura 2.17. Representação de Estados em Statecharts.....	26
Figura 2.18. Estados, Eventos e Transições.....	27
Figura 2.19. Estados Ortogonais	27
Figura 2.20. (a) Múltiplas Entradas com Condições (b) Entrada por Condição.....	28
Figura 2.21. Estado Parametrizado <i>Processor/Disc</i>	28
Figura 2.22. Estado com Delay de 5 segundos	29
Figura 2.23. Solução Preferencial para um Modelo	31
Figura 3.1 Gráfico em Barras para a Soma de Pontos no Lançamento de Dois Dados.....	39
Figura 3.2 Gráfico da Distribuição Binomial para $n=5$	43
Figura 3.3 Gráfico da Distribuição de Poisson para Valores Distintos de λ	44
Figura 3.4 Gráfico da Distribuição Normal	45
Figura 3.5. Probabilidade igual à Área sob a Curva, entre a e b	46
Figura 3.6. Gráfico da Distribuição Exponencial para Três Valores Diferentes de λ	47
Figura 3.7. Modelo de Erlang e sua Distribuição	48
Figura 3.8. Modelo Hiperexponencial	48
Figura 3.9. Variáveis Aleatórias em um Sistema de Filas.....	49
Figura 3.10. Matriz de Transição	51
Figura 3.11. Matriz de Taxas.	54
Figura 3.12. <i>Links In</i> e <i>Out</i> para Modelos Fechados	59
Figura 4.1 Estados com Retardo e Imediatos	66
Figura 4.2 Escolha por Probabilidade	67
Figura 4.3 Servidor de Arquivos em Notação Statecharts	69
Figura 4.4 Servidor de Arquivos e seus Estados com Retardos Associados	73
Figura 4.5. Linha do Tempo com suas Configurações e Passos.....	74
Figura 4.6. <i>Templates</i> para Fontes Geradoras de Clientes	75
Figura 4.7. <i>Template</i> para Filas de Servidores	76
Figura 4.8. <i>Template</i> para Servidores.....	76

Figura 4.9. <i>Template</i> para Sorvedouros	77
Figura 4.10. <i>Template</i> para Fonte.....	80
Figura 4.11. <i>Template</i> para Fila	80
Figura 4.12. <i>Template</i> para Servidor	80
Figura 4.13. <i>Template</i> para Estados Parametrizados.....	81
Figura 4.14. <i>Template</i> para Centro de Serviço	81
Figura 4.15. <i>Template</i> para Sorvedouro	81
Figura 4.16. Servidor de Arquivos em Representação Queuing Statecharts.....	82
Figura 4.17. <i>Template</i> para Servidor Hiperexponencial.....	84
Figura 4.18. <i>Template</i> para Servidor de Erlang	84
Figura 5.1 Algumas Possíveis Abordagens para a Solução Analítica	88
Figura 5.2 Relações entre Evento, Processo e Atividade	93
Figura 5.3 Servidor de Arquivos Especificado em Queuing Statecharts.....	96
Figura 5.4 Correlação entre Funções smpl e Eventos Statecharts para o Servidor de Arquivos.....	101
Figura 6.1. Especificação do Modelo PVM em Statecharts Estocásticos	111
Figura 6.2. Especificação do Modelo PVM em Queuing Statecharts.....	112
Figura 6.3. Especificação do Modelo PVM em GSPN	113
Figura 6.4. Algoritmo AVM Exato para Modelos Fechados	116
Figura 6.5. Medidas de Desempenho Obtidas pelo Método AVM	117
Figura 6.6. Relatório smpl para 100 Milhões de Unidades de Simulação.....	118
Figura A.1. Função Densidade de Probabilidade para o Serviço PVM.....	138
Figura A.2. Função Densidade de Probabilidade para o Serviço Lasdix	139
Figura A.3. Função Densidade de Probabilidade para o Serviço Lasdpc08.....	139
Figura A.4. Função Densidade de Probabilidade para o Serviço Lasdpc10.....	140
Figura A.5. Função Densidade de Probabilidade para o Serviço Lasdpc11.....	140
Figura B.1. Relação entre Distribuições Normal e t	149
Figura C.1. Diagrama de Estado do Processo de Nascimento-e-Morte	153
Figura C.2. Diagrama do Processo de Nascimento-e-Morte para o Modelo M/M/1	153

Lista de Tabelas

Tabela 3.1. Soma dos Valores dos Pontos Amostrais (x) e suas Probabilidades Associadas ($f(x)$).....	39
Tabela 4.1. Eventos-Padrão e suas Semânticas	77
Tabela 4.2. Nomes de Estados-Padrão e suas Semânticas.....	78
Tabela 4.3. Eventos-Padrão de Queuing Statecharts e suas Semânticas	82
Tabela 5.1. Correlação entre Eventos Statecharts e Funções <i>smpl</i>	100
Tabela 6.1. Descrição do Ambiente Instrumentado	108
Tabela 6.2. Tempos Médios de Serviço.....	109
Tabela 6.3. Resultados Obtidos Analiticamente e por Simulação, onde U (Utilização).....	118
Tabela 6.4. Utilizações Obtidas pela Simulação GSPN.....	119
Tabela 6.5. Comparação entre Resultados das Simulações Equiprováveis (1) e Ponderadas (2)	120
Tabela B.1. Principais Funções Utilizadas em <i>smpl</i>	142

Lista de Siglas e Abreviaturas

SIGLA/ABREVIATURA	SIGNIFICADO
AMIGO	Dyn A Mical Flex I ble Schedul I ng Envir O nment
AVM	Análise do Valor Médio
BCMP	Basktt, Chandy, Muntz e Palacios
DNS	Domain Name System
FCFS	First Come First Served
FTP	File Transfer Protocol
GSPN	Generalized Stochastic Petri Net
HTTP	HyperText Transfer Protocol
LCFS	Last Come First Served
MPI	Message Passing Interface
NIC	Network Interface Card
PVM	Parallel Virtual Machine
QS	Queuing Statecharts
RP	Rede de Petri
RR	Round Robin
SC	Statecharts
SE	Statecharts Estocásticos
SPN	Stochastic Petri Net
smpl	S i M ulation P rogramming L anguage
v.a.	Variável Aleatória

Este trabalho apresenta uma nova abordagem para a avaliação de desempenho, inserindo-se nesse contexto a especificação Statecharts. Essa inserção está devidamente associada a uma solução viável para o modelo.

São formalizadas duas extensões aos Statecharts: (1) os Statecharts Estocásticos, os quais utilizam a notação original dos Statecharts, modificando-se somente a semântica formal, e (2) os Queuing Statecharts, que não possuem a notação Statecharts pura, mas sim uma aglutinação entre a representação Statecharts e a de redes de filas. Na formalização proposta, são redefinidos alguns elementos básicos dos Statecharts, tais como eventos e condições, além de alguns conceitos da dinâmica do sistema, por exemplo, passos e configurações.

As especificações propostas descrevem o funcionamento básico de um sistema de filas genérico, através de *templates* e eventos-padrão. A esses eventos estão associadas tanto uma solução analítica quanto uma solução por simulação *smpl* (*SiMulation Programming Language*).

Com o intuito de demonstrar a aplicabilidade das especificações propostas, é desenvolvido um estudo de caso referente à programação distribuída baseada em PVM (*Parallel Virtual Machine*). Com os parâmetros obtidos empiricamente, os modelos em Statecharts Estocásticos, Queuing Statecharts e redes de Petri estocásticas generalizadas (GSPN) são alimentados e resolvidos tanto através da solução da análise do valor médio (AVM) quanto por simulação *smpl*. Inicialmente, a análise é feita conforme o esquema realizado pelo PVM e, posteriormente, é acrescentado um mecanismo de escalonamento que leva em consideração o tempo médio de serviço das máquinas. Há uma discussão sobre os resultados obtidos a partir dos mecanismos adotados para realizar o escalonamento.

Por fim, são expostas algumas ponderações sobre as idéias apresentadas nos capítulos componentes desta tese, além de serem apresentadas as contribuições provenientes desta pesquisa e alguns trabalhos futuros propostos a partir desta tese.

Abstract

This work presents a new approach to performance evaluation, introducing Statecharts in this context. That addition is associated with a viable solution for the model.

Two extensions for Statecharts have been formalized: (1) Stochastic Statecharts, which use the original notation, with a simple modification in the formal semantics, and (2) Queuing Statecharts, which do not have the pure Statecharts notation, but a join between Statecharts and queuing network representations. In the proposed formalization, some basic elements of Statecharts are redefined, such as events and conditions, besides some concepts referring to the dynamic system behavior, e.g. steps and configurations.

The proposed specifications show the basic behavior of a generic queuing system, by means of templates and standard events. There are solutions, both analytical methods and a `smpl`¹ simulation, associated with each standard event.

Aiming at demonstrating the applicability of the specifications proposed, a case study referring to distributed programming (based on PVM – Parallel Virtual Machine) has been developed. Using the parameters empirically obtained, the models specified in Stochastic Statecharts, Queuing Statecharts and generalized stochastic Petri nets (GSPN) have been solved both by the Mean Value Analysis (MVA) and a `smpl` simulation. At the beginning, the mechanism of PVM for process scheduling is used. Afterwards, a scheduling scheme that considers the mean service time of each machine is added. A discussion about the results is carried out and some conclusions are presented.

At the end, some questions about the ideas presented are exposed, along with some contributions from this thesis and some proposed future lines of research.

¹ `smpl` (SiMulation Programming Language) is an extension to the C language for generating event-based simulations.

Introdução

"O homem se designou a si próprio como um ser que mede e valora, como o animal estimador". A citação do filósofo F. Nietzsche (Marías, 1985) reflete com propriedade o anseio do ser humano por tentar quantificar tudo que está à sua volta, atribuindo a todos os fenômenos uma medida, através da qual se pode caracterizá-los em relação aos demais.

Assim, caracterizar situações se constitui no limiar da tarefa de entendê-las. Se essas situações se configuram através de sistemas¹, então as tarefas de quantificar e comparar podem não ser tão triviais, pelo número de variantes envolvidas. Nesse contexto, é importante que se tenham mecanismos que avaliem o comportamento dos sistemas humanos (sejam eles de cunho social, religioso, tecnológico, etc.) para que se possa ter a noção de quanto benefício ou malefício eles podem gerar àqueles que com eles interagem.

Restringindo-se os sistemas para os que geram algum tipo de bem de consumo ou serviços, tais como os de manufatura, de transporte ou os computacionais, pode-se ter a noção da complexidade do tema. Os sistemas são os mais diversos possíveis, e cada qual possui suas próprias peculiaridades, sensibilidades e mazelas, o que gera uma pletera de incertezas sobre como tratar o assunto. Essa diversidade incitou (e incita até hoje) os filósofos² a entenderem (seja *a priori* ou *a posteriori*) o comportamento de tais sistemas.

Com a evolução tecnológica experimentada nas últimas décadas, outros tipos de requisitos foram acrescentados aos sistemas que propiciam serviços aos seus usuários, como por exemplo o desempenho - aqui, utilizado para conotar algum tipo de satisfação gerada pela eficiência na realização de alguma tarefa. De maneira que a execução dessa tarefa possa ser mais ou menos eficiente, tomando-se algum parâmetro como critério.

A maneira pela qual será avaliado o desempenho depende diretamente das características do sistema envolvido. Em se tratando de sistemas computacionais, pode-se dividir a avaliação em dois paradigmas distintos: aquele que precisa obter

¹ Aqui, o termo é usado em seu caráter mais geral, indicando um "conjunto de elementos, materiais ou ideais, entre os quais se possa encontrar ou definir alguma relação" (Ferreira, 1999).

² No sentido etimológico da palavra: "aquele que tem a intenção de ampliar incessantemente a compreensão da realidade..." (Ferreira, 1999)

medidas no próprio sistema, e a partir delas o avaliador pode estabelecer uma interpretação em relação ao desempenho, e o paradigma que se baseia em criar-se uma abstração do sistema, através da qual se pode estabelecer um certo prognóstico (com um grau de incerteza associado) a respeito do desempenho desse sistema. Logicamente, há uma relação custo/benefício embutida na questão dos paradigmas, que está relacionada ao quanto se pode permitir de intrusão e qual o grau de imprecisão que é aceitável. Contudo, há outra questão que deve ser levantada: quando se pretende ter a idéia daquilo que o sistema pode proporcionar? Essa talvez seja a questão que norteie a escolha por um dos paradigmas. Se uma avaliação *post-mortem* é admitida, então pode ser mais natural utilizar-se a idéia da instrumentação do sistema. Entretanto, o atual estado da arte de certa forma leva à noção de que é necessário um conhecimento antecipado daquilo que está porvir, ou seja, de tudo que um determinado sistema pode oferecer aos seus potenciais usuários, antes mesmo que esse sistema exista. De certa forma, esse grau de "maturidade" se deve ao montante de custos e esforços envolvido, na tentativa de exaurir ou de minimizar as perdas.

Este trabalho se atém ao paradigma da abstração, comumente referenciado como modelagem, a qual, em uma definição simplista, baseia-se em uma associação de uma representação gráfica, denominada de modelo, e um método de cunho aleatório, que forneça medidas para avaliar-se o desempenho do sistema que foi modelado. O termo *modelagem* possui várias conotações nas subáreas da ciência da computação, tal qual a *modelagem de dados* utilizada na engenharia de software. Entretanto, modelagem para fins de avaliação de desempenho, como assumido neste trabalho, é um processo complexo (detalhado no Capítulo 2) que fornece medidas pelas quais pode-se ter uma idéia do comportamento do sistema, geralmente a longo prazo.

Talvez pelo fato de a avaliação de desempenho ser uma área proveniente da teoria das probabilidades, de certa forma, tenha-se negligenciado em relação à representação do modelo, dando-se uma ênfase maior à solução matemática aplicada a esse modelo. Fazendo-se conjecturas, talvez pelo grau de especialização das pessoas envolvidas com o tema, um conjunto de equações seja tão (ou mais) claro que uma representação gráfica do mesmo problema. Essa possibilidade, apesar de compreensível, pode gerar um certo distanciamento entre o desenvolvimento do conhecimento e sua efetiva utilização pela comunidade menos

especializada, que em tese deveria ser o alvo dos benefícios oriundos desse desenvolvimento.

Esta tese se propõe a investigar, dentro de um determinado escopo, as benesses que podem advir de um tratamento mais aprimorado da representação gráfica (a partir de agora referenciada como especificação gráfica ou simplesmente especificação). Mais especificamente, aqui é apresentada a idéia de proporcionar uma especificação em alto nível, baseada nos preceitos da representação Statecharts (Harel, 1987), voltada para a problemática da avaliação de desempenho. Tendo-se sempre a preocupação de não gerar uma especificação que seja estanque à solução, isto é, observando-se a modelagem como um conjunto de fases, dentre as quais se encontram a especificação e a resolução do modelo.

Para delimitar-se mais o problema a ser estudado, o contexto de avaliação de desempenho que se pretende investigar no decorrer deste texto está relacionado ao fato de que o ingrediente determinante no desempenho de um sistema está associado à disputa dos recursos disponíveis pelos clientes que utilizam os serviços prestados por esses recursos. Assim, o sistema (para fins de avaliação de desempenho) pode ser visualizado como um conjunto de recursos que prestam serviços (comumente chamados de servidores) e de clientes que solicitam algum tipo de atendimento nesses recursos (eventualmente gerando filas de solicitações em um determinado servidor). Para estudar o relacionamento entre clientes e servidores, existe uma vertente dos processos estocásticos, denominado de teoria das filas.

Partindo-se da premissa de que a disputa pelo recurso é determinante no desempenho e ainda que tão importante quanto a solução dada ao modelo é a sua especificação, esta tese sugere uma forma original de obter-se a união especificação/solução baseada em duas extensões à representação Statecharts, denominadas de Statecharts Estocásticos e Queuing Statecharts.

De certo que ambas as extensões são fundamentalmente semelhantes aos Statecharts, no que concerne à idéia de vislumbrar-se um sistema como um conjunto de estados, aos quais são aplicados certos eventos, que fazem com que haja transições entre esses estados, à medida que o tempo passa. É importante ressaltar que essa é a abordagem usada em todas as técnicas de especificação referenciadas neste trabalho e, via de regra, na literatura pertinente à área. Dessa forma, tanto redes de filas (Kleinrock, 1975) quanto redes de Petri (Maciel et al.,

1996), apresentadas no Capítulo 2, são conceitualmente semelhantes aos Statecharts, diferindo basicamente na funcionalidade gráfica oferecida.

Entretanto, o fato das técnicas de especificação serem conceitualmente semelhantes pode levar à suposição equivocada que elas possuem o mesmo poder de representação, o que não se constitui em uma regra. Na verdade, originalmente, as redes de filas foram idealizadas para um propósito bem mais específico (o de representar sistemas de filas), daí advém o seu tipo de representação peculiar. Já redes de Petri e Statecharts foram concebidos inicialmente para representar sistemas genéricos (não necessariamente sistemas de filas), o que proporcionou um poder de representação relativamente mais amplo. Contudo, redes de Petri e Statecharts possuem representações diferenciadas, cada uma com suas características próprias - diferenças enfatizadas no decorrer deste texto.

As redes de Petri, idealizadas em (Petri, 1966), já possuem suas extensões voltadas à avaliação de desempenho, tais como as redes de Petri estocásticas (Moloy, 1982) e redes de Petri estocásticas generalizadas (Chiola et al., 1993). Já os Statecharts não tinham o caráter estocástico contemplado. Essa situação começou a ser modificado com alguns trabalhos que apontaram para essa direção, tais como (Francês, 1998) e (Vijaykumar, 1999). Esses trabalhos associavam os Statecharts às cadeias de Markov a tempo discreto e contínuo, respectivamente, provendo dessa maneira uma solução analítica aos Statecharts.

Esta tese apresenta uma generalização dos trabalhos citados anteriormente, os Statecharts Estocásticos, que oferecem *templates* (conjuntos de estados e eventos pré-definidos para designar fontes de geração de clientes, filas, servidores e sorvedouros), através dos quais se pode descrever o funcionamento básico de um sistema de fila, sem uma modificação gráfica dos elementos básicos dos Statecharts. Essa especificação é genérica o suficiente de tal forma que seja possível a representação de elementos distintos às filas e aos servidores. Além dos Statecharts Estocásticos, aqui também é proposta uma representação bem mais específica voltada exclusivamente aos sistemas de filas, os Queuing Statecharts. Essa especificidade é, na verdade, uma aglutinação entre as representações dos Statecharts e das redes de filas, herdando destas a estrutura diagramática e daqueles a abordagem de propagação de eventos e ações entre os diversos componentes do modelo (centros de serviço).

Além da formalização de acordo com os preceitos da semântica estabelecida por D. Harel, em (Harel et al., 1987), as especificações estão sempre atreladas a um

tipo de solução que seja factível. Na verdade, a idéia é que a especificação formalizada nesta tese seja genérica o suficiente para permitir que a mesma possa ser resolvida tanto por simulação, quanto por algum método analítico. O ponto-chave da proposição se solidifica no fato de que sistemas de filas possuem um comportamento relativamente uniforme, quando observados em função dos eventos que são relevantes à avaliação de desempenho. Assim, para descrever-se um sistema de filas há alguns eventos básicos que devem ser levados em consideração, tais como a geração dos clientes a uma determinada taxa, a chegada desses clientes à fila, a tomada e a liberação de um determinado servidor por um cliente. De certa forma, há uma relação (nem sempre tão explícita) entre esses eventos e os eventos constantes nos métodos de solução (tanto analíticos quanto por simulação). Algumas dessas relações, tais como redes de Jackson (Allen, 1990) e simulação *smpl* (MacDougall, 1987), são apresentadas durante este trabalho.

Estruturalmente, esta tese está composta por um capítulo de Introdução, dois capítulos de revisão bibliográfica, um capítulo apresentando a formalização das especificações, um capítulo descrevendo duas soluções (uma analítica e outra por simulação) para o mesmo modelo, um capítulo de estudo de caso e um capítulo de conclusões. Além disso, são utilizados mais três apêndices para descrever os testes estatísticos elaborados para este trabalho, os códigos-fonte dos programas utilizados na solução analítica e na simulação. Os parágrafos subsequentes descrevem em mais detalhes os capítulos que compõem este trabalho.

O Capítulo 2 (após esta introdução) apresenta o processo de modelagem e suas fases componentes. Mais especificamente, esse capítulo realiza uma revisão de algumas das principais técnicas de especificação relatadas na literatura especializada: redes de filas, redes de Petri e Statecharts. Cada técnica é observada sob o prisma de sua representação. Assim, de redes de filas são abordados aspectos como elementos básicos de sua representação, distribuições de probabilidades associadas a cada elemento, composição e estruturas possíveis de centros de serviço e redes abertas, fechadas e mistas. Das redes de Petri, são apresentados os elementos básicos, uma definição formal, as redes elementares através das quais são compostas todas as demais, as redes coloridas e, sucintamente, as estocásticas. Analogamente, dos Statecharts são apresentados os elementos básicos (principalmente estados e eventos), algumas características peculiares, tais como paralelismo e hierarquia entre componentes, entrada por condição, estados parametrizados e *delay* e *timeout*. Ao final de cada seção que

discorre sobre uma técnica, são feitas algumas considerações sobre vantagens e desvantagens daquela representação.

O Capítulo 3 apresenta uma revisão bibliográfica condensada sobre probabilidade e estatística no contexto de avaliação de desempenho. Na primeira parte do capítulo, processa-se a definição de conceitos básicos de estatísticas utilizados no trabalho. Alguns desses conceitos são: população e amostra, medidas de locação (por exemplo, médias, mediana e moda) e medidas de dispersão (tais como, variância e desvio padrão). A segunda parte do capítulo trata da teoria das probabilidades, descrevendo suas definições, variáveis aleatórias, definições de distribuições discretas e contínuas, algumas das principais distribuições discretas (binomial e de Poisson) e contínuas (normal, exponencial, hiperexponencial e de Erlang). Como extrapolação da teoria das probabilidades, são ilustrados conceitos de processos estocásticos e, em particular, o processo estocástico denominado de cadeias de Markov (tanto a parâmetro discreto quanto a contínuo). Finalizando o capítulo, é exposta uma introdução à teoria de filas, para servir como base do capítulo seguinte, onde são definidos os *templates*³ Statecharts para sistemas de filas.

No Capítulo 4, é descrita a formalização das duas especificações propostas neste trabalho: os Statecharts Estocásticos e Queuing Statecharts. Primeiramente, são apresentadas as definições de acordo com a semântica formal estabelecida em (Harel et al., 1987). Em seguida, os conceitos são reformulados segundo a visão estocástica para os Statecharts Estocásticos. Dessa forma, são redefinidos eventos, estados com retardo, escolha por probabilidade, passos e configurações, através das funções que determinam os tempos de configuração e passo sucessores. Após isso, são definidos *templates* para os elementos básicos de um sistema de filas genérico, que são fonte, fila, servidor (em suas variações) e sorvedouro, assim como os eventos-padrão que determinam o relacionamento entre cada componente. Subseqüentemente, é definida formalmente a extensão Queuing Statecharts, a qual herda algumas definições dos Statecharts Estocásticos, assim como são inseridas outras definições necessárias. De maneira simplista, há uma migração dos conceitos de hierarquia e *broadcasting* de comunicação (difusão) dos Statecharts para os diagramas tradicionais de redes de filas. Uma pequena modificação nos eventos-padrão é introduzida, em virtude da mudança no tratamento das filas.

³ Durante este trabalho, o termo *template* será usado para designar um conjunto de estados utilizado para representar o funcionamento de um determinado componente de um sistema de filas.

As soluções possíveis para um modelo são abordadas no Capítulo 5. Nesse capítulo, a partir de um modelo de um servidor de arquivos (caracterizando-se como uma rede aberta, onde há geração externa de clientes), que é tratado analiticamente pelo teorema de Jackson, adota-se a mesma seqüência de eventos-padrão estabelecida pelos *templates*. Em seguida, para o mesmo modelo, é aplicada a solução por simulação baseada em *smpl* (extensão à linguagem C, a qual provê simulação baseada em eventos). A simulação também se vale da mesma estrutura de eventos-padrão e *templates*, sendo que a cada evento da especificação é realizada uma associação com a função *smpl* correspondente, o que cria uma relação biunívoca. Os resultados obtidos pela solução de Jackson e pela simulação *smpl* são comparados. Ao final do capítulo, são levantadas algumas restrições de cada solução, que podem ser fatores limitantes ao uso de uma dessas soluções em determinadas situações.

Com o intuito de exemplificar a aplicabilidade da teoria desenvolvida nos capítulos anteriores, um estudo de caso foi elaborado e é apresentado no Capítulo 6. O estudo de caso é referente a um experimento realizado empiricamente em um ambiente distribuído. Esse experimento se constitui na execução de uma aplicação distribuída por uma rede local baseada em Linux. O software usado para distribuir os processos é o PVM (*Parallel Virtual Machine*), que atende a aplicação e envia os processos para quatro máquinas envolvidas no processamento. A idéia é poder observar as diferenças obtidas ao adotar-se uma política *round robin* como a usada no PVM, considerando-se todas as máquinas com a mesma probabilidade de receber os processos, ou adicionar-se algum tipo de "inteligência" à distribuição, na qual haja um certo tipo de ponderação em relação às máquinas envolvidas. Particularmente, nesse estudo, a ponderação é feita em função dos tempos médios de serviço de cada máquina, após a execução de cem tarefas de ordenação de um vetor. Considerando-se que o modelo descreve uma rede fechada, não havendo geração externa de clientes, aplicou-se a solução analítica conhecida como análise do valor médio (AVM) para a obtenção das medidas de desempenho. Após isso, simulou-se o mesmo modelo, validando-se os valores obtidos na simulação pela comparação com o AVM. Uma vez validada a simulação, realizou-se uma extrapolação admitindo-se que as máquinas não eram equiprováveis, e que a ponderação de cada uma delas estava condicionada ao seu tempo médio de serviço (obtido empiricamente). A título de comparação, o mesmo modelo em redes de Petri também é apresentado, simulado com o auxílio da ferramenta DNANet (Attieh et al.,

1995). Algumas considerações são tecidas a partir dos resultados obtidos com o esquema PVM puro e com a suposição de que havia um escalonador “embutido” no mecanismo de distribuição de processos.

Ao final desta tese, no capítulo de conclusões, são realizadas algumas ponderações sobre tudo aquilo que foi abordado no decorrer do texto. O tom dessas ponderações está fortemente ligado às possibilidades que as especificações apresentadas fornecem, quais as suas funcionalidades, quais os seus domínios de aplicação e algumas de suas limitações. Após essas ponderações, são apresentadas as contribuições deste trabalho, que vão da revisão bibliográfica crítica em relação ao tema à formalização de extensão viável para sistemas de filas baseada em especificação Statecharts. Ao final do Capítulo 7, são enumeradas algumas sugestões para trabalhos futuros que dêem continuidade a este trabalho (ou façam uso dele).

Capítulo 2

Modelagem e Avaliação de Desempenho

2.1. Considerações Iniciais

Quando se pretende avaliar um sistema complexo, o primeiro desafio está em encontrar a técnica adequada para levar a cabo a avaliação. Essa técnica, idealmente, deve deixar a análise incólume. Todavia, nocivamente, ela própria pode ser um fator degenerador do desempenho.

Apesar de, via de regra, as técnicas serem de caráter geral, elas também comumente possuem aplicações preferenciais. Assim, o domínio da aplicação é um dos fatores que pode determinar a escolha de uma técnica em detrimento das demais. Entretanto, independente do domínio da aplicação, intrusões em sistemas já estabelecidos nem sempre são bem-vindas.

De uma maneira genérica, podem-se agrupar as técnicas de avaliação de desempenho em dois grupos, de certa forma, complementares. O primeiro relaciona aquelas técnicas que realizam experimentação no sistema e o segundo grupo relaciona as técnicas que criam abstrações desse sistema, através das quais são feitas inferências sobre o seu funcionamento e de seus componentes. Uma taxonomia para essas técnicas é proposta em (Santana et al., 1997), onde os autores sugerem uma divisão em duas classes: as técnicas de aferição e as de modelagem, sendo que o grande diferencial entre os dois grupos está em ter-se ou não o sistema implementado. Para os casos em que o sistema já existe e, conseqüentemente, pode ser averiguado empiricamente, as técnicas de aferição são mais recomendadas. Estão nessa classe, por exemplo, os *benchmarks*, os protótipos e a coleta de dados (através de monitores de *hardware* e/ou de *software*). Mais referências sobre esse grupo de técnicas podem ser encontradas em (Cortés, 1999) e (Jain, 1991). Em contrapartida, para os sistemas inexistentes, Santana et al. (1997) sugerem as técnicas de modelagem.

Modelagem, no contexto de avaliação de desempenho, é um processo complexo e com um forte teor matemático, mas que de maneira simplista pode ser definida como a utilização de uma abstração que contemple em seu cerne as características essenciais de um sistema real, sendo que através da solução desse

modelo, pode-se ter uma aproximação de como o sistema se comportaria se fosse efetivado. A modelagem é o principal enfoque deste trabalho.

2.2. O Processo de Modelagem

Uma modelagem é um conjunto de etapas que são independentes, entretanto intimamente inter-relacionadas. A Figura 2.1 apresenta as etapas seguidas em um processo de modelagem.

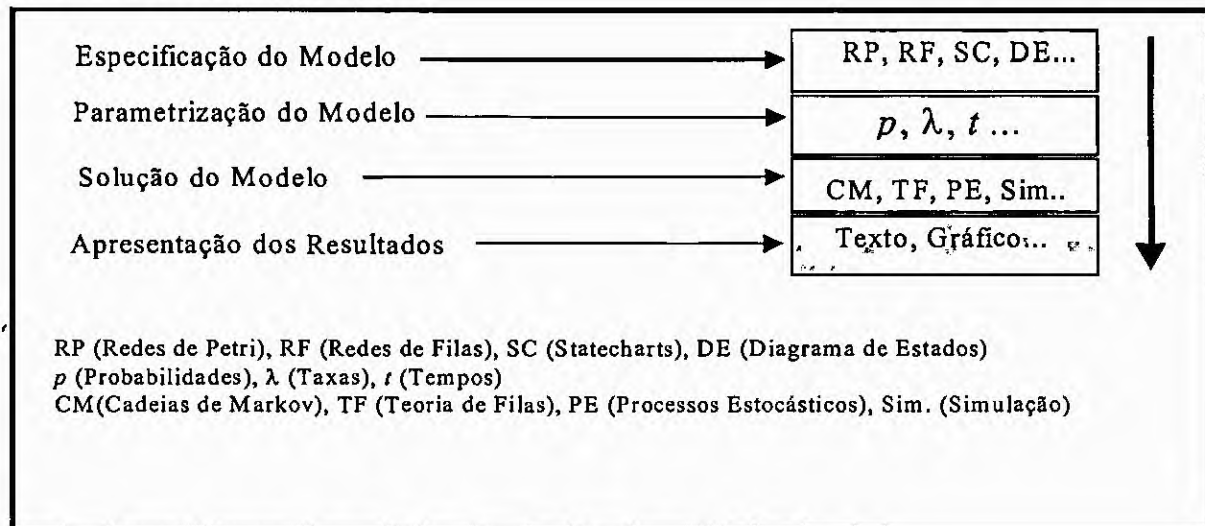


FIGURA 2.1. Etapas do Processo de Modelagem.

Na fase inicial da modelagem, deve-se criar uma *especificação* condizente com o sistema real, na qual devem estar contidos os componentes do sistema relevantes à avaliação, além do relacionamento entre eles. Algumas técnicas usadas para especificação são redes de filas, redes de Petri e Statecharts. Após a confecção do modelo com uma das técnicas citadas, deve-se parametrizá-lo com elementos que serão dados de entrada para a próxima fase (solução). Na parametrização, são comuns taxas, tempos médios e probabilidades. A solução do modelo aplica um método matemático (estocástico), automatizado ou não, para adquirir medidas de desempenho a partir das entradas. Alguns métodos analíticos (cadeias de Markov e teoria de filas, por exemplo) ou simulação são usados nessa fase. Por fim, a modelagem deve apresentar seus resultados através de uma maneira conveniente: gráficos e arquivos-texto são, geralmente, utilizados nessa fase.

Este trabalho aborda os níveis da modelagem, sugerindo tanto aspectos de especificação quanto a associação entre representação gráfica e solução

matemática viável para o modelo, sendo que neste capítulo é dado um enfoque maior às propriedades de especificação características de cada técnica. Posteriormente, as demais fases da modelagem serão abordadas com maior ênfase.

2.3. Redes de Filas (RF)

No limiar desta seção, é importante fazer uma ressalva relativa ao nível de profundidade que a mesma comporta. Na verdade, a abordagem da seção se centraliza bem mais nos aspectos de representação gráfica, passando-se por conceitos fundamentais superficialmente. Entretanto, pela importância dada ao tópico, esse assunto volta à tona no Capítulo 3, no qual é dispensada uma seção na íntegra para discutir-se mais apropriadamente redes de filas¹ e sua teoria associada.

A teoria de filas é um ramo da probabilidade que estuda o fenômeno da formação de filas de solicitantes de serviços, que são providos por um determinado recurso. Ao recurso costuma-se chamar de *servidor*, assim como são chamados de *clientes* os solicitantes de serviços. Ao conjunto fila de clientes e servidor chama-se de centro de serviço. Já uma *rede de filas* é uma coleção de centros de serviço (Soares, 1992). A Figura 2.2 apresenta os elementos gráficos de um centro de serviço.

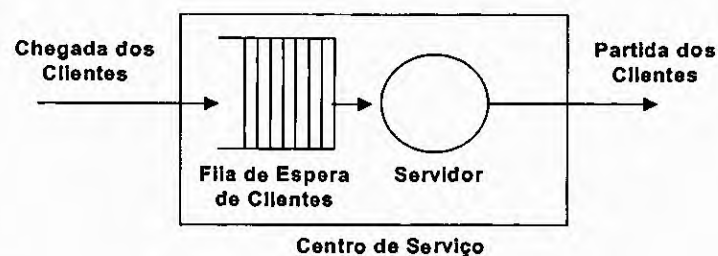


FIGURA 2.2. Centro de Serviço

Uma notação bastante aceita para redes de filas é a notação de Kendall (nome dado em homenagem ao seu criador, o matemático David Kendall). Essa notação descreve uma rede de filas através de seis elementos: $A/B/c/K/m/Z$, onde:

- A - descreve a distribuição dos intervalos entre chegadas ;
- B - descreve a distribuição do tempo de serviço;
- c - é a quantidade de servidores;
- K - é a capacidade máxima do sistema (número máximo de clientes no sistema);
- m - é o tamanho da população que fornece clientes;

¹ Redes e sistemas de filas, neste trabalho, são usados como sinônimos, apesar de, conceitualmente, um sistema de fila com apenas um centro de serviço não constituir uma rede de filas (uma coleção de centros de serviço).

- Z - é a disciplina da fila.

2.3.1. Distribuições de Probabilidades

Uma distribuição de probabilidade descreve o comportamento de uma variável aleatória no decorrer do tempo. Já uma variável aleatória (v.a.) é uma regra de associação de um valor numérico a cada ponto de um espaço amostral. Mais precisamente, uma v.a. é uma função matemática que retorna um número (geralmente um número real) a cada ponto do espaço amostral.

As distribuições de probabilidades, em sistemas de filas, representam o padrão (ritmo) em que os clientes chegam a um centro de serviço e também o padrão de tempo de serviço destinado a cada cliente. Entretanto, elas não definem qual a ordem de atendimento dos clientes. Essa ordem é estabelecida através de algoritmos de escalonamento denominados disciplina de filas.

As distribuições de probabilidades não são abordadas com profundidade neste capítulo. Contudo, pela relevância do assunto, há uma seção dedicada ao tema no Capítulo 3.

2.3.2. Disciplinas de Atendimento aos Clientes.

A disciplina de fila é o mecanismo responsável por decidir qual deverá ser o próximo cliente a ser atendido pelo servidor. Algumas das técnicas usuais são:

FCFS (First Come - First Served) é uma disciplina bastante comum e muito usada. Ela baseia o seu atendimento na ordem de chegada dos clientes, isto é, o primeiro a chegar será o primeiro a ser atendido. Essa disciplina não é apropriada para filas cujos clientes possuam prioridade, ou seja, clientes que mereçam um tratamento diferenciado.

A disciplina *LCFS (Last Come - First Served)* é exatamente o oposto da FCFS: o último a chegar será o primeiro a ser atendido pelo servidor. Essa disciplina possui a estrutura de uma pilha.

RR (Round Robin) é a disciplina na qual um cliente é atendido durante um pequeno intervalo de tempo, denominado *quantum*. Caso esse intervalo não seja suficiente para a realização de todo o serviço requisitado pelo cliente, este é colocado no final da fila novamente, até que o seu serviço seja completado.

As filas, entretanto, podem requerer a atribuição de *prioridades* para os clientes. As prioridades podem ser *preemptivas*, isto é, o atendimento de um cliente

é interrompido caso chegue um cliente com maior prioridade, ou *não preemptivas*, onde o cliente em atendimento não é afetado, porém o próximo a ser atendido será sempre o de maior prioridade.

Uma simplificação da notação $A/B/c/K/m/Z$ usualmente empregada é a notação $A/B/c$, onde se admite que não existe limite para o tamanho da fila, a fonte de clientes é infinita, e a disciplina da fila é FCFS (*First Come - First Served*, primeiro a chegar, primeiro a ser servido). Assim, um sistema de fila $M/M/1$ é aquele caracterizado por apresentar tempos entre chegadas de clientes e tempo de serviço obedecendo à distribuição exponencial e possuindo um único servidor.

2.3.3. Relação Fila/Servidor em Centros de Serviços

Em (Jain, 1991), há uma descrição de algumas estruturas possíveis em um sistema de filas. Na verdade, essas estruturas estão ligadas à relação fila/servidor e ao fluxo de clientes dentro (através) do sistema.

Centros de serviços podem ter um ou vários servidores atrelados a cada uma de suas filas. A Figura 2.3 apresenta uma fila com vários servidores em um centro de serviço. A situação da figura é bastante comum nas atividades humanas cotidianas, como as filas de agências bancárias e casas lotéricas.

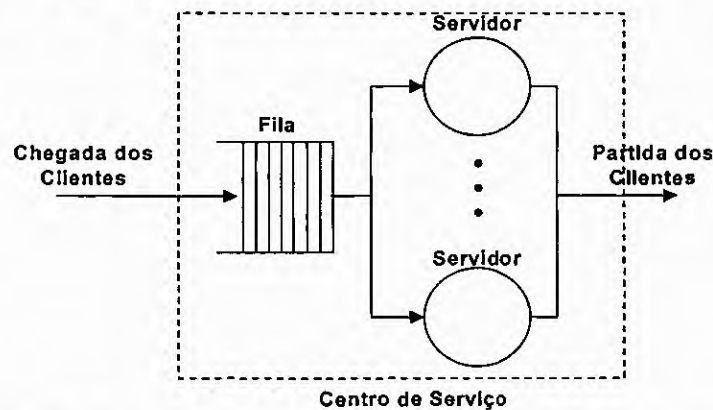


FIGURA 2.3. Centro de Serviço com múltiplos servidores.

Além do número de servidores em cada centro, pode-se levar em consideração a capacidade de cada centro de absorver seus clientes. Por esse ponto de vista, os centros podem ser *de capacidade fixa* ou *centros delay*. No primeiro caso, pela sua capacidade fixa, os clientes competem pelos recursos. Já nos centros *delay*, não há competição por existir um número muito grande de

servidores disponíveis, o que dá uma conotação de serviço infinitamente disponível. A Figura 2.4 apresenta as possíveis capacidades de centros de serviço.

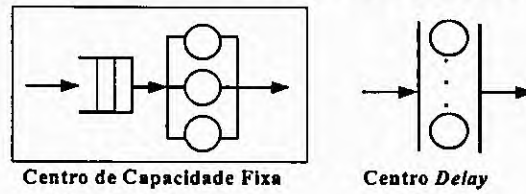


FIGURA 2.4. Tipos de Absorção de Centros de Serviço

Além da capacidade de absorver seus clientes, as redes de filas podem ser observadas de acordo com o fluxo e o número de clientes que circulam no sistema. Sob esse prisma, as redes podem ser abertas, fechadas ou mistas. As redes abertas são aquelas em que há uma geração externa de clientes e, obrigatoriamente, o número de clientes que entra no sistema é igual ao número de clientes que sai. Já nas redes fechadas, o sistema não recebe clientes externos, e sim há um número fixo de clientes circulando no sistema. Uma combinação de ambos também é possível, originando uma rede mista, que para certos clientes é aberta e para outros é fechada. A Figura 2.5 apresenta essas possibilidades.

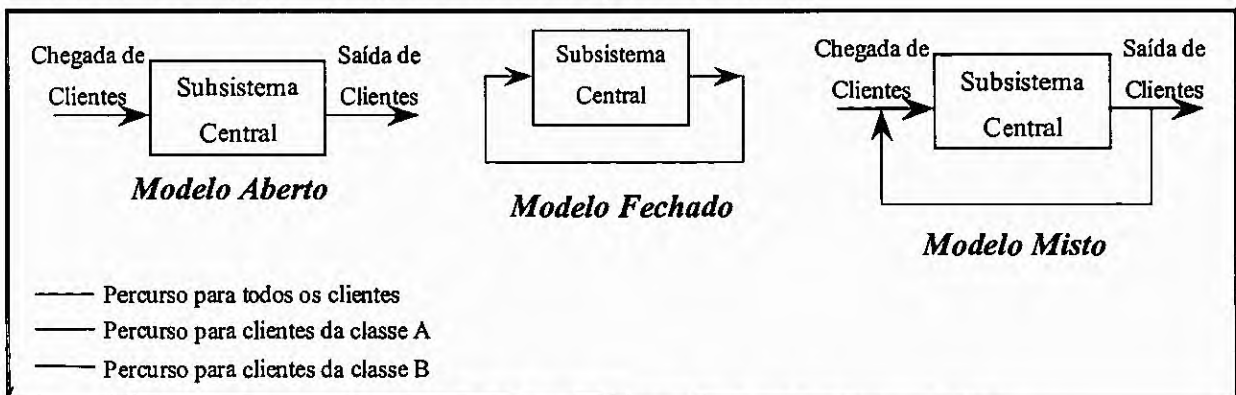


FIGURA 2.5. Redes abertas, fechadas e mistas.

Outra visão que pode se ter de um sistema de filas é a dos tipos de clientes que o mesmo pode possuir. Em sistemas reais é comum que haja classes diferentes de clientes solicitando o mesmo serviço. Processos competindo por um processador em sistemas computacionais é um exemplo típico dessa situação, pois pode haver, por exemplo, processos *batch* e interativos disputando pelo mesmo recurso. Redes que possuem classes diferentes de clientes possuem um tratamento diferenciado e são bastante discutidas em (Walrand, 1990).

2.3.4. Representação Gráfica de Redes de Filas - Discussão

As redes de filas possuem uma base matemática bastante solidificada, contudo sua representação gráfica oferece apenas os elementos fila e servidor, o que em muitos casos é suficiente. Entretanto, em alguns sistemas, desejam-se representar situações que não constituem necessariamente uma fila ou um servidor. Por exemplo, um processo em um computador pode se encontrar em três possíveis situações (ou estados): Processando, Bloqueado ou Pronto (Tanenbaum, 1995). Essas situações são estados abstratos que são facilmente representados em técnicas como redes de Petri e Statecharts, mas por não consistirem em filas ou servidores, não são tratadas apropriadamente em redes de filas. E mesmo filas e servidores necessitam de uma representação mais minuciosa, através da qual possam especificar as várias situações que tanto fila (uma fila pode estar vazia ou não) quanto servidor (um servidor pode estar livre ou ocupado) podem se encontrar em um determinado instante.

A despeito dessas limitações, a representação do caminho linear que os clientes traçam através do sistema é descrita com bastante propriedade, noção que na maioria das técnicas é perdida com facilidade.

2.4. Redes de Petri (RP)

Redes de Petri é uma técnica que permite a representação de sistemas, utilizando como alicerce uma forte base matemática. Essa técnica possui a particularidade de permitir modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos.

A representação gráfica de uma rede de Petri básica é formada por três componentes: um ativo chamado de *transição* (barra), outro passivo denominado *lugar* (círculo) e os *arcos direcionados*, que são elementos de ligação entre os dois primeiros. Os lugares equivalem às variáveis de estado e as transições correspondem às ações realizadas pelo sistema. Os arcos podem ser únicos ou múltiplos. A Figura 2.6 mostra os elementos básicos de um grafo associado às redes de Petri.

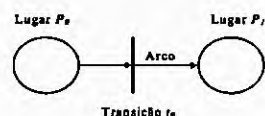


FIGURA 2.6. Elementos Básicos de uma Rede de Petri.

À primeira vista, pode parecer estranha a ausência de *tokens* (marcas) como um dos elementos básicos das RP. Entretanto, na definição formal, a seguir, será mostrado que o conjunto da marcação pode ser um conjunto vazio, o que, portanto, desobriga a rede de possuir *tokens*.

2.4.1. Definição de RP

As redes de Petri podem ser enfocadas através de três fundamentações diferentes. A primeira utiliza a teoria de *bags* como suporte. A segunda usa os conceitos da álgebra matricial. A última se fundamenta na estrutura definida por relações. A seguir é apresentada a definição formal baseada na álgebra matricial. A explicação das três definições pode ser encontrada em (Francês, 1998).

- Definição baseada na Álgebra Matricial: a estrutura de uma rede de Petri, segundo o ponto de vista matricial, é uma quintupla $R = (P, T, I, O, K)$, onde P é um conjunto finito não-vazio de lugares. T é um conjunto finito não-vazio de transições, $I : P \times T \rightarrow N$ é a matriz de pré-condições. $O : P \times T \rightarrow N$ é a matriz de pós-condições. K é o vetor das capacidades associadas aos lugares ($K : P \rightarrow N$) (Peterson, 1981). A Figura 2.7 será usada para elucidar a definição.

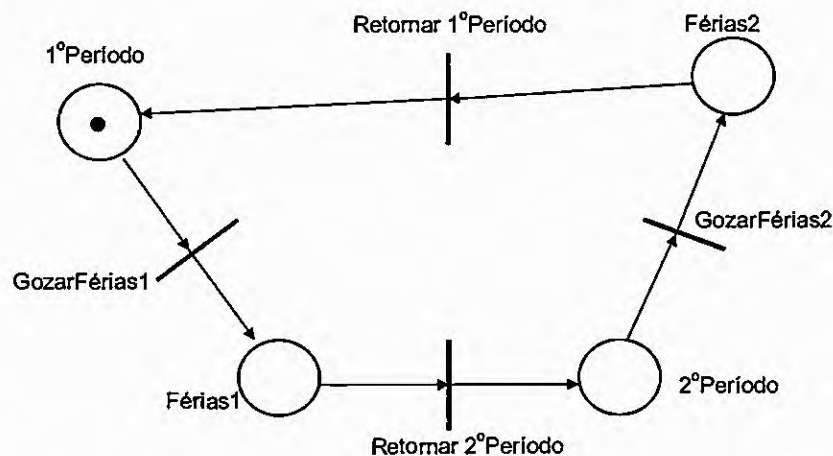


FIGURA 2.7. Ano Letivo representado em Rede de Petri.

De acordo com a definição e tomando-se como exemplo a rede da Figura 2.7, tem-se:

$$R_{\text{Ano_Letivo}} = (P, T, I, O, K), \text{ onde}$$

o conjunto de lugares P é

$$P = \{1^\circ\text{Período}, \text{Férias1}, 2^\circ\text{Período}, \text{Férias2}\};$$

o conjunto de transições T é

$T = \{\text{GozarFérias1}, \text{Retornar2}^\circ\text{Período}, \text{GozarFérias2}, \text{Retornar1}^\circ\text{Período}\};$

A matriz I (pré-condições) é

$$I = \begin{bmatrix} \text{GozarFérias1} & \text{Retornar2}^\circ\text{Período} & \text{GozarFérias2} & \text{Retornar1}^\circ\text{Período} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} 1^\circ\text{Período} \\ \text{Férias1} \\ 2^\circ\text{Período} \\ \text{Férias2} \end{array}$$

A matriz O (pós-condições) é:

$$O = \begin{bmatrix} \text{GozarFérias1} & \text{Retornar2}^\circ\text{Período} & \text{GozarFérias2} & \text{Retornar1}^\circ\text{Período} \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{array}{l} 1^\circ\text{Período} \\ \text{Férias1} \\ 2^\circ\text{Período} \\ \text{Férias2} \end{array}$$

É importante ressaltar que as matrizes I e O representam as pré e pós-condições, respectivamente, de todas as transições da rede.

O vetor K das capacidades, de acordo com o exemplo, é todo unitário, ou seja, cada lugar terá, no máximo, 1 *token*.

$$K = \begin{bmatrix} 1^\circ\text{Período} & \text{Férias1} & 2^\circ\text{Período} & \text{Férias2} \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

As redes podem ser marcadas ou não. Marcas (*tokens*) são informações atribuídas aos lugares, para representar a situação (estado) da rede em um determinado momento. Define-se uma rede de Petri marcada pela dupla $RM = (R, M_0)$, onde R é a estrutura da rede e M_0 a marcação inicial (Maciel et. al., 1996). Assim, para simular o comportamento dinâmico dos sistemas, a marcação das redes de Petri é modificada a cada ação realizada (transição disparada).

Existem algumas redes básicas, através das quais podem ser implementadas todas as demais. Essas redes são chamadas de elementares. A seguir são apresentadas as principais.

2.4.2. Redes de Petri Elementares

Nesta seção, são apresentadas algumas redes que, a partir delas, derivam muitas outras redes mais complexas. São discutidas as redes representativas de seqüenciamento, distribuição, junção e escolha não-determinística.

- Seqüenciamento: é a rede que representa a execução de uma ação, desde que uma determinada condição seja satisfeita. Após a execução dessa ação, pode-se ter outra ação, desde que satisfeita outra determinada condição (Figura 2.8).

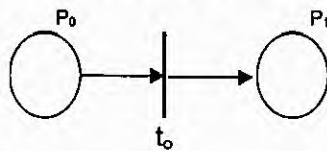


FIGURA 2.8. Seqüenciamento.

- Distribuição: é a rede elementar utilizada na criação de processos paralelos a partir de um processo pai. Os processos filhos são criados através da distribuição dos *tokens* encontrados no processo (lugar) pai. A distribuição é mostrada na Figura 2.9.

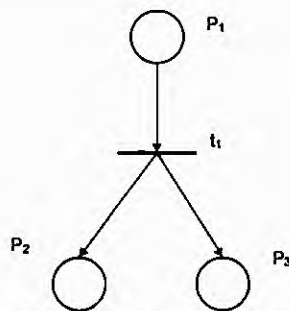


FIGURA 2.9. Distribuição.

- Junção: é a rede que modela a sincronização entre atividades concorrentes. No exemplo da Figura 2.10, a transição t_1 só dispara quando existirem fichas tanto em P_1 , quanto em P_2 , estabelecendo, assim, o sincronismo.

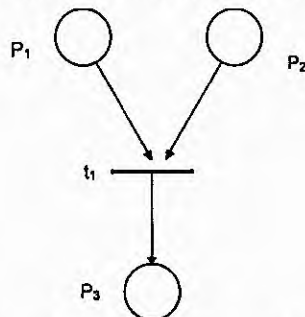


FIGURA 2.10. Junção.

- Escolha Não-Determinística: é uma rede que ao se disparar uma transição, inabilita-se a outra. Entretanto, não existe possibilidade de escolha (conforme Figura 2.11). O fator não-determinístico dessa rede gera uma situação chamada de conflito (Maciel et. al., 1996). O conflito pode ser classificado como estrutural ou efetivo. Ambos os conflitos estão associados ao fato de duas transições possuírem o mesmo lugar como entrada. Porém, se a rede não possuir *tokens*, o conflito é dito estrutural. Contudo, se há uma única marca no lugar comum às transições, diz-se que o conflito é efetivo. A Figura 2.12 ilustra os dois tipos de conflito.

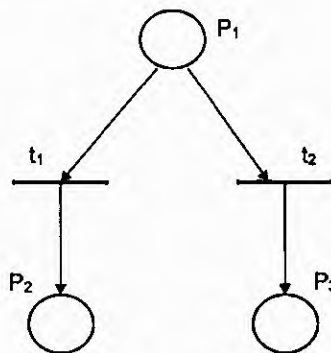


FIGURA 2.11. Escolha Não-Determinística.

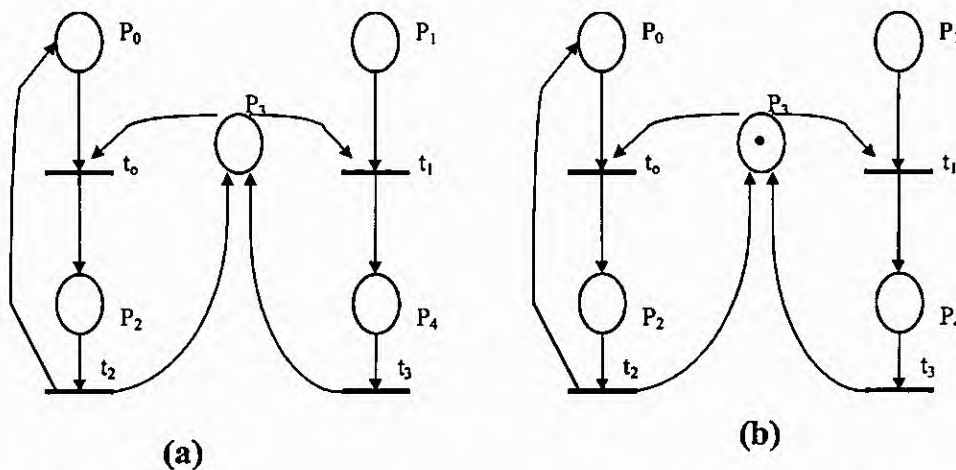


FIGURA 2.12. (a) Conflito Estrutural e (b) Conflito Efetivo.

Os modelos de rede vistos até o momento são todos de redes do tipo ordinário. As redes de Petri podem ser basicamente de dois tipos: Ordinárias ou de Alto Nível (consideradas extensões às redes de Petri). As redes de Petri ordinárias são aquelas que possuem o tipo de marca mais trivial de todos: marcas do tipo inteiro não-negativo. Já as redes de alto nível se caracterizam pelo oposto: podem possuir marcas de tipos mais complexos. Isso significa que na mesma rede podem estar contemplados vários tipos de *token*, caso seja necessário. São exemplos de

redes de alto nível as RP Predicado-Evento, as RP Coloridas e as RP Hierárquicas. A seguir, são apresentadas as redes de Petri coloridas, como exemplo de redes de alto nível. Uma discussão extensa sobre as demais redes de alto nível é apresentada em (Maciel et al., 1996).

2.4.3. Redes de Petri Coloridas

As redes de Petri coloridas têm por objetivo reduzir o tamanho do modelo, permitindo que os *tokens* sejam individualizados, através de cores atribuídas a eles. Assim, diferentes processos ou recursos podem ser representados em uma mesma rede. As cores não significam apenas cores ou padrões. Elas podem representar tipos de dados complexos, usando a nomenclatura de colorida apenas para referenciar a possibilidade de distinção entre os *tokens* (Jensen et. al., 1990). A Figura 2.13 apresenta uma rede colorida, possuindo a representação original, onde são realmente utilizadas cores para diferenciar os *tokens*. Nessa figura, os arcos são rotulados com cores (DNS, Telnet, FTP, HTTP²).

No exemplo, há um cliente TCP/IP que pode solicitar quatro tipos de serviços a quatro servidores diferentes (o servidor DNS, o servidor Telnet, o servidor FTP e o servidor HTTP). Então, as solicitações do cliente devem ser diferenciadas, de acordo com o servidor que vai atendê-las. A associação de cores e marcas individualizadas para cada requisição e as restrições colocadas nos arcos determinam quais marcas podem atravessar esses arcos. Na Figura 2.13, utiliza-se o modo mais elementar de redes coloridas, no qual se associa ao arco uma determinada cor. Assim, o *token* se destinará ao arco cuja cor for idêntica à da marca. Desse modo, pode-se perceber que os *tokens* de "Cliente" não habilitarão as transições t_1 e t_2 , pois os arcos que ligam Cliente às transições t_1 e t_2 só aceitam cores do tipo Telnet e FTP, e o lugar Cliente só possui marcas do tipo DNS e HTTP. O que significa dizer que o cliente só pode solicitar serviços de DNS e HTTP.

Ainda que rudimentar, a rede colorida original provê mecanismos que possibilitam efetuar uma escolha determinística. Esse poder de escolha já significa um grande avanço em direção a uma representação mais clara de um modelo. Porém, modificações (acréscimos) posteriores vieram dar maior adequação às redes coloridas, com relação à representação das escolhas não-determinísticas.

² DNS (Domain Name System); FTP (File Transfer Protocol); HTTP (HyperText Transfer Protocol), (Tanenbaum, 1996).

As redes de Petri coloridas, como as usadas atualmente, não acrescentam apenas uma diferenciação gráfica das marcas e aplicações de restrições aos arcos. Elas possuem mecanismos muito mais sofisticados para tornar um modelo mais conciso e mais claro.

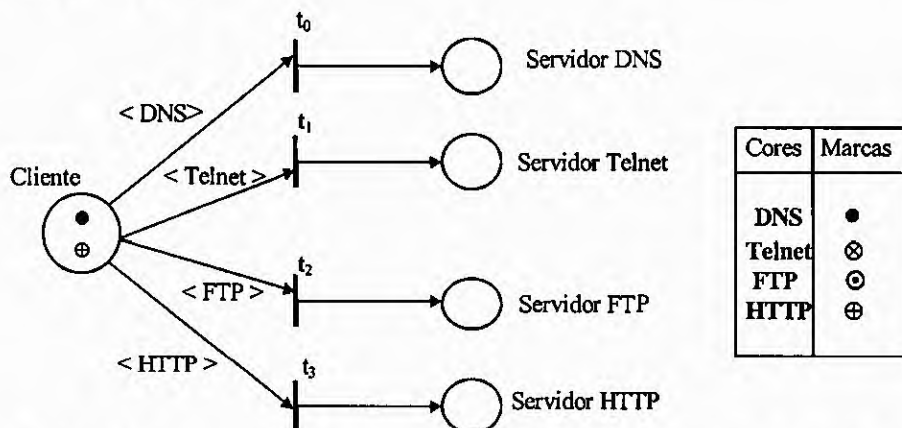


FIGURA 2.13. Rede de Petri Colorida de um Cliente TCP/IP.

As redes de Petri coloridas atuais são compostas por três partes diferentes (Maciel et. al., 1996):

- Estrutura,
- Declarações,
- Inscrições.

Estrutura é um grafo dirigido com dois tipos de vértices (lugares e transições). Os lugares são representados graficamente por círculos (ou por elipses) e as transições por retângulos. Essa representação herda a propriedade das redes coloridas originais de poder armazenar em cada lugar marcas de tipos diferentes, além de poder representar valores associados a tipos de dados mais complexos. *Declarações* compreendem a especificação dos conjuntos de cores e declarações de variáveis. As *inscrições* variam de acordo com o componente da rede. Os lugares possuem três tipos de inscrições: nomes, conjunto de cores e expressão de iniciação (marcação inicial). As transições têm dois tipos de inscrições: nomes e expressões-guarda, e os arcos apenas um tipo de inscrição dado pela expressão. Como formas para distinguir as inscrições, nomes são escritos com letras normais, cores em itálico, expressões de iniciação sublinhadas e as expressões-guarda são colocadas entre colchetes. Nomes, quando associados a lugares, não têm significado formal, apenas facilitam a identificação. As expressões-guarda

associadas às transições são expressões *booleanas* que devem ser atendidas para que seja possível o disparo das transições.

Como exemplo comparativo, a Figura 2.14 apresenta um sistema de manufatura em RP ordinária que possui dois tipos de processos (p e q), sendo que cada processo pode estar em um determinado número de estados locais. Para o processo p ($sp_0, sp_1, sp_2, sp_3, sp_4$), e ($sq_0, sq_1, sq_2, sq_3, sq_4$) para o processo q . Têm-se uma instância do processo q e duas do processo p (representados pelas marcas nos lugares sq_0 e sp_0 , respectivamente). As marcas dos lugares r e s representam os recursos que devem ser compartilhados pelos processos. É importante ressaltar que se houvesse mais processos, seria necessário descrever novamente todos os estados locais desses novos processos, assim como as ações e relações com os recursos.

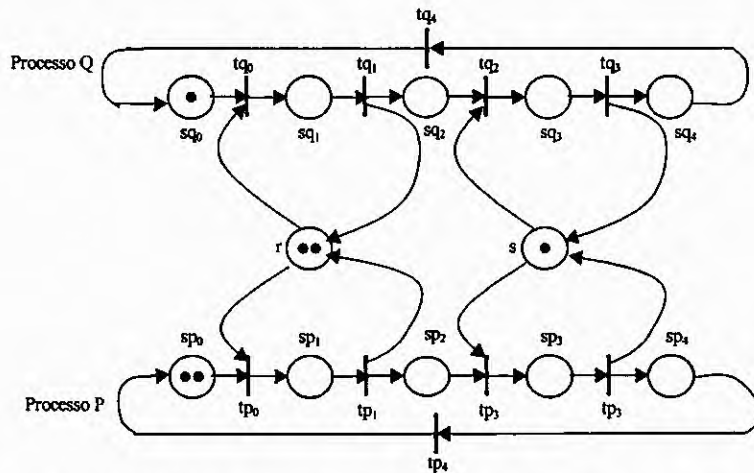


FIGURA 2.14. Linha de Manufatura em Redes Ordinárias.

A Figura 2.15 apresenta o mesmo modelo de 2.14, agora em RP coloridas. A rede de 2.15 é formada por sete lugares, sendo que s_0 se refere ao processo p , e s_1 se refere ao processo q . O restante dos lugares é comum aos dois processos. Todos os lugares são da cor $Proc^*l$, onde $Proc = p|q$ e $l = Int$. Os recursos são da cor $Rec = r|s$. As funções de iniciação dos lugares podem ser de dois tipos: $2'(p,0)$ e $1'(q,0)$, indicando o número de marcas no lugar, além do número de iterações já ocorridas; ou $2'r + 1's$, indicando apenas a marcação inicial e o tipo dos *tokens* presente no lugar.

Em 2.15, duas transições podem ser disparadas, tanto t_0 quanto t_1 . Se t_0 for disparada, o tipo de processo a ser tratado é o processo do tipo p . Já se t_1 é que for disparada, o tipo de processo será q . Supondo que a primeira hipótese seja verdadeira, a expressão do arco a ser avaliada será $p \Rightarrow 1'r$, significando que um

recurso do tipo r será disponibilizado para o processo p . A seqüência continua até que o processo p termine de realizar seus estados locais e incremente o contador i , indicando que haverá outro processo na linha de manufatura.

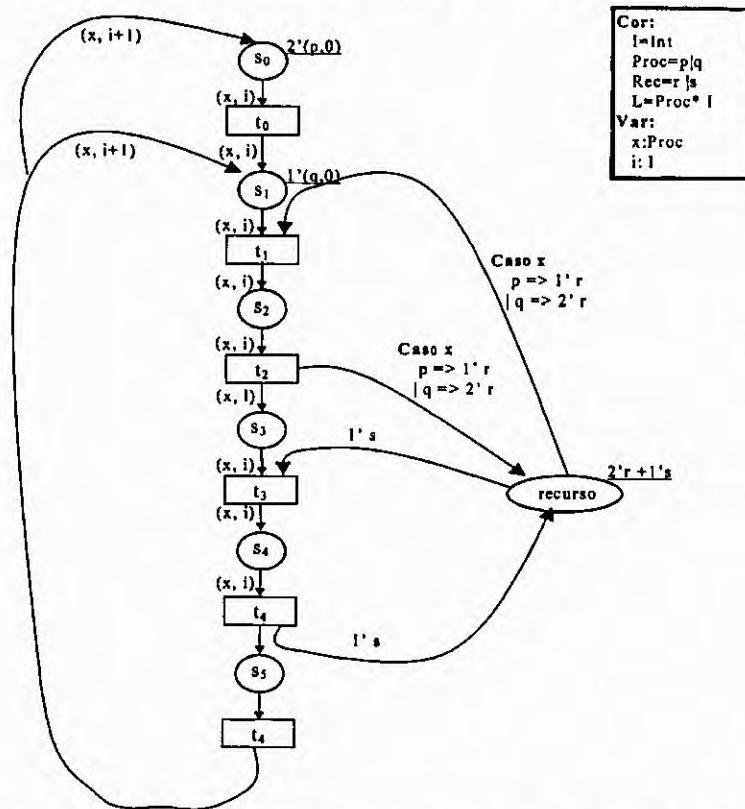


FIGURA 2.15. Linha de Manufatura em Redes Coloridas.

A diferença entre os modelos de 2.14 e 2.15 está em utilizar-se (em RP coloridas) a mesma estrutura para números de processos distintos, apenas os diferenciando através dos vários tipos de *tokens* e expressões que otimizam a criação do modelo, sem perder o poder semântico.

As redes de Petri coloridas e suas peculiaridades são discutidas com bastante abrangência em (Jensen, 1992).

2.4.4. Redes de Petri Estocásticas.

As redes de Petri, ao contrário das redes de filas, não foram desenvolvidas originalmente para prover avaliação de desempenho, apesar de toda a sua potencialidade para representar sistemas complexos, os quais naturalmente requerem cuidados a esse respeito. Esse panorama se modificou em 1982, quando M. K. Molloy (Molloy, 1982) apresentou as redes de Petri estocásticas (*Stochastic Petri Nets - SPN*) como uma técnica capaz de, além de especificar sistemas, também apresentar uma análise probabilística dos mesmos.

Molloy definiu que todas as transições em uma SPN eram temporizadas (*timed*), e que possuíam um retardo exponencialmente distribuído. Através dessa implicação, as SPN seriam isomórficas às cadeias de Markov, e assim poderiam prover medidas de desempenho. As cadeias de Markov são apresentadas no Capítulo 3, como exemplo de processos estocásticos.

Posteriormente, G. Chiola (Chiola et. al. 1993) apresentou uma melhoria às SPN, denominada Redes de Petri Estocásticas Generalizadas (*Generalized Stochastic Petri Nets - GSPN*), cuja diferença fundamental está em admitir que as transições também podem ser não-estocásticas, isto é, uma transição também pode ser imediata, como nas RP convencionais. Chiola definiu que as transições imediatas deveriam ter retardo de disparo igual a zero, e que somente as transições estocásticas tinham retardos associados diferentes de zero.

2.4.5. Representação Gráfica de Redes de Petri - Discussão

Redes de Petri possuem características bastante interessantes em sua representação. Uma das mais interessantes é a possibilidade da individualização de clientes, recurso que não é usual na maioria das técnicas. Há situações em que essa característica é primordial, por exemplo, na migração de processos para balanceamento de carga entre máquinas, onde se pode desejar migrar exatamente um determinado processo e, por isso, deve-se saber exatamente onde ele se encontra.

Entretanto, há uma certa dificuldade na representação de processos paralelos, mesmo utilizando-se da rede distribuição. Esse aspecto piora à medida que o modelo cresce, o que aliás é outro aspecto a ser ponderado. As redes originais, por não possuírem nenhum mecanismo de hierarquia, tendem a fazer com que os modelos cresçam substancialmente, quando a complexidade dos mesmos aumenta. Há algumas extensões que tentam minimizar esse efeito, como as RP Hierárquicas, que são baseadas em um elemento denominado superpágina (Maciel et al., 1996). O inconveniente dessa extensão é que as superpáginas são caixas-pretas, ou seja, são retângulos que escondem uma complexidade que, em certos casos, é necessária para a compreensão do modelo. Além disso, algumas extensões desfiguram a notação original, o que, às vezes, parece ser outra técnica à parte e não uma derivação das redes de Petri.

2.5. Statecharts (SC)

A abordagem sugerida nesta seção é baseada em (Harel, 1987) e (Harel & Politi, 1998), cujo teor é mais informal, com um caráter mais expositivo. Semântica e sintaxe formais, como apresentadas em (Harel et al., 1987), são discutidas no decorrer do Capítulo 4.

Statecharts, assim como redes de Petri, é uma técnica de representação de sistemas através da visão de seus estados e a modificação deles em consequência à ocorrência de uma determinada interferência.

Statecharts é uma extensão do diagrama tradicional de estados-transições, aos quais foram adicionadas algumas características peculiares. Os principais conceitos adicionados são hierarquia entre estados (*depth*), ortogonalidade (representação de atividades paralelas) e interdependência entre estados (mecanismos de comunicação). Além disso, Statecharts são fundamentados nos seguintes elementos básicos: Estados, Eventos, Condições, Ações, Expressões, Variáveis, Rótulos e Transições. Sucintamente, cada elemento básico é apresentado a seguir.

Estados são usados para descrever componentes de um determinado sistema (e suas possíveis situações). Por exemplo, um processador ocioso à espera de um processo pode se encontrar no estado “Esperando” (W). À chegada de um processo, o processador passa a tratar esse processo, mudando para um estado “Processando” (P). Além disso, o processador pode falhar no meio do processamento (estado de “Falha” (F)), conforme apresentado na Figura 2.16 (Vijaykumar, 1999).

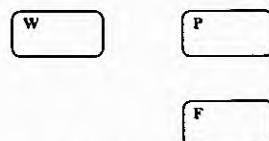


FIGURA 2.16. Estados W, P e F.

Os estados de um *statechart* (que representam os valores das variáveis do sistema em um determinado instante) podem ser classificados em dois grupos: básicos e não-básicos. Os *estados básicos* são aqueles que não possuem subestados. Já os *não-básicos* são decompostos em subestados. Essa decomposição pode ser de dois tipos: OR ou AND. Se a decomposição é do tipo OR, então o sistema sempre estará em um único subestado em um certo instante.

Entretanto, se a decomposição é do tipo AND, o estado estará em mais de um subestado, simultaneamente.

A Figura 2.17 apresenta as formas de representação de estados em Statecharts. À esquerda, um estado X do tipo OR e seus subestados (X_1 , X_2 , X_3), indicando que o sistema não estará em mais de um subestado ao mesmo tempo (subestados disjuntos). À direita, um estado Y do tipo AND e seus subestados (Y_1 , Y_2 , Y_3), indicando que o sistema estará em mais de um subestado simultaneamente. A concorrência de estados (ou subestados) é representada através de linhas tracejadas. Além da concorrência, a figura em questão ainda introduz a idéia de hierarquia entre os estados, com subestados contidos em um estado denominado estado-pai (no exemplo, X e Y). Reciprocamente, um estado-pai contém estados-filho (subestados) dentro de si.

Eventos são considerados a entidade que causa uma interferência no comportamento atual do sistema, levando esse sistema a outro comportamento. No exemplo do processador, a chegada de um processo é um evento, pois leva o sistema do estado W para o estado P. O final do processamento também é outro evento, pois leva o processador do estado P novamente para W. Opcionalmente, a um evento pode ser anexada uma condição (entre parênteses), também chamada de condição-guarda, de maneira que o evento só ocorrerá se satisfeita aquela determinada condição. Os Statecharts proporcionam alguns eventos especiais como *true* (condição), *false* (condição), *entered* (X) e *exited* (X), abreviados na notação Statecharts para *tr* (condição), *fs* (condição), *en* (X) e *ex* (X), respectivamente.

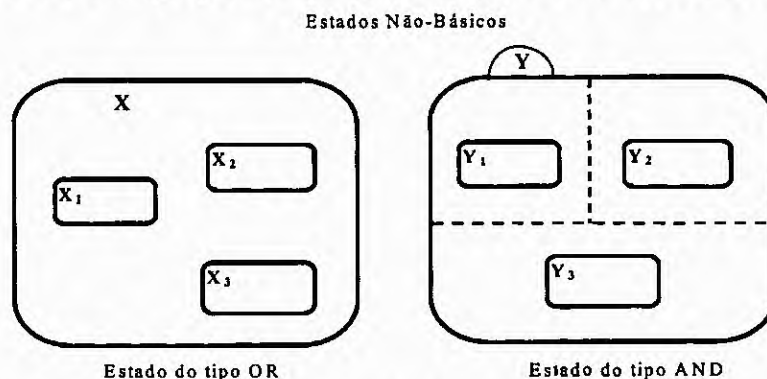


FIGURA 2.17. Representação de Estados em Statecharts.

O elemento ação é considerado para representar os efeitos do paralelismo em Statecharts (a influência de um estado paralelo em outro, também ortogonal). Ações podem ser uma mudança de uma expressão, uma mudança de uma variável ou eventos que são disparados em outros componentes paralelos.

As transições são a representação gráfica para denotar uma mudança de estado dentro do sistema. Rótulos podem ser acrescentados às setas para prover algum significado adicional. Ainda no exemplo do processador, a transição a pode representar a passagem do estado “Esperando” para o estado “Processando”, enquanto que s representa o oposto (Figura 2.18). A sintaxe completa de um rótulo de transições é $ev [c] / a$, onde o evento ev só é disparado se a condição c for satisfeita. E se isso ocorre, a ação a é tomada em algum outro componente (estado) do sistema.

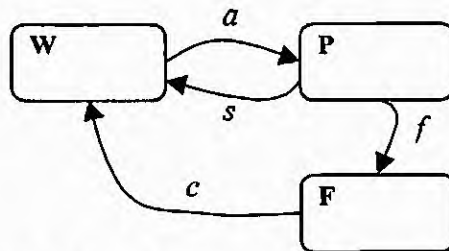


FIGURA 2.18. Estados, Eventos e Transições.

2.5.1. Paralelismo e Hierarquia de Componentes

A Figura 2.19 apresenta uma especificação de um sistema hipotético, onde A é um estado do tipo AND, com dois componentes paralelos (A_1 e A_2), que por sua vez também possuem subcomponentes (por exemplo, C e D em A_1).

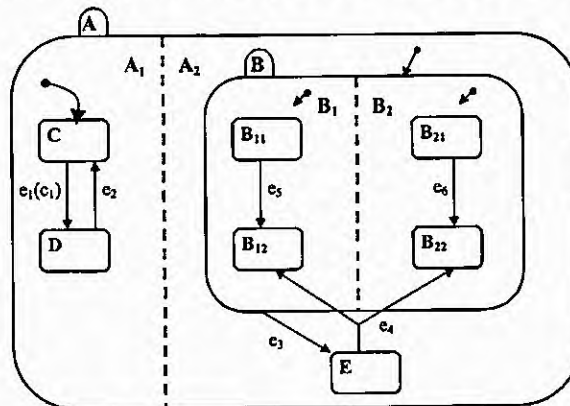


FIGURA 2.19. Estados Ortogonais.

Além de tratar o paralelismo de forma explícita, os Statecharts permitem um tipo de representação não linear entre diferentes níveis de abstração no mesmo modelo, isto é, a hierarquização, através de seus subestados, permite que se tenha um maior nível de detalhes, de acordo com a necessidade.

Além dos elementos básicos citados anteriormente, os Statecharts possuem vários outros recursos que ampliam o seu poder de especificação. Alguns desses recursos são apresentados a seguir.

2.5.2. Entrada por Condição

Os Statecharts permitem a especificação da escolha de um estado dentre vários, através de uma condição preestabelecida. A Figura 2.20 (b) (Harel, 1987) apresenta uma entrada por condição alternativa à notação tradicional dos Statecharts (2.20 (a)). A condição hipotética C pode assumir três valores distintos (R, Q e P). Dependendo do valor assumido por C, será habilitado apenas um dos três arcos existentes.

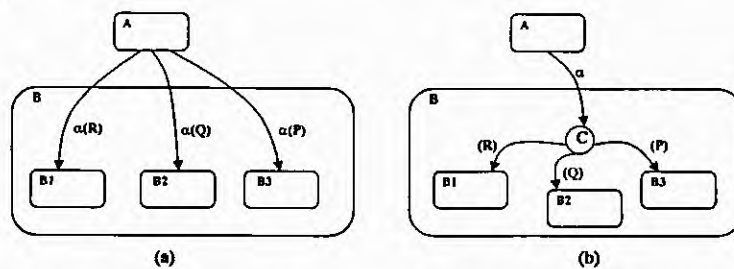


FIGURA 2.20. (a) Múltiplas Entradas com Condições (b) Entrada por Condição.

2.5.3. Estados Parametrizados

Em certos casos, diversos estados possuem a mesma estrutura interna. Para evitar a repetição de vários estados com estrutura similar, podem-se usar estados parametrizados. Por exemplo, a Figura 2.21 apresenta um sistema com dezesseis conjuntos processador/disco (*processor/disc*), utilizando a especificação de estados AND parametrizados.

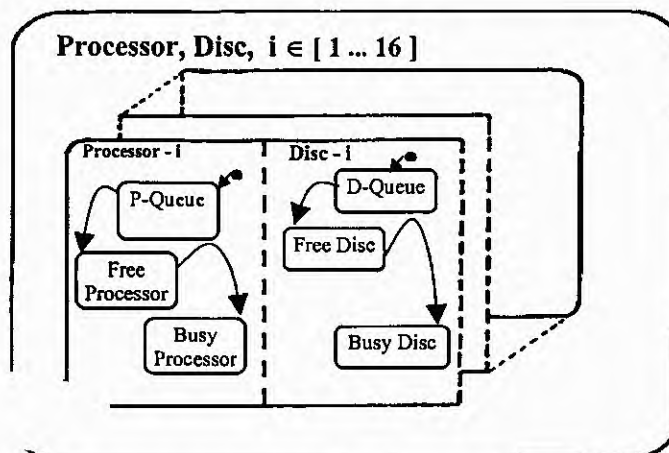


FIGURA 2.21. Estado Parametrizado *Processor/Disc*.

2.5.4. Delay e Timeout

Em muitas situações é interessante introduzir a idéia de tempo de espera, após a ocorrência de um determinado evento (como se o estado tivesse um temporizador implícito). Essa situação também pode ser vista como um *delay* associado a um estado, já que, em sua especificação original, não há associação de tempo de permanência em cada estado (posteriormente, será apresentada a idéia de tempos exponencialmente distribuídos em estados). A expressão de *timeout* é *(evento, número)*, onde o sistema só irá transicionar para outro estado após decorrido o número de unidades de tempo (*número*) da ocorrência do evento especificado na expressão do *timeout*. A notação para o *delay* é mostrada na Figura 2.22, onde se deve esperar 5 segundos para realizar um determinado evento.

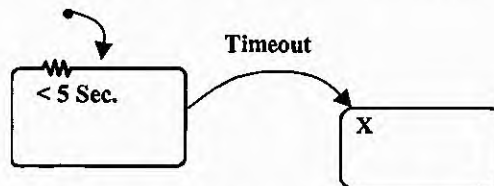


FIGURA 2.22. Estado com *Delay* de 5 segundos, extraída de (Harel, 1987).

2.5.5. Representação Gráfica dos Statecharts - Discussão

Na representação Statecharts, a forma explícita de algumas características (como o paralelismo e a hierarquia entre estados) é um atrativo considerável. Essa peculiaridade permite que relacionamentos complexos entre componentes de um determinado sistema sejam mostrados de maneira mais efetiva, o que não é contemplado na maioria das técnicas. Estados *default* também acrescentam a idéia de passo inicial do sistema, o que nas redes de Petri é realizado através da presença dos *tokens* (marcação inicial).

Entretanto, uma possibilidade que os Statecharts não contemplam é, quando há a presença de estados ortogonais, o caminho linear seguido por um determinado cliente (que é claro em RF e RP), pois pode haver vários subestados ativos em um determinado momento, o que não significa que todos os clientes (ou um cliente em particular) tenham seguido por aquela trajetória. Além disso, a individualização de clientes (como feito com a utilização dos *tokens*), fundamental em certas situações, não é implementada em Statecharts.

2.6. Soluções para o Modelo

Após proceder-se à escolha da técnica utilizada para a representação do modelo, deve-se decidir qual a resolução a ser dada ao mesmo. Existem, basicamente, duas técnicas de solução disponíveis: a analítica e a simulação. As duas técnicas têm suas vantagens e desvantagens, que normalmente são os fatores determinantes na escolha de uma em detrimento da outra.

2.6.1. Solução Analítica

Kleinrock, em (Kleinrock, 1976), afirma que o método analítico é geralmente mais rápido e, por isso, o preferido; porém, nem sempre ele é aplicável. Uma solução analítica nem sempre é aplicável em virtude de, geralmente, ser necessário fazer um certo número de simplificações para poder-se resolver o modelo. Deve-se entender por simplificações algumas restrições impostas pela solução analítica, que às vezes não correspondem às situações verificadas nos sistemas reais. Apesar de ser um método que propicie resultados expressivamente exatos, as simplificações introduzidas podem fazer com que o modelo não seja uma representação fiel do sistema real, fato que pode simplesmente tornar sem sentido a utilização do modelo.

Para exemplificar a dificuldade imposta por certas restrições, na solução analítica de modelos baseados em redes de filas não se podem estabelecer prioridades para as disciplinas de filas (Soares, 1992), isto é, todos os usuários de um determinado recurso possuem a mesma prioridade, o que inviabilizaria, por exemplo, a representação do esquema de prioridades atribuídas aos processos no sistema UNIX. Ainda há algumas outras restrições impostas à solução analítica. A assunção de certas distribuições de probabilidade (por exemplo, a exponencial) que possuem um tratamento matemático mais factível, em certas situações, pode ser um fator limitante.

Nas situações em que várias simplificações (restrições) comprometem a exatidão dos resultados do modelo, pode ser mais conveniente adotar-se a solução por simulação.

Há vários métodos destinados a prover solução analítica ao modelo, sendo que esses métodos, via de regra, levam em consideração se o sistema é aberto ou fechado. Alguns desses métodos são sintetizados nos Capítulos 5, 6 e no Apêndice C.

2.6.2. Solução por Simulação

Se o modelo proposto para o sistema envolve uma grande gama de informações e/ou exige que não se façam algumas das simplificações vistas na seção anterior, o modelador pode optar por resolver o modelo através de simulação.

Em computação, a simulação se refere ao emprego de um processo computacional para implementar um modelo de algum fenômeno ou sistema dinâmico (sistemas cujos estados se alteram com o tempo) (Prado, 1999).

A simulação tem sido muito aceita e empregada por alguns fatores que lhe são, de certa maneira, favoráveis (por exemplo, flexibilidade, relativa facilidade de utilização e custo relativamente baixo). As utilizações da simulação são as mais diversas possíveis, destacando-se, por exemplo, as aplicações médicas, em engenharia aeronáutica, em engenharia de transportes, em engenharia nuclear e em computação.

As fases, assim como o tratamento estatístico dado à saída da simulação são abordados no capítulo das soluções para o modelo (Capítulo 5). Naquele capítulo, também será introduzida uma discussão a respeito da controvérsia que incita a escolha de uma abordagem para solução. Não sendo um assunto consensual, há duas correntes, cada qual com suas justificativas peculiares, ambas abordadas com o máximo de isenção possível (Capítulo 5).

A Figura 2.23 apresenta o inter-relacionamento entre as duas soluções de um modelo, assim como as variantes que determinam quando, preferencialmente, utilizar uma solução em detrimento da outra.

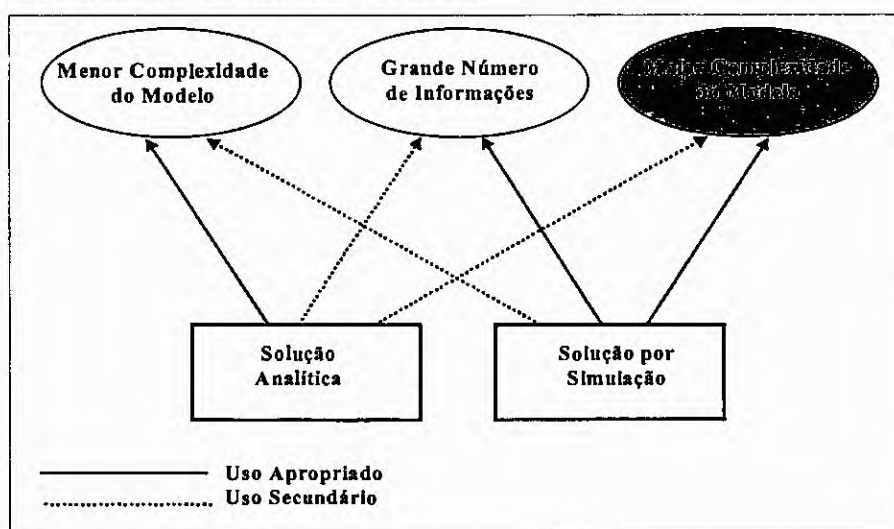


FIGURA 2.23. Solução Preferencial para um Modelo.

2.7. Considerações Finais

Este capítulo apresentou uma visão geral do processo de modelagem, abordando algumas nuances do tema. Sem pretender esgotar o assunto, foram discutidas algumas características interessante de três técnicas amplamente difundidas na literatura especializada: redes de filas, redes de Petri e Statecharts. Também foram abordadas algumas possíveis melhorias que poderiam ser acrescentadas a cada uma delas, visando a resolver problemas específicos ("limitações").

Além do aspecto de realizar uma revisão bibliográfica sobre o tema, o capítulo deixa em aberto o debate sobre a falta de utilização dos Statecharts para avaliação de desempenho, mesmo tendo essa técnica sido idealizada para especificar sistemas complexos reativos, nos quais, via de regra, a avaliação de desempenho é uma tarefa indispensável. Essa aproximação da especificação Statecharts em direção à área de desempenho é uma das contribuições deste trabalho, no sentido de poder tornar factível o uso de uma representação gráfica bastante genérica (como a de Statecharts), associada a um método de solução para o modelo, oferecendo uma alternativa viável e com características peculiares de interesse à área de avaliação de desempenho.

A solução do modelo também foi apresentada com certa superficialidade. Por ser um assunto eminentemente controverso, ele será retomado posteriormente (Capítulo 5), em mais detalhes, com o intuito de explorar as características de cada abordagem e estabelecer uma discussão preliminar sobre o tema.

Capítulo 3

Probabilidade, Estatística e Processos Estocásticos em Avaliação de Desempenho

3.1. Considerações Iniciais

No decorrer do texto deste trabalho, os fenômenos estudados possuem algum grau de incerteza associado a eles. Por esse motivo, é necessário que se processe uma discussão, mesmo que sucinta, sobre o caráter aleatório desses fenômenos e seus fundamentos matemáticos.

Com o intuito de prover um embasamento elementar para o entendimento do conteúdo deste trabalho, e sem ter-se a pretensão de esgotar o assunto, são apresentados alguns conceitos de estatística, probabilidade e processos estocásticos (sendo que é dada uma maior ênfase a um tipo particular de processos estocásticos, denominado de cadeias de Markov). Ao final do capítulo, é apresentada uma introdução à teoria de filas, através da qual se deseja criar uma associação com a extensão à especificação Statecharts (Capítulo 4).

Vale ressaltar que a fundamentação apresentada aqui será usada também no capítulo de soluções para os modelos, tanto com a abordagem analítica quanto com a por simulação.

3.2. Coleta e Análise de Dados

Um estudo de um determinado fenômeno sob a luz da estatística requer a realização de um passo inicial denominado de aquisição de dados. Essa aquisição pode ser feita sobre todas as possibilidades de um certo fenômeno ou em cima de uma parte dessas possibilidades. Assim, uma aquisição de dados pode ser realizada sobre dois conjuntos de observações definidos como segue:

- **População:** é a coleção de todas as observações potenciais sobre determinado fenômeno.
- **Amostra:** é um subconjunto da população que deve ser efetivamente observado, por poder, em algum determinado aspecto, representar, proporcionalmente, a população como um todo.

Uma observação sobre a população é mais precisa, entretanto tende a ser, dependendo da dimensão de população, mais dispendiosa e laboriosa. Por exemplo, é o caso dos grandes censos realizados pelos governos dos países sobre seus habitantes.

Não obstante, pode-se trabalhar com uma parcela da população (denominada de amostra), a qual representa o todo, em uma escala reduzida, tal qual é feito nas pesquisas de intenção de voto, em períodos eleitorais.

Todavia, em muitas situações, é preciso que algumas decisões sejam tomadas a partir dos dados da população ou da amostra. Esse tipo de tomada de decisão se constitui o problema central da *inferência estatística*, sendo que a cada decisão está associado também um grau de incerteza. É o caso dos analistas financeiros que estudam dados sobre a situação da economia, procurando gerar tendências do mercado para o futuro.

Pela imprecisão intrínseca aos resultados, a inferência está intimamente relacionada à utilização de probabilidade (mais discutida na seção 3.4) em seus modelos estatísticos, utilizando um número ou uma função matemática para quantificar o grau de incerteza de uma determinada situação.

3.3. Medidas Estatísticas

Esta seção apresenta dois tipos fundamentais de medidas estatísticas: as medidas de *locação* ou de *tendência central* e as medidas de *dispersão* ou de *variabilidade*.

As medidas de locação são aquelas que mostram uma tendência, valor representativo, em torno do qual os dados tendem a agruparem-se, sintetizando em um número um conjunto de dados observado. Já as medidas de dispersão mostram o grau de afastamento dos valores observados em relação àquele valor representativo (tendência central).

3.3.1. Medidas de Locação

- Média Aritmética Simples: em um conjunto de n observações x_1, x_2, \dots, x_n , é o quociente da divisão por n da soma dos valores dessas observações (\bar{x}):

$$\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n} = \frac{\sum x_i}{n}$$

- Média Aritmética Ponderada: em um conjunto de n observações x_1, x_2, \dots, x_n , com pesos p_1, p_2, \dots, p_n , representada por \bar{x}_p é definida como:

$$\bar{x}_p = \frac{x_1 p_1 + x_2 p_2 + \dots + x_n p_n}{p_1 + p_2 + \dots + p_n}$$

- Mediana: em um conjunto de n observações x_1, x_2, \dots, x_n , é o valor do meio do conjunto, quando os dados estão dispostos em ordem crescente. Se n é ímpar, esse valor é único; se n é par, a mediana é a média aritmética simples entre os dois valores centrais (\tilde{x}):

Exemplo: 2,7 2,8 2,9 3,0 3,1 3,1 3,2 3,3 3,4 3,5 (seqüência de dez valores).

$$\tilde{x} = \frac{3,1 + 3,1}{2} = 3,1$$

- Média de dados agrupados: se os dados estão dispostos em classes, com um determinado número n_i em uma classe C_i com ponto médio x_i , então a média é:

$$\bar{x} = \frac{\sum x_i n_i}{\sum n_i}$$

- Moda: é o valor com maior freqüência em um conjunto de observações individuais. Para os casos de dados agrupados em classes, moda é o ponto médio da classe que possuir a maior freqüência. Essa classe é denominada de classe modal.

3.3.2. Medidas de Dispersão

- Amplitude: é definida como a diferença entre o maior e o menor valor do conjunto. Via de regra, essa medida não é muito significativa pelo fato de levar em consideração apenas dois valores do conjunto, desprezando os demais.
- Variância: variância é a soma dos quadrados dos desvios em relação à média de um conjunto de dados, denotada por s^2 e definida como:

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n}$$

- Desvio Padrão: é a raiz quadrada da variância, definida como:

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$$

- Variância e Desvio Padrão para dados agrupados: análogos à média, variância e desvio padrão de dados agrupados em classes são definidos por:

$$s^2 = \frac{\sum x_i f_i}{n} - \bar{x}^2 \quad s = \sqrt{s^2}$$

- Coefficiente de Variação: é uma medida relativa de variabilidade que compara a média ao desvio padrão, cuja utilidade está em permitir a comparação das variabilidades entre diferentes conjuntos de dados, através da relação entre duas medidas. O coeficiente de variação é dado por:

$$cv = \frac{s}{\bar{x}}$$

3.4. Probabilidade

Esta seção se destina a apresentar uma medida que possa exprimir a incerteza que é inerente a determinadas situações, em termos de uma escala que varia do impossível ao certo (Soares et al., 1991). Essa medida é denominada de probabilidade. Antes, porém, de definir-se mais apropriadamente probabilidade, são necessários alguns conceitos preliminares, expostos a seguir:

3.4.1. Conceitos Preliminares

- Um *experimento aleatório* é o processo de coleta de dados relativos a um fenômeno que possui uma variabilidade em seus resultados.
- *Espaço amostral* é o conjunto de todos os resultados possíveis de um experimento. Um espaço amostral é dito discreto se ele é composto por um número finito ou infinito numerável de eventos. Já se os eventos são números reais de um determinado intervalo, então o espaço amostral é chamado de contínuo.
- *Evento* é um subconjunto do espaço amostral. Se esse subconjunto é unitário, então chama-se de evento simples.

3.4.2. Definições de Probabilidade

Há duas definições de probabilidade bastante referenciadas na literatura: a definição clássica e a definição freqüentista (Dantas, 1997).

- Definição Clássica: considerando-se um espaço amostral S com N eventos simples, que supõem-se igualmente possíveis. Seja A um evento de S composto por m eventos simples. A probabilidade de A , denotada por $P(A)$, é definida por:

$P(A) = \frac{m}{N}$, correntemente citada como “número de casos favoráveis pelo número de casos possíveis”

É interessante observar-se que a definição clássica só se aplica a espaços amostrais nos quais os eventos simples são igualmente possíveis, tal qual acontece na maioria dos jogos de azar. Entretanto, se esses jogos são repetidos inúmeras vezes, deve-se levar em consideração a probabilidade como a freqüência relativa (proporção) que um evento ocorre em uma série suficientemente grande de realizações de um experimento, em condições idênticas. Daí, então, o surgimento de uma definição baseada em uma interpretação freqüencial.

- Definição Freqüentista: Seja $n(A)$ o número de vezes que o evento A ocorreu nas n repetições do experimento. A razão $f_{n,A} = \frac{n(A)}{n}$ é denominada freqüência relativa de A nas n repetições do experimento.

3.4.3. Probabilidades Condicionais e Eventos Independentes

Existem diversas situações em que se tratam dois ou mais eventos cujas ocorrências estão, de alguma forma, interligadas. Assim, se é sabido que um evento B ocorre, então a probabilidade de um evento A ocorrer condicionado a B é:

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

Em contrapartida, há situações em que o conhecimento da ocorrência de um determinado evento B não influencia em nada a ocorrência de um certo evento A . Assim, é razoável definir-se dois eventos A e B , como sendo independentes se:

$P(A | B) = P(A)$, e usando-se a fórmula da probabilidade condicional anterior, então $P(A \cap B) = P(A) P(B)$.

Por exemplo, no teste de vida de um determinado componente eletrônico, é razoável supor que o tempo de vida de uma válvula (evento A) independe do tempo de vida dos demais componentes (evento B).

3.4.4. Variáveis Aleatórias (v.a.) e Distribuições de Probabilidades

Determinados experimentos possuem seus espaços amostrais cujos elementos são números, tal como o conjunto das possibilidades em um lançamento de um dado $S = \{1, 2, 3, 4, 5, 6\}$. Entretanto, há outros espaços amostrais que não são representados de forma natural por elementos numéricos, tal como o experimento de um lançamento de um foguete, cujos resultados possíveis são “sucesso” e “insucesso”. Contudo, a despeito da impossibilidade de estabelecer-se um espaço amostral numérico, é conveniente que se tenham números associados aos resultados. A essa associação entre um resultado e um número se denomina *variável aleatória* (v.a.).

Na verdade, *uma variável aleatória é uma função que associa, a cada ponto do espaço amostral, um número (geralmente um número real)*, sendo que tradicionalmente se usa a notação de uma única letra para representar a função v.a., tal como X , em lugar de $X(s)$, o que corresponde ao emprego de y no lugar de $f(x)$, para uma função no cálculo (Clarke & Disney, 1985). Portanto, se acaso X especifica as alturas de um conjunto de cidadãos e um determinado “cidadão Kane possui a altura de 1,78m”, então $X(\text{cidadão Kane}) = 1,78\text{m}$.

Se, porém, a cada valor de v.a. associar-se o valor de sua probabilidade, então se obtém o “comportamento” das probabilidades, ao qual se denomina *distribuição de probabilidades*, e se há uma regra (ou uma função) que implementa essa associação, então essa função é denominada função de probabilidade, representada por $f(x)$.

De maneira simplista, quando se pretende tomar decisões baseadas no grau de incerteza de determinadas situações, basicamente, deve-se identificar a variável aleatória de interesse e obter a sua distribuição de probabilidade. Com relação às distribuições de probabilidades, baseando-se no tipo da v.a., podem-se agrupá-las em duas classes distintas: as distribuições de probabilidades discretas e as contínuas. Essa classificação é apresentada na próxima seção, assim como, posteriormente, algumas das distribuições mais importantes de cada classe.

3.4.5. Distribuição de Probabilidades Discreta

A distribuição de probabilidade de uma variável aleatória discreta X , definida em um espaço amostral S , é uma tabela que associa a cada valor de X sua probabilidade (Dantas, 1997).

Para que uma função $f(x)$ seja uma distribuição de probabilidade, é necessário que:

- i. $f(x) \geq 0$
- ii. $\sum f(x) = 1$ (somatório para todos os valores de x do domínio de X)
- iii. $P(X = x) = f(x)$

Como exemplo, considere-se a soma dos pontos que aparecem no lançamento de dois dados, com seus valores possíveis da soma X e suas probabilidades associadas $f(x) = P(X=x)$, conforme Tabela 3.1 a seguir.

TABELA 3.1. Somas dos valores dos pontos amostrais (x) e suas probabilidades associadas ($f(x)$)

X	2	3	4	5	6	7	8	9	10	11	12
$f(x)$	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36

O gráfico da distribuição de probabilidades apresentada na Tabela 3.1 é mostrado na Figura 3.1.

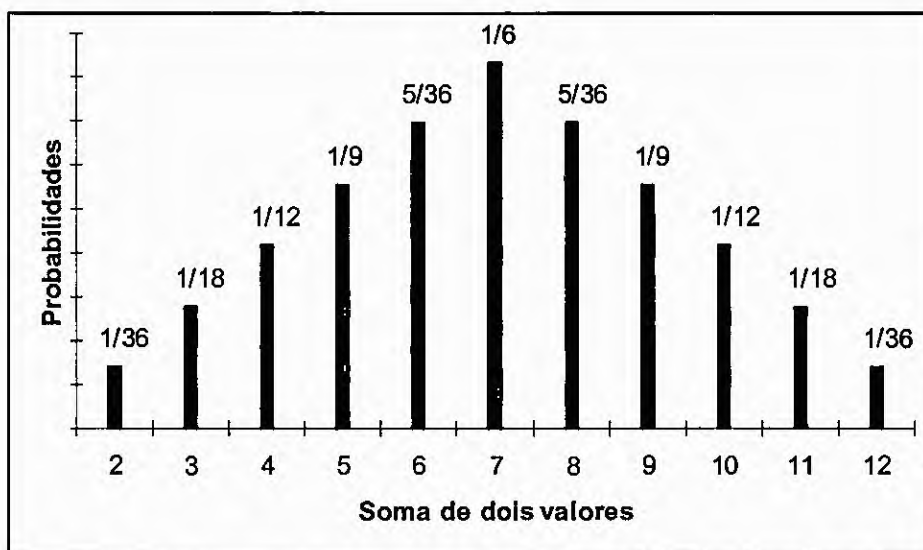


FIGURA 3.1. Gráfico em barras para a soma de pontos no lançamento de dois dados.

Também podem-se definir para as distribuições de probabilidades as mesmas medidas de tendência central e de dispersão. Dessa maneira, a média de uma variável aleatória, ou de sua distribuição de probabilidade, é dada pela soma dos produtos dos diversos valores x_i da v.a. pelas respectivas probabilidades.

A média de uma v.a. é comumente chamada de *valor esperado* ou de *esperança matemática* ou simplesmente de *esperança de X*. É representada por $E(X)$, sendo definida como:

$$E(X) = \sum x_i f(x_i)$$

$E(X)$ é uma média ponderada dos x_i , onde os pesos são as probabilidades associadas.

Já a variância pode ser definida como:

$$\sigma^2 = \text{Var}(X) = E[(X - \mu)^2] = E(X^2) - \mu^2, \text{ onde } \mu \text{ é uma constante.}$$

O *desvio padrão* (σ) é dado pela raiz quadrada positiva da variância, como segue:

$$\sigma = \sqrt{\sigma^2}$$

Uma outra definição importante está relacionada ao fato de, em vez da v.a. assumir o valor x : $f(x) = P(X=x)$, haver uma função que retorna a probabilidade de a variável aleatória x tomar um valor não superior a x : $P(X \leq x)$. Essa função é chamada de *função de distribuição acumulada* de uma v.a. X com função de distribuição de probabilidade $f(x)$ é dada por:

$$F(x) = P(X \leq x) = \sum_{l \leq x} f(l)$$

Até aqui, foram apresentadas as distribuições cujos espaços amostrais contêm um número finito, ou infinito contável de pontos. A seguir é apresentado o caso em que as distribuições possuem espaços amostrais incontáveis.

3.4.6. Distribuição de Probabilidades Contínua

Pode-se considerar uma variável aleatória que pode tomar todos os valores de um dado intervalo, tais como mensurações de altura, temperatura, tempo de espera em uma fila, etc. Apesar dessas mensurações, na prática, serem tratadas com aproximação de inteiro, elas possuem a natureza de v.a. essencialmente contínua.

Dessa forma, uma distribuição contínua pode ser encarada como um refinamento de uma distribuição discreta, à medida que se aumenta a precisão da

mensuração, até que, no limite, tem-se uma curva contínua. Essa curva contínua é denominada de *função da densidade de probabilidade*, usualmente designada por $f(x)$ (Soares et al., 1991). A função da densidade de probabilidade possui as seguintes propriedades:

- i. A área total sob a função de densidade é 1.
- ii. $P(a \leq X \leq b) =$ área sob a curva de densidade entre os pontos a e b .
- iii. $f(x) \geq 0$
- iv. $P(X=x_0) = 0$

Há uma discussão interessante em (Soares et al., 1991) a respeito da propriedade *iv*, como sintetizada a seguir: em uma distribuição contínua só faz sentido falar-se sobre a probabilidade de uma v.a. em um intervalo, sendo que se esse intervalo reduzir-se a um ponto (se os extremos a e b são coincidentes), a probabilidade é zero. Essa propriedade aparentemente gera um paradoxo de, por exemplo, não se poder afirmar que a altura de um indivíduo é exatamente 1,78m. Na verdade, admite-se que há uma imprecisão inerente aos equipamentos de medição, e que o valor 1,78m não se distingue de qualquer outro valor do intervalo [1,7845; 1,7857] ou [1,7895; 1,7901]. Então, o que realmente interessa é fato de a variável aleatória pertencer a um determinado intervalo, por menor que ele seja, o que assegura que a probabilidade já seja maior que zero.

As distribuições contínuas também possuem suas medidas de tendência central e de variabilidade, assim como as discretas, as quais serão apresentadas em cada distribuição contínua especial abordada na seção 3.4.8.

3.4.7. Distribuições de Probabilidades Discretas Clássicas

Nesta seção são apresentadas duas distribuições de probabilidades clássicas: a *Binomial* e a de *Poisson*. O sentido do termo “clássica” se refere à vasta citação dessas duas distribuições nas referências da área, além da gama de aplicações que pode ser descrita através das referidas distribuições. Entretanto, há algumas outras distribuições discretas bastante citadas na literatura, tais como hipergeométrica, geométrica, de Pascal, entre outras. Tais distribuições são apresentadas em (Allen, 1990), (Lazowska et al., 1984) e (Clarke & Disney, 1985).

• Distribuição Binomial:

Considerem-se situações em que se tem um conjunto de provas que satisfazem as seguintes condições:

- i. As diversas provas se realizam sob condições idênticas;
- ii. Cada prova comporta apenas dois resultados possíveis, mutuamente excludentes, denominados de sucesso (S) e falha (F);
- iii. A probabilidade de sucesso $p = P(S)$ é a mesma em cada prova, e permanece constante durante todo o experimento. A probabilidade de falha, também constante, é $P(F) = 1 - p = q$, de modo que $p + q = 1$;
- iv. As provas são independentes umas das outras, isto é, o conhecimento do sucesso (ou da falha) de uma delas não modifica a probabilidade de sucesso (ou de falha) nas provas subseqüentes.

Experimentos repetidos que obedecem às condições de (a) a (d) são chamados de provas de Bernoulli¹. Então, seja um experimento composto de n provas de Bernoulli, com probabilidade p para o sucesso e $1-p$ para o insucesso; e X , a variável aleatória que representa o número x de sucessos em n provas.

A distribuição de probabilidade da v.a. X é chamada *distribuição binomial* com n provas e probabilidade p de sucesso. Designa-se por $(n; p)$, onde n e p são os parâmetros da distribuição, e X é um inteiro positivo.

A função de probabilidade de X é:

$$f(x) = p(X = x) = \binom{n}{x} p^x q^{n-x}, \quad x = 0, 1, 2, \dots, n.$$

Exemplificando-se, a Figura 3.2 apresenta a distribuição binomial para $n=5$.

$p = 0,5$	$P\{x = 0\}f(0) = 0,0313$
	$P\{x = 1\}f(1) = 0,1562$
	$P\{x = 2\}f(2) = 0,3125$
	$P\{x = 3\}f(3) = 0,3125$
	$P\{x = 4\}f(4) = 0,1562$
	$P\{x = 5\}f(5) = 0,0313$
	1,0000

¹ Nome associado à família de matemáticos suíços (séculos XVII e XVIII) que cederam notória contribuição a vários ramos da ciência, incluindo-se o cálculo das probabilidades.

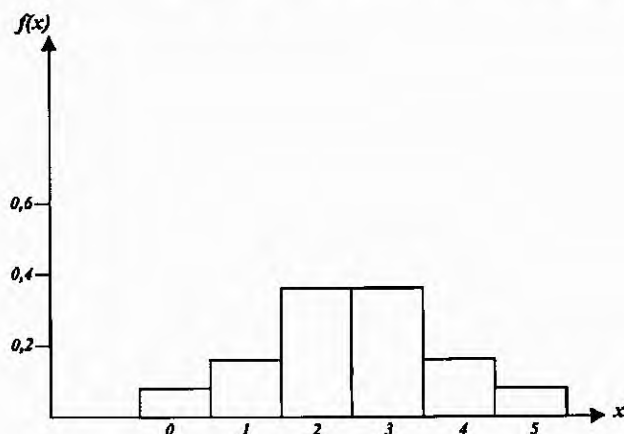


FIGURA 3.2. Gráfico da distribuição binomial para $n=5$ (Soares et al., 1991).

A média, considerando a v.a. de Bernoulli X definida como:

$$X_k \begin{cases} 1 \text{ no caso de sucesso na } k\text{-ésima prova} \\ 0 \text{ no caso de falha na } k\text{-ésima prova} \end{cases} \quad k = 1, 2, \dots, n$$

Então, $E(X) = E(X_1) + E(X_2) + \dots + E(X_n) = np$

E como as provas são independentes, a variância de x é a soma das variâncias individuais:

$$Var(X) = Var(X_1) + Var(X_2) + \dots + Var(X_n) = npq$$

e o desvio padrão $\sigma = \sqrt{npq}$

- Distribuição de Poisson

Uma distribuição de Poisson² é definida através de sua função de distribuição de probabilidade da seguinte forma:

$$P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!}, \quad x = 0, 1, 2, \dots; \lambda \text{ é o parâmetro da distribuição.}$$

Se X é uma variável aleatória com distribuição de Poisson e parâmetro λ , então média e variância são, respectivamente:

$$E(X) = \lambda \quad Var(X) = \lambda$$

A Figura 3.3, extraída de (Clarke & Disney, 1985), apresenta o gráfico da distribuição de Poisson para três valores distintos de λ .

² O nome da distribuição é em homenagem ao matemático francês S. Denis Poisson (1781-1840), que é autor de um livro no qual trata da aplicação da teoria das probabilidades na justiça comum.

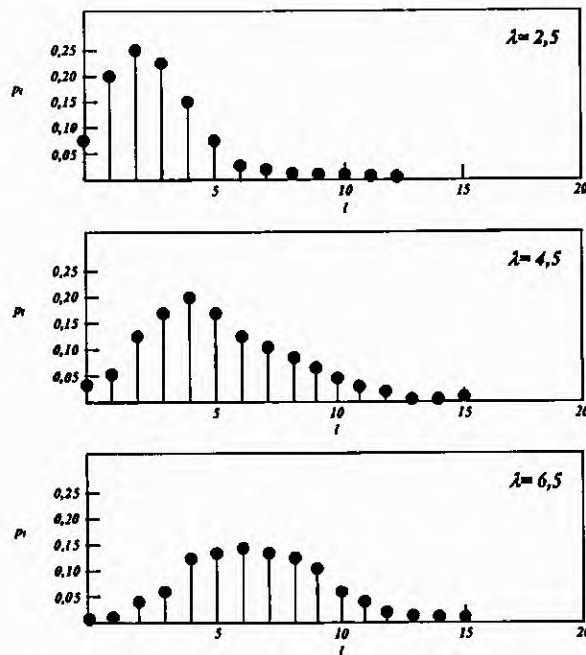


FIGURA 3.3. Gráfico da distribuição de Poisson para valores distintos de λ .

A distribuição de Poisson é característica para situações em que certos eventos acontecem em determinados instantes de tempo, tal como a chegada de um indivíduo em uma agência bancária.

Supondo-se que as hipóteses a seguir são válidas (Soares et al., 1991):

- i. O número de ocorrências de um evento em um intervalo de tempo é independente do número de ocorrências do evento em qualquer outro intervalo disjunto;
- ii. A probabilidade de duas ou mais ocorrências simultâneas é praticamente nula;
- iii. O número médio de ocorrências por unidade de tempo, λ , é constante ao longo do tempo.

Então, tem-se um *processo de Poisson*, podendo-se mostrar que o número de eventos (por exemplo, número de pessoas que entram em um banco), em um intervalo de tempo de amplitude t , tem distribuição de Poisson com parâmetro λt .

A distribuição de Poisson é chamada, por certos autores, de *distribuição dos eventos raros*, não no sentido de que o evento ocorre apenas vez por outra, mas sim porque o número de ocorrências é pequeno em relação ao número de provas. Portanto, um evento raro é aquele que apresenta uma pequena probabilidade de ocorrência.

A distribuição de Poisson descreve com propriedade os processos de chegadas em sistemas reais. Daí a grande importância dada a ela na literatura especializada.

3.4.8. Distribuições de Probabilidades Contínuas Clássicas

Nesta seção são apresentadas a distribuição normal e as distribuições para tempos de vida: exponencial, hiperexponencial e de Erlang. Há outras distribuições citadas na literatura específica, tais como a uniforme, triangular e a lognormal, que podem ser encontradas em (Dantas, 1997), (Soares et al., 1991) e (Allen, 1990).

• A Distribuição Normal:

É a mais proeminente das distribuições de probabilidade. Ela é associada aos erros de mensuração, pois quando se efetuam mensurações amíúde, com um aparelho equilibrado, não se chega ao mesmo resultado, e sim a valores que oscilam, de modo aproximadamente simétrico, em torno do verdadeiro valor, descrevendo uma “curva em forma de sino”. Entretanto, essa distribuição não é universal como se supunha inicialmente, quando se pensava que todos os fenômenos da vida real se ajustavam à curva normal³.

A distribuição normal tem sua densidade dada por:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{(x - \mu)^2}{2\sigma^2}, \text{ onde } \mu \text{ e } \sigma \text{ são os parâmetros da distribuição.}$$

A Figura 3.4 ilustra uma curva típica da distribuição normal

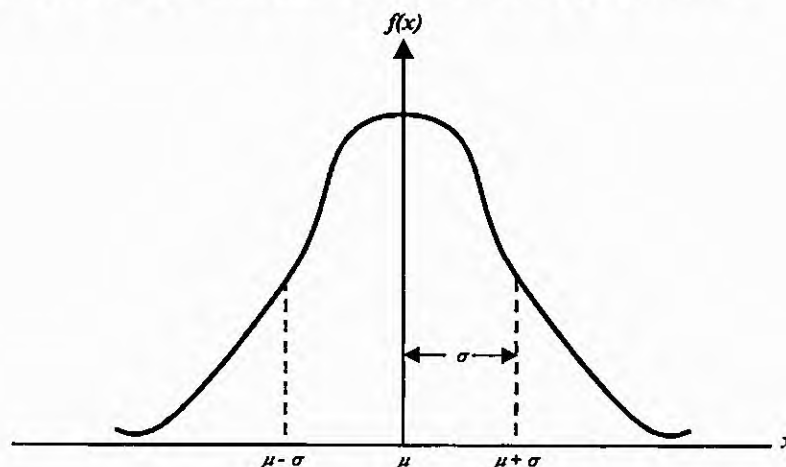


FIGURA 3.4. Gráfico da distribuição normal .

³ Daí o nome “normal”, indicando que qualquer fenômeno que não se enquadrasse na simetria da curva em forma de sino era consequência de anormalidades no processo de coleta dos dados.

As principais características da distribuição normal são as seguintes:

- i. A média da distribuição é μ ,
- ii. O desvio padrão é σ ,
- iii. A moda ocorre em $x = \mu$,
- iv. A curva é simétrica em relação a um eixo vertical passando por $x = \mu$,
- v. A curva tem inflexões nos pontos $x = \mu \pm \sigma$, sendo côncava para baixo se $\mu - \sigma < x < \mu + \sigma$, e côncava para cima caso contrário;
- vi. A área total sob a curva normal e acima do eixo horizontal é igual a 1.

A probabilidade de uma v.a. normal X estar entre a e b é igual à área sob a curva e acima do segmento horizontal $[a;b]$, conforme ilustra a Figura 3.5.

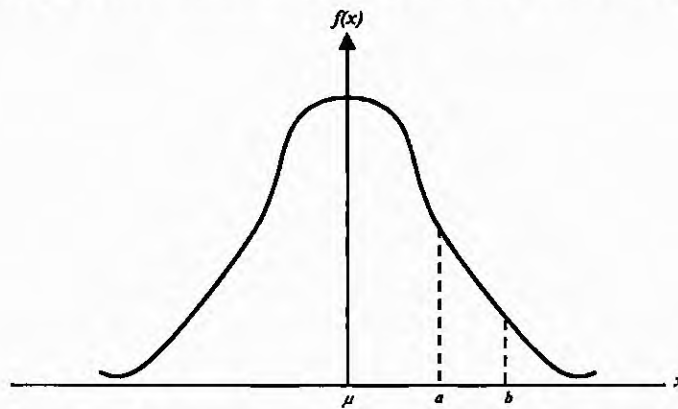


FIGURA 3.5. Probabilidade igual à área sob a curva, entre a e b .

- A Distribuição Exponencial:

A distribuição exponencial é amplamente usada para descrever tempos de vida de componentes ou de sistemas, além de representar com propriedade tempos de atendimentos em recursos.

A característica mais importante da exponencial é a propriedade do “esquecimento” (*memoryless*), que estabelece que o restante do tempo de uma determinada atividade independe de quanto essa atividade já consumiu anteriormente. Além do esquecimento, a exponencial também possui a característica de ter uma grande variância, mas a despeito disso, ela possui um tratamento matemático relativamente fácil, o que faz com que ela seja assumida em vários estudos.

Uma v.a. X tem distribuição exponencial se sua função de densidade de probabilidade é da forma:

$$f(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0$$

0 em caso contrário

Onde λ é o parâmetro da distribuição.

O gráfico da distribuição exponencial para três valores diferentes do parâmetro λ é apresentado na Figura 3.6.

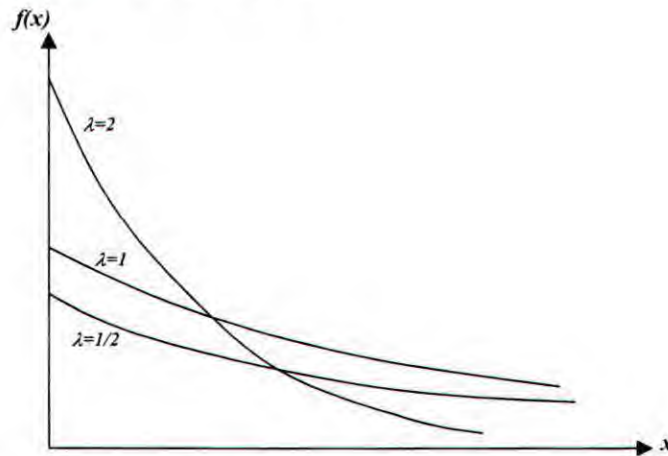


FIGURA 3.6. Gráfico da distribuição exponencial para três valores diferentes de λ .

A média e a variância da distribuição exponencial são, respectivamente:

$$\mu = \frac{1}{\lambda} \quad \sigma^2 = \frac{1}{\lambda^2}$$

- Distribuição de Erlang:

A distribuição de Erlang⁴ é derivada da soma de um número inteiro de variáveis aleatórias independentes e exponencialmente distribuídas. Ela é usada em teoria de filas, quando uma atividade ou tempo de serviço ocorre em fases, com cada fase exponencialmente distribuída.

Uma variável aleatória T é dita ser uma v.a. com distribuição de Erlang de estágio k , com parâmetros k e μ se sua função de densidade de probabilidade é dada por:

$$f(x) = \begin{cases} \frac{\mu^k (\mu k t)^{k-1}}{(k-1)!} e^{-\mu k t} & \text{para } t > 0 \\ 0 & \text{para } t \leq 0 \end{cases}$$

⁴ Em homenagem ao engenheiro dinamarquês A.K. Erlang, que aplicou a distribuição para estudar ligações em centrais telefônicas, em 1917.

O modelo físico da distribuição de Erlang está representado na Figura 3.7, na qual é ilustrado um determinado servidor com k estágios idênticos e independentes, cada um com uma distribuição exponencial. Assim, o tempo de serviço, T , é a soma de k variáveis aleatórias exponencialmente distribuídas, cada uma com parâmetro μk .

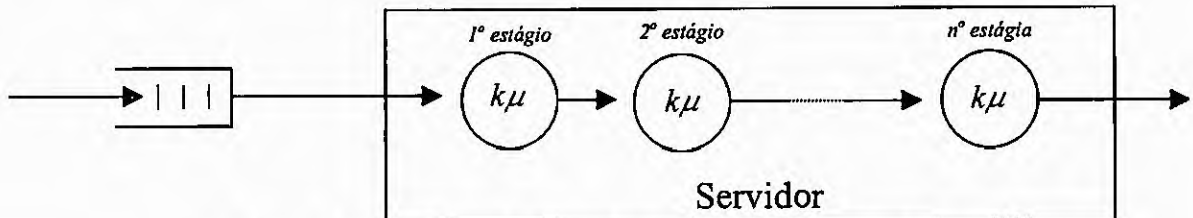


FIGURA 3.7. Modelo de Erlang e sua distribuição.

Distribuição Hiperexponencial:

A distribuição hiperexponencial é adequada para sistemas de filas em que o tempo de serviço possui um grande valor de desvio padrão relativo à média. Nesse caso, o termo "hiperexponencial" está mais próximo do termo "superexponencial".

Um modelo representativo da distribuição hiperexponencial é apresentado na Figura 3.8. Nessa Figura há dois estágios paralelos, cada um com um parâmetro (μ_1 para o estágio superior, e μ_2 para o estágio inferior). Um cliente que acesse o servidor pode ir para o estágio superior com probabilidade q_1 , e para o estágio inferior com probabilidade q_2 (com $q_1 + q_2 = 1$), com ambos os estágios prestando serviço a uma taxa μ_i exponencialmente distribuída. Assim, a função de densidade de probabilidade para o tempo de serviço é dada por:

$$f(t) = q_1\mu_1e^{-\mu_1t} + q_2\mu_2e^{-\mu_2t}, \quad t \geq 0.$$

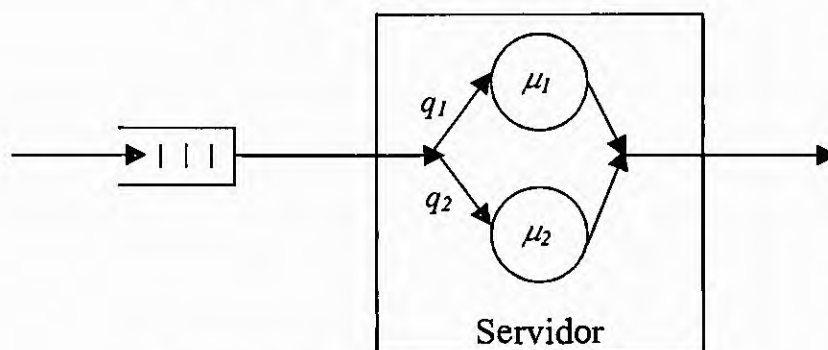


FIGURA 3.8. Modelo Hiperexponencial.

Distribuições de probabilidades são fundamentais para descrição de sistemas de filas, como vistos na seção 2.3.1. Esses sistemas, na verdade, são descritos por uma combinação de variáveis aleatórias. A Figura 3.9 apresenta as possíveis v.a. constantes em um sistema de filas.

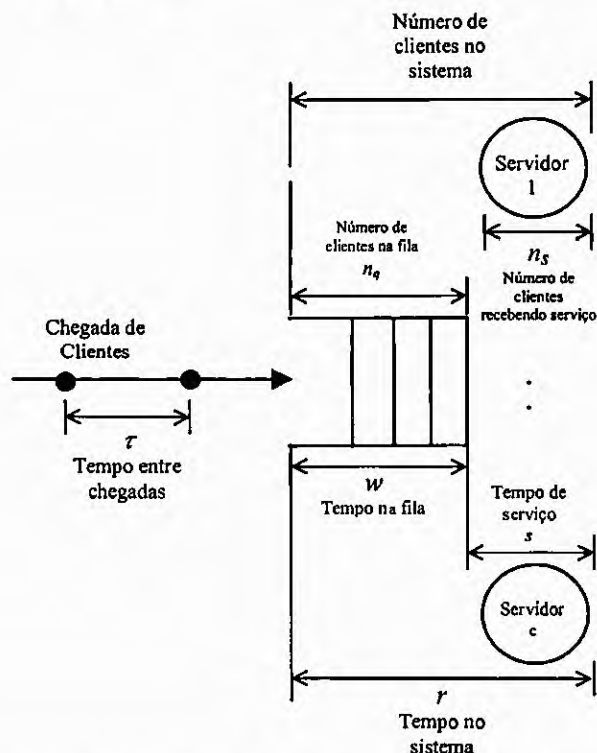


FIGURA 3.9. Variáveis Aleatórias em um Sistema de Filas (Allen, 1990).

3.5. Processos Estocásticos e Cadeias de Markov

Esta seção apresenta o estudo de situações em que são feitas observações quanto a um período de tempo. Situações essas influenciadas por efeitos aleatórios, durante todo o intervalo de tempo (ou seqüência de tempos) que se está a considerar. Fenômenos desse perfil são denominados de *processos estocásticos*, e possuem uma gama bastante variada de exemplos, tais como mutações genéticas ou número de pacotes em uma rede de comunicação.

Em especial, o interesse estará focado em um tipo particular de processos estocásticos denominado de cadeias de Markov, em ambas as suas abordagens temporais (discreta e contínua).

3.5.1. Processos Estocásticos

Genericamente, um processo estocástico é um fenômeno que varia em algum grau, de maneira imprevisível, à medida que o tempo passa. Entretanto, uma experiência aleatória, nesse contexto, não é mais representada por um valor único (a probabilidade), e sim por uma função que represente uma seqüência ou uma série de valores de probabilidade (Kovacs, 1996).

A observação de um processo estocástico é feita no decorrer do tempo. Sendo assim, deve-se especificar um conjunto T de tempos envolvidos no fenômeno. Dessa forma, os tempos envolvidos podem ser interpretados de duas formas: uma abordagem *contínua* e outra *discreta*. Na primeira, o tempo é observado continuamente, tal como no experimento de registrarem-se os carros que passam em um determinado cruzamento. Já a segunda abordagem, T pode ser uma seqüência de tempos, tal como o registro do número de usuários no sistema, a cada 20 unidades de tempo.

O tempo discreto é uma seqüência de inteiros consecutivos $T = \{0, 1, 2, 3, \dots\}$ ou $T = \{1, 2, 3, \dots\}$. Já T contínuo consiste de todos os tempos subseqüentes a alguma origem dada: $T = \{t: 0 \leq t \leq \infty\}$.

Neste trabalho, está-se interessado em um tipo particular de processos estocástico, cadeias de Markov⁵, apresentados na seção seguinte.

3.5.2. Cadeias de Markov de Parâmetro Discreto

Considere-se um processo aleatório de parâmetro discreto e de espaço de estados discreto.

$$\{X_n\} = \{X_0, X_1, X_2, \dots\}$$

Então a estrutura de probabilidade de um processo aleatório de parâmetro discreto é determinada pelas probabilidades conjuntas:

$p(j_0, j_1, \dots, j_k) = P[X_0 = j_0, X_1 = j_1, \dots, X_k = j_k]$ para todo k finito e para toda seqüência j_0, j_1, \dots, j_k de estados.

Tal processo é denominado um processo de Markov ou cadeia de Markov se, para cada k , a probabilidade condicional de que o sistema esteja em um dado estado após k passos, conhecendo-se os estados do sistema em todos os passos

⁵ Em homenagem ao matemático russo Markov.

anteriores, é a mesma que a probabilidade condicional, conhecendo-se apenas o estado em um passo imediatamente anterior (Clarke & Disney, 1985). Ou seja:

$p(j_k | j_0, j_1, \dots, j_{k-1}) = P[X_k = j_k | X_{k-1} = j_{k-1}]$, para cada k e para cada seqüência j_0, j_1, \dots, j_k de estados.

A representação das transições entre os estados da cadeia é sintetizada pela matriz de transição do processo. Essa matriz tem a propriedade de que a soma de todas as componentes de cada linha é 1, e por isso é denominada de *matriz estocástica de transição* (Figura 3.10).

$$P = (p_{ij}) = \begin{pmatrix} p_{00} & p_{01} & p_{02} & \dots & K \\ p_{10} & p_{11} & p_{12} & \dots & K \\ \dots & \dots & \dots & \dots & \dots \\ M & M & M & \dots & M \end{pmatrix}$$

FIGURA 3.10. Matriz de Transição.

Além de estabelecer-se uma matriz que apresente as probabilidades de transição entre os estados, é importante classificar esses estados, de modo que se possam identificar certos tipos de cadeias, de acordo com os estados que a compõe. Para classificação de estados em processos markovianos discretos, seguem-se as definições (Clarke & Disney, 1985):

- i. Supondo-se que o processo comece em algum estado j . Se k é um estado tal que $p_{jk}^{(n)} > 0$ para algum n , então se diz que o estado k poderá ser alcançado do estado j . Se, além disso, $p_{kj}^{(m)} > 0$ para algum m , o estado j poderá ser alcançado do estado k , e se diz que os estados j e k se comunicam.
- ii. Se C é um conjunto de estados, tal que nenhum estado fora de C pode ser atingido, partindo-se de qualquer estado dentro de C , então o conjunto C é dito *fechado*. C tem a propriedade de que, uma vez alcançado, o processo não o deixará mais. Se, além disso, cada par de estados dentro de C se comunicarem, então C é chamado de *classe de comunicação fechada*.
- iii. Se um conjunto fechado contém somente um estado, então ele é chamado de *estado absorvente*.

- iv. Se uma cadeia markoviana discreta não contém conjuntos fechados, com exceção do conjunto de todos os estados, então a cadeia é dita *irredutível*. Uma cadeia é irredutível se todo estado puder ser alcançado partindo-se de todos os estados.

Para efeito de avaliação de desempenho, as cadeias de Markov são interessantes para representar sistemas que estejam funcionando há algum tempo. Mais precisamente, está-se à procura de um vetor de probabilidades que "despreza" a irregularidade dos "efeitos de partida" do sistema, tal que:

$$v_j = \lim_{n \rightarrow \infty} p_j^{(n)}$$

As quantidades v_j são, certas vezes, referidas como as *probabilidades de estado permanente*. Uma interpretação interessante dada ao vetor é que ele pode ser visto como a proporção de tempo, a longo prazo, que o processo gasta no 'estado j '; pois é razoável se pensar que, ao longo de um grande período de tempo, a influência do estado inicial do processo comece a desaparecer, e que as probabilidades limite não sofram a influência das probabilidades iniciais. Dessa forma, se cada v_j realmente não depende do estado inicial, a matriz $\mathbf{P}^{(n)} = (p_{ij}^{(n)})$, matriz de transição, convergirá para uma matriz \mathbf{V} , com $n \rightarrow \infty$, na qual cada linha é idêntica ao vetor \mathbf{v} com componentes v_j (vetor de probabilidades limite).

$$P^{(n)} \rightarrow V = \begin{pmatrix} v \\ v \\ \vdots \\ v \\ M \end{pmatrix} \quad \text{com } n \rightarrow \infty, \text{ onde } (v_0 \quad v_1 \quad \dots \quad v_j \quad \dots)$$

Assim, obtido o vetor de probabilidade, têm-se os tempos médios de permanência em cada estado, e as medidas de desempenho que são derivadas diretamente desse tempo podem ser calculadas.

A respeito da existência das probabilidades limite, os seguintes teoremas (extraídos de (Clarke & Disney, 1985)) são apresentados sem prova:

Teorema 1. *Em qualquer cadeia aperiódica de Markov, todos os limites $v_j = \lim_{n \rightarrow \infty} p_j^{(n)}$ existem.*

Teorema 2. *Em qualquer cadeia aperiódica de Markov todos os limites $v_j = \lim_{n \rightarrow \infty} p_j^{(n)} = \lim_{n \rightarrow \infty} p_{ij}^{(n)}$ não dependem da distribuição inicial.*

Teorema 3. *Em qualquer cadeia de aperiódica irreduzível⁶, finita de Markov, o vetor limite $v=(v_0, v_1, \dots, v_j, \dots)$ é o único vetor de probabilidade estacionária do processo.*

Esses teoremas são, via de regra, suficientes para permitir que se discuta e se determine o vetor de probabilidades limite nas cadeias de Markov finitas encontradas na prática.

3.5.3. Cadeias de Markov de Parâmetro Contínuo

Esta seção apresenta não mais apenas uma seqüência de variáveis aleatórias, mas também uma variável aleatória X , que depende de um parâmetro t , o qual pode assumir uma gama contínua de valores (via de regra, números reais positivos). Dessa forma, o interesse gira em torno de uma família de variáveis aleatórias da forma $\{X_t : 0 \leq t < \infty\}$, onde geralmente t é interpretado como tempo.

Um processo aleatório de parâmetro contínuo é uma função $X_t(s)$, de duas variáveis, t e s , onde t é uma variável de tempo contínuo e s representa um ponto amostral em algum espaço amostral fixado S , que é um conjunto discreto chamado *espaço de estados*.

A propriedade do "esquecimento", válida para as cadeias de Markov de parâmetro discreto, pode ser prontamente estendida de modo a aplicar-se a processos de parâmetro contínuo.

Para discorrer-se sobre a matriz geradora infinitesimal, correspondente à matriz estocástica de transição do processo de parâmetro discreto, recorrer-se-á a um exemplo extraído de (Clarke & Disney, 1985). Imagine-se um processo aleatório de dois estados, 0 e 1, onde 0 representa o estado "em reparo" e 1 representa o estado "em funcionamento", admitindo-se a existência da propriedade de esquecimento. Assumindo-se que a extensão média de um período de reparo é $1/\lambda$ e que a extensão média de um período de funcionamento é $1/\mu$. Por definição, as taxas de passagem para o estado de reparo é λ e para o estado de funcionamento é μ . A matriz Λ é de ordem 2, e possui a propriedade da soma de cada linha ser igual a 0. Assim, Λ , para o exemplo, seria (Figura 3.11):

⁶ É aquele processo no qual todo estado pode ser alcançado, partindo-se de qualquer um dos estados do sistema.

$$\Lambda = \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix} \end{matrix}$$

FIGURA 3.11. Matriz de Taxas.

Da matriz Λ , podem-se deduzir as equações, chamadas de equações progressivas de Kolmogorov:

$$\begin{aligned} p_{i0}(t) &= -\lambda p_{i0}(t) + \mu p_{i1}(t) \\ p_{i1}(t) &= \lambda p_{i0}(t) - \mu p_{i1}(t), \quad \text{com } i=0, 1. \end{aligned}$$

Assim, podem-se calcular as probabilidades de transição para o exemplo de dois estados.

Neste ponto, volta-se à discussão sobre a existência das probabilidades limite. O seguinte teorema especifica a existência ou não do vetor de probabilidades limite.

Teorema

Seja X_t um processo irreduzível que satisfaça as hipóteses apresentadas (Clarke & Disney, 1985):

- i. Se o sistema $q \Lambda$, onde $q = (q_0, q_1, q_2, \dots)$, tem uma solução trivial $(0, 0, 0, \dots)$ como sendo a única solução para a qual $\sum q_j < \infty$, então $q_j = 0$, para todo j .
- ii. Se o sistema $q \Lambda$ tem exatamente uma solução $q = (q_0, q_1, q_2, \dots)$, na qual cada $q_j > 0$ e $\sum q_j = 1$, então o sistema é recorrente positivo e esta solução dá as probabilidades limite.
- iii. Se o espaço de estados é finito, o caso (ii) deve ocorrer.

Uma questão relevante a ser abordada é a interpretação dada às probabilidades limite no caso contínuo. À risca, as interpretações são análogas às apresentadas para parâmetro discreto (novamente listadas a seguir).

A primeira interpretação pode ser dada ao q_j como a probabilidade de que o sistema estará no estado j "muito tempo" depois de $t = 0$. Dessa forma, q_j está relacionado com probabilidades a longo prazo de estar-se no estado j , ou a *probabilidade de estado de equilíbrio (Steady State)* para o estado j .

Outra interpretação possível, muito importante para a visão de avaliação de desempenho, é que q_j é a *proporção de tempo que o sistema gasta no estado j* ,

durante um longo prazo. Essa interpretação gera várias medidas de desempenho derivadas do tempo médio de permanência em estados, apresentadas na seção das leis operacionais. Ambas as interpretações, além da existência de um q_j , são discutidas com bastante propriedade em (Clarke & Disney, 1985).

Na próxima seção, será apresentada uma aplicação da teoria de processos markovianos: o estudo das filas de espera.

3. 6. Introdução à Teoria de Filas

Em sistemas reais, há uma abundância de situações, nas quais se evidencia a ocorrência de filas diante de determinados recursos. Para tratar adequadamente o problema, há um ramo da teoria das probabilidades, denominado de *teoria de filas*, cujo cerne está em delimitar e compreender eventos, estados e variáveis envolvidas em sistemas de filas.

Em (Jain,1991), há uma descrição das variáveis aleatórias constantes em um sistema de filas. A definição de cada uma delas é exposta a seguir:

- τ = tempo entre chegadas de clientes sucessivos.
- λ = taxa média de chegada, que por definição $\lambda = 1/E[\tau]$, onde $E[\tau]$ é o tempo médio entre chegadas.
- s = tempo de serviço por cliente.
- μ = taxa média de serviço por servidor = $1/E[s]$, onde $E[s]$ é o tempo médio de serviço. Por extensão, taxa de serviço total para c servidores é $c\mu$.
- n = número de clientes no centro (em algumas obras, referenciado como comprimento da fila). É importante observar que n inclui tanto os clientes que estão recebendo serviço quanto os clientes que estão à espera na fila.
- n_q = número de clientes esperando para receber serviço, ou seja, esperando na fila para serem atendidos pelo servidor. Sempre, n_q é menor que n .
- n_s = número de clientes recebendo serviço.
- r = tempo de resposta ou tempo no sistema. Inclui tanto o tempo de espera como o tempo de recebimento de serviço (ou de atendimento).
- w = tempo de espera ou tempo de fila, que é o intervalo de tempo compreendido entre a chegada e o instante em que inicia o serviço.

Com exceção de λ e μ (que são valores médios), todas são *variáveis aleatórias*, ressaltando-se que λ e μ são, respectivamente, médias das taxas chegada e de atendimento (ambas também variáveis aleatórias). As equações apresentadas nesta seção são resultados de relações algébricas existentes entre essas variáveis. Enfatiza-se que da seção 3.6.1 à seção 3.6.3, as equações são válidas apenas para sistemas de fila única. As seções subseqüentes (leis operacionais) são válidas indistintamente com relação ao número de filas presente.

3.6.1. Condição de Estabilidade

Um sistema é dito instável quando a quantidade de clientes aumenta de forma contínua, tendendo ao infinito. Para que um sistema seja considerado estável, a taxa média de chegada deve ser menor que a taxa média de serviço (Jain, 1991).

$$\lambda < c\mu, \text{ onde } c \text{ é número de servidores.}$$

Assim, com a condição de estabilidade satisfeita, consegue-se dar vazão ao sistema, evitando-se que a quantidade de clientes tenda ao infinito. Essa condição é inerente aos sistemas cujas filas possuem capacidade infinita, haja vista que se houver uma limitação no tamanho da fila, aqueles clientes excedentes serão descartados, o que implica impossibilidade de geração de filas tendendo ao infinito.

3.6.2. Clientes no Sistema e Clientes na Fila

O número de clientes em um sistema é sempre igual à soma da quantidade de clientes na fila e a quantidade de clientes recebendo atendimento.

$n = n_q + n_s$, considerando-se as componentes da equação como variáveis aleatórias.

Pode-se extrapolar a equação anterior para as médias das v.a., ou seja, o número médio de clientes no sistema é igual a soma do número médio de clientes na fila e o número médio de clientes recebendo serviço.

$$E[n] = E[n_q] + E[n_s]$$

3.6.3. Tempo de Resposta e Tempo de Fila

O tempo de resposta é o tempo dispensado por um cliente no sistema, incluindo o seu tempo de espera na fila e o tempo que o mesmo passou recebendo serviço, ilustrado na equação:

$$r = w + s$$

Analogamente ao tempo de resposta, pode-se extrapolar a equação para as médias dos tempos de resposta e de fila e também para as suas variâncias :

$$E[r] = E[w] + E[s]$$

3.6.4. Leis Operacionais

As *leis operacionais* estabelecem relações algébricas simples, entretanto bastante relevantes à teoria de filas, permitindo que modelos de rede de filas tenham o seu desempenho avaliado.

No contexto das Leis Operacionais, é importante o conceito de *amostras operacionais*, as quais correspondem a valores que podem ser medidos durante um período de observação finito. Na observação de um dispositivo i (servidor) qualquer, durante um tempo T finito, obtêm-se, por exemplo, as seguintes variáveis operacionais (Jain,1991):

- A_i = número de chegadas de clientes;
- C_i = número de clientes completados;
- B_i = tempo em que o servidor permaneceu ocupado.

A partir dessas variáveis, podem-se obter outras variáveis operacionais, que são chamadas de *amostras operacionais derivadas*:

- *Taxa de chegada* $\lambda_i = \frac{A_i}{T}$
- *Throughput* $X_i = \frac{C_i}{T}$
- *Utilização* $\rho_i = \frac{B_i}{T}$
- *Tempo médio de serviço* $S_i = \frac{B_i}{C_i}$

As amostras operacionais podem mudar de um período de observação para outro, todavia há certas relações algébricas que se mantêm em todo o período de observação. Essas relações são denominadas *Leis Operacionais*.

Para tanto, é tomada a seguinte assunção: se o período de observação é suficientemente longo, o número de chegadas deve ser aproximadamente igual ao número de clientes que são completados ($C \approx A$). De maneira que se pode assumir que $\lambda \approx X$ (taxa de chegada é aproximadamente igual ao *throughput* do sistema). A esse fenômeno é dado o nome de *Balanceamento de Fluxo*, no qual o fluxo de trabalho que sai do sistema é balanceado com o fluxo de trabalho que entra no mesmo. Na descrição das leis operacionais, é assumido o *Balanceamento de Fluxo*.

3.6.4.1. Lei de Little

A Lei de Little, a mais proeminente das leis operacionais, especifica que sistemas cheios (um valor de n grande) estão associados a longos atrasos de clientes (grande valor de T), e vice-versa (Menascé & Almeida, 1998). Onde n é número médio de clientes observados no sistema durante o período de tempo de observação T .

Considerando n como sendo o número médio de clientes no sistema durante o período de observação, e o r como o tempo médio gasto pelos clientes no sistema (também chamado de *tempo de permanência*) e definindo-se r_n como o tempo gasto no sistema pelo n -ésimo cliente, então $r = \sum r_n / n$. O número médio de clientes no sistema é, então, $n = r \frac{C}{T}$, e admitindo-se que o *throughput* é dado por $X = \frac{C}{T}$, a Lei de Little pode ser descrita por:

$$n = X.r$$

Considerando o Balanceamento de Fluxo ($\lambda \approx X$):

$$n = \lambda . r$$

A Lei de Little também pode ser extrapolada para componentes específicos, e não apenas ao sistema como um todo. De maneira que, por exemplo, podem-se considerar, para um determinado dispositivo i , com n_i como o número médio de clientes e r_i o tempo médio de resposta do dispositivo i , as seguintes aplicações da Lei de Little:

$$n = X_i . r_i \text{ ou } n = \lambda_i . r_i$$

3.6.4.2. Lei do Fluxo Forçado

A Lei do Fluxo Forçado estabelece uma relação entre o *throughput* do sistema e o *throughput* individual dos dispositivos. Para tanto, há que se fazer uma diferenciação na abordagem do *throughput* em um sistema aberto e em um sistema fechado. Dessa forma, é assumido o seguinte:

- Em um modelo aberto, o número de clientes que chega ao sistema por unidade de tempo representa o *throughput* do sistema.
- Em um modelo fechado, por não haver clientes chegando ao sistema, deve-se fazer a seguinte adaptação: considera-se que um modelo fechado pode ser visto como um modelo aberto com realimentação (todos clientes que saem voltam imediatamente para o sistema), onde a saída e a entrada são representadas, respectivamente, pelos artifícios denominados de *link OUT* e *link IN* (Figura 3.12).

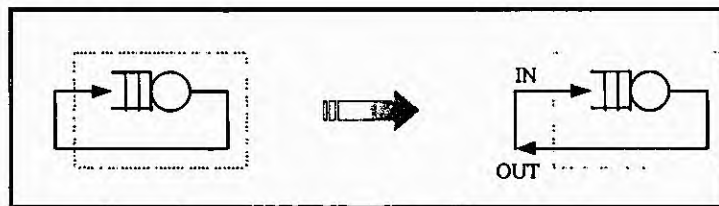


FIGURA 3.12 – Links In e Out para Modelo Fechados

Assumindo-se as premissas anteriores e se, no período de observação T , o número de clientes que chega em cada dispositivo é igual ao número de clientes que sai ($A_i = C_i$), diz-se que os dispositivos satisfazem a *hipótese de Fluxo Balanceado*.

Tendo um fluxo de clientes balanceado, supondo-se que cada cliente faça V_i requisições ao i -ésimo dispositivo do sistema, o número de clientes que sai do sistema (ou que são completados) (C_0) e o número de clientes que visitam o i -ésimo dispositivo (C_i) são descritos, respectivamente, por:

$$C_i = C_0 \cdot V_i \quad \text{ou} \quad V_i = \frac{C_i}{C_0}, \text{ onde a variável } V_i \text{ é a taxa de visitas ao } i\text{-ésimo dispositivo.}$$

Assim, durante o tempo de observação, o *throughput* do sistema é dado por:

$$X = \frac{C_0}{T}$$

O *throughput* para um determinado dispositivo i é dado por:

$$X_i = \frac{C_i}{T} = \frac{C_i}{C_0} \cdot \frac{C_0}{T}$$

Por analogia, a Lei do Fluxo Forçado é expressa por:

$$X_i = X \cdot V$$

3.6.4.3. Lei de Utilização

Admitindo-se a Utilização como sendo dada por $\rho_i = \frac{B_i}{T}$, e também que $\frac{B_i}{T} = \frac{C_i}{T} \cdot \frac{B_i}{C_i}$. Admitindo-se ainda que $\frac{C_i}{T} = X_i$ e $\frac{B_i}{C_i} = S_i$, a Lei de Utilização pode ser expressada da seguinte maneira:

$$\rho_i = X_i S_i$$

Considerando-se ainda a existência do *Balanceamento de Fluxo*, tem-se:

$$\rho_i = S_i \lambda$$

Uma variante pode ser obtida através da combinação da Lei de Utilização com a Lei do Fluxo Forçado, obtendo-se:

$$\rho_i = X_i S_i = X V_i S_i$$

3.6.4.4. Lei Geral do Tempo de Resposta

Baseando-se na Lei de Little, que relaciona número de clientes, tempo de resposta e *throughput*, como segue:

$$n = X \cdot r$$

E admitindo-se que n_i é número de clientes que está apenas no dispositivo i , n pode ser calculado por:

$$n = n_1 + n_2 + \Lambda + n_M, \text{ com } n = 0, 1, \dots, M.$$

Fazendo-se $n_i = X_i \cdot r_i$ (Lei de Little), obtém-se:

$$Xr = X_1 r_1 + X_2 r_2 + \Lambda + X_M r_M$$

Dividindo-se ambos os lados dessa equação pelo *throughput*, obtém-se, por intermédio da Lei do Fluxo Forçado, a Lei Geral do Tempo de Resposta:

$$r = V_1 r_1 + V_2 r_2 + \Lambda + V_M r_M$$

ou

$$r = \sum_{i=1}^M V_i r_i$$

As equações das lei operacionais, denominadas por certos autores como leis fundamentais, são abordadas em várias publicações inerentes à área, tais como (Jain, 1991), (Menascé & Almeida, 1998) e (Allen, 1990).

3.7. Considerações Finais

Este capítulo abordou de maneira sucinta a fundamentação que dá suporte à avaliação de desempenho, em sua vertente de modelagem, seja utilizando-se de soluções analíticas ou de soluções por simulação.

A probabilidade e suas extrapolações (como a teoria de filas) permitem que se tenha uma idéia a longo prazo de um determinado sistema, o que é recurso extremamente válido e desejável em certas situações reais. Através dessas inferências, são possíveis realizações de planejamentos mais racionais, atenuando-se custos e esforços desnecessários.

A área de processo estocásticos possui uma vasta aplicação, pois muitos fenômenos existentes são essencialmente aleatórios. Exemplos desses fenômenos variam desde aplicações em computação até difusão molecular, o que mostra o poder de abrangência que a matéria possui. Por esse motivo, sem ter-se a pretensão de esgotar o assunto, foi traçado um mapa simplificado das principais abordagens da área, com o intuito de servir de arcabouço para aquilo que será necessário neste trabalho.

A despeito da falta de exemplos em algumas definições efetivadas neste capítulo, vale ressaltar que, posteriormente, em capítulos subseqüentes, será feito uso mais intensivo dos conceitos apresentados aqui, tais como medidas centrais e de dispersão, distribuições de probabilidades e princípios de teoria de filas.

Capítulo 4

Statecharts Estocásticos e Queuing Statecharts

4.1. Considerações Iniciais

Este capítulo apresenta a sintaxe e a semântica formais para Statecharts, como definidas originalmente por D. Harel (Harel, 1987), expondo a dinâmica criada a partir dos conceitos de passos e configurações. Baseado nesses conceitos, processa-se uma discussão a respeito da necessidade de introduzirem-se novas definições, as quais devem ser requisitos para a especificação de sistemas de filas: alguns desses conceitos são os de estados com tempos de permanência associados e a escolha por probabilidade, semelhante à entrada por condição, onde a condição é uma probabilidade.

Além desses dois conceitos, são estabelecidas as funções que determinam o tempo de cada passo, assim como os instantes em que ocorrem o início e fim de cada configuração. A partir dessas funções, há uma redefinição daqueles conceitos originais de D. Harel (Harel et al., 1987), que estabelecem a dinâmica dos Statecharts.

Ao final do capítulo, é apresentada uma aglutinação entre duas técnicas complementares em suas especificações: redes de filas e Statecharts. A essa extensão é dado o nome de Queuing Statecharts.

4.2. Sintaxe dos Statecharts

Segundo D. Harel (Harel et al., 1987), a sintaxe dos Statecharts é definida através de um conjunto de cinco elementos básicos: estados, transições, eventos primitivos, condições primitivas e variáveis. A partir desses elementos, podem-se definir conjuntos estendidos de eventos, condições, expressões e rótulos, além da associação entre esses elementos. Segue a descrição dos elementos básicos, como definida em (Harel et al., 1987):

- **Estados:** o conjunto S de estados é definido por uma função ρ de hierarquia, uma função ψ de tipo, um conjunto H de símbolos de história e uma função δ default.

A função hierárquica $\rho : S \rightarrow 2^S$ define para cada estado os seus subestados. Tem-se que se $\rho(x) = \rho(y)$ então $x=y$.

Existe um único estado $r \in S$ tal que $\forall s \in S, r \notin \rho(s)$; r é a raiz do Statechart.

ρ^* e ρ^+ são extensões de ρ definidas por:

$$\rho^*(s) = \cup \rho^l(s), l \geq 0;$$

$$\rho^+(s) = \cup \rho^l(s), l \geq 1.$$

A Função Tipo $\psi : S \rightarrow \{\text{AND}, \text{OR}\}$ define o tipo de cada estado.

O conjunto de Símbolos de História, H , está relacionado ao conjunto de estados pela função $\gamma : H \rightarrow S$ tal que:

Define-se $\omega : S \cup H \rightarrow S$ por $\omega(z)$ é:

$$z \text{ se } z \in S, \text{ e}$$

$$\gamma(z) \text{ se } z \in H$$

A Função Default $\delta : S \rightarrow 2^{S \cup H}$ define, para um determinado estado s , um conjunto de estados e símbolos de história que estão contidos nesse estado.

Se $x \in \delta(s)$ então: para $x \in S, x \in \rho^+(s)$, e $x \in H, \gamma(x) \in \rho^*(s)$

$\delta(s)$ é o conjunto *default* para s .

- Expressões: o conjunto de variáveis é denotado por V_p . O conjunto de expressões, V , é definido indutivamente como segue:
 - i. Se k é um número, então $k \in V$;
 - ii. Se $v \in V_p$, então $v \in V$;
 - iii. Se $v \in V$, então $\text{current}(v) \in V$;
 - iv. Se $v_1, v_2 \in V$ e op é uma operação algébrica, então $op(v_1, v_2) \in V$.

$cr(v)$ é a abreviação de $\text{current}(v)$.
- Condições: o conjunto de condições primitivas é denotado por C_p . O conjunto de condições, C , é definido indutivamente como segue:
 - i. $T, F \in C$; T, F correspondem a *true* e *false*, respectivamente;
 - ii. Se $c \in C_p$, então $c \in C$;
 - iii. Se $s \in S$, então $\text{in}(s) \in C$;
 - iv. Se $e \in E$, então $\text{not_yet}(e) \in C$;
 - v. Se $u, v \in V, R \in \{=, >, <, \neq, \geq, \leq\}$, então $u R v \in C$;
 - vi. Se $c \in C$, então $\text{current}(c) \in C$;
 - vii. Se $c_1, c_2 \in C$, então $c_1 \wedge c_2, c_1 \vee c_2, \neg c_1 \in C$.

$\text{ny}(e)$ é a abreviação de $\text{not_yet}(e)$, $cr(v)$ é a abreviação de $\text{current}(v)$.

- **Eventos:** o conjunto de eventos primitivos é denotado por E_p . O conjunto de condições, E , é definido indutivamente como segue:
 - i. $\lambda \in E$; λ é o evento nulo;
 - ii. Se $e \in E_p$, então $e \in E$;
 - iii. Se $c \in C$, então $\text{true}(c)$, $\text{false}(c) \in E$;
 - iv. Se $v \in V$, então $\text{changed}(v) \in E$;
 - v. Se $s \in S$, então $\text{exit}(s)$, $\text{entered}(s) \in E$;
 - vi. Se $e_1, e_2 \in E$, então $e_1 \wedge e_2$, $e_1 \vee e_2 \in E$;
 - vii. Se $e \in E$, $c \in C$, então $e[c] \in E$.

$\text{tr}(c)$, $\text{fs}(c)$ são abreviações de $\text{true}(c)$ e $\text{false}(c)$, respectivamente;

$\text{ch}(v)$ é abreviação de $\text{changed}(v)$;

$\text{ex}(s)$, $\text{en}(s)$ são abreviações de $\text{exit}(s)$ e $\text{entered}(s)$, respectivamente;

e é atômico se ele é da forma i-v.

- **Ações:** o conjunto de ações, A , é definido indutivamente como segue:
 - i. $\mu \in E$; μ é o evento nulo;
 - ii. Se $c \in C$, $d \in C$, então $c := d \in A$;
 - iii. Se $v \in V_p$, $u \in V$, então $v := u \in A$;
 - iv. Se $a_i \in A$, $i = 0, \dots, n$, então $a_0; \dots; a_n \in A$;

a é atômico se ele é da forma i-iii.

- **Rótulos:** o conjunto de rótulos, L , é o conjunto de pares $E \times A$, sendo que para $l = (e, a)$, $e \in E$, $a \in A$, escreve-se e/a . Informalmente, se e/a é o rótulo de uma transição t , então é ativada por e e a é executada quando t é disparada.
- **Transições:** o conjunto de transições, T , é definido como o conjunto de triplas $T \subset 2^S \times L \times 2^{S \cup H}$. A transição $t = (X, l, Y)$ é composta de um conjunto origem X , um conjunto destino Y , denotados por $\text{source}(t)$ e $\text{target}(t)$, respectivamente, e um rótulo l . Informalmente, se $l = e/a$, o sistema está em X e e ocorre, então t está habilitada e pode ser disparada, a é executada e o sistema vai então para Y .

4.3. Adaptações Necessárias à Visão Estocástica (Statecharts Estocásticos)

Para efeito de caracterização do problema, será especificado o funcionamento básico de um sistema de filas, através de eventos que acontecem indistintamente em um sistema de filas genérico.

Em um sistema de filas, há quatro eventos-padrão que caracterizam bem a dinâmica do sistema. O primeiro é o ato da geração dos clientes em uma determinada fonte, com um determinado ritmo. O segundo evento se constitui na chegada dos clientes à fila. O terceiro, na tomada do servidor por um determinado cliente, obedecendo a um determinado algoritmo de escalonamento. O último evento, pela saída do cliente do interior do servidor. É importante observar que todos esses eventos são tomados em um determinado instante, através de uma observação contínua do tempo. Assim, um evento fica bem caracterizado como uma função de estados de origem e de destino e do instante em que ele ocorre:

$$ev: f(s_i \rightarrow t_j), \quad \forall s_i \in S, t_j \in T, \text{ com } i, j = 1, K, n$$

Onde S é o conjunto discreto de estados, T é o conjunto contínuo dos tempos.

Com relação ao conjunto T, há um subconjunto dos instantes em que os eventos ocorrem, que pode ser descrito da seguinte maneira:

$$IT = \{\tau^g\} \cup \{\tau^{cf}\} \cup \{\tau^{ps}\} \cup \{\tau^{ss}\}, \quad IT \subset T.$$

Onde τ^g é o instante da geração, τ^{cf} é o instante de chegada à fila, τ^{ps} é o instante em que o cliente toma posse do servidor, τ^{ss} é o instante de saída do servidor. As diferenças entre os elementos $\tau_{i+1}^g - \tau_i^g$ e $\tau_i^{ss} - \tau_i^{ps}$ ($i=1,2,\dots,n$) descrevem as variáveis aleatórias *tempo entre chegadas* e *tempo de serviço*, respectivamente, as quais devem ser valores conhecidos *a priori* em qualquer avaliação de desempenho pretendida em sistemas de filas.

Pelas definições anteriores, pode-se observar que há sempre um tempo entre ocorrências de eventos, apesar de a ocorrência dos eventos, por definição, não demandar tempo, isto é, os eventos ocorrem de maneira imediata. O que há é um tempo entre as gerações e para o atendimento dos clientes, tarefas que acontecem em um determinado componente do sistema (por exemplo, um processador), o qual é representado em Statecharts (ou em algum outro diagrama orientado a estados) por um estado (ou um agrupamento deles).

Assim, da idéia de que tarefas são realizadas dentro de determinados estados, consumindo um certo tempo gera a proposição de estados que possuem tempos de permanência associados, mesmo que esse tempo seja igual a zero (caso dos estados que transicionam de maneira imediata). A notação gráfica, assumida neste trabalho, para diferenciar os estados com retardos e os estados imediatos é apresentada a seguir (Figura 4.1).

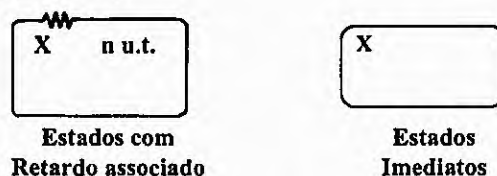


FIGURA 4.1. Estados com Retardo e Imediatos.

Apesar de utilizar a representação idêntica à sugerida por D. Harel (Harel, 1987) para estados com *delays* e *timeouts*, a semântica aqui é ligeiramente modificada, com o intuito de obter uma melhor adequação às necessidades das situações de sistemas de filas. Na semântica original, a partir da ocorrência de um evento, o estado dispara um temporizador implícito, que conta o número de unidade de tempo apresentado na inequação (conforme Figura 2.22, Capítulo 2). Apenas após decorrido esse tempo, o sistema abandona o estado, funcionando como um *timeout* para uma determinada atividade. Já na representação sugerida neste trabalho, o sentido é que tempo ou é uma média dos tempos entre chegadas (para o caso dos estados-fonte) ou ele é um tempo médio de serviço (para os casos de servidores), ambos assumidos como exponencialmente distribuídos. Assim, o tempo que o temporizador conta, a partir da entrada no estado, é uma das variáveis aleatórias que devem ser conhecidas previamente, como comentado anteriormente.

Essa nova abordagem dada ao *delay* traz em seu cerne uma complexidade que, à primeira vista, pode não estar muito aparente: a determinação da próxima configuração, levando-se em consideração que o passo não é mais a única unidade de tempo relacionada à execução dos Statecharts. A seção 4.3.3 apresenta as funções que determinam a próxima configuração a ser tomada pelo modelo, a partir da observação dos tempos nos estados não imediatos.

4.3.1. Escolha por Condição Probabilística

Em sistemas de filas, é necessário atribuir um valor probabilístico a cada possível caminho a ser seguido por um cliente. Essa circunstância leva à situação

na qual a escolha dentre os vários caminhos possíveis está atrelada a uma probabilidade, o que introduz a idéia de uma escolha realizada através desse parâmetro. A Figura 4.2 apresenta uma notação semelhante àquela utilizada para entrada por condição (seção 2.5.2, Capítulo 2), substituindo-se apenas a letra C pela letra P (de probabilidade).

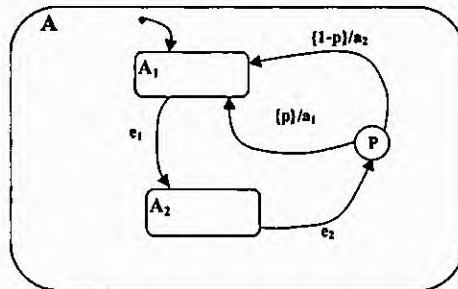


FIGURA 4.2. Escolha por probabilidade

A interpretação dada à Figura é que uma vez abandonado o estado A_2 , e_2 pode estar condicionado à probabilidade $\{p\}$ ou à $\{1-p\}$, e a partir do caminho escolhido, será disparada ou a ação a_1 ou a ação a_2 , em algum componente paralelo não especificado na Figura.

Formalmente, a condição probabilística é definida como:

- $\exists P = \{p_1, p_2, \dots, p_n\}$, onde P é o conjunto das probabilidades associadas aos eventos e p_i é cada ponto do espaço amostral definido por P : $p_i \in P$, com $i=1,2,\dots,n$;
- $0 \leq p_i \leq 1, \forall i = 1, 2, \dots, n$;
- $\sum_{i=1}^n p_i = 1$;
- Assim, admite-se que $P \subset C \therefore p_i \in C$.

Por extensão, se $c \in C$ e $p_i \in C$, então $ev(c)$ e $ev(p_i)$ são evento atrelado a uma condição c e evento condicionado a uma probabilidade p_i , respectivamente. Com o intuito de diferenciar a condição probabilística das demais, usa-se a notação particular $ev\{p_i\}$, em lugar da notação original $ev(c)$, conforme sugerido em (Vijaykumar, 1999).

4.3.2. A Semântica dos Statecharts: Passos, Configurações e Micro-configurações.

A semântica Statecharts tem como base uma seqüência de instantes de tempo $\{\sigma_i\} i \geq 0$, correspondente à taxa de execução do sistema. Por definição

(Harel et al., 1987), existe um conjunto de intervalos de tempos definido por $I = (\sigma_i, \sigma_{i+1})$, onde cada $\Delta\sigma$ ($\sigma_{i+1} - \sigma_i$) representa o tempo dispensado a um determinado passo i . O sistema irá reagir a cada final de intervalo $\sigma_i \rightarrow \sigma_{i+1}$, ou seja, em σ_{i+1} o sistema reagirá, apresentando uma nova configuração de estados.

Um estímulo externo, em σ_{i+1} , é uma tripla (Π, θ, ξ) , onde Π é um conjunto de eventos primitivos externos que ocorrem em I_i , θ é o conjunto de condições primitivas externas cujo valor é verdadeiro (true) em (σ_i, σ_{i+1}) , e ξ é uma função determinada pelo ambiente externo, de maneira que, para uma variável v , $\xi(v) = x$ se o valor de v for x em (σ_i, σ_{i+1}) . Assim, uma configuração de sistema associada ao instante σ_{i+1} é uma tupla (X, Π, θ, ξ) , onde X é a configuração de estados maximal do estado raiz e (Π, θ, ξ) é um estímulo externo associado a σ_{i+1} .

Uma reação do sistema é um par (Υ, Π^*) , onde Υ é o conjunto de transições denominado *passo* e Π^* é um conjunto de eventos atômicos gerados por Υ (Sugeta et al, 1999). Informalmente, um passo é um conjunto de transições que podem ser habilitadas e que são induzidas por estímulos externos, sendo que esse passo gera um conjunto de eventos que ocorrem como resultado das transições que o passo habilita. É importante ressaltar que todas as transições constantes em um passo Υ são disparadas simultaneamente.

Além da visão apresentada anteriormente, um passo pode também ser definido como uma seqüência de micro-passos, que gera configurações intermediárias (micro-configurações), sendo que cada micro-passo está contido em Υ . Para exemplificar a idéia do micro-passo, vai-se recorrer à especificação de um servidor de arquivos básico apresentado em (Francês et al., 2000), onde há os seguintes estados para representar os componentes do sistema (Figura 4.3).

A interpretação do modelo da figura será explicitada de maneira mais completa posteriormente (pois nela deve haver a inclusão de estados com retardo), mas a despeito da idéia de retardos em estados, será adotada aqui a premissa de que todas as configurações obedecem à noção original de passo, ou seja, se há transições que podem ser habilitadas, então os eventos relacionados a ela ocorrerão. Assim, admitindo-se que o evento jp (geração de *jobs*) pode ser executado, a ação inc_p (incremento da fila do processador) pode ser realizada no estado ortogonal $Proc_Q$. Desta forma, a segunda configuração Statecharts seria $SC2 = (Ready, Busy.Proc_Q, Idle.Proc, Idle.Disq_Q, Idle.Disc, Idle.Destination)$.

Logicamente, a primeira configuração (SC1) é constituída pelos estados *default*. Já a terceira configuração seria SC3=(Ready, Idle.Proc_Q, Busy.Proc, Idle.Disc_Q, Idle.Disc, Idle.Destination). Um micro-passo, que gera uma micro-configuração (uma configuração intermediária), poderia ser a ocorrência do evento $tr[not\ in\ (Idle.Proc_Q)]$ sem a execução da ação dec_p , que geraria uma configuração intermediária entre SC2 e SC3, tal que SC_i , onde $2 < i < 3$, não é igual à configuração SC2 e nem à sucessora SC3. SC_i seria uma configuração $SC_i=(Ready, Busy.Proc_Q, Busy.Proc, Idle.Disc_Q, Idle.Disc, Idle.Destination)$. No exemplo, está-se admitindo que só houve a geração de um *job* no período entre SC1 e SC3.

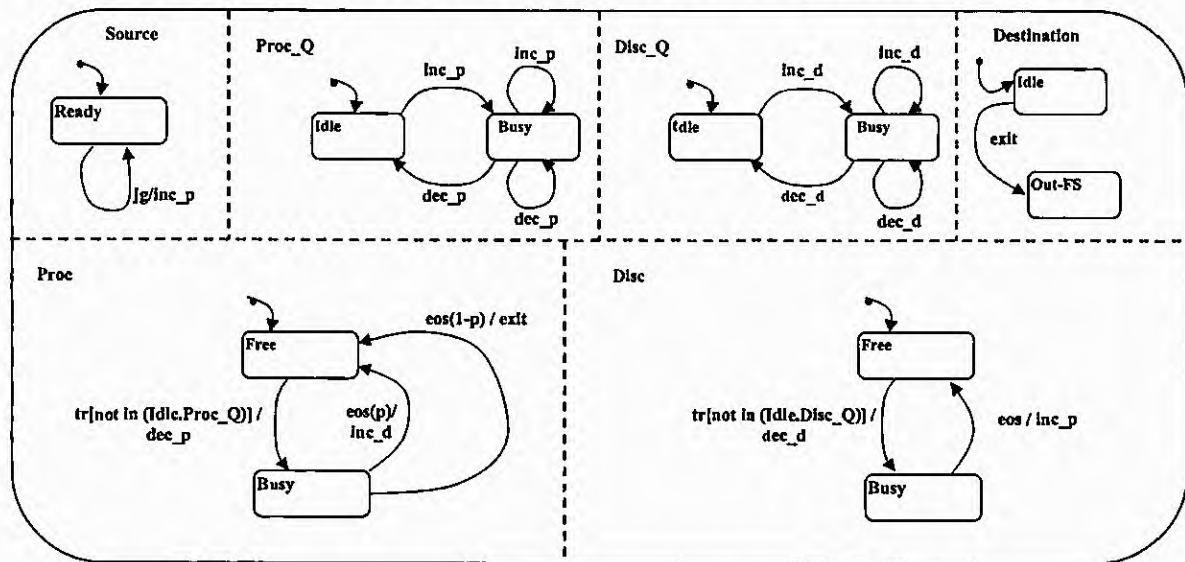


FIGURA 4.3. Servidor de Arquivos em notação Statecharts.

Também se assume que todos os estados são imediatos, por isso não há inconsistência entre as configurações e micro-configurações. Entretanto, pode-se imaginar a situação em que os estados *Busy* dos servidores (Proc e Disc) do modelo da Figura 4.3 possuem um determinado τ_i , que efetivamente representa valores de tempo e, possivelmente, valores diferentes, os quais são interpretados como uma média de uma determinada distribuição de probabilidade (aqui assumida como exponencial). Pode-se imaginar ainda que, para o exemplo, $\tau_{source} > \tau_{busy.proc} \wedge \tau_{source} > \tau_{busy.disc}$. Assim, em valores hipotéticos, $\tau_{source} = 5$ u.t., $\tau_{busy.proc} = 3$ u.t. e $\tau_{busy.disc} = 4$ u.t.

Nessas circunstâncias, SC1 ainda seria a dos estados *default*, SC2 seria (Ready, Busy.Proc_Q, Idle.Proc, Idle.Disc_Q, Idle.Disc, Idle.Destination), após o evento $!g$ e a ação inc_p e SC3 continuaria sendo (Ready, Idle.Proc_Q, Busy.Proc, Idle.Disc_Q, Idle.Disc, Idle.Destination). Entretanto, SC4 já geraria um conflito

estrutural de qual seria o evento responsável por disparar os micro-passos que compõem SC4. Se a transição que contém eos é habilitada, então o sistema poderá alcançar duas possíveis configurações SC4: (1) se a probabilidade p é admitida, então SC4 será (Ready, Idle.Proc_Q, Idle.Proc, Busy.Disc_Q, Idle.Disc, Idle.Destination); (2) se a probabilidade $1-p$ é admitida, então SC4 será (Ready, Idle.Proc_Q, Idle.Proc, Idle.Disc_Q, Idle.Disc, Out_FS). Além disso, se a transição habilitada for a que contém o evento jg , então a configuração SC4 seria (Ready, Busy.Proc_Q, Busy.Proc, Idle.Disc_Q, Idle.Disc, Idle.Destination).

Uma forma de manter a consistência na escolha da configuração sucessora é através da adoção de uma linha do tempo e de temporizadores em cada estado que possui um retardo associado. Assim, a escolha estaria condicionada ao menor de todos os tempos presentes no sistema, uma vez que aquele estado que transcorrer primeiro o seu tempo, poderá habilitar a(s) sua(s) transição(ões) de saída, imediatamente. A seguir, a dinâmica instituída por Harel (Harel et al., 1987) é apresentada. Após, as adaptações necessárias à abordagem estocástica são discutidas.

Formalmente (Harel et al., 1987):

- Definição 1: uma *configuração* de estados de s é um conjunto ortogonal relativo a s cujos membros são estados básicos, definida pela tupla (X, Π, θ, ξ) , conforme especificado anteriormente;
- Definição 2: um *passo* é uma seqüência maximal de micro-passos definido como: seja SC uma configuração do sistema, um passo Υ a partir de SC é uma seqüência $(\mu\Upsilon_0, \dots, \mu\Upsilon_m)$, em que:
 - i. $\mu\text{SC}_0 = \text{SC}$;
 - ii. $\mu\Upsilon_i$ é um micro-passo a partir de μSC_i ; para $i=0, \dots, m$;
 - iii. μSC_{i+1} é a micro-configuração do sistema alcançada por $\mu\Upsilon_i$ para $i=0, \dots, m$;
 - iv. O conjunto $\mu\Upsilon_0 \cup \dots \cup \mu\Upsilon_m$ é um conjunto estruturalmente consistente de transições;
 - v. Se t está habilitada numa determinada μSC_{m+1} , então $\{t\} \cup \mu\Upsilon_0 \cup \dots \cup \mu\Upsilon_m$ não é estruturalmente maximal, ou seja, a seqüência é maximal.

Um passo resulta em uma nova configuração do sistema, similar à última configuração alcançada pela seqüência de micro-passos que compõem o passo.

Desde que não haja transições a partir da última micro-configuração do sistema, o sistema espera durante um intervalo de tempo por um novo estímulo externo.

- **Definição 3:** Seja SC' uma configuração do sistema e $Y = (\mu Y_0, \dots, \mu Y_m)$ um passo a partir de SC' realizado no instante de tempo σ_i . $SC = (X, \Pi, \theta, \xi)$ é a *configuração do sistema alcançada* por Y se:
 - i. $X = \mu X_{m+1} \cup \mu Y_{m+1}$;
 - ii. Π é o conjunto de eventos primitivos que ocorrem no intervalo de tempo i ;
 - iii. Para $c \in C_p$, c se mantém em σ_i se e somente se $\mu SC_{m+1} \rightarrow c r(c)$, θ é o conjunto de condições primitivas que se mantém em (σ_i, σ_{i+1}) , para algum $\sigma \geq \sigma_i$;
 - iv. Para $v \in V_p$, o valor de v em σ_i é x se e somente se $\mu \xi_{m+1}(cr(v))=x$, $\xi(v) = x$ se o valor de v em (σ_i, σ_{i+1}) é x , para algum $\sigma \geq \sigma_i$.
- **Definição 4:** seja SC uma configuração do sistema no instante i e $Y = (\mu Y_0, \dots, \mu Y_m)$ um passo a partir de SC realizado em σ_i . O conjunto de *eventos gerados* por Y é $\Pi^* = \{e\}$ e gerado por μY_i para algum $i\} \cup \{e \mid \mu SC_{m+1} \rightarrow e, e \text{ atômico}\}$.
- **Definição 5:** uma *execução do sistema* é uma seqüência $\{(SC_i, Y_i, \Pi_i^*)\}_{i \geq 0}$, em que:
 - i. $SC_0 = (X_0, \Pi_0, \theta_0, \xi_0)$, X_0 é a configuração inicial de estados, e (Π_0, θ_0, ξ_0) são os estímulos externos que ocorrem no primeiro intervalo de tempo I_0 ;
 - ii. Y_i é um passo realizado a partir de SC_i no instante de tempo σ_{i+1} , para $i \geq 0$;
 - iii. SC_{i+1} é a configuração de sistema alcançada a partir de SC_i por Y_i , para $i \geq 0$;
 - iv. Π_i^* é o conjunto de eventos gerados por Y_i , para $i \geq 0$.

4.3.3. Redefinição de Passos, Configurações e Micro-configurações para Statecharts Estocásticos.

Neste ponto, volta à tona a discussão a respeito do caráter temporal dos passos e configurações, que, se negligenciado, poderá gerar um caráter de não-determinismo entre as sucessivas configurações do sistema. Pela definição de D. Harel (Harel et al., 1987), um sistema é não-determinístico em SC se há duas reações possíveis (Y_1, Π_1^*) e (Y_2, Π_2^*) , tal que $\Pi_1^* \neq \Pi_2^*$ ou $Y_1 \neq Y_2$.

Assim, uma configuração e um passo têm uma interpretação se todos os seus estados são considerados imediatos, e têm outra interpretação diante da presença de estados com retardos associados. Diante do exposto:

- **Definição 1a:** um estado é considerado *com retardo* quando existe uma variável τ , a qual quando avaliada determina :
 - i. O tempo médio entre chegadas de clientes, caso o estado seja a fonte geradora desses clientes (*Source*). Admitindo-se que os tempos entre chegadas são exponencialmente distribuídos;
 - ii. O tempo médio de serviço destinado aos clientes, caso o estado seja um servidor (em seu estado *Busy*). Admitindo-se que os tempos de serviço são exponencialmente distribuídos.
- **Definição 2a:** um estado é considerado imediato quando seu retardo é considerado zero ($\tau = 0$).
- **Definição 3a:** O tempo (σ_{i+1}) da próxima configuração alcançável $SC = (X, \Pi, \theta, \xi)$ obedece às seguintes premissas:
 - i. A variável avaliada em ξ é τ , onde τ é cada retardo associado a um estado s_i , com $i=1, \dots, n$;
 - ii. $\min(\tau)$ é a função que indica o menor dos tempos de retardo em uma determinada configuração SC_j , com $j=1, \dots, n$;
 - iii. O tempo gasto em cada passo é $\tau_{ij} = \tau_{i,j-1} + \min(\tau)$, com $j=1, \dots, n$. O tempo τ_{ij} é igual ao tempo final de uma configuração SC_i ou ao tempo inicial de uma configuração SC_{i+1} ;
 - iv. Se $j=1$, primeiro passo, então $\tau_{ij} = \min(\tau)$, pois $\tau_{i,j-1} = 0$;
 - v. $\tau_{i,resto} = \{\tau\} - \min(\tau)$, $\forall \tau \neq \min(\tau)$, onde $\tau_{i,resto}$ é o restante de tempo de retardo de um estado que extrapola para o próximo passo;
 - vi. $\tau_{total} = \sum_{j=1}^n \tau_{ij}$;
 - vii. $\forall \{\tau\} - \{\min(\tau)\}$, $\tau := \tau_{i,resto}$, a cada final de passo;
 - viii. Se $\tau = \min(\tau)$, então $\tau_{i,resto} = 0$ e τ no passo seguinte começa com valor 0.
- **Definição 4a:** uma escolha por probabilidade leva o sistema a reações diferentes $(Y_1, \Pi^*_1), (Y_2, \Pi^*_2), \dots, (Y_n, \Pi^*_n)$, tal que $\Pi^*_1 \neq \Pi^*_2 \neq \dots \neq \Pi^*_n$ ou $Y_1 \neq Y_2 \neq \dots \neq Y_n$.
- **Definição 5a:** se, em um Statechart, as suas configurações e o tempo de seus passos são determinados pelas definições de 1 a 4, então esse Statecharts é Estocástico.

Pelas definições anteriores e, utilizando o exemplo da Figura 4.3, agora na notação estendida para a visão estocástica (Figura 4.4), com os valores hipotéticos $\tau_{source} = 5$ u.t., $\tau_{busy.proc} = 3$ u.t. e $\tau_{busy.disc} = 4$ u.t, pode-se traçar uma linha do tempo para determinar o tempo de cada passo, que são o início e o fim de cada configuração. Além disso, pode-se determinar a ordem em que as configurações ocorrem.

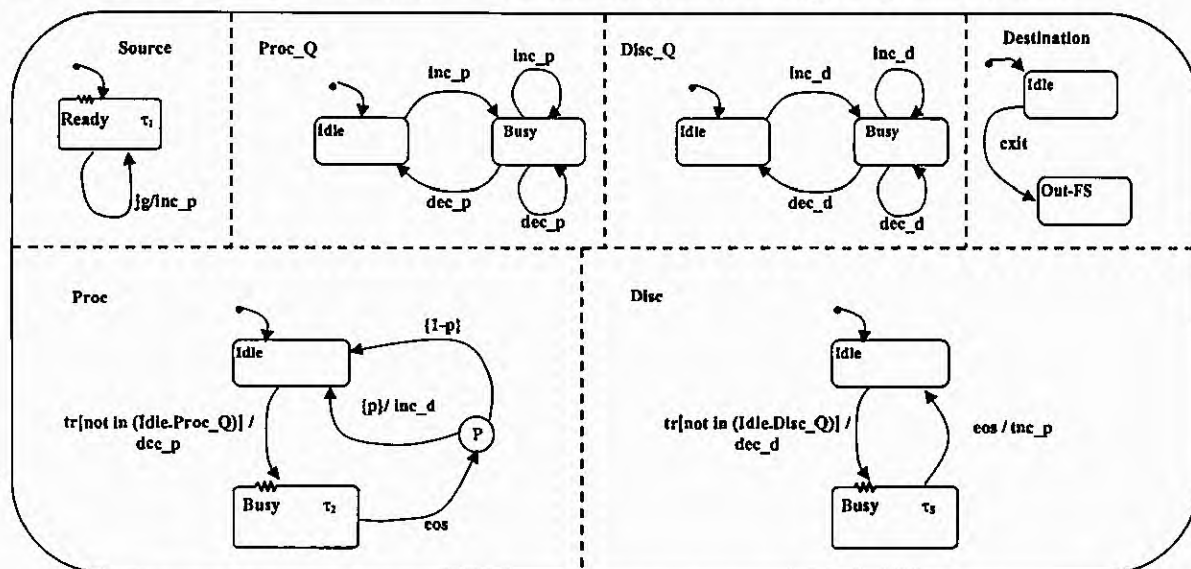


FIGURA 4.4. Servidor de Arquivos e seus Estados com Retardos Associados.

Seja SC1 a configuração dos estados *default* que apresenta o ingresso do sistema no estado *Source* (um estado com retardo de 5 u.t.), e seja SC2 a configuração sucessora, que é função da primeira, em relação ao tempo. Uma vez alcançado *Source*, o sistema “espera” por 5 u.t. até gerar o primeiro cliente (evento *tg*). Após decorridas as 5 u.t., o passo 1 é completado com a execução da ação *inc_p* (em *Proc_Q*), levando a *Busy.Proc_Q*. Na configuração seguinte, SC3 habilita de maneira imediata as transições dos eventos *tg* e *tr[not in(Idle.Proc_Q)]*, e alcança simultaneamente *Source* e *Busy.Proc*. O tempo do passo e a próxima configuração serão determinados pelas funções descritas anteriormente (de acordo com a visão estocástica), da seguinte maneira (sendo $\tau_1 = \tau_{source}$, $\tau_2 = \tau_{busy.proc}$, $\tau_3 = \tau_{busy.disc}$):

- Tempo do passo 1 é $\tau_{r1} = \min_{passo1} (\tau_{source}) = 5$, pois só há um estado com retardo (*Source*);
- $\min_{passo2} (\tau_{source}, \tau_{busy.proc}) = (5, 3) = 3$;
- Tempo do passo 2 é $\tau_{r2} = \tau_{r1} + \min(\tau_1, \tau_2) \Rightarrow \tau_{r2} = 5 + \min(5, 3) = 8$, ou seja, o próximo passo começa em 8 u.t.;
- O tempo que extrapola o tempo do passo 2 para os estados diferentes daquele que possui o tempo mínimo é $\tau_{i.resto} = \tau_i - \min(\tau_i) \Rightarrow \tau_{1.resto} = \tau_1 - \min(\tau_1, \tau_2) = 5 - 3 =$

2, isto é, o tempo que extrapola do passo 2 para o passo 3 é de 2 u.t. no estado *Source*;

- Quando o passo 2 completar 8 u.t. (ao seu término), então $\forall \{\tau_i\} - \{\min(\tau_i)\}$, $\tau_i = \tau_{i,resto}$, significando que os estados que não completaram o seu retardo em um determinado passo, começam o próximo com os seus tempos iguais ao restante do passo anterior. Para o exemplo, $\tau_1 = \tau_{1,resto} = 2$. A Figura 4.5 apresenta a linha do tempo, com os tempos de cada passo, de acordo com os valores obtidos anteriormente.

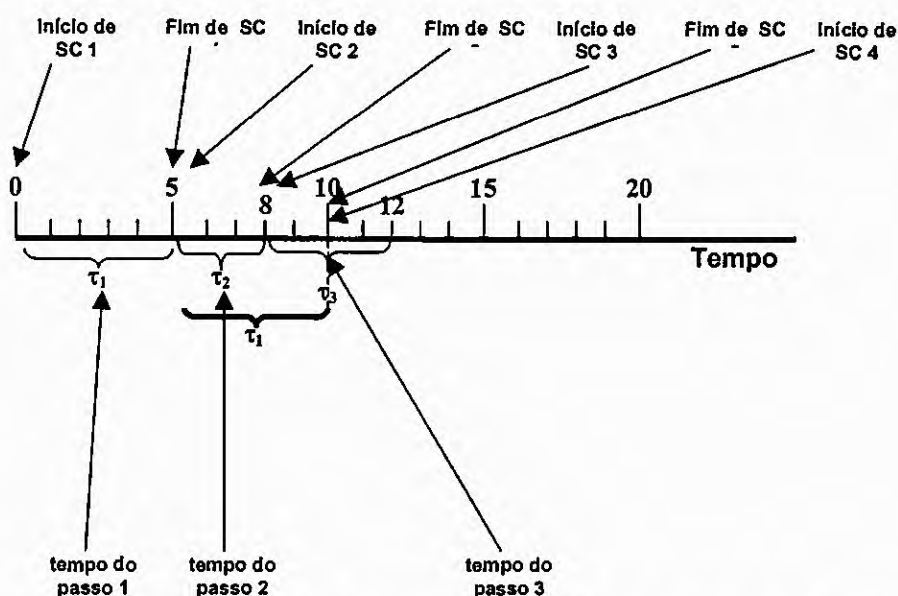


FIGURA 4.5. Linha do Tempo com suas Configurações e Passos.

Os valores temporais para SC3 são calculados como segue:

- $\min_{\text{passo2}} (\tau_{source}, \tau_{busy.disc}) = (2, 4) = 2$;
- Tempo do passo 3 é $\tau_{\gamma3} = \tau_{\gamma2} + \min(\tau_1, \tau_3) \Rightarrow \tau_{\gamma3} = 8 + \min(2, 4) = 10$, ou seja, o próximo passo começa em 10 u.t.;

O tempo que extrapola o tempo do passo 3 para os estados diferentes daquele que possui o tempo mínimo é $\tau_{i,resto} = \{\tau_i\} - \min(\tau_i) \Rightarrow \tau_{3,resto} = \tau_3 - \min(\tau_1, \tau_3) = 4 - 2 = 2$, isto é, o tempo que extrapola do passo 3 para o passo 4 é de 2 u.t. no estado *Busy.Disc*.

Para proceder-se admitindo-se as premissas anteriores, devem-se estabelecer algumas considerações. Primeiramente, é importante esclarecer onde se encontra o caráter aleatório da especificação e das funções admitidas. A aleatoriedade é intrínseca às distribuições de geração e de atendimento ao cliente, pois são essas distribuições que geram os ritmos de chegada e de serviço (variáveis

aleatórias que servem de entrada para a solução do sistema). As funções definidas como premissas garantem que os eventos possam manter a ordem esperada para um sistema de filas genérico, o que não prejudica em nada a abordagem estocástica.

Outro ponto a ser considerado é o fato de haver um certo padrão de eventos e de estados que representam os sistemas de filas, de um modo geral. Na verdade, há um padrão referente aos estados e eventos para representar sistemas de filas, apesar das possíveis variações de nomenclatura. A próxima seção apresenta um conjunto de *templates* que visa à representação mais uniforme desses sistemas característicos.

4.3.4. Templates e Eventos-Padrão para Sistemas de Filas.

Esta seção se destina a apresentar um conjunto de quatro *templates* e seus eventos-padrão para especificação de sistemas de filas. No contexto deste trabalho, *template* é um conjunto de estados (possivelmente unitário) que define o funcionamento básico de um determinado componente do sistema. A idéia é que, à exceção de algum parâmetro variável (por exemplo, valores dos tempos de serviço em um servidor), o *template* seja aplicável para um determinado componente, com o mínimo possível de modificação.

Para um sistema de filas simples, são estabelecidos os seguintes *templates* (Figuras 4.6 a 4.9):

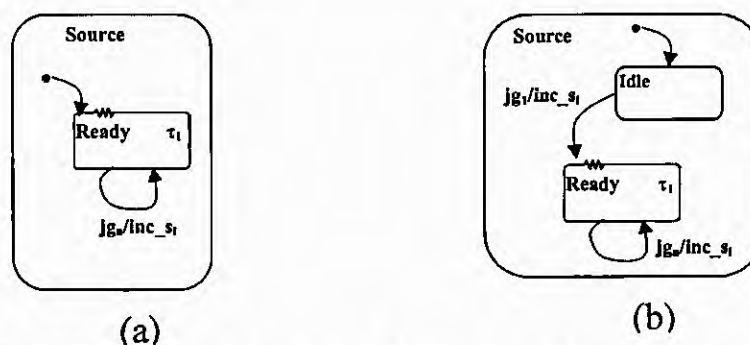


FIGURA 4.6. Templates para fontes geradoras de clientes.

No *template* (a), não há uma geração imediata do primeiro cliente, pois o estado *default* já é um estado com retardo, que uma vez alcançado, deve-se esperar transcorrer o τ_1 associado a ele. Já no *template* (b), há a geração de um cliente no instante zero, em virtude do estado *default* ser um estado imediato, o que não demanda tempo. Em ambos, o evento *yg* (*job generation*) é responsável pela

geração de um cliente, e a ação `inc_s` acrescenta o cliente gerado na fila de um servidor `s`. A escolha entre os dois depende exclusivamente das características do sistema em estudo.

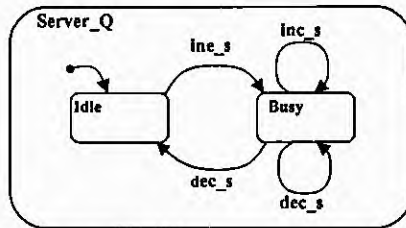


FIGURA 4.7. *Template* para filas de servidores.

O *template* fila é composto por dois subestados: `Idle` e `Busy`. Por *default*, a fila se encontra vazia (`Idle`), e a cada geração de cliente, há um acréscimo unitário na fila, levando o estado para `Busy`. Ocorrências reiteradas do evento `inc_s` mantêm a fila em `Busy`, assim como reiterações de `dec_s` decrementam a fila até o valor limite unitário, a partir do qual, a fila muda para `Idle`, indicando a ausência de clientes. Vale ressaltar que um decréscimo na fila indica que um cliente alcançou o servidor.

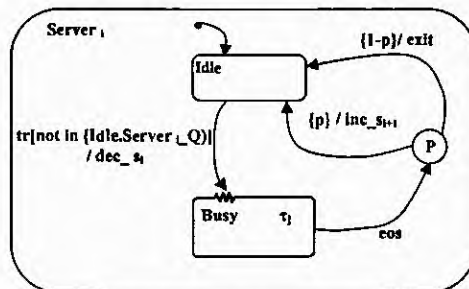
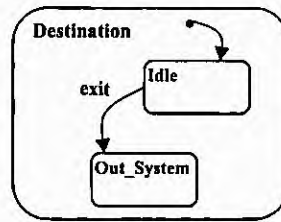


FIGURA 4.8. *Template* para Servidores.

O *template* para servidores é composto por dois subestados: `Idle` e `Busy`, com os significados de certa forma óbvios, ressaltando-se que em `Busy` há um retardo de τ_s , que significa um valor médio de uma distribuição de probabilidade. Se houver algum cliente na fila, o evento `tr[not in (Idle.Server_Q)] / dec_s` pode ser executado e sua transição é habilitada e a fila decrementada, em virtude de ter ido um cliente para o servidor. Após o atendimento em `Busy`, `eos` pode ser executado através de uma de suas condições probabilísticas. A ida a outro servidor implica envio de um cliente para a fila desse servidor (`inc_{s+1}`), e, caso contrário, o cliente abandona o sistema. É importante observar que pode haver mais de duas escolhas possíveis, cada uma com a sua respectiva probabilidade, assim como pode existir apenas uma escolha possível, onde se faz desnecessária a utilização da escolha por probabilidade.

FIGURA 4.9. *Template* para Sorvedouros.

O *template* de sorvedouro (Destination) é composto por dois subestados, que indicam que o sorvedouro está à espera de alguma resposta (Idle) ou que ele recebeu a resposta, podendo então abandonar o sistema.

De maneira a uniformizar também os eventos interessantes à avaliação de desempenho, alguns eventos são padronizados na especificação baseada nos *templates* sugeridos anteriormente, o que não impede que se criem outros eventos que aumentem a clareza da especificação, quando isso for necessário. A Tabela 4.1 apresenta o rótulo dos eventos, além da semântica atribuída a cada um deles.

TABELA 4.1. Eventos-Padrão e suas Semânticas.

Rótulo do Evento	Semântica Atribuída
tg (<i>Job Generation</i>)	Geração de um cliente obedecendo a uma determinada distribuição de probabilidade.
inc_S (<i>Increment of Server Queue</i>)	Incrementa a fila de um determinado servidor S.
dec_S (<i>Decrease of Server Queue</i>)	Decrementa a fila de um determinado servidor S.
tr[not in (Idle.Server_Q)]	Assegura que a fila do servidor não está vazia, e que próximo cliente pode ir ao servidor.
eos {p} (<i>End of Service</i>)	Indica o término do atendimento a um cliente, e uma escolha probabilística para determinar o caminho a ser seguido pelo cliente.
exit	Indica que o cliente sai do sistema que o provê de um determinado serviço e vai até um sorvedouro.

Além dos eventos estarem padronizados, também há uma necessidade de se padronizar os nomes de estados, de maneira que cada estado seja único no sistema. A Tabela 4.2 apresenta um conjunto de nomes de estados possíveis.

TABELA 4.2. Nomes de Estados-Padrão e suas Semânticas.

Nome de Estados	Semântica Atribuída
Source	Estado que representa um componente genérico do sistema responsável pela geração de um cliente, obedecendo a uma determinada distribuição de probabilidade.
Server_Q	Estado que representa a fila de um determinado servidor. Apesar da fila ser representada por apenas um lugar para armazenamento, admite-se a possibilidade da fila infinita, por conveniência da fundamentação matemática.
Server	Estado que representa o componente responsável pela prestação de um determinado serviço do sistema. Assim como acontece em <i>Source</i> , o serviço prestado obedece a uma distribuição de probabilidade.
Destination	Estado que representa o componente do sistema que absorve os clientes após a execução de um serviço.

Os estados da Tabela 4.2 são, em sua totalidade, superestados que contêm uma especificação mínima para o entendimento do funcionamento do componente que cada superestado representa. O refinamento da especificação apresentada fica à mercê da abordagem que se pretende atribuir ao modelo.

4.4. Queuing Statecharts

Esta seção se destina a apresentar uma alternativa a uma deficiência da especificação dos Statecharts: a dificuldade de se representar o caminho linear que um determinado cliente traça durante a sua passagem pelo sistema. Essa representação é suficientemente coberta pela especificação de redes de filas, através da ligação seqüencial estabelecida entre os componentes do modelo.

Não obstante, redes de filas não representam adequadamente a complexidade que os servidores necessitam para que suas descrições se tornem esclarecedoras, o que pode ser feito de forma bastante natural em Statecharts.

Assim, para representar sistemas de filas, redes de filas e Statecharts possuem representações complementares.

Partindo-se da premissa anterior, propõe-se uma aglutinação entre a especificação de redes de filas e a de Statecharts, visando a prover uma especificação mais completa para sistemas de filas.

4.4.1. Definição de Queuing Statecharts e seus Templates

Queuing Statecharts (QS) são definidos, através de seus componentes, conforme a seguir:

Centro de Serviço (Service Center):

- Definição 1b: o estado s' é um servidor se e somente:
 - i. $\exists S^s \subset S$, que é o conjunto de todos os servidores do QS, tal que todo $s' \in S^s$;
 - ii. $s' \in S \wedge \rho(s') = 2 = \{\text{Idle.Server}, \text{Busy.Server}\}$;
 - iii. $\psi(s') = \text{OR}$;
 - iv. $\delta(s') = \{\text{Idle.Server}\}$.
 - v. As ações de incremento e decremento ($\text{inc}_$, $\text{dec}_$) são sempre realizadas por um servidor s' em uma fila s'' .
- Definição 2b: o estado s'' é uma fila, se e somente se:
 - i. $\exists S^q \subset S$, que é o conjunto de todas as filas do QS, tal que todo $s'' \in S^q$;
 - ii. $s'' \in S \wedge \rho(s'') = \emptyset \therefore s''$ é um estado básico;
 - iii. s'' possui uma variável Q_j (clientes na fila), com $j=1,2,\dots,k$, que representa a variável aleatória número de clientes na fila;
 - iv. Q_j é definida como:
 - (a) $Q_j \in V$, op é uma operação algébrica, então $op(Q_j) \in V$;
 - (b) $Q_j, v \in V, R \in \{=, >, <, \neq, \geq, \leq\}$, então $v R Q_j \in C$.
 As únicas operações realizadas em Q_j são: $Q_j := Q_j + w$, $Q_j := Q_j - w$, com $w=1, 2, \dots, n$.
- Definição 3b: o estado s é um centro de serviço, se e somente se:
 - i. $s \in S \wedge \rho(s) = 2 = \{\text{Server}_Q, \text{Server}\}$, onde $(S^s \cup S^q) \subset S$;
 - ii. $\psi(s') = \text{AND} \therefore \text{Server}_Q \perp \text{Server}$;
 Ou seja, $\rho(s) \neq \emptyset$ e $\psi(s') = \text{AND}$, então trata-se de uma decomposição AND de s ;

- **Definição 4b:** s^o é um estado fonte (Source) se e somente se:
 - $s^o \in S \wedge \rho(s^o) = 1 = \{\text{Ready}\}$;
 - Ready ($\in s^o$) possui uma variável τ_i , que representa a variável aleatória tempo entre geração de clientes, obedecendo a uma distribuição de probabilidade exponencial;
- **Definição 5b:** os é um estado sorvedouro (Destination) se e somente se:
 - $^os \in S \wedge \rho(^os) = \emptyset \therefore ^os$ é um estado básico.

Além das definições anteriores, novamente são estabelecidos *templates* para cada componente definido de 1 a 5. As Figuras 4.10, 4.11, 4.12, 4.13, 4.14 e 4.15 apresentam, respectivamente *templates* para fonte, fila, servidor, vetor de servidores, centro de serviço e sorvedouro.

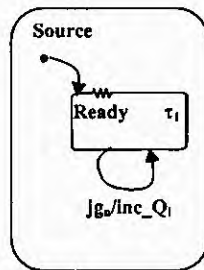


FIGURA 4.10. *Template* para Fonte (Source).



FIGURA 4.11. *Template* para Fila (Queue).

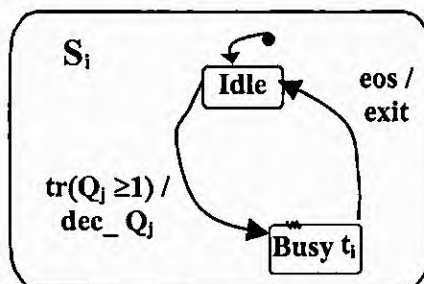


FIGURA 4.12. *Template* para Servidor (Server).

Em certas situações, há a necessidade de se representar uma série de estados que possuem estruturas internas idênticas. Para esses casos, a especificação pode se valer de vetores de estados, condensados em um único estado com parâmetros, denominado de estado parametrizado. A Figura 4.13

apresenta um grupo de processadores idênticos, que estão agrupados em um estado parametrizado *Proc*, cujo parâmetro indica o número de componentes do conjunto.

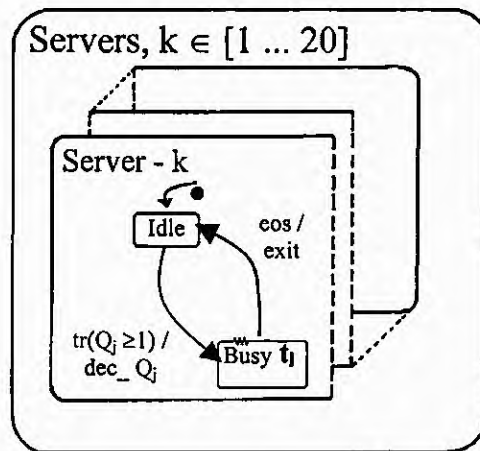


FIGURA 4.13. *Template* para Estados Parametrizados.

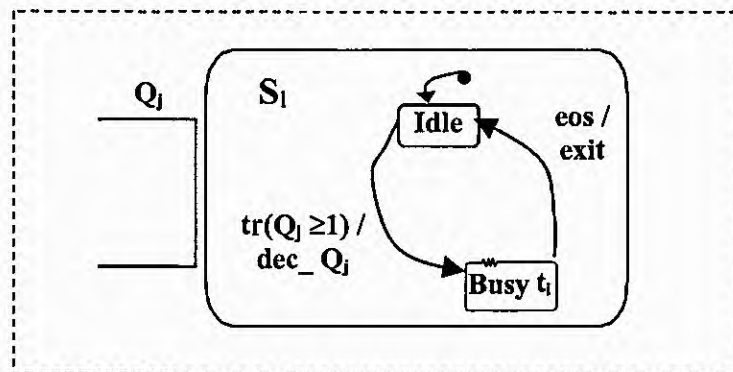


FIGURA 4.14. *Template* para Centro de Serviço (Service Center).

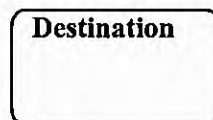


FIGURA 4.15. *Template* para Sorvedouro (Destination).

Assim como nos Statecharts Estocásticos, para Queuing Statecharts também há um conjunto de eventos-padrão, usado para definir o comportamento básico de um sistema de filas. A Tabela 4.3. apresenta o conjunto de eventos definidos para Queuing Statecharts.

Há a manutenção de todas as definições apresentadas anteriormente, tanto aquelas definidas em (Harel et al., 1987), quanto as definidas na seção (4.2), na qual são especificadas as diretrizes para os Statecharts Estocásticos. Desta forma, as definições de escolha por probabilidade, estados com retardos associados, além

das redefinições de Passos, Configurações e Micro-configurações para Statecharts Estocásticos são aqui também assumidas.

TABELA 4.3. Eventos-Padrão de Queuing Statecharts e suas Semânticas.

Rótulo do Evento	Semântica Atribuída
$ \text{jg} \text{ (Job Generation)} $	Geração de um cliente obedecendo a uma determinada distribuição de probabilidade.
$ \text{inc_}Q_j \text{ (Increment the number of clients in the Server Queue)} $	Incrementa a variável aleatória que representa o número de clientes na fila de um determinado servidor S_j .
$ \text{dec_}n_j \text{ (Decrease the number of clients in the Server Queue)} $	Decrementa a variável aleatória que representa o número de clientes na fila de um determinado servidor S_j .
$ \text{tr}(Q_j \geq 1) $	Assegura que a fila do servidor não está vazia, e que próximo cliente pode ir ao servidor.
$ \text{eos } \{p\} \text{ (End of Service)} $	Indica o término do atendimento a um cliente, e uma escolha probabilística para determinar o caminho a ser seguido pelo cliente.
$ \text{exit} $	Indica que o cliente sai do sistema que o provê de um determinado serviço e vai até um sorvedouro.

A Figura 4.16 apresenta o servidor de arquivos das Figuras 4.3 e 4.4, agora utilizando a notação de Queuing Statecharts.

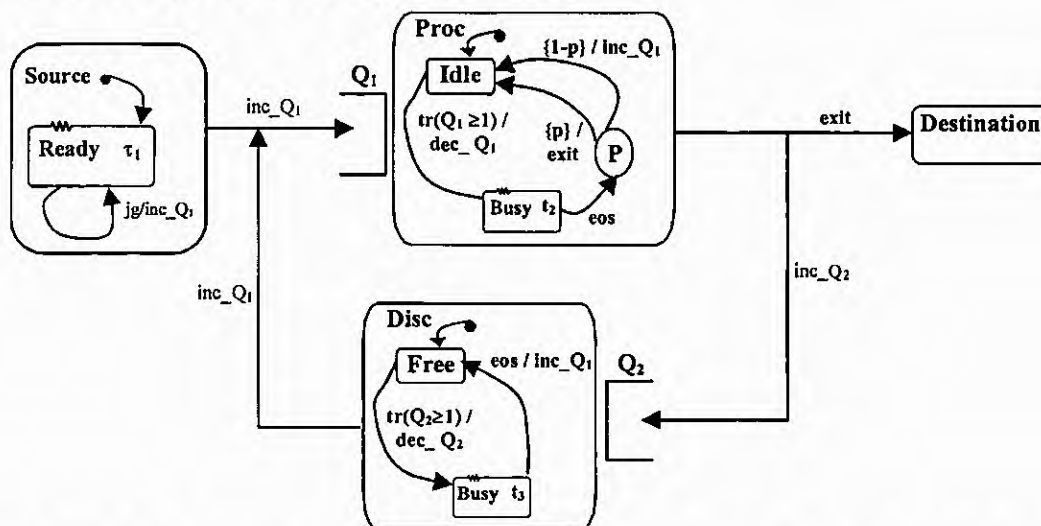


FIGURA 4.16. Servidor de Arquivos em Representação Queuing Statecharts.

Com a especificação de Queuing Statecharts há uma aglutinação entre características interessantes tanto de redes de filas quanto de Statecharts. Queuing Statecharts é um meio-termo entre a especificidade das redes de filas e o caráter generalista dos Statecharts.

4.4.2. Templates para as Distribuições Hiperexponencial e de Erlang

Como foi apresentado no Capítulo 3 (seção 3.4.8), as distribuições hiperexponencial e de Erlang são “arranjos” feitos a partir da distribuição exponencial. Essas duas distribuições também são usadas para descrever tempos de serviço, como sugerido em (Prado, 1999).

Esta seção formaliza dois *templates* para servidores (um para a hiperexponencial e outro para a de Erlang), com o objetivo de fornecer um leque maior de possibilidades para descrever tempo de serviço, utilizando-se a especificação proposta neste trabalho.

□ Servidor Hiperexponencial

- Definição 1c: o estado s_H é um servidor hiperexponencial se e somente:
 - i. $\exists S^s \subset S$, onde S^s é o conjunto de todos os servidores do QS, tal que todo $s_H \in S^s$;
 - ii. $s_H \in S \wedge \rho(s_H) = 2 = \{\text{Idle}.s_H, \text{Busy}.s_H\}$;
 - iii. $\rho(\text{Busy}.s_H) = I = \{B_1, B_2, \dots, B_i\}$, com $i = 1, 2, \dots, n$;
 - iv. $\psi(s_H) = \text{OR}$; $\psi(\text{Busy}.s_H) = \text{OR}$;
 - v. $\delta(s')$ é uma escolha por condição probabilística P , onde a cada arco está associada uma probabilidade p_i (com $i = 1, 2, \dots, n$) e $\sum_{i=1}^n p_i = 1$.

O índice i define o grau (ou número de estágios) da distribuição hiperexponencial.

No servidor hiperexponencial, um cliente tem uma probabilidade p_i de ser atendido no estado B_i , com um tempo de serviço t_i (exponencialmente distribuído). Na Figura 4.17, se houver algum cliente na fila ($tr(Q_i \geq 1)$), a fila é decrementada (dec_Q) e um cliente é escalonado (com uma probabilidade p_i) para atendimento em um dos estados B_i , com um tempo de serviço t_i . Após o atendimento (eos), o cliente libera o servidor e, no caso de 4.17, incrementa a fila de outro servidor (inc_Q_{j+1}).

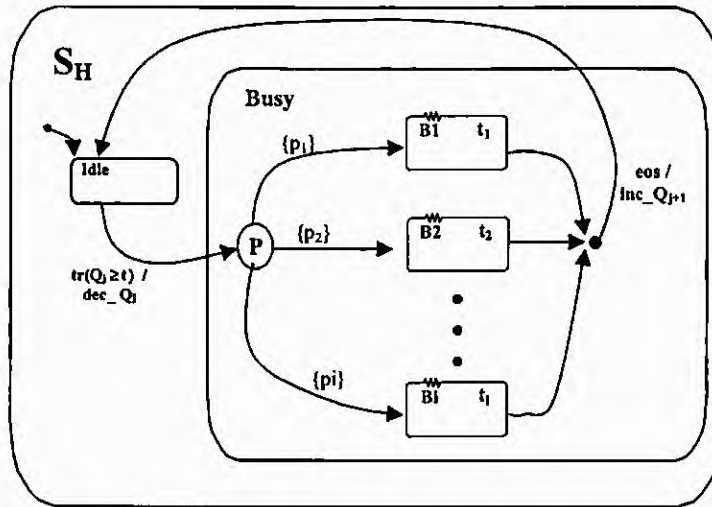


FIGURA 4.17. *Template* para Servidor Hiperexponencial.

- **Definição 1d:** o estado s_E é um servidor de Erlang se e somente:
 - vi. $\exists S^s \subset S$, onde S^s é o conjunto de todos os servidores do QS, tal que todo $s_E \in S^s$;
 - vii. $s_E \in S \wedge \rho(s_E) = 2 = \{\text{Idle}.s_E, \text{Busy}.s_E\}$;
 - viii. $\rho(\text{Busy}.s_E) = i = \{B_1, B_2, \dots, B_i\}$, com $i = 1, 2, \dots, n$;
 - ix. $\psi(s_E) = \text{OR}$; $\psi(\text{Busy}.s_E) = \text{OR}$;
 - x. $\delta(s_E) = \{\text{Idle}.s_E\}$; $\delta(\text{Busy}.s_E) = \{B_1\}$;

O índice i define o grau (ou número de estágios) da distribuição de Erlang.

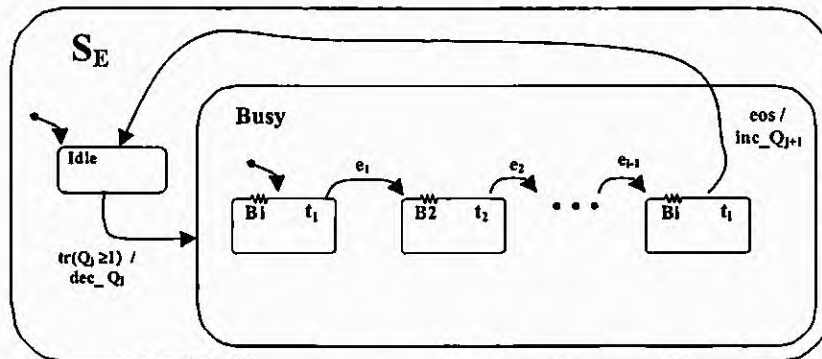


FIGURA 4.18. *Template* para Servidor de Erlang.

Na figura 4.18, se há pelo menos um cliente na fila ($tr(Q_i \geq 1)$), ele é escalonado para ocupar o servidor, sendo atendido por i unidades B , cada uma com tempos de serviço exponencialmente distribuídos. Ao final do atendimento, o cliente libera o servidor e parte para a fila de um outro servidor (inc_Q_{j+1}).

4.5. Considerações Finais

Este capítulo apresentou um panorama das possibilidades de usar-se Statecharts para prover avaliação de desempenho, semelhante àquela já realizada pelas outras técnicas abordadas neste trabalho (redes de filas e de Petri).

A idéia é oferecer um alternativa viável e formal que insira no contexto da avaliação de desempenho as características interessantes intrínsecas aos Statecharts. Essa alternativa foi primeiramente apresentada como uma extensão que se baseia fundamentalmente na especificação Statecharts (Statecharts Estocásticos), na qual apenas foram adicionadas definições novas necessárias à abordagem estocástica. Em um segundo momento, foi apresentada uma aglutinação entre as características positivas das redes de filas e dos Statecharts, no que concerne à representação de sistemas de filas. Essa segunda abordagem é denominada de Queuing Statecharts.

Em ambas as extensões, há a oferta das características positivas dos Statecharts na representação de sistemas complexos: hierarquia, paralelismo, *broadcasting* de comunicação, entre outros. Essas características são mantidas na íntegra nos Statecharts Estocásticos, assim como também são mantidas algumas limitações dessa representação (o caminho seqüencial dos clientes no sistema, mais enfaticamente). Em contrapartida, essa restrição é amenizada em Queuing Statecharts.

É importante ressaltar que Queuing Statecharts (QS) é uma proposição para um domínio de aplicação bastante específico: sistemas de filas. Para esse propósito, QS podem ser uma representação mais natural. Contudo, também vale ratificar a abrangência da especificação dos Statecharts (e de seu subconjunto: Statecharts Estocásticos), que permite a representação de componentes quaisquer, além de filas e servidores.

Capítulo 5

Soluções para Modelos Statecharts

5.1. Considerações Iniciais

Este capítulo apresenta algumas soluções possíveis para sistemas de filas, ilustrando tanto abordagens analíticas quanto por simulação. Primeiramente, são expostos alguns métodos teóricos¹ e, em particular, as redes de Jackson são discutidas com mais detalhes. Além disso, algumas das peculiaridades que envolvem a solução por simulação são discutidas.

Com o objetivo de aplicar ambas as soluções nas especificações propostas anteriormente, é apresentada uma associação entre essas especificações e as soluções de Jackson e simulação simpl. Para tanto, um modelo de um servidor de arquivos, já discutido no Capítulo 4, é novamente utilizado, agora parametrizado com taxas de chegada e de serviços hipotéticas, a partir do qual são feitas inferências e comparações em relação às medidas de desempenho obtidas.

Pelo fato de não haver consenso entre ambas as correntes de solução (analítica e simulação), também é realizada uma breve exposição de vantagens e desvantagens de cada uma delas, com um objetivo apenas informativo.

A título de esclarecimento, é importante observar que não há uma nomenclatura padrão entre os autores da área. Assim, na maioria das obras, são usadas indistintamente as expressões "modelos analíticos" e "modelos de simulação", referindo-se às possíveis soluções. Neste trabalho, entretanto, recorre-se à nomenclatura sugerida em (Santana et al., 1997), onde o modelo se apresenta de forma independente de sua resolução (esta sim podendo ter uma conotação analítica ou por simulação).

Neste trabalho, então, usam-se as expressões "solução analítica" e "solução por simulação", em detrimento das anteriormente citadas.

¹ Métodos (ou soluções) teóricos é uma expressão usada como sinônimo de métodos (ou soluções) analíticas em algumas obras pertinentes à área, tais como (Kant, 1992) e (Allen, 1990).

5.2. Solução Analítica

Solucionar um modelo implica obter medidas de desempenho, tais como utilização e tempo de resposta, a partir de uma parametrização inicial (geralmente, taxas de chegadas e de serviços).

Especificamente, a solução analítica é a abordagem que requer que equações sejam resolvidas, utilizando-se algum método matemático (tipicamente, sistemas lineares e matrizes), para que se obtenham medidas de desempenho como resposta.

A solução analítica pressupõe que o sistema modelado esteja em equilíbrio. Nesse contexto, encontrar-se em equilíbrio também implica que o sistema seja estável, isto é, com $\rho < 1$, onde ρ representa a utilização, que pode ser definida por

$\rho = \frac{\lambda}{c\mu}$. Há uma relação diretamente proporcional da utilização (ρ) em relação à

taxa de chegada (λ) e inversamente proporcional ao fator ($c\mu$). Dessa forma, para que $\rho < 1$ seja verdadeiro, $\lambda < c\mu$, ou, em outras palavras, a taxa de chegada ao sistema deve ser menor que a capacidade desse sistema atender os clientes. Neste trabalho, nos casos estudados através da solução analítica, sempre se está admitindo que o sistema alcança o equilíbrio.

R. Jain (Jain, 1991) e A. Allen (Allen, 1990) relacionam os métodos analíticos de solução de acordo com o fato de a rede ser aberta ou fechada. A Figura 5.1, extraída de (Silva, 2000), apresenta alguns desses métodos a partir do critério anterior (rede aberta ou fechada).

Especificamente neste trabalho, será usado o método das redes de Jackson para modelos abertos, entretanto alguns dos métodos (tanto para modelos abertos quanto para fechados) são apresentados sucintamente no Apêndice C.

Observando-se a Figura 5.1, pode-se coligir que os processos de Markov² (discutidos no Capítulo 3) não fazem parte do conjunto de soluções analíticas. Por isso é importante ressaltar que o processo de nascimento-e-morte é um caso particular dos processos markovianos, no qual uma transição só pode ser feita para um estado vizinho (um estado n só pode transicionar para o estado $n+1$ ou para $n-1$). O processo de Poisson também possui a propriedade de Markov. O processo de nascimento-e-morte é apresentado no Apêndice A.

² Processos que têm um espaço de estados discreto e que possuem a propriedade da falta de memória.

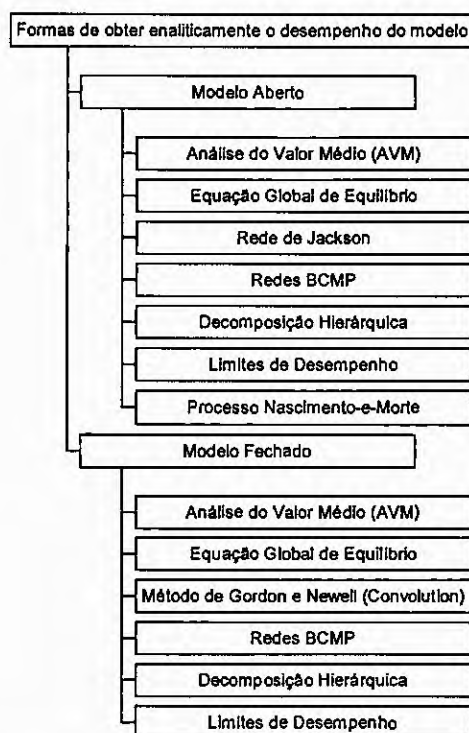


FIGURA 5.1. Algumas Possíveis Abordagens para a Solução Analítica.

5.2.1. Redes de Jackson

James R. Jackson, em (Jackson, 1957), definiu alguns tipos de redes, para as quais ele enunciou o seguinte teorema, aqui apresentado sem prova.

Teorema de Jackson: Supondo-se um modelo de rede de filas com M nós que satisfaz às seguintes condições:

1. Cada nó consiste de c_i servidores exponenciais idênticos, cada um com taxa média de serviço μ_i , onde $i = 1, 2, \dots, M$.
2. Os clientes chegam ao nó i provenientes do lado de fora do modelo em um padrão *Poisson* (exponencial) com taxa média de chegada λ_i (os clientes também podem chegar ao nó i através de outros nós do modelo), para $i = 1, 2, \dots, M$.
3. Uma vez atendido no nó i , um cliente parte, instantaneamente, para o nó j , onde $j=1,2, \dots,M$, com probabilidade p_{ij} ; ou deixa o modelo com probabilidade $1 - \sum_{j=1}^M p_{ij}$.

Os modelos que satisfazem a essas condições são denominados de *redes de Jackson*.

Para cada nó i de uma rede de Jackson, a taxa média de chegada (ou taxa média de entrada) para o nó, Λ_i , é dada por (Allen,1990):

$$\Lambda_i = \lambda_i + \sum_{j=1}^M p_{ji} \cdot \Lambda_j$$

Além disso, se $p(n_1, n_2, \dots, n_m)$ denota a probabilidade do estado estacionário de que há n_i clientes no i -ésimo nó, para $i = 1, 2, \dots, M$, e se $\Lambda_i < c_i \cdot \mu_i$, para $i = 1, 2, \dots, M$, então:

$$p(n_1, n_2, \dots, n_m) = p_1(n_1) p_2(n_2) \dots p_M(n_M)$$

Onde, $p_i(n_i)$ é a probabilidade do estado estacionário de que há n_i clientes no i -ésimo nó, se ele é tratado como um modelo $M/M/c_i$ com uma taxa média de chegada Λ_i e tempo médio de serviço $1/\mu_i$ para cada um dos c_i servidores. Além disso, cada nó se comporta como se fosse um modelo $M/M/c_i$ independente, com taxa média de chegada Λ_i (Allen,1990). No caso em que todos os nós do modelo são filas de servidores únicos, cada nó se comporta como se fosse uma fila $M/M/1$ independente.

A associação feita entre a especificação Queuing Statecharts e a solução de Jackson é apresentada na seção 5.4.1.

5.3. Solução por Simulação

A solução por simulação é uma alternativa atrativa em relação à analítica. Apesar de ser uma aproximação, pois ela tenta reproduzir o comportamento de um sistema real, e a partir dessa reprodução, é realizada uma interpretação das estatísticas obtidas durante a execução do programa.

Neste trabalho, a solução é obtida adotando-se a extensão funcional `smpi`³, embora, os eventos e estados propostos nas especificações Statecharts discutidos nesta tese são de caráter geral para sistemas de filas, o que possibilita que outras extensões funcionais ou ambientes de simulação (que possuam orientação a eventos) possam ser adotados.

O processo que constitui a simulação é composto das fases apresentadas a seguir.

³ Visto com mais detalhes na seção 5.3.4

5.3.1. Fases de uma Simulação.

Segundo L. F. G. Soares (Soares, 1992), as seguintes fases podem ser identificadas em um processo de simulação:

1. Estudo do Sistema e Definição dos Objetivos.
2. Construção do Modelo.
3. Determinação dos Dados de Entrada e Saída.
4. Tradução do Modelo.
5. Verificação.
6. Validação.
7. Experimentação.
8. Análise dos Resultados.
9. Documentação.

O marco zero do processo de uma simulação é o estudo e entendimento do sistema considerado. O sistema deve ser definido com precisão e as metas pretendidas devem ser claras e objetivas. Um mau entendimento dos objetivos do sistema e da simulação provavelmente poderá levar todo o processo ao insucesso.

A segunda etapa do processo é a construção do modelo. Três técnicas para construção de modelos foram abordadas neste trabalho: redes de filas, redes de Petri e Statecharts. Essas técnicas, basicamente, possibilitam uma ampla abstração do sistema real, além de proporcionar uma maneira de solução do modelo. A grande complexidade dessa fase é fazer com que o modelo (sendo elaborado sob o ponto de vista do modelador) expresse realisticamente as características essenciais do sistema real.

Uma vez construído o modelo, devem-se gerar os dados de entrada que o alimentarão. Essa fase de formulação, via de regra, utiliza valores de entrada inicialmente hipotéticos ou baseados em alguma análise preliminar. Entretanto, nada impede que esses dados sejam modificados, tendo como base fatores considerados válidos pelo modelador. Porém, independente do juízo adotado para escolha dos dados de entrada, eles podem obedecer, fundamentalmente, a dois critérios: os dados podem ser vistos como *Estocásticos* (Probabilísticos) ou *Determinísticos* (seqüência de passos de trabalho).

Após a formulação dos dados de entrada, deve-se traduzir o modelo para uma forma aceitável pelo computador. Vários software de simulação estão

disponíveis para realizar esse propósito. A gama vai de extensões funcionais de linguagens de programação convencionais a linguagens específicas ou pacotes de simulação.

As duas fases seguintes são, de certa maneira, complementares. A quinta etapa (de verificação) determina se o modelo traduzido para o computador executa como desejado (a verificação é chamada por alguns autores de depuração, por assegurar que o modelo faz o que era esperado). Algumas das técnicas utilizadas na verificação podem ser, por exemplo, o uso de modelos determinísticos, para os quais é mais fácil determinar os valores de saída das variáveis, ou a execução de casos simplificados, com, por exemplo, menos clientes no sistema. A sexta fase (de validação) consiste em determinar se o modelo é uma representação razoável do sistema. R. Jain (Jain, 1991) relaciona as seguintes técnicas de validação: intuição de um especialista, medidas realizadas no sistema real e resultados teóricos (obtidos através de algum método analítico). Neste trabalho, são sempre utilizados os resultados teóricos para validação dos modelos. Assim, as duas fases em questão exprimem, respectivamente, se o modelo funciona e se ele realmente representa o sistema real.

As etapas de experimentação e de análise dos resultados são aquelas em que, efetivamente, há a execução do modelo e uma posterior interpretação das saídas. Nessas duas etapas, a simulação se faz mais clara, pois então se obtêm saídas que podem atestar a eficácia de todo o processo.

Para assegurar a validade desses valores (variáveis de saída) deve ser aplicado à simulação um planejamento estatístico, onde alguns dos aspectos a serem considerados são:

- o uso de warm-up significa que os dados colhidos pelo programa durante o chamado período de aquecimento não são significativos à avaliação. Esse período de aquecimento chega ao final quando o sistema chega ao equilíbrio. Um sistema entra em equilíbrio quando atinge o estado estacionário.
- a validade estatística dos valores de saída de uma simulação são dependentes do grau de aleatoriedade dos números aleatórios gerados. Sendo assim, devem-se usar corretamente os números aleatórios, de forma que sejam independentes, isto é, não deve existir correlação na seqüência de números aleatórios. Além disso, devem ser geradas seqüências diferentes de números aleatórios, a partir da utilização de sementes diferentes.

- tendo resolvido a questão do *warm-up* (através do descarte dos dados obtidos durante o período transiente), podem-se então analisar as variáveis de saída. A precisão dessas variáveis é verificada por intermédio do cálculo de intervalos de confiança⁴. Dentre os métodos utilizados para estimar os intervalos de confiança, destacam-se o método de replicação, o método das médias por lotes e o método da regeneração.
- pela estimação duvidosa dos parâmetros de entrada do modelo, medidas de desempenho podem ser produzidas contendo erros. Isso acontece, uma vez que essas entradas (distribuições do tempo de serviço, distribuições do tempo de chegada, probabilidades de rotas, disciplinas de filas) correspondem às estimativas dos parâmetros reais, assim pequenos erros cometidos em sua estimativa podem redundar em grandes erros nas medidas de desempenho.
- o fim de uma simulação pode ser determinado através de vários modos, denominados métodos de parada. Algumas linguagens de simulação permitem a aplicação de vários métodos, podendo, por exemplo, terminar uma simulação considerando o tempo de execução (determinando o tempo de execução) ou métodos de parada automática (construídos em torno de cálculos sobre intervalos de confiança).

A etapa da documentação consiste na elaboração de documentos que retratem de maneira clara o desenvolvimento do processo de simulação, através de todas as suas etapas, especificando as características importantes de cada fase, assim como os resultados obtidos.

Apesar das fases da simulação terem sido apresentadas de uma maneira que subentendesse um aspecto seqüencial, nada impede que haja interações entre as diversas etapas do processo. Na realidade, a maioria das processos de simulação é feita de maneira não-seqüencial.

5.3.2. Orientações para Simulação.

Basicamente, há três entidades que podem proporcionar orientações distintas à simulação: atividades, eventos e processos. O importante é ressaltar que essas entidades não são disjuntas. Na verdade, umas são componentes das outras. O conjunto dessas entidades representa o comportamento e o desempenho do

⁴ Cujo algoritmo para o cálculo é apresentado no Apêndice B.

o sistema. A Figura 5.2, reproduzida de (Soares, 1992), apresenta o inter-relacionamento entre as três entidades.

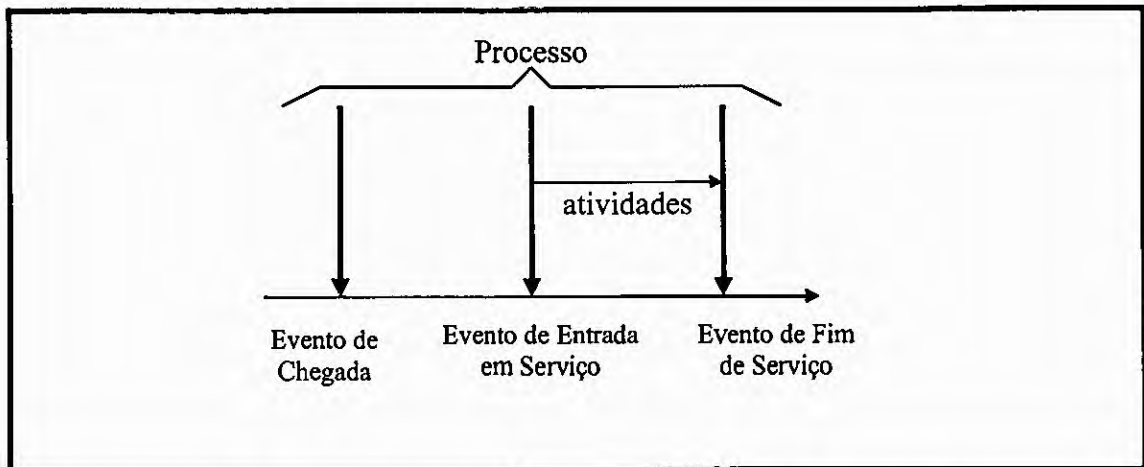


FIGURA 5.2. Relações entre Evento, Processo e Atividade.

Todas as entidades mostradas na Figura 5.2 dependem do nível de detalhamento (visão particular) do sistema considerado. O importante é perceber que o relacionamento entre elas ocorre da seguinte maneira: eventos ocorrem e mudam o estado do sistema. Os eventos são motivados por algum tipo de atividade. E o conjunto de eventos, logicamente relacionados e suas atividades motivadoras, constituem o processo (que é um superconjunto contendo eventos e atividades).

Para exemplificar o relacionamento entre as entidades, supõe-se que se queira representar uma situação já bastante explorada neste trabalho: processos (que aqui serão referenciados por *jobs*, para evitar ambigüidade) concorrendo por um processador. Um *job* chega à fila do processador (evento) e, após algum período de tempo, ele é atendido (evento), o processador o processa (atividade) e, após o processamento, o *job* abandona o processador (evento). Toda essa seqüência de eventos e atividades é chamada de processo. Assim, a simulação discreta pode ser orientada de acordo com a entidade básica (atividades, eventos e processos) que direciona a ótica da simulação.

5.3.3. Software para Simulação.

De uma maneira geral, os *software* para implementar a simulação podem ser agrupados nas seguintes subclasses (Soares, 1992):

- Linguagens de programação convencionais;
- Linguagens de simulação;

- Extensões funcionais;
- Pacotes de uso específico.

Linguagens de Programação Convencionais, apesar de não terem sido desenvolvidas com o objetivo de simular, podem implementar simulações, logicamente, sujeitas a várias restrições inerentes às peculiaridades da própria linguagem. Entretanto, essas linguagens (como C, Pascal, etc.) possuem a vantagem de serem amplamente conhecidas e, conseqüentemente, haver mais pessoas habituadas com o estilo da linguagem.

Linguagens de Simulação já são projetadas com uma finalidade específica: a simulação de sistemas. Essas linguagens obedecem a uma das orientações vistas na seção 5.3.2, conforme as necessidades que o modelo (ou o modelador) impõe. Em outras palavras, as linguagens de simulação podem ser orientadas ao exame da atividade, a evento ou a processo.

Extensões Funcionais são bibliotecas acrescentadas a linguagens de propósito geral (C, Pascal, Fortran, por exemplo), que são denominadas linguagens hospedeiras. A grande vantagem das extensões reside no fato de não se necessitar de aprendizagem de uma nova linguagem para realizar simulação, pois a linguagem utilizada é conhecida. O único problema está no fato de que é necessário um prévio conhecimento da filosofia da simulação adotada. São exemplos de extensões funcionais o *smpl* (extensão da linguagem C orientada a evento) (MacDougall, 1987), e o *HPSIM* (extensão da linguagem Modula 2 orientada a processo) (Soares, 1992). A extensão *smpl* é objeto de estudo da seção 5.3.4, na qual é apresentada uma aplicação para essa extensão funcional.

Entretanto, existem software para simulação que são direcionados à avaliação de sistemas peculiares. Esses software são denominados pacotes de simulação e, para casos bastante específicos, realizam soluções de simulação satisfatórias. Um possível inconveniente dos pacotes de simulação está na inflexibilidade do software, isto é, pequenas mudanças no modelo simulado podem gerar uma extrapolação do escopo da ferramenta, inviabilizando a sua utilização.

5.3.4. A Simulação Baseada em *smpl*.

Esta seção descreve a extensão funcional para a linguagem C para o provimento de simulação orientada a eventos, **SiMulation Programming Language**

(*smpl*), desenvolvida e apresentada por M. H. MacDougall (MacDougall, 1987). Aqui, são apresentadas as entidades elementares da filosofia *smpl*. Um sumário das funções básicas do *smpl*, juntamente com suas respectivas semânticas e parâmetros, é apresentado no Apêndice B.

Segundo a visão *smpl* de um sistema, há três tipos de entidades básicas: os recursos (*facilities*), as fichas (*tokens*) e os eventos (*events*), descritos a seguir:

- *Facilities (Recursos)*. Um sistema é visto como um conjunto de *facilities*, onde cada uma delas representa tipicamente um recurso do sistema que está sendo modelado, tais como CPU e discos rígidos, em se tratando de um sistema computacional. O *smpl* provê funções que definem, reservam, liberam e realizam preempção nas *facilities*.
- *Tokens (Fichas)*. As entidades *tokens* servem para representar o comportamento dinâmico do sistema. Esse comportamento é modelado através do movimento dos *tokens* através dos recursos. Uma finalidade dos *tokens* é a reserva de um determinado recurso, definindo-se a ocupação e a desocupação de uma certa *facility*, através da posse do *token*.
- *Events (Eventos)*. Eventos são as entidades responsáveis pelas mudanças de estado dos recursos do sistema modelado. Assim, quando um *token* aloca um determinado recurso (que estaria anteriormente livre, excetuando-se a situação onde há preempção), este passa de um estado "livre" para um estado "ocupado".

Pelo fato do *smpl* ter sido proposto com uma orientação a eventos, há uma correlação direta com os eventos Statecharts dos *templates* que descrevem o funcionamento básico de um sistema de filas. Essa correlação é apresentada na seção 5.4.2, onde os eventos-padrão dos *templates* são associados às funções *smpl*.

5.4. Soluções Aplicadas às Especificações Statecharts

Esta seção visa a ilustrar a aplicabilidade das especificações propostas nesta tese, utilizando-se tanto solução analítica quanto simulação. Primeiramente, será aplicada a solução de Jackson à especificação Queuing Statecharts. Em seguida, uma solução baseada em *smpl* é utilizada para a especificação dos Statecharts

Estocásticos. A idéia é enfatizar o caráter genérico das especificações, as quais foram elaboradas para propiciar uma certa independência da solução adotada.

5.4.1. Solução de Jackson Aplicada aos Queuing Statecharts

Recorrer-se-á novamente ao modelo do servidor de arquivos apresentado no Capítulo 4. É viável aplicar-se a solução de Jackson a essa situação pelo fato do modelo ser aberto (com uma realimentação – *feedback* – do centro 2 para o centro 1), conforme ilustra a Figura 5.3, além de obedecer às restrições apresentadas no teorema de Jackson (seção 5.2.1). A escolha pela notação Queuing Statecharts se deve ao fato das redes de Jackson serem baseadas na figura do centro de serviço (transições entre eles), sendo que esses centros estão definidos mais explicitamente em Queuing Statecharts. Contudo, nada impede a utilização da representação de Statecharts Estocásticos, unindo-se os estados *fila* e *servidor* em um superestado *centro de serviço*.

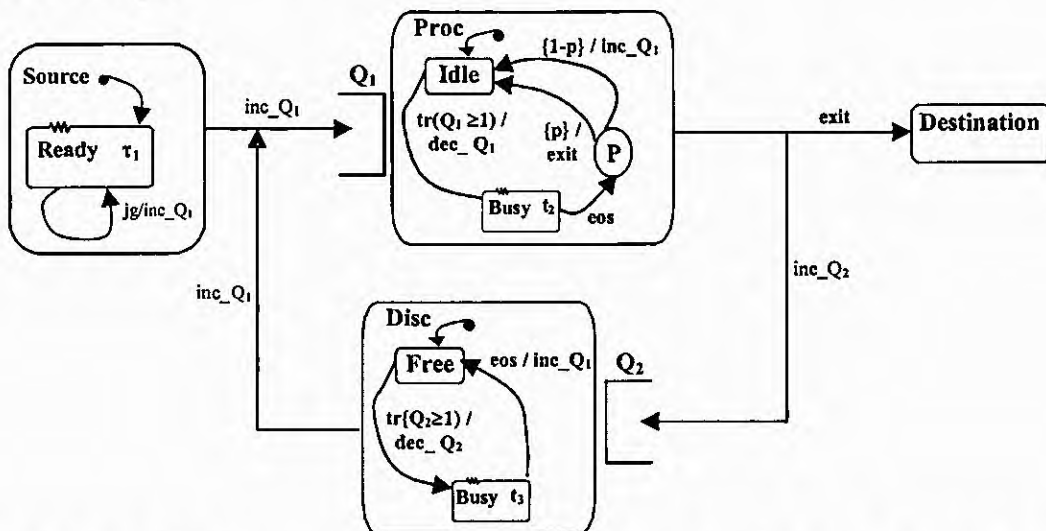


FIGURA 5.3. Servidor de Arquivos especificado em Queuing Statecharts.

Antes de apresentar-se a solução da especificação Queuing Statecharts, são necessárias algumas considerações sobre como se associam os eventos da especificação com as variáveis das equações utilizadas na solução de Jackson. Assim, as seguintes premissas são adotadas:

- O termo "nó", utilizado nas definições do teorema de Jackson, aqui é tomado como sinônimo de centro de serviço (conjunto de fila e servidor), representado por C_i , com $i = 1, 2, \dots, M$.

- Há também uma diferenciação entre taxa de chegada (λ - proveniente de uma geração externa de clientes) e taxa de entrada (Λ - proveniente de outro centro, mesmo que acrescida à taxa de chegada).
- p_{ji} é a probabilidade de estar-se em um estado j e ir ao estado i , onde $i, j = 1, 2, \dots, M$. Se $i = j$, então tem-se a probabilidade de permanecer no estado atual.
- Na especificação de Queuing Statecharts, p_{ji} está sempre associada à ocorrência do evento inc_Q_k , com $k = 1, 2, \dots, M$, sendo o evento inc_Q_k produzido ou por uma fonte externa ou por outro centro de serviço. K especifica também o próximo centro a ser visitado.
- Não há possibilidade de um cliente, após atendido, permanecer no centro de serviço, ou seja, $p_{ji} = 0$ (se $i = j$).
- Por definição, a taxa média de chegada é o inverso do tempo médio entre chegadas ($\tau = 1/\lambda$) e a taxa média de serviço é o inverso do tempo médio de serviço ($\tau = 1/\mu$).
- As taxas de entrada (Λ) são sempre conseqüências de eventos de incrementos das variáveis Q_k ($k=1, 2, \dots, M$).

A seguir são considerados parâmetros para o exemplo, extraídos de (Francês et al., 20001), assim como a semântica que está associada na especificação de cada um desses parâmetros:

- Tempo Médio entre chegadas: $\tau_1 = 0,083$ ($\lambda = 12$), dado pela ocorrência do evento cg e ação inc_Q_1 no estado *Source*;
- Tempo Médio de Serviço em Proc: $\tau_2 = 0,04$ ($\mu_1 = 25$), dado pela permanência no estado *Busy.Proc*;
- Tempo Médio de Serviço em Disc: $\tau_3 = 0,05$ ($\mu_2 = 20$), dado pela permanência no estado *Busy.Disc*;
- Probabilidade de, após o atendimento em Proc, o cliente deixar o sistema: $p = 0,6$, dada pela ocorrência do evento probabilístico $\{p\}$ e ação *exit*;
- Probabilidade de, após o atendimento em Proc, o cliente ir ao disco realizar uma operação de entrada/saída: $1-p = 0,4$, dada pela ocorrência do evento probabilístico $\{1-p\}$ e ação *exit*;

Considerando-se chegada e atendimentos exponencialmente distribuídos e aplicando-se o método de Jackson ao exemplo:

$$\rho_i = \frac{\Lambda_i}{\mu_i} \quad \text{e} \quad \Lambda_i = \lambda_i + \sum_{j=1}^M p_{ji} \cdot \Lambda_j$$

Dessa forma, tem-se um sistema linear:

Para o centro C_1 (Processador e sua fila):

$$\Lambda_1 = \underbrace{\lambda_1}_{\lambda} + \underbrace{p_{11}}_0 \Lambda_1 + \underbrace{p_{21}}_1 \Lambda_2 \rightarrow \Lambda_1 = \lambda + \Lambda_2$$

Taxa de entrada no centro 1 Taxa de chegada no centro 1 Probabilidade de permanecer no centro 1 Probabilidade de ir do centro 2 ao centro 1 Taxa de entrada no centro 2

Para o centro C_2 (Disco e sua fila):

$$\Lambda_2 = \underbrace{\lambda_2}_0 + \underbrace{p_{12}}_q \Lambda_1 + \underbrace{p_{22}}_0 \Lambda_2 \rightarrow \Lambda_2 = q\Lambda_1$$

Assim, podem-se calcular as taxas de entrada para cada centro de serviço:

$$\Lambda_1 = \lambda + q\Lambda_1 = \frac{\lambda}{1-q} = \frac{12}{0,6} = 20$$

$$\Lambda_2 = 0,4 \times 20 = 8$$

Tendo-se os valores das taxas de entrada Λ_1 e Λ_2 , calcula-se a utilização de cada centro:

$$\rho_i = \frac{\Lambda_i}{\mu_i} \begin{cases} \rho_1 = \frac{20}{25} = 0,80 \\ \rho_2 = \frac{8}{20} = 0,40 \end{cases}$$

Além disso, como em uma rede de Jackson se considera cada centro como um modelo M/M/1, podem-se calcular outras medidas de desempenho para cada centro por intermédio das equações a seguir, considerando-se as características M/M/1:

▪ Número médio de clientes

- Para o centro C_1 : $n_1 = \frac{\rho_1}{(1-\rho_1)} = 4,0$

- Para o centro C_2 : $n_2 = \frac{\rho_2}{(1-\rho_2)} = 0,6667$

- Número médio de clientes na fila
 - Para o centro C_1 : $nq_1 = \frac{\rho_1^2}{(1-\rho_1)} = 3,2$
 - Para o centro C_2 : $nq_2 = \frac{\rho_2^2}{(1-\rho_2)} = 0,2667$
- Tempo médio de resposta
 - Para o centro C_1 : $r_1 = \frac{1/\mu_1}{(1-\rho_1)} = 0,20$
 - Para o centro C_2 : $r_2 = \frac{1/\mu_2}{(1-\rho_2)} = 0,0833$
- Tempo médio de fila
 - Para o centro C_1 : $w_1 = \rho_1 \frac{1/\mu_1}{(1-\rho_1)} = 0,16$
 - Para o centro C_2 : $w_2 = \rho_2 \frac{1/\mu_2}{(1-\rho_2)} = 0,0333$

A adoção da solução de Jackson se justifica pelo fato do modelo ser aberto, contendo uma realimentação (*feedback*) em um dos servidores, além de que esse modelo obedece às restrições impostas pelo teorema de Jackson. Entretanto, qualquer outro método que contemple aquelas restrições poderia ter sido empregado para o caso em questão. Um exemplo é o estudo apresentado em (Francês et al., 20001), no qual é realizada a avaliação de desempenho utilizando-se cadeias de Markov de parâmetro contínuo, conforme sugerido em (Vijaykumar, 1999).

A solução de Jackson é bastante aplicável a situações reais, pois sistemas reais, via de regra, possuem a característica de geração externa de clientes, conforme a abordagem dada no exemplo anterior. Todavia, há algumas situações em que se deseja estudar sistemas, cujos clientes são internos e que não vão a um servidor externo. Para esses casos, foi proposta uma extensão ao método de Jackson, denominada BCMP, que é apresentada sucintamente no Apêndice C.

5.4.2. Solução por Simulação Aplicada aos Statecharts Estocásticos

Uma solução alternativa à analítica é a *simulação*, que é apresentada, para o mesmo exemplo, na seção seguinte.

Pelo fato do smpl ter sido proposto com uma orientação a eventos, há uma correlação direta com os eventos Statecharts dos *templates* que descrevem o funcionamento básico de um sistema de filas. Essa correlação é sintetizada na Tabela 5.1, onde os eventos padrão dos *templates* são associados às funções smpl.

A correlação eventos Statecharts e funções smpl é mostrada na Figura 5.4, na qual cada evento dos *templates* (e conseqüentemente de um sistema de filas) tem uma correspondência unívoca com as funções smpl.

TABELA 5.1. Correlação entre Eventos Statecharts e Funções smpl.

Eventos Padrão da Especificação Statecharts	Funções smpl
<i>jg</i>	<code>schedule(ev, expntl(Ta1), Customer)</code>
<i>inc_s</i>	<code>request(Servern, Customer, pri)</code>
<i>tr[not in(Free.Server1_Q)]</i>	<code>schedule(ev, expntl(Ts1), Customer)</code>
<i>eos</i>	<code>release (Servern, Customer)</code>
{ <i>p</i> } com um valor 60% e {1- <i>p</i> } para um valor 40%.	<code>k = random(i, j); if ((6001 <= k) && (k <= 10000)) schedule(ev, te, Customer);</code>

A semântica da seqüência do Statechart da Figura 5.4 tem início com a geração de clientes (*jg*) a uma distribuição exponencial com tempo médio entre chegadas t_1 (`expntl(Ta1)`). Cada geração de um novo cliente (*inc_S*) acarreta em mais uma requisição de serviço (`request (Server_n, Customer, pri)`), que terá que ser acondicionada na fila do servidor solicitado. Se o servidor estiver livre, ele é tomado pelo cliente (*tr[not in(Idle.Proc_Q)]*), o qual será atendido obedecendo a uma distribuição exponencial com tempos médios de serviço t_2 ou t_3 (para *Proc* e *Disc*, respectivamente). Ao término do serviço, através do evento *eos* (função `release (Servern, Customer)`), o cliente toma um caminho probabilístico (admitindo-se também o caminho com probabilidade 1).

É importante ressaltar que há funções que, mesmo não tendo relação direta com os eventos Statecharts descritos nos *templates*, devem acontecer por força da própria estrutura de implementação smpl. Tal fato ocorre, por exemplo, com as funções que têm o parâmetro *ev*, isto é, aquelas funções que estão relacionadas à ocorrência de um evento (tal como `schedule()`). Para que essas funções sejam executadas é necessário que anteriormente tenha havido uma chamada à função

cause(ev, Customer), a qual é responsável pela efetivação de um evento, que deve ser retirado da lista de eventos⁵.

Analogamente, há eventos que são representados nos templates que possuem o objetivo específico de tornar mais clara a especificação, mas que não têm uma correlação direta com uma função smpl. Os eventos que decrementam as filas no momento em que um cliente adentra um servidor possuem essa característica, pois esse mecanismo está implícito na própria simulação, não tendo uma função particular para realizar a tarefa.

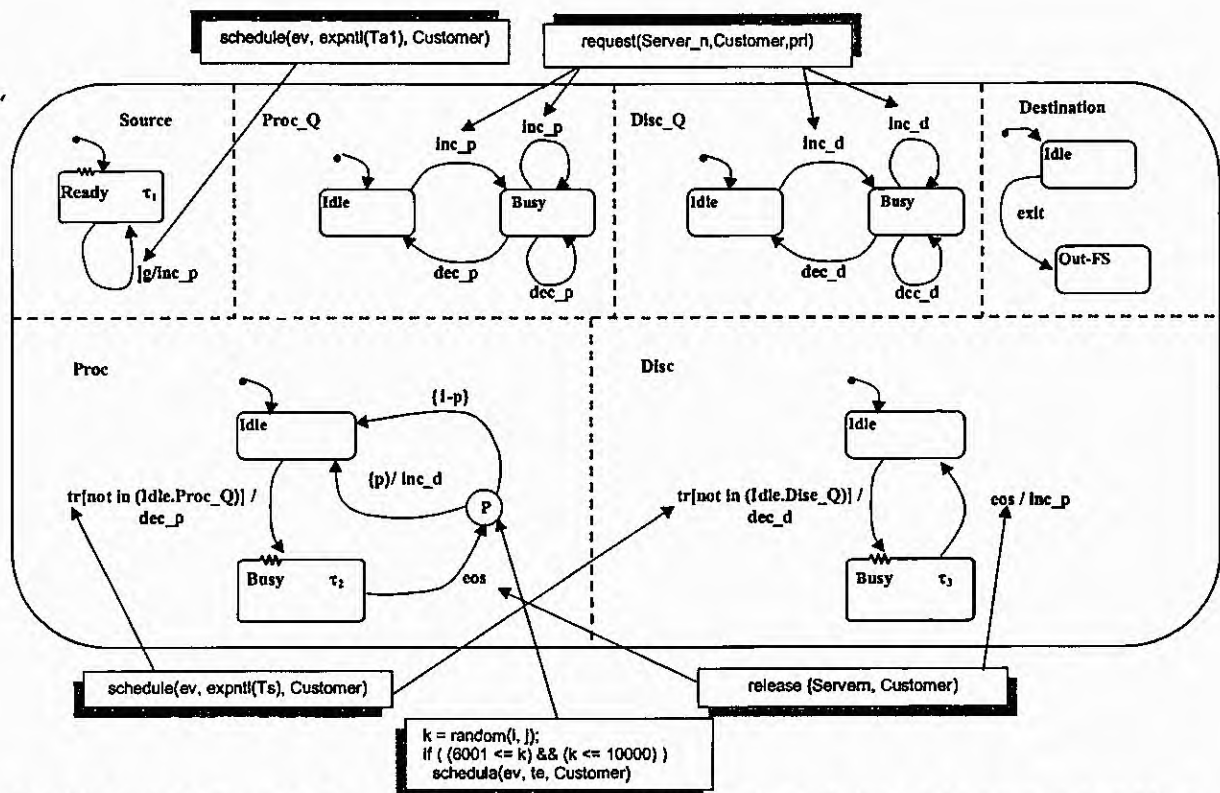


FIGURA 5.4. Correlação entre Funções *smpl* e eventos Statecharts para o servidor de arquivos.

A partir da associação descrita na Tabela 5.1 e na Figura 5.4, e admitindo-se os mesmos parâmetros de entrada da solução analítica, os seguintes valores foram obtidos por simulação, considerando-se oito execuções, um intervalo de confiança⁶ de 95% e distribuições exponenciais para tempo entre chegadas e de serviço.

Para $\tau = 0,0833$ (tempo entre chegadas de clientes);

$S_1 = 0,04$ (tempo de serviço do Processador);

⁵ Estrutura que é responsável por guardar, de maneira seqüencial, todos os eventos que devem ser realizados durante a simulação

⁶ O Apêndice B apresenta o algoritmo para o cálculo do intervalo de confiança, conforme sugerido em (MacDougall, 1987).

$S_2 = 0,05$ (tempo de serviço do Disco);

$t_e=10000000$ (tempo de execução da simulação).

Utilização do Processador (U1)	0.800
Utilização do Disco (U2)	0.400
Throughput do Sistema (X)	12.00
Throughput do Sistema (X1)	20.00
Throughput do Sistema (X2)	8.003
Nº médio de clientes no centro C_1 (N1)	4.006
Nº médio de clientes no centro C_2 (N2)	0.667
Nº médio de clientes no sistema (N)	4.673
Tempo médio de fila no Processador (W1)	0.160
Tempo médio de fila no Disco (W2)	0.033
Tempo médio de resposta do centro C_1 (R1)	0.200
Tempo médio de resposta do centro C_2 (R2)	0.083

5.4.3. Limitações das Soluções Abordadas.

Esta seção ilustra as possíveis deficiências das soluções apresentadas anteriormente, sem a pretensão de apontar a melhor solução.

Atendo-se à *solução analítica*, R. Jain - em (Jain, 1991) - enumera um conjunto de treze limitações relacionadas a essa solução. Algumas delas são listadas a seguir.

- Tempos de serviços não-exponenciais. A maioria dos modelos requer que se assuma o tempo de serviço como uma variável aleatória exponencial. Via de regra, os resultados obtidos através dessa assunção não geram uma diferença significativa. Entretanto, apesar da distribuição exponencial ter-se mostrado aplicável para representar tempos entre chegadas, para tempos de atendimento (conforme sugerido em (Prado, 1999)), são boas candidatas as distribuições hiperexponencial e Erlang.
- Análise do Estado Transiente. A maioria dos métodos analíticos se baseia nas características do estado de equilíbrio (*steady state*). Alguns sistemas são tão dinâmicos, que o equilíbrio quase nunca é alcançado. Nesses casos, a análise transiente pode ser mais útil que, eventualmente, a do *steady state*.

- Exclusão Mútua. Clientes diferentes podem tentar usar um mesmo recurso ao mesmo tempo (por exemplo, um cliente deseja ler um arquivo e outro deseja escrever no mesmo arquivo). Para essa situação, deve haver algum mecanismo de exclusão mútua, geralmente implementado em situações reais. Entretanto, tratar casos de exclusão mútua através de métodos analíticos não é uma tarefa trivial.
- Posse Simultânea de Recursos. Um cliente pode requerer para si o uso de mais de um recurso simultaneamente (por exemplo, um processo *multi-thread* solicitando operações de E/S e processamento ao mesmo tempo). Esse tipo de fenômeno também impõe uma dificuldade adicional ao tratamento analítico.
- Recursos Passivos. Apesar de serem recursos do sistema, alguns componentes, como memória principal, não realizam o controle sobre si mesmos, sendo dependentes da ação de outros servidores. O tratamento de recursos com essa característica não pode ser considerado trivial.

A respeito da solução por simulação, algumas restrições extraídas de (Silva & Muntz, 1992) e (Soares, 1992) são apresentadas a seguir.

- Aproximação dos Resultados. Se o problema que está sendo modelado admite as restrições impostas pela solução analítica, ela sempre proverá o resultado mais exato. Em contrapartida, a simulação é uma aproximação, mesmo que, eventualmente, ela venha a ser uma ótima aproximação, em certos casos.
- Dependência de Abordagem Estatística. A qualidade dos resultados de uma simulação é expressa em função de *intervalos de confiança*, o que implica que seja necessária a execução de algumas execuções do experimento, para que haja subsídios estatísticos para a avaliação (admitindo-se que ocorra uma convergência para o estado estacionário). Para essas situações, dependendo da dimensão do problema, a solução analítica pode resumir-se à resolução de um sistema linear mais elementar (como no exemplo apresentado neste capítulo, para a solução de Jackson).
- A Complexidade Introduzida por Algumas Linguagens e Ferramentas. Algumas linguagens e ferramentas introduzem uma complexidade a mais na solução do problema, que está relacionada ao aprendizado e

manipulação delas próprias. Apesar de nem sempre ser trivial a utilização desses mecanismos de simulação, há uma certa evolução (principalmente em relação às ferramentas) em direção a uma interação mais amigável com o modelador. Esse caráter amistoso também pode levar o usuário à negligência no conhecimento da complexidade que está por trás da ferramenta (o conhecimento do processo da simulação), o que em muitos casos é desejável.

A despeito da controvérsia gerada pelo assunto, uma boa opção pode ser o uso combinado das duas soluções para minimizar os esforços. Soluções analíticas podem apontar para valores (mesmo que admitindo-se as restrições já comentadas) que devem ser obtidos pela simulação. Dessa forma, a primeira solução pode ser usada para validar a segunda (mesmo que seja uma constatação parcial para um conjunto restrito de parâmetros).

A escolha da solução também está relacionada ao tamanho e à complexidade do modelo estudado. Sob esse ponto de vista, a simulação parece oferecer uma flexibilidade maior. Entretanto, métodos bastante aprimorados têm oferecido características adicionais à solução analítica, tal como o BCMP, apresentadas no Apêndice C, que minimizam algumas das restrições inerentes aos métodos tradicionais.

5.5. Considerações Finais

Este capítulo apresentou uma avaliação particular sobre a questão da escolha da solução para um modelo que se pretende estudar. Com o intuito de estabelecerem-se subsídios mínimos para essa escolha, foram explanadas as duas abordagens que podem prover medidas de desempenho: a analítica e a simulação.

Com relação à solução analítica, as redes de Jackson, para modelos abertos, foram ilustradas de maneira concisa (para o exemplo do servidor de arquivos). Estratégia idêntica foi usada para a simulação, isto é, ao mesmo modelo foi aplicada a solução por simulação, baseada em *smpl*. Além do interesse em validar as soluções, havia um interesse maior em apresentar o caráter generalista das especificações propostas, que se configura no principal objetivo do capítulo.

Os *templates* (conjunto de estados e eventos padrão) definidos no Capítulo 4 formam um alicerce mínimo para a especificação de sistemas de filas, que possuem um funcionamento relativamente padrão, via de regra, com mudanças nos valores

dos parâmetros de entrada. O cerne da idéia é que, para o mesmo problema, as especificações em Statecharts Estocásticos e Queuing Statecharts se equivalham, apesar das notações gráficas apresentarem filosofias ligeiramente diferentes no trato do caminho percorrido pelos clientes.

A exposição de possíveis vantagens e desvantagens de cada abordagem apresentada no capítulo pode servir como uma referência inicial sobre o assunto. Contudo, não estão contemplados outros fatores determinantes, que possuem caráter mais subjetivo, tais como conhecimento *a priori* ou intuição do modelador, tempo dispensado para realizar a avaliação e recursos materiais disponíveis.

A despeito da subjetividade intrínseca à escolha da solução, alguns fatores abordados devem ser ratificados aqui. Um ponto fundamental é o estado que deve ser tomado na avaliação: o equilíbrio ou o transiente. Nesse ponto é que se edifica a formalização de passos e configurações sugerida no Capítulo 4, pois a especificação proposta pode tanto representar os estados transientes quanto o estado de equilíbrio. Se negligenciada a definição dos passos sucessivos, no decorrer do tempo, a análise deveria ater-se ao estado estacionário. Entretanto, nas formalizações sugeridas no Capítulo 4, contempla-se a flexibilidade da escolha entre as abordagens. Esse caráter abrangente aumenta o leque de aplicações das especificações propostas.

Capítulo 6

Estudo de Caso: Programação Distribuída Baseada em PVM

6.1. Considerações Iniciais

Este capítulo apresenta um estudo de caso envolvendo a programação distribuída e seus mecanismos utilizados para escalonamento de processos, que constitui uma área onde a modelagem pode prover uma ajuda interessante.

Mais precisamente, o experimento realizado envolve a alocação de processos de uma determinada aplicação em elementos de processamento distribuídos, utilizando-se ambientes de passagem de mensagens e seus respectivos métodos de escalonamento.

Um experimento realizado empiricamente vai servir como base para a parametrização inicial da modelagem e, a partir dessa parametrização, extrapola-se a situação para comportamentos futuros, levando-se em consideração mecanismos mais aprimorados para realizar a distribuição dos processos.

Situações típicas de programação distribuída e escalonamento de processos constituem casos de estudo interessantes, dada a complexidade inerente ao problema. Nessas situações, pode ser conveniente não apenas a análise do estado estacionário do sistema, mas também observações do estado transiente, pois pode acontecer que certas decisões tenham de ser tomadas a cada iteração (passo) da realização do experimento. A formalização proposta no Capítulo 4 abrange a possibilidade de ser necessária uma observação passo a passo do ambiente modelado.

Via de regra, as soluções observam o sistema em seu estado estacionário, através do qual podem ser obtidas medidas de desempenho. Entretanto, ambas as soluções prevêm a possibilidade de observações intermediárias, isto é, em passos determinados de um experimento. As soluções analíticas podem observar as probabilidades de transição entre estados em n passos (no caso discreto) ou probabilidade de transição após decorrido um determinado intervalo de tempo Δt (para o caso contínuo). As simulações, por se basearem em estatística, naturalmente trabalham com os n passos até o estado estacionário. É importante observar que para a finalidade de avaliação de desempenho, geralmente, são

tomadas as médias, admitindo-se o equilíbrio. Entretanto, em certas ocasiões, o sistema pode não convergir para o estado estacionário, o que não impede que seja feita a avaliação.

Nas seções seguintes, são apresentadas as características de um ambiente real, no qual foram feitas experimentações e obtidos valores de entrada para a modelagem. Esse ambiente utiliza um software para distribuir o processamento da aplicação, sobre uma rede de computadores. A partir dos dados reais, são feitas extrapolações e observadas algumas características interessantes do problema modelado.

É importante observar que o estudo de caso realizado neste capítulo não é o objeto principal desta tese e, por isso, não se pretende apresentar uma série de observações conclusivas à área de escalonamento de processos. O estudo de caso descrito neste trabalho visa a ilustrar a aplicabilidade das especificações formalizadas no Capítulo 4. As especificações se constituem na contribuição precípua desta tese e, nesse contexto, um exemplo real, que ilustre todo o processo de modelagem, é uma boa maneira de demonstrar a eficácia das especificações propostas.

6.2. Descrição do Experimento

O experimento realizado consiste na distribuição dos processos de uma aplicação, em vários elementos de processamento que se encontram espalhados pela rede local de computadores, situada no Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC), do Instituto de Ciências Matemáticas e de Computação (ICMC), da Universidade de São Paulo (USP).

6.2.1. Objetivo do Estudo

Um estudo de distribuição de processos, como o realizado aqui, tem a sua eficiência condicionada ao grau de "inteligência" que o algoritmo de escalonamento utilizado possui. Se o escalonador¹ possui algum critério de ponderação ao realizar a distribuição, é razoável esperar-se um melhor aproveitamento dos recursos do sistema.

¹ Escalonador, neste trabalho, é utilizado para designar o software responsável por decidir para quais máquinas os processos devem ser encaminhados, a fim de serem atendidos.

Partindo-se da premissa anterior, pretende-se investigar a relação entre o comportamento do sistema utilizando um mecanismo que considere todos os elementos de processamento como equiprováveis (tal como é feito no PVM), e o comportamento desse mesmo sistema, adotando-se um algoritmo de distribuição que realize algum tipo de ponderação (aqui, a taxa de processamento da máquina é adotada como critério).

A obtenção dos tempos médios, tomados de maneira empírica serve como parametrização para a modelagem usada para fazer algumas inferências a respeito do sistema. A utilização da modelagem, via de regra, se apresenta como uma alternativa interessante para estudar o desempenho de sistemas sensíveis à aferição realizada *in loco*, a qual pode interferir no resultado final da análise.

6.2.2. Características de Hardware e de Software do Ambiente Utilizado

O ambiente utilizado é composto por uma rede local de quatro computadores pessoais (arquitetura Intel), executando o sistema operacional Linux. As características de cada máquina são sintetizadas na Tabela 6.1.

TABELA 6.1. Descrição do Ambiente Instrumentado.

Máquina	Processador	Memória	Disco	NIC ²	SO
lasdix	Pentium MMX 200MHz	32MB	4GB	TrendNET 10/100	Linux 2.2.16-13cl
lasdpc08	Pentium II 400 MHz	64MB	13GB	TrendNET 10/100	Linux 2.2.16-13cl
lasdpc10	Pentium II MMX 233 MHz	64MB	12GB	TrendNET 10/100	Linux 2.2.16-13cl
lasdpc11	Pentium MMX 200MHz	32MB	3GB	TrendNET 10/100	Linux 2.2.16-13cl

Os software utilizados no experimento são listados a seguir:

- O PVM - *Parallel Virtual Machine* - (Beguelin, 1994) é um pacote de software que permite a criação de uma máquina paralela virtual sobre uma rede de computadores heterogêneos, de maneira que essa rede de computadores (possivelmente de propósito geral) possa emular um computador paralelo. Ambientes como o PVM e o MPI - *Message Passing Interface* - (McBryan, 1994) são denominados ambientes de passagem de mensagem. A versão do PVM usada no experimento foi o PVM 3.4.2.
- Sistema Operacional Linux versão 2.2.16-13cl.

² NIC - Network Interface Card - designa o modelo da placa de rede utilizada .

- A aplicação se constitui em um programa de ordenação de um vetor de 300 (trezentas) mil posições, utilizando o algoritmo *quick-sort* para realizar essa ordenação. A aplicação utiliza o paradigma SPMD (*Single Program Multiple Data*), no qual um processo mestre envia dados a processos escravos, que aplicam um mesmo programa aos conjuntos de dados.

6.2.3. O Processo de Instrumentação do Sistema

Para obterem-se os tempos médios de serviço, foi usada a função *gettimeofday*³ da linguagem C no início e no final de cada atendimento de clientes, tanto no PVM quanto em cada processador.

Foram gerados cem processos para cada máquina tratar. A distribuição desses processos utilizou o esquema *round robin* do PVM, o qual supõe que todas as máquinas são equiprováveis entre si. Esses tempos foram agrupados em classes, a partir das quais foram montados os histogramas correspondentes. Cada amostra de cem tempos descreve uma distribuição exponencial deslocada da origem. Os gráficos das funções de densidade de probabilidade e o algoritmo do teste de aderência de Kolmogorov-Smirnov (Jain, 1991), realizado para verificação da presença da distribuição exponencial, estão constantes no Apêndice A. Os tempos médios (em milissegundos) para cada serviço são listados na Tabela 6.2.

TABELA 6.2. Tempos Médios de Serviço.

Servidor	Tempo Médio (ms)
PVM	1,53686
Lasdix	135,7939
lasdpc08	31,32981
lasdpc10	82,705
lasdpc11	136,6733

Pelos valores apresentados na Tabela 6.2, o menor tempo foi atribuído ao PVM, em virtude desse software realizar basicamente o serviço de roteamento das tarefas para os processadores. Quanto aos tempos de processamento, houve relação direta entre os menores tempos obtidos e a potência computacional do processador utilizado, o que seria razoável esperar-se.

³ A função *gettimeofday* retorna uma estrutura contendo dois campos: um campo com o tempo em segundos e outro com o tempo em micro-segundos.

6.3. O Processo de Modelagem do Experimento

Ratifica-se, neste ponto, o objetivo de investigarem-se técnicas de escalonamento de processos diferentes da usada pelo PVM, sem, entretanto, haver a necessidade de alteração no código-fonte do ambiente de passagem de mensagem ou a realização de qualquer interferência no funcionamento do sistema real.

O processo de modelagem segue as fases apresentadas no Capítulo 2, sendo que cada etapa desse processo é descrita mais detalhadamente nas seções seguintes.

6.3.1. Especificação do Modelo.

A seguir são apresentadas três possibilidades de especificação para o sistema de programação distribuída utilizando PVM: Statecharts Estocásticos, Queuing Statecharts e redes de Petri. As Figuras 6.1, 6.2 e 6.3 apresentam essas representações, seguidas de uma breve descrição de suas semânticas.

A Figura 6.1 é baseada nos *templates* definidos no Capítulo 4. Por se tratar de um modelo fechado, não há uma fonte externa de geração de clientes. Os clientes são gerados por um servidor dentro do próprio sistema (no caso, o PVM, ao gerar os processos paralelos⁴). Assim, o PVM gera os processos ao atender uma requisição da aplicação, na qual o PVM está inserido. São gerados 400 (quatrocentos) processos, que também é outra característica de um modelo fechado (número fixo de clientes circulando no sistema). A geração se dá quando o PVM é chamado, considerando-se que não há outros processos em sua fila (o evento `tr [in (Idle.PVM_Q)]`). Quando ocorre a realimentação do PVM, efetivada pelos processadores, então a ação do PVM é retornar ao curso normal do código da aplicação que requisitou a distribuição (evento `cg`, no estado `Code`). A especificação tomada aqui representa uma aplicação real, mas nada impediria que houvesse várias requisições da aplicação solicitando a criação de processos paralelos, o que geraria uma fila no PVM também para a distribuição, e não somente quando da realimentação feita pelos processadores.

⁴ A função `pvm_spawn` é responsável pela criação de processos paralelos (Beguelin, 1994).

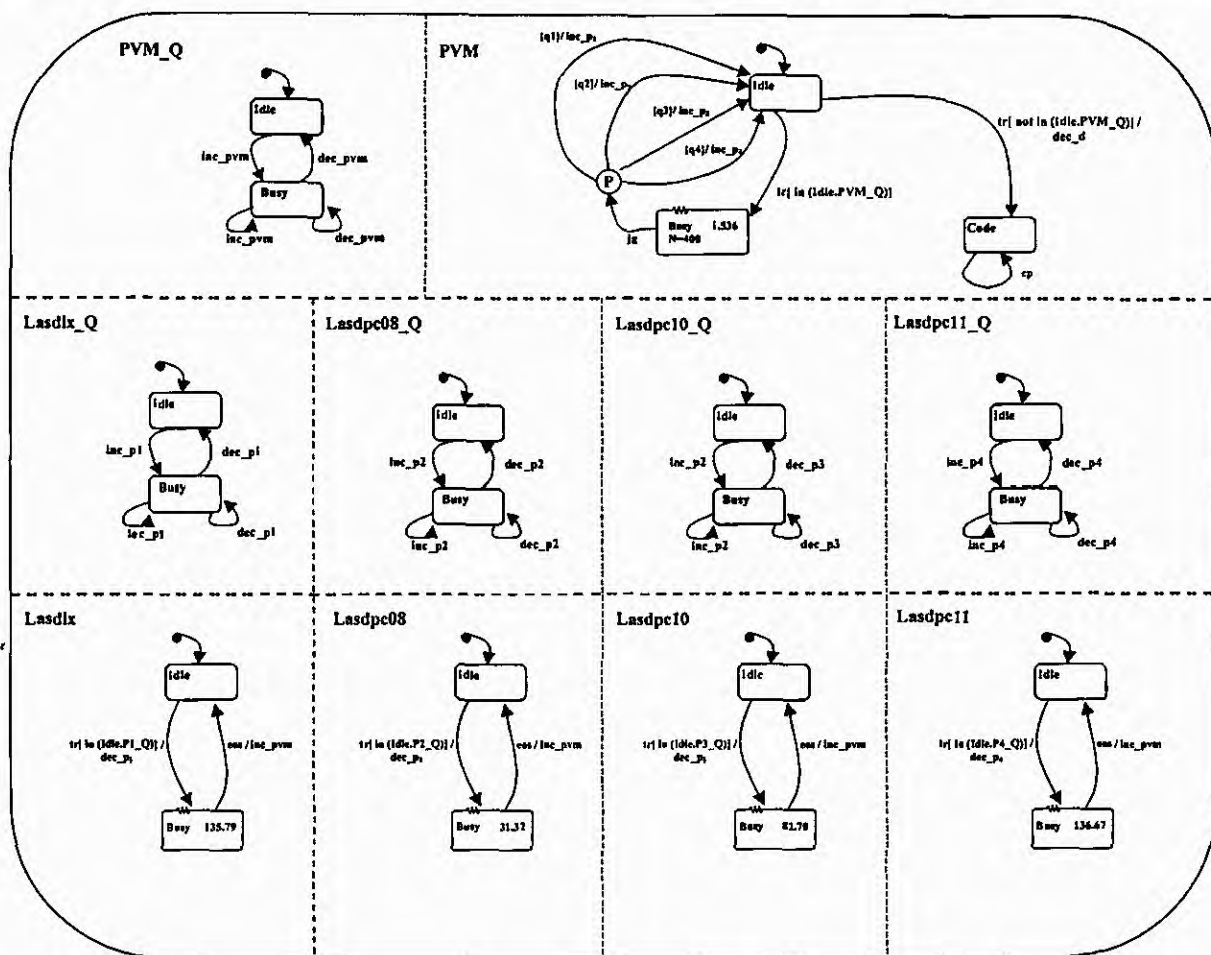


FIGURA 6.1. Especificação do Modelo PVM em Statecharts Estocásticos.

A Figura 6.2 apresenta a mesma situação, entretanto utilizando a notação proposta para os Queuing Statecharts.

A representação da Figura 6.2 remete à idéia da especificação de redes de filas, contudo é possível perceber com maior clareza o inter-relacionamento entre os componentes do sistema, além de seus possíveis estados. Semanticamente semelhantes, as Figuras 6.1 e 6.2 diferem basicamente pela simplificação do estado fila (proposta nos Queuing Statecharts), que herda a notação original das redes de filas, acrescentado-se uma variável Q_i , que é responsável por indicar o número de clientes na fila de um determinado servidor. Além disso, uma outra característica interessante é propiciada pela notação: a trajetória linear que é descrita por um determinado cliente, em sua passagem pelo sistema.

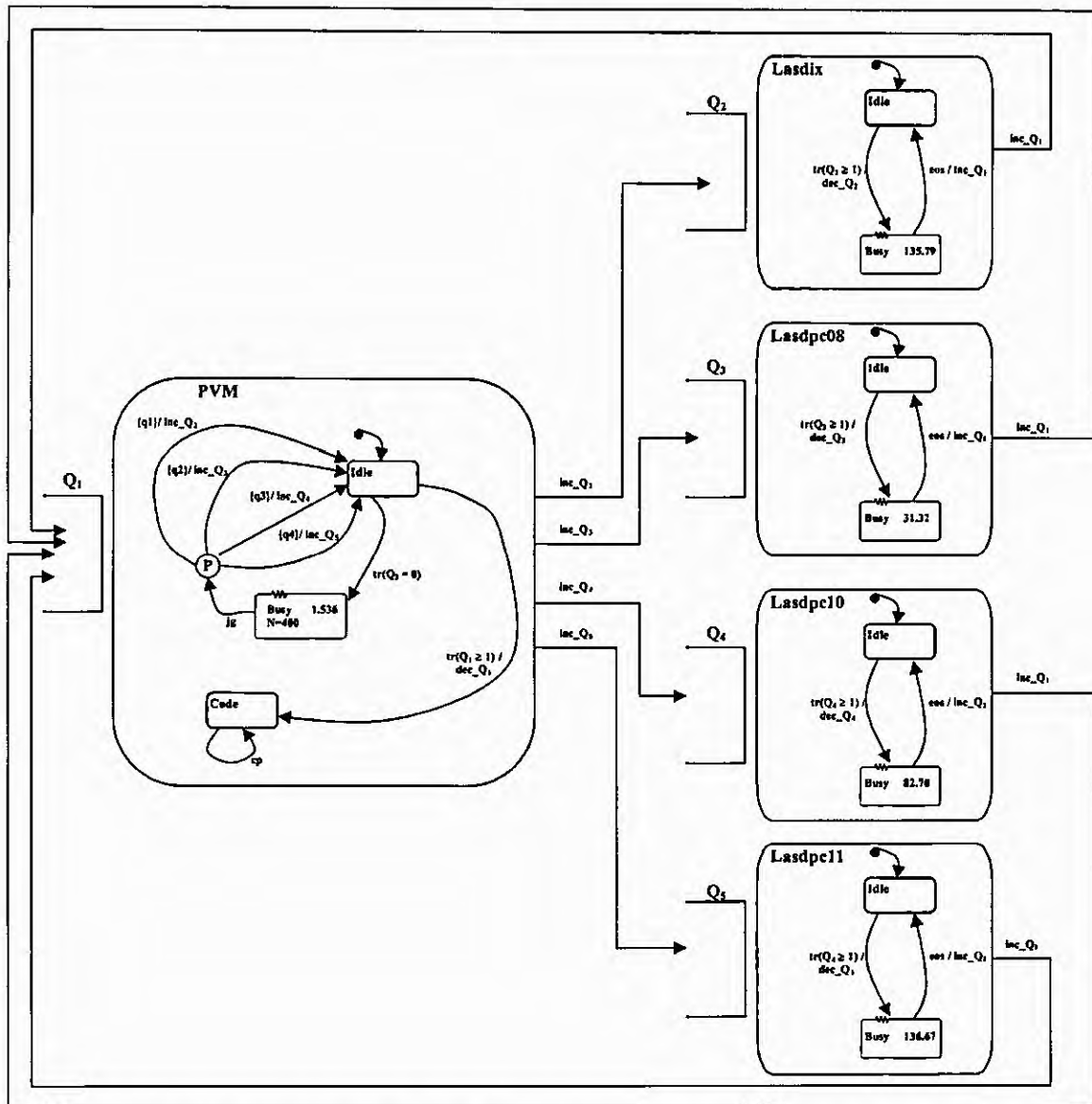


FIGURA 6.2. Especificação do Modelo PVM em Queuing Statecharts.

Uma outra notação importante é a de redes de Petri, elaborada para a mesma situação. A Figura 6.3 apresenta a especificação em redes de Petri, mais precisamente em GSPN.

A especificação em redes de Petri apresenta muitas características interessantes, tal como individualização de clientes. Entretanto, a idéia de orientação a componentes (um conjunto de estados que retratam o comportamento de um determinado elemento do sistema), como a existente nas técnicas baseadas em Statecharts, é uma característica que pode ampliar a visão de como esses componentes se relacionam, formando o conjunto que está sendo o objeto da análise. Essa orientação a componentes não está definida de forma explícita em redes de Petri.

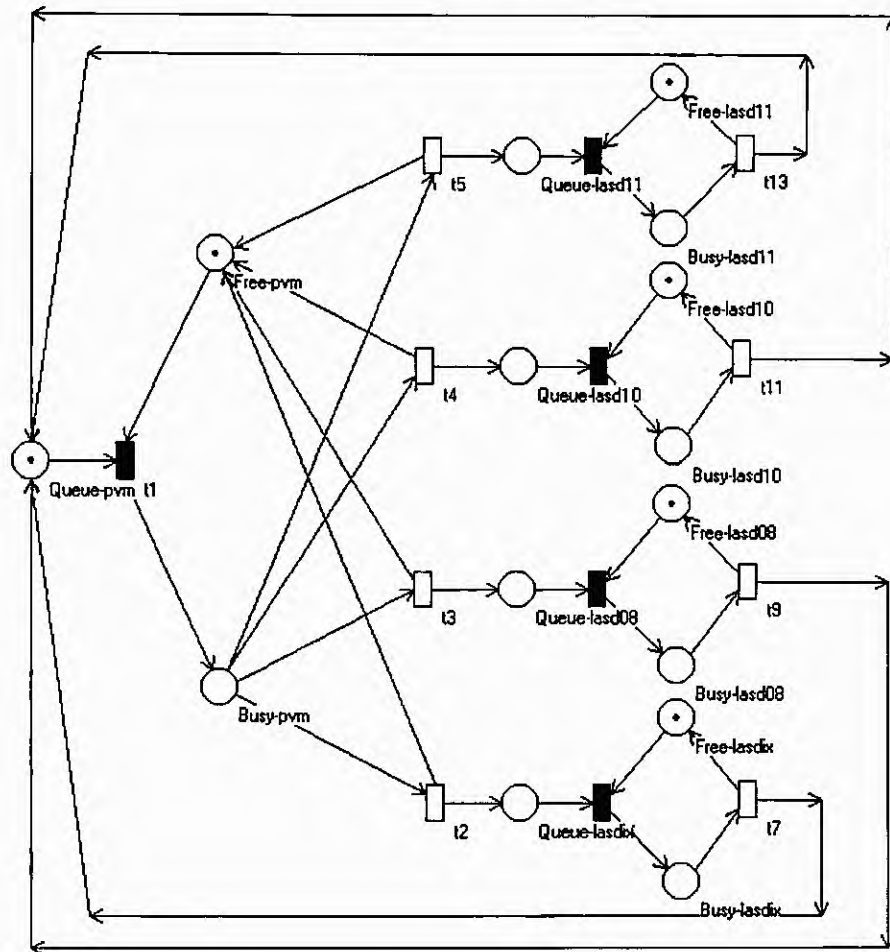


FIGURA 6.3. Especificação do Modelo PVM em GSPN.

6.3.2. Parametrização do Modelo.

O modelo possui parâmetros de entrada para os tempos médios de serviços apresentados na Tabela 6.2. Além disso, o número de clientes que circula no sistema é igual a 400 (quatrocentos) clientes, distribuídos pelo PVM com probabilidade de 25% (vinte e cinco por cento) para cada máquina. Considera-se, por esse motivo, que cada máquina recebe a visita de cerca de 100 (cem) processos.

Essa parametrização inicial fornece o conjunto de valores necessários para a geração de algumas medidas de desempenho, tais como utilização e comprimento médio de fila, provendo uma maneira de observação a longo prazo do sistema.

6.3.3. Solução do Modelo.

Nesta seção, são elaboradas tanto uma solução analítica quanto uma solução por simulação. A solução analítica utiliza o método da Análise do Valor Médio (AVM) e a simulação usa como base a linguagem *simpl*.

- A Solução pelo Método da Análise do Valor Médio (AVM)

A Análise do Valor Médio pode ser aplicada tanto a modelos abertos quanto a fechados, além de existir uma variação para centros dependentes de carga. Para modelos fechados, há duas possibilidades: o AVM exato e o aproximado (ambos diferindo pelo valor assumido para o número médio de clientes vistos no centro i , no instante em que um novo cliente chega). Pelo perfil do fenômeno estudado neste trabalho, apenas o enfoque ao AVM exato para modelos fechados será estudado. Uma discussão completa sobre as peculiaridades desse método é apresentada em (Jain, 1991).

No caso de modelos fechados, o método AVM consiste na aplicação de uma seqüência de equações, apresentada pelo algoritmo 6.1, onde os resultados são obtidos através de N iterações, sendo que N é o número de clientes no sistema.

Na obtenção do Algoritmo AVM exato, consideram-se as Leis Operacionais, discutidas no Capítulo 3, juntamente com a Equação (6.1), onde o termo $A_i(N)$ é equivalente a $N_i(N-1)$.

$$R_i(N) = S_i \cdot [1 + A_i(N)] \quad (6.1)$$

Adaptando-se a Equação (6.1) para o AVM exato, tem-se:

$$R_i(N) = S_i \cdot [1 + N_i(N-1)] \quad (6.2)$$

A seguinte semântica é atribuída à Equação (6.2): considere-se a chegada do N -ésimo cliente a um sistema com $N-1$ clientes. À chegada ao i -ésimo centro, esse cliente vê $N_i(N-1)$ clientes, constatando o número de cliente no i -ésimo centro quando existem $N-1$ clientes no sistema. Dessa maneira, o N -ésimo cliente irá esperar $N_i(N-1) \cdot S_i$ unidades de tempo até que seja atendido. Assim, o tempo de resposta para esse cliente será de $S_i \cdot [1 + N_i(N-1)]$ (Jain, 1991).

Para modelos fechados, da mesma forma que para modelos abertos, combinam-se as leis operacionais com a equação do tempo de resposta (Equação (6.2)) para a obtenção do Algoritmo AVM Exato:

- Dado o tempo de resposta de cada centro (Equação 6.2), o tempo de resposta do sistema, utilizando-se a lei geral do tempo de resposta é:

$$R(N) = \sum_{i=1}^M V_i R_i(N)$$

- O *throughput* do sistema usando a lei do tempo de resposta iterativo é:

$$X(N) = \frac{N}{R(N) + Z} \quad (6.3)$$

- O *throughput* do centro medido em termo de clientes por unidade de tempo é:

$$X_i(N) = X(N) \cdot V_i$$

- O número médio de clientes no centro, quando existem N clientes no sistema, utilizando a lei de Little é:

$$N_i(N) = X_i(N) \cdot R_i(N) = X(N) \cdot V_i \cdot R_i(N) \quad (6.4)$$

Baseando-se nas equações anteriores, desenvolve-se o algoritmo 6.1 (Figura 6.4), extraído na íntegra de (Jain,1991), que calcula algumas medidas de desempenho para modelos fechados de classe única de clientes. No caso de múltiplas classes, para cada iteração, aplicam-se para as equações do algoritmo cada uma das classes.

Para compreender o funcionamento desse algoritmo, deve-se ressaltar que, no caso de modelos fechados, o tempo de resposta do i -ésimo centro, quando existem N clientes no sistema, representado por $R_i(N)$ (Equação 6.1), é computado se usando o número médio de clientes no i -ésimo centro, quando existiam $N-1$ clientes no sistema, $N_i(N-1)$. Dessa forma, para o cálculo do tempo de resposta atual de um centro, necessita-se do número médio clientes anterior desse centro (Allen,1990) (Jain,1991).

Sendo assim, trabalhando-se com modelos fechados, o desempenho do sistema deve ser obtido por intermédio de iterações entre as medidas de desempenho, calculadas no momento em que se tem N e $N-1$ clientes no sistema. Para isso, inicia-se a avaliação do modelo considerando que o número médio de clientes é zero para todos os centros, quando o sistema ainda está vazio ($N_i(0)=0$), possibilitando calcular o tempo de resposta quando se tem 1 cliente no sistema ($R_i(1) = S_i \cdot [1 + N_i(0)]$) e assim sucessivamente, até que o número de clientes no sistema seja igual a N .

Parâmetros de Entrada:

N = número de clientes

Z = tempo de pensar

M = número de centros (sem incluir os terminais)

 S_i = tempo de serviço por visita ao i -ésimo centro V_i = Número de visitas ao i -ésimo centroParâmetros de Saída:X = taxa de chegada externa ou *throughput* do sistema N_i = número médio de clientes no i -ésimo centro R_i = tempo de resposta do i -ésimo centro

R = tempo de resposta do sistema

 ρ_i = utilização do i -ésimo centroIniciação:Para $i=1$ a M faça $N_i=0$ Iterações:

Início

Para $i=1$ a M faça $R_i = \begin{cases} S_i \cdot (1 + N_i) & (\text{Centros de capacidade fixa}) \\ S_i & (\text{Centros Delay}) \end{cases}$

$$R = \sum_{i=1}^M R_i V_i$$

$$X = \frac{N}{R + Z}$$

Para $i=1$ a M faça $N_i = X \cdot V_i \cdot R_i$

Fim

Throughputs dos centros: $X_i = X \cdot V_i$ Utilização dos centros: $\rho_i = X \cdot S_i \cdot V_i$

FIGURA 6.4. Algoritmo AVM Exato para Modelos Fechados.

Para resolver o modelo deste estudo de caso, foi implementado o algoritmo da Figura 6.3 em um programa elaborado em linguagem C (apresentado no Apêndice C), utilizando-se dos parâmetros definidos na seção 6.3.2. A Figura 6.5 apresenta um resumo das medidas obtidas como saída do método AVM exato aplicado aos modelos das Figuras 6.1 e 6.2.

```

Throughput do sistema: 0.000073

Tempo de Resposta do sistema: 5484996.000000

X[0]=> 0.029171 (throughput PVM)
U[0]=> 0.044831 (utilização PVM)
X[1]=> 0.007293 (throughput lasdix)
U[1]=> 0.990303 (utilização lasdix)
X[2]=> 0.007293 (throughput lasdpc08)
U[2]=> 0.228479 (utilização lasdpc08)
X[3]=> 0.007293 (throughput lasdpc10)
U[3]=> 0.603142 (utilização lasdpc10)
X[4]=> 0.007293 (throughput lasdpc11)
U[4]=> 0.996716 (utilização lasdpc11)

```

FIGURA 6.5. Medidas de Desempenho Obtidas pelo Método AVM.

- A Solução pela Simulação *smpl*

A simulação utilizou os mesmos parâmetros da solução analítica, realizando a associação entre eventos-padrão dos *templates* e funções *smpl* correspondentes, proposta nos Capítulos 4 e 5.

Para realizar a simulação, foram adotadas as seguintes estratégias: (1) foi adotada a técnica de replicação da simulação, na qual são feitas várias execuções da simulação, variando-se a semente da seqüência aleatória adotada em cada uma das execuções; (2) foram utilizadas todas as quinze sementes que são disponibilizadas pelo *smpl*; (3) o *warm-up* adotado foi de 10% (dez por cento), que é tido como razoável em (MacDougall, 1987); (4) o intervalo de confiança de 95% (noventa e cinco por cento) é calculado para cada execução, tomando-se um tempo de simulação de 100 (cem) milhões de unidades. Nessa primeira parte do experimento, foram adotadas probabilidades de 25% (vinte e cinco por cento) para cada máquina no sistema.

Na verdade, a replicação para as quinze sementes foi repetida para cada dez milhões de unidades de simulação, começando em dez milhões e se encerrando em cem milhões. O objetivo de executarem-se dez replicações foi atestar o comportamento da simulação também em função do efeito temporal, e não só em relação à mudança das sementes. O relatório de saída do *smpl* para cem milhões de unidades de simulação (semente 0) é apresentado na Figura 6.6.

smpl SIMULATION REPORT						
MODEL: pvm-v1			TIME: 100000005.087			
			INTERVAL: 90000002.370			
FACILITY	UTIL.	MEAN BUSY PERIOD	MEAN QUEUE LENGTH	OPERATION COUNTS		
				RELEASE	PREEMPT	QUEUE
pvm	0.0447	4024445.994	0.050	3943	0	14863
lasdix	0.9880	88916674.168	105.91	1764	0	9625
lasdpc08	0.2285	26231.198	0.295	784	0	18750
lasdpc10	0.6010	54085524.253	1.50	1098	0	96
lasdpc11	0.9939	89449585.289	292.40	1864	0	5977

FIGURA 6.6. Relatório smpl para 100 milhões de Unidades de Simulação.

A Tabela 6.3 apresenta a comparação entre os resultados obtidos pela solução analítica AVM e pela simulação smpl (e seus respectivos intervalos de confiança).

TABELA 6.3. Resultados Obtidos Analiticamente e por Simulação, onde U (utilização).

Servidor	U(AVM)	U (smpl)
PVM	0.044831	0.0447 ±0,00002837
lasdix	0.990303	0.9874 ±0,00040794
lasdpc08	0.228479	0.2284 ±0,00005365
lasdpc10	0.603142	0.6012 ±0,00038534
lasdpc11	0.996716	0.9942 ±0,00021510

Observando-se a Tabela 6.3, e tendo-se como base as medidas apresentadas, pode-se constatar uma certa distorção na utilização dos recursos. Como exemplo, pode ser citada a utilização do computador com maior potência de processamento (lasdpc08) como sendo a menor entre todas, sobrecarregando assim as demais.

A anomalia constatada na Tabela 6.3 pode ser fruto de um mecanismo ineficiente de distribuição de processos, o que intuitivamente pode sugerir que uma ponderação nas probabilidades que leve em consideração a taxa de processamento das máquinas possa ser uma alternativa mais eficiente.

Para efeito de comparação, foi simulado o modelo GSPN da Figura 6.7, utilizando-se a ferramenta DNAet (Attieh et al, 1995). Os resultados obtidos para a utilização dos recursos, além dos seus respectivos intervalos de confiança, estão sintetizados na Tabela 6.4.

TABELA 6.4. Utilizações Obtidas pela Simulação GSPN .

Servidor	Utilização
PVM	0.045800 +/- 0.008400
lasdix	0.988326 +/- 0.035238
lasdpc08	0.223423 +/- 0.011646
lasdpc10	0.580202 +/- 0.024037
lasdpc11	0.991806 +/- 0.035343

Simulando-se novamente, realizando-se a seguinte ponderação: a máquina mais lenta (lasdpc11) é o denominador de todas as relações das demais máquinas e calculando-se um índice de potência (IP) a partir de um fator comum. Assim:

- $IP(lasdix) = T_{lasdix} / T_{lasdpc11} = 135.7/136.6 = 0.99;$
- $IP(lasdpc08) = T_{lasdpc08} / T_{lasdpc11} = 31.32/136.6 = 0.23;$
- $IP(lasdpc10) = T_{lasdpc10} / T_{lasdpc11} = 82.7/136.6 = 0.60;$
- $IP(lasdpc11) = T_{lasdpc11} / T_{lasdpc11} = 136.6/136.6 = 1.$

Assim, o lasdix possui a potência de 1.01 lasdpc11, o lasdpc08 equivale a 4.35 lasdpc11 e o lasdpc10 é igual a 1,67 lasdpc11. A partir das relações anteriores, podem-se estabelecer as probabilidades ponderadas pelas potências das máquinas. Dessa forma, têm-se:

$$p_{11} + p_{ix} + p_{08} + p_{10} = 1$$

$$p_{11} + 1.01 p_{11} + 4.35 p_{11} + 1.67 p_{11} = 1$$

$$p_{11} = 0.12; p_{ix} = 0.13; p_{08} = 0.54; p_{10} = 0.21$$

A partir das probabilidades obtidas, executaram-se novamente as replicações da simulação para cem milhões de unidades com as quinze sementes smpl. Os resultados da simulação ponderada e da simulação equiprovável estão sintetizados na Tabela 6.5

Pela observação de 6.5, pode-se inferir que há uma maior utilização dos recursos disponíveis no sistema. As utilizações ficaram mais equilibradas em todas as máquinas. Cerca de metade dos processos é encaminhada para lasdpc08, pelo fato dessa máquina ser a de maior potência computacional.

TABELA 6.5. Comparação entre Resultados das Simulações Equiprováveis (1) e Ponderadas (2).

Servidor	U (1)	U(2)
PVM	0.0447 ±0,00002837	0.0836 ±0,00522877
Lasdix	0.9880 ±0,00040794	0.9378 ±0,10364316
Lasdpc08	0.2285 ±0,00005365	0.9536 ±0,00649479
Lasdpc10	0.6010 ±0,00038534	0.9738 ±0,00467259
Lasdpc11	0.9939 ±0,00021510	0.9229 ±0,00533046

O estudo simples (mas não trivial), apresentado nesta seção, pode dar uma idéia de expectativas que podem ser verificadas sem, entretanto, necessitar-se de mais um elemento que pode ser um fator degenerador no desempenho final do sistema. Há alguns software que implementam algoritmos inteligentes para escalonar processos em ambientes distribuídos. Um exemplo desses ambientes é o AMIGO (DynAMical Flexible Scheduling Environment)⁵, o qual utiliza um algoritmo de escalonamento semelhante ao apresentado aqui neste estudo de caso. Na verdade, o estudo desenvolvido neste capítulo foi inspirado na filosofia proposta pelo AMIGO. Maiores detalhes sobre o AMIGO e seus algoritmos de escalonamento podem ser obtidos em (Souza et al., 1999).

6.4. Considerações Finais

Este capítulo apresentou um experimento real servindo como base para a modelagem do sistema instrumentado empiricamente. O estudo através do processo de modelagem cria uma flexibilidade e uma posterior independência do sistema real que, em muitas situações, podem ser imperativas. As inferências feitas sobre a abstração do modelo dão uma idéia, em muitos casos bastante satisfatório, do comportamento a longo prazo do sistema, sem entretanto ser necessária uma intrusão no mesmo.

A abordagem dada no estudo de caso aplica uma estratégia que é interessante: a de validar-se a simulação através da solução analítica adotada. Quando possível, a solução analítica pode validar a simulação, e a partir daí, pode-se usar a flexibilidade da simulação para se estabelecerem deduções a respeito do comportamento futuro do sistema modelado.

⁵ Idealizado por P.S.L. de Souza em (Souza, 2000), o AMIGO é um software que permite que sejam incorporados nele diferentes algoritmos de escalonamento, cabendo ao usuário a escolha do algoritmo mais apropriado.

O estudo de caso também se propôs a analisar um tipo particular de sistemas que ainda não havia sido observado com maior rigor neste trabalho: as redes de filas fechadas, isto é, aquelas cuja geração de clientes não é externa, não havendo uma taxa de chegada. Além disso, há um número pré-definido de clientes circulando no sistema. Modelos como esses são comuns em ambientes onde há geração estática de processos, tal qual na programação distribuída provida por ambientes de passagem de mensagens.

A especificação statechart se mostra adequada aos fenômenos que possuem muitas peculiaridades a serem representadas. Inserção da idéia de possíveis estados de servidores, número de clientes nas filas, eventos condicionados às probabilidades, tempos médios de serviço, tempos médios entre chegadas, número de clientes em redes fechadas, são algumas das nuances de sistemas complexos que podem ser representadas através da notação proposta neste trabalho (tanto Statecharts Estocásticos quanto Queuing Statecharts), o que pode contribuir com uma representação mais próxima da realidade dos sistemas com um grau mais elevado de complexidade.

Capítulo 7

Conclusões

7.1. Ponderações Finais desta Tese

Este trabalho propôs uma nova filosofia para tratar a avaliação de desempenho, baseada na premissa de que a especificação de um modelo que represente um problema a ser estudado é tão importante quanto a solução que vai ser dada a esse modelo. A idéia precípua do trabalho é apresentar uma interseção entre a especificação em alto nível, utilizando-se de técnicas de representação como os Statecharts, e os métodos de solução considerados (analíticos e por simulação).

Para dar suporte à proposta, elaborou-se uma revisão bibliográfica (Capítulos 2 e 3) capaz de contemplar, de uma maneira condensada, as subáreas que compõem a avaliação de desempenho. Essa revisão visa a proporcionar um "mapa" de alguns possíveis caminhos que podem ser tomados, uma vez que se deseje enveredar pela realização da modelagem, para fins de desempenho. Um ponto importante, intencionalmente focado ao longo desta tese, é a falta de consenso tanto na questão da especificação quanto na solução aplicada ao modelo. Sem a pretensão de polemizar ou de oferecer uma panacéia ao tema, este trabalho apresenta contrapontos que, dependendo das características do sistema estudado, podem ser decisivos na adoção de uma abordagem em detrimento das demais.

Ao apresentar as divergências, tais como aquelas descritas nas seções em que se discutem "deficiências" das representações (Capítulo 2), ou quando se discutem algumas "restrições" associadas às soluções de um modelo (Capítulo 5), também são apresentados alguns pontos para os quais há uma convergência entre os especialistas da área. Em relação à especificação, por exemplo, as técnicas possuem em seu cerne a mesma filosofia: os sistemas são vistos como um conjunto de estados, sendo que a transição entre esses estados se dá pela ocorrência de eventos, seja qual for a nomenclatura usada para denominar estados e eventos.

A despeito da falta de consenso observada na área, houve a preocupação de ressaltarem-se as benesses que cada representação contém e, na medida do possível, de agrupá-las em uma técnica que fosse não apenas uma extensão à especificação Statecharts, mas uma junção de características interessantes das três mais usuais representações baseadas em espaço de estados: redes de filas, redes

de Petri e Statecharts. Além disso, houve uma preocupação em atribuir um formalismo a características pertinentes a desempenho, que são verificadas em sistemas reais, mas que, pela intenção inicial dos Statecharts, não faziam parte de sua formulação original. Assim, o Capítulo 4 descreve duas abordagens de representações baseadas na semântica original de D. Harel (Harel et al., 1987) e que, quando conveniente, foi adaptada às situações peculiares de sistemas de filas. Vale ressaltar que o próprio D. Harel, em (Harel, 1987), já havia vislumbrado a possibilidade de sua especificação ser usada para avaliar desempenho, o que ele batizou à época de Markov-charts.

Quanto à definição formal de Statecharts Estocásticos e Queuing Statecharts, houve algumas preocupações que, no decorrer dos capítulos anteriores, podem ter sido abordadas com certa parcimônia, mas que são ratificadas nesta seção. Por exemplo, na formalização do Capítulo 4, de certa forma, procurou-se utilizar notação semelhante àquela adotada por D. Harel, em (Harel et al., 1987), mantendo-se sempre que possível uma coerência com os conceitos originais, tais como passos e configurações. Também houve a intenção explícita de manter-se a representação gráfica original (exceto na representação de filas em Queuing Statecharts, onde se optou por manter a notação característica das redes de filas), modificando-se, apenas em certos casos, o significado atribuído a alguns elementos gráficos originais (por exemplo, os estados *delay*, cujo tempo associado neste trabalho é uma variável aleatória exponencialmente distribuída). A adoção de uma definição formal tenta minimizar a interpretação baseada em conjecturas, que poderiam advir na falta de uma abordagem baseada em formalismo.

Um ponto que vale a pena ser enfatizado como ponderações finais é a apresentação de uma justificativa concreta para ter-se uma especificidade dos Statecharts Estocásticos, denominada nesta tese por Queuing Statecharts. A justificativa vem do fato de a modelagem neste trabalho ser encarada como um processo complexo - como realmente o é - composto por etapas de cunho complementar, cujo conjunto, quando bem equacionado, gera um resultado satisfatório. Pensando-se na modelagem como esse processo complexo, e enfocando-se a fase de solução, pode-se observar que na maioria dos métodos analíticos adotados (método de Jackson e AVM, por exemplo), há sempre a idéia de transição, e conseqüente visita, de clientes entre centros de serviços (com uma probabilidade associada a essa transição). Dessa forma, para sistemas de filas (o

objeto deste estudo), é fundamental que haja uma representação dos centros de serviços, além do inter-relacionamento entre eles. Na notação puramente Statecharts, adotada nos Statecharts Estocásticos, o único mecanismo de relacionamento entre esses centros (que estão subentendidos pela conjunção dos superestados fila e servidor) é o *broadcasting* de comunicação realizado pela execução dos eventos e suas ações subseqüentes. Entretanto, essa notação, para esse propósito, pode não ser tão clara quanto a utilizada em Queuing Statecharts, a qual mostra todo o itinerário descrito por um cliente, durante a sua passagem pelo sistema.

Logicamente que há as implicações em utilizar-se uma especificação com o poder de representação bem mais restrito (como Queuing Statecharts), cujo conjunto de elementos sugere que seja compatível com um domínio menor de aplicações. Se acaso admite-se que se queiram representar alguns elementos adicionais, mesmo que eles não tenham impacto direto na avaliação de desempenho (que está relacionada diretamente ao estudo de filas e servidores), então a especificação Queuing Statecharts pode tornar-se limitada. Entretanto, na maioria dos estudos concernentes à avaliação de desempenho, é realizada uma abstração do sistema, sintetizando-o, basicamente, em filas e servidores, o que pode transformar a escolha em uma decisão mais de caráter subjetivo do que técnico.

Tomou-se o cuidado de apresentar a especificação sempre atrelada a uma solução viável, evitando-se a interpretação enganosa de que não há uma correspondência biunívoca entre essas duas etapas. Dessa forma, *templates* e eventos foram idealizados já se pensando em suas correspondências nos métodos de solução adotados. Essa associação é possível graças ao fato de que os sistemas de filas funcionam com uma certa homogeneidade, no que concerne aos eventos e estados constantes nesse tipo de sistema.

Quanto à situação tomada como estudo de caso, novamente propositadamente, optou-se por retratar um sistema real, cuja "sensibilidade" era aguçada o bastante, a ponto de uma aferição *in loco* poder distorcer as medidas resultantes dessa instrumentação. Esse é um caso típico no qual a inferência a partir de uma abstração pode levar a considerações fundamentais ao experimento. Na programação distribuída baseada em PVM, como a apresentada no Capítulo 6, que é realizada em um ambiente de rede local, onde há um conhecimento prévio das

máquinas envolvidas, pode-se fazer uma distribuição mais aprimorada, levando-se em consideração a potência computacional, como foi sugerido no estudo de caso. Entretanto, os problemas envolvidos no estudo de escalonamento de processos transcendem as especulações efetivadas naquele capítulo, pois há muito mais variantes além do tempo médio de serviço, o que não invalida a intenção do estudo, que estava mais direcionado à flexibilidade oferecida pela modelagem e sua aplicabilidade em situações reais, do que em oferecer resultados conclusivos ao problema estudado.

Ao final dessas ponderações é importante citar que, pelo seu forte teor matemático, a aplicação do processo de modelagem não é uma tarefa trivial, o que às vezes inviabiliza a sua utilização em problemas cotidianos. Nesse contexto, ferramentas que otimizem o processo podem ser bem vindas. Algumas das ferramentas utilizadas aqui, tais como o ASiA¹, DNANet e BestFit², mesmo que contribuam em apenas uma parte do processo, podem ser de grande valia, ressaltando-se que elas são, em geral, de uso relativamente simplificado, e que elas "escondem" de certa forma a complexidade inerente à modelagem. Entretanto, é importante observar que o ato de ocultar a complexidade do usuário deve ser bastante ponderado, pois a ocultação total pode levar o usuário a desconhecer detalhes que são decisivos nos resultados obtidos, tais como o método analítico adotado ou características como *warmup*, semente e intervalo de confiança para a simulação.

7.2. Contribuições desta Tese

Durante todo o texto desta tese, existiu a preocupação de abordar os aspectos gerais da área de avaliação de desempenho, e mais do que realizar uma exposição teórica, houve a intenção de demonstrar a aplicação em situações computacionais mais específicos, apesar da especificação proposta ser de caráter geral. Esse direcionamento se deu em função de que apesar da área de computação possuir um caráter majoritariamente estocástico, não se usa como praxe a avaliação de desempenho.

¹ O Ambiente de Simulação Automático (ASiA) é uma ferramenta que a partir da confecção de um modelo em redes de filas, através de uma interface gráfica, é gerado o código em smpl, além de um relatório com algumas medidas de desempenho. Para maiores referências sobre o ASiA, consultar (Santana et al., 1996).

² BestFit é marca registrada de Palisade Corporation, e se destina, entre outras opções, a realizar testes de aderência sobre um conjunto de dados, apontando um *rank* das possíveis distribuições que aqueles dados descrevem.

Partindo-se do princípio de que modelagem e avaliação de desempenho são perfeitamente aplicáveis à computação, este trabalho oferece uma série de contribuições:

- A realização de uma análise crítica da bibliografia existente, tanto no que tange à especificação quanto à solução aplicada, o que costuma ser tratado como dois assuntos distintos, sem interseção (com raras exceções, como as redes de Petri estocásticas e suas extensões). O fato de tratar esses dois assuntos como complementares sugere um melhor entendimento do processo de modelagem como um todo.
- O levantamento das divergências relativas tanto à especificação quanto à solução leva à discussão das vantagens e desvantagens de cada abordagem. Esse levantamento foi de fundamental importância para a incorporação de conceitos que foram adotados nas especificações propostas neste trabalho.
- A ratificação da taxonomia proposta em (Santana et al., 1997) como uma tentativa de minimizar o efeito negativo gerado pela falta de uma denominação padrão para as técnicas de avaliação de desempenho.
- A formalização de conceitos associados ao caráter aleatório presente nos sistemas reais, permitindo, dessa forma, a utilização da especificação Statecharts para prover avaliação de desempenho. Essa formalização foi realizada através de duas vertentes: a primeira para adicionar os conceitos estocásticos à especificação genérica dos Statecharts e a segunda para criar uma especificidade da primeira, com um escopo totalmente voltado às redes de filas. Dentre as definições inseridas na formalização, podem ser citadas:

1) Para os Statecharts Estocásticos:

- Escolha por condição probabilística e evento probabilístico;
- Estados imediatos e estados com retardo associado;
- *Templates* para fonte geradora de clientes, fila, servidor e sorvedouro;
- Eventos-padrão (e suas respectivas ações associadas) para cada *template*;
- Variáveis aleatórias *tempo entre chegadas* para o estado fonte e *tempo de serviço* para os servidores;
- Configuração sucessora e próximo passo, redefinidos agora em função do tempo decorrido, modificando a dinâmica original dos modelos Statecharts;

- Em consequência da redefinição anterior, possibilidade de avaliação do estado transiente do sistema, estendendo a visão de observar-se apenas o estado de equilíbrio.
- 2) Para os Queuing Statecharts, além de herdar as definições formalizadas para os Statecharts Estocásticos, são acrescentadas:
- Os centros de serviço, componentes básicos dos sistemas de filas;
 - Filas herdando a notação utilizada nas redes de filas, incluindo-se a idéia da variável aleatória Q (número médio de clientes na fila) e suas operações básicas de acréscimo e decréscimo;
 - *Templates* redefinidos para fila, centro de serviço e estados parametrizados para servidores;
 - *Templates* para servidores hiperexponencial e de Erlang;
 - Redefinição dos eventos-padrão (e suas ações associadas) para cada *template*;
 - Redefinição da idéia de broadcasting de comunicação, agora reforçado pelas setas que unem os componentes do modelo, e não apenas pelo disparo de eventos e ações.
- A realização da associação entre as especificações propostas e algumas soluções existentes, através da padronização de eventos e estados constantes em sistemas de filas, contribuem para que seja factível a avaliação de desempenho a partir de modelos baseados na representação em alto nível dos Statecharts. A proposição de uma especificação que fosse passível de solução foi uma preocupação constante no decorrer desta pesquisa, o que demonstra a intenção de criar-se uma representação que realmente seja aplicável a situações que requeiram um maior nível de detalhamento, em função de uma maior complexidade do sistema estudado.
 - A associação tratada no item anterior foi apresentada neste trabalho através das soluções analíticas de Jackson (para modelos abertos), análise do valor médio – AVM (para modelos fechados, no exemplo descrito nesta tese) e simulação *smpl* (para ambos os casos). Nas duas abordagens de solução, não houve a necessidade de modificações nos *templates* ou em seus eventos-padrão, o que ratifica o teor genérico idealizado para as especificações propostas.

- A criação de uma nova abordagem gráfica para a área de avaliação de desempenho no grupo de Sistemas Distribuídos e Programação Concorrente do ICMC-USP, possibilitando que novas pesquisas sejam elaboradas a partir (ou fazendo uso) deste trabalho.
- O caráter interinstitucional da pesquisa, envolvida diretamente com o Laboratório de Matemática e Computação Aplicada (LAC), do Instituto Nacional de Pesquisas Espaciais (INPE), ressaltando-se que o LAC-INPE é uma instituição de referência na área de pesquisa operacional.
- As publicações geradas tendo como base esta pesquisa, que se constituem em uma boa maneira de divulgação do trabalho à comunidade acadêmica, além de (e principalmente) ser uma ótima forma para a obtenção de críticas e sugestões ao trabalho, algumas das quais foram assimiladas e incorporadas à pesquisa. São publicações relacionadas a esta tese:
 - ✓ FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C. Statecharts Estocásticos e Queuing Statecharts: Novas Abordagens para Avaliação de Desempenho Baseadas em Especificação Statecharts. 15º Simpósio Brasileiro de Engenharia de Software (SBES2001), Rio de Janeiro, novembro de 2001. (artigo aceito, por publicar).
 - ✓ VIJAYKUMAR, N. L., FRANCÊS, C. R. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C., ABDURAHIMAN, V. Analytical and Simulation Solutions for Performance Models Based on Statecharts: a Case Study of a File Server. In: the 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001), 2001, Orlando, 2001 (artigo aceito, por publicar).
 - ✓ FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C., ABDURAHIMAN, V. Performance Evaluation using Stochastic Extension to Statecharts: a Case Study to a File Server. In: Eurosim 2001 Congress, June 26 - 29, 2001, GA Delft, 2001.
 - ✓ FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, R. H. C., SANTANA, M. J., CARVALHO, S. V., ABDURAHIMAN, V. The Use of Analytical and Simulation Solutions with Statecharts for Performance Evaluation: A Case Study of a File Server Model. In: 2nd IEEE LATIN-AMERICAN TEST WORKSHOP - LATW2001, 2001, Cancun - Digest of Paper, 2001. p.136 - 141.

- ✓ FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C. Stochastic Extension to Statecharts for Representing Performance Models: an Application to a File Server. In: The International Conference on Information Society in the 21 Century: Emerging Technologies and New Challenges, 2000, Aizu-Wakamatsu City, 2000. p.455 - 462.
- ✓ FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C. Stochastic Extension to Statecharts for Representing Performance Models: an Application to a File System. In: Simpósio Brasileiro de Arquitetura de Computadores e Computação de Alto Desempenho, 2000, São Pedro-SP. In the Proceedings of the SBAC-PAD'2000.
- ✓ FRANCÊS, C. R. L., SANTANA, M. J., SANTANA, R. H. C. Statecharts and Markov Chains for Performance Evaluation: Applications and Implications. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), 1999, Las Vegas. In the Proceedings of The PDPTA, 1999. v.III. p.1606 - 1612.

7.3. Sugestões para Trabalhos Futuros

As especificações apresentadas nesta tese formam um arcabouço básico para o estudo de sistemas de filas. Generalizações ou especificidades que se adequem a determinadas situações reais são factíveis e desejáveis. Algumas dessas generalizações e especificidades, apresentadas como sugestões para trabalhos futuros, são:

- A representação e o tratamento de posse simultânea de recursos, que não foram estudados neste trabalho e poderiam ser acrescentados na formalização proposta no Capítulo 4.
- O tratamento de distribuições que não sejam combinações da exponencial (hiperexponencial e de Erlang), haja vista que a formalização realizada neste trabalho é inteiramente baseada no comportamento da distribuição exponencial. Em um estudo empírico feito envolvendo o relacionamento entre o PVM e o AMIGO, verificou-se a existência das duas características abordadas, ou seja, o PVM permanecia bloqueado até que o AMIGO retornasse o controle ao PVM. Assim, uma observação instantânea do sistema apresentaria o mesmo cliente mobilizando tanto os serviços do PVM quanto do AMIGO (posse múltipla). Também foi verificado que os tempos de serviços do PVM, AMIGO e de alguns

processadores descreviam uma composição de duas distribuições Gama, o que necessitaria de uma redefinição das variáveis aleatórias tempo entre chegadas e tempo de serviço.

- A utilização de métodos analíticos que tratem distribuições diferentes da exponencial, tal como as redes BCMP (apresentadas no Apêndice C), que ainda permitem o estudo de modelos abertos ou fechados e diferentes classes de clientes.
- A proposição de um *template* para fonte de geração de clientes que possibilite a geração em lote (*batch*) desses clientes. O formalismo inicial proposto neste trabalho é direcionado ao padrão Poisson de chegada, que descreve com propriedade uma boa quantidade de situações reais. Entretanto, situações específicas como as aplicações envolvendo programação distribuída podem requerer a especificação de geração em lotes de clientes, denominada por alguns autores, por exemplo (Jain, 1991), de *train arrivals*.
- Incorporação da especificação Queuing Statecharts na interface gráfica do Ambiente de Simulação Distribuída Automático (ASDA) (Bruschi et al., 2000), projeto de doutorado desenvolvido no Grupo de Sistemas Distribuídos e Programação Concorrente do ICMC-USP, cuja incorporação já se encontra em andamento.
- A junção em um único ambiente de um simulador (possivelmente baseado em *smpl*) e uma biblioteca para resolver modelos através de cadeias de Markov de parâmetro contínuo, desenvolvida no LAC-INPE (Andrade et al., 1997), de maneira a disponibilizar duas possíveis soluções para o mesmo modelo, cuja escolha entre ambas poderia ser decidida através de algum critério de inferência realizada sobre o modelo.
- Estudo de alguns sistemas complexos reais como o de telefonia móvel, no qual um serviço de uma ligação pode ser proporcionado por vários servidores (antenas), o que poderia caracterizar uma distribuição de Erlang. Um outro estudo poderia ser a avaliação de desempenho de servidores Web, utilizando-se as especificações propostas nesta tese, aproximando-se a distribuição de calda pesada (bastante utilizada para descrever o comportamento desses tipos de servidores) por uma distribuição hiperexponencial, conforme apontam estudos apresentados em (Menascé & Almeida, 1998) e (Crovella & Taqqu, 1999). Em

suma, vários estudos de sistemas cotidianos que necessitam de especificação mais detalhada poderiam ser conduzidos a partir da abordagem Statecharts.

- Inclusão da idéia de individualização de clientes, como utilizada em redes de Petri através dos *tokens*, com o objetivo de referir-se a um cliente em particular. Há situações em que essa característica é bastante recomendável, tal como em escalonamento com migração de processos, no qual se deve saber exatamente qual processo deve ser migrado. Além da individualização de clientes, cada um deles pode ter atributos diferentes, o que remete à estratégia utilizada nas redes de Petri coloridas, nas quais cores podem indicar desde classes diferentes de clientes até estruturas internas complexas para os *tokens* (tal qual o registro de algumas linguagens de programação). Statecharts marcados e coloridos podem ser algumas extensões ao formalismo proposto neste trabalho.

Referências Bibliográficas

- ALLEN, A. O. *Probability Statistic and Queueing Theory - with Computer Applications*. San Diego, Academic Press, 1990.
- ANDRADE, V.M.B., CARVALHO, S.V., VIJAYKUMAR, N.L. Desenvolvimento de um Software para análise de desempenho de sistemas através de modelos markovianos. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 29, Salvador, 1997. *Anais*. Salvador, SOBRAPO, 1997.
- ATTIEH, A., BRADY, M.C., KNOTTENBELT, W.J., KRITZINGER. (1995, May). Functional and Temporal Analysis of Concurrent Systems. <http://www.cs.uct.ac.za/~william/DNAnet.html>.
- BEGUELIN, A.. *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing*. The MIT Press, 1994.
- BRUSCHI, S. M.; Santana, R. H. C.; Santana, M. J. ASDA - An Automatic Distributed Simulation Environment. In: 2000 Summer Computer Simulation Conference(SCSC'2000), Vancouver, 2000. Proceedings. Vancouver, SCSC, 2000.
- CHIOLA, G., MARSAN M. A., CONTE, G. Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications. *IEEE Transactions on Software Engineering*, vol. 19, n. 2, p. 89-106, 1993.
- CLARKE, A. B., DISNEY, R. L. *Probability and Random Processes - A First Course with Applications*. 2ª. Ed., New York, JOHN WILEY Professional, 1985.
- CORTÉS, O. A. C. *Desenvolvimento e Avaliação de Algoritmos Numericos Paralelos*. Dissertação (Mestrado). ICMC-USP, São Carlos, 1999.
- CROVELLA, M.E., TAQQU, M.S. Estimating the Heavy Tail Index from Scaling Properties. *Methodology and Computing in Applied Probability*, v. 1, n. 1, p. 55-79, 1999.

DANTAS, C.A.B. *Probabilidade: Um Curso Introdotório*. São Paulo, EDUSP, 1997.

FERREIRA, A.B.H. *Novo Dicionário Aurélio – Século XXI*. Rio de Janeiro, Nova Fronteira, 1999.

FRANCÊS, C.R.L. *Stochastic Feature Charts - Uma Extensão Estocástica para os Statecharts*. São Carlos, 1998. 125p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo (USP).

FRANCÊS, C.R.L, SANTANA, M.J., SANTANA, R.H.C. Stochastic Statecharts for Performance Evaluation: Applications and Implications. *In: International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, 1999. *Proceedings*. Las Vegas, PDPTA'99, v.III. p.1606 – 1612, 1999.

FRANCÊS, C. R. L, VIJAYKUMAR, N. L., SANTANA, M. J., CARVALHO, S. V., SANTANA, R. H. C. Stochastic Extension to Statecharts for Representing Performance Models: an Application to a File System. *In: Simpósio Brasileiro de Arquitetura de Computadores e Computação de Alto Desempenho, São Pedro, 2000*. *Proceedings*. São Pedro, SBAC-PAD, p-365-37, 22000.

FRANCÊS, C. R. L., VIJAYKUMAR, N. L., SANTANA, R. H. C., SANTANA, M. J., CARVALHO, S. V., ABDURAHIMAN, V. The Use of Analytical and Simulation Solutions with Statecharts for Performance Evaluation: A Case Study of a File Server Model. *In: 2nd IEEE LATIN-AMERICAN TEST WORKSHOP - LATW2001, Cancun, 2001*. *Digest of Paper, Cancun, LATW*, p.136 – 14, 2001.

HAREL, D. Statecharts: a Visual Formalism for Complex Systems. *Science of Computer Programming*, n. 8, p. 231-74, 1987.

HAREL, D., PNUELI, A., SCHMIDT, J., SHERMAN, R. On the formal semantics of Statecharts. *IEEE Symposium on Logic, In Computer Science*, Ithaca, s.n, p. 54-64, 1987.

HAREL, D., POLITI, M. *Modeling Reactive Systems with Statecharts*. s.l., MCGRAW HILL TRADE, 1998.

JACKSON, J.R. Networks of Waiting Lines. *Operations Res.*, 5(4), p. 518-521, 1957.

JAIN, R. *The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation e Modeling*. s.l., John Wiley & Sons, Inc, 1991.

JENSEN, K.; HUBER, P.; SHAPIRO, R. M. Hierarchies in Coloured Petri Nets. *Lectures Notes in Computer Science*, v.483, p. 313-341, 1990.

JENSEN, K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. New York, v. 1, Springer-Verlag, 1992.

KANT, K. *Introduction to Computer System Performance Evaluation*. New York, McGraw-Hill, 1992.

KLEINROCK, L. *Queueing Systems - Volume I: Theory*, Wiley-Interscience, s.l., Wiley-Interscience, 1975.

KLEINROCK, L. *Queueing Systems - Volume II: Computer Applications*. s.l., Wiley-Interscience, 1976.

KOVACS, ZSOLT L. *Teoria da Probabilidade e Processos Estocásticos*. s.l., Edição ACADÊMICA, 1996.

LAZÓWSKA, E.; ZAHORJAN, J.; GRAHAM; G. SEVICK, K. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models*. New Jersey, Prentice-Hall, Inc, 1984.

MACIEL, P.R.M., LINS,R.D., CUNHA, P. R. F. *Introdução às Redes de Petri e Aplicações*. Campinas, 10ª Escola de Computação, 1996.

MCBRYAN, O. A. A Overview of Message Passing Environments. *Parallel Computing*, v. 20, p. 417-444, 1994.

MACDOUGALL, M.H. *Simulating Computing Systems Techniques and Tools*, Massachusetts, MIT PRESS, 1987.

MARÍAS, J. *Introdução à Filosofia*. 4^a ed., s.l., Livraria Duas Cidades, 1985.

MENASCÉ, D.A., ALMEIDA, V.A.F. *Capacity Planning for Web Performance – Metrics, Models, and Methods*. New Jersey, Prentice Hall, PTR, 1998.

MOLLOY, M.K. Performance Evaluation Using Stochastic Petri Nets. *IEEE Trans. Comput.*, v. C-31, n. 9, p. 913-17, 1982.

PETERSON, J.L. *Petri Nets: an Introduction*. s.l., Prentice Hall, Inc., 1981.

PETRI, C.A. *Kommunikation mit Automaten*. Schriften des IIM Nr. 2, Institut für Instrumentelle Mathematik, Bonn, 1962. English Translation: Technical Report RADC-TR-65-377, Griffiths Air Force Base, New York, v. 1, Suppl.1, 1966.

PRADO, D. *Teoria de Filas e Simulação*. Belo Horizonte, EDG, 1999.

SANTANA, M.J.; SANTANA, R.H.C.; FRANCÉS, C. R. L.; ORLANDI, R.C. Tools and Methodologies For Performance Evaluation of Distributed Systems – A Comparison Study. In The Proceedings of the: SUMMER COMPUTER SIMULATION CONFERENCE, Arlington, Virginia, 1997. *Proceedings*. Arlington, p. 124-28, 1997.

SANTANA, M.J.; SANTANA, R.H.C.; SPOLON, R.; SPOLON, R. Automatic Generation of Discret-System Simulation Programs. In The Proceedings of the SUMMER COMPUTER SIMULATION CONFERENCE, Portland, Oregon, 1996. *Proceedings*. Portland, p. 133-38, 1996.

SILVA, E.A.S., MUNTZ, R.R. *Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação*. Gramado. Instituto de Informática da UFRGS, 1992.

- SILVA, A. R. F. *Modelos de redes de filas para Sistemas Computacionais Distribuídos-Simulação X Métodos Analíticos*. São Carlos, 2000. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação (ICMC), Universidade de São Paulo (USP).
- SOARES, J. F., FARIAS, A. A., CESAR, C. C. *Introdução à Estatística*. Rio de Janeiro, Guanabara Koogan S.A., 1991.
- SOARES, L. F.G. *Modelagem e Simulação Discreta de Sistemas*. Rio de Janeiro, Campus Ltda, 1992.
- SOUZA, P.S.L., Santana, M.J., Santana, R.H.C., A New Scheduling Environment for Near-Optimal Performance. In: International Conference on Parallel and Distributed Processing Techniques and Applications - PDPTA'99, Las Vegas, 1999.Proceedings. Las Vegas, PDPTA, 1999.
- Souza, P. S. L. *AMIGO: Uma Contribuição para a Convergência na Área de Escalonamento de Processos*. Tese (Doutorado). IFSC-USP, São Carlos, 2000.
- TANENBAUM, A. S. *Distributed Operating Systems*. s.l., Prentice Hall International Inc., 1995.
- TANENBAUM, A. S. *Computer Networks*. 3^a ed., New Jersey, Prentice Hall, Inc., 1996.
- VIJAYKUMAR, N.L., CARVALHO, S.V., ABDURAHIMAN, V. *Statecharts: Their Use in Specifying and Dealing with Performance Models*. São José dos Campos, 1999. Tese (Doutorado) – Instituto Tecnológico da Aeronáutica (ITA).
- WALRAND, J. *An Introduction to Queueing Networks*. New Jersey, Prentice Hall, 1990.

Apêndice A

Inferência Estatística para o Estudo de Caso

Este apêndice apresenta os testes de aderência realizados nas amostras dos tempos de serviço de cada servidor (PVM e máquinas lasdpc). Com o intuito de otimizar o processo, foi utilizada a ferramenta BestFit 4.04®. Essa ferramenta realiza vários testes de aderência (*goodness-of-fit*), tais como o qui-quadrado, o Anderson-Darling (A-D) e o Kolmogorov-Smirnov (K-S). Os três testes são discutidos com abrangência em (Allen, 1990). Neste trabalho, foi utilizado o K-S teste, conforme implementado em BestFit®, cujo algoritmo é apresentado a seguir e, posteriormente, o K-S é aplicado a cada distribuição obtida empiricamente.

O K-S é um teste aplicável a distribuições contínuas. Ele compara os valores de uma amostra empírica com os esperados para uma distribuição conhecida (neste trabalho, a exponencial). Essa comparação é feita, tomando-se como base a distância vertical entre cada ponto obtido empiricamente e o valor teórico que seria esperado para a distribuição. Essa diferença (usualmente representada pela letra D^1) é calculada como especificado a seguir:

$$D^+ = \sup_x \{S_n(x) - F(x)\},$$

$$D^- = \sup_x \{F(x) - S_n(x)\},$$

$$D = \max \{D^+, D^-\}.$$

Onde $S_n(x)$ é o valor empírico e $F(x)$ é o valor teórico esperado.

Após calculado o valor de D , deve-se escolher um nível de significância (α) para o teste (usualmente 0,05 ou 0,01, mas podendo ser qualquer valor entre 0 e 1). A idéia do α é estabelecer um valor de probabilidade para a região de rejeição (também chamada de região crítica), na qual o K-S não garante a hipótese que se deseja testar. O valor encontrado para o nível de significância α é denominado de C-valor (por referir-se à região crítica). A hipótese admitida nos testes realizados a seguir é: rejeitar-se a hipótese que a população é exponencial, se o valor de D exceder o C-valor. Caso contrário ($D < \text{C-valor}$), aceita-se a hipótese da população exponencial pelo K-S teste.

¹ Alguns autores utilizam a letra K para representar a mesma diferença, tal qual R. Jain (Jain, 1991).

- PVM

A partir da amostra de 100 (cem) tempos de serviço coletados para o PVM, ilustrado na função densidade de probabilidade da Figura A.1, admitindo-se um $\alpha = 0,05$, tem-se a aceitação da hipótese da população exponencial ($D < C\text{-Valor}$).

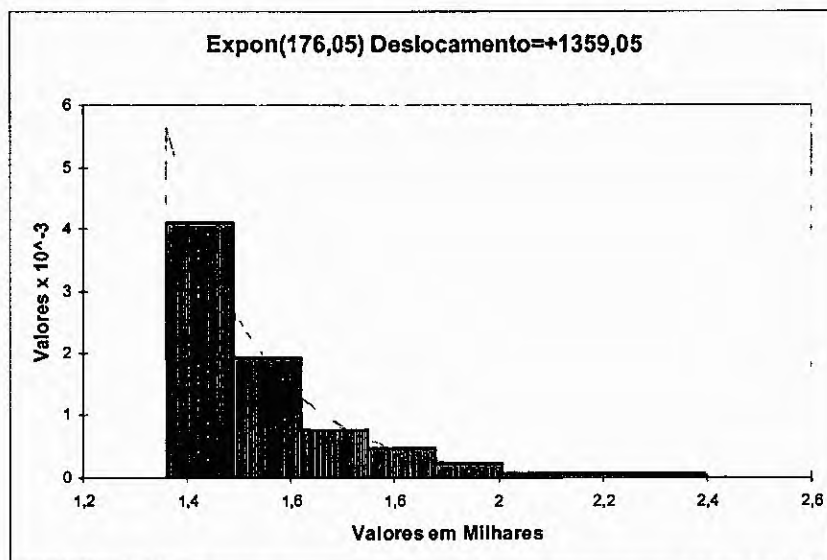


FIGURA A.1. Função Densidade de Probabilidade para o Serviço PVM.

O C-valor obtido (para $\alpha = 0,05$) foi 0,1078 e o valor de D foi 0,0298, logo $D < C\text{-valor}$; portanto aceita-se a hipótese da população exponencial, para o serviço PVM, pelo K-S teste.

- Lasdix

Para os 100 (cem) tempos obtidos utilizando-se a máquina Lasdix, gráfico (Figura A.2) e K-S teste são apresentados a seguir.

C-valor (para $\alpha = 0,05$) é igual a 0,1078 e D obtido é 0,0731, logo $D < C\text{-valor}$; portanto aceita-se a hipótese da população exponencial, para o serviço Lasdix, pelo K-S teste.

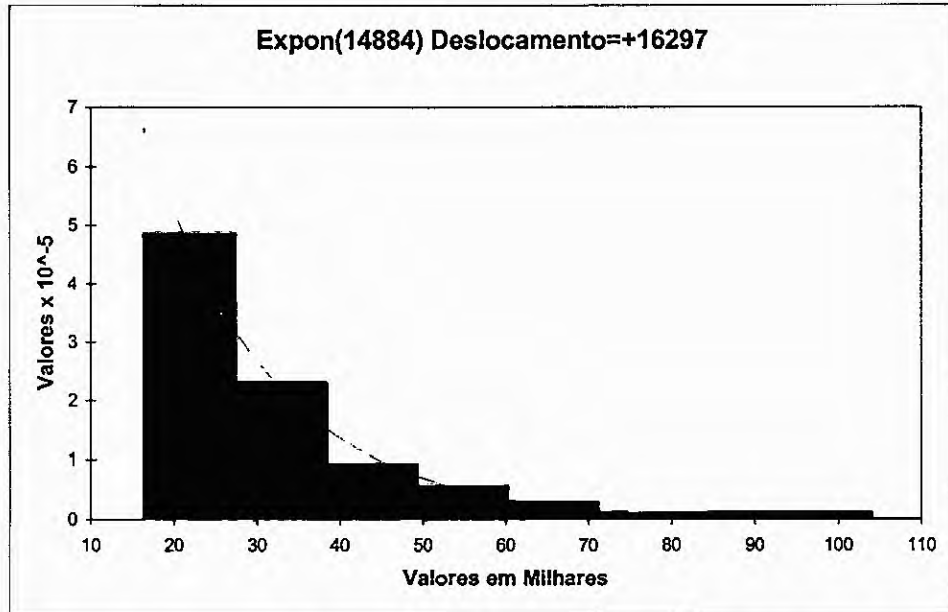


FIGURA A.2. Função Densidade de Probabilidade para o Serviço Lasdix.

- Lasdpc08

Para os 100 (cem) tempos obtidos utilizando-se a máquina Lasdpc08, gráfico (Figura A.3) e K-S teste são apresentados a seguir.

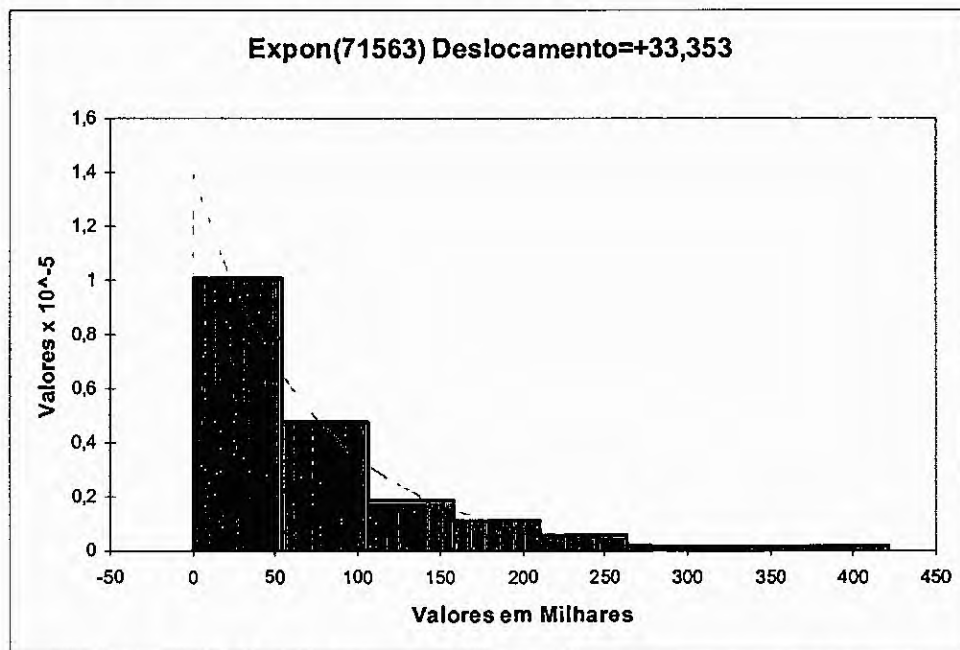


FIGURA A.3. Função Densidade de Probabilidade para o Serviço Lasdpc08.

C-valor (para $\alpha = 0,05$) é igual a 0,1078 e D obtido é 0,0640, logo $D < C$ -valor, portanto aceita-se a hipótese da população exponencial, para o serviço Lasdpc08, pelo K-S teste.

- Lasdpc10

Para os 100 (cem) tempos obtidos utilizando-se a máquina Lasdpc10, gráfico (Figura A.4) e K-S teste são apresentados a seguir.

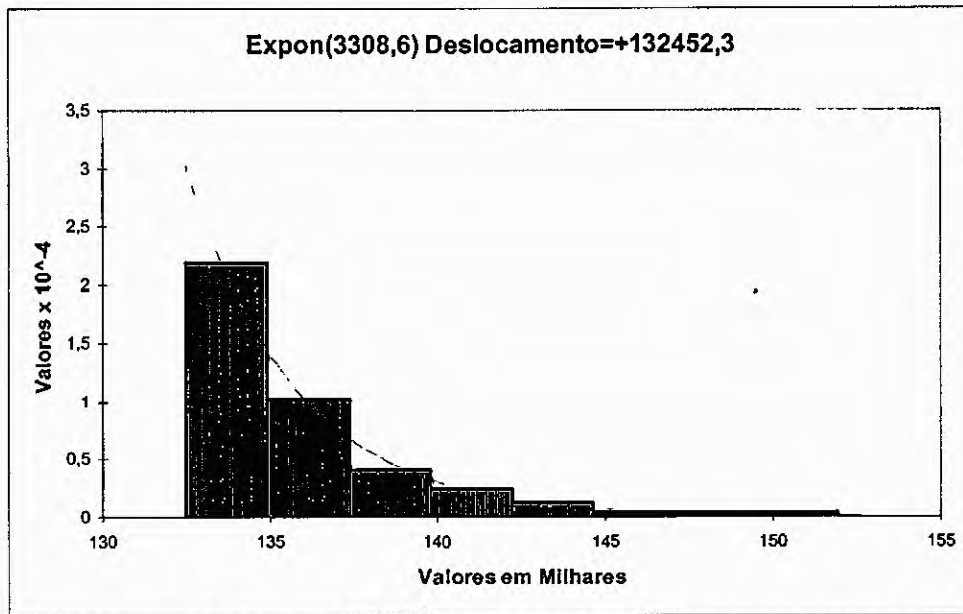


FIGURA A.4. Função Densidade de Probabilidade para o Serviço Lasdpc10.

C-valor (para $\alpha = 0,05$) é igual a 0,1078 e D obtido é 0,0501, logo $D < C$ -valor, portanto aceita-se a hipótese da população exponencial, para o serviço Lasdpc10, pelo K-S teste.

- Lasdpc11

Para os 100 (cem) tempos obtidos utilizando-se a máquina Lasdpc11, gráfico (Figura A.5) e K-S teste são apresentados a seguir.

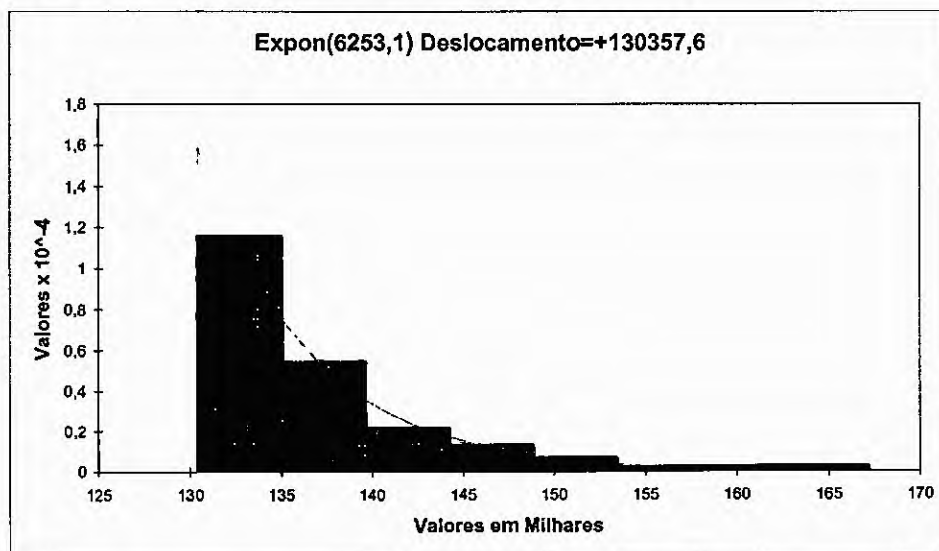


FIGURA A.5. Função Densidade de Probabilidade para o Serviço Lasdpc11.

C-valor (para $\alpha = 0,05$) é igual a 0,1078 e D obtido é 0,0715, logo $D < C\text{-valor}$, portanto aceita-se a hipótese da população exponencial, para o serviço Lasdpc11, pelo K-S teste.

É importante ressaltar que a ferramenta utilizada (BestFit®) também apresenta um *Rank* (classificação) das possíveis distribuições para a amostra que está sendo estudada. Em todas as cinco amostras analisadas neste trabalho, a distribuição exponencial foi classificada em primeiro lugar no *Rank*, tanto pelo K-S quanto pelo A-D teste.

Apêndice B

smpl – Principais Funções, Códigos-fonte e Intervalo de Confiança

Este apêndice apresenta uma relação das principais funções da extensão funcional smpl, extraída de (MacDougall, 1987), algumas das quais foram utilizadas no código-fonte da simulação do modelo do estudo de caso (Capítulo 6), que também é apresentado neste apêndice. O algoritmo para cálculo do intervalo de confiança, referenciado em alguns capítulos desta tese, é ilustrado a seguir.

◆ Funções smpl

As principais funções smpl, além das semânticas atribuídas a elas, estão ilustradas na Tabela B.1.

TABELA B1. Principais Funções Utilizadas em smpl.

INICIAÇÃO	
smpl (0, s)	Inicia subsistema de simulação
DEFINIÇÃO, OPERAÇÃO E DE FACILITIES	
f=facility (s, n)	Define e retoma descritor
r=request(f, j, p)	Reserva <i>facility</i> : r=1 se requisição for enfileirada
r=preempt(f, j, p)	Realiza preempção em um recurso: r =1 se requisição for enfileirada
release(f, j)	Libera um recurso e desenfileira a requisição
r=status(f)	Retorna o <i>status</i> atual do recurso: r =1 se recurso está ocupado
n=inq(f)	Retorna o comprimento atual da fila
u=U(f)	Retorna a utilização de um recurso
b=B(f)	Retorna o período médio de permanência em um recurso
l=Lq(f)	Retorna o comprimento médio da fila
ESCALONAMENTO DE EVENTOS	
schedule(e,t, j)	Escalona um evento
cause(e,j)	Causa um evento
j=cancel(e)	Cancela um evento e retorna o <i>token</i>
t=time()	Retorna o tempo corrente da simulação
Onde: <i>e</i> é um ponteiro para um evento, <i>j</i> é um ponteiro para um cliente, <i>f</i> diz respeito a uma <i>facility</i> (recurso), <i>t</i> é tempo em que um evento <i>e</i> deve ocorrer.	

◆ Códigos-Fonte Utilizados no Estudo de Caso

A seguir é apresentado o código-fonte utilizado na primeira parte do estudo de caso, na qual as máquinas são equiprováveis (tal qual o funcionamento do PVM).

```

/* -----
*/
/* ----- Codigo PVM Puro -----
*/
#include <dir.h>
#include <stdio.h>
#include <conio.h>
#include "smpl.h"
#include "rand.h"

main()
{
  /* definicoes */
  float Te = 100000000;
  char Flag_Reset = 1;
  int Event = 1, Customer = 1, Aleatorio, i = 8;
  real Ts1 = 1.53686, Ts2 = 31.32981,
        Ts3 = 82.705, Ts4 = 135.7939,
        Ts5 = 136.6733;

  int Server1, Server2, Server3, Server4, Server5;
  FILE *p, *saida;
  saida = fopen("pvm2.out", "w");
  if ((p = sendto(saida)) == NULL)
    printf("Erro na saída\n");

  /* prepara o sistema de simulacao e dah nome ao modelo */
  smpl(0, " pvm-v1");

  /* cria e dah nome aas facilidades (recursos) */
  Server1 = facility("pvm",1);
  Server2 = facility("lasdix",1);
  Server3 = facility("lasdpc08",1);
  Server4 = facility("lasdpc10",1);
  Server5 = facility("lasdpc11",1);

  /* escalona a chegada dos clientes */
  for (i=0; i<=100; i++){
    schedule(1,0, Customer);}

  while ( (time() < Te) ) /* Estabelece critério de parada da simulação */

    if ((time() >= 10000000) && (Flag_Reset)) /* Descarta o Warmup */
    {
      reset ();
      Flag_Reset = 0;

      cause(&Event, &Customer);
      switch(Event)
    {

```

```

case 1:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    schedule(2,0.0,Customer);
    break;
case 2:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    if (request(Server1,Customer,0) == 0)
        schedule(3,expntl(Ts1),Customer);
    break;
case 3:
    release(Server1, Customer);
    Aleatorio = random(1,10000); /* Escolha equiprovavel das maquinas */
    if ( (0 <= Aleatorio) && (Aleatorio <= 2500) )
        schedule(6, 0.0, Customer);
    if ( (2501 <= Aleatorio) && (Aleatorio <= 5000) )
        schedule(8, 0.0, Customer);
    if ( (5001 <= Aleatorio) && (Aleatorio <= 7500) )
        schedule(10, 0.0, Customer);
    if ( (7501 <= Aleatorio) && (Aleatorio <= 10000) )
        schedule(4, 0.0, Customer);
    break;
case 4:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    if (request(Server2,Customer,0) == 0)
        schedule(5,expntl(Ts2),Customer);
    break;
case 5:
    release(Server2, Customer);
    schedule(2, 0.0, Customer);
    break;
case 6:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    if (request(Server3,Customer,0) == 0)
        schedule(7,expntl(Ts3),Customer);
    break;
case 7:
    release(Server3, Customer);
    schedule(2, 0.0, Customer);
    break;
case 8:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    if (request(Server4,Customer,0) == 0)
        schedule(9,expntl(Ts4),Customer);
    break;
case 9:
    release(Server4, Customer);
    schedule(2, 0.0, Customer);
    break;
case 10:
    stream(i);    /* Escolhe sequencia aleatoria 8 */
    if (request(Server5,Customer,0) == 0)
        schedule(11,expntl(Ts5),Customer);
    break;
case 11:
    release(Server5, Customer);
    schedule(2, 0.0, Customer);
    break;
}
}

/* gera o relatorio da simulacao */

```



```

report();
fclose(saida);
}
/* ----- Fim do Codigo PVM Puro -----
*/
/* -----
*/

```

A seguir é apresentado o código-fonte utilizado na segunda parte do estudo de caso, na qual o escalonamento leva em consideração a potência computacional das máquinas (como é sugerido no funcionamento do AMIGO).

```

/* -----
*/
/* ----- Codigo PVM com Ponderação -----
*/

#include <dir.h>
#include <stdio.h>
#include <conio.h>
#include "smpl.h"
#include "rand.h"

main()
{
/* definicoes */
float Te = 100000000;
char Flag_Reset = 1;
int Event = 1, Customer = 1, Aleatorio, i=0;
real Ts1 = 1.53686, Ts3 = 31.32981,
    Ts4 = 82.705, Ts2 = 135.7939,
    Ts5 = 136.6733;

int Server1, Server2, Server3, Server4, Server5;
FILE *p, *saida;
saida = fopen("pvm2.out","w");
if ((p = sendto(saida)) == NULL)
    printf("Erro na saida\n");

/* prepara o sistema de simulacao e dah nome ao modelo */
smpl(0," pvm-v1");

```

```

/* cria e dah nome aas facilidades (recursos) */
Server1 = facility("pvm",1);
Server2 = facility("lasdix",1);
, Server3 = facility("lasdpc08",1);
Server4 = facility("lasdpc10",1);
Server5 = facility("lasdpc11",1);

/* escalona a chegada dos clientes */
for (i=0; i<=100; i++){
    schedule(1,0, Customer);}

while ( (time() < Te) ) /* Estabelece critério de parada da simulação */
{
    if ((time() >= 10000000) && (Flag_Reset)) /* Descarta Warmup */
    {
        reset ();
        Flag_Reset = 0;
    }
    cause(&Event,&Customer);
    switch(Event)
    {
        case 1:
            stream(i); /* Escolhe sequencia aleatoria 0 */
            schedule(2,0.0,Customer);
            break;
        case 2:
            stream(i); /* Escolhe sequencia aleatoria 0 */
            if (request(Server1,Customer,0) == 0)
                schedule(3,expntl(Ts1),Customer);
            break;
        case 3:
            release(Server1, Customer);
            Aleatorio = random(1,10000); /* Escolha ponderada das maquinas */
            if ( (0 <= Aleatorio) && (Aleatorio <= 5400) ) /*lasdpc08*/
                schedule(6, 0.0, Customer);
            if ( (5401 <= Aleatorio) && (Aleatorio <= 7500) ) /*lasdpc10*/
                schedule(8, 0.0, Customer);
            if ( (7501 <= Aleatorio) && (Aleatorio <= 8700) ) /*lasdpc11*/
                schedule(10, 0.0, Customer);
            if ( (8701 <= Aleatorio) && (Aleatorio <= 10000) ) /*lasdix*/

```

```
    schedule(4, 0.0, Customer);
    break;
case 4:
    stream(i);    /* Escolhe sequencia aleatoria 0 */
    if (request(Server2, Customer, 0) == 0)
        schedule(5, expntl(Ts2), Customer);
    break;
case 5:
    release(Server2, Customer);
    schedule(2, 0.0, Customer);
    break;
case 6:
    stream(i);    /* Escolhe sequencia aleatoria 0 */
    if (request(Server3, Customer, 0) == 0)
        schedule(7, expntl(Ts3), Customer);
    break;
case 7:
    release(Server3, Customer);
    schedule(2, 0.0, Customer);
    break;
case 8:
    stream(i);    /* Escolhe sequencia aleatoria 0 */
    if (request(Server4, Customer, 0) == 0)
        schedule(9, expntl(Ts4), Customer);
    break;
case 9:
    release(Server4, Customer);
    schedule(2, 0.0, Customer);
    break;
case 10:
    stream(i);    /* Escolhe sequencia aleatoria 0 */
    if (request(Server5, Customer, 0) == 0)
        schedule(11, expntl(Ts5), Customer);
    break;
case 11:
    release(Server5, Customer);
    schedule(2, 0.0, Customer);
    break;
}
}

/* gera o relatorio da simulacao */
```

```

report ();
fclose(saida);
}
/* ----- Fim do Codigo PVM com Ponderação -----
*/
/* -----
*/

```

◆ Cálculo do Intervalo de Confiança

Um estimador pontual com base em uma amostra, como a média ou a variância, produz um único número como parâmetro. Entretanto, pelo grau de incerteza que envolve as variáveis aleatórias, muitas vezes deve-se considerar, além do estimador, a precisão com que se estima esse parâmetro. A forma usual de obter-se essa precisão é através dos intervalos de confiança.

A questão envolvida no cálculo de um intervalo de confiança é quão próximo o valor do estimador pontual da amostra está em relação ao valor real, quando tomada a população. Via de regra, o estimador envolvido é a média da amostra:

$$\bar{Y} = \sum_{i=1}^N Y_i / N$$

Define-se $1 - \alpha$ como a probabilidade do valor absoluto da diferença entre a média amostral e a média da população (μ) seja igual ou menor que H :

$$P[|\bar{Y} - \mu| \leq H] = 1 - \alpha$$

Assim, o intervalo de confiança para a média amostral é definido como:

$$P[\bar{Y} - H \leq \mu \leq \bar{Y} + H] = 1 - \alpha$$

O intervalo $\bar{Y} - H$ a $\bar{Y} + H$ é chamado de intervalo de confiança de Y . H , que representa a metade do intervalo de confiança, é uma função de variáveis aleatórias, sendo também uma variável aleatória. Dessa forma, o intervalo $\bar{Y} \pm H$ é um intervalo aleatório.

No estudo de caso, Y_1, Y_2, \dots, Y_n são variáveis aleatórias independentes de uma distribuição com média μ , onde H é calculado por (MacDougall, 1987):

$$H = t_{\alpha/2; N-1} s,$$

H obedece à distribuição t de *Student*, com um grau de liberdade $n-1$ ¹, sendo n o tamanho da amostra. A forma da distribuição t é semelhante à normal. É simétrica em relação a 0 (zero), mas apresenta caudas mais “grossas”, ou seja, maior variância do que a normal. À medida que n aumenta, a distribuição t tende à normal². A Figura B.1, extraída de (Soares et al., 1991), apresenta a relação entre as curvas de t e normal.

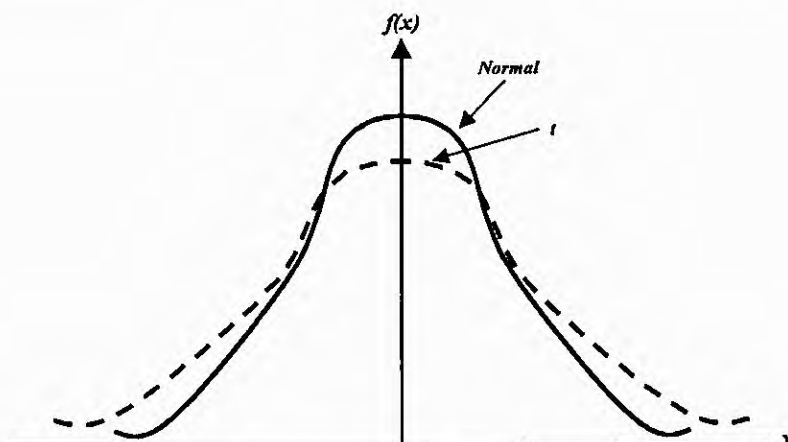


FIGURA B.1. Relação entre Distribuições Normal e t .

Para cada valor de $n-1$ (grau de liberdade), tem-se uma distribuição diferente. Segundo J. F. Soares, em (Soares et al., 1991), uma amostra é considerada pequena se seu tamanho não excede 30 elementos³. A amostra apresentada no estudo de caso possui 15 elementos, cada um correspondendo a uma execução da simulação com uma seqüência diferente proporcionada pelo *simpl*. A planilha B.1 apresenta os valores obtidos para os intervalos de confiança das utilizações para cem milhões de unidades de simulação, com todas as seqüências aleatórias do *simpl*.

¹ O grau de liberdade $N-1$ estabelece a forma da curva da distribuição.

² Teorema do Limite Central, discutido em detalhes em (Dantas, 1997).

³ Há autores que citam 25 elementos, tal qual (MacDougall, 1987).

Cálculo do Intervalo de Confiança para 100 milhões de unidades de simulação, utilizando-se ponderação

100 milhões de unidades de simulação, utilizando-se ponderação na escolha das máquinas															E[U]	DP[U]	NS	IC	
	seq0	seq1	seq2	seq3	seq4	seq5	seq6	seq7	seq8	seq9	seq10	seq11	seq12	seq13	seq14				
U[pvm]	0,0865	0,0866	0,0882	0,0863	0,0463	0,0863	0,0863	0,0864	0,0862	0,0864	0,0862	0,0862	0,0862	0,0863	0,0863	0,0836	0,010332326	0,05	0,00522877
U[lasdlx]	0,9931	0,9906	0,9902	0,9909	0,1975	0,9908	0,9900	0,9920	0,9898	0,9905	0,9893	0,9907	0,9907	0,9920	0,9891	0,9378	0,204804185	0,05	0,10364316
U[lasdpc08]	0,9512	0,9520	0,9497	0,9504	0,9998	0,9508	0,9497	0,9505	0,9504	0,9510	0,9502	0,9484	0,9484	0,9493	0,9516	0,9536	0,01283404	0,05	0,00649479
U[lasdpc10]	0,9774	0,9776	0,9767	0,9746	0,9406	0,9772	0,9766	0,9749	0,9758	0,9755	0,9765	0,9760	0,9760	0,9754	0,9766	0,9738	0,009233283	0,05	0,00467259
U[lasdpc11]	0,9206	0,9211	0,9196	0,9173	0,9606	0,9204	0,9216	0,9210	0,9175	0,9227	0,9195	0,9199	0,9199	0,9196	0,9227	0,9229	0,010533258	0,05	0,00533046

Cálculo do Intervalo de Confiança para 100 milhões de unidades de simulação, sem ponderação

100 milhões de unidades de simulação, considerando-se as máquinas equiprováveis															E[U]	DP[U]	NS	IC	
	seq0	seq1	seq2	seq3	seq4	seq5	seq6	seq7	seq8	seq9	seq10	seq11	seq12	seq13	seq14				
U[pvm]	0,0447	0,0447	0,0447	0,0447	0,0447	0,0446	0,0447	0,0447	0,0446	0,0447	0,0447	0,0446	0,0447	0,0448	0,0446	0,0447	0,000056061	0,05	0,00002837
U[lasdlx]	0,9880	0,9880	0,9866	0,9870	0,9882	0,9866	0,9865	0,9883	0,9873	0,9865	0,9886	0,9876	0,9861	0,9880	0,9882	0,9874	0,000806108	0,05	0,00040794
U[lasdpc08]	0,2285	0,2285	0,2284	0,2282	0,2285	0,2284	0,2283	0,2285	0,2286	0,2285	0,2284	0,2284	0,2286	0,2285	0,2284	0,2284	0,00010601	0,05	0,00005365
U[lasdpc10]	0,6010	0,6010	0,6004	0,6020	0,6014	0,6001	0,6018	0,6006	0,6004	0,6024	0,6012	0,6006	0,6027	0,6013	0,6009	0,6012	0,000761452	0,05	0,00038534
U[lasdpc11]	0,9939	0,9939	0,9946	0,9940	0,9943	0,9950	0,9942	0,9939	0,9945	0,9943	0,9935	0,9943	0,9945	0,9935	0,9940	0,9942	0,000425049	0,05	0,00021510

Onde E[U] e DP[U] são, respectivamente, a média e o desvio padrão da variável aleatória U - Utilização.

NS é o nível de significância e IC é o intervalo de confiança

Apêndice C

Métodos de Solução Analítica

Este apêndice apresenta sinopses de alguns métodos analíticos para solução de modelos de sistemas de filas. O conjunto de métodos exposto a seguir está baseado na compilação apresentada em (Silva, 2000), cujo trabalho expõe de forma condensada os principais métodos apresentados em (Jain, 1991), (Allen, 1990) e (Lazowska, 1984).

A classificação leva em consideração a característica de o modelo ser aberto ou fechado, conforme discutido no Capítulo 5. Assim, a seguir são apresentados os métodos BCMP e limites assintóticos (aplicáveis a modelos abertos e fechados) e o processo de nascimento-e-morte (para modelos abertos).

Ao final deste apêndice, é mostrado o código-fonte que implementa o método AVM, utilizado no estudo de caso do Capítulo 6.

• Redes BCMP

O método BCMP, junção das iniciais de *Basktt*, *Chandy*, *Muntz* e *Palacios*, generalizara a rede de *Jackson* para que seja aplicável a:

- modelos abertos, fechados e mistos.
- distribuições de tempos de serviço diferentes da exponencial;
- diferentes classes de clientes, cada uma com diferentes tempos de serviço, permitindo ainda, que clientes mudem de classe após completado o serviço no centro, antes de ir para outro centro de serviço;
- modelos que são abertos para determinadas classes de clientes e fechados para outras.

Para isso, *Basktt*, *Chandy*, *Muntz* e *Palacios* mostrarem que soluções de forma-produto¹ existem para uma classe abrangente de modelos. Essa classe consiste em modelos que satisfazem os seguintes critérios (Allen, 1990):

- Disciplinas de Fila: todos os centros de serviço possuem uma das seguintes disciplinas de fila: *First Come First Served* (FCFS), também referenciada como *First In Firsts Out* (FIFO); *Processor-Sharing* (PS),

Servidores Infinitos (IS), também chamados de *Centros Delay*; e *Last Come First Served - Preemptive* (LCFS-P), também conhecida como *Last In Firsts Out* (LIFO).

- Classes de Clientes: os clientes permanecem em uma única classe enquanto estão esperando ou recebendo serviço no centro de serviço, mas podem mudar de classe e de centro de acordo com as probabilidades de saída do centro.
- Distribuições de Tempo de Serviço: em centros de serviço com disciplinas de fila FCFS, as distribuições de tempo de serviço devem ser idênticas e exponenciais para todas as classes de clientes. Em centros de serviço com outra disciplina de fila, diferentes classes de clientes podem possuir distribuições diferentes.
- Serviço Dependente de Carga: em um centro de serviço com disciplina de fila FCFS, o tempo de serviço pode depender apenas do total de número de clientes do centro. Em centros de serviço com disciplinas de fila PS, LCFS-P e SI, o tempo de serviço para uma classe de clientes pode depender do número de clientes no centro pertencentes a essa classe, mas não sofre influência do número de clientes no centro pertencentes a outra classe. Além disso, a taxa de serviço global de um modelo pode depender do número total de clientes nele existente.
- Chegada de Clientes: para um modelo aberto, o tempo entre chegadas de uma classe deve ser exponencialmente distribuído. As taxas de chegada podem ser dependente de carga, de forma que, para modelos abertos existem dois possíveis padrões de chegada *Poisson* (exponencial). No primeiro, a taxa de chegada do lado de fora do modelo, $\lambda(N)$, tem uma taxa média que depende do número de clientes, N , no sistema. No segundo padrão, existe um fluxo de chegada *Poisson* para cada classe de cliente, onde a taxa de chegada da i -ésima classe depende apenas do número de clientes presentes nessa classe.

Os modelos que satisfazem a esses critérios são conhecidos como redes BCMP, que são geralmente solucionados pelo método de Análise do Valor Médio, discutido no Capítulo 6, obedecendo às restrições do AVM.

¹Em uma série com M filas de servidores únicos, com tempos de serviço exponenciais e chegadas *Poisson*, cada fila pode ser analisada independente das demais (Jain, 1991).

• Limites de Desempenho

Este método, que pode ser aplicado a modelos abertos e fechados, consiste em calcular limites inferiores e superiores do *throughput* do sistema e do tempo de resposta, como funções do número ou da taxa de chegada dos clientes. Na prática, a grande atração desse método é a simplicidade de suas técnicas. São duas as técnicas para o cálculo de limites: *Limites Assintóticos* e *Limites de Sistema Balanceado* (Lazowska, 1984).

O cálculo de limites de desempenho, além de relativamente simples, é um método rápido, podendo até ser feito manualmente. Sua aplicação provê uma valiosa introspeção a respeito de fatores que afetam o desempenho do sistema, como "gargalo(s)" do sistema.

As técnicas de cálculo de limites podem ser usadas para eliminar alternativas inadequadas, fornecendo, por meio de um simples cálculo, informações valiosas. E. Lazowska, em (Lazowska et al., 1984), aponta uma aplicação sobre dimensionamento do sistema, que envolvem considerações de um grande número de possíveis configurações. No caso de dimensionamento do sistema, geralmente um único recurso é o interesse dominante e o resto do sistema é configurado em função desse recurso.

O termo *limites otimistas* é utilizado para referenciar o limite superior do *throughput* e o limite inferior do tempo de resposta, que indicam a situação de melhor desempenho. A situação contrária é referenciada como *limites pessimistas*. A técnica de cálculo de Limites Assintóticos, por abranger uma classe maior de sistemas, via de regra, é mais referenciada do que a técnica de cálculo de Limites de Sistemas Balanceados. Seus limites otimistas e pessimistas são derivados de condições extremas de carga: alta e baixa. A validade dos limites é garantida pelo fato de que a demanda de serviço independe da quantidade e da localização de clientes no sistema, ou seja, não importa a quantidade de clientes e em que centros de serviço os mesmos estão locados (Lazowska, 1984).

Assim, a demanda de serviço de cada centro k do modelo, denominada D_k , é o total de tempo de serviço que um cliente requer nesse centro. A demanda D_k pode ser calculada através do produto de V_k (número de visitas que um cliente faz ao centro k) e S_k (tempo de serviço por visita). Para modelo abertos, o valor de D_k pode ser também calculado dividindo-se ρ_k (utilização do centro k) por λ (taxa de chegada dos clientes). Assim, denota-se a demanda total de serviço de um cliente

em todos os centros por $D = \sum_M^I D_k$, onde M é quantidade de centros de serviço do modelo (Lazowska,1984).

• **Processo Nascimento-e-Morte**

Um processo nascimento-e-morte, aplicável a modelos abertos, é um processo markoviano² onde as transações estão restritas aos estados vizinhos.

No processo nascimento-e-morte, o estado do modelo é determinado pelo número de clientes no sistema (n). Assim, quando o sistema está no estado n , estão nele n clientes. A chegada de um novo cliente, representada por λ , muda o estado do sistema para $n+1$, sendo chamado de *nascimento*. Da mesma forma, a saída de um cliente, representada por μ , muda o estado para $n-1$, sendo chamado de *morte*. O diagrama de estados do processo nascimento-e-morte é mostrado na Figura C.1.

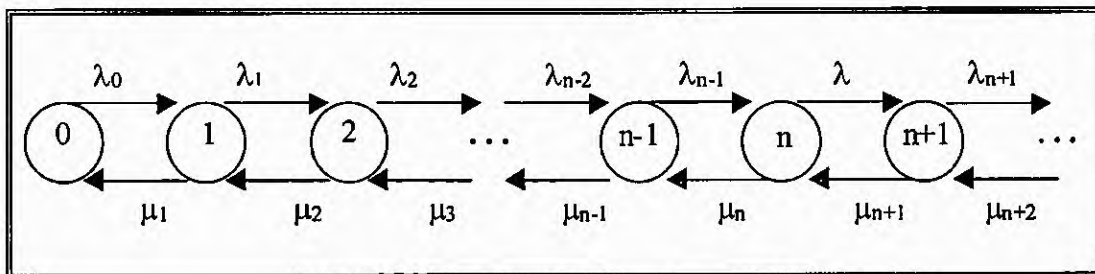


FIGURA C.1 – Diagrama de Estado do Processo Nascimento-e-Morte

A Figura C.2 ilustra o diagrama de estado do processo nascimento-e-morte para um modelo M/M/1, onde as taxas de entrada (λ) e de saída (μ) são constantes.

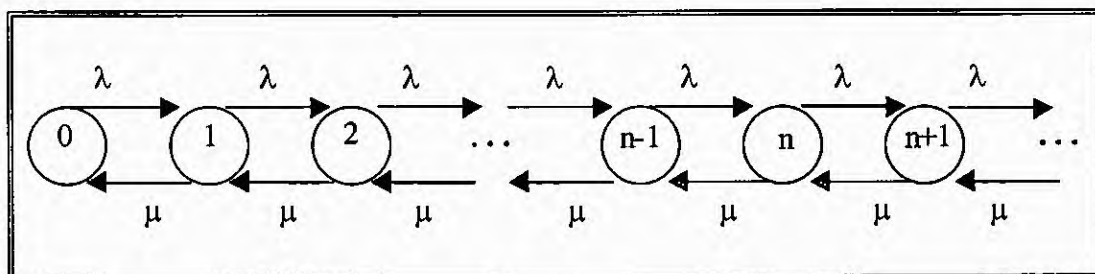


FIGURA C.2 - Diagrama do Processo de Nascimento-e-Morte para o Modelo M/M/1

² processos markoviano (discutidos com mais detalhes no Capítulo 3) são processos estocásticos em que seus estados futuros são independentes do passado e dependem apenas do presente.

Considerando-se um processo de nascimento-e-morte, têm-se as seguintes equações para o caso do modelo M/M/1, quando $\rho < 1$ (Jain,1991):

- Intensidade do tráfego = Utilização $\rho = \frac{\lambda}{\mu}$
- Número médio de clientes no sistema ----- $E[n] = \frac{\rho}{(1-\rho)}$
- Número médio de clientes na fila ----- $E[n_q] = \frac{\rho^2}{(1-\rho)}$
- Tempo médio de resposta ----- $E[r] = \frac{1/\mu}{(1-\rho)}$
- Tempo médio de fila ----- $E[w] = \rho \frac{1/\mu}{(1-\rho)}$

• **Código-Fonte do Método AVM**

Esta seção apresenta uma possível implementação em linguagem C do algoritmo do método AVM, discutido na seção 6.3.3 do Capítulo 6.

```
----- Inicio AVM -----
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main(void)
{

int N = 400,M = 5, i, n;

float S[5], V[5], X[5], Q[5], R[5], U[5];
float XTput = 0,SomaR;

FILE *fp;

fp = fopen("arq1.txt","w+");

S[0]=1.53686;
S[1]=31.32981;
S[2]=82.705;
S[3]=135.7939;
S[4]=136.6733;

V[0]=400;
V[1]=100;
V[2]=100;
V[3]=100;
V[4]=100;

for(i=0; i<M; i++)
Q[i]=0.0;

for(n=0; n<N; n++)
```

```

{
fprintf(fp, "\n Iteração %d\n", n);
for(i=0; i<M; i++)
{
R[i] = S[i]*(1+Q[i]);
fprintf(fp, " R[%d]=> %6.6f\n ", i, R[i]);
}
SomaR=0.0;

for(i=0; i<M; i++)
SomaR = SomaR + (R[i]*V[i]);

fprintf(fp, "\nSoma de R[%d]: %f\n\n", n, SomaR);

XTput = N/SomaR;

for(i=0; i<M; i++)
{
Q[i] = XTput*V[i]*R[i];
fprintf(fp, " Q[%d]=> %6.6f\n ", i, Q[i]);
}
}
fprintf(fp, "\n Throughput do sistema eh %6.6f\n ", XTput);
fprintf(fp, "\n Tempo de Resposta do sistema %6.6f\n\n", SomaR);

for(i=0; i<M; i++)
{
X[i] = XTput*V[i];
fprintf(fp, " X[%d]=> %6.6f\n ", i, X[i]);
U[i] = XTput*S[i]*V[i];
fprintf(fp, " U[%d]=> %6.6f\n ", i, U[i]);
}
fclose(fp);
return 0;
}
----- Fim AVM -----

```

Glossário

Neste Glossário, estão relacionados alguns termos, cujo significado poderia suscitar dúvidas. Esses termos são listados a seguir com a semântica assumida neste trabalho.

- Escalonador de Processos: é um software responsável por decidir para quais máquinas os processos devem ser encaminhados, a fim de serem atendidos.
- Estados com *Delay*: neste trabalho, quando um estado que possui um *delay* (retardo) é alcançado, há um tempo de permanência nele, e só depois de decorrido esse tempo é que o sistema transiciona para outro estado.
- Modelagem: Nesta tese, o termo designa o processo de, a partir de uma especificação, prover a solução de um modelo e conseqüente obtenção de medidas de desempenho. O termo também é usado com outras conotações dentro da computação, tal qual “modelagem de dados”, na engenharia de software.
- Queuing Statecharts: extensão para descrever sistemas de filas que apresenta uma aglutinação entre a especificação Statecharts e a representação de redes de filas.
- Simulação: é um termo genérico, mas, em computação, a simulação se refere ao emprego de um processo computacional para implementar um modelo (geralmente estocástico) de um fenômeno ou sistema dinâmico (sistemas cujos estados mudam com o decorrer do tempo).
- Statecharts Estocásticos: extensão Statecharts (para descrever sistemas de filas) que mantém a notação original, definida por D. Harel, exceto pela adição de eventos condicionados a probabilidades e estados com retardo exponencialmente distribuído.
- *Template*: conjunto de estados de Statecharts (reunidos em um superestado) que representa o funcionamento básico de um componente de um sistema de filas.
- Teste de Aderência: teste realizado em uma determinada amostra, para determinar se ela faz parte de uma certa distribuição. Alguns desses testes são: o qui-quadrado, o Kolmogorov-Smirnov (K-S) e o Anderson-Darling (A-D).