Pós-Graduação em Ciência da Computação

"JMSCAPACITY – UM TOOLKIT PARA AUXILIAR NO PLANEJAMENTO DE CAPACIDADE DE MIDDLEWARE ORIENTADO A MENSAGEM"

Por

ROBERTO DELGADO ARTEIRO

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, JULHO/2009

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ROBERTO DELGADO ARTEIRO

"JMSCAPACITY – UM *TOOLKIT* PARA AUXILIAR NO PLANEJAMENTO DE CAPACIDADE DE *MIDDLEWARE* ORIENTADO A MENSAGEM"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: NELSON SOUTO ROSA
CO-ORIENTADOR: PAULO ROMERO MARTINS MACIEL

RECIFE, JULHO/2009

Arteiro, Roberto Delgado
JMSCapacity- um toolkit para auxiliar no planejamento de capacidade de middleware orientado a mensagem / Roberto Delgado Arteiro. - Recife : O autor, 2009.
xxiv, 233 páginas : il., fig., tab.

Dissertação (mestrado) - Universidade Federal de Pernambuco. CIN. Ciência da Computação, 2009.

Inclui bibliografia, apêndice e golssário.

1. Sistemas distribuídos. 2.Avaliação de desempenho. 3. Petri, Redes de. 4. Integração de aplicações corporativas (sistemas de computação). I. Título.

004.24 CDD (22.ed.) MEI-2010-019

Agradecimentos

Primeiramente a Deus pela energia e luz iluminando meus caminhos.

Em especial a minha esposa, Iveruska, por nunca ter desistido de me incentivar, empurrar e também me ajudar durante esta dissertação. Tantas vezes eu pensei em desistir, mas ela me fazia continuar. Meu amor, meu profundo e eterno agradecimento, esse resultado também pertence a você.

A meu filho, Pedro, que tantas vezes precisou de mim e eu não podia estar com ele. Pelas tantas vezes que me perguntou: - Papai, já terminou o Capítulo 5? E o Capítulo 7?

Aos meus pais, José Fernandes e Odete Maria, pela incansável luta de formar e educar seis filhos. Agradeço também pelo exemplo de vida que meu pai sempre meu deu, demonstrando como poucos que sucesso não vem sem muito trabalho. A minha mãe o meu eterno carinho.

A minha sogrinha, Ivânova e meu sogrão Roosevelt, que se tornaram meus pais também, por todo o incentivo, puxões de orelha e o grande suporte durante essa jornada longa, mas gratificante. Vocês foram fundamentais para a conclusão deste trabalho, o meu eterno agradecimento.

Aos meus amigos Ricardo Massa e Leopoldo França que acreditaram em mim em um dos momentos mais difícies da minha vida, e me proporcionaram as primeiras oportunidades na área de docência, me encaminhando profissionalmente.

Ao meu amigo Fábio Souza que durante seu mestrado me proporcinou o aprendizado necessário para a concretização deste trabalho. Com certeza uma parte dos conhecimentos aqui depositados são oriundos das infindáveis reuniões e discussões que travamos nos anos de seu mestrado.

Aos meus professores pelos ensinamentos em sala de aula, pela troca de experiências e principalmente pelos incentivos.

Aos meus orientadores, Nelson Rosa e Paulo Maciel, pelo aprendizado obtido, pelos direcionamentos dos trabalhos, e sobretudo por terem me ajudado nesta jornada.

Resumo

Um dos maiores desafios atuais das organizações é a integração das aplicações corporativas permitindo o adequado gerenciamento dos processos de negócio. Atualmente, sistemas de middleware orientados a mensagem (MOMs) têm sido amplamente utilizados como ferramentas para realizar esta integração. Diante desse contexto, o seu planejamento e gerenciamento de capacidade passa a ter relevância significativa para o sucesso da integração, uma vez que, se o MOM falhar ou ficar indisponível, toda a interação entre as aplicações será comprometida.

Este trabalho propõe o JMSCapacity, um *toolkit* formado por um processo, uma biblioteca de componentes e ferramentas para auxiliar no planejamento e gerenciamento de capacidade de MOMs baseados na especificação *Java Message Service* (JMS).

O processo define como utilizar a biblioteca e as ferramentas para realizar o planejamento de capacidade sendo a principal contribuição deste trabalho. No processo proposto, adotou-se uma abordagem baseada em modelos para permitir que especificações de alto nível da integração sejam mapeadas em modelos formais em redes de Petri estocásticas. Isso viabiliza a realização de predições de desempenho em múltiplos cenários de carga, facilitando a identificação do ponto de saturação do MOM e da capacidade máxima de entrega de mensagens.

Palavras-chave: Middleware, Integração de Aplicações, Avaliação de Desempenho, Redes de Petri

Abstract

Enterprise application integration is one of the greatest challenges faced by modern corporations. To design and develop loose-coupled and robust enterprise integration solutions, those organizations are using message-oriented middleware systems (MOMs). Considering its critical role in the integration process, capacity planning of MOMs has gained its momentum.

This work proposes JMSCapacity, a toolkit composed by a process, a component library and tools to aid in capacity planning for MOMs based in the Java Message Service (JMS) specification.

The process is considered a central contribution of this work and proposes a model-based approach that allows mapping of high level integration specifications into stochastic Petri nets models. These models enable the accomplishment of performance predictions in multiple workload scenarios allowing the identification of the MOM´s saturation point and its maximum delivery capacity.

Keywords: Middleware, Application Integration, Performance Evaluation, Petri nets.

Índice Analítico

Lista de Figuras

Lista de Tabelas

Glossário

ACK	*Acknowledge. Mensagem de Confirmação de Recebiment.*
Administered Objects	*Objetos JMS criados pelo administrador, pré-configurados e armazendos em um serviço de diretório ou serviço de nomes.*
API	*Application Programming Interfae. Interface de Programação de Aplicativos.*
ATM	*Automatic Teller Machine. Terminal Bancário de Auto-Atendimento.*
BACEN	*Banco Central do Brasil.*
Batch	*Arquivo de lote. Aplicações batch representam aplicações não interativas.*
Benchmark	*Ato de executar um programa de computador, um conjunto de programas ou outras operações, a fim de avaliar a performance relativa de um objeto, normalmente executando uma série de testes padrões e ensaios nele.*
BI	*Business Inteligence. Inteligência de negócios, refere-se ao processo de coleta, organização, análise, compartilhamento e monitoramento de informações que oferecem suporte à gestão de negócios.*
BPEL4WS	*Business Process Execution Language for Web Services. Linguagem de execução para sistemas BPM.*
BPM	*Business Process Management. Gestão de Processos de Negócio é um conceito que une*

	gestão de negócios e tecnologia da informação com foco na otimização dos resultados das organizações através da melhoria dos processos de negócio.
Canal de Comunicação	*Abstração do meio (fila ou tópico) utilizado para transportar uma mensagem do emissor ao receptor.*
CBMG	*Customer Behavior Model Graph. Grafo representando o comportamento do consumidor.*
CIO	*Chief Information Officers. Principal executivo de TI de uma organização.*
Commit	*Comando que efetiva a transação em execução.*
Consumer	*idem Consumidor.*
Consumidor	*Entidade receptora de mensagens.*
CRM	*Customer Relationship Management. Gestão de Relacionamento com o Cliente.*
CTMC	*Continuous-Time Markov Chain. Cadeia de Markov de Tempo Contínuo.*
Data Centers	*Centros de Processamento de Dados.*
Deadlock	*Caracteriza uma situação em que ocorre um impasse pela alocação de um mesmo recurso computacional.*
DEBS	*Distributed Event-Based Systems. Sistemas baseados em eventos distribuídos.*
Destination	*Canal de Comunicação de um MOM JMS.*
DoS	*Deny of Service. Recusa de Serviço.*
DS	*Demanda de Serviço.*
DSPN	*Determinisc Stochastic Petri Nets. Tipo estocástico de redes de Petri.*
DTMC	*Discrete-Time Markov Chain. Cadeias de Markov de tempo discreto.*
EAI	*Enterprise Application Integration. Princípios de arquitetura de sistemas utilizados no processo de Integração de Aplicações Corporativas.*

E-commerce	*Comércio Eletrônico.*
EIS	*Enterprise Information System. Sistemas de Informações Executivas.*
ERP	*Enterprise Resource Planning. Sistemas Integrados de Gestão Empresarial.*
ESB	*Enterprise Service Bus. Plataforma tecnológica para integração de aplicações corporativas utilizada na abordagem SOA.*
Fila	*Tipo específico de Destination.*
GSPN	*Generalized and Stochastic Petri Nets. Tipo estocástico de redes de Petri.*
Idle Time	*Tempo em que o processo não está em uso.*
JCA	*Java EE Connector Architecture. Solução tecnológica baseada na linguagem de programação Java para conectar um servidor de aplicação a um sistema de informação corporativo.*
JEE	*Java Enterprise Edition. Framework de funcionalidades e padrões para o Java, destinados ao desenvolvimento de aplicações servidoras e corporativas.*
JMS	*Java Message Service. Padrão de implementação de um MOM definido pela Sun Microsystems para a Linguagem de Programação Java.*
JMS Clients	*Programa ou componente de aplicação que produz ou consome mensagens.*
JMS Messages	*Objetos que comunicam informações entre JMS Clients, e podem conter diferentes tipos de dados, desde os mais simples, como texto plano, até os mais complexos, como stream de bytes ou objetos serializados.*
JMS Provider	*Sistema de entrega de mensagens que implementa a interface especificada pela API JMS, além de funcionalidades como balanceamento de carga, clustering e serviço de persistência.*
JVM	*Java Virtual Machine. Interpretador de byte code Java, necessário para executar qualquer*

	programa neste linguagem.
LQN	*Layered Queuing Networks. Tipo de redes de fila.*
LRS	*Linear Reachability Set. Conjunto de alcançabilidade linear de um grafo.*
LTS	*Labeled Transition Systems. Formalismo gráfico baseado em estados e transições rotuladas para representação de sistemas.*
MOM	*Middleware Orientado a Mensagem.*
mps	*Mensagens por segundo.*
NPNT	*Non Persistent and Non Transaction. Modelo de confiabilidade (QoS) de uma comunicação sem persistência e sem transação.*
PCP	*Aplicações de planejamento e controle da produção.*
PDV	*Ponto de Venda.*
Performance Tuning	*Processo de ajuste de parâmetros do sistema para otimizar o seu desempenho.*
PNT	*Persistent and Non Transaction. Modelo de confiabilidade (QoS) de uma comunicação com persistência e sem transação.*
Producer	*idem Produtor.*
Produtor	*Entidade responsável por enviar mensagens.*
Provider	*idem JMS Provider.*
PT	*Persistent and Transactional. Modelo de confiabilidade (QoS) de uma comunicação com persistência e com transação.*
PTP	*Point to Point. Estilo de comunicação baseado na troca de mensagens ponto-a-ponto.*
Pub/Sub	*Publish and Subscribe. Paradigma de mensagens assíncronas, onde cada mensagem é enviada para todos os assinantes do tópico (Destination).*
QN	*Queueing Networks. Formalismo baseado em redes de fila.*
QoS	*Quality of Service. Qualidade de Serviço.*

QPN	*Queueing Petri Net. Formalismo que associa as redes de fila às redes de Petri.*
RFID	*Radio-Frequency Identification. Método de identificação automática através de sinais de rádio freqüência.*
RG	*Reachibility Graph. Grafo de alcançabilidade.*
Rollback	*Comando que descarta as mudanças da transçação em execução.*
rps	*Receives per second. Unidade de medida que representa a taxa de recuperação de mensagens de um consumidor síncrono.*
RRG	*Reduced Reachability Graph. Grafo de alcançabilidade reduzido.*
RS	*Reachibility Set. Conjunto de alcançabilidade de um grafo.*
RSFN	*Rede do Sistema Financeiro Nacional.*
RUP	*Rational Unified Process. Plataforma de processo de desenvolvimento de software.*
Ramp-Down	*Período de tempo final em um processo de medição, destinado a eliminar fatores transientes.*
Ramp-Up	*Período de tempo inicial em um processo de medição, destinado a eliminar fatores transientes.*
SCM	*Supply chain management. Sistemas de Informação para gerenciamento da cadeia de suprimentos.*
Sizing	*Dimensionamento.*
SLA	*Service Level Agreement. Parte de contrato de serviços entre duas ou mais entidades no qual o nível da prestação de serviço é definido formalmente.*
SLO	*Service Level Objectives.*
SOA	*Service Oriented Architecture. Estilo de arquitetura de software que preconiza que as funcionalidades implementadas pelas aplicações*

	devem ser disponibilizadas na forma de serviços freqüentemente organizados através de um ESB.
SOAP	*Simple Object Access Protocol, Protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída.*
SPB	*Sistema de Pagamento Brasileiro.*
SPN	*Stochastic Petri Nets. Redes de Petri estocásticas.*
Stock Trading	*Negociação de ações em bolsas de valores.*
SUT	*System Under Test. Sistema submetido a testes de desempenho.*
TCO	*Total Cost of Ownership. Custo Total de Propriedade.*
Thread	*Designa um processo de execução de código em linguagens de programação.*
Throughput	*Vazão.*
TI	*Tenologia da Informação.*
Token	*Elemento das redes de Petri que representa a marcação de um lugar.*
Tópico	*Tipo específico de Destination.*
Traces	*Conjuntos de registros (eventos) ordenados no tempo que foram submetidos ao sistema real.*
TTK	*Time-to-live. Tempo de vida*
W3C	*World Wide Web Consortium. Organização internacional responsável por padrões utilizados na internet.*
WS	*Web Services. Padrão tecnológico mantido pela W3C para implementação de serviços.*
WSDL	*Web Service Definition Language. Linguagem de definição para Web Services.*
XML	*eXtensible Markup Language. Linguagem de representação de dados baseada em tags. É considerada uma extensão do HTML.*

CAPÍTULO

1

Introdução

"Veni, vidi, vici"

Julius Caesar.

Durante muitos anos a área de tecnologia da informação (TI) teve um papel importante dentro das organizações, sendo uma de suas principais missões disponibilizar e tratar as informações de forma que elas pudessem auxiliar o processo de tomada de decisão. Entretanto, no contexto de governança corporativa, TI passa a desempenhar um papel estratégico: suportar e liderar a melhoria dos processos de negócios com o objetivo de garantir uma maior efetividade no cumprimento dos objetivos de negócio estabelecidos, além de uma maior transparência dos atos efetuados pela gestão da organização. Essa mudança de status passa fundamentalmente por um processo de integração flexível entre as aplicações corporativas, garantindo uma melhor qualidade e confiabilidade das informações à medida que os processos de negócio vão sendo modificados. Essas informações são utilizadas para avaliar sistematicamente indicadores de desempenho estabelecidos pela implantação da governança corporativa.

Este capítulo tem como objetivo introduzir como o planejamento e o gerenciamento da capacidade de sistemas de entrega de mensagens podem contribuir para o sucesso do processo de integração. Para isso, contextualiza-se a evolução do gerenciamento de TI dentro das organizações, levando a uma diversidade tecnológica de suas aplicações e, conseqüentemente, a uma necessidade de integração. Em seguida, o processo de integração é abordado, mostrando as principais soluções adotadas pelo mercado e as tendências nessa área. Na seqüência, são destacados alguns desafios da integração que caracterizam o problema alvo a ser resolvido por esta dissertação. Por fim, é

apresentada a proposta para o tratamento desses problemas e a estrutura do restante da dissertação.

1.1 Evolução de TI nas Organizações

Inicialmente, as organizações utilizavam TI para processamento intensivo de dados nos mais variados segmentos de atuação: acadêmico-científico, engenharia, militar, financeiro, industrial e comercial. Essa utilização acontecia através de aplicações *batch* que processavam dados com objetivos específicos. Um exemplo típico são as aplicações de folha de pagamento e contabilidade.

Com a evolução tecnológica e o surgimento dos sistemas interativos, as aplicações passaram a apoiar e suportar alguns processos de negócio, tais como: aplicações de faturamento, controle de estoque, compras, automação comercial (frente de loja), automação de força de venda, automação do atendimento ao cliente bancário (ATM), entre outras.

Mais recentemente, as empresas fizeram um grande esforço para adotar aplicações corporativas gerenciais, com intuito de melhorar o apoio aos seus processos de negócio. Essas aplicações normalmente são formadas por vários módulos integrados e possuem um enorme conjunto de processos customizáveis que exigem uma mudança na cultura interna, de forma a adaptar a empresa às aplicações, implicando em uma implantação lenta e custosa. Os exemplos mais comuns são as aplicações de gestão administrativa (ERP), planejamento e controle da produção (PCP), gerenciamento do relacionamento com o cliente (CRM). Nessa fase, o foco de TI passa a ser o de apoiar a tomada de decisão, principalmente no nível tático das organizações.

Por outro lado, a área estratégica demandava por informações consolidadas e integradas de todas as suas áreas de negócio. Diante dessa necessidade, iniciativas como os sistemas de informações executivas (EIS), e mais recentemente ferramentas de *Business Inteligence* (BI) dominaram o escopo das aplicações de TI. O problema principal dessas soluções voltadas para a área estratégica é que dependem de informações geradas e manipuladas pelas aplicações do nível gerencial (ERP, CRM, PCP), refletindo de forma direta a qualidade e confiabilidade dessas informações.

Outro aspecto importante é a crescente necessidade das empresas de formarem cadeias de valor, construindo parcerias estratégicas com o intuito de somar competências na busca frenética pela diminuição dos custos corporativos, além de uma maior eficiência e rentabilidade. Um exemplo típico são as aplicações de gerenciamento da cadeia de suprimentos (SCM). A dificuldade principal dessas aplicações é conectar as empresas formadoras da cadeia de valor de forma a manter a independência de cada uma, e ao mesmo tempo permitir processos flexíveis e mutáveis.

Além de tudo isso, a necessidade de implantar as melhores práticas em governança corporativa associada a um processo de globalização, que aumentou consideravelmente a competitividade entre as empresas, levou as organizações a buscarem uma maior eficiência em seus processos, sem, contudo, perder a capacidade de atingir resultados. O maior problema nessa busca é que a mudança contínua dos processos demanda alterações nos sistemas gerenciais, que por sua vez afetam a qualidade e confiabilidade das informações.

Como resultado desse contexto apresentado, as organizações utilizam atualmente um número crescente de aplicações construídas ou adquiridas em momentos diferentes, escritas por pessoas diferentes, utilizando diferentes linguagens, suportadas por plataformas de hardware distintas, usando sistemas operacionais diversos, além de estarem provendo diferentes funcionalidades. A verdade é que muitas dessas aplicações têm, freqüentemente, muito pouco em comum, resultando em funcionalidades isoladas, com múltiplas instâncias dos mesmos dados. Essas condições podem resultar em atividades redundantes, custos elevados e uma resposta ineficiente para seus clientes.

Essa diversidade tecnológica encontrada faz com que as aplicações da forma que estão atualmente em funcionamento reflitam com muito pouca fidelidade os processos de negócio da organização. O principal motivo dessa precariedade é a ausência ou a ineficiência do processo de integração entre essas aplicações, que produz uma área de TI burocrática e complexa. Se tudo isso não bastasse, a falta de integração ainda leva a uma consolidação de informações estratégicas de baixa qualidade, comprometendo a governança corporativa.

1.2 Integrando Aplicações Corporativas

O desafio atual das organizações é integrar as aplicações existentes para implementar os processos de negócio, alinhando-os aos objetivos das organizações que estão em constante evolução. Respondendo a essa necessidade surge o conceito de integração de aplicações, que segundo [Linthicum 2000] é o compartilhamento de processos e/ou dados, de forma segura e orquestrada, entre aplicações corporativas.

De acordo com [Gartner 2003], menos de 10% das aplicações corporativas possuia algum mecanismo de integração, e dessas, apenas 15% utilizavam uma plataforma de integração baseada em tecnologia de *middleware*. As demais 85% utilizavam processos de integração extremamente rudimentares e pouco flexíveis, como integração via exportação/importação de arquivos. Esse cenário levou os CIOs (*Chief Information Officers*) a definirem como uma de suas prioridades nesses últimos anos os projetos de integração.

Em geral, as soluções de integração precisam tratar alguns desafios fundamentais [Hohpe e Woolf 2004]:

- **Redes não são confiáveis**: integrar aplicações consiste em transportar dados de um computador para outro através de uma rede de computadores. Essa característica comum a todos os ambientes distribuídos faz com que a solução de integração precise tratar um enorme conjunto de possíveis problemas;
- **Redes são lentas**: enviar dados através da rede é muito mais lento que fazer uma chamada de método local dentro de única aplicação. Essa característica tem que ser levada em consideração para as soluções de integração, pois podem afetar muito o desempenho das aplicações;
- **Aplicações são normalmente diferentes**: soluções de integração precisam tratar a heterogeneidade entre as plataformas operacionais, as linguagens de programação e os formatos dos dados. Elas necessitam ser capazes de conectar diferentes tecnologias;

- **Mudanças são inevitáveis**: as aplicações estarão sempre em constante evolução, sendo a mudança nos requisitos uma das poucas certezas. Manter soluções de integração em meio a tantas mudanças é uma tarefa que exige muita flexibilidade, e principalmente pouca dependência entre as aplicações. Com isso, a solução de integração precisa conectar as aplicações, mantendo-as fracamente acopladas.

Apesar de existerem abordagens distintas para resolver esses desafios, um ponto em comum entre todas essas soluções é a utilização de mensagens como um dos elementos básicos da integração. A principal justificativa para isso é que através de um sistema de entrega e troca de mensagens, utilizando métodos confiáveis e assíncronos, a maioria desses desafios é contornada.

A baixa confiabilidade da rede pode ser resolvida através de mecanismos do tipo *store-and-forward*, onde a plataforma de integração armazena a mensagem antes de transmití-la para o próximo nó, sendo capaz de se recuperar após alguma falha. A lentidão e as mudanças podem ser atenuadas com o conceito de assincronia, uma vez que as aplicações não precisam tomar conhecimento, nem depender das demais aplicações para disponibilizar seus dados. A interligação de ambientes heterogêneos é uma característica inerente ao conceito de *middleware*.

Banavar [Banavar et al. 1999] introduziu o conceito de MOM (Middleware Orientado a Mensagem) como uma tecnologia que permitiria a criação de uma "cola" entre as aplicações dentro e fora das organizações, sem a necessidade de reengenharia dos componentes individuais. Desde então, os sistemas de MOM tornaram-se a solução mais adotada pelo mercado, representando ainda hoje a grande maioria das inciativas formais de integração utilizando uma plataforma de *middleware*.

Vários produtos que disponiblizam uma plataforma de MOM existem atualmente no mercado. Um conjunto signficativo deles adota uma especificação padrão denominada *Java Message Service* (JMS) [Sun 2002]. Essa especificação é aberta, perimtindo certo nível de interoperabilidade entre os produtos de MOM, o que a fez tornar-se um padrão "de fato".

No Brasil, um dos principais casos de sucesso na integração de aplicações corporativas é o Sistema de Pagamento Brasileiro (SPB), que utiliza como solução de integração a plataforma de MOM IBM WebSphere MQSeries [BACEN 2007]. O SPB permite que instituições financeiras do país efetuem transações *on-line* de forma eletrônica e automatizada. As aplicações dos bancos interagem de forma assíncrona e fracamente acoplada, através de um sistema de MOM.

Entretanto, os sistemas de MOM normalmente não endereçam todos os aspectos necessários para uma boa integração. A incapacidade de transformação de dados, de roteamento dinâmico baseado em regras são exemplos dessas limitações. Para resolver esses pontos foi criado o *Message Broker*, plataforma de troca de mensagens montada acima do MOM. Ele é capaz de interligar aplicações dentro e fora das organizações, através de roteamentos sofisticados e inteligentes. Além disso, são capazes de efetuar transformação nos dados trafegados nas mensagens, de forma a compatibilizar modelos de dados distintos entre aplicações. Mas essas soluções são proprietárias, tornam a

implementação dependente do produto utilizado para integração e elevam substancialmente o custo desse tipo de projeto. Esses aspectos dificultaram muito a adoção em larga escala dessas soluções.

Motivado pelo aumento significativo da importância que os projetos de integração passaram a ter dentro das áreas de TI nos últimos anos, algumas abordagens novas despontaram, sendo a principal delas a S*ervice Oriented Architecture* (SOA). Essa abordagem constrói uma arquitetura de integração baseada no conceito de serviço.

Um serviço é definido como uma função discreta qualquer que pode ser oferecida a um consumidor externo, podendo ser uma função de negócio individual ou uma coleção de funções que juntas formam um processo. As aplicações colaboram chamando os serviços uma das outras, enquanto que os serviços são compostos em uma seqüência para implementar os processos de negócio das organizações.

SOA especifica a necessidade da existência de um mecanismo consistente para que os serviços se comuniquem. Este mecanismo deve ser fracamente acoplado e suportar o uso de interfaces explícitas. SOA defende ainda a aplicação dos conceitos do desenvolvimento orientado a objetos, do projeto baseado em componentes e da tecnologia de *Enterprise Application Integration* (EAI) [Linthicum 2000].

Somando ao conceito de SOA, surge também o *Web Service* (WS), que é uma especificação padrão estabelecida pelo *World Wide Web Consortium* (W3C) para implementação de serviços, utilizando troca de mensagens XML, e contratos de serviço formalmente definidos utilizando uma linguagem de definiçao de interface chamada *Web Service Definition Language* (WSDL). É importante ressaltar que uma abordagem SOA não implica necessariamente na utilização de WS.

O surgimento do *Enterprise Service Bus* (ESB), ou Barramento de Serviços Corporativos [Chappell 2004] foi o resultado das experiências da indústria de TI na tentativa de construir um padrão para uma rede de integração utilizando a abordagem SOA implementada sob uma plataforma de MOM e utilizando mensagens XML. Esse barramento permite conectar de forma segura e coordenada um número significativo de aplicações corporativas com controle transacional.

As pesquisas de mercado [Gartner 2003] [Gartner 2004] [IDC 2003a] [IDC 2003b] apontam o ESB como a mais promissora plataforma tecnológica em projetos de integração. Um dos motivos é que ele está fortemente baseado em padrões de mercado: JMS, JCA, J2CA, SOAP, WSDL, BPEL4WS, XML.

Nessa nova abordagem as aplicações não precisam passar por reengenharias, apenas precisa-se desenvolver conectores de integração das aplicações ao barramento, normalmente implementados utilizando *Web Service*. Esses conectores trocam mensagens XML de forma assíncrona pelo barramento. A utilização do XML é motivada pela capacidade de interoperabilidade do padrão. Uma vez que a mensagem é recebida pelo barramento, esse possui toda a inteligência de roteamento e transformação de dados (normalmente encontrada nos *Message Brokers*) necessária para consolidar o processo de integração.

Para isso, a arquitetura do ESB possui em seu núcleo um MOM, que provê a infra-estrutura básica de uma rede de canais virtuais que o barramento utiliza para rotear mensagens dentro e fora das organizações. A utilização do MOM garante ao barramento a capacidade de desacoplar as aplicações, permitindo que elas conversem, sem necessariamente saberem uma da existência da outra. Essa característica assíncrona do MOM torna-se fundamental em um processo de integração, garantindo escalabilidade e confiabilidade no tratamento das mensagens.

Resumindo, o ESB combina o poder dos MOMs com o padrão na implementação de serviços provido pelos *Web Services*, com a inteligência dos *Message Brokers*, com a adoção de padrões de mercado. Com isso, torna-se a principal solução para implementação de SOA, e conseqüentemente para projetos de integração.

1.3 Desafios na Adoção de uma Plataforma de MOM

O MOM além de ser atualmente a principal solução em uso para integração de aplicações corporativas compõe o núcleo da arquitetura do ESB, a principal iniciativa do mercado para substituí-lo. Dessa forma, fica clara a relevância dessa classe de *middleware* nos projetos atuais e futuros da área de integração. Um fato que ajuda a corroborar essa afirmação é o lançamento recente do SPECjms2007 [SPEC 2007], um *benchmark* padrão projetado para avaliar o desempenho e a escalabilidade de plataformas de MOM compatíveis com o padrão JMS.

Diante desse contexto, onde a responsabilidade final de encaminhamento de todas as mensagens destinadas à integração das aplicações corporativas fica a cargo de um sistema de MOM, o planejamento e o gerenciamento da capacidade desse sistema passa a ter relevância significativa para o sucesso da integração, uma vez que se esse sistema falhar, ou ficar indisponível, toda a interação entre as aplicações ficará comprometida.

Menascé e Almeida [Menascé e Almeida 2002] definem planejamento de capacidade como o processo de prever e identificar possíveis pontos de saturação no ambiente, e definir um plano de adequação para continuar a prestar o serviço no nível especificado considerando o menor custo possível. Já o gerenciamento de capacidade é o processo de dimensionar (*sizing*) e monitorar os recursos necessários para que um sistema desempenhe suas atividades dentro dos níveis acordados.

Sendo assim, para realizar de forma adequada o planejamento e o gerenciamento de capacidade de sistemas de MOM um problema que precisa ser resolvido e que passa a ser o foco desta dissertação é:

Em uma integração de aplicações, como determinar a capacidade máxima de um MOM JMS, mantendo o nível de serviço acordado?

1.4 Proposta

A proposta desta dissertação é apresentar e demonstrar o uso do *JMSCapacity Toolkit* no auxílio do planejamento e gerenciamento de capacidade de sistemas de entrega de mensagens (MOM). O *toolkit* é formado por uma biblioteca de componentes (modelos de elementos básicos), ferramentas e um processo. O

processo define como utilizar a biblioteca para a construção de modelos de desempenho, que, posteriormente, são submetidos a técnicas de simulação para obtenção de informações de desempenho sobre o sistema real. As ferramentas auxiliam na validação do modelo, através de medições do sistema real.

O objetivo principal do *toolkit* é auxiliar o gerente de capacidade na obtenção do ponto de saturação do MOM – através do monitoramento da utilização dos seus principais recursos (fila, CPU) – e na obtenção da capacidade máxima de entrega de mensagens mantendo um nível de serviço acordado (ex. latência máxima na entrega das mensagens).

De fato essas não são as únicas informações necessárias para se realizar um planejamento e gerenciamento de capacidade de um MOM. Alguns outros recursos (ex. informações sobre o custo da solução) são imprescindíveis para tal atividade. O Capítulo 2 desta dissertação discute em mais detalhes outros aspectos que precisam ser determinados para completar o processo de planejamento de capacidade. Porém, é importante ressaltar que este trabalho não tem a pretensão de realizar o planejamento e gerenciamento completa da capacidade de um MOM, e sim oferecer um auxílio nessa atividade.

Em particular, os modelos de desempenho para o MOM e para a carga de trabalho são desenvolvidos como uma composição de componentes básicos (parte integrante da biblioteca de componentes). Os modelos de desempenho são especificados através do formalismo das *Generalized and Stochastic Petri Nets* (GSPN) [Marsan et al. 1984]. A escolha desse formalismo para a modelagem foi motivada pelos seguintes aspectos: natureza estocástica que possibilita a avaliação de desempenho; possibilidade de representação e visualização gráfica dos modelos; suporte a técnicas de avaliação tanto analíticas como de simulação; e amplo suporte de ferramentas.

1.5 Estrutura da Dissertação

Essa dissertação está estruturada em sete capítulos e quatro apêndices, sendo o primeiro capítulo esta introdução. O Capítulo 2 introduz todos os conceitos básicos necessários ao entendimento do que está sendo proposto na dissertação: *Middleware* Orientado a Mensagem (MOM), detalhando também os principais aspectos do padrão *Java Message Service* (JMS), e planejamento de capacidade, apresentando uma metodologia existente para realização dessa atividade.

Os Capítulos 3 e 4 apresentam o *toolkit* em detalhes. Inicialmente um processo para planejamento de capacidade é apresentado. No detalhamento desse processo são propostas métricas, parâmetros, fatores, entre outros aspectos úteis no planejamento de capacidade de um MOM. Em seguida, o Capítulo 4 apresenta a biblioteca de componentes e algumas ferramentas que podem ser utilizadas durante o processo.

O Capítulo 5 demonstra a utilização do *toolkit*, através de um guia passo a passo da realização da atividade de planejamento e gerenciamento de capacidade de um caso prático, contribuindo para validar e sedimentar o *toolkit* como um todo (processo, componentes e ferramentas). O Capítulo 6 apresenta um estudo detalhado do estado da arte em avaliação de desempenho de MOMs, além de analisar iniciativas para planejamento e gerenciamento de capacidade

de plataformas de *middleware*. O Capítulo 7 apresenta as conclusões e trabalhos futuros.

O apêndice A apresenta os conceitos básicos sobre avaliação de desempenho, enquanto que o Apêndice B introduz os conceitos sobre redes de Petri, destacando principalmente as GSPN. O Apêndice C apresenta a verificação e validação da biblioteca de componentes. Por último, o Apêndice D documenta a notação utilizada na definição do processo deste *toolkit*.

CAPÍTULO

2

Conceitos Básicos

"Se os fatos não se encaixam na teoria, modifique os fatos"

Albert Einstein.

Este capítulo introduz os conceitos básicos das áreas de conhecimento exploradas nesta dissertação. O objetivo é servir como referencial teórico para o leitor sobre os temas MOM e Planejamento de Capacidade. Para tanto, inicialmente são apresentados os principais aspectos que caracterizam as plataformas de MOM, fundamentando os conceitos de *middleware* e seus serviços. Em seguida, o *Java Message Service (JMS),* principal padrão para implementação de sistemas de MOM, é detalhado, ressaltando suas características, nomenclaturas, arquitetura e modelo de funcionamento. Na sequência, os conceitos sobre planejamento e gerenciamento de capacidade são discutidos, inciando pelos conceitos de modelagem de sistemas e finalizando com a apresentação de uma metodologia para planejamento de capacidade baseada na construção de modelos.

2.1 *Middleware* Orientado a Mensagem (MOM)

O mundo atual demanda uma complexidade de aplicações que força o desenvolvedor moderno a projetar sistemas para ambientes cada vez mais distribuídos, heterogêneos e complexos. Nesse contexto, surge o conceito de *middleware* como uma solução para facilitar a construção de sistemas distribuídos fornecendo uma camada de software que disponibiliza uma abstração de programação, assim como o mascaramento da heterogeneidade das redes, do hardware, de sistemas operacionais e linguagens de programação.

10 Middleware Orientado a Mensagem (MOM)

O conceito de *middleware* é bastante abrangente por conta desse termo ser usado em diversos contextos de desenvolvimento de software, ser aplicado a diversos tipos de programas, desenvolvidos em contextos diferentes e com propósitos diferentes. Uma das características comuns entre as diversas plataformas de *middleware* é a disponibilização de componentes que implementam serviços de comunicação, facilitando a distribuição das aplicações.

Campbel [Campbell et al. 1999] define *middleware* como uma classe de tecnologias de software projetadas para ajudar a gerenciar as complexidades inerentes a sistemas distribuídos. Nessa visão, ele fornece aos programadores uma abstração no processo de transmissão e recepção de informações entre os componentes do sistema distribuído [Emmerich 2000]. De acordo com [Bernstein 1996], um *middleware* provê serviços comuns de infra-estrutura de software necessários a uma grande quantidade de aplicações científicas e comerciais.

Essencialmente, um *middleware* é uma ferramenta que facilita a construção de sistemas distribuídos. Ele consegue isso escondendo as dificuldades inerentes ao desenvolvimento desses sistemas. Ele também se preocupa em abstrair dos programadores como os problemas de heterogeneidade são resolvidos, por exemplo, quando componentes são escritos em diferentes linguagens de programação e executando em diferentes plataformas operacionais. Além disso, várias falhas podem ser resolvidas automaticamente pelo *middleware* sem que nenhum trabalho seja feito pelo programador ou pela aplicação.

Na definição proposta por [Bernstein 1996], os serviços providos pelo *middleware* são de propósito geral e situados entre a plataforma (hardware, sistema operacional e pilha de protocolos de rede) e as aplicações. Eles são caracterizados por serem definidos por uma API, por suportarem múltiplos protocolos de comunicação, pela portabilidade e compatibilidade entre plataformas heterogêneas, por serem essencialmente distribuídos, e pela capacidade de integração com outros serviços.

Como existe uma grande diversidade de requisitos e de necessidades associados a processos de comunicação entre aplicações, os sistemas de *middleware* são classificados em categorias relacionadas ao tipo de primitiva fornecida para interação entre as aplicações: *middleware* procedural (chamada remota de procedimento), *middleware* orientado a mensagem (passagem de mensagem), *middleware* transacional (transação distribuída) e *middleware* orientado a objetos (invocação de método remoto) [Emmerich 2000].

Dentre essas categorias, os sistemas de *middleware* orientado a mensagem (MOM) são os mais amplamente utilizados como infra-estrutura de comunicação de aplicações corporativas, principalmente quando se trata de integração entre aplicações distintas ou entre componentes da mesma aplicação.

O *Java Message Service* (JMS) [Sun 2002] é um exemplo de uma especificação padrão para um serviço de entrega de mensagens, que define um conjunto de interfaces e requisitos para que aplicações desenvolvidas em *Java* possam trocar mensagens entre si.

2.1.1 Visão Geral de MOM

Middleware Orientado a Mensagem (MOM) [Banavar et al. 1999] é um conceito que envolve a passagem de dados entre aplicações utilizando um canal de comunicação que trafega unidades autônomas de informação chamadas de mensagens. Em um ambiente de comunicação baseado em MOM, mensagens são usualmente enviadas e recebidas de forma assíncrona e geralmente de forma ordenada. Por conta disso, as mensagens são processadas e os resultados destes processamentos disponibilizados para verificação futura [Emmerich 2000].

A Figura 2-1 apresenta duas aplicações ou dois componentes de uma mesma aplicação que se comunicam através de um MOM e assumem o papel de transmissor e receptor respectivamente. O transmissor e receptor são desacoplados, pois não precisam se conhecer previamente. Em vez disso, eles enviam e recebem mensagens de e para o MOM, sendo responsabilidade deste definir abstrações (canais) que façam com que as mensagens cheguem aos destinatários reais através de notificações proativas do sistema de mensagem, ou ainda pela ação direta do receptor.

Uma plataforma de MOM não fornece para aplicações ou componentes de uma mesma aplicação um mecanismo de acesso transparente ao serviço de comunicação, pois os transmissores e receptores precisam utilizar explicitamente um canal, através de primitivas de comunicação (ex. *send*, *receive*) disponibilizadas pelo MOM, para se comunicar com outros componentes.

A utilização do conceito de canal permite ainda que transmissores e receptores possam ser migrados de um computador para outro em tempo de execução. Essa característica habilita uma operação ininterrupta e transparente, uma vez que software e hardware podem ser alocados sem a necessidade de parada do ambiente. Outra prática importante que permite a migração da aplicação é a utilização consistente do serviço de nomes.

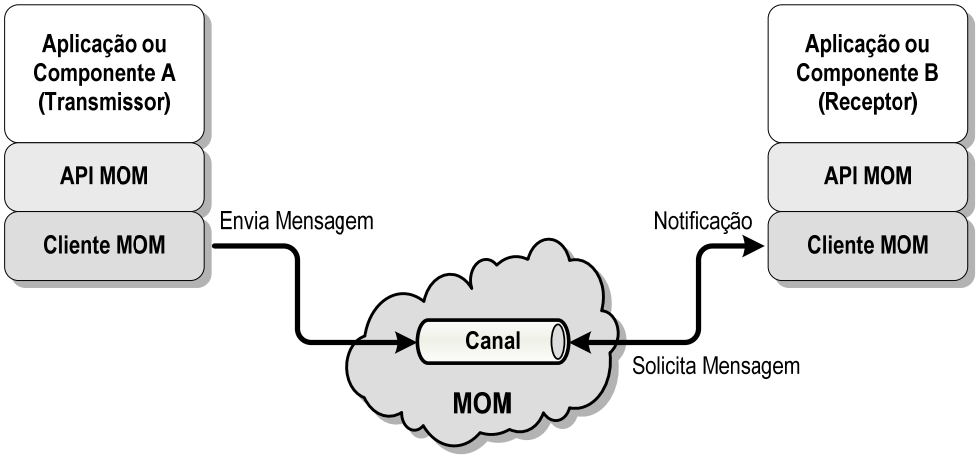


Figura 2-1. Comunicação baseada em uma plataforma de MOM.

Para facilitar a comunicação, o MOM disponibiliza uma *Application Programming Interface* (Interface de Programação de Aplicativos, ou API) e um cliente que são integrados aos transmissores e receptores. A aplicação utiliza a API para se comunicar com o cliente, sendo este responsável pelo envio e recebimento de mensagens através do MOM. Essa arquitetura comum

dos sistemas de mensagem deve permitir que a aplicação possa ser independente de uma implementação específica de MOM, sendo necessária apenas a compatibilidade das APIs.

As mensagens podem ser tratadas pelas aplicações como eventos, sendo esses ordenadamente tratados e processados. Além disso, as mensagens são armazenadas em filas de processamento dentro do canal, permitindo que elas possam ser priorizadas, descartadas caso estejam obsoletas, ou mesmo mantidas naquelas se houver necessidade [Tran e Greenfield 2002].

Uma mensagem é tipicamente composta de três partes básicas: cabeçalho, propriedades e corpo. O cabeçalho é utilizado tanto pelo MOM quanto pela aplicação para prover informações sobre a mensagem: destino, tipo, tempo de expiração, e assim por diante. As propriedades podem conter um conjunto de informações definidas pela aplicação no formato nome-valor. Essas propriedades são essencialmente partes do corpo da mensagem que foram promovidas para uma seção especial no intuito de permitir a realização de filtros pelos consumidores, assim como um roteamento especializado por parte do MOM. O corpo da mensagem possui várias implementações possíveis, sendo os principais: texto plano, código XML e formato binário.

O MOM é responsável pelo gerenciamento dos pontos de conexão dos múltiplos clientes e por gerenciar múltiplos canais de comunicação entre os pontos de conexão. Quando uma aplicação está se comunicando dessa forma, ela está atuando como um produtor ou um consumidor para o canal. Os produtores assumem o papel de transmissores produzindo mensagens e os consumidores, receptores consumindo mensagens. Uma aplicação pode ser ao mesmo tempo produtor e consumidor.

O produtor não necessita conhecer quais aplicações estão recebendo uma mensagem, ou em alguns casos nem mesmo quantas aplicações a estão recebendo. Do mesmo modo, o consumidor não necessita conhecer quais aplicações estão enviando os dados, sabe somente que recebe uma mensagem. Produtores e consumidores precisam apenas conhecer qual o formato das mensagens e que canal utilizar para enviá-las ou recebê-las. Se houver necessidade de uma resposta, esse fato é informado através da mensagem com a presença de um destino para identificar para que canal deve ser emitido uma resposta.

2.1.2 Canais de Comunicação

Os produtores e consumidores são fracamente acoplados uma vez que não se comunicam diretamente. Em vez disso, eles são abstratamente conectados um ao outro através de um canal virtual. Os canais podem ser classificados de acordo com o estilo de comunicação: ponto-a-ponto (PTP) e *publish-subscribe* (*Pub/Sub*). Os canais associados a cada um dos estilos são freqüentemente referenciados como filas e tópicos, respectivamente. A Figura 2-2 ilustra os dois modelos de canais.

No modelo *Pub/Sub* (ver Figura 2-2a), múltiplos consumidores (assinantes) podem registrar interesse em um tópico, realizando a assinatura do tópico. Um produtor ao publicar uma mensagem para um tópico faz com que cada assinante deste receba uma cópia da mensagem (comunicação por difusão ou *broadcast*).

Um tópico não é exclusivo de um produtor, logo vários produtores podem publicar mensagens para ele.

Uma mensagem somente é considerada consumida quando todos os assinantes do tópico receberem a notificação. As mensagens podem ser descartadas se não houver nenhum assinante para o tópico. Isso não significa que o assinante precisa estar conectado no momento que a mensagem foi produzida, apenas que ele tenha assinado o tópico previamente.

Para implementar essa coordenação, é criado um canal de saída para cada assinatura realizada em um tópico, com um único consumidor (o assinante). Com isso, quando uma mensagem é publicada nesse tópico, o canal de entrada (que recebeu a mensagem do produtor) gera uma cópia da mensagem original para cada canal de saída. Dessa forma, o MOM consegue coordenar o processo de entrega de mensagens nesse tipo de canal.

Banavar [Banavar et al. 1999] define um exemplo típico da utilização do modelo *Pub/Sub* quando apresenta uma solução para *Stock Trading*. Outros exemplos podem ser encontrados em [Calabria 2004] com sistemas de automação de frente de loja e em [Souto et al. 2004] com aplicações em redes de sensores.

No modelo PTP (ver Figura 2-2b), somente um consumidor pode receber uma mensagem enviada para uma fila. Se não existir nenhum consumidor conectado à fila, a mensagem fica retida no MOM até que algum receptor se habilite a recuperá-la.

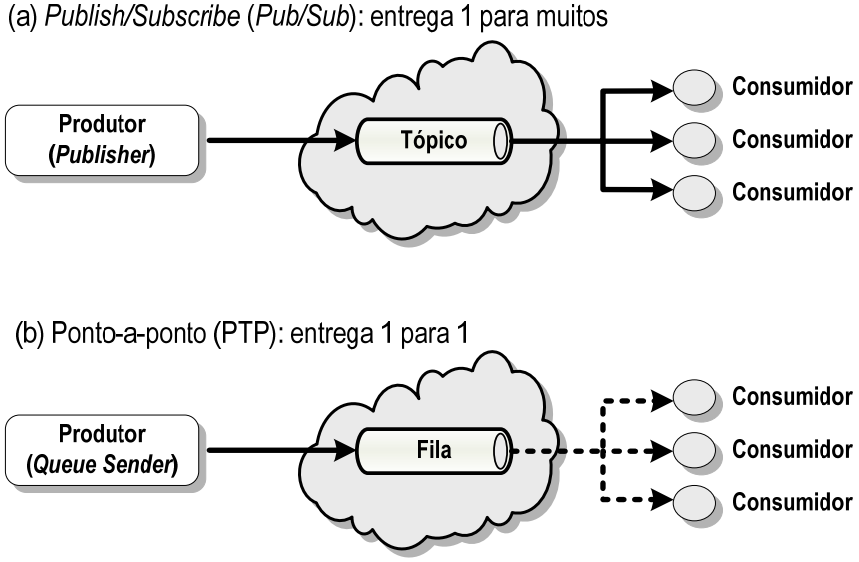


Figura 2-2. Modelos de canais de comunicação.

Pode ser desejável que vários consumidores estejam conectados a uma mesma fila, permitindo o consumo concorrente de mensagens. Essa estratégia é utilizada normalmente para aumentar a capacidade de processamento de mensagens, realizando uma distribuição de carga entre os vários consumidores. Outra aplicação dessa estratégia é na criação de sistemas tolerantes a falhas estabelecendo redundância quanto ao processamento das mensagens. É sempre bom ressaltar que somente um receptor pode consumir cada mensagem individual.

O Banco Central do Brasil desenvolveu o novo Sistema de Pagamento Brasileiro (SPB) que opera sob o suporte da Rede do Sistema Financeiro Nacional (RSFN) [BACEN 2007]. Essa plataforma exemplifica a utilização do modelo PTP em uma aplicação crítica de grande porte. A RSFN é responsável pelo tráfego de mensagens entre as instituições financeiras do país, o próprio BACEN e outras entidades ligadas ao setor financeiro, permitindo entre outras coisas que transações interbancárias sejam efetivadas em poucos minutos.

As plataformas de MOM disponíveis no mercado normalmente implementam os dois tipos de canais de comunicação e possibilitam que sejam definidos níveis de serviço para cada canal. Esse controle do nível de serviço está relacionado com características configuráveis quanto à confiabilidade (*reliability*), prioridade (*priority*), expiração (*expiration*), integridade transacional (*transaction*), escalabilidade e segurança na entrega das mensagens, permitindo que os canais possam atender a requisitos mais complexos das aplicações.

2.1.3 Confiabilidade no Canal

O fato da comunicação ser realizada de forma assíncrona entre produtor e consumidor faz com que o MOM necessite implementar mecanismos que possam garantir a confiabilidade na entrega das mensagens quando for requerida pelo produtor. Uma vez entregue uma mensagem ao MOM, o produtor continua seu processamento e assume que ela será entregue ao consumidor posteriormente. Dessa forma, ele pode requerer do sistema de entrega alguma garantia contra possíveis falhas do MOM, da rede ou do consumidor.

O MOM consegue ser tolerante a falhas implementando canais de comunicação que armazenam (persistem) as mensagens temporariamente em um repositório local. Para tanto, as mensagens enviadas precisam de uma marcação indicando o modo de entrega: *persistente*. Em alguns casos as aplicações não precisam alta confiabilidade na entrega da mensagem, utilizando, portanto, o modo de entrega: *não-persistente*. Um dos aspectos que pode motivar a não persistência das mensagens é a necessidade de uma maior agilidade no processo de transmissão, levando, obviamente, a uma redução da robustez do canal, além de aumentar a possibilidade de perda de mensagens em casos de contingência.

O serviço de persistência pode ser provido para uma mensagem ou para um canal (fila ou tópico). Na implementação desse serviço o MOM utiliza o mecanismo *store-and-forward* para cumprir o contrato com o produtor. O mecanismo de armazenamento (*store*) é utilizado para persistir a mensagem em disco, garantindo que a mesma possa ser recuperada em caso de falhas no MOM ou no consumidor. O mecanismo de encaminhamento (*forward*) é responsável em recuperar (*retrieve*) as mensagens armazenadas em disco e em seguida fazer o roteamento e a entrega delas ao consumidor.

Outra característica importante normalmente associada à confiabilidade do sistema de mensagens é a capacidade deste entregar as mensagens na mesma ordem recebida e sem duplicatas. Como a entrega fim-a-fim (produtor-consumidor) é realizada em duas etapas (produtor/MOM e MOM/consumidor), o MOM para garantir o ordenamento precisa criar mecanismos de identificação, reconhecimento (*acknowledgement*) e retransmissão (*redelivered*) das mensagens.

A Figura 2-3 apresenta o fluxo de comunicação entre duas aplicações utilizando dois modelos de canais confiáveis. O primeiro fluxo (ver Figura 2-3a) apresenta o modelo *Pub/Sub*, iniciando o processo na assinatura (*subscribe*) do tópico por parte do consumidor. Feito isso, algumas implementações de MOM permitem que o consumidor possa ficar até desconectado, pois uma vez assinado o tópico, o MOM se responsabiliza em guardar uma cópia da mensagem para cada consumidor registrado, essa característica é chamada de assinatura durável (*durable subscriptions*).

O fluxo mostrado na figura demonstra claramente a técnica de *store-and-forward* (persistência) discutida anteriormente. Outro aspecto relevante demonstrado através dele é que o MOM somente libera o produtor (com um ACK) depois de persistido (*persist*) a mensagem. Além disso, o sistema de entrega utiliza os reconhecimentos (ACKs) para garantir que nenhuma mensagem é retirada do canal (*remove*) sem antes ser confirmado o seu recebimento pelo consumidor.

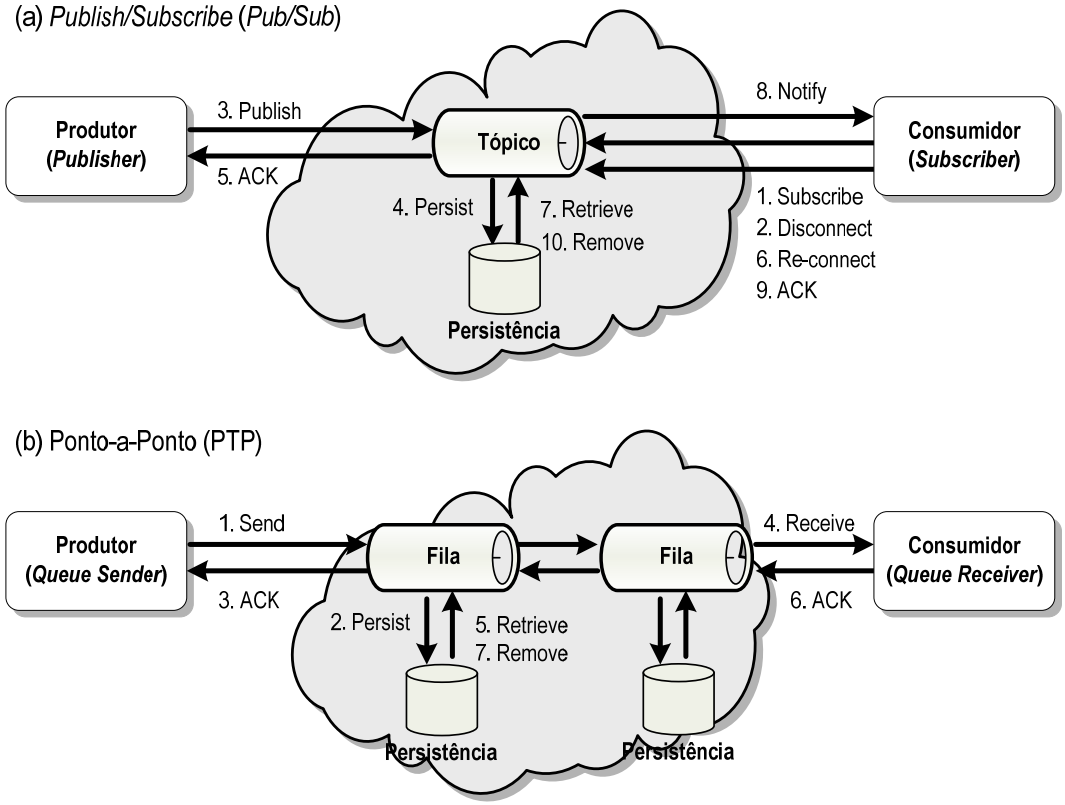


Figura 2-3. Fluxo de comunicação de canais confiáveis (persistência).

O fluxo relativo ao modelo PTP (ver Figura 2-3b) apresenta um canal com implementação distribuída, onde cada fila que integra o canal virtual fim-a-fim age utilizando a mesma técnica de *store-and-forward* apresentada no modelo *Pub/Sub*. Essa propriedade garante ao canal uma maior confiabilidade. A principal modificação no fluxo do modelo PTP é que o consumidor não precisa realizar uma assinatura prévia do canal. Uma vez recebida uma mensagem pelo canal, essa fica disponível para o consumidor independentemente deste estar *online*. A única situação fora dessa regra é quando a mensagem possui prazo para expiração definido.

Mais uma vez, pode-se notar que a mensagem somente é retirada de uma fila quando estiver entregue a outra fila (implementação distribuída com roteamento entre várias filas dentro do MOM), ou ao consumidor final. Além disso, o comportamento do consumidor normalmente é ativo na escuta do canal. Nesses casos, ele utiliza a API para verificar a disponibilidade de mensagens no canal, através de primitivas do tipo *receive*. Mas, é bom ressaltar que no modelo PTP a entrega também pode ser realizada de forma proativa através de uma notificação emitida pelo MOM, assim como acontece predominantemente no modelo *Pub/Sub*.

Quanto ao tratamento em situações de falha, o MOM na impossibilidade de transmissão de uma mensagem deve retransmití-la posteriormente, logo que a conexão com o consumidor do canal for normalizada. Falhas nas aplicações clientes (produtor ou consumidor) ou na rede não devem causar perdas de mensagens que já tiverem sido recebidas para transmissão pelo MOM, independente do modo de entrega do canal (com ou sem persistência, ou seja, em canais confiáveis ou não).

2.1.4 Integridade Transacional

Em um sistema de mensagens, produtores e consumidores não devem participar de uma mesma transação, pois se isso acontecesse violaria a característica de fraco acoplamento desses sistemas. Em vez disso, as aplicações (produtores ou consumidores) trocam mensagens com o MOM utilizando transações locais, pois a utilização de transações globais (fim-a-fim) exigiria que todas as entidades (produtores e consumidores) estivessem conjuntamente conectadas.

No contexto de um produtor ou de um consumidor individual, as plataformas de MOM disponibilizam um modelo para agrupar várias operações como uma única transação que possui escopo local para o produtor ou consumidor em questão. Nesse caso, as mensagens fruto dessas operações ou são todas processadas ou nenhuma deverá ser. Isso impede que parte das mensagens sejam entregues e que as demais fiquem no canal aguardando alguma contingência.

As transações locais podem conter operações de produção (*send*) e de consumo (*receive*) indistintamente, desde que pertencentes a um mesmo produtor ou consumidor. A Figura 2-4 apresenta exemplo de duas transacões locais. Na primeira o produtor também faz o papel de consumidor, integrando numa mesma transação local três operações (uma de consumo e duas de produção). A segunda transação local agrupa duas operações de consumo.

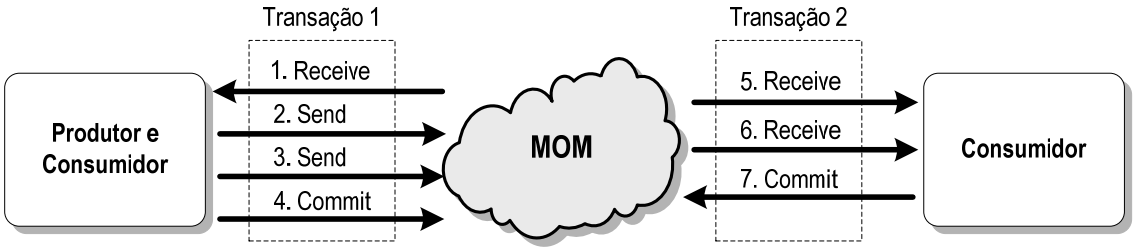


Figura 2-4. Fluxo de envio de mensagens utilizando transações locais.

Na perspectiva do produtor, todas as mensagens vão sendo retidas no sistema de mensagens até que aquele envie um comando *commit* para o MOM, que libera o

encaminhamento das mensagens até o consumidor. Se uma falha ocorrer em qualquer momento durante a transação, ou ainda, se o produtor enviar um comando *rollback* para o MOM, todas as mensagens que fazem parte dessa transação são descartadas. É importante ressaltar que nenhuma mensagem é liberada para o consumidor antes que o MOM receba a confirmação (*commit*) do produtor, efetivando a transação.

Na perspectiva do consumidor, as mensagens são entregues tão rapidamente quanto possível, mas durante o processo de entrega de mensagens utilizando uma transação local, as mensagens somente serão removidas do canal pelo MOM, quando este receber a confirmação (*commit*) do consumidor. Enquanto isso, as mensagens são armazenadas temporariamente em algum mecanismo recuperável. Se uma falha ocorrer durante essa transação, o MOM tentará retransmitir todas as mensagens.

O fato do produtor enviar mensagens agrupadas em uma transação não obriga que o consumidor as receba agrupadas, ou seja, pode-se ter controle de transação apenas em uma das pontas do comunicação.

Em alguns casos é necessário coordenar o envio e o recebimento de mensagens de uma transação local com atualizações em outro recurso transacional, assim como banco de dados. Essa necessidade tipicamente envolve um gerente de transações distribuídas. Um MOM pode fornecer esse serviço, porém o mais comum é que ele atue como um componente compatível com protocolos destinados a essa função, como, por exemplo, o *XOpen/XA two-phase-commit*. Vale ressaltar que apesar de suportar mecanismos distribuídos envolvendo vários recursos de várias entidades em uma única transação, não é aconselhável agrupar mais de um cliente (produtor ou consumidor) na mesma transação, com o intuito de implementar transações globais (fim-a-fim), pois essa ação viola o princípio do fraco acoplamento entre as entidades envolvidas na comunicação utilizando um MOM.

2.1.5 Escalabilidade

A escalabilidade de um sistema de MOM pode ser analisada em duas dimensões. Na primeira, a avaliação refere-se à capacidade de um canal atender uma demanda crescente de clientes, seja eles produtores ou consumidores. A segunda dimensão trata da capacidade do sistema manipular um número crescente de canais.

Um sistema de MOM permite ainda que a escalabilidade do sistema como um todo (aplicação e MOM) possa ser alcançada sem necessariamente serem implementados mecanismos específicos para isso. Por exemplo, considere que múltiplos clientes (produtores) enviam solicitações para serem processadas por um único consumidor, obtendo-se pouca escalabilidade da solução. Porém, adicionando múltiplos consumidores para esse canal eleva-se a capacidade de processamento significativamente. É importante ressaltar que no modelo PTP uma mensagem será entregue a apenas um único consumidor, estabelecendo uma distribuição de carga natural entre esses consumidores.

2.1.6 Segurança

A capacidade de efetuar escuta clandestina num canal *Pub/Sub* pode se tornar uma desvantagem desse modelo. Se o MOM transmite dados entre sistemas críticos, esses não podem permitir que qualquer outro cliente possa acessar esse canal, seja para escrita (produção) ou até mesmo para leitura (consumo). Os canais PTP de certa forma tratam melhor essa questão uma vez que cada mensagem somente é recebida por um único consumidor. Logo, se houver algum intruso escutando no canal, a aplicação provavelmente sentirá falta da mensagem lida de forma clandestina. Isso não resolve o problema de segurança, mas pelo menos faz o sistema detectar possíveis falhas.

Para resolver essas questões é importante considerar mecanismos que garantam a confidencialidade, a autenticidade e a integridade da mensagem. Para tanto, são utilizadas tecnologias de autorização e autenticação.

Além das características descritas até este ponto, a maioria das plataformas de MOM de mercado implementa outras características que possibilitam a adição de funcionalidades mais complexas aos seus mecanismos de transmissão assíncrona: balanceamento de carga, transparência de localização, transparência de migração, transparência de replicação, heterogeneidade e formas distintas de processamento das mensagens.

2.2 Java Message Service (JMS)

Atualmente, um dos padrões de mercado para implementação de MOMs é o *Java Message Service* (JMS) [Sun 2002], que na verdade é uma especificação de API para acessar sistemas de entrega de mensagens. Essa API define uma interface entre as aplicações e o próprio MOM (ver Figura 2-1).

Em conjunto com a API, o JMS define uma semântica que permite aplicações desenvolvidas em Java comunicar-se com plataformas de MOM, habilitando as aplicações a criar, enviar, receber e ler mensagens.

A especificação da API JMS define entre outros aspectos: regras para o comportamento operacional dos modelos de sistemas de mensagem (*Pub/Sub* e PTP); um conjunto rico e flexível de definições sobre o formato das mensagens; regras estritas para implementação do mecanismo *store-and-forward*; garantia de entrega; múltiplos mecanismos de reconhecimento de mensagens (ACK); controle de transação e recuperação de falhas. JMS também provê um modelo de transação multi-recurso, integrando entidades externas a uma transação.

2.2.1 Arquitetura JMS

Uma aplicação JMS é composta das seguintes partes: *JMS Provider*, *JMS Clients*, *JMS Messages*, e *administered objects (ver* Figura 2-5). O *JMS Provider* é um sistema de entrega de mensagens que implementa a interface especificada pela API JMS, além de funcionalidades adicionais não previstas na especificação, como balanceamento de carga e *clustering*. As principais implementações de mercado para *JMS Providers* são: JBoss Messaging, IBM Websphere MQ, FioranoMQ, JORAM, Sun Message Queue, SonicMQ, Apache ActiveMQ, TIBCO Enterprise for JMS.

O *JMS Client* é um programa ou componente de aplicação que produz ou consome mensagens. Para efetuar essas operações, ele se conecta ao *JMS Provider* através do *JMS Provider Client* .

As *JMS Messages* são objetos que comunicam informações entre *JMS Clients*, e podem conter diferentes tipos de dados, desde os mais simples, como texto plano, até os mais complexos, como *stream* de bytes ou objetos serializados.

Os *administered objects* são objetos JMS criados pelo administrador, pré-configurados e armazendos em um serviço de diretório ou serviço de nomes (ex. *JNDI Namespace*). Eles são acessados em tempo de execução pelos *JMS Clients* através de pesquisas no diretório (*lookup*). Os dois tipos de *administered objects* são os *Destinations* e as *ConnectionFactories*, responsáveis pela abstração dos canais (filas ou tópicos) e das conexões com os clientes, respectivamente. Eles formam a base para se escrever aplicações portáveis, sendo fundamentais para desacoplar os *JMS Clients*.

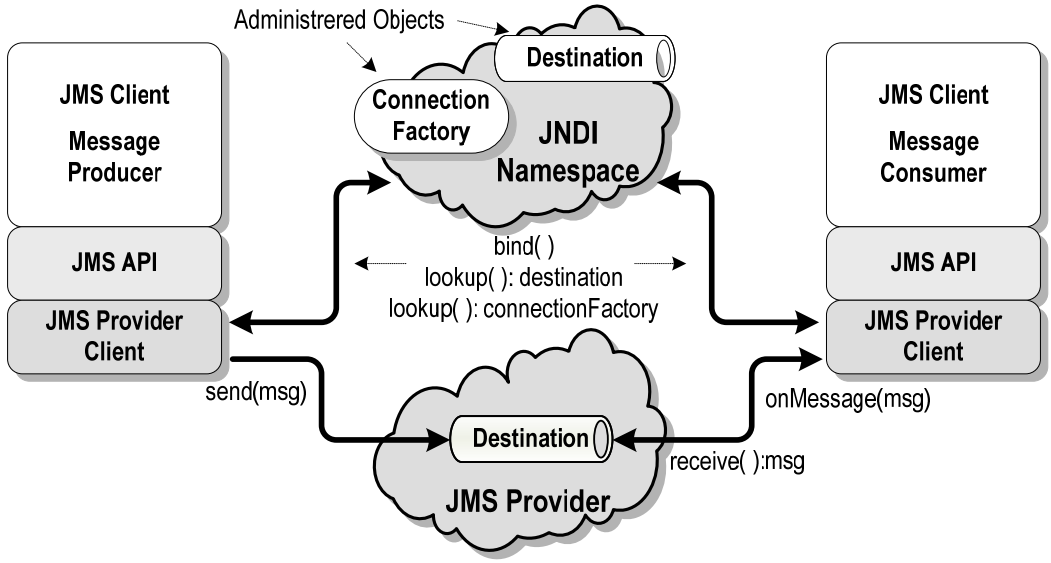


Figura 2-5. Comunicação entre aplicações na arquitetura JMS.

As *ConnectionFactories* são utilizadas para criar conexões (*connections*) que podem representar um *socket* TCP para o *Provider*. As *connections* por sua vez também agem como fábricas, e criam as sessões (*Sessions*).

As sessões são contextos *mono-thread* para enviar e receber mensagens, e são utilizadas como fábricas para criar os mais diferentes tipos de mensagens. Elas também são utilizadas para criar os objetos que representam o produtor (*Message Producer*) e o consumidor.(*Message Consumer*).

Os *Message Producer* e *Consumer* criados são associados a canais (*Destinations*) específicos do *JMS Provider*. Assim, consumidor e produtor podem receber/enviar mensagens apenas de/para o *destination* que eles estão associados.

Uma aplicação JMS pode utilizar tanto o estilo de mensagem PTP quanto o *Pub/Sub* para organizar sua comunicação, inclusive conjuntamente. Esses dois estilos são freqüentemente denominados pela especificação JMS de domínios.

Nem todos os objetos e métodos em JMS suportam acesso concorrente por múltiplas *threads* cliente. *Destinations*, *connections* e *connection factories*

suportam acesso concorrente, enquanto que *sessions*, *message producers* e *consumers* não suportam.

Uma das razões para as sessões (*sessions*) não suportarem acesso concorrente é que elas podem suportar transações, e múltiplas *threads* enviando e recebendo mensagens sobre uma mesma transação poderia complicar muito a sua coordenação. A mesma justificativa vale para os consumidores e produtores uma vez que eles são criados pelas sessões.

Outra razão para o acesso não concorrente, é que as sessões suportam consumo de mensagens de forma assíncrona. O *Provider* sempre serializa a entrega de mensagens para todos os consumidores associados com uma sessão configurada para operar assincronamente (via notificação), pois ele não pode exigir que os clientes (consumidores) sejam capazes de manusear concorrentemente múltiplas mensagens. Clientes mais sofisticados que desejem operar de forma concorrente utilizam múltiplas sessões.

2.2.2 Mensagem JMS

JMS define um conjunto de interfaces que representam diferentes tipos de mensagem, classificadas pelo tipo de conteúdo que elas possuem. A estrutura de uma mensagem JMS típica é apresentada na Figura 2-6. Independente do tipo da mensagem, todas elas são compostas por três elementos básicos: cabeçalho, propriedades e corpo da mensagem.

Figura 2-6. Estrutura da Mensagem JMS.

Os cabeçalhos das mensagens contêm um conjunto de campos padrões definidos pela especificação JMS. Os *Providers* têm que fornecer acesso a todos esses campos nas suas classes de implementação. Os valores desses campos são utilizados para rotear e processar as mensagens. Esses campos servem para identificar o canal destino, o ID da mensagem, modo de entrega (persistência), prioridade, tempo para expiração, entre outros aspectos.

As propriedades da mensagem são utilizadas no JMS para adicionar campos opcionais ao cabeçalho das mensagens. Eles podem possuir qualquer um dos tipos primitivos de Java, e são divididos em três categorias: padrão (extensão do cabeçalho), aplicação (utilizados pelos *JMS Clients*) e *provider* (proprietárias de cada implementação de MOM).

O corpo da mensagem pode suportar diferentes conjuntos de conteúdos, definindo o tipo da mensagem como um todo. Sendo assim, as mensagens JMS podem ser: *ByteMessage*, *TextMessage*, *ObjectMessage*, *StreamMessage*, e *MapMessage*.

2.2.3 Sessões

As sessões JMS são responsáveis pelo envio e recebimento de mensagens e são utilizadas para uma variedade de outros propósitos:

- Prover um método alternativo para a criação de canais (*destinations*);
- Criar canais temporários, produtores, consumidores e mensagens dos mais variados tipos;
- Suportar uma série simples de transações que combinem os consumidores e produtores em uma mesma seção automicamente;
- Definir um ordenamento para as mensagens que ela consome ou produz; e
- Reter mensagens consumidas até que elas sejam reconhecidas (Ack).

As sessões podem ser opcionalmente definidas como transacionais, podendo tratar um conjunto de mensagens enviadas e recebidas como uma unidade atômica simples. Quando uma conexão (*connection*) cria uma sessão, a interface recebe um parâmetro que especifica se a sessão a ser criada será transacional ou não.

O consumo de mensagens em uma sessão segue uma ordem serial o que é importante porque define o efeito do reconhecimento (*Ack*) das mensagens. O JMS define que as mensagens enviadas por uma sessão para um canal devem ser recebidas na ordem em que elas foram enviadas. Em compensação, o JMS não define nenhuma restrição para recepção das mensagens enviadas por múltiplas sessões, dependendo exclusivamente do controle de cada uma das aplicações que se utilizar desse recurso.

Alguns aspectos podem afetar a ordem em que as mensagens serão recebidas: mensagens de mais alta prioridade podem passar na frente de mensagens já enviadas; um cliente pode não receber uma mensagem devido a uma falha do JMS *Provider* (se utilizou um canal sem persistência); mensagens com diferentes modos de entrega (persistente, não-persistente) não possuem garantia de entrega ordenada.

2.2.4 Recepção Síncrona versus Assíncrona

Os consumidores (*MessageConsumers)* são entidades utilizadas pelos clientes JMS para receber mensagens, e são criados pelos *Sessions*. Eles podem receber mensagens de forma síncrona ou assíncrona.

Para realizar a recepção assíncrona, cada consumidor precisa registrar um *MessageListener* na sua sessão. Todos os consumidores associados a uma única sessão podem ter diferentes *MessageListeners*, mas a entrega de mensagens para cada um desses é serializada. Para cada mensagem que chega ao canal, o *Provider* entrega aos consumidores regristrados (via *Listeners*) chamando o método *OnMessage* definido pela interface JMS *MessageListener*. Esse mecanismo é chamado de notificação.

Quadro 2-1. Métodos para recepção síncrona.

```
public Message receive() throws JMSException
public Message receive(long timeout)throws JMSException
public Message receiveNoWait()throws JMSException
```

A recepção síncrona é realizada pelo consumidor fazendo chamadas a um dos três métodos apresentados no Quadro 2-1. O primeiro método realiza a chamada que bloqueia a *thread* até que exista uma mensagem disponível no canal associado com o consumidor. O segundo método tem a mesma funcionalidade, porém o bloqueio somente é realizado durante o tempo estabelecido pelo parâmetro *timeout*. O último método faz a leitura de mensagens com disponibilidade imediata no canal, retornando imediatamente mesmo que não encontre mensagem alguma.

2.2.5 Reconhecimento das Mensagens

Se uma sessão é transacional, o reconhecimento das mensagens (*Ack*) é manuseado automaticamente pelo *commit*, enquanto que na recuperação (*recovery*) é pelo *rollback*. Para as sessões não transacionais existem três modos para reconhecimento das mensagens, os quais são definidos quando a sessão é criada pela conexão: AUTO_ACKNOWLEDGE (reconhecimento automático para cada mensagem), CLIENT_ACKNOWLEDGE (reconhecimento sob responsabilidade do *JMS Client*) e DUPS_OK_ACKNOWLEDGE (reconhecimento em bloco realizado pelo *provider*).

2.2.6 Modo de Entrega

JMS suporta os modos de entrega (*delivery mode*) com e sem persistência. No modo com persistência (PERSISTENT) o *Provider* é instruído para garantir que a mensagem não seja perdida no trânsito, na ocorrência de falhas nele. Já no modo sem persistência (NON_PERSISTENT), falhas no *Provider* levam a perda das mensagens ainda não entregues.

A entrega NON_PERSISTENT deve garantir que até uma mensagem seja entregue ao consumidor, ou seja, ele pode perder a mensagem, mas nunca entregá-la duas vezes. Já na entrega PERSISTENT, o *Provider* deve garantir que não vai perder a mensagem. Além disso, ele deve controlar a entrega para não fazê-la mais de uma vez, pois cada mensagem deve ser entregue apenas uma única vez.

Os dois modos de entrega permitem aos clientes JMS escolherem entre desempenho e confiabilidade, pois o modo PERSISTENT afetará o desempenho em troca da confiabilidade, e o NON_PERSISTENT ao contrário.

A simples utilização do modo PERSISTENT não garante que todas as mensagens serão entregues para todos os consumidores elegíveis. Essa discussão é mais detalhada mais adiante na subseção que avalia a confiabilidade do canal.

2.2.7 Controlando o Tempo de Vida da Mensagem

Um cliente pode especificar um tempo de vida (*Time-to-live* ou TTL) para cada mensagem enviada. Esse valor define o tempo de expiração da mensagem, ou

seja, a validade da informação pode ser controlada pelo produtor. A especificação JMS não define a precisão que o *Provider* deve estabelecer para controlar mensagens expiradas, apenas não permite que ele despreze essa informação.

Nenhuma notificação de descarte está prevista na especificação, apenas ela afirma que os clientes não deveriam receber mensagens expiradas, mas não garante que não acontecerá.

2.2.8 Assinaturas no *Pub/Sub*

No domínio *Pub/Sub* os consumidores precisam se inscrever no canal (tópico) para que recebam notificações (mensagens assíncronas) com as mensagens que forem publicadas. Essa inscrição é chamada de assinatura do tópico. O JMS prevê dois modelos de assinatura para os consumidores *Pub/Sub*: assinatura normal e durável.

As assinaturas normais somente possuem validade enquanto a sessão estiver estabelecida e ativa entre o consumidor e o *Provider*. Se por algum motivo houver uma desconexão da sessão, o consumidor perderá as mensagens que chegarem durante esse tempo de inatividade.

A um custo elevado, os assinantes podem realizar suas assinaturas de forma a permanecerem válidas, mesmo quando o consumidor estiver inativo. Esse mecanismo é chamado de assinatura durável (*Durable Subscription*). Quando o consumidor voltar a atividade, todas as mensagens pendentes serão entregues, desde que não estejam expiradas. Para tanto, o *Provider* precisa reter essas mensagens até que o assinante durável volte a estabelecer uma sessão.

Normalmente, os *Providers* implementam o modelo *Pub/Sub* criando para cada assinatura uma fila de saída para aquele canal (tópico). Quando as mensagens chegam ao canal, ele apenas move uma cópia para cada fila de saída válida. Se as assinaturas não são duráveis, ao finalizar a sessão o *Provider* descarta a fila de saída para aquela assinatura.

2.2.9 Confiabilidade e Níveis de Serviço

Os clientes podem utilizar produtores que enviam mensagens PERSISTENT e NON_PERSISTENT para o canal simultaneamente, configurando o modo de entrega mensagem a mensagem. Com isso, a confiabilidade desejada pode ser ajustada de acordo com o tipo e a importância da mensagem.

No caso dos consumidores, eles normalmente processam totalmente cada mensagem antes de reconhecer sua recepção para o *JMS Provider*. Isso garante que o JMS não descarte a mensagem parcialmente processada, e diante de uma falha no cliente, a mensagem não seja perdida. O cliente implementa essa estratégia utilizando sessões transacionais ou com reconhecimento CLIENT_ACKNOWLEDGE.

Quando todas as mensagens precisam ser recebidas, e o domínio em questão for *Pub/Sub*, um consumidor utilizando uma assinatura durável deveria ser usado. JMS garante que mensagens publicadas enquanto um assinante durável está inativo são retidas pelo JMS, e entregues quando o assinante voltar a atividade.

Assinaturas não duráveis somente deveriam ser utilizadas se perdas de mensagens são aceitáveis.

Sendo assim, para implementar um serviço de comunicação de alta confiabilidade, a expectativa é que as mensagens sejam produzidas com o modo de entrega PERSISTENT e através de uma sessão transacional. Além disso, elas precisam ser consumidas por uma sessão transacional e utilizar um canal não temporário. Se a comunicação for *Pub/Sub*, deve-se utilizar uma assinatura durável. Essa estratégia garante produção apropriada, entrega confiável e precisão no consumo.

Mensagens NON_PERSISTENT, assinaturas não duráveis e canais temporários são por definição não confiáveis. Se o *Provider* falha ou é desligado, muito provavelmente causará perda de mensagens NON_PERSISTENT, além de todas as mensagens que estiverem em canais temporários ou em assinaturas não duráveis. O término da aplicação cliente (consumidora) pode causar uma perda de mensagens mantidas para suas assinaturas não duráveis e para seus canais temporários.

O nível de serviço provido pelo canal de comunicação está relacionado com características configuráveis, normalmente associadas à sessão. Essas características incluem: modo de entrega, confiabilidade, prioridade, expiração, integridade transacional, tipo de assinatura (modelo *Pub/Sub*), e método de reconhecimento.

Uma vez apresentado os conceitos básicos sobre o MOM, faz-se necessário para que o leitor possa obter referencial teórico adequado a compreensão plena deste trabalho, explorar o tema do planejamento de capacidade utilizando modelos de desempenho. Nas próximas seções são apresentados conceitos sobre modelagem de sistemas, planejamento de capacidade, e por fim é apresentada uma metodologia para realização de planejamento de capacidade de sistemas baseado na utilização de modelos.

2.3 Modelagem de Sistemas

Um modelo é uma representação abstrata de um sistema, assim como equações matemáticas que descrevem o comportamento de forma quantitativa de um determinado sistema. Uma vantagem de representar sistemas reais através de modelos é a possibilidade de prever a reação dos sistemas diante de situações nunca antes encontradas, ou de difícil representação no mundo real. Uma vez o modelo pronto, pode-se utilizá-lo para analisar o sistema real.

As principais técnicas de modelagem diferem, entre outros aspectos, quanto ao poder de representação, assim como a facilidade para modelar aspectos e características dos sistemas. Estas técnicas são classificadas como: combinatorial e baseada em espaço de estados [Balakrishnan e Trivedi 1995].

A modelagem de um sistema utilizando técnicas combinatórias é baseada na composição/combinação de fatores e aspectos individuais dos seus componentes. Cada componente é modelado individualmente a partir do fornecimento de suas características. Esta abordagem tem como principais limitações a dificuldade de representação de estruturas não triviais (onde uma composição serial e/ou paralela não seja evidente) e de dependências entre componentes. O problema de planejamento e gerenciamento da capacidade de

sistemas computacionais requer a utilização de modelos que facilitem a especificação dessas características, fazendo com que essas técnicas não sejam adequadas.

A modelagem baseada em espaço de estados consiste num conjunto de estados do sistema e de transições rotuladas entre esses estados. As cadeias de Markov é um exemplo de formalismo utilizado para modelagem de sistemas baseado em espaço de estados [Bolch et al. 1998]. A representação gráfica das cadeias de Markov é apresentada na Figura 2-7, e é baseada em um sistema de transição rotulado (LTS – *Labeled Transition Systems*), onde cada nó representa um estado do sistema modelado. Os arcos representam as transições entre os estados, sendo λ a probabilidade (tempo discreto) ou a taxa (tempo contínuo) com que esse evento ocorre. As cadeias representam um processo estocástico com uma boa relação entre o custo para obter a solução e a capacidade de modelagem.

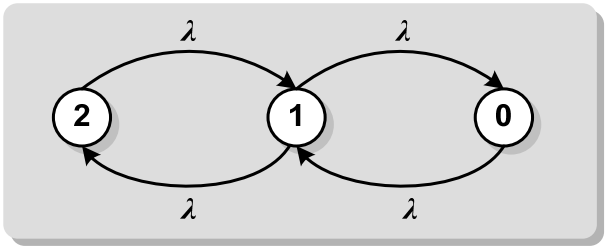


Figura 2-7. Cadeia de Markov para análise de confiabilidade de um sistema multi-processado.

Uma dificuldade na utilização das técnicas baseadas em geração do espaço de estados é que dependendo da complexidade do sistema, o espaço de estado do modelo pode ter dimensões muito grandes, o que eleva significativamente a complexidade computacional para realizar a sua avaliação precisa. Uma alternativa possível consiste na decomposição do modelo em sub-modelos, avaliação dos sub-modelos e posterior composição dos resultados obtidos para cada sub-modelo [Balakrishnan e Trivedi 1995].

Outra dificuldade é a construção do espaço de estados, quando precisam-se representar sistemas complexos, elevando a propensão a erros. Mais uma vez é fundamental a análise das propriedades qualitativas do modelo para evitar essas situações. No Apêndice A são apresentadas algumas propriedades qualitativas relacionadas aos modelos. A presença ou ausência destas propriedades é fundamental para a escolha do método de avaliação (resolução analítica ou simulação) ou ainda do método de execução (estacionária ou transiente).

Uma alternativa para as Cadeias de Markov são as redes de Petri (PN) [Murata 1989], que é um termo genérico associado a um conjunto de formalismos que possibilita a representação de estados locais (lugares), representados por círculos, e ocorrência de eventos (transições), representadas por retângulos (ver Figura 2-8). Os lugares armazenam marcas (*tokens*) que representam recursos ou condições específicas.

Dentre os formalismos que compõe a família redes de Petri, as redes de Petri Estocásticas (SPN – *Stochastic Petri Nets*) e suas extensões têm sido utilizadas como mecanismos de alto nível para geração automática de cadeias de Markov

[Bolch et al. 1998]. Isto permite que as dificuldades associadas à construção das cadeias sejam reduzidas.

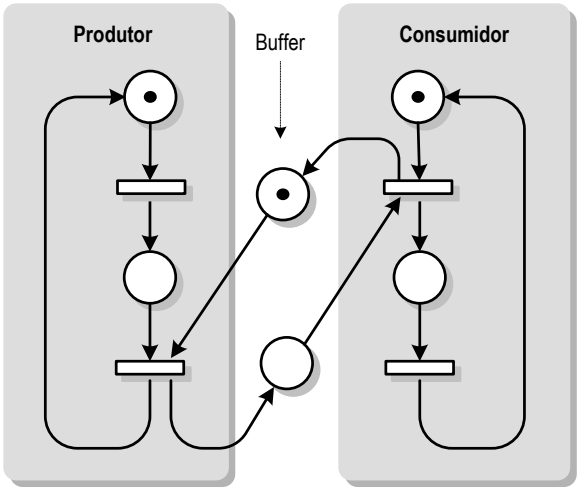


Figura 2-8. Modelo em redes de Petri: problema do produtor e consumidor.

Nas SPNs, a modelagem dos sistemas é baseada em estados locais, ou seja, foca-se na modelagem dos componentes ao invés de necessariamente ter que observar o sistema como um todo. Neste caso, é possível para o modelador concentrar-se mais no problema a ser solucionado do que suplantar dificuldades e restrições associadas à técnica de representação. Outra vantagem das SPNs é que permitem a análise e verificação de propriedades qualitativas dos modelos.

Uma limitação das SPNs é o fato de que as transições devem ser representadas por distribuições exponenciais (eventos markovianos), levando a uma limitação na modelagem de fenômenos cujos tempos de ocorrência não sejam exponencialmente distribuídos. Para contornar essa limitação, podem-se utilizar métodos de aproximação por distribuições de fases exponenciais [Neuts 1975]. Uma boa aproximação dessas distribuições ditas não-markovianas pode ser obtida usando equalização de momento (*moment matching*). Mais detalhes sobre SPNs, suas extensões, e o tratamento de eventos não-markovianos são apresentados no *Apêndice B*.

Os modelos mais ricos em detalhes normalmente são executados por simulação, consumindo assim um tempo maior para obter sua avaliação. Esse nível maior de detalhe é necessário quando se quer maior fidelidade desses modelos com o sistema real. Como conseqüência, esses modelos requererem um poder computacional maior durante a avaliação, levando a uma precisão dos resultados melhor.

Os modelos podem ser avaliados, também, através de uma resolução analítica (matemática). Nesse caso, o espaço de estados gerado deve possuir dimensões tratáveis computacionalmente. Por conta disso, os modelos tendem a capturar menos detalhes do sistema real, fazendo-os menos fidedignos. A vantagem desse tipo de modelo é que o tempo necessário para avaliá-lo é relativamente pequeno se comparado com modelos avaliados via simulação.

2.4 Planejamento de Capacidade

O intuito do planejamento de capacidade é garantir que as demandas futuras de carga possam ser atendidas de maneira efetiva considerando-se os acordos de nível de serviço e as restrições tecnológicas e de custo que forem aplicáveis. Enquanto isso, o gerenciamento de capacidade se preocupa em determinar se os níveis de serviço estão atualmente sendo atendidos pela capacidade instalada do sistema. Ou seja, enquanto o planejamento de capacidade tem a atenção voltada para o futuro o gerenciamento se detém no presente.

Na prática, planejamento de capacidade é o processo de prever quando os níveis futuros de carga saturarão o sistema (analisando para isso a evolução da carga devido às aplicações existentes e futuras), ou ainda quando eles comprometerão o nível de serviço prestado, e determinar o modo mais econômico de adiar a saturação ou o comprometimento ao máximo [Menascé e Almeida 2002].

Nessa definição, dois aspectos merecem destaque: previsão e saturação. Não é interessante esperar que a intensidade de carga aumente para depois descobrir o que acontece com o nível de serviço (por exemplo, com o tempo de resposta). A falta de um procedimento de planejamento pró-ativo pode levar a problemas inesperados de indisponibilidade e desempenho causados pela sobrecarga (ou saturação) de alguns recursos do sistema. Um sistema é dito saturado quando um de seus recursos se aproxima de 100% de utilização, tornando-se um gargalo para o seu perfeito funcionamento, levando ao comprometimento do seu desempenho.

O nível de serviço de um sistema possui forte relação com o custo da infra-estrutura necessária para suportá-lo. Essa relação forma a base do planejamento da capacidade.

A qualidade de serviço prestado por um sistema normalmente está associado ao seu desempenho. Se este for ruim pode gerar insatisfação dos usuários do serviço, ocasionando perdas para o negócio. Logo, o planejamento de capacidade está intimamente ligado à avaliação constante do desempenho de um sistema.

Outro motivo para realizar antecipadamente o planejamento de capacidade é que a solução de um problema de desempenho pode não ser simples de ser implantada. Não basta ter recursos financeiros disponíveis para resolvê-lo. Muitas vezes a solução implica em mudanças arquiteturais na infra-estrutura ou até mesmo no sistema, consumindo um tempo bastante considerável para especificação, compra e implementação. Mesmo que essa mudança não seja necessária, a empresa precisa de um tempo considerável para adquirir, instalar e configurar os recursos adicionais suficientes para garantir que o nível de serviço seja restabelecido.

Resumidamente, o planejamento de capacidade é importante para evitar perdas financeiras, motivadas pela queda de desempenho ou até mesmo a indisponibilidade do sistema; para garantir a satisfação do usuário; para preservar a imagem externa da empresa; e porque problemas de capacidade não podem ser resolvidos instantaneamente.

A utilização das técnicas de planejamento de capacidade não deve ser exclusiva de situações onde grandes mudanças sejam demandadas na carga de trabalho.

Elas integram também a atividade de gerenciamento da capacidade do sistema, pois ao contrário de situações com grandes mudanças, a carga de trabalho deve ser monitorada constantemente, detectando tendências na sua evolução.

Da mesma forma, o nível de serviço deve ser monitorado para identificar possíveis problemas com a capacidade instalada, fazendo assim os ajustes necessários (*tuning*) no sistema para garantir a manutenção da qualidade do serviço no nível desejado. Esses ajustes muitas vezes implicam apenas em alterações nas configurações do sistema, ou até mesmo realocação de recursos existentes. No gerenciamento da capacidade, a configuração do ambiente e a carga atual são entradas no processo de *tuning*.

Entretanto, no planejamento de capacidade primeiramente é realizado uma previsão da carga futura baseada nos dados obtidos a partir de uma monitoração de longa duração do sistema, elaborando-se cenários futuros de carga. Em seguida, se junta a esses cenários as diferentes alternativas de configuração possíveis para melhorar o desempenho do sistema. A partir daí, é realizado um dimensionamento (*sizing*) da solução necessária para manter os níveis de serviço acordados, evitando a saturação do sistema.

É importante prever com antecedência o impacto das mudanças no desempenho do sistema como um todo. Para isso, a utilização de técnicas de avaliação de desempenho baseadas em modelos (analíticos ou simulação) torna-se um diferencial, pois vários cenários podem ser avaliados para a escolha do mais adequado. Essa estratégia vale tanto para o planejamento quanto para o gerenciamento da capacidade, pois em ambas as situações um modelo de desempenho para o sistema pode ser desenvolvido.

2.4.1 Dificuldades no Planejamento de Capacidade

A tarefa de planejar a capacidade de sistemas enfrenta uma série de dificuldades e problemas que foram resumidos e apresentados por [Jain 1991]. Inicialmente, uma das dificuldades é o fato de não existir uma nomenclatura padrão para a área. Alguns fabricantes vendem produtos que julgam serem para planejamento de capacidade, quando na verdade auxiliam apenas o *tuning* de sistemas. Enquanto que outros vendem produtos de *sizing* para serem utilizados no planejamento, porém não permitem a caracterização da carga de trabalho.

Um problema de difícil resolução nessa área é prever o comportamento do sistema quando da inserção de novas tecnologias. Como não se tem parâmetros para determinar o comportamento futuro dos elementos novos, tende-se a defini-lo em função das tecnologias atuais, o que normalmente distorce os resultados da previsão.

Os modelos gerados para representar o sistema, assim como os modelos de carga, precisam de parâmetros de entrada que nem sempre podem se obtidos via medição. Isso acontece, por exemplo, quando o sistema é fechado, fazendo com que não se consiga instrumentá-lo para obtenção de resultados internos. Outro problema é a inviabilidade de se obter certas medidas relativas ao comportamento do usuário do sistema, como, por exemplo, o *think time*.

Para que se possam utilizar os modelos na representação do sistema real são fundamentais suas validações para garantir que os resultados obtidos pelos modelos possam substituir com relativa precisão o ambiente real. O problema é

que a validação nem sempre é fácil de ser realizada. O procedimento ideal de validação é comparar os resultados obtidos pelo modelo com resultados medidos no sistema real para vários cenários de carga diferentes, fazendo com que o modelo possa ser testado em circunstâncias diversas. Sem uma validação desse tipo é suspeito o planejamento de capacidade utilizando esses modelos.

Outro aspecto importante é que a avaliação de desempenho do sistema é apenas uma parte do planejamento de capacidade. Precisa-se ainda acoplar a esses resultados informações sobre os custos das soluções com o intuito de realizar uma análise de capacidade versus custo, fazendo essa atividade ainda mais complexa.

Como não existe nenhuma definição aceita como padrão para capacidade de um sistema, alguns a definem como a vazão máxima (*throughput* máximo) obtida pelo sistema. Entretanto, outros definem capacidade como a quantidade máxima de usuários que o sistema pode suportar enquanto mantém seus objetivos de desempenho. Nesse caso, o número de usuários é apenas um exemplo de uma unidade de carga.

Outra dificuldade importante ressaltada por [Jain 1991] é que existe um número de diferentes capacidades para o mesmo sistema. Normalmente um sistema possui uma capacidade nominal (teórica), uma capacidade útil (prática) e ainda uma capacidade no joelho (ponto de inflexão). Mais detalhes podem ser encontrados no *Apêndice A*, na Seção A.3.

2.4.2 Definindo a Capacidade Adequada de um Sistema

A definição da capacidade adequada para um sistema depende de três elementos principais: acordos de nível de serviço (SLA, *Service Level Agreement*); tecnologias e padrões especificados; e restrições de custo.

A capacidade para ser considerada adequada precisa garantir que os níveis de serviço prestados pelo sistema estejam alinhados com os níveis estabelecidos pelo SLA. A oferta da capacidade adequada para atender ao SLA é realizada utilizando um conjunto de tecnologias e padrões especificados (ex. servidores, sistemas operacionais, MOM, banco de dados, rede, etc.) que podem ser configurados e combinados das mais diversas formas. A decisão de quais recursos utilizar, e como combiná-los, normalmente, seguem aspectos de desempenho, apesar de que em algumas organizações essa decisão é baseada na facilidade de administração, na familiaridade com o ambiente, no custo, na quantidade e qualidade dos fornecedores.

O problema de oferecer a capacidade adequada para atender ao SLA seria bem mais fácil de resolver se houvesse recursos financeiros ilimitados. As restrições de custo são estabelecidas pela gerência e limitam o espaço das possíveis soluções. Os custos envolvidos em uma solução compreendem o custo de aquisição, desenvolvimento, instalação, treinamento, operação, suporte, manutenção. A somatória desses custos compõe o custo total de propriedade (TCO, *Total Cost of Ownership*).

Resumidamente, um sistema possui a capacidade adequada se o SLA é continuamente atendido para tecnologia e padrões especificados, e se os serviços são fornecidos dentro das restrições de custo definidas [Menascé e Almeida 2002].

2.5 Metodologia para Planejamento de Capacidade

Em [Almeida e Menascé 2002], os autores apresentam um processo para planejamento de capacidade de *Web Service*, baseado na representação dos serviços através de modelos de desempenho e de carga. Esse processo é a evolução da metodologia apresentada em [Menascé e Almeida 2002]. Em [Menascé et al. 2004] os autores apresentam uma nova versão da metodologia incorporando o conceito do processo definido recomendando um conjunto de ações e boas práticas na condução de processos de planejamento e de capacidade na área de *Software Performance Engineering*.

Ao longo dessa seção a metodologia de Menascé et al. é apresentada em detalhes. Inicialmente, ela define objetivos de desempenho, depois segue um ciclo formado de oito passos, conforme apresentado na Figura 2-9. Os objetivos de desempenho são utilizados para estabelecer os níveis de serviço desejados, assim como para especificar as métricas de desempenho que são adotadas para avaliar o sistema sob análise.

Durante o passo de entendimento do ambiente são obtidas informações acerca do sistema sob análise, incluindo a sua arquitetura (hardware e software) e os padrões de uso (por exemplo, períodos de pico de utilização). Normalmente, a coleta dos dados relativos aos padrões de uso é realizada através de técnicas de instrumentação e monitoração do sistema. Um aspecto importante durante o entendimento do ambiente é a identificação de parâmetros do sistema, que são utilizados na construção de cenários típicos de utilização.

Com base nos dados coletados, é realizada uma caracterização da carga de trabalho que envolve a sua decomposição em componentes básicos, identificando para cada um desses componentes os parâmetros (intensidade de carga e demanda por recursos) que os caracterizam.

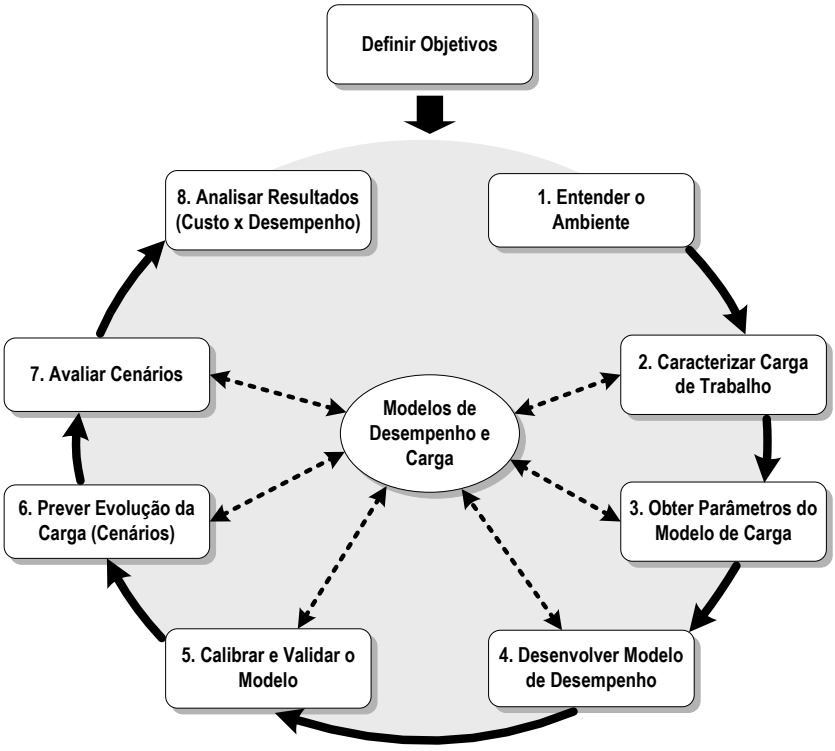


Figura 2-9. Metodologia para planejamento e gerenciamento de capacidade.

O terceiro passo consiste na medição do sistema real para obtenção de valores para os parâmetros do modelo de carga. A partir desses dados é elaborado um modelo de carga, capaz de representar a carga real submetida ao sistema que foi descrita no processo de caracterização.

O quarto passo da metodologia trata do desenvolvimento de um modelo de desempenho para o sistema. Esses modelos são utilizados para representar o comportamento de sistemas complexos. A grande vantagem da utilização de modelos é a capacidade de realizar predições de desempenho analisando modificações na carga ou na arquitetura do sistema.

O Passo 5 cuida da validação e calibragem dos modelos. Um modelo para que possa ser utilizado em experimentos de planejamento de capacidade precisa ser validado. Ele é considerado válido se as métricas de desempenho obtidas a partir dele se aproximarem suficientemente dos valores correspondentes medidos diretamente do sistema real. Tratando-se de planejamento de capacidade margens de erro de até 30% são aceitáveis [Menascé e Almeida 2002].

Depois de validado os modelos, a metodologia estabelece a necessidade de realizar a previsão da evolução da carga de trabalho, através da construção de cenários prováveis a que o sistema será submetido. Nesse passo, os modelos de carga são utilizados para prever e modelar o comportamento futuro da intensidade de carga e o seu reflexo nas demandas de serviço para cada recurso do sistema.

Uma vez construído os cenários, uma predição de desempenho utilizando os modelos de carga e de desempenho do sistema avalia os cenários definidos valorando as métricas estabelecidas. A predição de desempenho pode ser definida como o processo de estimar *métricas* de desempenho para um determinado conjunto de *parâmetros* [Menascé et al. 2004]. Tais parâmetros podem ser inerentes ao próprio sistema ou à carga a que este pode ser submetido. Os parâmetros de carga são determinados durante o Passo 3, enquanto que os parâmetros do sistema durante o Passo 2.

Finalmente, a metodologia recomenda que os resultados de cada cenário sejam analisados em conjunto com informações de custo para se determinar a forma mais barata de se manter os níveis de serviço acordados (definidos em função das métricas) por mais tempo, adiando assim a saturação do sistema.

As próximas subseções apresentam mais detalhes das recomendações e boas práticas recomendadas pelos autores para cada um dos passos definidos na metodologia.

2.5.1 Definir Objetivos

Os usuários de serviços de TI geralmente não estão preocupados com métricas como *Utilização de CPU*, *Largura de Banda de Rede*, *Taxa de Falha* e outros indicadores de desempenho. Pelo contrário, eles normalmente se interessam por métricas relacionadas à qualidade do serviço prestado (QoS). Essa qualidade é percebida através de métricas como *Tempo de Resposta*, *Disponibilidade*, *Confiabilidade*, *Segurança* e *Custo*.

Os objetivos de desempenho devem ser estabelecidos de forma clara, de preferência através de um contrato ou acordo entre os usuários e os gestores do serviço. Esse contrato é chamado de *Service Level Agreement* (SLA), e estabelece, além dos aspectos legais e contratuais, os níveis de serviço mínimos que deve ser prestado de forma a satisfazer as necessidades dos usuários. Esses níveis mínimos compõem uma parte do contrato denominado de *Service Level Objectives* (SLO), que define limites para métricas percebidas pelo usuário, porém muitas vezes para que essas métricas sejam alcançadas precisam-se monitorar outras métricas específicas do sistema em si.

Os serviços de TI são suportados por um sistema complexo formado por servidores, redes de comunicação, plataformas de *middleware*, aplicações, entre outros recursos. Logo, para cumprir o SLO é necessário um estudo da capacidade desse sistema, determinando se ela está adequada ou não. Esse planejamento da capacidade precisa utilizar os objetivos de níveis de serviço estabelecidos como metas a serem alcançadas.

2.5.2 Entender o Ambiente

Uma vez definido os objetivos de desempenho para um sistema, é necessário o entendimento do ambiente onde ele está inserido. O primeiro passo é uma revisão em profundidade da sua arquitetura com o enfoque de desempenho, determinando as características configuráveis (parâmetros) que podem afetar ou comprometer a obtenção dos objetivos estabelecidos. O passo seguinte é compreender os padrões de uso do sistema, mapeando os parâmetros que ajudam a descrever a carga de trabalho. Por fim, é preciso identificar quais desses parâmetros afetam o desempenho do sistema e de que forma afetam.

A compreensão dos padrões de uso impostos pela aplicação é importante para estabelecer cenários de carga adequados durante um processo de planejamento de capacidade. Uma discussão bastante importante é se o dimensionamento de um ambiente deve ser estabelecido pelos padrões de uso no pico, ou na média.

Quando se utiliza os valores de pico no dimensionamento, garante-se que os níveis de serviço são atendidos em todas as situações. Quando se utiliza a média, sabe-se previamente que durante os momentos de pico, alguns níveis de serviço serão descumpridos.

O processo de planejamento e gerenciamento da capacidade deve estudar situações de pico para determinar a capacidade máxima do sistema, além de avaliar o impacto desses picos nos níveis de serviço prestado [Menascé e Almeida 2002]. Com essas informações os gestores podem avaliar a relação custo-benefício de cada configuração de capacidade avaliando qual o dimensionamento adequado em cada situação.

2.5.3 Caracterizar a Carga de Trabalho

Depois de concluído o estudo sobre o ambiente, é necessário realizar a caracterização da carga de trabalho, que pode ser definida como o conjunto de todas as informações de entrada que o sistema recebe do seu ambiente durante qualquer período de tempo determinado [Menascé e Almeida 2002].

O primeiro passo no processo de caracterização da carga de trabalho é a identificação dos seus componentes básicos. O componente básico refere-se à

uma unidade genérica de trabalho que chega a um sistema a partir de uma fonte externa [Menascé e Almeida 2002].

No processo de planejamento de capacidade definido nesta metodologia, a avaliação de desempenho é conduzida através da modelagem analítica (ou para simulação) do sistema a ser avaliado (*System Under Test*, SUT), levando necessariamente à utilização de modelos não-executáveis para representar a carga submetida ao SUT [Menascé e Almeida 2002].

Os modelos de carga não-executáveis são descritos por um conjunto de valores médios de parâmetros que reproduzem o mesmo uso de recursos básicos do SUT imposto pela carga de trabalho real. A atenção para o contexto de planejamento de capacidade é apenas com os recursos que possuem impacto sobre o desempenho do sistema e que podem sofrer esgotamento levando o SUT ao ponto de saturação, comprometendo os níveis de serviço desejados.

Cada um dos parâmetros do modelo de carga indica um aspecto do comportamento de execução do componente básico no SUT e precisa descrever o comportamento dos centros de serviço (recursos de hardware e software) ou dos clientes (requisições, transações). Uma composição específica de valores escolhidos para esses parâmetros forma um cenário de carga.

De forma geral, com base nos dados coletados, é realizada uma caracterização da carga de trabalho que envolve a sua decomposição em componentes básicos de acordo com a sua intensidade de carga e com a sua demanda por recursos. A partir desses dados é elaborado um modelo de carga, capaz de representar os vários cenários de carga descritos na caracterização.

Porém, como cada sistema possui características peculiares na caracterização de sua carga de trabalho, [Menascé e Almeida 2002] apresenta um conjunto de etapas para efetuar essa atividade, e que é comum a todos os sistemas:

1. Identificação dos componentes básicos da carga de trabalho;
2. Seleção do conjunto de parâmetros que capturam as características mais relevantes da carga de trabalho para a finalidade do estudo;
3. Monitoração do sistema para coletar dados brutos de desempenho;
4. Particionamento da carga de trabalho em classes de componentes semelhantes;
5. Cálculo dos valores dos parâmetros que caracterizam cada classe; e
6. Construção e validação de um modelo de carga de trabalho.

Durante esse passo de caracterização da carga de trabalho a metodologia define que devam ser identificados os componentes básicos e selecionado o conjunto de parâmetros que representam os aspectos de desempenho da carga submetida ao MOM.

2.5.4 Obter Parâmetros do Modelo de Carga

O próximo passo da metodologia para planejamento e gerenciamento da capacidade consiste na medição do sistema real para obtenção de valores para os parâmetros do modelo de carga. Esse passo complementa a caracterização da carga de trabalho, realizando as etapas de monitoração do sistema,

particionamento da carga de trabalho, cálculo dos valores dos parâmetros e construção do modelo de carga de trabalho.

Monitoração do Sistema

A monitoração de desempenho de um sistema permite o entendimento dos seus parâmetros e o estabelecimento da ligação entre o sistema e o modelo, atribuindo valores para os parâmetros. A melhor forma de estudar o desempenho de um determinado sistema é executar a carga de trabalho real, sob condições operacionais reais e medir os resultados [Menascé e Almeida 2002].

A atividade de medição é uma parte chave para todas as tarefas da área de *performance engineering* [Menascé et al. 2004], formando a base para a elaboração de um modelo capaz de representar a carga real submetida ao SUT. Segundo [Menascé e Almeida 2002], medições de desempenho devem ser coletadas de diferentes referenciais, cuidadosamente escolhidos para observar e monitorar o SUT. Porém, nem sempre é possível utilizar essa técnica.

Os *benchmarks* se apresentam como uma alternativa viável nessas situações, pois definem um conjunto de cargas "artificiais" que exploram diversas características dos sistemas avaliados nas mesmas condições operacionais. Os *benchmarks* são muito utilizados para realizar a comparação entre sistemas, pois permite que uma carga "bem escolhida" seja repetida para vários sistemas.

Em ambas as situações, é necessário utilizar monitores para coletar as informações de desempenho do sistema. De uma forma geral, os monitores de desempenho e os sistemas de contabilização e registro (*accounting and logging systems*) são utilizados para se determinar os valores dos parâmetros para cada componente básico da carga de trabalho [Menascé et al. 1994].

Monitores são ferramentas utilizadas para observar as atividades de um sistema, e em geral coletam informações estatísticas, analisam os dados e apresentam os resultados [Jain 1991].

As ferramentas de contabilização e registro são normalmente partes integrantes dos sistemas operacionais, ou das aplicações, e fornecem informações sobre as atividades realizadas nos principais recursos da plataforma (ex. CPU, I/O).

O problema principal dessas ferramentas é que a maioria delas provê medidas agregadas para os níveis dos recursos (ex. número total de pacotes transmitidos pela rede, ou a utilização total de CPU para máquina Java), dificultando se determinar quanto está associado a um componente básico específico.

Mais informações sobre medição de sistemas podem ser obtidas no Apêndice A e em [Menascé e Almeida 2002] [Menascé et al. 1994] [Menascé e Almeida 1998] [Menascé e Almeida 2000] [Menascé et al.2004].

Particionamento da Carga de Trabalho

Após a monitoração do sistema, obtém-se um conjunto de componentes básicos heterogêneos. Essa heterogeneidade leva a uma diminuição da precisão se todos esses componentes básicos forem incluídos em uma única classe de carga.Logo, o particionamento da carga de trabalho em classes de carga é motivado pela melhoria da precisão dos modelos, além de uma melhoria significativa da representatividade da caracterização. As técnicas de particionamento dividem

sua carga de trabalho em classes de componentes homogêneos, agrupando componentes que sejam de alguma forma semelhantes.

Podem-se utilizar alguns atributos do componente básico para agrupá-lo, estabelecendo a semelhança entre eles. O uso de recursos (ex. utilização de CPU) é um importante atributo para agrupar componentes em classes. Em situações reais, é provável que você precise agrupar componentes básicos utilizando mais de um atributo.

Cálculo dos Valores dos Parâmetros

Após particionar a carga de trabalho em uma série de classes, o gerente de capacidade precisa determinar o valor dos parâmetros para cada classe de componente básico. As técnicas mais utilizadas são: média, agrupamento e através de distribuições probabilísticas.

A técnica para caracterizar a carga de trabalho está focada na demanda por recursos do sistema que cada componente básico impõe. Quando a carga de trabalho é particionada em classes se procura encontrar semelhança entre os componentes. Porém, nem sempre o particionamento consegue criar uma homogeneidade entre os componentes da mesma classe.

Quando existe homogeneidade dentro da classe, pode-se utilizar a técnica do valor médio, que se preocupa em encontrar a média aritmética de cada parâmetro entre todos os componentes (mensagens) da classe. Essa média é calculada para cada parâmetro, ou seja, uma vez particionada a carga em classes calcula-se a média aritmética de todos os valores obtidos para um parâmetro.

Um aspecto importante que o gerente de capacidade precisa estar atento é se a amostra coletada com as mensagens para compor a carga de trabalho, depois de particionada em classes, está realmente homogênea. Isso pode ser avaliado através de uma análise de medições estatísticas de variabilidade, tais como: desvio padrão, variância, coeficiente de variação, e amplitude (diferença entre máximo e mínimo).

Se a classe não estiver homogênea, deve-se utilizar a técnica do agrupamento, para dividir a classe de carga em grupos de componentes homogêneos. Esta técnica tenta encontrar grupos naturais de componentes com base em requisitos semelhantes quanto à utilização de recursos. Eles procuram agrupar os componentes de forma que o valor do *centróide*[1] de cada grupo possa representar todos os elementos do grupo, de forma que a variabilidade dentro do grupo seja pequena, e a variabilidade entre os *centróides* seja a maior possível.

Em alguns casos, existe a necessidade de realizar tratamento de *outlier* (ponto fora da curva), para retirar valores indesejáveis, que podem distorcer o cálculo dos *centróides*. Deve-se ter cuidado, porém, para não retirar componentes que caracterizam cargas com cauda pesada, determinadas através de eventos raros que afetam o desempenho do sistema no instante quando ocorrem. Um exemplo de evento que não deve ser considerado como *outlier* é a entrada do coletor de lixo da JVM (*garbage collector*).

[1] Centróide é o ponto cujo valor representa a média aritmética dos valores do grupo.

Outra forma de calcular o valor de um parâmetro de uma classe de carga não homogênea é através da identificação de uma distribuição probabilística que se comporte de forma similar aos valores medidos. Parâmetros expressos como taxas e atrasos são variáveis aleatórias contínuas. O *Apêndice A* aprofunda a discussão sobre variáveis aleatórias, mostrando que algumas distribuições (ex. Erlang) podem ser representadas ou aproximadas por uma combinação de exponenciais através de uma técnica denominada de *Phase Type Distribution*.

Em situações reais, alguns parâmetros possuem distribuições ditas empíricas, obtidas através da apuração estatística dos valores medidos. Para representar essas distribuições pode-se utilizar a técnica de equalização de momentos (*moment matching*), para aproximá-las com as distribuições Erlang e Hiperexponencial. Em seguida, estas são representadas por uma combinação de exponenciais, levando a um modelo markoviano, ou semi-markoviano. Mais detalhes são fornecidos pelos *Apêndices A e B*.

Construção do Modelo de Carga de Trabalho

Uma vez encontrado os valores para cada parâmetro de cada classe da carga de trabalho, a próxima etapa é desenvolver um modelo que consiga representar o comportamento dessa carga, permitindo submetê-lo ao modelo de desempenho do sistema.

O modelo precisa ser capaz de representar um *mix* de classes de mensagens utilizando uma distribuição percentual para cada classe de carga, podendo esse comportamento do modelo variar no tempo.

2.5.5 Desenvolver o Modelo de Desempenho

Depois de concluído o processo de caracterização da carga de trabalho, com a obtenção dos parâmetros do modelo de carga, é preciso desenvolver um modelo de desempenho para o sistema, para que em conjunto com o modelo da carga de trabalho possam auxiliar na previsão de desempenho de forma a inferir se os níveis de serviço desejados serão alcançados.

A previsão de desempenho é o processo de estimar métricas de desempenho de um sistema para um determinado conjunto de parâmetros [Menascé e Almeida 2002]. Os parâmetros são divididos nas seguintes categorias:

- *Parâmetros do sistema* caracterizam aspectos da arquitetura do ambiente, definindo desde parâmetros relacionados à plataforma operacional, quanto ao *Middleware* em si. Esses parâmetros foram detalhados no estágio de entendimento do ambiente;
- *Parâmetros do recurso* definem as características intrínsecas de um recurso que afetam o seu desempenho, e conseqüentemente o desempenho do sistema;
- *Parâmetros da carga de trabalho* foram definidos durante a caracterização da carga de trabalho, e são divididos em parâmetros de intensidade de carga, e de demanda de serviço.

A realização de previsão de desempenho exige a utilização de modelos para representar o sistema. Quando se constrói um modelo, o nível de detalhe incorporado nele depende de sua finalidade. Do ponto de vista de desempenho, dois tipos de modelos podem ser construídos para um sistema: modelos de

simulação e modelos analíticos. Os dois tipos precisam considerar a disputa pelos recursos e as filas que surgem em cada recurso do sistema (ex. CPU, discos, roteadores, enlaces de comunicação). As filas também surgem para recursos de software (ex. sessões, *threads*).

Nos modelos de simulação, a descrição do sistema é embutida em um programa de computador que ao executar simula a sua operação. Nestes modelos, o estado do sistema é normalmente representado através de variáveis que assumem valores discretos. Ao longo de um experimento de simulação, as operações do sistema são simuladas repetidas vezes e as métricas configuradas são computadas. Um experimento de simulação deve continuar em execução até que o intervalo de confiança para a média de cada métrica configurada no modelo atinja uma precisão especificada. As dimensões do modelo e o intervalo de confiança desejado, além da quantidade de métricas coletadas, possuem um grande impacto no tempo necessário para a finalização do experimento.

Nos modelos analíticos, a descrição do sistema é realizada através de um conjunto de formulações matemáticas que podem ser solucionadas para um conjunto de parâmetros de entrada, permitindo a derivação das métricas de desempenho. Para que um modelo analítico possa ser tratável é necessário que suas dimensões sejam pequenas, ou seja, que ele apresente uma visão do sistema em alto nível de abstração. Os modelos analíticos geralmente embutem uma menor quantidade de detalhes que os modelos de simulação. Em contrapartida, os modelos analíticos executam mais rapidamente e, quando solucionados, fornecem resultados exatos, sem a necessidade de especificação de intervalos de confiança. É importante esclarecer que o termo exatidão se refere ao fato de que os resultados são soluções de sistemas de equações, não indicando que o modelo reflete, necessariamente, o sistema real sendo representado.

Quando se utiliza modelos para representar o comportamento de desempenho de um sistema, eles são mais detalhados se utilizados para *tuning* (gerenciamento de capacidade) e mais abstratos quando se precisa realizar *sizing* (planejamento de capacidade). Além disso, modelos para *tuning* são normalmente específicos para um sistema (dependentes do sistema), enquanto que os modelos para *sizing* são mais generalistas (independentes do sistema).

2.5.6 Calibrar e Validar os Modelos

A utilização de um modelo, em substituição ao sistema original, para predição de desempenho em vários cenários de carga diferentes depende fundamentalmente da capacidade desse modelo em representar comportamentos de interesse da avaliação. Para garantir esta capacidade de representatividade do modelo é necessária a realização da sua verificação e validação.

A atividade de verificação consiste basicamente em garantir que o modelo está corretamente implementado e que faz exatamente o que o projetista queria que ele fizesse. Jain [Jain 1991] sugere algumas técnicas para essa atividade que se aplicam ao contexto deste projeto: casos de teste simplificados, teste de continuidade, teste de degeneração e teste de consistência.

A aplicação de casos de teste simplificados para verificar o modelo destina-se a construir cargas simples o suficiente para que o resultado possa ser avaliado

diretamente pelo projetista do modelo. O fato do modelo passar por essa verificação não garante que em casos de teste mais complexos ele passará. Porém, se ele não conseguir passar por esta verificação, automaticamente já garante que ele possui problemas e precisa ser ajustado (calibrado).

O teste de continuidade consiste na execução de múltiplos e sucessivos experimentos de simulação com o modelo realizando uma pequena variação em apenas um dos parâmetros de entrada do modelo. Essa variação deve gerar igualmente pequena variação nos resultados das métricas avaliadas no modelo. Qualquer variação excessiva nos resultados deve ser investigada, pois pode se tratar de erros de modelagem.

O teste de degeneração deve avaliar o comportamento do modelo nas situações extremas para verificar se ele continua representando o sistema real. O planejador de capacidade deve avaliar se a combinação de valores extremos dos parâmetros de entrada pode resultar em um comportamento anormal do modelo.

O teste de consistência verifica se o modelo se comporta de forma coerente e consistente quando os valores dos parâmetros de entrada são modificados de forma harmônica.

Finalizada a verificação e os ajustes necessários, o modelo agora precisa ser validado. A validação consiste em garantir que ele, se corretamente implementado, produz resultados que o aproximem do comportamento do sistema real. Essa consideração deve ser válida pelo menos para um conjunto de cenários pré-estabelecidos e para as métricas escolhidas.

Jain [Jain 1991] sugere que os resultados produzidos pelo modelo podem ser submetidos à validação por comparação, utilizando como fonte a intuição de especialistas, as medições realizadas no sistema real, ou ainda resultados teóricos. Dentre essas fontes, a medição do sistema é sem dúvida a mais precisa e a mais utilizada.

Segundo [Menascé e Almeida 2002], diferenças no intervalo de 10% a 30% entre os resultados obtidos através da medição do sistema real e os resultados obtidos através dos modelos são aceitáveis em processos de planejamento de capacidade.

Quando o modelo não consegue reproduzir os resultados do sistema real dentro da margem de erro considerada aceitável, é necessário realizar uma calibragem no modelo, com o intuito de refiná-lo, tornando possível a sua utilização nesse tipo de avaliação. Esse processo de calibragem muitas vezes envolve a representação dos parâmetros de entrada dos modelos de carga e desempenho através de distribuições probabilísticas mais complexas de serem modeladas, visto que as redes de Petri somente são capazes de representar distribuições exponenciais (processo markoviano). Nos *Apêndices A e B* é discutida a modelagem de processos não markovianos utilizando redes de Petri.

Algumas vezes a calibragem do modelo exige uma análise de *outliers* para os valores medidos durante a fase de obtenção dos parâmetros, ou até mesmo a retirada desses pontos de desvio do resultado das métricas obtidas.

2.5.7 Prever a Evolução da Carga de Trabalho

Uma vez que o modelo foi verificado e validado, ele está pronto para ser utilizado na predição de desempenho do sistema, com o objetivo de realizar o seu planejamento e gerenciamento de capacidade.

Para o gerenciamento de capacidade (*tuning*) é suficiente utilizar a carga de trabalho medida e representada no modelo de carga. Já para o planejamento de capacidade (*sizing*), é necessário prever a evolução da carga de trabalho com o intuito de avaliar o comportamento de desempenho do sistema sob esses cenários futuros de carga. Isso permite identificar quando o sistema atingirá possíveis pontos de saturação, ou ainda, qual a sua capacidade máxima.

As técnicas de previsão podem ser divididas em duas categorias: quantitativa e qualitativa [Menascé e Almeida 2002]. As técnicas quantitativas contam com a existência de dados históricos sobre a evolução da carga nos últimos tempos para estimar o valor futuro dos parâmetros das cargas de trabalho. As técnicas qualitativas, por sua vez, representam um processo subjetivo, baseado em critérios, intuição, opiniões especializadas, analogia histórica e conhecimento do negócio.

Uma metodologia de previsão orientada ao negócio é apresentada em [Menascé e Almeida 2002]. Nessa metodologia leva-se em consideração que o volume de atividade do negócio está relacionado às demandas de recursos do sistema.

2.5.8 Avaliar Cenários

Depois de elaborado um conjunto de cenários de carga, o próximo passo da metodologia consiste na avaliação das métricas de desempenho do sistema para cada cenário.

Os modelos desenvolvidos devem permitir uma avaliação de desempenho tanto analítica quanto via simulação. No apêndice A é apresentada uma discussão detalhada sobre qual técnica utilizar. Basicamente, o que determinará se o modelo será resolvido analiticamente ou via simulação será a dimensão do espaço de estados gerado.

É muito comum que modelos complexos levem a uma explosão do espaço de estados, fazendo com que a simulação seja o único recurso. Em um processo de simulação a questão central é quanto ao critério de parada, que normalmente está relacionado com a precisão dos resultados (nível de significância).

Se o espaço de estados puder ser gerado, a técnica analítica é utilizada. Para isso, as ferramentas de avaliação de desempenho geram uma cadeia de Markov equivalente ao modelo e realizam a computação matemática das métricas.

Uma grande vantagem da técnica analítica para realizar planejamento de capacidade é que o tempo consumido para se valorar as métricas para um cenário é relativamente pequeno, quando se comparado a técnica de simulação. Como é necessária a avaliação de vários cenários diferentes durante o processo de planejamento, isso torna o trabalho mais eficaz.

O desempenho de um sistema normalmente depende de mais de um fator ligado ao sistema ou a carga de trabalho. Uma avaliação adequada requer que os efeitos de cada fator no desempenho de um sistema sejam compreendidos de

forma isolada. Para conseguir esse isolamento é necessária a elaboração de um planejamento de experimentos, denominada de análise e desenho de experimentos (*experimental design and analysis*), que tem o objetivo de maximizar as informações obtidas com o mínimo de experimentos [Jain 1991].

Jain [Jain 1991] define que os fatores são variáveis (ou parâmetros) que afetam alguma métrica (ou variável de resposta) do sistema. Cada fator possui níveis (valores) que são utilizados para avaliar as métricas do sistema, permitindo a análise o seu impacto no desempenho do sistema.

A partir desses conceitos, pode-se definir a análise e desenho de experimentos como a especificação do número de experimentos, dos níveis para cada fator em cada experimento, além do número de repetições de cada experimento. Isso tudo é definido baseado em um conjunto de hipóteses que precisam ser confirmadas ou não pelo avaliador.

2.5.9 Analisar Resultados

Depois de concluído todos os experimentos de desempenho, o gerente de capacidade deve obter para cada cenário o seu custo. É importante nesse momento relembrar parte da definição de planejamento de capacidade:

> *"...planejamento de capacidade é o processo de prever quando os níveis futuros de carga saturarão o sistema...e determinar o modo mais econômico de adiar a saturação ou o comprometimento ao máximo"* [Seção 2.4].

Esse passo da metodologia tem o objetivo de avaliar os resultados obtidos em cada cenário e determinar quais desses cenários podem adiar ao máximo a saturação do sistema, levando em consideração o custo para implantação de cada cenário, de forma a escolher a opção mais econômica.

A comparação dos diversos cenários gera um plano de configuração e um plano de investimento [Menascé e Almeida 2002]. O plano de configuração especifica as melhorias nas plataformas de hardware e software, além de mudanças na topologia da rede e na arquitetura do *middleware*. Já o plano de investimento determina quando as mudanças são necessárias, em função da avaliação de quando o sistema irá saturar ou mesmo a partir da identificação da capacidade máxima mantendo os níveis de serviço definidos.

Uma vez finalizado todos os passos do ciclo definido pela metodologia de planejamento de capacidade apresentada neste capítulo, é necessário entender que um processo de planejamento e gerenciamento de capacidade deve ser contínuo. Sendo assim, um novo ciclo precisa ser iniciado, buscando o entendimento novamente do ambiente, para verificar possíveis mudanças, ou ainda, mudanças de percepções ou interesses de avaliação.

2.6 Considerações Finais

Este capítulo introduziu os conceitos básicos necessários ao entendimento do restante desta dissertação. Para realizar esse embasamento, o capítulo detalhou duas áreas de conhecimento distintas: sistemas distribuídos e avaliação de desempenho.

Na área de sistemas distribuídos foi abordado mais especificamente o entendimento sobre *middleware* orientado a mensagem (MOM) e sua especificação mais popular, o *Java Message Service* (JMS).

Na área de avaliação de desempenho, o foco foi para o planejamento e gerenciamento da capacidade de sistemas. Inicialmente foi adotada uma abordagem conceitual sobre a área. Em seguida, foi apresentada de forma detalhada uma metodologia baseada na construção e avaliação de modelos.

CAPÍTULO

3

JMSCapacity: Processo

"Não planeje a capacidade de uma ponte contando o número de pessoas que atravessam atualmente o rio nadando."

Autor desconhecido.

O JMSCapacity é um *toolkit* para auxiliar no planejamento de capacidade de *JMS Providers*, e é composto por três elementos básicos: processo, biblioteca de componentes e ferramentas. O processo descreve como realizar o planejamento de capacidade de MOM, a biblioteca auxilia na atividade de modelagem do sistema e da carga de trabalho, e o conjunto de ferramentas suporta algumas das atividades do processo.

Este capítulo tem o objetivo de apresentar o primeiro elemento do *toolkit*. Para tanto, inicialmente é apresentada uma visão geral do processo, mostrando sua origem e a formalização realizada neste trabalho. Em seguida, cada atividade do processo é detalhada e instanciada no domínio de integração de aplicações. Ao final, é apresentado um diagrama geral do processo com todas as atividades, papéis e artefatos.

3.1 Visão Geral do Processo

Um processo é um conjunto de atividades que devem ser realizadas para se atingir um resultado ou objetivo. O processo apresentado neste *toolkit* trata-se de uma sistematizatização e formalização do processo informal presente na metodologia de planejamento de capacidade apresentado por [Menascé et al. 2004] e discutida no Capítulo 2. A escolha dessa metodologia está baseada no fato que ela contempla a construção de modelos na representação do sistema real (modelos de desempenho e de carga de trabalho) tanto para resolução

analítica quanto para utilização em simulação, facilitando a avaliação de múltiplos cenários distintos. Além disso, ela foi definida para a representação de serviços na Web, o que de certa forma possui semelhança com o fato de um sistema de *middleware* fornecer uma coleção de serviços para o desenvolvimento de sistemas distribuídos. Outro aspecto relevante para a adoção dessa metodologia é que os serviços na Web em conjunto com as plataformas de MOM compõem a nova geração de tecnologia para integração de aplicações corporativas, o *Enterprise Service Bus* (ESB).

A sistematização e formalização do processo se inicia com uma série de adaptações relacionadas a sequência e ao escopo das atividades. Em seguida, é adotada uma nova notação que possibilita a representação de elementos fundamentais para a formalização de um processo, como definição de responsabilidades sobre a execução das atividades, entradas e saídas formalmente estabelecidas e acordadas, além de um fluxo de trabalho claro e não ambíguo. Por fim, este trabalho realiza a sistematização das atividades aplicadas no domínio de integração de aplicações, possibilitando que parte das definições aqui realizadas possam ser utilizadas quando da realização do planejamento de capacidade de um cenário real.

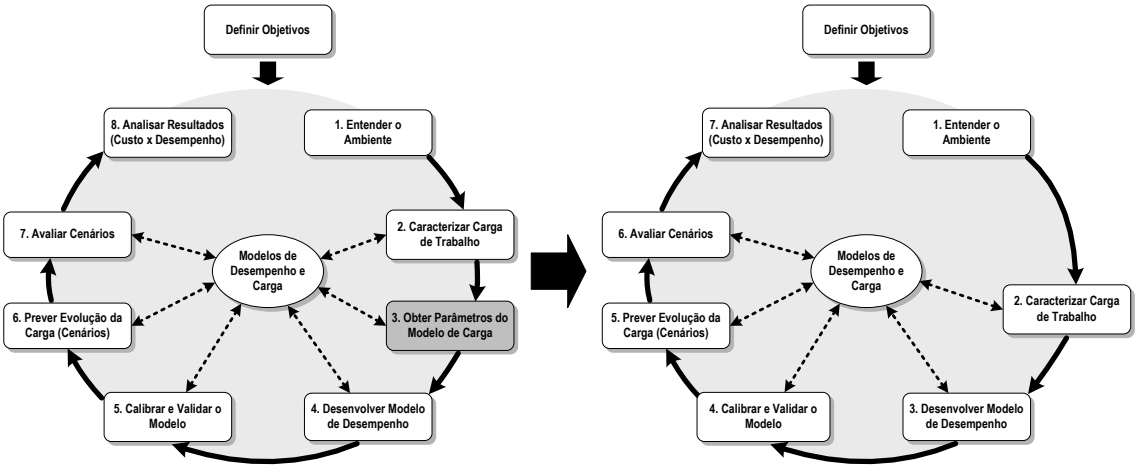


Figura 3-1. Adaptação 1: composição dos passos 2 e 3.

A primeira adaptação no processo definido por [Menascé et al. 2004] é representada na Figura 3-1. Os passos dois e três (*Caracterizar Carga de Trabalho e Obter Parâmetros do Modelo de Carga*, respectivamente) são aglutinados em um único passo. A motivação para essa modificação é que ambos formavam atividades complementares, que objetivavam ao final do passo três a formulação de um modelo de carga de trabalho. Inclusive, os autores definem um conjunto de atividades para construção do modelo de carga e dividem essas atividades em dois passos distintos do seu proceso. Com isso, o novo passo do processo passa a ter o objetivo de elaborar e entregar um importante artefato de saída, o modelo da carga de trabalho.

Outra intervenção proposta (ver Figura 3-2) é a representação explicítica de uma atividade importante que é a parametrização do modelo de desempenho do sistema. Essa atividade estava contemplada dentro do passo *Desenvolver Modelo de Desempenho*, e com a modificação passa a compor uma atividade do passo *Parametrizar, Calibrar e Validar o Modelo*. No processo original, os autores deixavam explícita a parametrização do modelo de carga de trabalho e

implícita a parametrização do modelo de desempenho. A justificativa dessa modificação deve-se ao fato que essa atividade não é passível de uma automatização, enquanto que o desenvolvimento do modelo de desempenho pode ser sistematizado ao ponto de ser gerado de forma automática. É importante ressaltar que este trabalho realiza a sistematização da atividade, mas não se propõe a realizar a geração automática de modelos. A fase seguinte fica responsável por um refinamento do modelo gerado, fazendo a inserção dos parâmetros, a calibragem do modelo, assim como a sua validação.

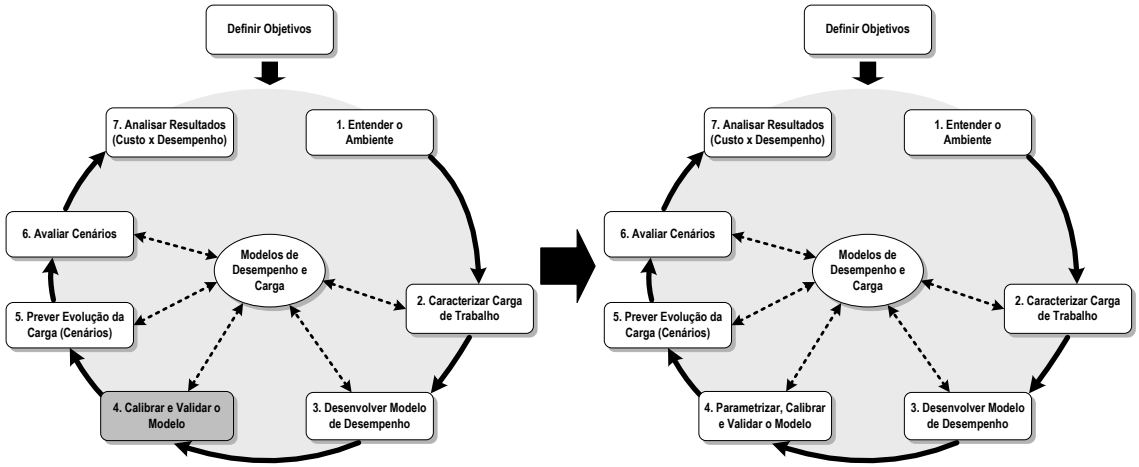


Figura 3-2. Adaptação 2: inserção da atividade de Parametrização no passo 4.

A terceira intervenção proposta no processo original modifica a ordem dos passos definidos (ver Figura 3-3). O passo *Prever Evolução da Carga* é uma atividade normalmente desempenhada por um gerente de capacidade, utilizando como entrada informações sobre a arquitetura do serviço de integração e o modelo de carga, artefatos gerados pelos passos 1 e 2 respectivamente. Esse passo pode ser antecipado e executado em paralelo com o desenvolvimento do modelo de desempenho do sistema (passo 3) e a parametrização, calibragem e validação do modelo (passo 4). A inovação proposta nesse ponto foi a quebra do aspecto seqüencial dos passos, pois no detalhamento do processo mais aditante ficará claro a distribuição de papéis diferentes para as atividades envolvidas nesses passos, justificando esse paralelismo de ações.

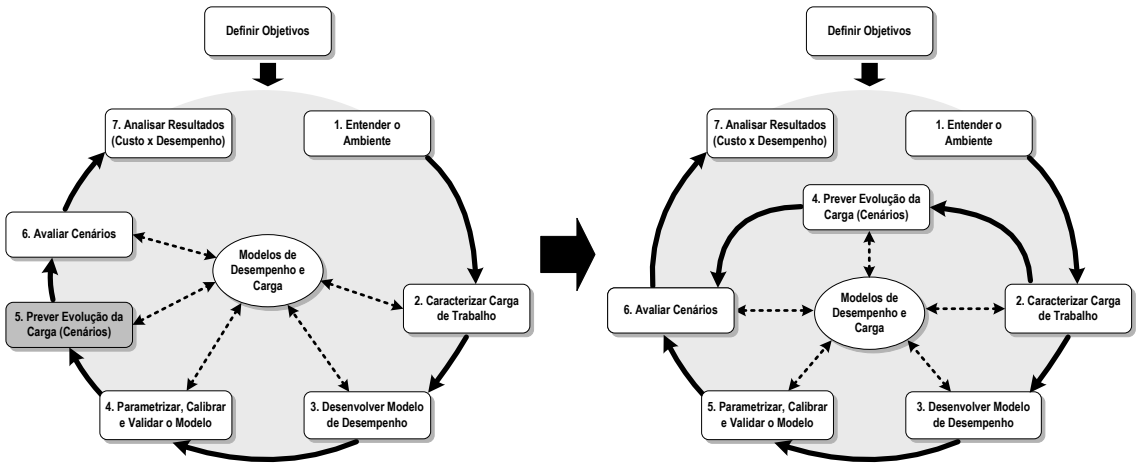


Figura 3-3. Adaptação 3: modificação na ordem de execução dos passos.

Depois de realizadas essas três intervenções no processo em si, este trabalho propõe a formalização do processo através da utilização da notação definida por [Kruchten 1998], e apresentada no Apêndice D desta dissertação.

3.1.1 Papel ou Função

No processo definido por esta dissertação estão identificados os seguintes papéis:

- *Integrador de Sistemas*: um indivíduo atuando como um integrador é um especialista que especifica e detalha a integração entre as aplicações corporativas, explicitando as principais interações envolvendo a troca de mensagens entre entidades comunicantes, e os níveis de serviço desejados e acordados para essas interações. Em resumo, ele é responsável por determinar os objetivos para o planejamento de capacidade.
- *Gerente de Capacidade*: um indivíduo atuando como gerente de capacidade possui domínio e conhecimento sobre o ambiente tecnológico onde estão inseridas as aplicações a serem integradas. Ele é responsável em definir as métricas para avaliação dos níveis de serviço definido pelo *Integrador de Sistemas*, além de possuir a habilidade de realizar uma previsão da evolução da carga de trabalho, com o intuito de planejar a capacidade necessária para atender.
- *Avaliador*: um indivíduo atuando como avaliador domina as técnicas de avaliação, permitindo utilizar os modelos em redes de Petri para valorar as métricas definidas pelo *Gerente de Capacidade*. As atividades desempenhadas são iterativas e geram informações que serão interpretadas pelo *Gerente de Capacidade* para elaboração do plano de capacidade.
- *Projetista*: um indíviduo atuando como projetista é capaz de construir componentes que representam parte do sistema através de modelos em redes de Petri, criando uma representação formal, que poderá ser combinada e refinada pelo *Avaliador*. O indivíduo para atuar neste papel deve possuir um bom entedimento sobre redes de Petri estocásticas, avaliação de desempenho, além de um bom entendimento do funcionamento das plataformas de MOM e das técnicas de integração.

É importante ressaltar que os papéis não identificam indivíduos, mas o seu comportamento e suas responsabilidades quando estão desempenhando cada papel. Apesar disto é comum expressar que “um *Avaliador* realizou uma atividade Y”, quando na verdade o correto seria dizer “um indíviduo X desempenhando o papel de *Avaliador* realizou uma atividade Y”.

3.1.2 Atividade

São exemplos de atividades do processo deste *toolkit*:

- Definir Objetivos
- Caracterizar a Carga de Trabalho
- Desenvolver Modelo de Desempenho

- Avaliar Cenários

Ao longo deste capítulo as atividades definidas pelo processo são apresentadas, indicando seus objetivos, artefatos gerados ou manipulados, assim como a que papel a atividade está associada.

3.1.3 Artefato

Ao longo desse capítulo com a descrição detalhada das atividades, será possível entender quais os artefatos são consultados por cada papel durante a execução do processo.

A Figura 3-4 apresenta as responsabilidades do *Integrador de Sistemas*. Ele produz dois artefatos: *Plano de Integração* e *Service Level Objectives*. Esses artefatos documentam o serviço de integração e os seus requisitos de desempenho de forma clara e objetiva em função de indicadores mensuráveis.

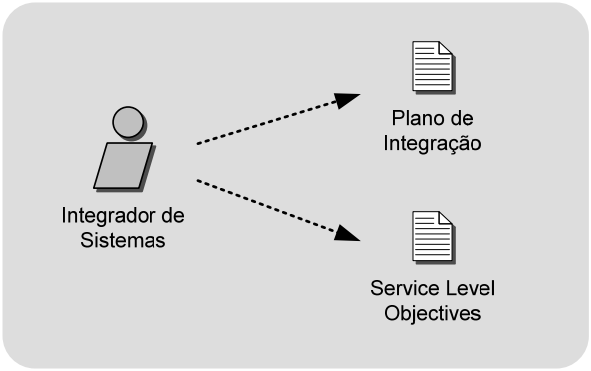


Figura 3-4. JMSCapacity: responsabilidades do Integrador de Sistemas.

O *Gerente de Capacidade* é responsável pela geração de três artefatos importantes no processo de planejamento de capacidade (ver Figura 3-5): *Arquitetura do Serviço de Integração*, *Cenários de Carga* e *Plano de Capacidade*. É importante ressaltar que o *Gerente de Capacidade* tem a responsabilidade de produzir o principal artefato do processo de planejamento de capacidade, na verdade trata-se do produto final a ser entregue depois de concluído o processo, o *Plano de Capacidade*. Os *Cenários de Carga* documentam uma projeção de evolução da carga de trabalho, enquanto a *Arquitetura do Serviço de Integração* identifica os recursos que precisam ter suas capacidades monitoradas pelo planejamento.

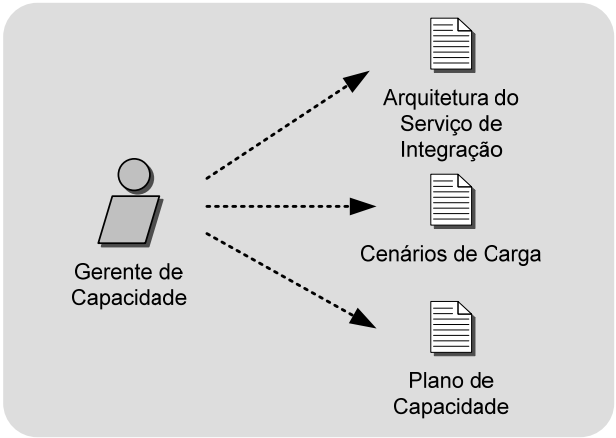


Figura 3-5. JMSCapacity: responsabilidades do Gerente de Capacidade.

O *Avaliador* produz artefatos relacionados com a modelagem e avaliação do sistema de entrega de mensagens. Para realizar a avaliação é necessária a produção de vários modelos, e sua posterior utilização na valoração das métricas estabelecidas, definindo a capacidade máxima do sistema, e com isso conseguir avaliar se os objetivos de desempenho do serviço de integração serão atendidos. A Figura 3-6 apresenta a relação dos artefatos sob responsabilidade do *Avaliador*.

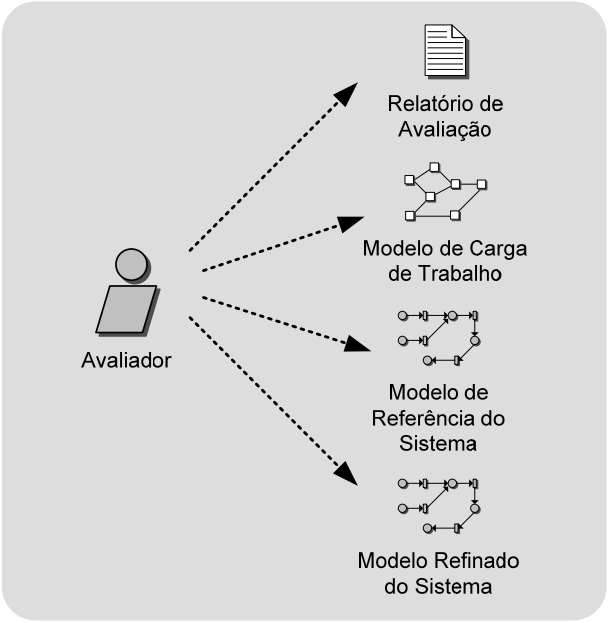


Figura 3-6. JMSCapacity: responsabilidades do Avaliador.

O *Projetista* é responsável pela construção dos componentes da biblioteca, que é utilizada pelo *Avaliador* para desenvolver os modelos de desempenho do sistema. Este trabalho atua nesse papel formando uma biblioteca de componentes tanto de carga quanto de desempenho. A biblioteca é detalhada no Capítulo 4, onde ficará mais claro o papel e a importância do *Projetista* no processo. A Figura 3-7 apresenta os artefatos produzidos pelo *Projetista*.

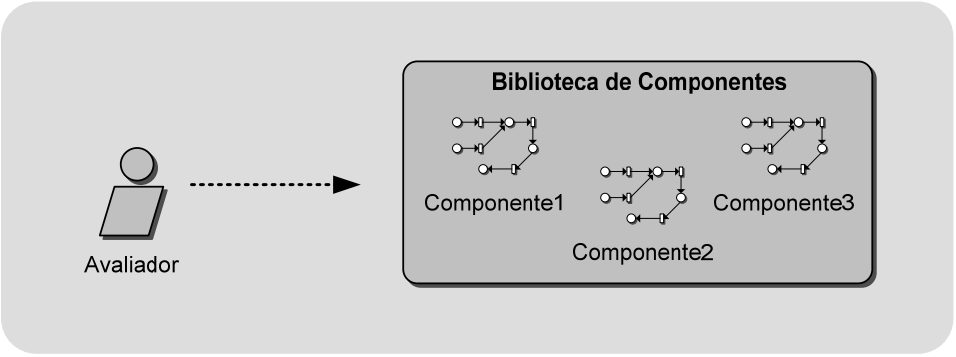


Figura 3-7. JMSCapacity: responsabilidades do Projetista.

Durante o detalhamento das atividades neste capítulo mais informações sob os artefatos são apresentadas, relacionando-os com as atividades responsáveis em criá-los, mantê-los ou consultá-los.

3.1.4 Fluxo de Trabalho (*workflow*)

O fluxo de trabalho do processo do JMSCapacity descreve a seqüência de atividades para geração do plano de capacidade do sistema de entrega de mensagens. A Figura 3-8 apresenta o fluxo de atividades do processo, descrevendo o papel responsável pela execução de cada atividade. O fluxo possui o mesmo seqüenciamento de atividades do processo adaptado da metodologia de [Menascé et al. 2004], e já apresentado na Figura 3-3. As atividades *Prever Evolução da Carga* e *Desenvolver Modelo de Desempenho* são paralelas e disparadas simultaneamente. Já a atividade *Avaliar Cenários* depende do término das atividades *Prever Evolução da Carga* e *Parametrizar, Validar e Calibrar o Modelo*.

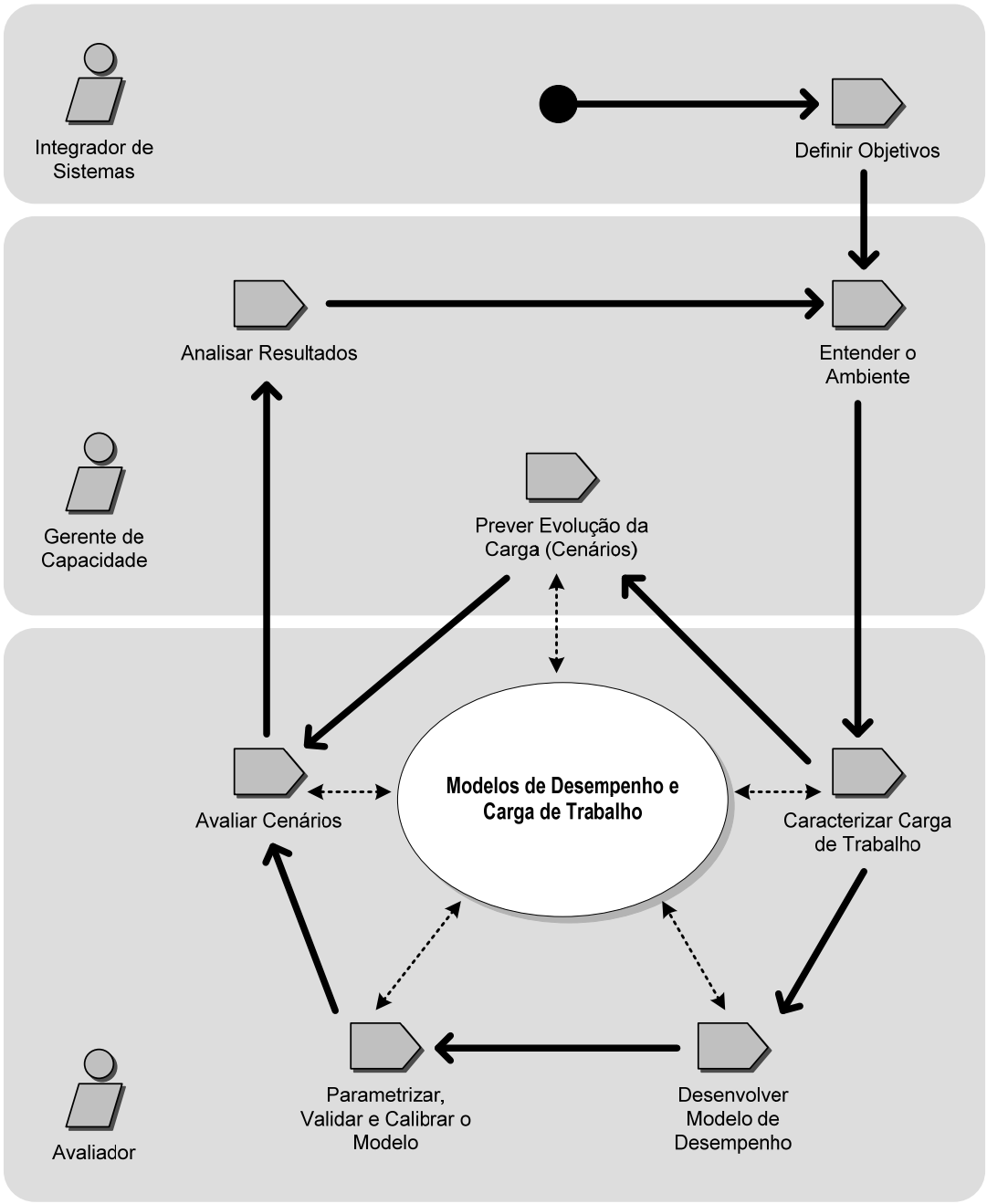


Figura 3-8. JMSCapacity: papéis, atividades e fluxo de trabalho.

Pode-se notar que o fluxo define um ciclo contínuo, pois o processo de planejamento de capacidade deve ser realizado de forma contínua nas organizações, refinando cada vez mais o plano de capacidade, com o intuito de prever mudanças nos cenários ou mesmo nos objetivos de desempenho do serviço de integração.

Outro aspecto a ressaltar é que o papel do *Projetista* não foi inserido nesse fluxo pois o JMSCapacity faz este papel, gerando a biblioteca de componentes que é utilizada pelo *Avaliador*. No fluxo completo apresentado ao final desse capítulo esse papel é incluído com a atividade desenvolvida pelo *toolkit*. O motivo dessa exclusão é para tornar claro que o usuário do JMSCapacity não precisa assumir o papel de *Projetista*.

O restante desse capítulo apresenta o detalhamento de cada uma das atividades do processo, já instanciando para o contexto do serviço de integração de aplicações corporativas utilizando um sistema de troca de mensagens baseado em uma plataforma de MOM compatível com a especificação JMS.

3.2 Definir Objetivos

Esta atividade desenvolvida pelo *Integrador de Sistemas* é responsável pela identificação e definição dos objetivos de desempenho do serviço de integração. Para isso ela está estruturada em dois passos: definição do *Service Level Objectives* (SLO); documentação e caracterização do serviço de integração.

Para que seja possível definir o SLO do serviço de integração, é necessário definir indicadores, que são métricas ou uma combinação delas que fornecem uma compreensão do serviço. Considerando que um sistema de entrega de mensagens (MOM) é responsável direto pelo encaminhamento das mensagens destinadas para implementação do serviço de integração, é preciso definir métricas para monitorar o sistema de entrega de mensagens, de forma a utilizá-las na construção do SLO do serviço de integração.

3.2.1 Métricas dos Sistemas de MOM

As métricas do sistema de entrega de mensagens quando combinadas a cenários podem ajudar a determinar o ponto de saturação do sistema de entrega, assim como a capacidade de entrega do sistema atendendo a um SLO.

Algumas das métricas definidas nesta seção são utilizadas para definição do SLO do serviço de integração, enquanto outras são utilizadas para monitorar a capacidade do sistema de entrega de mensagens com o objetivo de evitar a sua saturação.

Para cada métrica são definidos os seguintes pontos: descrição (o que está sendo medido); aplicabilidade (porque medir); categoria (produtividade, resposta, utilização); unidade de medida em que ela está sendo representada; forma de cálculo (como e quando medir); e ponto de medição (onde medir) identificando o componente do sistema onde é obtida a informação.

Taxa de Entrega de Mensagens (*Message Delivery Rate*)

Esta métrica de produtividade representa o número de mensagens entregues pelo MOM por unidade de tempo, possuindo como unidade de medida *mps*

(mensagens por segundo). A sua aplicabilidade é avaliar o desempenho global do sistema de entrega. No contexto de planejamento de capacidade, esse valor pode representar a capacidade do sistema, desde que o tamanho das mensagens seja igual.

O ponto de medição dessa métrica é nos consumidores e para efetuar o cálculo é necessário contar as mensagens entregues pelo MOM a todos os consumidores durante o teste e dividi-la pelo tempo total do teste.

Vazão do Sistema (*System Throughput*)

Esta outra métrica de produtividade representa a quantidade de dados entregues pelo MOM por unidade de tempo, possuindo como unidade de medida *KBps* (kilo bytes por segundo). Ela pode ser utilizada para avaliar o desempenho global do sistema de entrega. No contexto de planejamento de capacidade, esse valor pode representar a capacidade do sistema, para qualquer conjunto de mensagens trafegadas (independente do tamanho).

Esta métrica pode ser obtida multiplicando a métrica Taxa de Entrega de Mensagens, pelo tamanho da mensagem (supondo uniformidade nas mensagens enviadas). Se não houver uniformidade, esse cálculo deve ser realizado em tempo de execução sumarizando a quantidade de dados recebida por todos os consumidores durante o teste e dividindo esse total pelo tempo de duração do teste. Por esse motivo o ponto de medição também é nos consumidores.

Latência (*Latency*)

Esta métrica de resposta representa o tempo médio gasto pelo MOM para entrega de uma mensagem, ou seja, o atraso médio inserido pelo sistema. Esse tempo representa a soma do tempo gasto para entrega da mensagem pelo produtor, o tempo de processamento e roteamento da mensagem dentro do MOM, e o tempo de entrega da mensagem ao consumidor.

Esta métrica normalmente avalia o desempenho individual do sistema de entrega. No contexto de integração de aplicações, ela é utilizada para especificar o SLO do serviço de integração. A unidade de medida desta métrica é *s* (segundos).

Para efetuar o cálculo da métrica, o produtor insere um *timestamp* ao produzir a mensagem. Quando essa mensagem chega ao consumidor, o *timestamp* é comparado com o valor atual do relógio. A diferença entre ambos é a latência (ou retardo) inserida na entrega da mensagem. Para que esse cálculo seja preciso faz-se necessário a sincronização dos relógios nos sistemas operacionais que estiverem hospedando o produtor e o consumidor.

Tamanho da Fila (*Destination Size)*

Esta métrica de utilização representa o número médio de mensagens em um *destination* (ver Seção 2.2.1) específico em um dado instante de tempo, ou seja, o tamanho da fila ou tópico. Essa métrica varia com o tempo. No caso do *destination* possuir o estilo de mensagens *Pub/Sub*, esta métrica representa a quantidade de mensagens em todas as assinaturas (*subscriptions*) desse *destination*.

Essa métrica não é utilizada para avaliar o desempenho do sistema, nem globalmente, nem individualmente. Ela serve para monitorar o ponto de saturação do sistema, pois o *destination* (fila ou tópico) possui um comportamento de crescimento exponencial quando a carga de trabalho submetida ao sistema supera a sua capacidade. Logo, a monitoração dessa métrica permite identificar o momento em que a carga de trabalho atinge a capacidade do sistema, podendo ser útil para determinação da capacidade adequada, dentro do contexto de planejamento de capacidade.

Para se obter a valoração desta métrica é necessário monitorar o MOM via ferramenta específica disponibilizada pelo *toolkit* (JMXMonitor), ou através de mecanismos próprios de cada MOM. A unidade de medida é a quantidade de mensagens (msg).

Utilização de CPU (*CPU Utilization)*

Esta outra métrica de utilização representa o percentual de CPU utilizado pelo MOM. Esse valor não deve superar 70% para evitar a saturação do servidor, pois pode existir carga adicional submetida ao servidor por outros componentes ou até mesmo por outros serviços que estejam compartilhando esse servidor. Além disso, como a margem de erro do modelo para efeito de validação é de até 30%, contar com esse recurso pode ser um equívoco no planejamento de capacidade.

Essa métrica também não é utilizada para avaliar o desempenho do sistema, nem globalmente, nem individualmente. Ela serve para determinar o ponto de saturação do sistema, ou ainda o nível de comprometimento do seu principal recurso.

Monitorar o gasto de CPU do processo responsável pela execução do MOM é a forma mais simples de obter valores para esta métrica. No caso do *middleware* escrito em Java deve-se monitorar a JVM (java.exe). O ideal é monitorar essa métrica atavés de ferramentas disponibilizadas pelos sistemas operacionais.

3.2.2 Definindo o SLO

O primeiro artefato gerado pelo processo é um SLO para o serviço de integração, caracterizando os objetivos que guiaram o planejamento e gerenciamento de capacidade.

Em um contexto que utiliza um sistema de entrega de mensagens para a integração de aplicações corporativas, definir o SLO do serviço de integração depende necessariamente da definição de requisitos de desempenho, segurança e confiabilidade para o sistema de MOM.

No caso de objetivos de desempenho, o usuário (produtor) normalmente deseja que a mensagem entregue ao MOM seja encaminhada ao consumidor com a menor *Latência* possível. Um acordo nos valores limites para essa métrica é o principal componente de um SLO para um serviço de integração. A especificação JMS possui um mecanismo de prioridade capaz de melhorar essa *Latência,* pois mensagens podem ser submetidas com prioridade de entrega diferenciada, fazendo com que o MOM protele as demais em função destas.

Porém, não basta entregar a mensagem rapidamente, pois se tratando de tráfego de informação, o requisito de segurança deve ser levado em consideração. Para se obter uma entrega segura, precisa-se garantir a disponibilidade do sistema além da confidencialidade (sigilo) e da integridade da mensagem.

A especificação JMS não determina aspectos de confidencialidade, deixando a cargo das implementações atenderem a esse requisito. Já a disponibilidade do sistema, normalmente está associada à capacidade utilizada e tende a decrescer com a saturação do sistema, ou até mesmo com a utilização de um mecanismo de controle de admissão que possibilite a manutenção de uma latência máxima. Com a realização do gerenciamento da capacidade do sistema, pode-se impedir a sua saturação e com isso garantir um nível elevado de disponibilidade.

A integridade da mensagem é tratada pelo JMS, pois a especificação determina que as mensagens não possam ser manipuladas por nenhum intermediário (MOM, ou algum consumidor). Apenas o produtor da mensagem, é capaz de alterar suas propriedades. Uma vez entregue ao MOM, a mensagem não consegue ser mais alterada, mantendo a sua integridade durante todo o percurso até ser entregue ao consumidor.

Uma vez atendido aos requisitos de segurança (disponibilidade, confidencialidade e integridade), precisa-se ainda garantir que o sistema de entrega seja confiável. A confiabilidade de um MOM JMS é alcançada através da parametrização do *middleware* quanto a dois aspectos importantes do sistema: o modo de entrega e a integridade transacional da comunicação (ver Seções 2.1.3 e 2.1.4). A combinação dessas duas características normalmente é chamado de QoS do sistema de entrega. Isso leva a definição de três tipos de QoS: não-persistente e não transacional (NPNT), persistente e não-transacional (PNT) e persistente e transacional (PT). A persistência é que garante a tolerância a falhas, enquanto o esquema transacional garante a atomicidade no processamento de entrega de mensagens.

A confiabilidade de um sistema é proporcional a sua capacidade de ser tolerante a falhas. Nesse contexto, ser confiável é garantir que a mensagem recebida pelo MOM seja sempre entregue ao consumidor, desde que esse esteja disponível dentro do prazo máximo estabelecido pelo produtor. O JMS possui um mecanismo de expiração para evitar que mensagens voláteis sejam recebidas por consumidores fora de prazo. Esse mecanismo determina que sejam descartadas as mensagens recebidas fora do prazo de tempo estabelecido pelo produtor.

A partir do que foi apresentado, a definição do SLO para um serviço de integração compreende a especificação de uma latência máxima para entrega das mensagens, um nível de QoS a ser utilizado, assim como algumas outras características relacionadas à prioridade e expiração das mensagens. Como exemplo, pode-se definir um SLO por um conjunto de sentenças como:

- A *Latência* inserida pelo MOM para entrega de uma mensagem deve ser inferior a 5s;
- A mensagem deve ser entregue de forma confiável ao consumidor, sem a possibilidade de repetição, nem de falha. Essa definição leva a definição do QoS PT;

- Nenhuma mensagem pode ser entregue com mais de 1 minuto de *Latência*, devendo ter o TTL (Time To Live, ou tempo de expiração) definido adequadamente;
- O MOM precisa ter um nível de disponibilidade de 99,999%. Para isso, mecanismos tolerantes a falhas devem ser embutidos no sistema, além da necessidade de gerenciamento da sua capacidade com o intuito de evitar a saturação dos recursos;
- Algumas mensagens necessitam de prioridade na entrega, logo é necessária a definição de um serviço de entrega expresso para esse tipo de mensagem.

Esse SLO pode fazer sentido, por exemplo, em um ambiente de integração entre aplicações interativas, por isso o tempo definido para a *Latência*. O aspecto de confidencialidade (segurança) nesse caso seria mantido pela aplicação e não pelo MOM.

3.2.3 Definindo o Plano de Integração

Para finalizar a atividade de *Definir Objetivos* é necessário que o *Integrador de Sistemas* documente o serviço de integração com objetivo de repassar para o *Gerente de Capacidade* as infomações utilizadas no entendimento do ambiente. O artefato gerado com essa documentação é denominado *Plano de Integração*.

O *Plano de Integração* deve conter informações sobre as entidades que participam da integração (ex. aplicações ou componentes), os eventos que motivam a integração, as interações entre as entidades a partir de cada evento, além das mensagens trocadas.

O JMSCapacity sugere que esse artefato contenha uma diagrama de implantação (definido pelo diagrama UML de implantação) mostrando a disposição física das entidades comunicantes (ver Figura 3-9). É importante entender onde estão as entidades do ponto de vista de distribuição física, pois isso determinará os recursos computacionais envolvidos (ex. rede). As entidades são componentes ou aplicações que precisam interagir durante a integração. Os nós representam equipamentos onde essas entidades estão executando (ex. servidores). Além da distribuição, esse diagrama identifica quais as entidades que se comunicam.

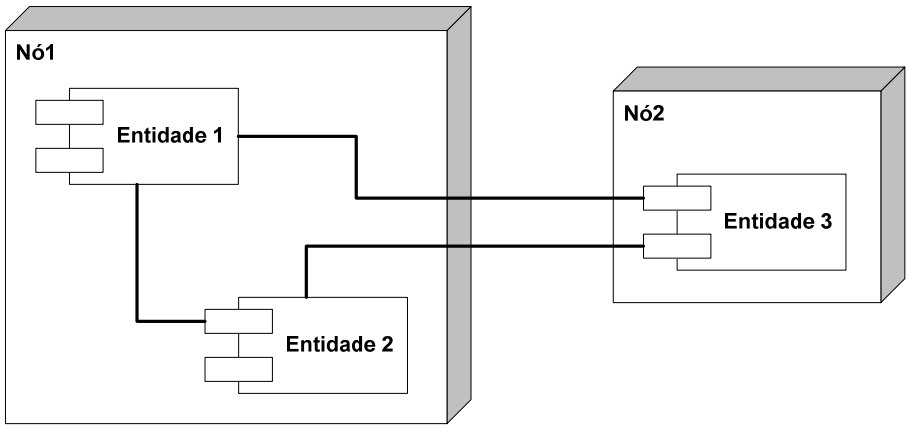


Figura 3-9. Diagrama de Implantação: entidades comunicantes.

Para documentar os eventos, as interações e as mensagens, o JMSCapacity utiliza o diagrama UML de seqüência (ver Figura 3-10), conforme foi adotado

em [Sachs et al. 2007a] para documentar as interações do benchmark SPECjms2007. Cada diagrama de seqüência inicia com um evento gerador, que normalmente pode ser mapeado como um evento do negócio (ex. um pedido de compra gera a Mensagem1) e documenta todas as mensagens trocadas pelas entidades comunicantes.

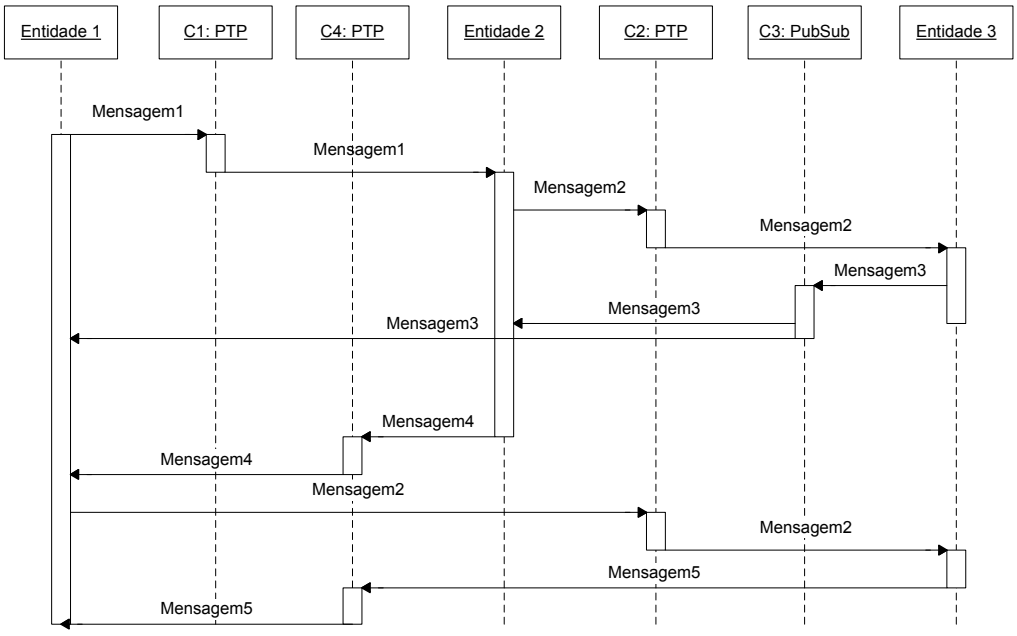


Figura 3-10. Diagrama de seqüência: mensagens e canais.

O *Plano de Integração* permite ao *Gerente de Capaciadade* mapear os recursos computacionais envolvidos com a implementação do serviço de integração.

3.3 Entender o Ambiente

Uma vez definido os objetivos de desempenho para o serviço de integração, esta atividade desenvolvida pelo *Gerente de Capacidade* é responsável pelo entendimento do ambiente tecnológico que suporta sua implementação, compreendendo as características do serviço como um todo, identificando os recursos e mapeando os parâmetros configuráveis que podem afetar o seu desempenho.

Os passos necessários para realizar esta atividade incluem: mapear e identificar os principais recursos na arquitetura que podem afetar o desempenho do serviço; definir as métricas para avaliação da capacidade de cada recurso, com o objetivo de auxiliar na identificação do ponto de saturação do serviço; definir os parâmetros configuráveis para cada camada que podem afetar os recursos identificados; compreender os padrões de uso do serviço, mapeando os parâmetros que ajudam a descrever a carga de trabalho; formular hipóteses sobre o impacto desses parâmetros no desempenho do serviço; e definir modelos de cenários para estudo (com seus fatores), em função da avaliação dos padrões de uso.

O artefato gerado nesta atividade denomina-se *Arquitetura do Serviço de Integração* e é composto dos seguintes elementos:

- Identificação dos recursos em cada camada do sistema responsável pelo serviço de integração;

- Métricas para avaliação da capacidade dos recursos;
- Identificação de parâmetros de desempenho por camada;
- Hipóteses sobre desempenho; e
- Modelos de cenários (com definição de quais parâmetros serão fatores para avaliar cada cenário).

O *Gerente de Capacidade* precisa receber o SLO e o *Plano de Integração* concluídos para realizar esta atividade. Ao final, a arquitetura do serviço de integração está documentada, fornecendo entendimento pleno sobre o ambiente tecnológico envolvido

3.3.1 Mapeando a Arquitetura

No contexto desta dissertação o serviço de integração é implementado por um sistema de entrega de mensagens, e sua arquitetura pode ser descrita por este conjunto de camadas: a plataforma operacional, o MOM e o modelo de comunicação da aplicação. Além dos recursos computacionais, associados a cada uma dessas camadas, identifica-se parâmetros importantes, que caracterizam o serviço e precisam ser analisados e levados em consideração durante o planejamento e gerenciamento de capacidade.

A camada da plataforma operacional suporta um ambiente de MOM, sendo composta por servidores interligados por uma rede de comunicação. Esses servidores no contexto JMS operam os *JMS Providers*, que recebem as mensagens de *JMS Producers* (produtores) e as encaminha para os *JMS Consumers* (consumidores). Podem existir vários *JMS Providers* em uma implementação típica de um sistema de entrega de mensagens, além de servidores de Banco de Dados (SGBD) para persistir as mensagens, monitores de transação, e os elementos de rede (*switches*, roteadores).

Os principais recursos computacionais dessa camada são: CPU, memória, sistema de armazenamento (disco) e rede. Destes, os mais críticos e que precisam ser monitorados são a CPU e o disco.

Nessa camada, podem-se destacar os seguintes parâmetros:

- Quantidade de servidores;
- Quantidade de estações cliente;
- Especificação da CPU e memória de cada servidor;
- Topologia da rede de comunicação; e
- Vazão da rede de comunicação (*network throughput*) por enlace.

Na camada de *middleware*, o administrador do ambiente pode configurar alguns parâmetros para melhorar o desempenho (*performance tuning*). Os fabricantes de MOM geralmente disponibilizam manuais ou guias de como realizar esses ajustes no desempenho de seus sistemas [Sonic 2007] [JBoss 2005] [IBM 2005]. Entre os principais parâmetros normalmente disponíveis podem-se destacar:

- Parâmetros da máquina virtual Java (JVM. Esses parâmetros definem entre outros aspectos o tipo da JVM (*client*, *server*), as configurações da memória *heap* e da pilha;
- Tamanho dos *buffers* para os *destinations* (tamanho da fila ou tópico);

- Tamanho da *cache* de disco (*memory cache*), acelerando as leituras e escritas no disco durante o encaminhamento das mensagens;
- Freqüência de limpeza dos *destinations*, onde o MOM verifica as mensagens expiradas que precisam ser retiradas;
- Quantidade máxima de *threads* para atender requisições (*thread pool size*).

Dos vários recursos da camada de *middleware* incluindo *pool* de *threads*, *pool* de instâncias, *pool* de conexões, gestores de transação e concorrência, o recurso *Destination* (fila/tópico) se destaca devido ao nível de concorrência no seu acesso. Para monitorar esse recurso, a métrica de utilização escolhida mede o número de mensagens armazenadas aguardando encaminhamento. Essa métrica indica o comportamento de equilíbrio do sistema, e pode mostrar facilmente uma situação de saturação.

Por fim, para concluir o mapeamento da arquitetura do sistema de entrega de mensagens, é necessário estudar a camada de comunicação da aplicação. Para documentar a arquitetura de comunicação das aplicações é necessário identificar o valor dos parâmetros para cada uma das interações que geram comunicação via MOM.

No contexto de um MOM JMS a arquitetura de comunicação de uma aplicação pode ser descrita pelos seguintes parâmetros, associados a cada interação:

- Quantidade e distribuição de produtores;
- Quantidade e distribuição de consumidores;
- Quantidade de *destinations* (filas ou tópicos) utilizados;
- Estilo das mensagens (*Pub/Sub*, PTP);
- Tamanho e tipo das mensagens;
- Confiabilidade do canal (integridade transacional e modo de entrega);
- Método de recepção da mensagem (síncrono ou assíncrono);
- Modelo de Assinatura (*subscription*); e
- Modelo de reconhecimento das mensagens.

3.3.2 Compreendendo os Padrões de Uso

Para estabelecer cenários de carga adequados durante um processo de planejamento de capacidade é importante compreender os padrões de uso impostos pela aplicação. No contexto de MOM, esses padrões identificam o comportamento de tráfego de mensagens ao longo do tempo, permitindo identificar situações de pico de utilização.

Para a identificação dos padrões de uso de um MOM JMS é preciso estudar o comportamento dos produtores e consumidores de mensagens ao longo do tempo. O comportamento dos produtores pode ser modelado a partir do seguinte parâmetro:

- *Taxa de Geração de Mensagens por Produtor ou Tempo médio entre mensagens*, que caracteriza como o volume de mensagens submetidas ao MOM está distribuído no tempo. Esse parâmetro pode assumir uma distribuição probabilística conhecida (ex. Poisson, Normal), ou empírica.

De forma análoga, o comportamento dos consumidores pode ser descrito pelos seguintes parâmetros:

- *Método de recepção das mensagens*, que informa se o consumidor procura o MOM para receber as mensagens (recepção síncrona, método *polling*) ou se ele se acopla ao serviço de notificação do MOM através de um *Listener* (recepção assíncrona, método *push*), e aguarda de forma passiva a recepção das mensagens;
- *Tempo de pensar (think time), que é o* tempo compreendido entre a recepção de uma mensagem e a condição de pronto para buscar (síncrona) ou receber (assíncrona) a próxima mensagem; e
- *Atraso no processamento da mensagem* (*process delay*), que caracteriza o tempo gasto pelo consumidor para processar a mensagem recebida.

3.3.3 Identificando o Impacto dos Parâmetros no Desempenho

Uma vez compreendido a arquitetura e o padrão de uso de um sistema de entrega de mensagens, é necessário compreender o possível impacto desses parâmetros no desempenho do sistema.

Esta subseção elabora hipóteses sobre o impacto de alguns parâmetros no desempenho do sistema, analisando, para isso, o comportamento das métricas para cada um deles. Nem todos os parâmetros são analisados, e quando analisados nem todas as métricas são levadas em consideração. As motivações dessa restrição são:

- O impacto de alguns parâmetros é facilmente observável, e.g., o parâmetro *Vazão da Rede* impactando a métrica *Vazão do Sistema*;
- O impacto de alguns parâmetros em algumas métricas é dependente do cenário e não pode ser generalizado;
- O impacto de alguns parâmetros em algumas métricas é difícil de ser previsto sem medições e testes precisos do sistema.
- Alguns parâmetros são específicos para cada implementação de um MOM JMS, cabendo a cada fabricante a análise do impacto deles no desempenho dos seus sistemas; e
- Algumas hipóteses são difíceis de serem comprovadas, levando a discussões improdutivas.

Parâmetros Relativos à Plataforma Operacional

Quantidade de Servidores

Quando se adiciona servidores, sem a construção de um *cluster*, para encaminhar mensagens com o objetivo de aumentar a *Taxa de Entrega de Mensagens* e diminuir a *Latência*, o que está sendo feito é a construção de outro sistema de entrega. Logo, o que é feito para o primeiro deve ser repetido para o segundo. Mas, se os servidores são adicionados em *cluster* a expectativa é a modificação completa da avaliação do sistema, inclusive da capacidade máxima de entrega. O que se espera nesse caso é o aumento substancial da *Vazão do Sistema* e da *Taxa de Entrega de Mensagens*, e a diminuição da *Latência*.

Porém, a análise de cenários com essa complexidade está fora do escopo dessa dissertação.

Outra possibilidade é a adição de servidores formando uma rede estruturada de encaminhamento de mensagens. Nesse tipo de situação espera-se que haja é o aumento da *Latência* devido ao tratamento que vários *JMS Providers* (MOM) têm que dar a cada mensagem ao longo do seu trajeto do produtor até o consumidor.

Parâmetros Relativos à Configuração do MOM

Tamanho Máximo dos *Destinations*

Aumentar ou diminuir o tamanho máximo de um *destination* (fila ou tópico) impacta diretamente na capacidade máxima do sistema, pois ao se aproximar do ponto de saturação o sistema tende a enfileirar as mensagens, aumentando a *Latência*, mas garantindo que não haja perda de mensagens. Fixar o tamanho máximo dos *destinations* em valor menor pode ser uma estratégia para estabelecer um controle de admissão controlando o fluxo de mensagens de forma a garantir que não haja acúmulo excessivo de mensagens, com o objetivo de limitar o crescimento da *Latência*. Para mais detalhes sobre estratégias para estabelecimento do contro de Admissão pode ser obtidos no Apêndice A.

O principal aspecto que determina o tamanho máximo ideal para cada *destination* é o estudo dos padrões de uso dos produtores e consumidores, pois se o cenário é de muitas mensagens entregues ao MOM em curto intervalo de tempo, e na seqüência um *Think Time* grande, a tendência é que a fila seja criada e depois dissipada, porém isso não desfaz o impacto na *Latência*.

Thread Pool Size

É importante observar que a especificação JMS define que cada sessão deve ser *monothread*, logo quanto maior o número de *threads* disponíveis no MOM, maior será o número de sessões abertas por produtores e consumidores.

Se a quantidade de sessões produtoras for maior que a quantidade de consumidoras, isto pode levar o sistema rapidamente ao ponto de saturação. Nem todos os MOM JMS permitem definir essa relação, logo é importante definir um número que comporte a estimativa de clientes (produtores e consumidores) simultâneos. Em alguns casos esse parâmetro é utilizado como um mecanismo de controle de admissão, porém o administrador deve se lembrar que isso somente é válido se as sessões consumidoras forem estabelecidas anteriormente às sessões produtoras, pois nesse caso o controle será estabelecido para os produtores.

Parâmetros Relativos ao Modelo de Comunicação da Aplicação

Os parâmetros analisados a seguir são impostos pelo modelo de comunicação das aplicações que integram o cenário de troca de mensagens. Logo, estes parâmetros só fazem sentido em um processo de planejamento e gerenciamento de capacidade se o administrador quiser avaliar o impacto da mudança de cenário (modificação nas aplicações). Por conta disto, a maioria desses parâmetros são fixados durante as avaliações, refletindo o cenário sob análise.

Quantidade de *Destinations*

O número de filas ou tópicos afeta o desempenho dos MOMs devido ao fato que cada *destination* aloca múltiplos recursos no sistema. Além disso, a expectativa é que o aumento desse quantitativo gere um maior número de clientes (produtores e consumidores), elevando assim a carga de trabalho submetida ao MOM.

A expectativa é que quanto maior o número de *destinations* menor será a capacidade máxima do sistema (limite superior para as métricas *Taxa de Entrega do Sistema* e *Vazão do Sistema*).

Estilo das Mensagens

No caso de sistemas *Pub/Sub*, acredita-se que o *overhead* gerado pelo controle de assinaturas dos tópicos e a manipulação de múltiplas filas podem: diminuir a *Vazão do Sistema*, aumentar a *Latência* e aumentar a *Utilização de CPU* do MOM quando comparado ao estilo PTP.

Tamanho e Tipo da Mensagem

Quanto maior for a mensagem, ou quanto mais complexa for sua estrutura (tipo), mais tempo o MOM deve levar para manipulá-la (receber do produtor, gravar, serializá-la, enviar para o consumidor), gerando um efeito na *Latência*. Mensagens de tamanho muito grandes podem afetar significativamente esta métrica.

Outro aspecto relativo ao tamanho da mensagem é que o espaço em memória destinado para um *destination* passa a armazenar uma quantidade menor de mensagens, levando o sistema a utilizar acesso ao disco após o esgotamento da memória. Isso compromete o desempenho geral do sistema.

Modo de Entrega

O fato de persistir a mensagem antes de enviá-la não deve afetar a *Vazão do Sistema*. Uma vez que o *destination* possua um número mínimo de mensagens que neutralize este efeito de retardo gerado pela persistência, a vazão deve permanecer em níveis equivalentes ao modelo sem persistência.

A persistência deve afetar a *Latência* de forma significativa, pois o tempo de processamento da mensagem dentro do MOM aumenta o *Tamanho da Fila*, ocasionando assim uma *Latência* maior na entrega da mensagem.

Método de Recepção da Mensagem

A tendência é que com o método assíncrono (*push*), a *Vazão do Sistema* seja superior, pois a mensagem é enviada ao consumidor (via notificação) à medida que elas chegam ao MOM, não dependendo de ação do consumidor.

Já com o método síncrono (*polling*), o tempo gasto pelo cliente consumidor entre a recepção de uma mensagem e a chamada subseqüente do método *receive* (*Atraso de Processamento*, parâmetro relativo ao padrão de uso do consumidor) pode afetar a *Taxa de Entrega de Mensagens* e a *Latência*.

Modelo de Assinatura

A utilização de assinaturas duráveis (*durable subscription*) com consumidores inativos por longos períodos faz com que o MOM fique retendo a mensagem até

que o consumidor volte a ficar *online*. Esse tipo de assinatura se mal utilizado pode inviabilizar um sistema de entrega.

Parâmetros Relativos ao Padrão de Uso dos Produtores

Taxa de Geração de Mensagens por Produtor

A *Vazão do Sistema* e a *Taxa de Entrega de Mensagens* são proporcionais a esse parâmetro, desde que o sistema não atinja o ponto de saturação. Se isso ocorrer, a *Latência* deve aumentar consideravelmente, enquanto a vazão e a taxa de entrega devem estabilizar. Se mesmo assim, a *Taxa de Geração* aumentar, o sistema tende a entrar em um estado de colapso fazendo cair a vazão e a taxa de entrega, enquanto a *Latência* continua crescendo exponencialmente.

Parâmetros Relativos ao Padrão de Uso dos Consumidores

Atraso no Processamento da Mensagem

O consumidor processa a mensagem assim que a recebe e normalmente antes de reconhecê-la. Esse mecanismo é fundamental para garantir não só que a mensagem foi recebida, mas que foi efetivamente processada. Isso somente é possível utilizando o modelo de reconhecimento MANUAL_ACK, no caso de recebimento síncrono. No método de recepção assíncrono, quando a notificação chega ao consumidor o agente do MOM local chama o *Listener* configurado. Essa chamada ao método *onMessage* garante que o reconhecimento somente é realizado após a conclusão do método. Normalmente, dentro desse método é realizado o processamento da mensagem.

O tempo que leva o consumidor para processar a mensagem afeta a sua disponibilidade para receber outra mensagem e, conseqüentemente, a *Vazão do Sistema* e *Taxa de Entrega de Mensagem*. A *Latência* não é afetada, pois o processamento se dá após o recebimento da mensagem. Já o *Tamanho da Fila* tende a crescer devido a indisponibilidade momentânea do consumidor.

3.3.4 Definindo os Fatores

A partir da análise de impacto e utilizando os objetivos definidos, o passo final para o entendimento do ambiente é a construção do conjunto de fatores para avaliação do MOM. Esses fatores são parâmetros que são variados ao longo do processo de avaliação.

Os objetivos de desempenho são descritos na forma de um SLO que deve ser atendido pelo sistema. Esse SLO por sua vez é definido em função das métricas estabelecidas.

Em um cenário de planejamento de capacidade, onde o objetivo central é adiar ao máximo a saturação do sistema com o menor custo possível, é fundamental encontrar a capacidade máxima do sistema que atende ao SLO definido. Assim, precisam-se estabelecer fatores que garantam que a capacidade máxima do sistema seja encontrada. Para isso, é preciso que os fatores possam variar a carga de trabalho submetida ao MOM mantendo o modelo de comunicação das aplicações (cenário), ou ainda que possam aperfeiçoar o funcionamento do MOM.

Considerando um cenário típico de um sistema de entrega de mensagem, os fatores escolhidos para variar a intensidade de carga (IC) e a demanda de serviço (DS) nos principais recursos poderiam ser:

- Quantidade de Produtores;
- Quantidade de Consumidores;
- Tamanho e Tipo das Mensagens;
- Taxa de Geração de Mensagens por Produtor.

No entanto, a escolha dos fatores depende do SLO definido. Se por exemplo, a *Latência* é um aspecto importante do SLO, pode-se adicionar no estudo o fator *Tamanho Máximo dos Destinations*, com o intuito de avaliar a implementação de um controle de admissão básico. Na prática, cada processo de planejamento de capacidade deve escolher quais parâmetros se transformam em seus fatores.

É importante ainda ressaltar que os parâmetros que não são fatores ficam com seus valores fixados formando a configuração do cenário a ser avaliado.

3.4 Caracterizar a Carga de Trabalho

Esta atividade desenovlvida pelo *Avaliador* é responsável pela identificação dos componentes que formam a carga de trabalho, com o objetivo de construir o modelo de carga submetido ao sistema alvo.

O componente básico da carga de trabalho, no caso de um sistema de entrega de mensagens, são as mensagens enviadas pelos produtores para serem encaminhadas aos consumidores em um intervalo de tempo.

O que caracteriza a carga de trabalho no contexto de MOM é um conjunto de informações sobre cada mensagem trafegada nesse sistema de comunicação. Essas informações descrevem a intensidade de carga submetida, ou ainda o uso dos recursos (demandas de serviço) do sistema para tratar cada mensagem.

No caso de um MOM, os principais recursos que precisam ser analisados são a fila e a CPU dos servidores. O esgotamento desses recursos levará o sistema em estudo (*System Under Test*, SUT) ao ponto de saturação, comprometendo os níveis de serviço desejados.

Esta é a primeira atividade realizada pelo *Avaliador* que recebe do *Gerente de Capacidade*: a *Biblioteca de Componentes* e a *Arquitetura do Serviço de Integração*, onde os recursos e todos os parâmetros que afetam o desempenho do MOM estão mapeados e documentados.

Conforme definido por [Menascé e Almeida 2002] os passos para efetuar essa atividade são:

1. Identificação dos componentes básicos da carga de trabalho;
2. Seleção do conjunto de parâmetros que capturam as características mais relevantes da carga de trabalho para a finalidade do estudo;
3. Monitoração do sistema para coletar dados brutos de desempenho;
4. Particionamento da carga de trabalho em classes de componentes semelhantes;
5. Cálculo dos valores dos parâmetros que caracterizam cada classe; e

6. Construção e validação de um modelo de carga de trabalho.

O artefato gerado por esta atividade é um modelo de alto nível representando a carga submetida ao sistema, denominado de *Modelo da Carga de Trabalho*. Nesse diagrama cada canal (*destination*) é representado com toda a carga consolidada submetida a ele por todas as interações.

As próximas subseções analisam cada uma das etapas para caracterização da carga de trabalho, focando no domínio dos sistemas de entrega de mensagens.

3.4.1 Identificando os Componentes Básicos

A natureza do serviço fornecido pelo sistema determina o tipo de componente básico da carga de trabalho. No caso de um sistema de entrega de mensagem o componente básico é a mensagem. As mensagens chegam ao MOM originadas dos produtores para serem entregues aos consumidores. Com a preocupação em determinar o impacto do MOM no encaminhamento de mensagens, o ponto de vista definido para caracterizar a carga de trabalho é o lado do *JMS Provider*. Desta forma, podem-se definir quatro tipos de componentes básicos em um sistema de entrega de mensagens:

- *Mensagem PTP recebida pelo MOM*. Todas as mensagens recebidas pelo MOM através de uma fila;
- *Mensagem PTP enviada pelo MOM*. Todas as mensagens encaminhadas pelo MOM para um consumidor;
- *Mensagem Pub/Sub recebida pelo MOM*. Todas as mensagens recebidas pelo MOM através de um tópico;
- *Mensagem Pub/Sub enviada pelo MOM*. Todas as mensagens encaminhadas pelo MOM para cada consumidor que assinou o tópico.

Essa classificação levou em consideração dois aspectos: modelo assíncrono de comunicação fim-a-fim e o estilo das mensagens. O fato do MOM implementar uma comunicação assíncrona fim-a-fim torna a recepção da mensagem um evento independente do envio. Portanto, se um ocorre em um momento não significa que o outro ocorrerá imediatamente em seguida (ex. o consumidor pode não estar pronto para receber a mensagem).

O estilo das mensagens também deve ser levado em consideração, pois o encaminhamento das mensagens é completamente diferente em cada estilo. Enquanto no estilo PTP cada mensagem consome uma vez apenas os recursos do MOM, no *Pub/Sub* cada mensagem consome recursos para cada consumidor que assinou o tópico. Essas considerações levam a parâmetros diferentes para cada tipo de componente básico (mensagem).

Outros tipos de mensagens podem ser considerados como componentes básicos: mensagens para estabelecer e finalizar conexões, mensagens de assinatura de um tópico. É importante, ressaltar que em estudos de planejamento de capacidade dificilmente esses componentes são levados em consideração, pois normalmente são desprezíveis no volume global de mensagens.

3.4.2 Selecionando os Parâmetros

Os componentes básicos são caracterizados por parâmetros que indicam a intensidade da carga de trabalho (ex. quantidade de produtores, taxa de chegada de mensagens) e a demanda de serviço em cada recurso (ex. tempo de CPU, tempo de acesso a disco).

Durante o mapeamento da arquitetura do ambiente (Seção 3.3.1) e o estudo dos padrões de uso do sistema (Seção 3.3.2), , os recursos e parâmetros que normalmente caracterizam a carga de trabalho em um sistema de entrega de mensagens foram apresentados. A Tabela 3-1 apresenta um resumo dos parâmetros para cada componente básico identificado no sistema.

Os parâmetros *Tipo/Formato da mensagem*, *Tamanho da mensagem* e *Confiabilidade* estão presentes em todos os componentes básicos e são associados à demanda de serviço (DS. Como exemplo, uma mensagem maior demandará mais tempo de CPU ou de disco para ser manipulada, além de consumir mais tempo de rede (transmissão ou recepção).

Os parâmetros *Tempo de CPU para receber (ou enviar/remover) mensagem*, *Tempo de acesso a disco para receber (ou enviar) mensagem* e *Tempo de recepção (ou transmissão) da mensagem* são classificados também como demanda de serviço, pois esses tempos representam o consumo de recursos importantes como CPU, disco e rede. É importante ressaltar que esses tempos se referem ao tempo gasto pelos recursos (CPU, disco e rede) do *JMS Provider* para efetuar cada uma dessas operações.

Tabela 3-1. Exemplo de parâmetros para cada tipo de componente básico (IC=intensidade da carga; DS=demanda de serviço)

Componente Básico e *Parâmetros*	**Tipo de Parâmetro**
Mensagem PTP recebida pelo MOM	
Quantidade de produtores	IC
Taxa de geração de mensagens por produtor (com *think time*)	IC
Tipo/Formato da mensagem	DS
Tamanho da mensagem	DS
Confiabilidade do canal (ex. NPNT, PT)	DS
Tempo de CPU para receber mensagem	DS
Tempo de acesso a disco para receber mensagem	DS
Tempo de recepção da mensagem (rede, incluindo o ACK)	DS
Mensagem PTP encaminhada pelo MOM	
Método de recepção das mensagens (polling, push)	IC
Think time do consumidor	IC
Tipo/Formato da mensagem	DS
Tamanho da mensagem	DS
Confiabilidade do canal (ex. NPNT, PT)	DS
Tempo de CPU para enviar/remover mensagem	DS
Tempo de acesso a disco para enviar mensagem	DS
Tempo de transmissão da mensagem (rede, incluindo o ACK)	DS

Componente Básico e *Parâmetros*	**Tipo de Parâmetro**
Mensagem *Pub/Sub* recebida pelo MOM	
Quantidade de produtores	IC
Taxa de geração de mensagens por produtor (com *think time*)	IC
Quantidade de consumidores (assinantes)	IC
Tipo/Formato da mensagem	DS
Tamanho da mensagem	DS
Confiabilidade do canal (ex. NPNT, PT)	DS
Modelo de assinatura (ex. Durable, Nondurable)	DS
Tempo de CPU para receber mensagem	DS
Tempo de acesso a disco para receber mensagem	DS
Tempo de recepção da mensagem (rede, incluindo o ACK)	DS
Mensagem *Pub/Sub* encaminhada pelo MOM	
Quantidade de consumidores (assinantes)	IC
Tipo/Formato da mensagem	DS
Tamanho da mensagem	DS
Confiabilidade do canal (ex. NPNT, PT)	DS
Modelo de assinatura (ex. Durable, Nondurable)	DS
Tempo de CPU para enviar/remover mensagem	DS
Tempo de acesso a disco para enviar mensagem	DS
Tempo de transmissão da mensagem (rede, incluindo o ACK)	DS
Atraso no processamento da mensagem (process delay)	DS

Os parâmetros relacionados com a rede devem incluir o tempo para transmistir (ou receber) a mensagem em si e o tempo necessário para receber (ou transmitir) o recohecimento da mensagem (ACK). O método de reconhecimento (ex. AUTO_ACK) está embutido indiretamente nesse parâmetro.

A *Quantidade de produtores* e a *Taxa de geraçãode mensagens por produtor* estão presentes nos componentes que representam as mensagens recebidas pelo MOM. Esses parâmetros são classificados como relativos à intensidade de carga (IC), pois eles determinam diretamente a quantidade de carga de trabalho (mensagens) submetida ao MOM.

No caso de mensagens *Pub/Sub*, a *Quantidade de consumidores* (assinantes) afeta diretamente a carga do MOM, pois cada mensagem deve ser entregue a cada consumidor (um-para-muitos). No caso de mensagens PTP, esse parâmetro não é relevante, pois mais consumidores conectados em uma mesma fila podem ser entendidos como um único consumidor mais rápido, pois cada mensagem é entregue apenas a um consumidor (um-para-um).

No caso do encaminhamento de mensagens PTP, o *Método de recepção das mensagens* torna-se um parâmetro importante classificado como de intensidade de carga (IC). Quando o método for *polling* (recepção síncrona por parte do consumidor, chamando o método *receive*) isso implica em uma carga de trabalho imposta para encaminhamento de mensagens atrelada a iniciativa do consumidor, não dependente do MOM. Dessa forma, outro parâmetro torna-se

necessário para expressar a intensidade de carga, o *Think time do consumidor* que representa o tempo gasto pelo consumidor para voltar a estar pronto (chamar novamente o método *receive*) para receber mensagens.

O parâmetro *Atraso no processamento da mensagem* também influencia na disponibilidade dos consumidores para receber nova mensagem, afetando a intensidade de carga de trabalho a qual o MOM está submetido. Esse atraso pode ser interpretado como uma parte do *think time* (tempo de pensar) para as mensagens síncronas (PTP), por isso não está presente nesses componentes. Para o modelo assíncrono, porém, ele determina a liberação de recursos dentro do MOM (a mensagem somente pode ser removida do *destination*, após ser recebida pelo consumidor). Ou seja, esse atraso impacta na utilização da fila (tamanho da fila) e, portanto, pode ser interpretado como um parâmetro relacionado à demanda de serviço (DS) da fila.

O parâmetro *Modelo de assinatura* somente faz sentido no estilo *Pub/Sub*, uma vez que se refere-se a como as assinaturas a tópicos são tratadas pelo MOM. Esse parâmetro é caracterizado como sendo de demanda de serviço (DS), pois assim como a fila, o tópico possui tamanho finito, e esse parâmetro pode influenciar de forma decisiva na alocação desse recurso.

3.4.3 Monitorando o Sistema

Muitos dos parâmetros que caracterizam os componentes básicos de carga fazem parte da descrição do cenário da aplicação e não precisam ser obtidos por medição, apenas especificados e determinados. Porém, alguns parâmetros relacionados com demanda de serviço (DS) precisam ser medidos: tempo de CPU, tempo de acesso a disco, tempo para transmissão (ou recepção) da mensagem (rede), taxa de geração de mensagens por produtor.

Os parâmetros de tempo de CPU, tempo de acesso a disco e tempo para transmissão (ou recepção) da mensagem são obtidos com a monitoração do processo (no sistema operacional) que representa o MOM. Essa monitoração pode ser realizada de várias formas, porém a mais natural é através de ferramentas disponibilizadas pelo próprio sistema operacional.

Durante a medição desses parâmetros, a instrumentação do ambiente precisa ser realizada de forma cuidadosa para evitar que outras aplicações estejam consumindo recursos durante o processamento das mensagens. É importante monitorar além da utilização total de CPU, a utilização apenas do MOM, para verificar se existe uma discrepância entre os dois valores, podendo indicar que a instrumentação ainda precisa ser melhorada.

A coleta de dados precisa ser repetida uma quantidade de vezes suficiente para que o valor apresente um nível de confiança estatístico aceitável. Mais informações sobre medição podem ser encontradas no Capítulo 5.

3.4.4 Particionando a Carga de Trabalho

No caso de um sistema JMS, alguns atributos associados às mensagens podem ser combinados para compor as classes de componentes básicos: tipo/formato da mensagem (ex. XML, text, object), tamanho da mensagem (ex. 1K, 10K, 100K), nível de confiabilidade da mensagem ou canal (ex. NPNT, PT), e método de recepção (ex. síncrono, assíncrono).

Em situações reais pode ser necessário agrupar componentes básicos utilizando mais de um atributo. Sachs [Sachs et al. 2007a] propõe dezenove classes de mensagens para compor a carga de trabalho do *benchmark* SPECjms2007 (ver Seção 6.2.2). Essas classes levam em consideração vários atributos clássicos do padrão JMS (tamanho e tipo da mensagem, *destination*, confiabilidade).

O parâmetro *Tempo de CPU* é um importante atributo para ajudar no particionamento das mensagens em conjunto com os atributos funcionais do JMS. Ou seja, a tendência é que as classes pré-definidas pela combinação dos atributos funcionais sejam agrupadas após a coleta de dados de desempenho, de acordo com a utilização de recursos.

É possível que após o particionamento dos componentes básicos em classes de carga seja necessário novamente a realização do passo de monitoração do sistema, a fim de obter os valores para os parâmetros (demanda de recursos, por exemplo) de forma isolada para cada classe de carga identificada.

3.4.5 Calculando os Valores dos Parâmetros

A valoração dos parâmetros obtidos através da monitoração do ambiente pode ser feita pela análise do valor médio, pelo agrupamento (cálculo dos centróides), ou ainda pela equalização de momentos. Os parâmetros relacionados com taxa, tempo e atraso são variáveis aleatórias contínuas e podem ser aproximadas por uma combinação especial de exponenciais (*Phase Type Distribution* [Neuts 1975]).

Na descrição da metodologia de [Menascé et al. 2004] no capítulo anterior (seção 2.5) foram discutidos detalhes que devem ser seguidos quanto ao cálculo desses parâmetros.

3.4.6 Construindo um Modelo de Carga de Trabalho

O objetivo desse passo é construir um modelo abstrato de alto nível que represente a intensidade de carga de trabalho submetida ao MOM. O modelo precisa ser capaz de representar classes de mensagens utilizando uma distribuição percentual para cada classe de carga, podendo esse comportamento do modelo variar no tempo. Dessa forma, o modelo consegue representar cargas de trabalho bastante complexas.

Após o particionamento da carga de trabalho, as mensagens estão agrupadas em classes homogêneas e precisam estar identificadas no modelo. Um dos parâmetros utilizados para o particionamento da carga e que afeta o modelo de carga é o canal utilizado (*destination*). Logo, o modelo de carga deve estar direcionado a identificar a carga de trabalho submetida a cada canal.

A construção do modelo de carga de trabalho é suportada pelo *toolkit* através de modelos de alto nível da biblioteca de componentes para representar os quatro tipos de componentes básicos identificados (ver Tabela 3-1) e que se dividem em mensagens recebidas pelo MOM e mensagens encaminhadas pelo MOM.

A primeira ação para a construção do modelo de carga de trabalho é a representação da comunicação existente em cada interação utilizando notação específica definida pelo *toolkit*. As comunicações foram representadas

inicialmente pelo *Integrador de Sistemas* utilizando uma notação de diagrama de sequência.

O conjunto dos modelos de carga para todas as comunicações existentes no *Plano de Integração* em conjunto com as informações adicionais sobre as mensagens formam o *Modelo de Carga de Trabalho*, artefato final da atividade de caracterização da carga de trabalho.

3.5 Desenvolver o Modelo de Desempenho

Depois de concluída a caracterização da carga de trabalho, o *Avaliador* passa a ser responsável por construir um modelo de desempenho para o sistema de entrega de mensagem (MOM). Para realizar esta atividade, ele recebe como entrada os seguintes artefatos: *Biblioteca de Componentes*, *Modelo de Carga de Trabalho* e *Arquitetura do Serviço de Integração*.

De forma geral, o *toolkit* disponibiliza um conjunto de componentes de desempenho (modelos formais) que representam os principais elementos de um sistema de entrega de mensagens (produtor, canal de comunicação e consumidor). Esses modelos quando são aplicados para implementar o *Modelo de Carga de Trabalho* já desenvolvido formam o *Modelo de Referência do Sistema* (*System Reference Model*). A construção desse modelo de desempenho segue algumas regras de composição e substituição detalhadas juntamente com a biblioteca. Os componentes de desempenho são modelados em redes de Petri (*Petri nets*, ou PNs), permitindo que eles sejam avaliados tanto analiticamente quanto via simulação.

Resumindo, esta atividade do processo basicamente consiste na combinação de vários componentes para formar um *Modelo de Referência do Sistema*. Um maior detalhamento dessa atividade somente é possível após a apresentação da biblitoeca de componentes. Logo, no Capítulo 4 durante a descrição da biblioteca essa atividade é caracterizada.

3.6 Parametrizar, Validar e Calibrar os Modelos

Esta atividade desenvolvida pelo *Avaliador* é responsável pela construção de um modelo de desempenho voltado para avaliação das métricas do sistema. O primeiro passo é a obtenção dos valores para os parâmetros do *Modelo de Referência do Sistema*. Em seguida, o modelo precisa ser validado e calibrado para representar o comportamento de desempenho do sistema real. Ao fim dessa atividade o artefato gerado é o *Modelo Refinado do Sistema*.

3.6.1 Parametrização

O *Modelo de Referência* é um modelo sob o qual é derivado um modelo específico para a plataforma onde o sistema encontra-se implantado. Essa derivação é realizada através do cálculo dos parâmetros de entrada do modelo (parâmetros do sistema, dos recursos, e da carga de trabalho). Esse cálculo é realizado a partir de medições específicas do SUT, que são suportadas por ferramentas descritas no próximo capítulo.

Depois que o *Modelo de Referência* recebe os valores dos seus parâmetros de entrada, torna-se um modelo específico para a plataforma alvo chamado *Modelo de Desempenho do Sistema*. Uma plataforma alvo representa o conjunto de

recursos computacionais que descrevem a configuração da arquitetura, o software, e a carga de trabalho do SUT.

3.6.2 Validação

A validação de um modelo garante que ele representa com certa fidelidade o comportamento do sistema real. Para realizar essa validação é necessário realizar primeiramente um teste de consistência que verifica se o modelo se comporta de forma coerente e consistente quando os valores dos parâmetros de entrada são modificados de forma harmônica.

Por exemplo, suponha que você está avaliando o MOM recebendo 10mps (mensagens por segundo) de apenas um produtor, e que você modifica para que ele passe a receber 5mps de dois produtores, ou seja, teoricamente o comportamento do MOM deveria ser similar, pois a carga é similar.

Na Seção 3.3.3 são definidos possíveis impactos no desempenho do MOM a partir da variação dos principais parâmetros. Esse conjunto de hipóteses pode e deve ser utilizado para verificar a consistência do modelo.

Após o teste de consistência, é necessário realizar um teste comparativo com o sistema real. Para isso, é necessário escolher cenários de validação que explorem os aspectos gerais e específicos do sistema.

Esses cenários precisam ser implementados tanto no modelo quanto no sistema real. Em seguida, o sistema real deve ser submetido a uma atividade de medição para avaliar métricas de desempenho escolhidas para o sistema. Os resultados obtidos na medição são comparados com os resultados obtidos por simulação ou resolução analítica do modelo.

Menascé [Menascé e Almeida 2002] define que para o planejamento de capacidade, resultados obtidos através de modelos que possuam taxas de erro entre 10% a 30% dos valores reais é bastante satisfatório.

3.6.3 Calibragem

Considerando que durante a validação o modelo apresente distorções nos resultados, seja na verficação de consistência, ou mesmo na validação comparativa com o sistema real, faz-se necessário a calibragem do modelo para que seus resultados passem a ser satisfatórios.

A calibragem normalmente consiste em refinamento estatístico dos valores dos parâmetros do modelo. Algumas vezes o procedimento para obtenção dos parâmetros embutiu erros estatísticos elevados, ou até mesmo os valores não passaram por tratamentos de *outliers*, por exemplo. O importante nesse passo é garantir que os parâmetros são representados de forma mais fiel possível. Mais detalhes sobre tratamento estatístico de medições pode ser encontrado no Apêndice A.

Uma vez realizado esse refinamento dos parâmetros o modelo precisa novamente passar pelo passo da *Validação* até que o modelo esteja representando com a fidelidade desejada o sistema real.

Por conta de tudo isso esse passo é muitas vezes denominado de refinamento, e produz ao final o *Modelo Refinado do Sistema*.

3.7 Prever a Evolução da Carga de Trabalho

Esta atividade desenvolvida pelo *Gerente de Capacidade* é responsável pela previsão da evolução da carga de trabalho, garantindo que o MOM não comprometerá o processo de integração.

Para isso, o *Gerente de Capacidade* necessita dos artefatos *Arquitetura do Serviço de Integração* e do *Modelo de Carga de Trabalho* e precisa realizar os seguintes passos: selecionar as demandas de serviço, definir métricas de negócio, estimar crescimento do negócio e relacionar métricas de negócio com a demanda por serviços.

Ao final, é produzido o artefato com os *Cenários de Carga* estimados para o futuro como uma evolução da carga de trabalho atual. A idéia central é estabelecer possíveis situações de evolução em que o sistema será submetido e ser capaz de avaliar a resposta de desempenho que o sistema deverá produzir em cada um dos cenários.

3.7.1 Selecionar as Demandas de Serviço

Algumas demandas de serviço precisam ser mapeadas como sendo objeto da previsão. Ou seja, a previsão deve estebelecer possíveis cenários de evolução das demandas escolhidas.

No contexto de integração de aplicações corporativas, a quantidade de mensagens geradas pelas interações de integração é uma boa demanda de serviço.

3.7.2 Definir Métricas de Negócio

As métricas de negócio que se relacionam com o contexto de integração podem ser obtidas pela observância das ocorrências de eventos geradores das interações de integração, ou seja, eventos que resultam em uma sequência de troca de mensagens. Na atividade de caracterização da carga de trabalho esses eventos foram identificados e modelados no *Modelo de Carga de Trabalho*.

3.7.3 Estimar Crescimento do Negócio

O próximo passo é a estimativa de crescimento do negócio em função das métricas de negócio definidas. No contexto de integração de aplicações utilizando MOMs, é natural procurar respostas para questões relacionadas ao crescimento da utilização dessas aplicações envolvidas na integração. Quanto mais usuários e transações, a tendência é que exista mais trabalho para o MOM. Ou seja, nesse passo procura-se entender a evolução do negócio para determinar depois o impacto na infra-estrutura de integração.

3.7.4 Relacionar Métricas de Negócio com Demandas de Serviço

O passo seguinte é relacionar dados históricos de demandas de serviço a indicadores de negócio.

Por fim, baseado na estimativa de crescimento do negócio e na relação entre medidas de negócio e demandas por recursos, elabora-se uma previsão de futuro construindo cenários possíveis para a evolução da carga de trabalho.

3.8 Avaliar Cenários

Como artefatos de entrada para esta atividade, o *Avaliador* necessita dos *Cenários de Carga* e do *Modelo Refinado*. Com esses artefatos, ele deve seguir os seguintes passos: transformar métricas em fórmula referente ao modelo, escolher o método de avaliação (simulação ou resolução analítica) e avaliar os modelos.

Ao final, a atividade produz um *Relatório da Avaliação*, onde consta a avaliação de cada métrica do sistema em cada cenário de carga estabelecido. Esse produto final é normalmente apresentado em forma de gráficos, mostrando a evolução das métricas versus os níveis dos fatores utilizados.

Detalhar esta atividade e a próxima sem apresentar a biblioteca de componentes e as ferramentas seria bastante abstrato para o leitor e de difícil compreensão. Logo essas duas atividade são detalhadas de forma prática no Capítulo 5 durante a demonstração do *toolkit*.

3.9 Analisar Resultados

Ao final das avaliações, e de posse dos artefatos *Relatório de Avaliação* e *Service Level Objectives* (SLO), o *Gerente de Capacidade* inicia a atividade de *Análise dos Resultados*. Esta atividade tem a responsabilidade de construir o *Plano de Capacidade*, principal artefato de todo o processo.

Para realizar essa atividade, dois passos são fundamentais: levantar custos dos cenários estudados; e comparar resultados de cada cenário com o custo associado.

O *Plano de Capacidade* é composto por um plano de configuração e um plano de investimento. O plano de configuração indica o que deve ser alterado para suportar a carga futura mantendo o SLO. Enquanto isso, o plano de investimento determina quando deve ser alterada a configuração em função da previsão de aumento da carga e da capacidade do sistema atual.

3.10 Fluxo Completo do Processo

A apresenta um fluxo completo do processo, incluindo além dos papéis e atividades, os artefatos de entrada e saída. Nesse fluxo aparece o papel do *Projetista*. Esse papel tem a responsabilidade de construir, expandir e manter a biblioteca de componentes do *toolkit*. Esta dissertação faz o papel do *Projetista* preparando um conjunto de componentes a serem utilizados durante a modelagem tanto da carga de trabalho, quanto do sistema como um todo.

Uma vez finalizado todos os passos do processo de planejamento de capacidade apresentado neste capítulo, é necessário entender que esse processo deve ser executado continuamente. Sendo assim, um novo ciclo precisa ser iniciado, buscando o entendimento novamente do ambiente, para verificar possíveis mudanças, ou ainda, mudanças de percepções ou interesses de avaliação. Após cada ciclo completo o *Gerente de Capacidade* obtém mais experiência com o ambiente, com o sistema, e com os objetivos a serem alcançados, tornando mais efetivo o planejamento a ser realizado.

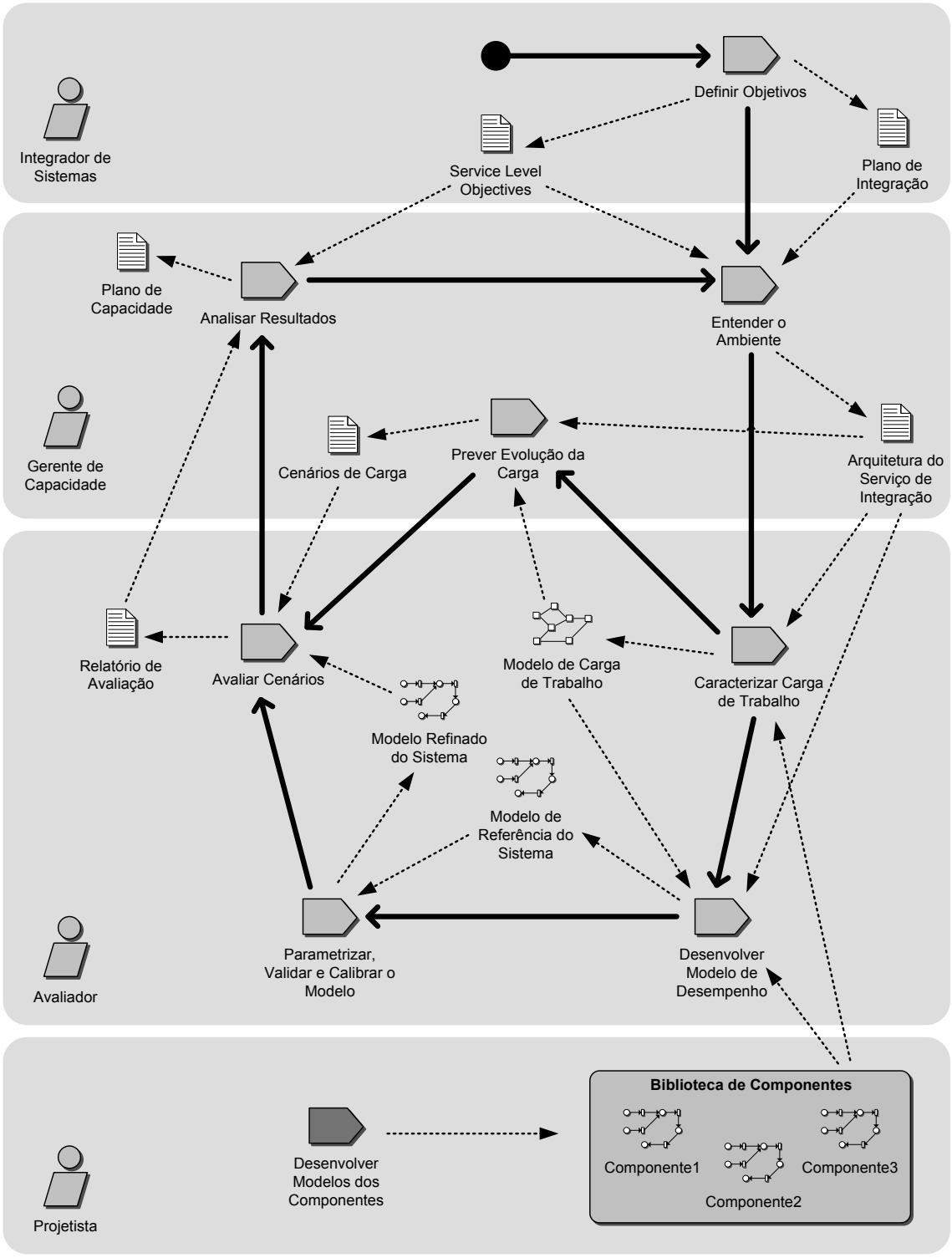


Figura 3-11. JMSCapacity: processo com fluxo completo de atividades.

3.11 Considerações Finais

Este capítulo apresentou o processo do JMSCapacity para planejamento e gerenciamento da capacidade de MOMs JMS. Inicialmente foi mostrado a origem do processo a partir da adaptação de um processo geral definido na literatura.

Em seguida, cada atividade do processo foi detalhada e instanciada para o domínio de MOM. Ao longo desse detalhamento, várias definições e recomendações podem ser reutilizadas quando da aplicação do processo para realização de um planejamento de capacidade real.

CAPÍTULO

4

JMSCapacity: Biblioteca e Ferramentas

"Um problema bem definido é um problema metade resolvido."

Charles F. Kettering.

Este capítulo tem como objetivo dar continuidade ao detalhamento do *JMSCapacity Toolkit*. Inicialmente são apresentados os componentes abstratos utilizados para construção do modelo de carga. Em seguida, são apresentados componentes formais utilizados para construção do modelo de desempenho do sistema. Para cada um dos tipos de componentes é detalhada a notação básica, além de orientações de como utilizá-los na construção dos modelos. Por fim, são apresentadas as ferramentas desenvolvidas ou recomendadas para suportar as atividades do processo apresentado no capítulo anterior.

4.1 Componentes de Carga

O modelo de carga é construído em um nível de abstração elevado, permitindo realizar uma derivação sistematizada a partir do diagrama de seqüência, que é parte integrante do artefato *Plano de Integração* e construído durante a atividade *Definir Objetivos* (ver Seção 3.2.3).

No diagrama de seqüência são identificadas as mensagens trocadas entre as entidades comunicantes. Essas trocas são mapeadas em interações e para cada interação é identificado o evento que a dispara. O modelo de carga representa cada um desses elementos descritos no *Plano de Intregação*, e torna-se fundamental como um modelo intermediário entre o diagrama de sequência e o modelo de desempenho. Esse modelo intermediário possibilita a sistematização da geração do modelo de desempenho e torna o *toolkit* extensível.

4.1.1 Notação Básica

A notação básica utilizada pelo *Avaliador* na construção dos modelos de carga de trabalho é composta por componentes que representam diretamente os elementos que compõem os diagramas do *Plano de Integração*.

Entidade

Uma entidade pode ser um programa, uma rotina, uma *thread*, ou, na nomenclatura JMS, uma *JMS Session*. A Figura 4-1 apresenta a notação utilizada para representar uma entidade.

Figura 4-1. Componente de carga: entidade.

As sessões na especificação JMS não são concorrentes, o que significa que cada entidade mapeada produz ou consome mensagens, nunca as duas atividades simultaneamente. Diante disso, é associado a cada entidade um tipo que representa o *JMS Client* criado a partir da *JMS Session*: *JProd*, *JSCons* ou *JASCons*, representando o *JMS Producer*, *JMS Consumer* com recepção síncrona e *JMS Consumer* com recepção assíncrona, respectivamente.

O tipo *JSCons* é abstrato, sendo necessárias versões concretas incorporando a semântica adequada para consumo de mensagens síncronas. Na biblioteca já estão disponíveis os tipos concretos para representar os consumidores síncronos com as várias modalidades de recepção previstas na especificação JMS: *JSCons-Block*, *JSCons-NoWait* e *JSCons-TimeOut*.

Uma simplificação utilizada neste *toolkit* é que, mesmo sabendo que uma *JMS Connection* pode iniciar várias *JMS Sessions*, considera-se que cada entidade possui uma *JMS Connection* com o *JMS Provider*. Apesar deste fato gerar uma sobrecarga maior no *JMS Provider*, ele não se constitui em uma grande limitação, desde que o número de entidades por Nó não seja elevado. O diagrama de instalação, parte integrante também do *Plano de Integração*, detalha essa relação, podendo o *Avaliador* saber antecipadamente se terá problemas por conta dessa simplificação.

Canal de Comunicação (*Destination*)

Um canal de comunicação representa um *JMS Destination*, podendo ser uma fila ou tópico. Os atributos de um canal de comunicação são: *JMS Provider* (o qual está instalado), lista de demandas de recursos consumidos para cada classe de carga submetida a este canal (e.g., CPU, Disco, Rede). A Figura 4-2 apresenta a notação utilizada para um canal de comunicação, onde a seta indica a direção do fluxo de dados. Como os atributos não são visualmente representados, é importante que o *Avaliador* crie um catálogo dos canais utilizados. Essa informação será útil na construção do modelo de desempenho.

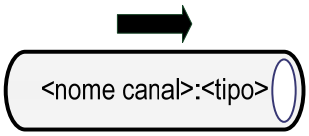


Figura 4-2. Componente de carga: canal de comunicação.

Os canais podem assumir dois tipos abstratos: *JDPTP* e *JDPS*, representando *JMS Destinations* PTP ou *Pub/Sub*, respectivamente. A biblioteca traz para cada tipo abstrato um conjunto de tipos básicos (concretos) representando o nível de confiabilidade implementado pelo canal. O *JDPTP* possui os subtipos *JDPTP-NPNT* e *JDPTP-PNT*. Para o *JDPS* também estão disponíveis os subtipos *JDPS-NPNT* e *JDPS-PNT*.

Algumas características associadas ao canal não fazem parte necessariamente de sua especificação, como o modo de entrega e integridade transacional. Mas, como usualmente não se misturam num mesmo canal, mensagens heterogêneas, este *toolkit* associa essas características de confiabilidade ao canal.

Comunicação

A comunicação entre duas entidades é representada através da ligação da entidade produtora ao canal e este à entidade consumidora. Essa ligação possui ainda uma cardinalidade representando a quantidade de instâncias de produtores e consumidores envolvidos na comunicação. A Figura 4-3 apresenta a notação utilizada para uma comunicação com duas entidades e um canal de comunicação.

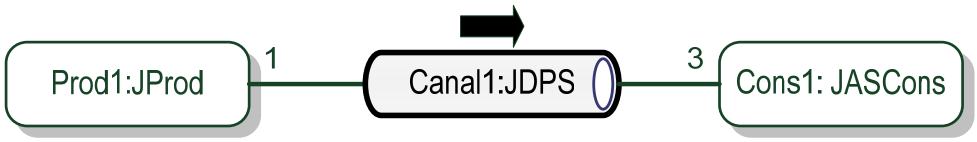


Figura 4-3. Componente de carga: esboço da comunicação.

A semântica de entrega de mensagens depende do canal, conforme detalhado no Capítulo 2 (Seção 2.2.6) quando foi descrita a especificação JMS. Alguns elementos precisam ser adicionados à essa comunicação para complementar o modelo: a mensagem trocada e o evento gerador da mensagem.

Mensagem

Componente básico da carga de trabalho, a mensagem é o elemento fundamental na especificação de um planejamento de capacidade de um MOM. Por conta disso, possui vários atributos importantes que precisam estar documentados, tais como: tipo, tamanho, tempo de expiração, e prioridade. Atualmente, o *toolkit* não possui implementações de componentes de desempenho para suportar a utilização de mensagens com tempo de expiração ou prioridade.

Figura 4-4. Componente de carga: mensagem.

A Figura 4-4 apresenta a notação para representar uma mensagem. A mensagem deve ser ligada atavés da linha pontilhada para a linha de comunicação que interliga a entidade produtora ao canal de comunicação.

Evento

Um evento é um fato, uma circustância ou acontecimento e pode ser compreendido através de métricas de negócio ou da simples chegada de uma mensagem em um consumidor específico. A Figura 4-5 apresenta a notação definida para este componente.

Figura 4-5. Componente de carga: evento.

Ele está primariamente associado à uma entidade produtora, indicando para esta produzir uma nova mensagem a cada ocorrência do evento. O evento pode ser especificado através de uma função *F(x)*, que determina a taxa com que o evento acontece, ou ainda pela ligação com alguma entrega já definida. Nesse caso, o elemento gráfico deve ficar associado também à direita da entidade consumidora, indicando o seu disparo.

Outra situação de utilização do evento é associado a consumidores síncronos (*JSCons*), indicando o momento em que o *consumer* deve procurar por uma nova mensagem no *provider*. Nesse caso, o evento deve ser colocado à esquerda da entidade consumidora síncrona.

No caso do evento ser especificado através de uma função *F(x)*, ela pode assumir características probabilísticas ou determinísticas, limitando-se, no caso de eventos probabilísticos, à distribuições exponenciais ou outras que possam ser aproximadas pela composição delas. Essa limitação está relacionada ao formalismo escolhido (GSPN) para especificar o modelo de desempenho.

4.1.2 Orientações para Construção do Modelo de Carga

A Figura 4-6 mostra um exemplo de *Modelo de Carga de Trabalho* envolvendo duas comunicações. Na primeira, a *Entidade1* produz a *Mensagem1* a cada ocorrência do evento *E1* entregando no *Canal1*. A *Entidade2* possui duas sessões ativas: uma consumidora (*Entidade2_S1*), outra produtora (*Entidade2_S2*). O canal é do tipo *JDPS*, logo, ele entrega cada mensagem para cada uma das três instâncias da *Entidade2_S1*.

A segunda comunicação está relacionada com a primeira, pois a cada mensagem que a *Entidade2_S1* recebe, ele dispara o evento *E2* que faz com que a *Entidade2_S2* produza uma nova mensagem (*Mensagem2*) para o *Canal2*. Como este é um canal PTP e a *Entidade3* é um consumidor síncrono, a mensagem ficará no canal à disposição do consumidor até que o evento *E3* ocorra, indicando o momento do consumo.

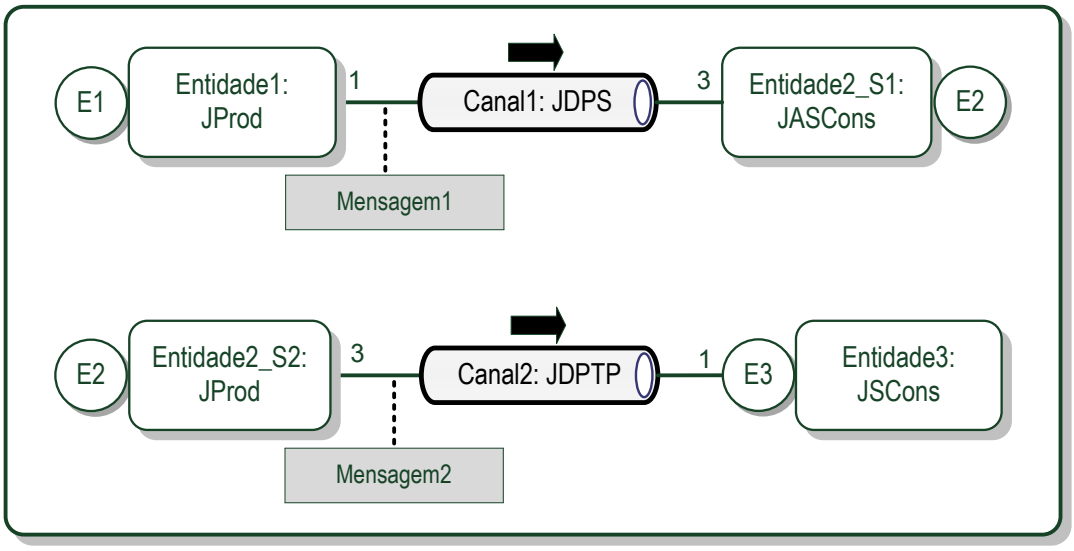


Figura 4-6. Modelo de Carga: exemplo da notação completa.

Para demonstrar que essa representação pode ser derivada do diagrama de seqüência definido no *Plano de Integração*, a Figura 4-7 apresenta o procedimento reverso.

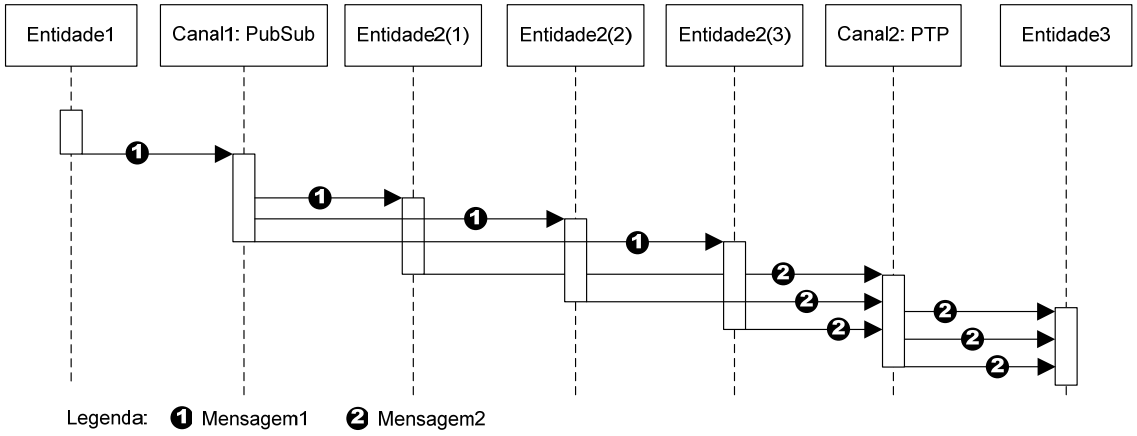


Figura 4-7. Diagrama de seqüência equivalente ao exemplo da Figura 4-6.

Durante a construção do modelo de carga é importante que o *Avaliador* procure reduzir ao máximo o número de linhas de comunicação no modelo. Isso possibilitará maior legibilidade, além de contribuir para a geração do modelo de desempenho.

Essa redução pode ser obtida agregando várias linhas de comunicação numa única linha. Para isso, é preciso que todas as linhas candidatas a serem reduzidas possuam em comum:

- Canal de comunicação; e
- Entidade consumidora (inclusive a cardinalidade).

Acontecendo as duas situações, as linhas de comunicação devem ser reduzidas, conforme apresentado no exemplo da Figura 4-8. Nessa redução, as entidades produtoras são ligadas ao mesmo canal, cada uma com sua cardinalidade, mensagem e evento gerador. No caso dos eventos disparados pela entidade consumidora, os dois eventos ficam ativos indicando que na implementação do modelo de desempenho esse aspecto precisa ser levado em consideração. Na

documentação dos eventos precisa estar especificada a condição de disparo, indicando a partir da chegada de qual mensagem ele é disparado. Essa especificação deve também informar qual a origem (entidade produtora).

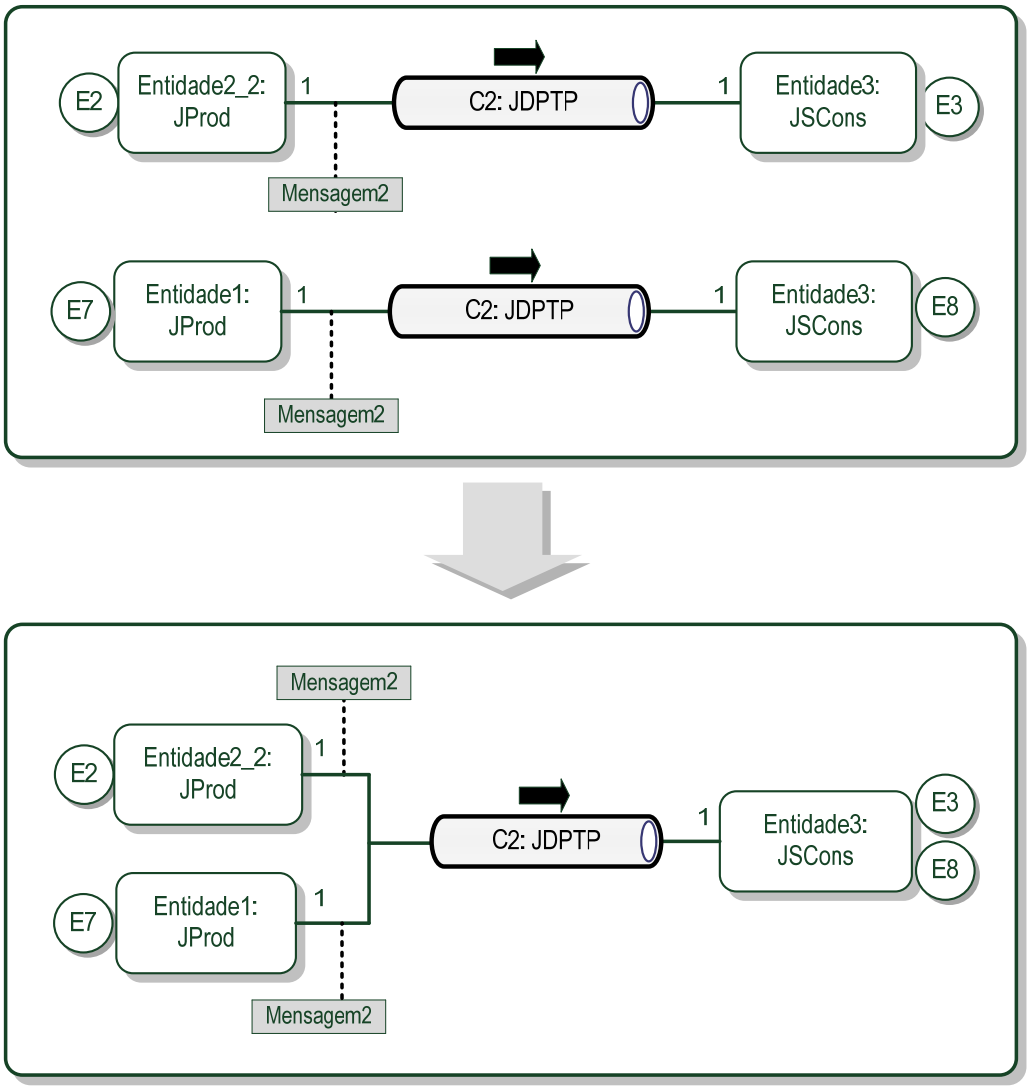


Figura 4-8. Modelo de carga: exemplo de redução.

Outra característica deste *toolkit* é que os tipos definidos aqui podem ser facilmente estendidos, necessitando apenas que, para cada novo tipo no modelo de carga, seja desenvolvido um componente de desempenho equivalente, conforme descrito na própria seção.

4.2 Componentes de Desempenho

A construção do modelo de desempenho é baseada e orientada pelo *Modelo de Carga de Trabalho* desenvolvido. Este *toolkit* organiza a atividade *Desenvolver Modelo de Desempenho* para possibilitar uma geração sistematizada do modelo de desempenho.

Inicialmente, são apresentados os componentes de desempenho básicos para representar os tipos definidos para produtor, canal de comunicação e consumidor. Em seguida, alguns tipos complexos são apresentados, demonstrando como a biblioteca pode ser facilmente estendida. Por fim, são

apresentadas orientações básicas sobre a construção do modelo de desempenho a partir da derivação do *Modelo de Carga de Trabalho*.

O modelo de desempenho é implementado utilizando o formalismo das redes de Petri (PN), em especial as GSPN. Maiores detalhes sobre a validação da biblioteca de componentes de desempenho especificadas em redes de Petri estão apresentados no Apêndice C.

4.2.1 Componente Básico: Produtor

Os produtores são representados no modelo de carga como uma entidade e um evento associado a ele que determina quando ele deve enviar uma nova mensagem. A junção da entidade produtora (*JProd*), o evento gerador e a mensagem gerada é implementada através de um único componente de desempenho denominado de Grupo de Produção (*ProducerGroup* ou *PG*). Cada linha de comunicação associada a um canal no modelo de carga é traduzida para um *PGn* associado ao canal. Um mesmo canal pode estar submetido ao tráfego de mensagens gerado por vários *PGn*.

Alguns aspectos determinam qual implementação (componente de desempenho) de *JProdGroup* o *Avaliador* deve utilizar para traduzir uma linha de comunicação. O evento determina se a escolha será por uma distribuição de probabilidade (*JProdGroup-Poisson*), uma função específica (*JProdGroup-Fx*), ou até mesmo pela conexão com a chegada de mensagens em outra linha de comunicação (*JProdGroup-Connect*).

Outro aspecto a ser considerado é se o grupo de produção atuará de forma síncrona ou assíncrona na geração de mensagens, ou seja, se o *PG* aguarda ou não a conclusão do envio da mensagem para iniciar o procedimento de espera de um novo ciclo de envio. Os *PGs* que tiverem a decisão de geração de uma nova mensagem baseada numa distribuição de probabilidade ou em uma função específica *F(x)* deverão possuir implementações síncronas e assíncronas para escolha do *Avaliador*.

A biblioteca traz algumas opções de componentes para implementar os diversos tipos de *PG*.

JProdGroup-Poisson

Componente de desempenho responsável por produzir novas mensagens a uma taxa definida por uma distribuição de *Poisson*. A Figura 4-9 apresenta o modelo PN que representa o *JProdGroup-Poisson*. Esse modelo é formado por dois lugares (*JProd_PG1*, *PG1_OutgoingMsg*) e uma transição temporizada (*SendMsg_PG1*). O rótulo *PG1* em todos os elementos indica que eles implementam o grupo de produção de número “um”.

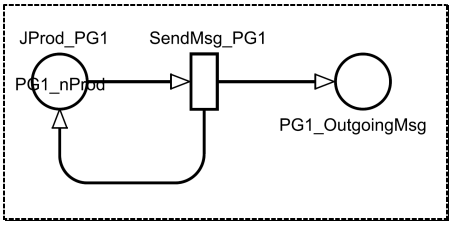


Figura 4-9. Componente de desempenho: *JProdGroup-Poisson*.

O lugar *JProd_PG1* representa os grupos de produção, sendo a marcação do lugar (número de *tokens*) a quantidade ativa na produção de mensagens. Cada mensagem produzida é representada por um novo *token* colocado no lugar *PG1_OutgoingMsg* pela transição *SendMsg_PG1*.

Os parâmetros desse modelo são *PG1_nProd* e *PG1_SendTime*. O primeiro define a quantidade de instâncias que compõem o *PG1*. Este número representa a cardinalidade associada à entidade produtora no modelo de carga. O segundo parâmetro é o *delay* da transição temporizada que determina o comportamento discreto de *Poisson* do modelo. A semântica de temporização dessa transição é *infinite_server* indicando concorrência e independência.

O lugar de saída do modelo, *PG1_OutgoingMsg*, representa o ponto de interconexão com os componentes de desempenho que implementam os canais de comunicação, estando presente também no modelo dos canais para indicar a forma de conexão.

4.2.2 Componente Básico: Canal de Comunicação (*Destination*)

Um *JMS Provider* é representado pela junção dos canais de comunicação que operam em um mesmo servidor. A abordagem utilizada na modelagem dos componentes de desempenho que implementam os canais de comunicação representa a demanda de recursos (CPU, Disco e Rede) do *provider* que cada componente básico impõe ao sistema

Os componentes que implementam a persistência das mensagens representam o sistema de armazenamento (Disco), enquanto que nos modelos não persistentes esse recurso é desconsiderado. Como os modelos se tornam bastante semelhantes, o primeiro deles é apresentado em detalhes enquanto nos demais apenas as diferenças são ressaltadas.

JDPTP-NPNT

Componente de desempenho responsável pelo encaminhamento das mensagens entre produtores e consumidores utilizando o estilo de comunicação PTP, adotando um nível de confiabilidade NPNT. Cada *PGn* associado a este canal deve receber uma nomeclatura própria (*PG1, PG2, e assim por diante).*

A Figura 4-10 apresenta o modelo PN para este componente. Ele possui quinze lugares e dez transições (cinco imediatas e cinco temporizadas). Além dos seus lugares, este modelo representa o ponto de interconexão (*PG1_OutgoingMsg*) com um grupo de produção *PG1*.

Os lugares que representam os recursos do *provider* estão rotulados com o prefixo *PRV*, representando CPU (*PRV_CPUs*) e rede. No caso da rede, estão modelados tanto o acesso exclusivo ao meio físico (*PRV_IncomingNetwork* e *PRV_OutgoingNetwork*), quanto às filas de recepção (*PRV_RXQueue*) e transmissão (*PRV_TXQueue*). Esse modelo não representa o descarte de pacotes se a rede estiver saturada, mas isso poderia ser facilmente implementado como uma extensão.

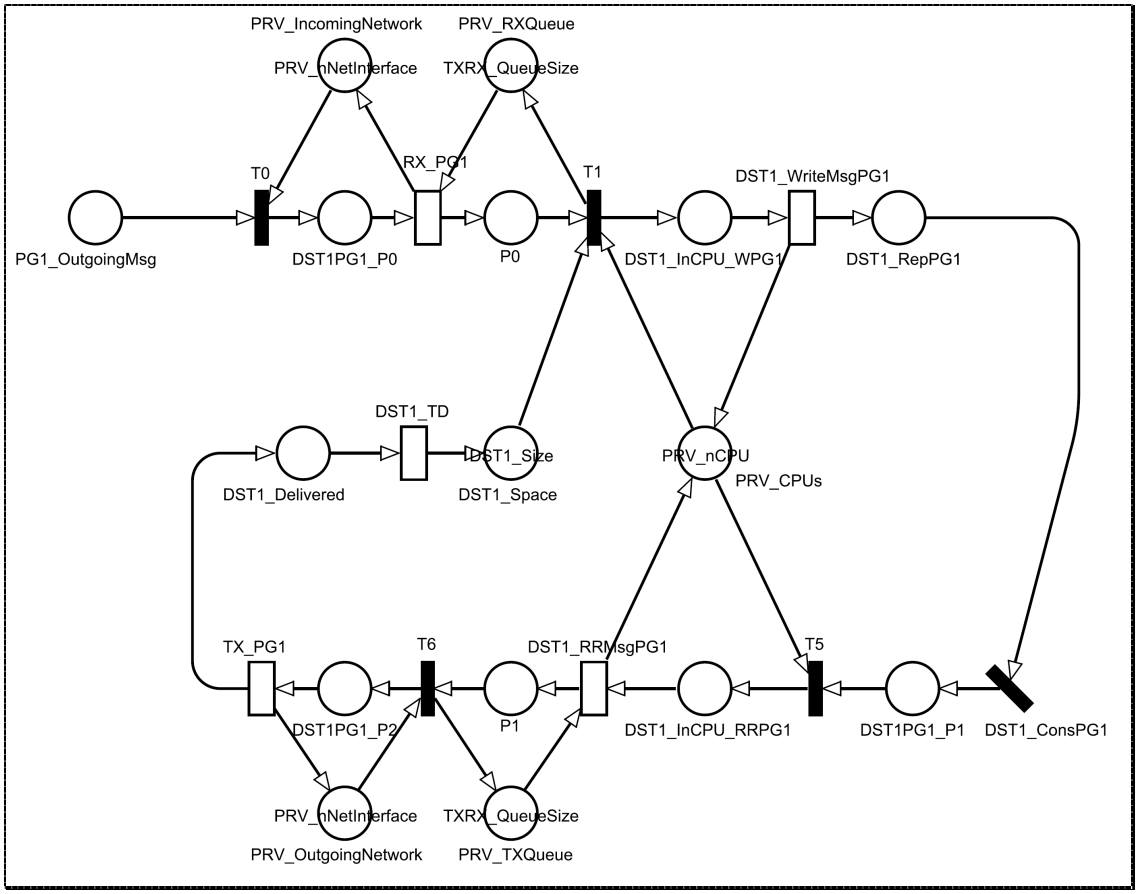


Figura 4-10. Componente de desempenho: *JDPTP-NPNT.*

Os parâmetros referentes aos recursos do *provider* são: *PRV_nCPU*, *PRV_nNetInterface*, *TXRX_QueueSize*, representando o número de CPUs no servidor, o número de interfaces de rede e o tamanho das filas de transmissão e recepção, respectivamente.

Os lugares específicos ao canal em si (*Destination* ou *DST*) recebem um rótulo *DSTn* identificando de qual canal o lugar faz parte, em conjunto com o rótulo *PGn*, indicando a qual grupo de produção ele faz referência. As mensagens que estão em trânsito no canal *DST1*, provenientes do grupo de produção *PG1*, são representadas por *tokens* nos lugares marcados por esses dois rótulos. A motivação para essa nomeclatura é facilitar a automatização da construção do modelo no futuro, pois como se pode ter mais de um *PG* associado a um *DST*, faz-se necessária essa identificação. É importante ressaltar que cada *PG* impõe uma demanda de serviço diferente ao canal em função.

O lugar *DST1_RepPG1* representa o repositório de mensagens do canal de comunicação *DST1* recebidas do grupo de produção *PG1* e ainda não encaminhadas os consumidores. Os lugares *DST1_InCPU_WPG1* e *DST1_InCPU_RRPG1* representam as mensagens do canal *DST1*, recebidas do grupo de produção *PG1* que estão consumindo recursos de CPU para escrita (*Write*) e para Leitura/Remoção (*Read/Remove*), respectivamente.

O lugar *DST1_Space* representa o recurso de espaço do repositório de mensagens do canal *DST1*. O parâmetro *DST1_Size* representa o tamanho máximo do *destination* e está modelado como a marcação inicial de *DST1_Space*, representando que inicialmente o *destination* está vazio.

O lugar *DST1_Delivered* representa as mensagens do canal *DST1* que acabaram de ser entregues ao consumidor. Se o canal possuir mais de um gupo de produção, este lugar não deve ser repetido, e todas as transições responsáveis pela transmissão das mensagens (neste caso a *TX_PG1*) devem convergir para este lugar. Os demais lugares possuem representatividade secundária no modelo, mas necessários para compor o modelo.

As transições imediatas *T0*, *T1*, *T5* e *T6* servem exclusivamente para alocação de recursos (rede, CPU e *destination*) objetivando o processamento das mensagens. As transições temporizadas *RX_PG1*, *DST1_WriteMsgPG1*, *DST1_RRMsgPG1* e *TX_PG1* modelam o consumo de recurso (rede e CPU) para efetuar as operações internas do *provider*. As transições imediatas removem *tokens* dos lugares que representam os recursos e as transições temporizadas processam a mensagem e devolvem o *token* ao lugar dos recursos (representando a liberação do recurso).

As transições temporizadas possuem parâmetros que definem o *delay* e a semântica de temporização. No caso do *delay*, os parâmetros são *TXRXTime_PG1*, *CPUTime_WriteMsgPG1*, *CPUTime_ReadRemoveMsgPG1* representando o tempo médio gasto para receber ou transmitir uma mensagem, o tempo médio de CPU gasto para escrever uma mensagem no *destination* e o tempo médio de CPU gasto para ler e remover uma mensagem do *destination*, respectivamente. Quanto à semântica de temporização, as duas transições relativas à rede são *exclusive_server* indicando serialização da comunicação, enquanto que as transições que representam o gasto de CPU são *infinite_server*, representando a possibilidade do *provider* possuir múltiplas CPUs. Neste último caso, a limitação é fornecida pelo parâmetro que representa o recurso CPU (*PRV_nCPU*).

A transição temporizada *DST1_TD* é um recurso de modelagem para facilitar o cálculo da produtividade do canal, ou seja, a apuração da métrica *Message Delivery Rate*. Ela possui um tempo de disparo (*delay*) muito pequeno (0.0000001) apenas para permitir o cálculo da métrica. Essa transição possui semântica de temporização *exclusive_server* para garantir que seja considerada no cálculo da métrica cada mensagem individualmente.

A transição imediata *DST1_ConsPG1* representa o ponto de interconexão com o componente de desempenho que representa o consumidor responsável pelo consumo das mensagens do canal *DST1* originadas do grupo de produção *PG1*. Ou seja, a entidade consumidora especificada no modelo de carga nesta linha de comunicação é interligada neste ponto. De forma equivalente ao ponto de interconexão dos produtores, esta transição é também representada nos modelos dos consumidores.

JDPTP-PNT

Componente de desempenho responsável pelo encaminhamento das mensagens entre produtores e consumidores utilizando o estilo de comunicação PTP, adotando um nível de confiabilidade PNT.

A Figura 4-11 apresenta o modelo PN para este componente. A diferença básica desse componente para o componente anterior é o mecanismo de persistência inserido no modelo. O lugar *PRV_Disks* representa o recurso de disco do

provider. Esse lugar possui uma marcação inicial definida por um parâmetro PRV_nDisk, que indica o número de sistemas de disco presentes no *provider*. Um número maior que um indicaria um sistema utilizando um *storage*, onde várias gravações e leituras podem ser realizadas simultaneamente.

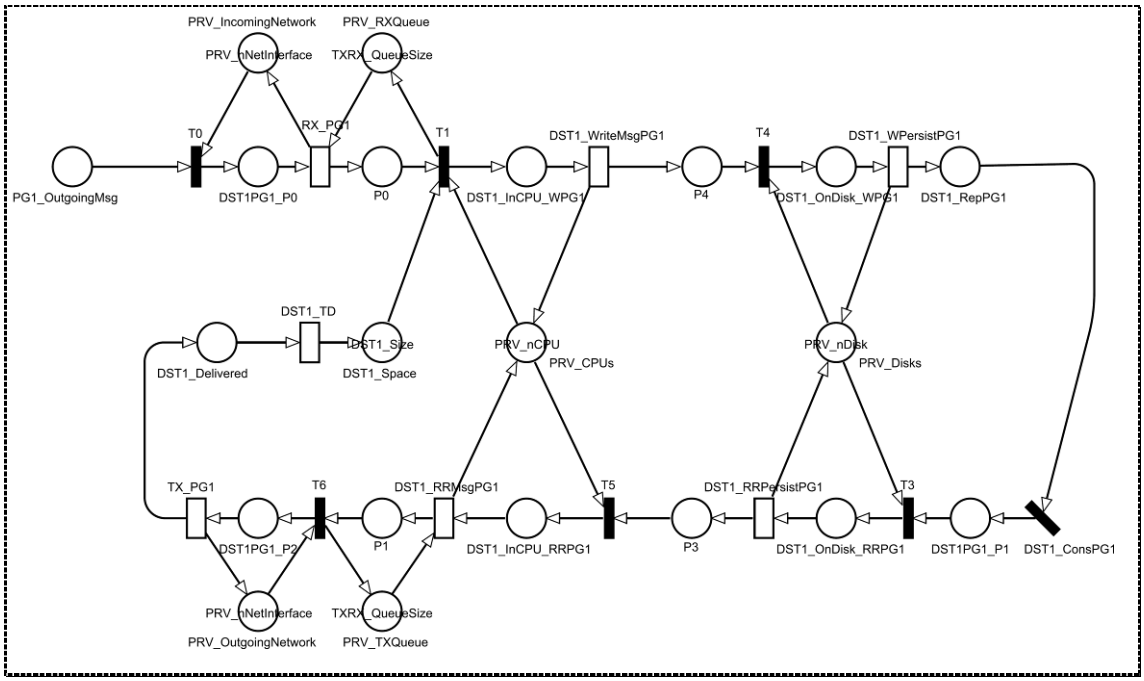


Figura 4-11. Componente de desempenho: *JDPTP-PNT.*

Os lugares *DST1_OnDisk_WPG1* e *DST1_OnDisk_RRPG1* representam as mensagens que se encontram utilizando o sistema de disco para escrita e para leitura/remoção, respectivamente.

As transições temporizadas *DST1_WPersistPG1 e DST1_RRPersistPG1* representam o tempo gasto pelas mensagens dentro do recurso. Essas transições possuem parâmetros para *delay* e semântica de temporização. O *delay* de cada transição é representadopelos parâmetros *DiskTime_WPersistPG1* e *DiskTime_RRPersistPG1; e*nquanto que a semântica de temporização de ambas é *infinite_server*.

JDPS-NPNT

Este componente é responsável pelo encaminhamento das mensagens entre produtores e consumidores utilizando o estilo de comunicação *Pub/Sub* e adotando um nível de confiabilidade NPNT.

A Figura 4-12 apresenta o modelo PN para este componente, o qual possui apenas duas diferenças em relação ao modelo do componente *JDPTP-NPNT*.

A primeira é a multiplicidade do arco que interliga a transição temporizada *DST1_WriteMsgPG1* ao lugar *DST1_RepPG1*. De acordo com a especificação JMS, para cada mensagem recebida em um canal *Pub/Sub* é armazenada uma cópia dela nas filas que representam os assinantes (consumidores) ativos no canal (tópico). Isso leva a uma multiplicação do número de mensagens e está modelado pelo parâmetro *DST1_nCons* (número de consumidores ativos) associado ao arco, gerando a multiplicidade de *tokens* (mensagens) no repositório.

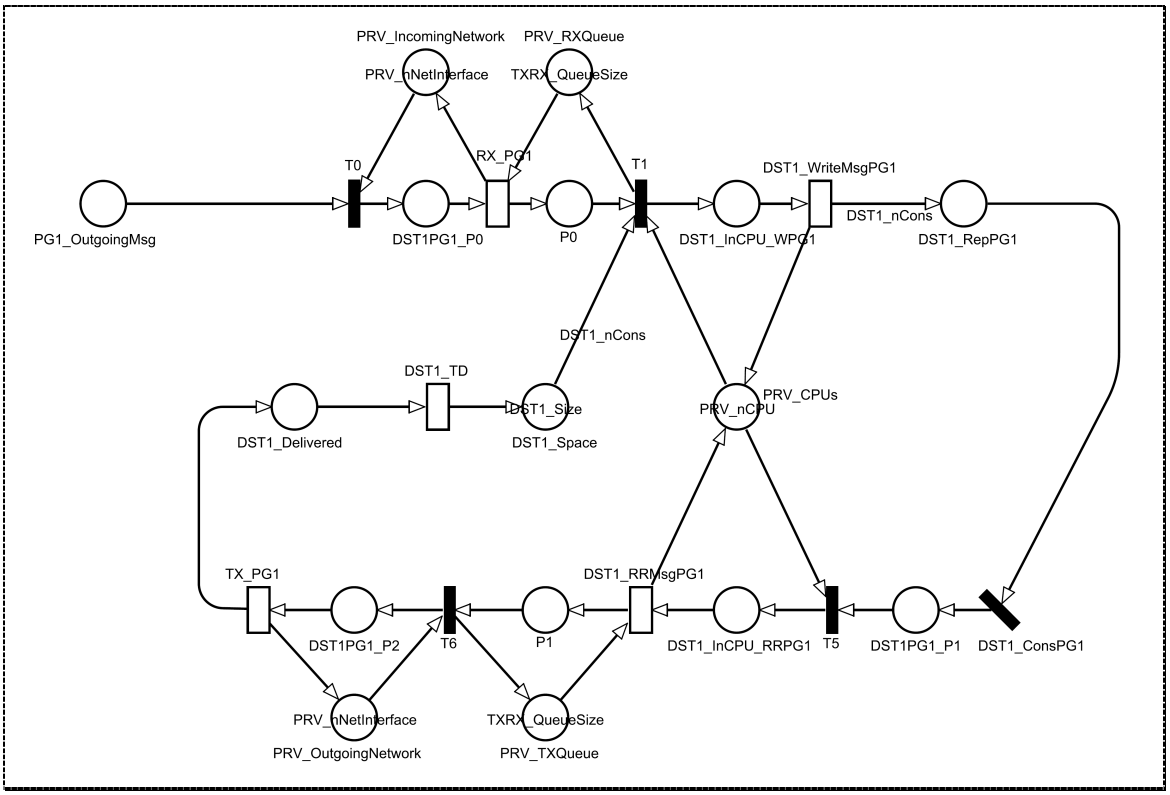


Figura 4-12. Componente de desempenho: *JDPS-NPNT*.

A segunda alteração refere-se à alocação de espaço no *destination* (recurso), fazendo com que, para cada mensagem a ser processada, o *destination* reserve uma quantidade de espaço suficiente para alocar as múltiplas cópias. Para isso, basta alterar a multiplicidade do arco interligando o lugar *DST1_Space* à transição imediata *T1* para o valor de *DST1_nCons*.

JDPS-PNT

Este componente é responsável pelo encaminhamento das mensagens entre produtores e consumidores utilizando o estilo de comunicação *Pub/Sub*, adotando um nível de confiabilidade PNT. A Figura 4-13 apresenta o modelo PN para este componente.

Esse modelo é semelhante ao modelo do componente *JDPTP-PNT*, com as mesmas ressalvas e diferenças detalhadas na apresentação do componente *JDPS-NPNT* em relação ao seu equivalente no estilo PTP. A única consideração adicional é que neste caso cada mensagem enviada ao canal consome recursos de disco para persistir cada cópia da mensagem na fila de saída, representando cada assinatura ativa.

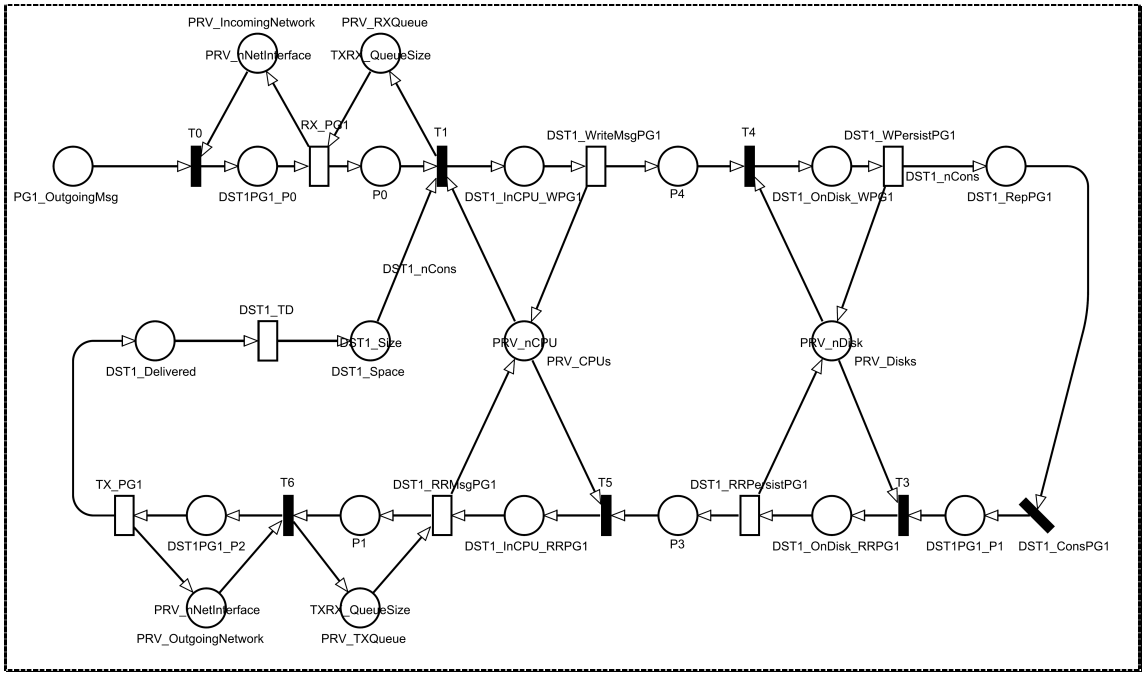


Figura 4-13. Componente de desempenho: *JDPS-PNT.*

Porém, diferentemente do outro modelo *Pub/Sub* (ver Figura 4-12) as cópias das mensagens são realizadas após a transição temporizada *DST1_WPersistPG1*, que contabiliza o custo de persitência da mensagem original, enquanto que a transição *DST1_RRPersistPG1* modela o custo com a persistência das cópias[2].

4.2.3 Componente Básico: Consumidor

No modelo de carga, as entidades consumidoras foram classificadas em tipos síncronos e assíncronos, conforme definido pela especificação JMS. No caso dos consumidores síncronos, dois outros aspectos se tornam relevantes para determinar qual componente de desempenho deve ser utilizado para implementar o modelo de carga: primitiva de recepção e evento associado.

A primitiva de recepção (*Block*, *NoWait*, *TimeOut*) estabelece o comportamento da chamada do método *receive*. O evento determina quando o consumidor deve enviar uma chamada *receive* para o *provider*. Ou seja, o componente de desempenho que deve ser utilizado para implementar a entidade consumidora do modelo de carga é determinado pelo tipo concreto associado à entidade (*JASCons*, *JSCons-Block*, *JSCons-NoWait*, *JSCons-TimeOut*) somado ao evento gerador, no caso de consumidores síncronos (*JSCons**).

Um ponto a considerar é que, segundo a especificação JMS, os consumidores associados a um mesmo canal devem ser do mesmo tipo básico: síncrono ou assíncrono. A biblioteca amplia essa característica, limitando não somente ao tipo básico, mas também ao tipo concreto somado ao evento gerador. A motivação para essa simplificação é tornar os modelos menos complexos, facilitando a avaliação. Além disso, em geral, nas aplicações reais os

[2] Esse comportamento está definido na especificação JMS, quando ela afirma que a mensagem deve ser recebida pelo MOM, confirmada com o produtor, para somente depois ser copiada para as filas de saída de cada assinatura ativa.

consumidores de um canal possuem comportamento semelhante e podem ser aproximados para efeito de planejamento de capacidade sem prejuízo do estudo.

JSCons-NoWait-Poisson

Componente de desempenho responsável pelo consumo síncrono de mensagens com tempo entre chamadas do método JMS *receiveNoWait* exponencialmente distribuído (distribuição de *Poisson*).

A Figura 4-14 apresenta o modelo PN que implementa este componente. Neste modelo estão representados três elementos que fazem parte do modelo do canal de comunicação: *DST1_RepPG1*, lugar que representa o repositório de mensagens aguardando consumo; *DST1_ConsPG1*, transição imediata que representa o primeiro ponto de interconexão com o componente consumidor; *DST1_TD*, segundo ponto de interconexão utilizado como recurso de modelagem para liberar recursos e medir a produtividade do canal.

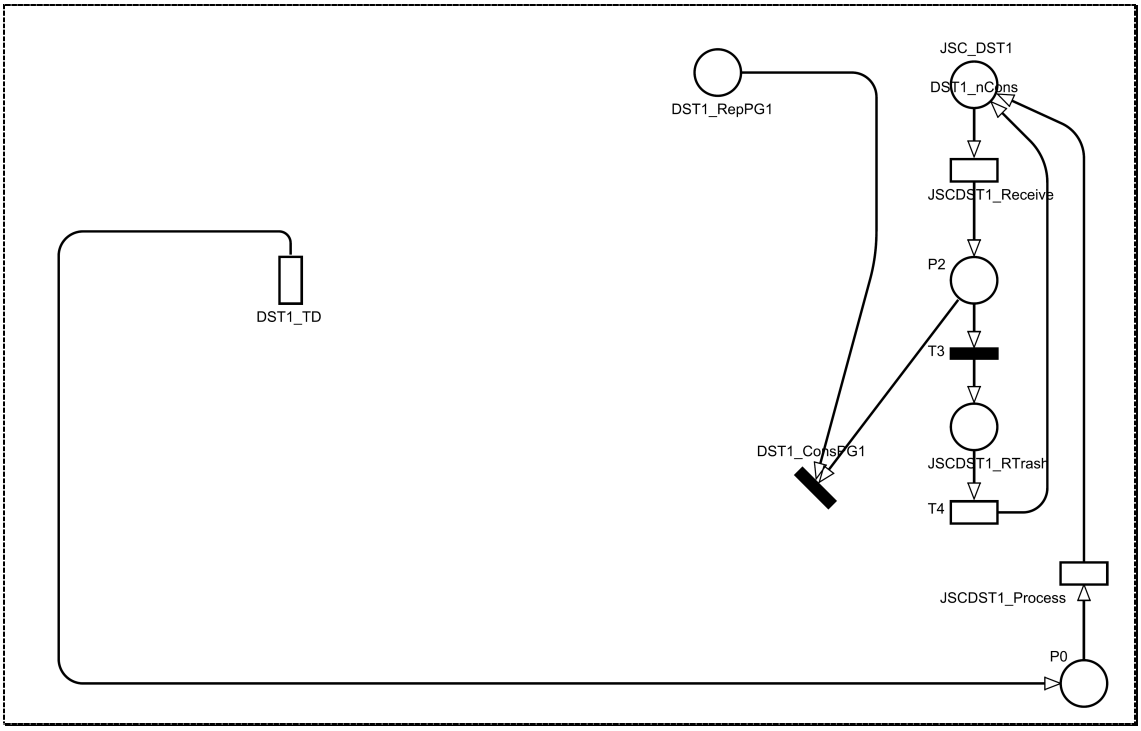


Figura 4-14. Componente de desempenho: *JSCons-NoWait-Poisson*.

A parte do modelo que representa o consumidor possui quatro lugares e quatro transições (uma imediata e três temporizadas). O lugar *JSC_DST1* representa os consumidores prontos para consumir mensagens. A marcação inicial desse lugar é o parâmetro *DST1_nCons*, representando o número de instâncias deste tipo de consumidor que está associado ao canal *DST1*.

A transição temporizada *JSCDST1_Receive* representa a geração do evento que indica ao consumidor o momento de realizar a chamada do método *receive*. O parâmetro *JSCDST1_ReceiveTime* representa o *delay* da transição; enquanto que a semântica de temporização é definida como *infinite_server*, indicando a concorrência e independência dos consumidores.

A transição imediata *T3* tem a missão de verificar se existe mensagem a ser consumida; se não tiver, ela dispara. Isso é implementado através de uma função de guarda na transição, conforme apresentado pelo Quadro 4-1.

Quadro 4-1. Componente *JSCons-NoWait-Poisson*: Função de guarda.

```
(#DST1_RepPG1=0)
```

A transição temporizada *T4* é um recurso de modelagem para medir a taxa com que as chamadas *receive* retornam sem mensagem. A monitoração dessa taxa indica se o consumidor está mais rápido do que necessário, permitindo o seu ajuste, evitando com isso a sobrecarga do *provider*. Esse é um recurso útil no gerenciamento da capacidade.

O lugar *P2* representa as chamadas *receive* aguardando mensagens, enquanto que o lugar *JSCDST1_RTrash* representa as chamadas que não obtiveram sucesso.

O lugar *P0* e a transição temporizada *JSCDST1_Process* são responsáveis por implementar o processamento da mensagem quando recebida pelo consumidor. Esse aspecto é importante pois, dependendo do tempo desta ação, isto pode fazer com que a carga submetida ao *provider* pelo consumidor seja menor, possibilitando um aumento da latência de entrega das mensagens. Isso pode exigir do *Gerente de Capacidade* um aumento no número de entidades consumidoras para compensar esse efeito.

Os parâmetos da transição temporizada de processamento da mensagem são *JSCDST1_ProcessTime* (delay) e *infinite_server* (semântica de temporização). Se não for interessante expressar no componente o tempo de processamento da mensagem, essa transição pode ser trocada por uma imediata.

JSCons-TimeOut-Poisson

Componente de desempenho responsável pelo consumo síncrono de mensagens com tempo entre chamadas do método JMS *receive(timeout)* exponencialmente distribuído. A Figura 4-15 apresenta o modelo PN para este componente.

A diferença desse modelo em relação ao do componente *JSCons-NoWait-Poisson* (ver Figura 4-14) é a presença de uma transição temporizada com tempo determinístico (*T0*) em vez da transição imediata anterior. Os modelos GSPN não permitem tal transição, mas uma extensão das GSPN denominada de DSPN (*Deterministic Stochastic Petri Net*) adiciona este comportamento determinístico ao formalismo.

Essa transição possui um parâmetro denominado *JSCDST1_ReceiveTimeOut* e representa o tempo pelo qual a chamada *receive* (*timeout*) aguarda pela mensagem. Após esse tempo, a transição dispara indicando que não obteve sucesso no consumo. Da mesma forma que o outro modelo, a taxa com que esse evento (falha) ocorre pode e deve ser medido.

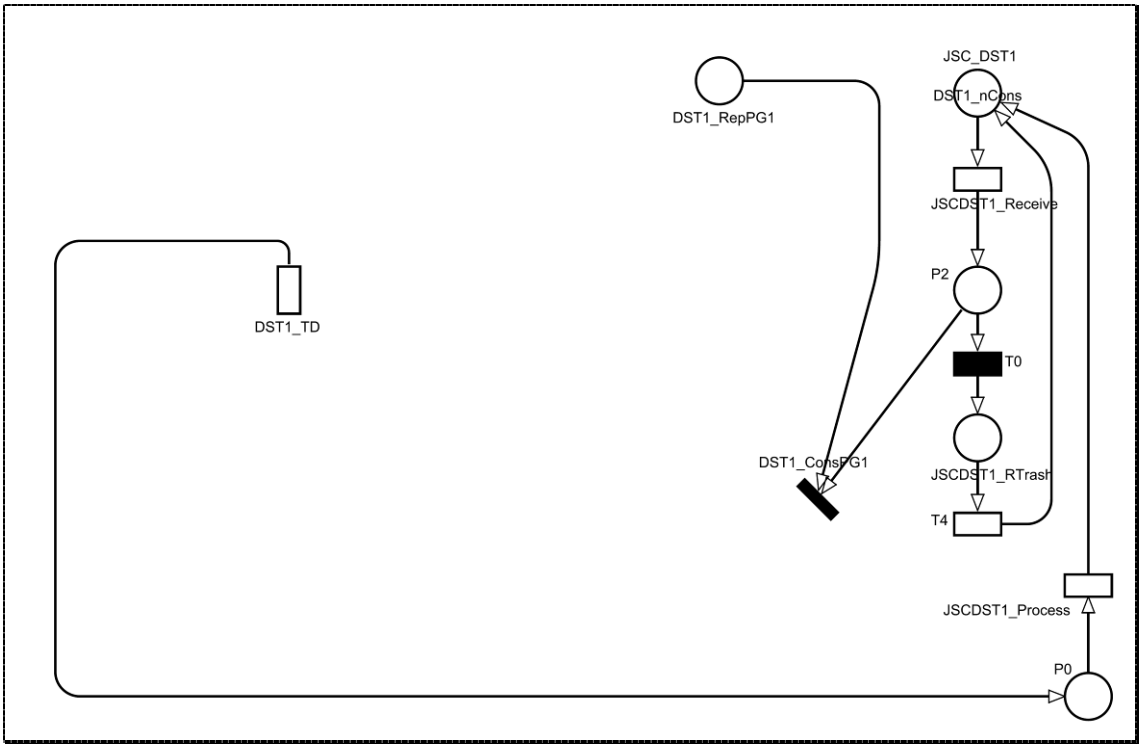


Figura 4-15. Componente de desempenho: *JSCons-TimeOut-Poisson.*

JSCons-Block-Poisson

Componente de desempenho responsável pelo consumo síncrono de mensagens com tempo entre chamadas do método JMS *receive* exponencialmente distribuído. A Figura 4-16 apresenta o modelo PN para este componente.

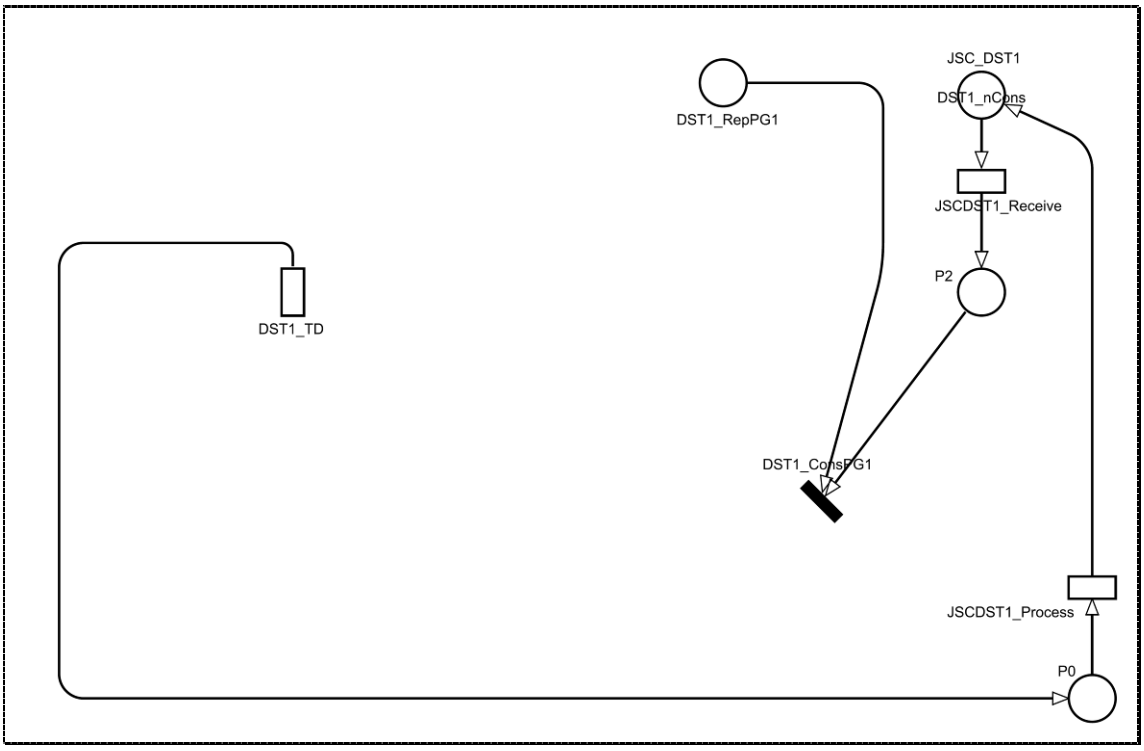


Figura 4-16. Componente de desempenho: *JSCons-Block-Poisson.*

A chamada nesse caso é bloqueante e deve retornar quando uma mensagem for lida. Esse modelo é verdadeiramente síncrono, pois os demais modelos "síncronos" previstos na especificação JMS podem retornar sem mensagem alguma. Nenhum recurso foi adicionado aos modelos, apenas retirada a condição de retorno antecipado.

JASCons

Este componente é responsável pelo consumo assíncrono de mensagens. A Figura 4-16 apresenta o modelo PN para este componente. Este modelo é implementado com dois lugares e uma transição. A transição representa o processamento da mensagem.

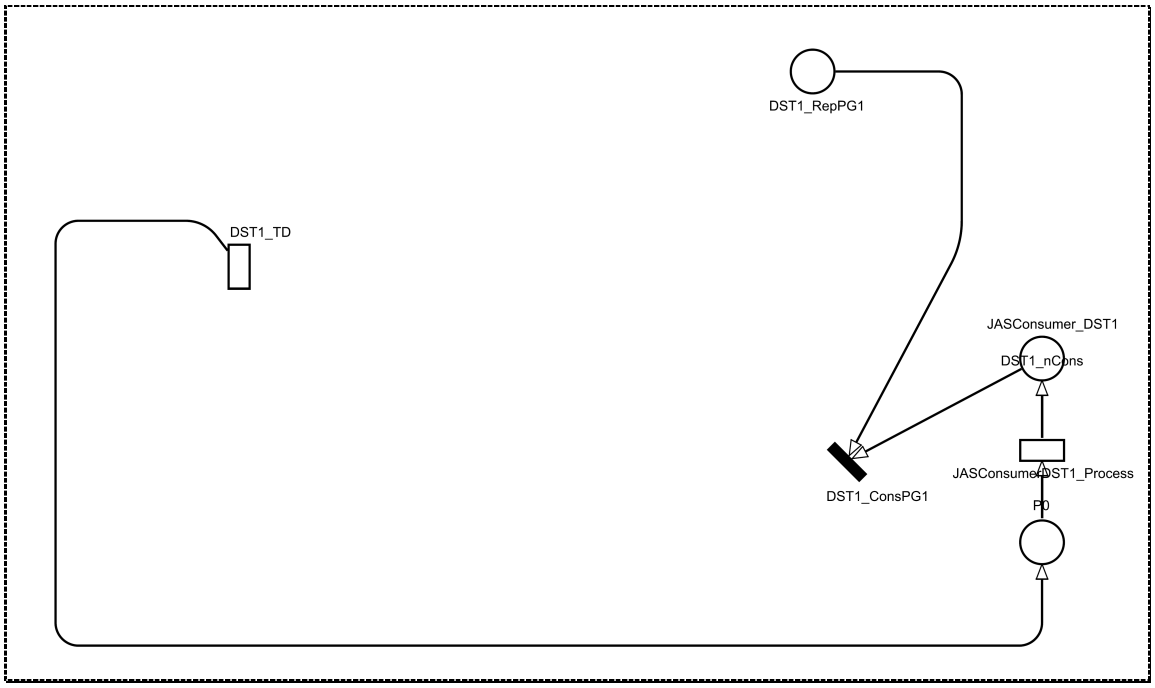


Figura 4-17. Componente de desempenho: *JASCons.*

O lugar *JASConsumer_DST1* possui marcação inicial definida pelo parâmetro *DST1_nCons*. O motivo dessa simplicidade do modelo é que, durante a entrega da mensagem, o *provider* fica responsável por todo o trabalho. A entrega é totalmente assíncrona, sem controle da aplicação consumidora. Ela apenas implementa a interface *MessageListener* chamada por uma *thread* do *provider* executando no cliente.

4.2.4 Componentes Complexos

Uma vez que os componentes básicos da biblioteca estão apresentados, pode-se aumentar a complexidade dos componentes sem prejuízo ao entendimento. Nesta seção são apresentados novos grupos de produção, canais de comunicação e consumidores, normalmente definindo extensões dos componentes básicos, representando, portanto, a capacidade de extensão do *toolkit*.

JProdGroup-Poisson-Sync

Este componente faz uma extensão do *JProdGroup-Poisson* implementando o envio síncrono de mensagens por parte do produtor. A Figura 4-18 apresenta o

modelo PN referente a este componente. A única alteração em relação ao seu par assíncrono é o arco de retorno, que em agora sai da transição *DST1_WriteMsgPG1*.

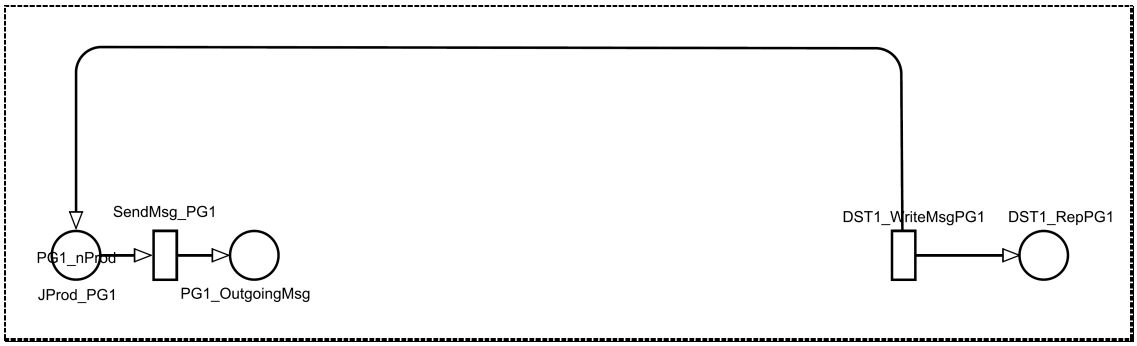


Figura 4-18. Componente de desempenho: *JProdGroup-Poisson-Sync.*

É importante ressaltar que, se este produtor estiver associado a um canal com persitência (ex. *JDPTP-PNT*, *JDPS-PNT*), o arco de retorno deve partir da transição *DST1_WPersistPG1* (ver Figura 4-11). Dessa forma, está sendo considerada no envio da mensagem a persistência apenas da mensagem original, e não de todas as cópias. O tempo das cópias é considerado na seqüência de transições que realizam a entrega das mensagens. Generalizando, o arco de retorno deve partir sempre da última transição temporizada que armazena a mensagem no repositório *DST1_RepPGn*.

JProdGroup-Connect

Componente de desempenho responsável pela produção de mensagens a partir da recepção de uma mensagem de outra linha de comunicação. A Figura 4-19 apresenta o modelo PN que implementa este componente.

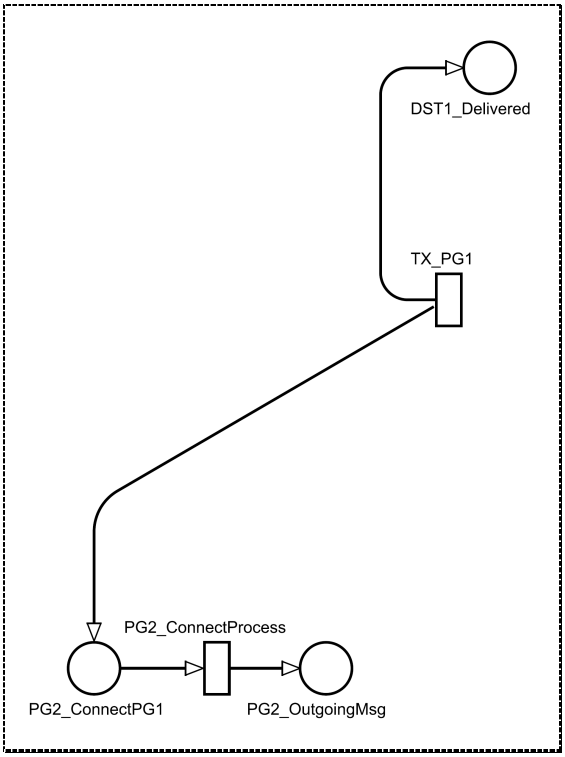


Figura 4-19. Componente de Desempenho: *JProdGroup-Connect*.

Essa implementação resolve a especificação do modelo de carga onde um evento é modelado do lado direito de uma entidade consumidora, indicando que a chegada de uma mensagem o dispara. Logo, é vital para a tradução da maioria das linhas de comunicação do modelo de carga em um processo de integração entre aplicações corporativas.

O novo grupo de produção (*PG2*) especificado pelo *JProdGroup-Connect* se conecta através da transição temporizada de transmissão da mensagem dentro do grupo de produção de origem (*TX_PG1*). Esta é a última transição antes da mensagem ser colocada no lugar *DST1_Delivered* indicando que a mensagem foi entregue ao consumidor, disparando o evento que está indicado como gerador de novas mensagens no *PG2*.

O modelo deste componente especificamente é representado por dois lugares (*PG2_ConnectPG1* e *PG2_OutgoingMsg*) e uma transição temporizada (*PG2_ConnectProcess*). O primeiro lugar representa a chegada da mensagem pela conexão com o *PG1*. O outro é o ponto de interconexão com o canal de comunicação. A transição representa o tempo de processamento que a entidade precisa para enviar a nova mensagem. Se não houver necessidade de expressar esse tempo, essa transição pode ser substituída por uma imediata. Os parâmetros da transição temporizada são *PG2_ConnectTime* (*delay*) e *infinite_server* (semântica de temporização).

JProdGroup-FX e JProdGroup-FX-Sync

Este componente é responsável por produzir mensagens seguindo uma função de distribuição exponencial com a taxa variável no tempo. Esse exemplo demonstra a possibilidade de construção de grupos de produção com comportamento bem distinto em relação ao que já foi mostrado ao longo dessa biblioteca. A Figura 4-20 apresenta o modelo PN que implementa esse componente.

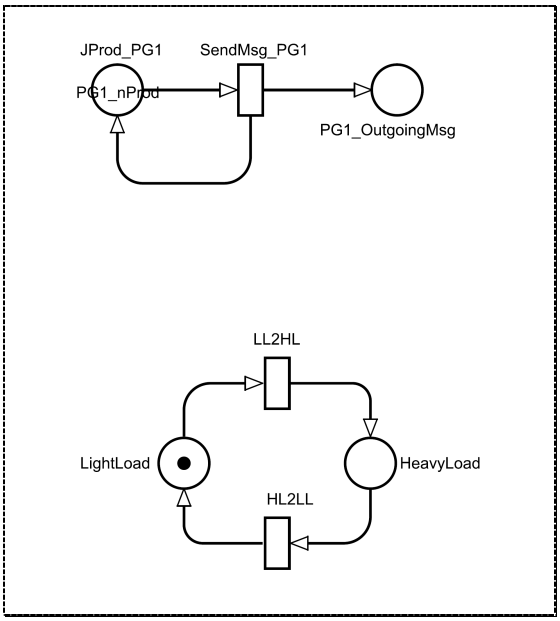


Figura 4-20. Componente de desempenho: *JProdGroup-FX*.

Esse modelo associa um *JProdGroup-Poisson* à uma máquina de estado que altera o parâmetro de *delay* da transição temporizada (exponencial)

SendMsg_PG1 ao longo do tempo, simulando dois estados: *LightLoad* e *HeavyLoad*. Esses estados representam situações de baixa e alta carga respectivamente, podendo representar um grupo produtor que alterna situações de intensidade de carga ao longo do tempo.

Para conseguir este efeito o parâmetro de *delay* da transição *SendMsg_PG1* possui um valor dependente de marcação, conforme apresentado no Quadro 4-2. Essa fórmula indica que quando a máquina de estado estiver indicando baixa carga (token no lugar *LightLoad*), o *delay* da transição é determinado pelo parâmetro *PG1_SendTimeLL*, caso contrário, o *delay* passa a ser definido pelo parâmetro *PG1_SendTimeHL*.

Quadro 4-2. *JProdGroup-FX*: Parâmetro *delay* dependente de marcação.

```
IF #LightLoad=1 : PG1_SendTimeLL ELSE PG1_SendTimeHL;
```

A Figura 4-21 apresenta o modelo PN do componente *JProdGroup-FX-Sync*, que implementa envio síncrono de mensagens para o *JProdGroup-FX*.

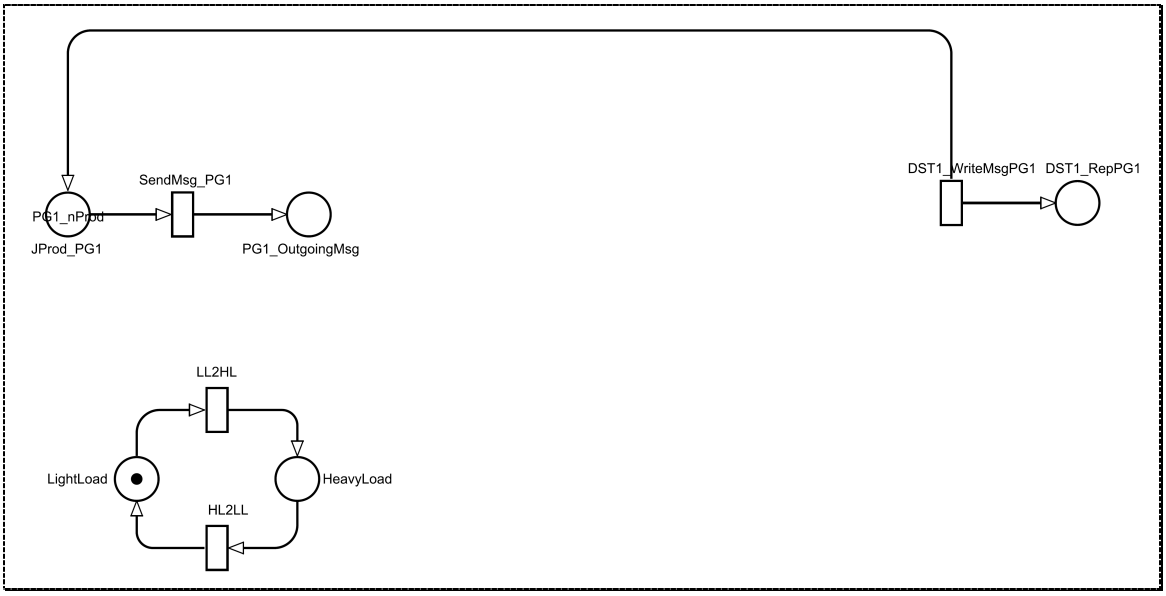


Figura 4-21. componente de desempenho: *JProdGroup-FX-Sync*.

JDPTP-NPNT-MPG

Componente de desempenho responsável pela implementação de um canal de comunicação (*DST1*) com estilo de mensagem PTP e confiabilidade NPNT, associado a duas linhas de comunicação simultâneas, ou seja, dois grupos produtores (*PG1* e *PG2*). A Figura 4-21 apresenta o modelo PN referente a este componente.

É importante analisar que todos os lugares que representam recursos compartilhadados no *provider* (rotulados com o prefixo *PRV*) são representados uma única vez no modelo. Além disso, eles são alocados, pelas transições imediatas, em ambas as linhas de comunicação, representando o compartilhamento do recurso.

O lugar *DST1_Space* que representa o recurso de espaço livre no *destination* também aparece uma única vez no modelo, indicando que ambos os grupos produtores compartilham o mesmo espaço para armazenar mensagens.

O lugar *DST1_Delivered* e a transição *DST1_TD* representam a finalização da entrega da mensagem ao consumidor e a liberação do espaço no *destination*, respectivamente. Portanto, também são modelados uma única vez.

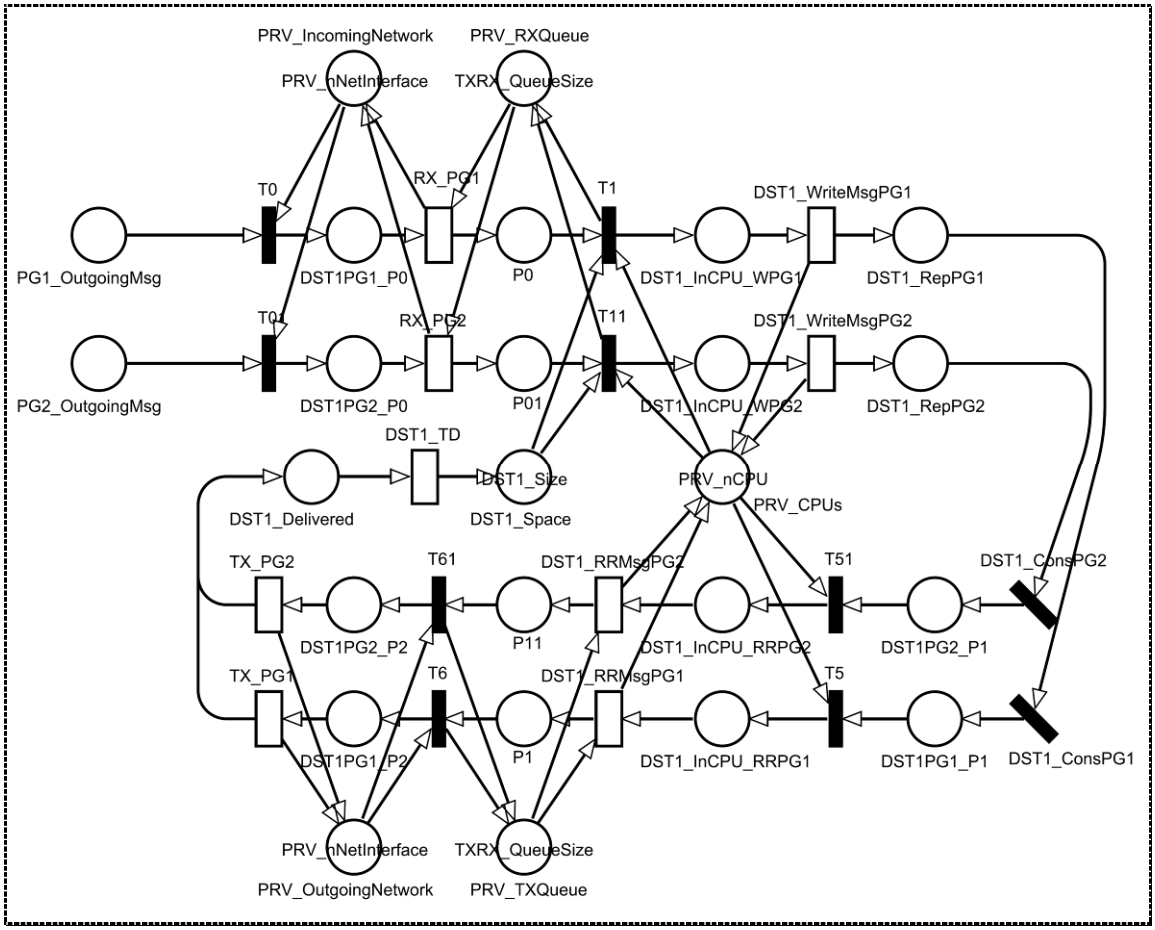


Figura 4-22. Componente de desempenho: *JDPTP-NPNT-MPG.*

Os demais elementos do modelo são repetidos para cada *PG*. A justificativa de tal fato é que cada linha de comunicação possui características que podem demandar recursos do *provider* distintos. Logo, as transições temporizadas que representam a demanda por recursos precisam ser duplicadas para que se possam definir valores diferentes para a carga de trabalho imposta por cada *PG*.

4.2.5 Orientações Adicionais sobre a Construção do Modelo de Desempenho

A construção do modelo de desempenho é realizada através da composição dos vários componentes de desempenho com objetivo de implementar o modelo de carga. O resultado quase sempre induz a um modelo de dimensões razoavelmente grande.

Quanto maior o número de lugares, transições e *tokens* circulantes na PN, maior será o espaço de estados gerado a partir desse modelo. O resultado prático disso é que, quase sempre, acontecerá o fenômeno conhecido como explosão de espaço de estado, indicando um número de estados não tratável computacionalmente. Isso leva a resolução dos modelos através de técnicas de simulação.

Diante desse cenário, é importante observar que sempre que possível, as linhas de comunicação devem ser reduzidas. Outra opção é realizar um planejamento de capacidade particionado por linha de comunicação e não do processo de integração como um todo.

É importante ressaltar que quando uma linha de comunicação dispara um evento para outra linha, a primeira precisa ser implementada, avaliada, tendo o seu resultado levado como parâmetro de entrada para compor a próxima linha, e assim sucessivamente. Ao final, o gerente de capacidade tem condições de reunir os resultados de todas as linhas (em termos de demandas por recursos necessitados) estimando o planejamento total.

Outro ponto a considerar é, sempre que possível, o Avaliador deve utilizar componentes que tornem o modelo final limitado. Caso não seja possível, é recomendável que ele intervenha no modelo resultante para torná-lo o mais limitado possível, reduzindo o custo computacional para sua resolução.

4.3 Ferramentas

Esta seção tem como objetivo orientar quais os tipos de ferramentas são necessárias para apoiar o *toolkit*, além de apresentar as escolhas realizadas durante a execução deste trabalho.

As ferramentas foram divididas em categorias que direcionam sua utilização nas diversas atividades e passos do processo.

4.3.1 Monitoração da Utilização de Recursos

O objetivo desta categoria de ferramentas é monitorar os recursos demandados no MOM quando do encaminhamento das mensagens. Esses recursos foram identificados no processo durante a atividade denominada *Entender o Ambiente*.

Esta categoria fornece suporte às atividades *Caracterizar a Carga de Trabalho* e *Parametrizar, Validar e Calibrar o Modelo*. Ela é normalmente formada por ferramentas disponibilizadas pelos fabricantes da plataforma operacional e do MOM, uma vez que se destina a monitorar recursos como CPU, disco, rede e fila.

Na caracterização da carga de trabalho, esta categoria de ferramentas auxilia na identificação de componentes básicos de carga, classificando-os quanto à utilização de recursos computacionais da plataforma.Na parametrização, essas ferramentas são responsáveis por monitorar o *provider* avaliando a utilização dos recursos de CPU, disco e rede (recursos modelados nos componentes de desempenho em PN), com o intuito de obter os parâmetros de entrada para o modelo de desempenho.

Na validação, ela auxilia na comprovação que os recursos do sistema real foram comprometidos com uma demanda equivalente a apresentada pelos modelos de desempenho, validando o comportamento do modelo em relação ao sistema real.

As duas ferramentas utilizadas para monitorar a utilização dos recursos foram o *Microsoft Performance & Reliability Monitor* e o *JBoss Logging Monitor*.

4.3.2 Geração de Carga Artifical

O objetivo desta categoria de ferramentas é gerar carga de trabalho artificial para o MOM, simulando as cargas especificadas durante a atividade *Caracterizar a Carga de Trabalho*. Uma característica útil neste tipo de ferramenta é a capacidade de avaliar métricas de resposta e produtividade ao longo dos testes.

Esta categoria fornece suporte a atividade *Parametrizar, Validar e Calibrar o Modelo*. Ela atua em conjunto com as ferramentas de monitoração de recursos proporcionando ao *Avaliador* determinar a carga exata de trabalho a ser submetida ao MOM em cada momento.

Na parametrização ela auxilia na reprodução de cargas especialmente escolhidas para determinar a demanda de recurso de cada tipo de componente básico de carga (mensagem).

Na validação, o *Avaliador* escolhe cenários para determinar se o modelo de desempenho possui o mesmo comportamento do sistema real. Nesse momento, esta categoria de ferramenta auxilia a reproduzir esses cenários de carga, devendo possuir mecanismos bastante flexíveis para este fim.

Diversas ferramentas de mercado existem com essa finalidade, algumas de código aberto, outras apenas com versão comercial. Das ferramentas de código aberto estudadas, a de maior destaque é a *Apache JMeter* [Apache 2006]. Porém, ela se mostrou pouco flexível, e motivou a criação de uma nova ferramenta denominada *JMSMeter* durante o desenvolvimento deste trabalho, tornando-se parte integrante deste *toolkit*.

JMSMeter

O JMSMeter é uma ferramenta de geração de carga de trabalho que consegue reproduzir os componentes de desempenho desenvolvidos: JProdGroup-Poisson, JDPTP-NPNT, JDPS-PNT, JSCons-NoWait-Poisson, JASCons etc. Ela foi desenvolvida em Java e portanto é suportada em várias plataformas.

Nesta ferramenta podem-se incluir novas métricas a serem capturadas, escolhendo o ponto exato de medição de cada uma delas, permitindo o controle sobre o que está sendo medido.

A geração de números aleatórios seguindo uma distribuição exponencial, elemento que faltava no JMeter, é uma importante contribuição para o sucesso das medições e validações, fornecendo inclusive suporte a ajuste contínuo para garantir a geração de carga numa distribuição de *Poisson*, independente do tempo de resposta do *provider*. Esse recurso garante que possa ser reproduzido o comportamento do *JProdGroup-Poisson* (assíncrono), por exemplo. Para garantir ao *Avaliador* que objetivo de manter uma taxa constante de geração está sendo cumprida, são disponibilizadas métricas de suporte (*SendDelay* e *ReceiveDelay*).

Outro ponto relevante, e que se constitui numa contribuição para a agilidade do procedimento de medição durante a validação dos modelos, é a apuração estatística dos resultados das métricas. A ferramenta disponibiliza um resumo estatístico de cada métrica e o cálculo de cada amostra (*sample*) que originou o resultado da métrica.

4.3.3 Modelagem

O objetivo desta categoria de ferramentas é facilitar a construção dos modelos de carga de trabalho e de desempenho, e sua posterior avaliação de desempenho. Ela fornece suporte a todas as atividades do *Avaliador*: *Caracterizar Carga de Trabalho*; *Desenvolver Modelo de Desempenho*; *Parametrizar, Validar e Calibrar o Modelo*; e *Avaliar Cenários*.

Na caracterização da carga, auxilia na construção do modelo de carga. No desenvolvimento do modelo de desempenho, implementa o formalismo das redes de Petri (PN). Na validação, permite avaliar os modelos PN para obter as métricas a serem comparadas com a medição do sistema real. Na calibragem do modelo, permite desenvolver aproximações de distribuições empíricas utilizando *Phase Type Distributions* (ver Apêndice B). Por fim, na avaliação dos cenários, permite realizar as simulações necessárias para a obtenção dos resultados que são utilizados no planejamento de capacidade.

As ferramentas utilizadas para modelagem foram o Microsoft Office Visio e o TimeNet. Este *toolkit* disponibiliza um *stencil* (.vss) com todos os componentes para construção do modelo de carga apresentados na Seção 4.1.

4.4 Considerações Finais

A biblioteca de componentes foi apresentada neste capítulo, incluindo um conjunto de orientações para o desenvolvimento do planejamento e gerenciamento da capacidade de plataformas de MOM que implementam a especificação JMS.

Inicialmente, foram apresentados os componentes de carga que auxiliam na construção do modelo de carga de trabalho, e, posteriormente, as implementações equivalentes em redes de Petri, formando o conjunto denominado de componentes de desempenho.

Durante esse capítulo foi possível observar a importância da criação de um modelo intermediário (modelo de carga) através de uma notação específica, onde elementos fundamentais da integração são extraídos de um diagrama de sequência, e possibilitam que a atividade de *Desenvolver o Modelo de Desempenho* possa ser realizada de forma sistematizada, possibilitando extensões da biblioteca, aumentando a flexibilidade do *toolkit* e tornando o processo mais fácil de ser utilizado.

CAPÍTULO

5

JMSCapacity: Passo a passo

"As pessoas preferem viver com um problema que não pode resolver do que aceitar uma solução que não pode compreender."

Woolsey and Swanson (1975).

Com o objetivo de validar o processo e facilitar a utilização deste *toolkit*, este capítulo apresenta um guia passo a passo da utilização do JMSCapacity para auxiliar no planejamento e gerenciamento de capacidade de um MOM em um cenário de integração de aplicações corporativas.

Inicialmente é definido um cenário ilustrativo de integração que será utilizado nesse guia, de forma a explorar de forma simples, mas completa, a maioria das facilidades do *toolkit*. Em seguida, o processo é aplicado ao cenário, demonstrando como utilizá-lo.

5.1 Descrição do Cenário

O cenário escolhido é baseado em [Calabria 2004] onde é apresentado um cenário real de utilização do Hermes (MOM proposto pelo autor) para integrar aplicações corporativas em um ambiente de empresas de varejo (supermercados, por exemplo). Nesse cenário, aplicações que estão em execução nos pontos de venda (PDVs) precisam se integrar com uma solução de gestão administrativa (ERP) que executa na retaguarda. O detalhamento do cenário é definido por este trabalho com o intuito de exemplificar a utilização do *toolkit*.

Podem-se identificar três entidades nessa integração: ponto de venda (PDV), loja (SM) e matriz (HQ). A integração entre as lojas e a matriz é intermediada por um sistema de troca de mensagens (MOM). A Figura 5-1 apresenta uma

visão geral da solução de integração. Nesse cenário hipotético, cada loja padrão possui cerca de sessenta PDVs que se comunicam com a matriz através da rede interna da loja. A Figura 5-2 apresenta o detalhamento de uma loja.

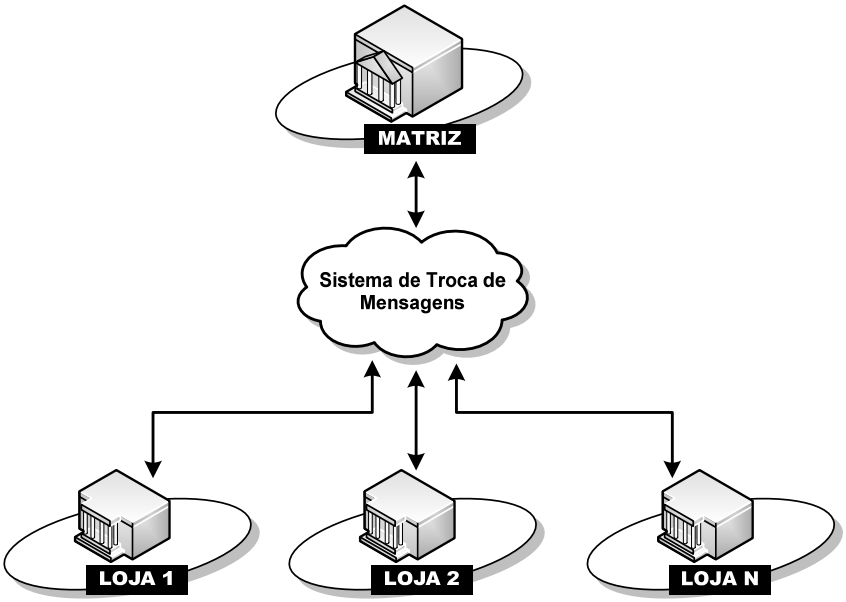


Figura 5-1. Cenário: visão geral.

Os PDVs são responsáveis em registrar as vendas aos clientes, precisando, para isso, estar continuamente com a tabela de preços atualizada. Diante desse contexto, duas interações entre as aplicações são identificadas: *Registro de Vendas* e *Atualização de Preços*.

O *Registro de Vendas* faz parte do processo de negócio responsável em manter a matriz informada em tempo real da situação das vendas em suas lojas. Isso garante que decisões de compra, reabastecimento e logística, por exemplo, estejam alinhados com a situação real de venda.

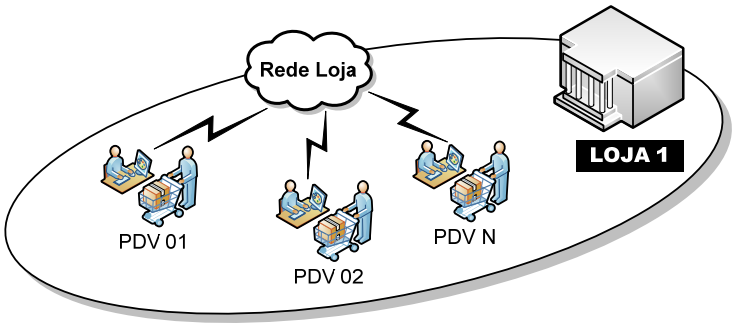


Figura 5-2. Cenário: detalhamento da loja.

A *Atualização de Preços* é uma atividade do processo de negócio que garante aos gestores de venda da empresa a capacidade de determinar para cada loja uma política contínua de ajustes de preço, baseada em pesquisas de mercado de concorrentes na região próxima, garantindo com isso um equilíbrio entre a competitividade e a margem de lucro.

O restante do capítulo apresenta a utilização do JMSCapacity para realizar o planejamento de capacidade do MOM responsável por essa integração. Para isso, dedica uma seção para cada uma das atividades do processo.

5.2 Definição de Objetivos

Na primeira atividade do processo, o *Integrador de Sistemas* precisa definir o *Service Level Objective* (SLO) e o *Plano de Integração*. Ele possui entendimento pleno do cenário desejado para o serviço de integração, envolvendo os PDVs e o ERP.

O SLO precisa ser definido para cada interação do serviço de integração, pois cada um pode requerer níveis de serviço diferentes. Se a interação utilizar mais de um canal de comunicação, pode-se definir o SLO para cada canal ou para a interação como um todo. Dessa forma, para simplificar a definição do SLO este guia detalha primeiro o *Plano de Integração*.

5.2.1 Plano de Integração

Na primeira interação, *Registro de Vendas*, os PDVs enviam para o ERP, na matriz, cada item registrado nos caixas, permitindo que a *Gestão de Vendas* acompanhe em tempo real o andamento das vendas. A Figura 5-3 apresenta uma visão geral dessa interação.

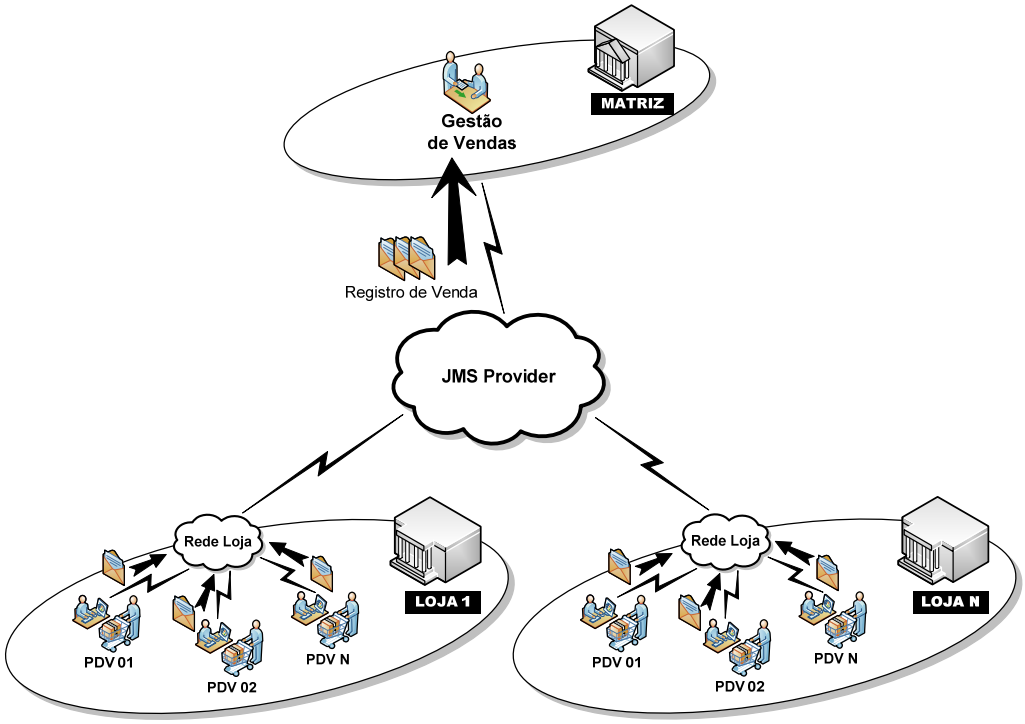


Figura 5-3. Interação *Registro de Vendas*: visão geral.

No PDV, a aplicação *Registrador de Vendas* é responsável em informar os itens registrados ao *Processador de Vendas* (no servidor de aplicação JBoss, na matriz), módulo responsável por comunicar ao ERP os itens que estão sendo vendidos. A Figura 5-4 apresenta o diagrama de implantação que representa a distribuição dos componentes de software envolvidos, indicando os nós onde cada um está em execução.

Existe uma necessidade que cada um dos PDVs funcione mesmo na indisponibilidade do *Processador de Vendas*, de forma transparente, garantindo que as informações enviadas cheguem ao destino com alta confiabilidade. O JMS Provider garante essa assincronia entre as aplicações. Outro aspecto

relevante que o diagrama especifica é que o JMS Provider está executando em nó isolado (*JBoss Application Server*). Ou seja, produtores e consumidores não compratilham CPU, disco e rede com o *provider*.

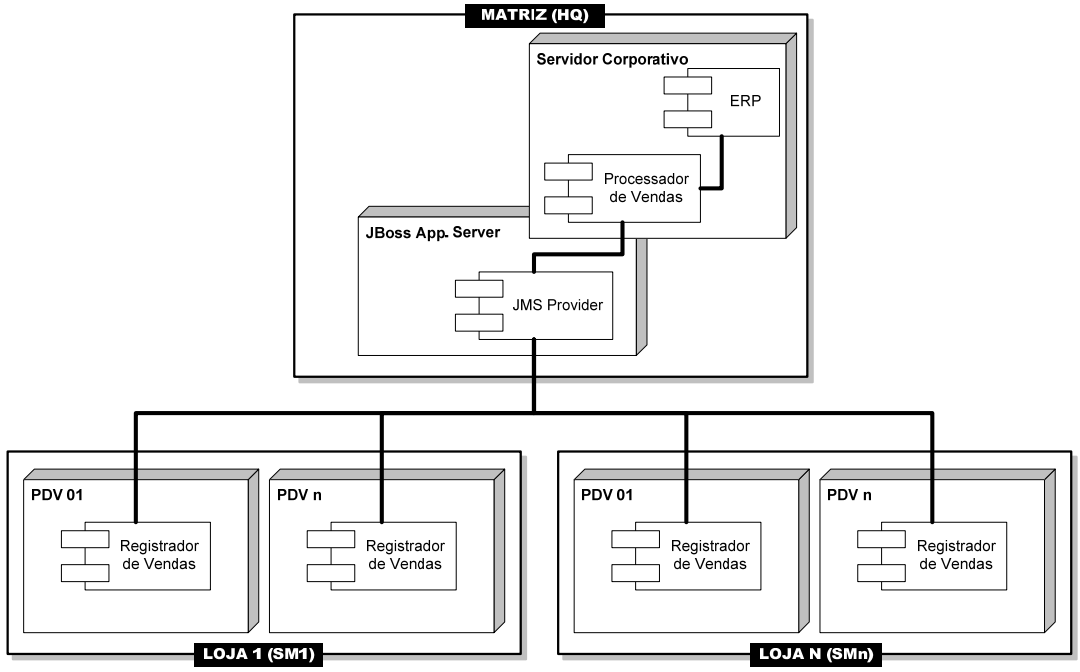


Figura 5-4. Interação *Registro de Venda*: diagrama de implantação.

A Figura 5-5 apresenta o diagrama de seqüência desenvolvido para representar todas as trocas de mensagens entre as aplicações nessa primeira interação. O JMS Provider é representado pelos canais de comunicação envolvidos no exemplo *C_HQ_SALEINF* (o prefixo C indica o canal). A aplicação *Registrador de Vendas*, executando no PDV01 da loja um (SM1), é representada pela entidade *SM1_PDV01*. O *Processador de Vendas*, em execução na matriz (HQ), é representado pela entidade *HQ_SALEMAN*.

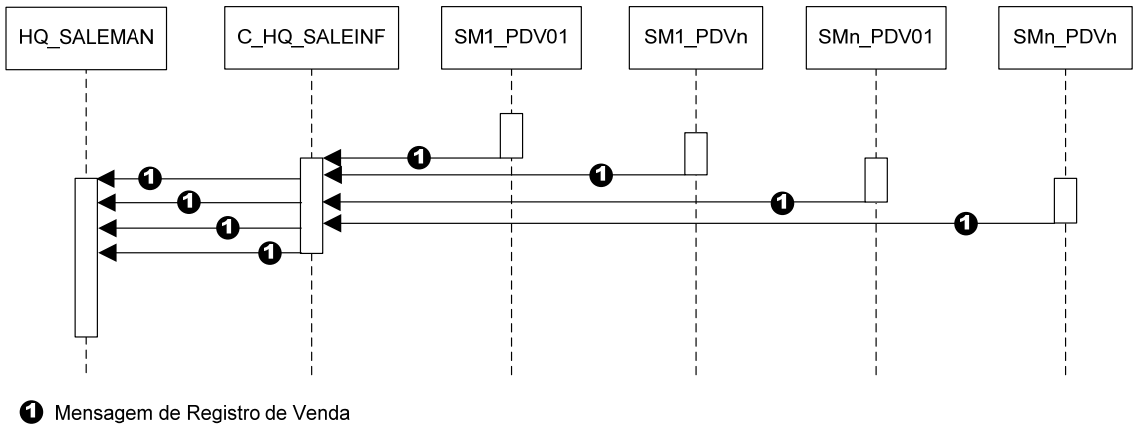


Figura 5-5. Interação *Registro de Venda*: diagrama de seqüência.

O diagrama especifica que todos os PDVs enviam uma *Mensagem de Registro de Venda* para o canal *C_HQ_SALEINF*, que as encaminha para a entidade *HQ_SALEMAN*. Para complementar o diagrama, precisa ser definido o evento gerador, responsável pelo início da interação, no caso do envio da mensagem. Esse evento é o registro de um novo produto no caixa, durante a apuração de uma venda.

Na segunda interação, a *Gestão de Vendas* comunica aos PDVs as atualizações de preços. Essa comunicação é contínua, pois a todo o momento a aplicação pode estar processando regras que levam em consideração o estoque atual, os preços da concorrência, o volume de vendas por produto, entre outros fatores. A Figura 5-6 apresenta uma visão geral dessa interação.

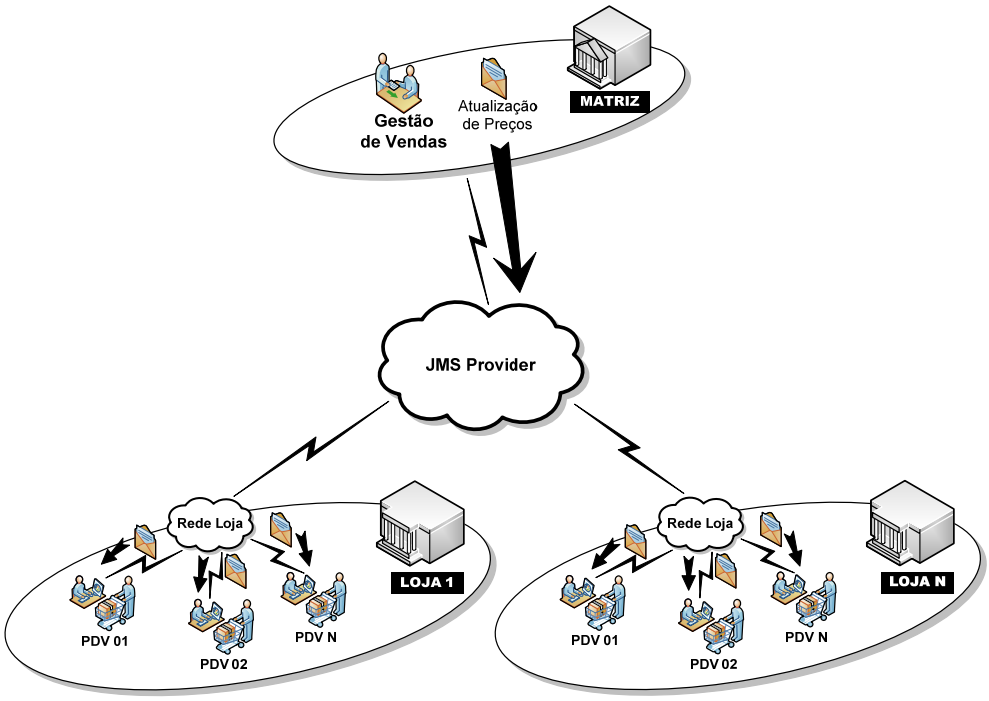


Figura 5-6. Interação *Atualização de Preços*: visão geral.

A Figura 5-7 apresenta o diagrama de implantação desta interação. Por simplificação do cenário avaliado neste capítulo, a aplicação *Gestor de Preços* (executando no servidor corporativo) envia as mensagens de atualização de preços para todos os PDVs, deixando a cargo da aplicação *Atualizador de Preço* (executando no PDV) determinar se deve ou não aplicar o reajuste naquela loja.

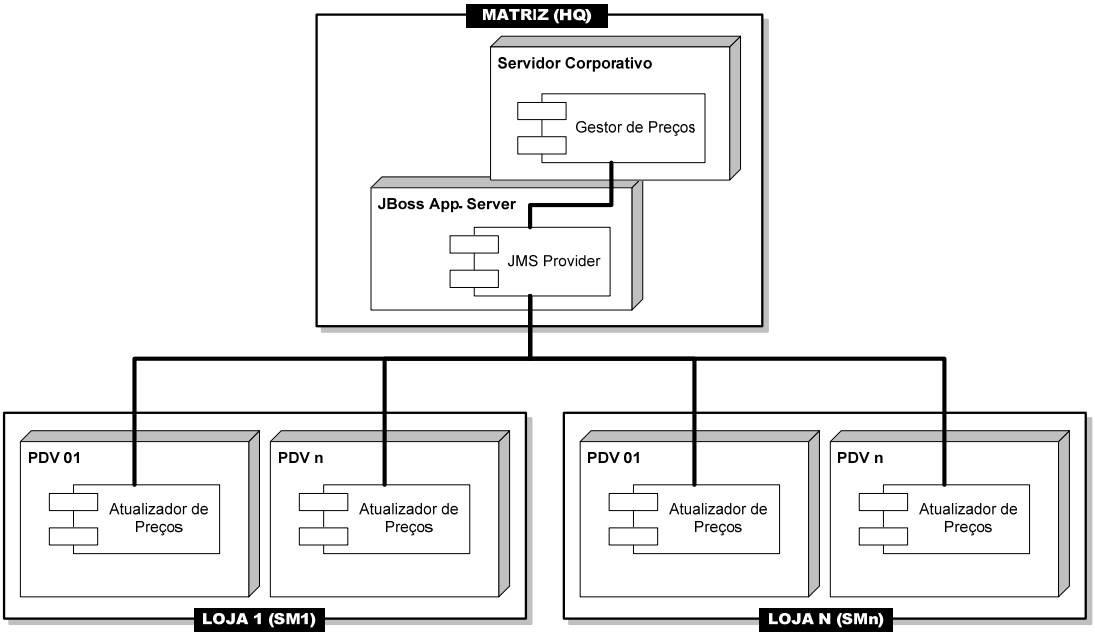


Figura 5-7. Interação *Atualização de Preços*: diagrama de implantação.

A troca de mensagens nesta interação envolve dois canais de comunicação, sendo um deles *Pub/Sub*. A Figura 5-8 apresenta o diagrama de seqüência desta interação. O evento gerador que dispara a execução dessa interação é a atualização de algum preço. Quando isso acontece, a aplicação *Gestor de Preços*, representada no diagrama pela entidade *HQ_PRICEMAN* envia a *Mensagem de Atualização de Preço* para o canal de comunicação *C_SM_PRICEUPD*. Este, por sua vez, encaminha uma cópia dessa mensagem para todos os PDVs em todas as lojas.

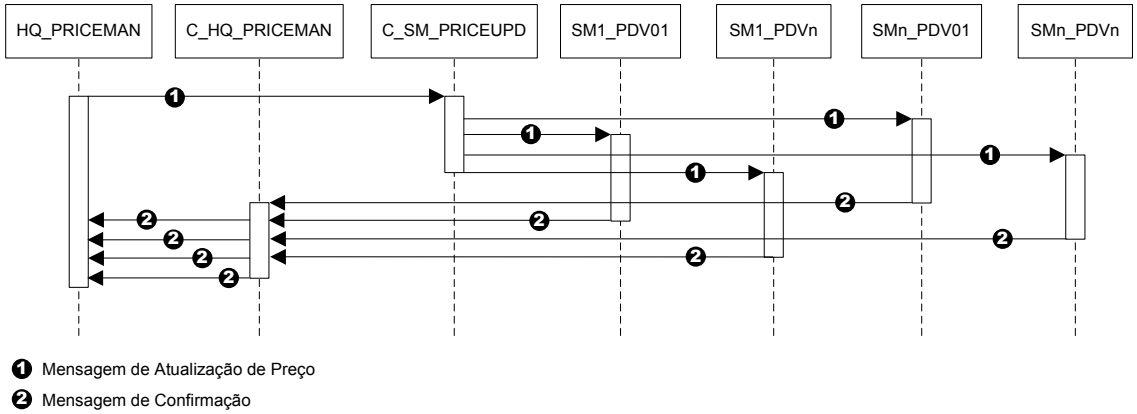


Figura 5-8. Interação *Atualização de Preços*: diagrama de seqüência.

Todos os PDVs, ao receberem e processarem a mensagem de atualização de preços, informam ao *Gestor de Preços* se ele efetuou a alteração solicitada. Essa comunicação é realizada através do canal de comunicação ponto-a-ponto (PTP) denominado *C_HQ_PRICEMAN*.

5.2.2 Service Level Objectives (SLO)

A interação *Registro de Vendas* necessita que a entrega das mensagens seja realizada com latência máxima de cinco segundos, e exige um QoS (confiabilidade e integridade transacional, ver Seção 3.2.2) que garanta a entrega das mensagens mesmo em caso de falha. Cada informação (registro de venda) é independente, logo, não necessita nem a garantia de entrega na mesma ordem de envio, nem o agrupamento de mensagens (entrega transacional). Por tudo isso, o nível de QoS do canal *C_HQ_SALEINF* é PNT (persistente e sem transação).

A interação *Atualização de Preços* possui duas linhas de comunicação, sendo cada uma delas através de um canal de comunicação diferente. Na comunicação de informação da atualização de preços, as mensagens devem ter latência máxima de um segundo, e precisam ter garantia de entrega e ordenamento. Logo, o canal *C_SM_PRICEUPD* precisa ter nível de QoS PNT.

Na segunda linha de comunicação da interação *Atualização de Preços*, as mensagens de confirmação de atualização devem ter latência máxima de cinco segundos, e não precisam ter garantia de entrega, pois essa informação é gerencial, e se perdida não compromete a gestão de preços. Desta forma, o canal *C_HQ_PRICEMAN* precisa ter nível de QoS NPNT (sem persistência e sem transação).

5.3 Entendimento do Ambiente

Nessa atividade o *Gerente de Capacidade* produz o artefato *Arquitetura do Serviço de Integração*. Esse artefato detalha os aspectos sobre a integração que são necessários para as demais atividades.

5.3.1 Arquitetura do Serviço de Integração

A arquitetura do serviço de integração do cenário analisado possui os seguintes recursos computacionais mapeados por camada:

- Plataforma Operacional: CPU, disco e rede;
- Middleware: *Destination.*

Para monitorar esses recursos as métricas escolhidas são *CPU Utilization* e *Destination Size*. Quanto ao disco e a rede não são monitoradas, pois a suposição é que o recurso de CPU é mais escasso que os demais. Assim, ele deve ser comprometido antes. Além disso, quando o sistema estiver saturado por qualquer dos recursos da plataforma, o tamanho do *destination* tende a um crescimento exponencial, justificando a sua monitoração.

Outro aspecto que precisa fazer parte desse artefato é o mapeamento dos parâmetros que afetam o desempenho do serviço, por camada. A Seção 3.3.1 apresenta um conjunto pré-definido pelo JMSCapacity que atende a essa necessidade de detalhamento.

Além de listar os parâmetros, o *Gerente de Capacidade* precisa definir um conjunto de hipóteses sobre o impacto desses parâmetros no desempenho do serviço de integração. Mais uma vez o processo já traz o detalhamento necessário para auxiliar esse passo da atividade (ver Seção 3.3.3).

Por fim, a arquitetura do serviço de integração precisa definir possíveis modelos de cenários, indicando quais dos parâmetros definidos passam a ser definidos como fatores para a avaliação.

As duas interações definidas no *Plano de Integração* indicam que os modelos de cenários a serem estudados nesse planejamento incluem: um PTP de muitos para poucos; e outro *Pub/Sub* de um para muitos. Duas comunicações utilizam o primeiro modelo de cenário, enquanto a outra utiliza o segundo modelo.

Para o modelo de cenário PTP, os parâmetros *número de produtores*, *taxa de produção* (geração de mensagens), *número de consumidores* e *método de recepção* passam a compor a lista de fatores. A justificativa dessa escolha é pelo fato que os produtores em questão representam os PDVs, que para um planejamento de capacidade do serviço de integração precisa ser variado, com objetivo de se definir qual o número máximo de PDVs suportado pelo serviço, por exemplo. A taxa de geração representa o número de itens registrados no caixa por unidade de tempo, ou o número de atualizações de preço realizadas por unidade de tempo pelo PDV. Novas tecnologias, como RFID, podem influenciar a variação desse parâmetro, levando-o a compor cenários futuros a serem avaliados durante o planejamento. O número de consumidores representa a quantidade de *Processadores de Venda* ou de *Gestores de Preços*. A variação desse parâmetro pode ser necessária pra avaliar a quantidade de recurso de servidores necessária para o recebimento e processamento das mensagens. Por

fim, o método de recepção precisa ser variado para avaliar a quantidade de consumidores que pode ser reduzida em função desse parâmetro.

No modelo de cenário *Pub/Sub*, os parâmetros *taxa de produção*, *tamanho da mensagem* e *número de consumidores* formam a lista de fatores. A taxa de produção representa a quantidade de atualizações de preços enviadas por unidade de tempo, enquanto que o tamanho da mensagem representa o número de artigos/produtos reajustados por mensagem de atualização enviada. A variação desses dois parâmetros pode avaliar o impacto da gestão sobre a capacidade do serviço de integração. A quantidade de consumidores nesse modelo de cenário representa o número de PDVs, que como já dito precisa ser variado para determinar o impacto dele na capacidade do serviço.

5.4 Caracterização da Carga de Trabalho

Esta atividade desempenhada pelo *Avaliador* tem como objetivo a construção do artefato *Modelo de Carga de Trabalho*. Esse modelo é formado por um diagrama, utilizando a notação básica apresentada na biblioteca de componentes (ver Seção 4.1), e um conjunto de informações adicionais sobre os eventos e as mensagens.

Para a construção do modelo de carga, o processo define inicialmente que se precisam identificar os componentes básicos de carga. No cenário sob estudo, os componentes básicos de carga são:

- Mensagem PTP recebida pelo MOM originada dos PDVs;
- Mensagem PTP encaminhada pelo MOM para a matriz;
- Mensagem *Pub/Sub* recebida pelo MOM originada da matriz;
- Mensagem *Pub/Sub* encaminhada pelo MOM para os PDVs.

Em seguida, precisam ser definidos os parâmetros para cada tipo de componente básico identificado. A Tabela 3-1 apresentada durante o detalhamento do processo define os parâmetros necessários.

Em um estudo real, o *Avaliador* precisaria monitorar o sistema real medindo as demandas de serviço para cada recurso (CPU, disco e rede) de cada componente básico, com objetivo de particionar a carga em classes homogêneas. Ou seja, a monitoração agruparia as mensagens em classes de carga em função da semelhança dos valores dos seus parâmetros, mas principalmente em função do consumo dos recursos CPU, disco e rede.

Porém, no cenário escolhido, é considerado que todas as mensagens de um tipo (ex: registro de vendas, atualização de preços) sejam homogêneas quanto aos valores de seus parâmetros. Essa simplificação forma duas classes de carga para cada tipo de mensagem. A primeira, relacionada com a recepção da mensagem pelo MOM (produção), e a segunda, relativa ao envio da mensagem pelo MOM para o consumidor (consumo).

O *Plano de Integração* define três tipos de mensagens: registro de vendas, atualização de preços e confirmação de atualização de preços, que passam a ser denominadas respectivamente de *saleInfo, priceUpdate, priceUpdateAck*. Sendo assim, conclui-se que o cenário sob estudo possui seis classes de carga de trabalho. A Tabela 5-1 apresenta os canais de comunicação utilizados para cada tipo de mensagem, conforme especificado no *Plano de Integração*.

Tabela 5-1. Canal de comunicação utilizado para cada tipo de mensagem.

Interação	Tipo de Mensagem	Canal de Comunicação
1	saleInfo	Queue (C_HQ_SALEINF)
2	priceUpdate	Topic (C_SM_PRICEUPD)
2	priceUpdateAck	Queue (C_HQ_PRICEMAN)

Para cada classe de carga identificada, precisam-se obter ou definir os valores para seus parâmetros. A Tabela 5-2 apresenta os valores para os parâmetros que dependem somente do tipo da mensagem, ou seja, eles se repetem nas duas classes de carga associadas (de produção e de consumo).

Tabela 5-2. Valores dos parâmetros por tipo de mensagem.

Valores para Parâmetros por Tipo de Mensagem	saleInfo	priceUpdate	priceUpdateAck
Fomato da mensagem	BytesMsg	BytesMsg	BytesMsg
Conteúdo da mensagem em *bytes*	1024	10240	1024
Cabeçalho da mensagem em *bytes*	203	203	203
Estilo das mensagens	PTP	*Pub/Sub*	PTP
Confiabilidade do canal	PNT	PNT	NPNT

O tamanho da mensagem é informado em dois campos, um referente ao tamanho do conteúdo da mensagem, e outro referente ao tamanho do cabeçalho JMS. O tamanho do conteúdo da mensagem foi arbitrado e uniformizado por tipo de mensagem para garantir que todas as mensagens do mesmo tipo formem a mesma classe de carga. O tamanho do cabeçalho da mensagem foi obtido a partir da captura de pacotes na rede, e depende do formato escolhido para a mensagem. O estilo da mensagem e a confiabilidade do canal foram definidos no *Plano de Integração*.

A Tabela 5-3 apresenta os valores dos parâmetros para as classes de carga de produção para cada tipo de mensagem. A expressão {#SM} é interpretada como a quantidade de lojas. Como cada loja possui cerca de sessenta PDVs, {#SM*60} representa a quantidade total de PDVs. No caso da mensagem *priceUpdateAck* , foi arbitrado para o cenário que dois terços dos PDVs iriam atualizar sua lista de preços. Com isso, o número de produtores diminui para {#SM*40}.

Para a mensagem *priceUpdateAck* não se aplica o parâmetro taxa de geração de mensagens, pois apenas é produzida uma mensagem desse tipo quando da chegada da mensagem *priceUpdate*. Para as outras duas mensagens foi arbitrada uma taxa de geração utilizando uma distribuição de *Poisson*.

Tabela 5-3. Valores dos parâmetros para as classes de carga de produção.

Valores para Parâmetros por Classe de Carga de Produção	saleInfo	priceUpdate	priceUpdateAck
Quantidade de produtores	#SM*60	1	#SM*40
Taxa geração de mensagens em *mps*	Poisson (1.0)	Poisson (0.1)	N/A

A Tabela 5-4 apresenta os valores dos parâmetros para as classes de carga de consumo para cada tipo de mensagem. A taxa de recuperação de mensagens representa a freqüência com que o consumidor síncrono faz chamada ao método *receive* da API JMS (*receives* por segundo, ou *rps*). Logo, essa taxa não se aplica a consumidores assíncronos. O modelo de assinatura somente se aplica a comunicações *Pub/Sub*.

O tempo de processamento das mensagens PTP foi arbitrado como nulo. No caso da mensagem *Pub/Sub* esse tempo foi definido como sendo de dois segundos (ou 2000ms). Isso significa que as mensagens de atualização de preço levam dois segundos antes de disparar a mensagem de confirmação.

Os parâmetros que estão faltando serem definidos são os relativos ao consumo de recursos de CPU, disco e rede para todas as seis classes de carga. Esses parâmetros são calculados e obtidos na atividade de *Parametrização, Validação e Calibragem do Modelo*.

Tabela 5-4. Valores dos parâmetros para as classes de carga de consumo.

Valores para Parâmetros por Classe de Carga de Consumo	saleInfo	priceUpdate	priceUpdateAck
Quantidade de consumidores	8	#SM*60	2
Método de recepção	síncrono	assíncrono	assíncrono
Taxa de recuperação de mensagens em *rps*	Poisson (30.0)	N/A	N/A
Modelo de assinatura	N/A	durable	N/A
Tempo de processamento de mensagens em *ms*	Nulo	2000	Nulo

5.4.1 Modelo de Carga de Trabalho

O modelo de carga de trabalho para o cenário sob estudo é composto por três linhas de comunicação, sendo uma na primeira interação e duas na segunda. A Figura 5-9 apresenta o modelo de carga para a primeira interação.

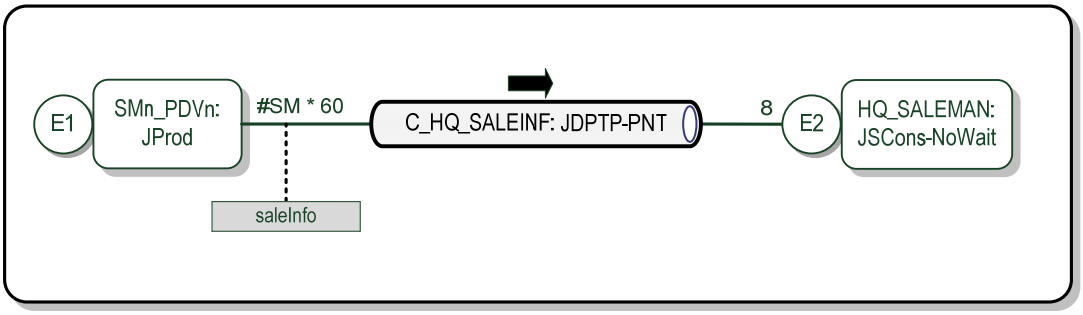


Figura 5-9. Interação *Registro de Vendas*: modelo de carga.

A construção desse modelo utilizou os parâmetros definidos, assim como a biblioteca de componentes de carga do JMSCapacity, para mapear o diagrama de seqüência construído no *Plano de Integração* em um modelo de carga. O evento *E1* está especificado pela taxa de geração das mensagens *saleInfo*, conforme Tabela 5-3, no valor de uma distribuição de Poisson com média de uma mensagem por segundo. O evento *E2* está definido pela taxa de recuperação das mensagens *saleInfo*, conforme Tabela 5-4, no valor de uma *Poisson* com média de trinta *receives* por segundo.

Os demais parâmetros são definidos conforme Tabela 5-1 (tipo e nome do canal de comunicação), Tabela 5-2 (estilo das mensagens e nível de confiabilidade), Tabela 5-3 (número de instâncias da entidade produtora) e Tabela 5-4 (número de instância da entidade cosumidora).

A Figura 5-10 apresenta o modelo de carga para a segunda interação do cenário. Duas linhas de comunicação estão modeladas. A primeira linha trata a comunicação *Pub/Sub* de envio da mensagem *priceUpdate*, enquanto que a segunda representa a comunicação PTP com o envio da mensagem *priceUpdateAck*. A entidade *HQ_PRICEMAN* possui duas sessões ativas: a primeira como produtora *Pub/Sub* (*HQ_PRICEMAN_S1*) e a segunda como consumidora assíncrona (*HQ_PRICEMAN_S2*).

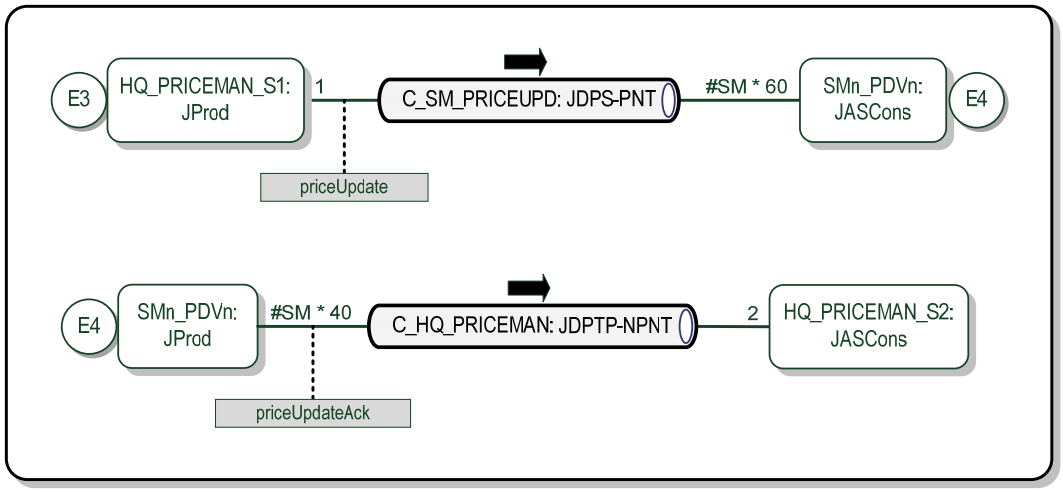


Figura 5-10. Interação *Atualização de Preços*: modelo de carga.

As duas linhas estão conectadas pelo evento *E4*. Esse evento determina que dois terços dos PDVs que receberem mensagens *priceUpdate* devem gerar uma mensagem *priceUpdateAck*, informando que atualizaram suas listas de preços. O evento *E3* está definido na Tabela 5-3 no valor de uma Poisson com média igual a uma mensagem a cada dez segundos (taxa de 0.1 mps).

Os demais parâmetros são definidos conforme Tabela 5-1 (tipo e nome dos canais de comunicação), Tabela 5-2 (estilo das mensagens e nível de confiabilidade de cada canal), Tabela 5-3 (número de instâncias da entidade produtora da mensagem *priceUpdate*) e Tabela 5-4 (número de instância das entidades consumidoras).

5.5 Desenvolvimento do Modelo de Desempenho

O *Avaliador* nessa atividade produz o *Modelo de Referência do Sistema*, através de um mapeamento do *Modelo de Carga de Trabalho*. Duas estratégias podem ser utilizadas para construção desse modelo: um único modelo para todas as interações ou um modelo para cada interação.

Na primeira opção o modelo resultante pode ser complexo demais para ser avaliado posteriormente, logo, somente é válido em situações com poucos canais de comunicação, mesmo que haja múltiplas linhas de comunicação para o mesmo canal. Na segunda estratégia, o *Avaliador* precisa compor o resultado das múltiplas interações para determinar o comprometimento total de recursos do JMS Provider.

A estratégia utilizada neste guia passo-a-passo é a construção de modelos para cada interação do modelo de carga. A Figura 5-11 apresenta o modelo de referência para a interação *Registro de Vendas*. Esse modelo é a simples composição dos componentes de desempenho da biblioteca: *JProdGroup-Poisson-Sync*, *JDPS-PNT* e *JSCons-NoWait-Poisson*.

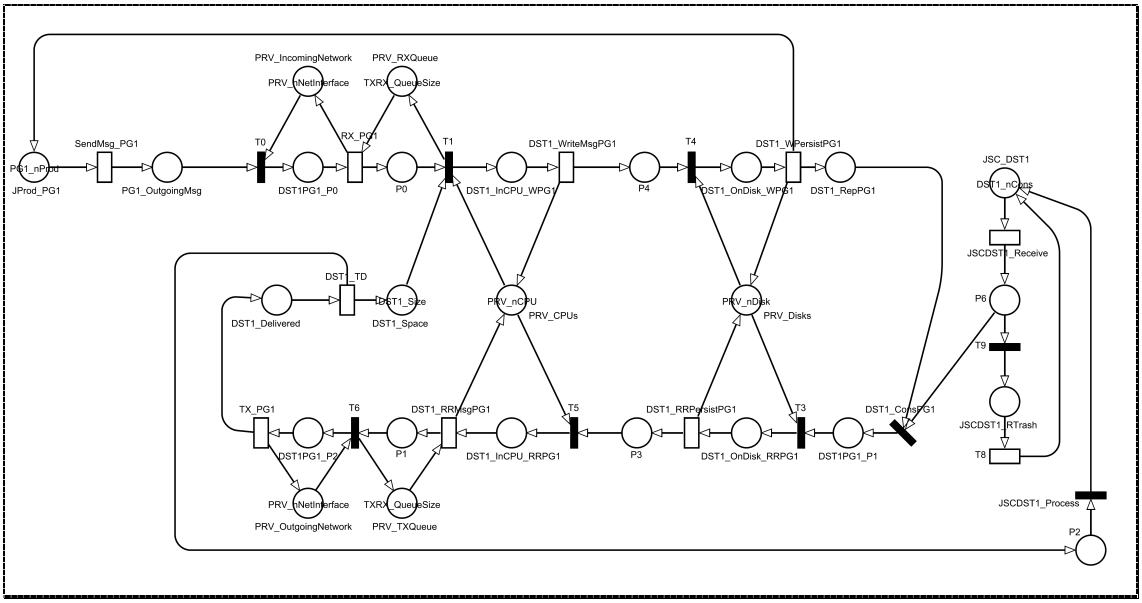


Figura 5-11. Interação *Registro de Vendas*: *Modelo de Referência do Sistema*.

O *JProdGroup-Poisson-Sync* é escolhido para mapear a entidade produtora do tipo *JProd* (do modelo de carga) que está associada a um evento (*E1*) definido por uma distribiuição de *Poisson*. A escolha pela opção síncrona (*Sync*) é pelo

fato dela representar com mais fidelidade o comportamento de produção nesse cenário, pois todos os PDVs somente enviam uma segunda mensagem *saleInfo* após a conclusão da primeira.

O *JSCons-NoWait-Poisson* é utilizado para representar o consumidor síncrono do tipo *JSCons-NoWait* (do modelo de carga) que está associado a um evento (*E2*), definido também por uma distribuição de *Poisson*. O canal de comunicação é um mapeamento direto do tipo definido no modelo de carga.

A interação *Atualização de Preços* é mapeada em um modelo de referência conforme apresentado na Figura 5-12. As duas linhas de comunicação são mapeadas em conjunto, uma vez que uma está ligada a outra pelo evento (*E4*). O modelo apresenta dois canais, sendo o primeiro *JDPS-PNT* (responsável pelo encaminhamento das mensagens *priceUpdate*) e o segundo *JDPTP-NPNT* (responsável pelas mensagens *priceUpdateAck*).

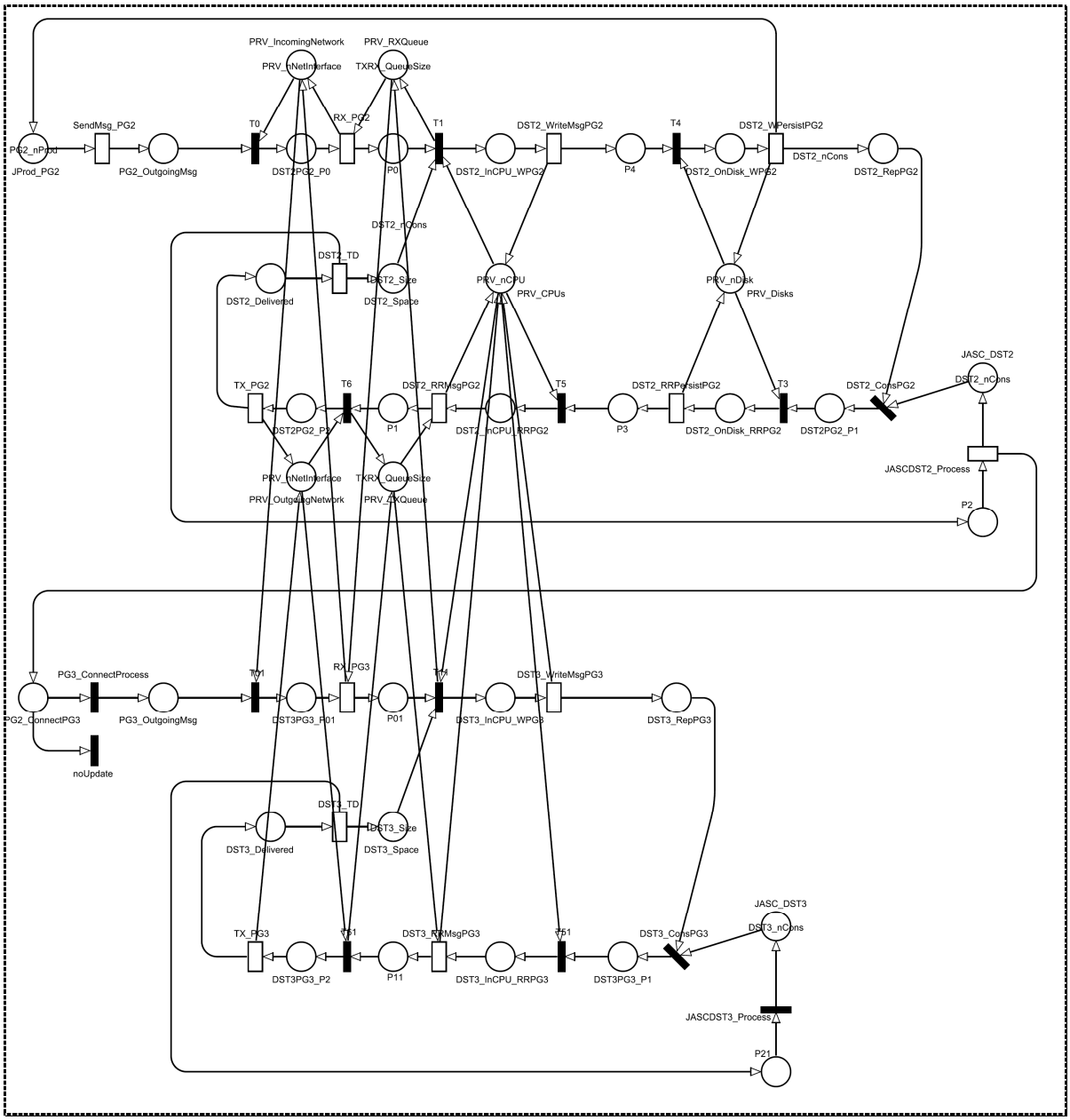


Figura 5-12. Interação *Atualização de Preços*: Modelo de Referência do Sistema.

Conectado ao primero canal é modelado um grupo de produção *JProdGroup-Poisson-Sync* por motivos semelhantes aos da primeira interação, e uma entidade consumidora *JASCons*, uma vez que o estilo *Pub/Sub* exige um consumidor assíncrono.

O segundo canal possui um grupo de produção complexo *JProdGroup-Connect* (versão 2), com disparo após o processamento da mensagem. Esse comportamento está especificado no modelo de carga, através do evento *E4* que conecta as duas linhas de comunicação.

Nesse caso, a especificação dos parâmetros de carga, determinam que apenas dois terços dos PDVs que recebem a mensagem *priceUpdate* efetuam a mudança das suas listas de preços e por conseguinte geram uma mensagem de confirmação *priceUpdateAck*. Essa restrição é implementada no modelo através da transição imediata noUpdate, implementada em concorrência com a transição *PG3_ConnectProcess* inerente ao componente *JProdGroup-Connect*. Essa transição adicional possui peso de um terço, enquanto que a transição padrão possui peso de dois terços, garantindo que apenas a quantidade especificada de *tokens* (mensagens) seja gerada para o novo canal.

Além do grupo de produção, o segundo canal possui uma entidade consumidora JASCons conectada a ele. O canal foi mapeado para o componente equivalente ao do modelo de carga *JDPTP-NPNT*, sendo a única diferença que os recursos que representam o *JMS Provider* não estão repetidos no segundo canal, pelo contrário, as transições que alocam e liberam recurso estão agora conectadas aos mesmos lugares que o primeiro canal, representando que ambos estão executando no mesmo *provider*.

5.6 Parametrização, Validação e Calibragem do Modelo

A parametrização do modelo de desempenho referente ao cenário em estudo é realizada em detalhes no Apêndice C, durante a validação da biblioteca de modelos. Isso é possível, pois os experimentos escolhidos para validação são baseados nas linhas de comunicação do modelo de carga do cenário.

O experimento *Exp_3* representa a interação *Registro de Vendas*, enquanto que os *Exp_4* e *Exp_5* representam as duas linhas de comunicação da interação *Atualização de Preços*.

A Tabela 5-5 apresenta um resumo dos parâmetros de entrada para o modelo de desempenho. O tempo de disco somente está associado a mensagens persistentes. Logo, não se aplica a mensagem *priceUpdateAck*, visto que a confiabilidade do canal utilizado é NPNT. Os demais tempos associados ao consumo de recursos são obtidos através de medições do sistema real, simulando uma carga especialmente preparada para isolar o consumo desses recursos por mensagem.

Além desses parâmetros obtidos via medição do sistema real, todos os demais estabelecidos na construção do cenário são utilizados para transformar o modelo de referência no *Modelo Refinado do Sistema*. Contudo, este modelo refinado somente é obtido após a sua devida validação e calibragem.

Sendo assim, o modelo precisa ser validado. Mas uma vez este guia utiliza o Apêndice C, que utiliza linhas de comunicação semelhantes a essas para validar a biblioteca, considerando-se, assim, o modelo validado por projeção.

Tabela 5-5. Parâmetros de entrada para modelo de desempenho.

Parâmetros de Entrada	saleInfo	priceUpdate	priceUpdateAck
Tempo de CPU para receber mensagem (write) em *ms*	1,214181	1,679877	0,320776
Tempo de disco para receber mensagem (write) em *ms*	0,029077	0,245193	N/A
Tempo de CPU para encaminhar mensagem (read/remove) em *ms*	0,963893	0,992983	0,593690
Tempo de disco para encaminhar mensagem (read/remove) em *ms*	0,018118	3,194549	N/A
Tempo de rede para receber ou transmitir mensagem (rx/tx) em *ms*	0,097733	0,800858	0,097733

Por último, precisa-se efetuar a calibragem do modelo. Neste cenário, é considerado que todos esses valores associados a consumo de tempo nos recursos são exponencialmente distribuídos (representação por uma transição exponencial no modelo GSPN). Porém, podem ser utilizadas técnicas de equalização de momentos para melhorar a aproximação dessas distribuições empíricas utilizando uma composição especial de exponenciais.

Para realizar essa equalização, utiliza-se o coeficiente de variação para se determinar o tipo de substituição a ser realizada (por uma *Erlang* ou uma *Hiperexponencial*). Esse algoritmo denominado *moment matching* não é utilizado neste capítulo por uma questão de simplificação, mas é apresentado em detalhes no Apêndice B.

Outra ação de calibragem é o tratamento de *outlier*, que é realizado nos dados obtidos das medições. No Apêndice C esse tratamento é realizado em todas as amostras na busca de um refinamento estatístico.

5.7 Previsão da Evolução da Carga de Trabalho

No cenário estudado algumas métricas de negócio estão relacionadas com a geração de demanda de serviço para a integração. A primeira é o volume de vendas, que afeta diretamente a quantidade de mensagens *saleInfo* dos PDVs para o ERP. Mudanças tecnológicas podem afetar bastante essa métrica: como exemplo a adoção de leitores RFID (Identificação por Rádio Freqüência), que podem ler todo um conjunto de itens em miléssimos de segundo, alterando a ordem de grandeza da taxa de geração de mensagens.

Outra métrica de negócio a ser considerada é a expansão no número de lojas, que impacta no número de PDVs. A concorrência de outros estabelecimentos, ou ainda, a volta de um ambiente macro-econômico inflacionário levaria a uma explosão no número de mensagens de atualização de preços (*priceUpdate* e *priceUpdateAck*).

5.7.1 Cenários Futuros de Carga

Diante da análise das métricas de negócio e sua relação com as demandas de serviço geradas, o *Gerente de Capacidade* determina possíveis cenários futuros de carga a serem considerados no planejamento de capacidade.

Cenário de Carga: Aumento no Número de Lojas

Novas lojas estão sendo abertas, possuindo cada uma sessenta PDVs. Logo, precisa-se determinar qual o número máximo de lojas suportadas pelo *JMS Provider* nas duas interações: *Registro de Vendas* e *Atualização de Preços*.

Cenário de Carga: Aumento do Volume de Vendas

Todos os PDVs atualmente registram em média uma informação de venda por segundo. Considerando a expansão de mercado e a utilização de novas tecnologias (ex. RFID) que podem ser associadas aos caixas para acelerar o processo de registro de venda, precisa-se estabelecer a quantidade de lojas suportadas pelo *JMS Provider* se o volume de informações de venda multiplicar por dez, levando a taxa de geração de mensagens *saleInfo* para uma Poisson com média de 10mps. Outra alternativa seria alterar o software de registro para agrupar várias mensagens em uma única, deixando a taxa no valor original (1mps), mas aumentando o tamanho da mensagem para 10KB em vez de 1KB.

Cenário de Carga: Retorno de Ambiente Inflacionário

Esse cenário se materializando pode elevar significativamente a necessidade de atualização de preços. Assim, precisa-se estabelecer a capacidade do *JMS Provider* em suportar um aumento da taxa de geração de mensagens *priceUpdate*, saindo dos atuais 0.1mps para valores próximo a 1.0mps.

5.8 Avaliação dos Cenários

O Avaliador de posse dos *Cenários de Carga* previstos e utilizando o *Modelo Refinado do Sistema* realiza experimentos de avaliação com o objetivo de obter valores para as métricas do MOM. Esses valores compõem o artefato *Relatório de Avaliação*.

Neste guia, a avaliação de desempenho é realizada através da ferramenta TimeNET, utilizando simulações estacionárias com nível de confiança de 99% e erro máximo relativo de 1%.

Para cada um dos modelos disponibilizados precisam ser estabelecidas medidas que possibilitem o cálculo das métricas desejadas. Como exemplo, para efetuar o cálculo da métrica *Message Delivery Rate* no modelo da Figura 5-11 precisa-se determinar o valor esperado de *tokens* no lugar *DST1_Delivered* e aplicar a lei de *Little*. Mais detalhes sobre as leis operacionais no Apêndice A.

A latência do canal possui um erro estatístico um pouco maior, pois ele está baseado em uma medida de pouca precisão (quantidade de mensagens dentro do MOM em atendimento, ou *Destination Size*). No Apêndice C esse fato está mais detalhado.

Um ponto a ressaltar é que, nas planilhas de resultados apresentadas ao longo dessa seção, cada linha é resultado de um experimento de simulação. Na interação *Registro de Vendas*, esses experimentos consumiram poucos minutos cada um deles. Na interação *Atualização de Preços*, porém, o tempo de simulação atingiu dezenas de minutos, ou em alguns casos algumas horas. A justificativa desse tempo excessivo é a maior complexidade do modelo.

O restante dessa seção apresenta os resultados das simulações para cada um dos cenários de carga. Esses resultados compõem o *Relatório de Avaliação*.

5.8.1 Avaliação de Cenário de Carga: Aumento no Número de Lojas

O impacto do aumento do número de lojas é avaliado em cada uma das interações isoladamente, cabendo ao *Gerente de Capacidade* consolidar os resultados, realizando as considerações que lhe convier.

Interação Registro de Vendas

Inicialmente, o fator *número de produtores* foi variado no intuito de determinar a quantidade máxima suportada pelo MOM. Como o cenário prevê o aumento no número de lojas e cada uma possui 60 PDVs, as simulações são realizadas com valores 60, 120, 180 e 240. Os resultados são mostrados na Tabela 5-6. Nela pode-se observar que o tamanho da fila cresceu exponencialmente, chegando a mais de 9mil mensagens para 240 PDVs. O resultado disso é uma latência superior a 38s, o que está em desacordo com o SLO definido, que permite latências máximas de 5s.

Percebe-se, porém, que o gargalo não é a CPU, pois a utilização desse recurso ficou próxima a 25%. O limitador, nesse caso, é a relação entre os parâmetros *quantidade de consumidores* e *taxa de recuperação de mensagens* especificados na Tabela 5-4. O número de consumidores é igual a oito, enquanto a freqüência com que cada um procura o canal pra recuperar uma mensagem é determinada por uma *Poisson* com média igual a trinta. O resultado da combinação desses dois parâmetros é uma produtividade nominal máxima de 240mps. A tabela com os resultados apresenta ainda números próximos a 240 PDVs para demonstrar a degradação exponencial do MOM.

Tabela 5-6. Aumento do Número de Lojas: avaliação da interação *Registro de Vendas* com parâmetros padrão.

PDVs	Message Delivery Rate (mps)	System Throughput (KBps)	CPU Utilization (%)	Destination Size (msg)	Latency (ms)
60	59,84	60	6,53%	0,39	6,58
120	119,53	120	13,06%	1,29	10,83
180	180,46	180	19,60%	3,70	20,52
230	228,67	229	25,00%	52,30	228,71
235	232,25	232	25,37%	64,83	279,14
240	232,44	232	25,71%	9.018,65	38.799,31

Para contornar essa dificuldade, uma alternativa é aumentar a taxa com que cada consumidor envia um comando *receive* para o MOM. A Tabela 5-7 apresenta os resultados da simulação para uma taxa de 150rps (*receives por segundo)*, em substituição ao valor de 30rps inicialmente utilizado. Os resultados mostram que o número máximo de lojas suportadas pelo MOM para essa interação aumentou de três para quatorze, representando o atendimento a 840 PDVs.

Tabela 5-7. Aumento do Número de Lojas: avaliação da interação *Registro de Vendas* com parâmetro *taxa de recuperação de mensagens* em 150rps.

PDVs	Message Delivery Rate (mps)	System Throughput (KBps)	CPU Utilization (%)	Destination Size (msg)	Latency (ms)
300	298,82	299	32,52%	1,06	3,55
360	358,96	359	39,05%	1,19	3,30
420	421,39	421	45,60%	1,64	3,90
480	479,84	480	51,99%	1,91	3,98
540	536,60	537	58,94%	2,44	4,55
600	598,61	599	65,29%	3,08	5,15
660	656,66	657	71,90%	4,13	6,29
720	713,02	713	78,13%	6,07	8,52
780	773,73	774	84,49%	11,55	14,93
840	834,59	835	90,82%	86,05	103,10
850	834,07	834	91,51%	169,78	203,56
860	833,45	833	91,93%	8.593,70	10.311,03

As boas práticas de planejamento de capacidade recomendam que não se deve considerar para cálculo da capacidade adequada, consumos de recursos superiores a 70%. Desta forma, a avaliação desse cenário para esta interação indica que a capacidade adequada deste MOM é entre 10 e 11 lojas.

Interação Atualização de Preços

A Tabela 5-8 apresenta os resultados da avaliação do MOM perante esta interação, quanto às métricas de utilização e resposta. Pode-se perceber que o comprometimento de CPU é mínimo, demonstrando que o custo maior desse cenário é no acesso a disco, na rede e na espera dos consumidores assíncronos. O acesso a disco e a rede são identificados pelo próprio valor atribuído nos parâmetros de entrada desse modelo de desempenho (ver Tabela 5-5), que possui ordem de grandeza superior aos seus pares nas outras linhas de comunicação.

Tabela 5-8. Aumento do Número de Lojas: avaliação da interação *Atualização de Preços* quanto às métricas de utilização e resposta.

PDVs	CPU Utilization (%)	Destination2 Size (msg)	Destination3 Size (msg)	Latency DST2 (ms)	Latency DST3 (ms)
60	0,49%	1,72	0,15	284,48	38,35
120	0,97%	4,12	-0,02	341,18	-1,99
180	1,40%	7,62	0,39	430,29	33,20
240	1,93%	12,54	0,12	527,25	7,74
300	2,40%	18,50	-0,22	616,46	-11,17
540	4,27%	56,00	0,03	1.053,65	0,97
600	4,74%	73,39	0,03	1.271,79	0,72

A conclusão é que, até nove lojas (540 PDVs) podem ser atendidos com este serviço de integração nesta configuração, pois o SLO definido indica que a mensagem *priceUpdate* deve ser entregue em no máximo um segundo, valor aproximado do obtido com essa carga de trabalho. A Tabela 5-9 apresenta o restante das métricas obtidas nesta avaliação.

Tabela 5-9. Aumento do Número de Lojas: avaliação da interação *Atualização de Preços* quanto às métricas de produtividade.

PDVs	Message Delivery Rate DST2 (mps)	DST2 Throughput (KBps)	Message Delivery Rate (mps)	System Throughput (KBps)
60	6,04	60	4,04	40
120	12,08	121	8,03	80
180	17,72	177	11,66	117
240	23,78	238	15,69	157
300	30,00	300	19,97	200
540	53,15	532	35,27	353
600	57,70	577	39,14	391

A espera pela liberação dos consumidores assíncronos (em função do processamento de mensagens) afeta um pouco a latência da linha de comunicação *Pub/Sub*. Para isso, fixou-se o parâmetro do número de produtores (PDVs) em 300 e utilizou o parâmetro *tempo de processamento* como fator para a avaliação. Os resultados obtidos são mostrados na Tabela 5-10, com o fator variando dos atuais 2.000ms até atingir 100ms. Essa variação causou um impacto percentual na diminuição de aproximadamente 15% na latência do canal, o que comprova a influência desse parâmetro no desempenho da interação como um todo.

Tabela 5-10. Aumento do Número de Lojas: avaliação da interação *Atualização de Preços* quanto ao impacto do parâmetro *tempo de processamento*.

Process Time (ms)	CPU Utilization (%)	Destination2 Size (msg)	Destination3 Size (msg)	Latency DST2 (ms)	Latency DST3 (ms)
2.000	2,40%	18,50	-0,22	616,46	-11,17
1.000	2,42%	15,88	0,00	527,91	-0,10
100	2,38%	15,75	0,04	530,57	1,87

5.8.2 Avaliação de Cenário de Carga: Aumento do Volume de Vendas

Este cenário de carga estabelece a relação entre o volume de vendas e o aumento da demanda de serviço de integração. Duas possíveis conseqüências são avaliadas: o aumento da taxa de produção e o aumento do tamanho da mensagem.

Aumento da Taxa de Produção

Se o número de vendas cresce, o parâmetro *taxa de produção* se eleva. A Tabela 5-11 apresenta os resultados da variação deste parâmetro com valores de 1mps e 10mps. Para avaliar o impacto, fixou-se os demais parâmetros. O resultados mostra que o crescimento da taxa representou um aumento equivalente na utilização de CPU.

Tabela 5-11. Aumento do Volume de Vendas: avaliação do impacto do parâmetro *taxa de produção de mensagens*.

PDVs	Taxa de Produção (mps)	Message Delivery Rate (mps)	System Throughput (KBps)	CPU Utilization (%)
60	1	59,96	60	6,55%
60	10	585,56	586	64,08%

Aumento do Tamanho da Mensagem

Outra abordagem para contornar o crescimento exponencial da utilização de recursos é aglutinar em uma mensagem uma quantidade maior de informação a ser enviada. Com isso obtém-se a mesma vazão com um número menor de mensagens.

Para realizar essa modificação é necessário realizar nova parametrização do modelo, uma vez que o tamanho da mensagem afeta diretamente a demanda de serviço nos principais recursos do *provider*. Mais uma vez recorrendo ao Apêndice C, o *Experimento 2* de validação possui a situação desejada. A Tabela 5-12 apresenta os novos parâmetros de entrada considerando uma mensagem com tamanho de 10K.

Tabela 5-12. Parâmetros de entrada para modelo de desempenho: nova configuração para a interação Registro de Vendas (mensagem = 10K).

Parâmetros de Entrada do Modelo	saleInfo
Tempo de CPU p/ receber mensagem (write) em *ms*	1,763008
Tempo de disco p/ receber mensagem (write) em *ms*	0,073326
Tempo de CPU p/ encaminhar mensagem (read/remove) em *ms*	1,056905
Tempo de disco p/ encaminhar mensagem (read/remove) em *ms*	0,012624
Tempo de rede p/ receber/transmitir mensagem (rx/tx) em *ms*	0,800858

Após a nova parametrização do modelo e execução das simulações, a Tabela 5-13 apresenta os resultados obtidos. Pode-se notar que a vazão do sistema atinge valores até então não alcançados (superiores a 6MBps). Com isso se atende uma quantidade maior de lojas, além de conseguir trafegar uma quantidade maior de informação.

Tabela 5-13. Aumento do Volume de Vendas: avaliação do impacto do parâmetro *tamanho da mensagem*.

PDVs	Message Delivery Rate (mps)	System Throughput (KBps)	CPU Utilization (%)	Destination Size (msg)	Latency (ms)
60	59,61	596	8,41%	0,49	8,15
600	595,78	5.958	84,02%	6,27	10,52
660	658,39	6.584	92,85%	17,35	26,35

Os resultados obtidos na avaliação desse cenários de carga demonstra a utilização do JMSCapacity na tomada de decisões quanto à implementação de modificações na configuração do serviço.

5.8.3 Avaliação de Cenário de Carga: Retorno de Ambiente Inflacionário

Este cenário de carga procura estabelecer o impacto do aumento do número de mensagens enviadas para o MOM por unidade de tempo, considerando a interação *Atualização de Preços.*

Para realizar esta avaliação o valor do parâmetro *tempo de processamento* é fixado em 100ms, considerando com isso que o consumidor é mais ágil na transição entre a linha de comunicação *Pub/Sub* e a PTP. Outro parâmetro que foi fixado é o número de consumidores da mensagem *priceUpdate* (que representa o número de PDVs) para 300. O parâmetro utilizado como fator é a *taxa de produção* de mensagens *priceUpdate*, que inicialmente estava fixada através de uma Poisson com média de 0.1mps. Essa taxa é aumentada até atingir o valor de 1mps.

A Tabela 5-14 apresenta os resultados dos experimentos de simulação considerando a variação da *taxa de produção*. Nessa tabela a taxa está expressa em termos de tempo de espera para envio de uma nova mensagem, que equivale ao inverso da taxa. Percebe-se que o aumento da taxa para 1mps (*Send Time* igual a 1.000ms) faz a produção do MOM elevar-se consideravelmente.

Tabela 5-14. Retorno do Ambiente Inflacionário: avaliação do impacto do parâmetro *taxa de produção* nas métricas de produtividade.

Send Time (ms)	Message Delivery Rate DST2 (mps)	DST2 Throughput (KBps)	Message Delivery Rate (mps)	System Throughput (KBps)
10.000	29,69	297	19,91	199
5.000	59,34	593	41,09	411
2.000	166,91	1.669	111,46	1.115
1.000	265,21	2.652	179,25	1.792

A Tabela 5-15 mosta que esse aumento da produtividade é acompanhado pelo aumento da utilização de CPU, demonstrando que o modelo responde satisfatoriamente ao aumento da carga de trabalho. Mais uma vez fica claro que nesta interação a comunicação *Pub/Sub* determina uma latência elevada para a entrega de mensagens. Considerando que a latência total do sistema é a soma da latência nos dois canais mais o tempo de processamento no primeiro canal (fixada em 100ms), verifica-se que uma *taxa de produção* definida por uma *Poisson* com média de 0.5mps (*Send Time* de 2.000ms), consegue atender os 300 PDVs dentro do SLO estabelecido.

Uma conclusão dessa avaliação é que se precisa flexibilizar o SLO para a interação *Atualização de Preços*, devido à característica do estilo *Pub/Sub* em adicionar latência na comunicação.

Tabela 5-15. Retorno do Ambiente Inflacionário: avaliação do impacto do parâmetro *taxa de produção* nas métricas de utilização e resposta.

Send Time (ms)	CPU Utilization (%)	Destination2 Size (msg)	Destination3 Size (msg)	Latency DST2 (ms)	Latency DST3 (ms)
10.000	2,38%	15,75	0,04	530,57	1,87
5.000	4,83%	35,46	0,00	597,62	-0,03
2.000	13,24%	131,27	0,07	786,46	0,65
1.000	21,61%	457,47	0,14	1.724,90	0,79

5.9 Análise dos Resultados

Com o Relatório de Avaliação contendo as análises dos cenários de carga, o *Gerente de Capacidade* precisa analisar para cada alternativa apontada a relação custo-benefício para a construção do *Plano de Capacidade* para o serviço integração do cenário ilustrativo sob estudo. Este plano determina a maneira mais econômica de adiar ou evitar a saturação do serviço de integração, mantendo os níveis de serviço acordados.

Para isso, são utilizados os resultados que indicam o comprometimento dos recursos do MOM com cada uma das interações, as indicações de pontos de saturação, ou de não atendimento ao SLO, além do indicativo sobre o comportamento do serviço quando submetido a diversos tipos de carga de trabalho.

Outro recurso bastante utilizado nessa atividade é a produção de gráficos a partir dos resultados fornecidos pelo *Relatório de Avaliação*. A Figura 5-13 ilustra como os dados podem ser apresentados de forma a relacionar e cruzar as informações obtidas no relatório.

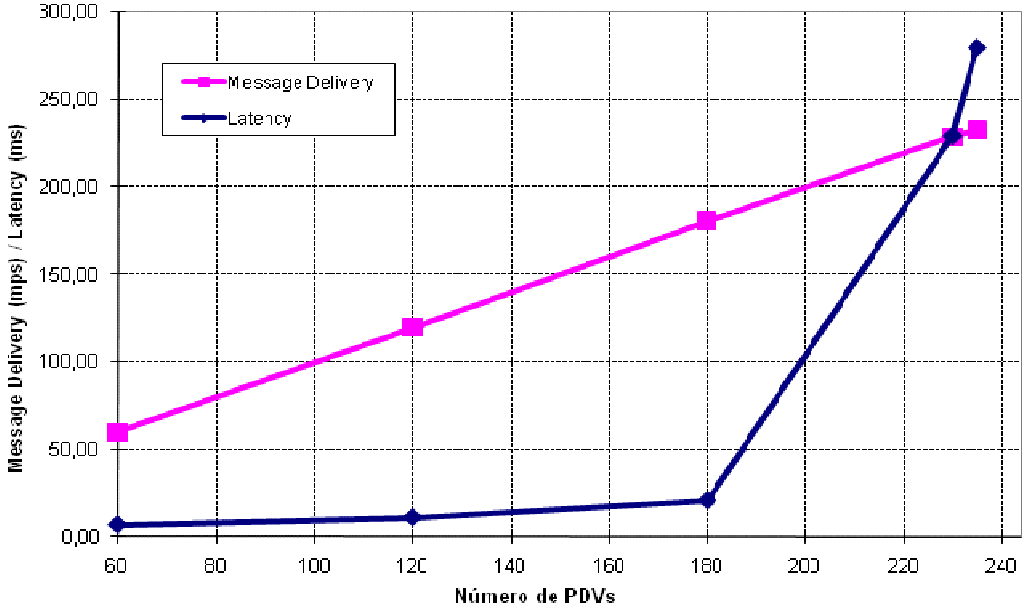


Figura 5-13. Cenário Aumento do Número de Lojas: interação *Registro de Vendas* com 30rps (Tabela 5-6).

Esse gráfico apresenta a latência versus a produtividade do canal, mostrando uma curva assintótica na produção, enquanto que a latência tem um crescimento exponencial, formando tipicamente uma configuração de ponto de saturação, conforme comentado na Seção 5.8.1.

Outra conclusão nessa análise é que no cenário de carga *Aumento no Número de Lojas*, os resultados obtidos nas duas interações (ver Tabela 5-7 e Tabela 5-8) leva à conclusão que o número máximo de lojas suportado pelo serviço de integração mantendo o SLO é nove (ou 540 PDVs). O gráfico apresentado na Figura 5-14 mostra o resultado da primeira interação, onde o número máximo de PDVs fica em torno de 600 mantendo a utilização de CPU abaixo de 70%.

Ao acrescentar o resultado da segunda interação o limite de PDVs diminui para 540, tornando-se a maior carga de trabalho submetida que faz o MOM responder atendendo todos os SLO das duas interações. O consumo de CPU estimado é 63,21%, a latência na primeira interação é de 4,55ms, enquanto que na segunda é de 1,15s para a soma das duas linhas de comunicação.

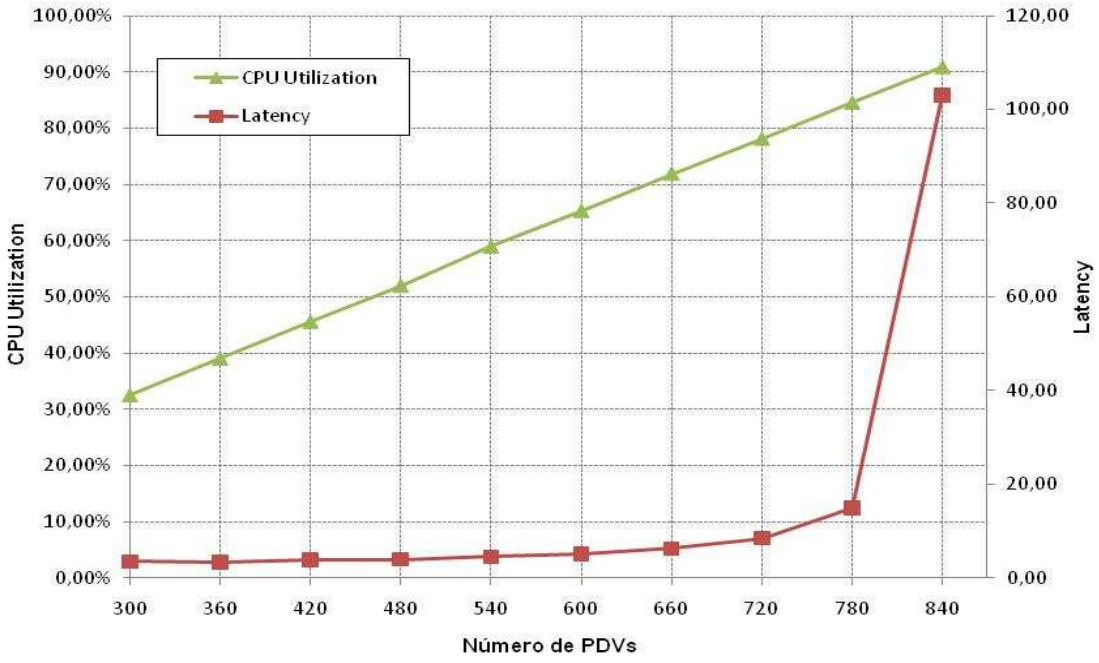


Figura 5-14. Cenário Aumento do Número de Lojas: interação *Registro de Vendas* com 150rps (Tabela 5-7).

Outras análises podem ser obtidas, porém este guia não possui a intenção de detalhar esta atividade, apenas indicar o que deve ser feito e ressaltar que é muito provável que durante as análises sejam necessárias avaliações de novos cenários e possibilidades, tornando essas duas últimas atividades do processo cíclicas e iterativas.

5.10 Considerações Finais

Neste capítulo foi demonstrado a utilização do JMSCapacity no auxílio do planejamento de capacidade de MOMs. Para isso, um cenário ilustrativo foi cuidadosamente escolhido para prover ao leitor um aprofundamento prático deste *toolkit*. Com isso, considera-se validado o processo proposto, a biblioteca de componentes, assim como as ferramentas.

CAPÍTULO

6

Trabalhos Relacionados

"Se eu vi mais longe, foi por estar de pé sobre ombros de gigantes."

Isaac Newton.

Desempenho é uma qualidade intrínseca dos produtos de software. Muitos são os fatores que podem afetá-lo, desde aspectos de construção do próprio software, até relativos à plataforma operacional, rede de comunicação, camada de middleware, etc. Muitos são os esforços para descrever e melhorar o desempenho dos sistemas de software. Esses trabalhos normalmente procuram resolver o problema através de duas possíveis abordagens: processo de medição ou avaliação através de modelos (analíticos ou de simulação) [Woodside et al. 2007].

Existem na literatura várias iniciativas no sentido de avaliar desempenho de sistemas de MOM. Esse capítulo traz para o leitor uma análise criteriosa dos principais trabalhos científicos desenvolvidos na área de avaliação de desempenho de software, mais especificamente na avaliação de plataformas de MOM. A análise tem o objetivo de mapear o estado da arte na área de estudo, fazendo uma comparação entre os trabalhos apresentados com a proposta desta dissertação.

Para tornar mais didática a apresentação do conhecimento científico produzido nos últimos anos, os trabalhos são divididos baseados na técnica utilizada para realizar a avaliação: medição ou modelagem.

Os últimos avanços na definição de *benchmarks* para plataformas de MOM são apresentados em separado, pois apesar de definirem ambientes tipicamente de medição, possuem características particulares que interessam a esta dissertação.

Por fim, alguns trabalhos que se referem especificamente a planejamento de capacidade de sistemas de software (inclusive *middleware*) também são tratados em separado, devido à relevância deles com o objetivo central desta dissertação.

6.1 Medição

Os trabalhos que utilizam medições, de uma forma geral, ressaltam a dificuldade imposta pelo método de avaliação para preparar o ambiente de teste e tratar os resultados, além, obviamente, do custo de se manter réplicas da infra-estrutura de produção quando se pretende realizar testes de *stress* para identificar pontos de saturação do sistema [Menascé e Almeida 2002].

Nas próximas subseções vários trabalhos de avaliação de desempenho de plataformas de MOM utilizando medições são apresentados. A importância desses trabalhos no desenvolvimento do JMSCapacity é indiscutível, pois a escolha de métricas, as dificuldades e limitações enfretadas durante as medições, a metodologia utilizada são contribuições fáceis de observar ao longo da apresentação desses trabalhos relacionados.

É importante ressaltar que apesar do JMSCapacity utilizar uma abordagem baseada em modelos para avaliar o desempenho do MOM, durante o processo descrito é necessário realizar medições para parametrizar e validar o modelo, onde as comparações com esses trabalhos podem ser realizadas.

6.1.1 Microsoft Message Queue e IBM WebSphere MQ Series

O relatório [NSTL 2000] realiza uma avaliação comparativa de desempenho entre o Microsoft Message Queue v2.0 (MSMQ) e o IBM MQSeries v5.1 através de técnicas de medição.

O estudo apresenta resultados que avaliam o desempenho de cada MOM individualmente, e depois um cenário onde ambos operam juntos através de uma conexão (*bridge*). Em cada um dos três cenários (*MSMQ standalone*, *MQSeries standalone*, *MSMQ bridge MQSeries*), são variados o tamanho das mensagens e a quantidade de threads. Para os dois primeiros cenários, são avaliadas as métricas: vazão do sistema (quantidade de mensagens por segundo) e utilização de CPU (durante o envio e a recepção de mensagens). No terceiro, a métrica da utilização da rede é adicionada ao estudo.

A IBM apresentou vários relatórios técnicos [IBM 2000a] [IBM 2000b] [IBM 2001]. Nos dois primeiros estudos eles refazem a comparação de desempenho dos mesmos produtos (MQSeries v5.1 e MSMQ v2.0) utilizando cenários bem mais elaborados e mais próximos de cenários reais de utilização de plataformas de MOM. No último, eles atualizam todos os testes para avaliar o IBM MQSeries v5.2.

Como resultado de todos os testes no MQSeries a IBM desenvolveu uma ferramenta (*Performance Harness for JMS*) para avaliação de JMS Providers [IBM 2005]. Essa ferramenta é testada e validada em uma nova avaliação do IBM Websphere MQ v6.0 (nova designação do MQSeries), onde os resultados são apresentados em [Carter 2005].

Todos esses estudos da IBM são conduzidos através de medições dos sistemas utilizando cenários tanto com o modo de entrega expresso (sem confiabilidade)

quanto com o modo transacional e persistente. Em ambos os cenários são avaliadas as métricas de vazão do sistema e taxa de entrega de mensagens variando o tamanho das mensagens, o número de clientes e sessões por cliente e o número de mensagens pré-carregadas na fila.

A IBM conduziu as medições em situações de estresse do sistema de entrega de mensagens. Desde o produtor até o consumidor, passando pelo MOM, o ambiente foi estressado próximo ao máximo no intuito de avaliar as métricas em condições, segundo o estudo, mais próximas da realidade.

Uma importante contribuição desse trabalho é a elaboração de cenários de avaliação mais compatíveis com o mundo real, levando em consideração aspectos relevantes quanto ao desempenho do MOM (persistência, transação).

Outra contribuição desse estudo é a discussão quanto à escalabilidade dos sistemas de entrega, avaliando cenários crescentes de carga (aumentando o número de clientes significativamente). Isso leva ao crescimento do tamanho das filas de mensagens e podendo ser avaliado a estabilidade do MOM sob circunstâncias extremas de carga. O estudo apresenta que os efeitos são potencializados em cenários com entrega confiável utilizando mensagens persistentes.

6.1.2 Projeto MTE (*Middleware Technology Evaluation*)

O CSIRO apresentou um relatório [Tran et al. 2001] avaliando três dos principais competidores no mercado de plataformas de MOM: Microsoft Message Queue v2.0, IBM MQSeries v5.2 e TIBCO Rendezvous v6.6. Esse relatório realiza medições dos sistemas submetendo-os a vários cenários. O objetivo principal é identificar os pontos fortes e fracos de cada plataforma fazendo com que a escolha do sistema mais adequado dependa da determinação de quais desses pontos são mais relevantes em cada situação específica. Os resultados da avaliação do produto da IBM são também apresentados de forma detalhada no artigo [Tran et al. 2002].

Esse projeto realiza a avaliação dos produtos através de medições em um ambiente de teste controlado. Além disso, eles definem uma métrica para a avaliação dos produtos e analisam o impacto no desempenho de alguns fatores, como o tamanho de *buffers* e o mecanismo de persistência.

Segundo o CSIRO um aspecto chave da medição de desempenho de um sistema de MOM é o estabelecimento de um cenário que melhor reflita o “mundo real” das aplicações baseadas em MOM. Baseado nesses cenários é necessário definir uma métrica para que a comparação de desempenho possa ser estabelecida.

Ainda segundo os pesquisadores, existe um número grande de fatores que podem influenciar o desempenho de um sistema de entrega de mensagens, entre eles: a qualidade do serviço de entrega determinada pelo nível de confiabilidade do canal de comunicação; os parâmetros de configuração do sistema; o tamanho das mensagens; e o número de *threads* utilizado pelos consumidores.

Os testes são realizados através de uma aplicação *multi-threaded* representando vários produtores enviando mensagens sobre uma rede comutada para uma aplicação também *multi-threaded* simulando vários consumidores. Os produtos MQSeries e MSMQ são testados com uma única fila e um único gerente de fila

em cada um dos lados (produtor e consumidor). Ou seja, não existe nesses cenários o papel do JMS Provider centralizado.

A aplicação que simula os produtores permite que seja configurada uma taxa de envio desejada variando para isso o número de *threads* destinado a realizar o envio de mensagens e o atraso (tempo de espera) entre um envio e outro. O número de *threads* na aplicação consumidora também permite variação, o que facilita a investigação do impacto desse aspecto na vazão total do sistema.

Em geral, os pesquisadores configuraram as plataformas de MOM de acordo com as recomendações fornecidas pelos fabricantes dos produtos. A exceção foi quanto à recomendação da Microsoft de configurar o MSMQ sobre um sistema de múltiplos discos para acelerar o tratamento de mensagens em cenários transacionais e com persistência.

O experimento de medição foi conduzido de forma que os consumidores são iniciados antes dos produtores, para evitar que mensagens sejam acumuladas na fila. Cada *thread* da aplicação produtora fica em *loop* enviando mensagens para os consumidores e esperando um pequeno intervalo de tempo antes de enviar outra mensagem. Já as *threads* consumidoras estão em loop contínuo buscando mensagens na fila. Os experimentos utilizam apenas mensagens dos produtores para os consumidores, sem exigir resposta, e somente ponto-a-ponto (PTP).

Os atributos básicos que foram variados durante o teste são: taxa de envio de mensagens, formada pelo número de *threads* de produção e pelo intervalo entre mensagens (10 *threads* no cenário com persistência e 50 *threads* para os demais cenários); número de threads de consumo (1, 5, 15, 20, 30 *threads*); e tamanho da mensagem (64, 1024, 4096 bytes).

Os resultados obtidos pelo projeto determinam que o sistema possa estar operando em dois possíveis estados: sustentável ou insustentável. O que determina em qual estado o sistema se encontra é a relação entre a taxa de envio e a taxa de recepção de mensagens. Se a relação entre essas taxas faz com que o tamanho da fila permaneça estável, é dito que o sistema está num estado sustentável. Caso contrário, o sistema tende a preencher a fila com mensagens, até que os recursos computacionais que suportam a implementação do mecanismo de armazenamento esteja completamente saturado. Nesse cenário, é dito que o sistema encontra-se em um estado insustentável.

A partir dessa observação, o projeto definiu a métrica da Vazão Máxima Sustentável (*Maximum Sustainable Throughput*, MST) como a principal medida de comparação e avaliação de plataformas de MOM. Ela representa a vazão do sistema no seu ponto de saturação. Esse nível de vazão não causa um incremento significativo da fila, permitindo que o sistema consiga mantê-lo sem exaurir os recursos computacionais. A medição dessa métrica deve ser conduzida com a fila inicialmente vazia, e com ambos os produtores e consumidores em execução.

O artigo [Tran et al. 2003] demonstra a importância da métrica MST para a avaliação de sistemas de MOM, testando três competidores voltados para integração de sistemas (*Message Brokers*): IBM Websphere MQ/MQIntegrator, TIBCO Rendesvous/MessageBroker v4.0 e Mercator Integration Manager v6.0.

Quanto à qualidade do serviço de entrega, as plataformas de MOM normalmente oferecem três níveis de serviço: melhor esforço, persistente e transacional. A partir da combinação desses níveis de serviço o projeto MTE definiu três alternativas para realizar as medições: NPNT (sem persistência e não transacional), PNT (persistente e não transacional) e PT (persistente e transacional). A partir de então, essa nomenclatura passou a ser largamente adotada nos processos de avaliação de sistema de entrega de mensagens.

Quantos aos demais fatores o projeto também inferiu através dos resultados obtidos que eles afetam de forma significativa o desempenho do MOM. O MST decrementa quase linearmente com o incremento do tamanho da mensagem para cenários NPNT. A persistência (PNT) faz com que o MST reduza de patamares próximos a 3.000 mps (mensagens por segundo) para cerca de 130 mps. Enquanto a escalabilidade para o modelo PNT é fraca, para os modelos transacionais eles determinaram que ela pode chegar a seis vezes com o acréscimo de *threads* de consumo.

As principais limitações desse estudo são:

- Os resultados de desempenho são específicos para os casos de teste desenvolvidos no projeto. Outras aplicações podem ter comportamento diferente, pois podem fazer uso de características das plataformas de MOM diferentes das avaliadas nesse estudo. Porém, o estudo afirma que o teste de comunicações PTP é predominante nas aplicações que fazem uso de sistemas de MOM e que espera que os resultados sejam relevantes em muitos cenários de aplicações corporativas;
- Os resultados de desempenho são específicos para a plataforma de hardware e rede utilizada. Ambientes diferentes (servidores, rede, sistemas operacionais) podem produzir resultados diferentes. Segundo eles, infelizmente não existe forma absolutamente correta de saber o desempenho em uma plataforma tecnológica diferente sem fazer as medições;
- Os casos de teste (cenários de avaliação) não utilizam o estilo de comunicação *publish/subscribe* (*Pub/Sub*), não sendo possível avaliar essa característica dos sistemas de MOM testados. Eles afirmam que esperam resultados bem diferentes para cenários *Pub/Sub* devido a grande diferença de comportamento dos sistemas de MOM nesse modelo de comunicação.

6.1.3 Sonic MQ

O relatório técnico [Jahming 2001] apresenta os resultados obtidos no estudo que apresenta o desempenho de produtos em conformidade com a especificação JMS. Os *JMS Providers* avaliados são o SonicMQ v4.0 e o IBM MQSeries v5.2.

Nenhum procedimento de *tuning* é realizado nos produtos avaliados, isso significa que os resultados obtidos podem ser maiores se tais procedimentos fossem adotados. Segundo os autores, o propósito deste estudo é testar e ilustrar as características de desempenho desses JMS Providers. Eles admitem que os resultados não devem ser projetados para outras situações e que eles

demonstram apenas o desempenho dos produtos nesses cenários, e na plataforma escolhida.

Para realizar os testes a Sonic Software desenvolveu um JMS Client chamado *MsgRunner* que permite uma grande variedade de configurações de cenários de teste. Esse cliente implementa a interface JMS fazendo com que o mesmo cliente possa ser utilizado para avaliar qualquer JMS Provider sem nenhuma alteração. Esse aspecto torna a avaliação um pouco mais isenta que outras iniciativas que utilizam clientes específicos para cada produto a ser avaliado.

Todos os experimentos são conduzidos sob as seguintes condições:

- Todas as conexões dos clientes são estabelecidas antes do período de *ramp-up* iniciar, não estando incluídas na janela de medição;
- Cada experimento inclui dois minutos para o período de *ramp-up* e dois minutos para *ramp-down*;
- Cada janela de medição tem quinze minutos de duração;
- Cada cliente roda apenas uma conexão JMS;
- Todos os testes são executados múltiplas vezes, garantindo a propriedade de repetição, e a obtenção de dados estatisticamente válidos;
- As mensagens são recebidas pelos consumidores de forma assíncrona, e utilizando o método de reconhecimento AUTO_ACKNOWLEDGE.
- Exceto durante os testes do cenário *Pub/Sub* com mensagens NonDurable com configuração de um produtor para muitos consumidores, nenhuma máquina cliente excede 50% da utilização de CPU, garantindo que nenhum gargalo é inserido por estes equipamentos;
- O banco de dados (*storage*) utilizado para persistir as mensagens é inicializado antes de cada experimento, garantindo que nenhum *overhead* é inserido no experimento para realizar tal limpeza;
- De forma equivalente, todas as filas ou tópicos são esvaziados, deletados e recriados antes de cada experimento;
- Durante os experimentos nenhuma outra aplicação está rodando e/ou consumindo recursos do SUT;
- Produtores são configurados para gerar mensagens tão rápido quanto possível, isto é, sem nenhuma inserção de *think time* entre as mensagens;
- Não é testado nenhum cenário com utilização de comunicação transacional.

A configuração dos clientes é realizada de forma que cada produtor e consumidor estejam em conexões separadas. Quando produtores e consumidores compartilham a mesma máquina, elas estão sobre a mesma JVM, porém em threads distintas. São utilizadas cerca de 10 máquinas clientes para gerar a carga de trabalho para o servidor.

Ambos os estilos de comunicação (PTP e *Pub/Sub*) são avaliados neste estudo. Quarenta cenários distintos são avaliados para cada um dos JMS Providers, que

executaram com suas configurações padrões, ou com pequenas modificações para permitir a avaliação dos cenários.

Desses cenários, quatorze utilizam o estilo PTP (todos utilizam mensagens com persistência), sendo os oito primeiros utilizando uma comunicação um para um, outros quatro simulando comunicação de poucos para muitos, enquanto que os últimos dois avaliam cenários de muitos para poucos. No primeiro grupo de cenários o objetivo é testar a capacidade do MOM em manipular um número crescente de conexões simultâneas. No segundo, o objetivo é avaliar a capacidade de encaminhamento de mensagens sem enfileiramento. Já no último grupo o intuito é testar os mecanismos de fluxo de mensagens com a presença de enfileiramento.

Os cenários *Pub/Sub* são elaborados a partir da variação do número de produtores, de consumidores e de tópicos, além do tamanho da mensagem e do modelo de assinatura (Durable ou NonDurable). Todos os cenários com assinaturas Durable adotam mensagens persistentes, enquanto que no caso de NonDurable as mensagens são não persistentes.

Os primeiros dezesseis cenários *Pub/Sub* simulam uma comunicação um para um, com o incremento do número de conexões e tópicos. Nesses experimentos, um produtor está produzindo mensagens para um único *Destination*, que por sua vez é assinado por um único consumidor. Os próximos oito cenários adotam uma comunicação de poucos para muitos. Nesses casos, existem múltiplos consumidores como assinantes de um tópico, fazendo com que cada mensagem enviada para o tópico seja encaminhada para cada assinante. Os últimos dois cenários *Pub/Sub* simulam uma situação de muitos para poucos, ou seja, os avaliadores pretendem avaliar o comportamento do controle de fluxo (controle de admissão).

Todos esses experimentos avaliam o desempenho do MOM através do acompanhamento de duas métricas: Taxa de Entrega de Mensagens e Latência.

A taxa de entrega de mensagens representa a razão entre a quantidade de mensagens recebidas pelos consumidores dentro da janela de medição e o tempo de duração da janela. Essa taxa pode se diferenciar da taxa de envio de mensagens, que não é objeto de medição do estudo.

A latência representa a quantidade média de tempo entre o momento que o produtor está pronto para enviar uma mensagem até o momento que o consumidor a recebe. Segundo os pesquisadores, nem todas as mensagens são incluídas nessa monitoração, pois o objetivo dessa métrica é ilustrar o efeito do MOM no encaminhamento de mensagens.

Uma conclusão importante deste estudo [Jahming 2001] é que a avaliação somente foi possível porque os JMS Providers implementam uma interface padrão (API JMS). Apesar disso, os resultados possuem bastante interferência por causa de fatores externos a configuração dos cenários, como: configuração do JMS Provider; plataforma operacional do servidor (sistema operacional, JVM, sistema de disco, quantidade de memória); configuração da plataforma operacional dos clientes; configuração da rede.

Em outro trabalho [Sonic 2003], é apresentada uma comparação do SonicMQ v5.0.2 com o TIBCO Enterprise for JMS v3.1. Nesse estudo, os autores

realizam medições em ambiente de teste, para cenários *Pub/Sub*, variando o nível de confiabilidade do serviço de entrega de mensagens (modo de entrega e integridade transacional).

Segundo os autores os cenários representam situações de estresse para aplicações do mundo real, como: aplicações para o sistema financeiro, comunicação de atualização de preço em empresas de varejo, distribuição de preços para portais B2B, aplicações de monitoramento na área de Telecom.

Os cenários para os experimentos definem sempre um número igual de produtores e consumidores (*publishers* e *subscribers*). Os testes examinam o desempenho do MOM suportando muitos clientes simultaneamente.

Durante esse trabalho a Sonic Software criou um *benchmark* para avaliação de desempenho de JMS Providers, disponibilizando o pacote de ferramentas *TestHarness* para download em seu site (http://www.sonicsoftware.com.br).

Os cenários são identificados pela relação entre Produtores, Tópicos e Consumidores como 10/10/10 (P/T/C), indicando que dez produtores se comunicam com dez consumidores, através de dez tópicos. O que determina o número máximo de clientes suportados no experimento é a garantia de que a plataforma cliente não estará saturada quanto a CPU ou memória.

A duração dos experimentos é de trinta e três minutos, sendo os dois primeiros considerados como *ramp-up*, e o último como *ramp-down*. Ou seja, a janela efetiva de medição é de trinta minutos. Os produtores e os consumidores executam em máquinas cliente diferentes, e se conectam com o servidor através de uma rede dedicada e controlada.

Os experimentos utilizam cinqüenta mensagens por transação entre o produtor e o JMS Provider. A sessão de entrega das mensagens (entre JMS Provider e o consumidor) não é transacional. O método de reconhecimento das mensagens é DUPS_OK_ACKNOWLEDGE para as sessões não transacionais.

O pacote de ferramentas possui algumas limitações, entre elas: uma única métrica avaliada (taxa de entrega de mensagens); geração de carga a partir de vários clientes (distribuída) é iniciada manualmente; não existe tratamento estatístico automatizado dos resultados.

6.1.4 FioranoMQ

A Krissoft Solutions realiza um estudo comparativo entre o FioranoMQ v7.5 e os principais JMS Providers de mercado: SonicMQ v6.0, Tibco EMS v4.0 e IBM WebSphereMQ v5.3. Os resultados desse estudo são apresentados no relatório técnico [Krissoft 2004].

O estudo avalia o desempenho dos JMS Providers utilizando o estilo de comunicação *Pub/Sub*. Segundo os autores, essa avaliação testa cenários de estresse com condições equivalentes a aplicações do mundo real. Em todos os cenários apenas um JMS Provider é responsável em encaminhar as mensagens de muitos produtores para muitos consumidores.

A metodologia utilizada para os testes são as mesmas desenvolvidas e publicadas pela Sonic Software [Sonic 2003] [Sonic 2004]. Todos os procedimentos já discutidos em [Sonic 2003] são rigorosamente seguidos,

inclusive o fato que todos os JMS Providers são configurados com opções padrão.

Todos os testes são conduzidos sobre as condições estabelecidas em [Jahming 2001] com exceção de:

- O método de reconhecimento utilizado por todos os consumidores é DUPS_OK_ ACKNOWLEDGE;
- Todas as configurações da JVM são exatamente as mesmas para todos os produtos, mesmo quando recomendado a utilização de algum parâmetro especial para melhorar o desempenho de algum dos produtos em particular.

Segundo os autores, os experimentos são realizados para os dois mais populares modelos de cenário *Pub/Sub*:

- Produtores gerando mensagens Não Persistentes e consumidores com assinatura NonDurable. Esse modelo é tipicamente utilizado por aplicações que trocam grandes volumes de mensagens e tem um requisito de baixa latência;
- Produtores gerando mensagens persistentes e consumidores com assinatura Durable. Esse modelo é normalmente empregado por aplicações que necessitam do máximo de confiabilidade no canal de comunicação, incluindo redundância, entrega única de cada mensagem independente de falhas.

Para cada modelo de cenário são definidos vários cenários com o intuito de avaliar a escalabilidade do JMS Provider em duas dimensões:

- Testes de Escalabilidade do Servidor – observam as características de desempenho do JMS Provider variando o número de tópicos utilizados, mantendo para cada um deles um número fixo de clientes. Ou seja, os resultados avaliam a capacidade do MOM de manipular várias conexões simultâneas, atendendo assim a um número crescente de clientes.
- Testes de Escalabilidade do Tópico – observam as características de desempenho do JMS Provider variando o número de clientes associados a um tópico, mantendo o número de tópicos fixo. Ou seja, os resultados avaliam a capacidade do MOM de manipular um número crescente de clientes associados a um mesmo tópico.

Um aspecto importante dos experimentos é que não foi adicionado nenhum tempo de processamento para os produtores ou consumidores, fazendo com que ambos produzam e consumam mensagens o mais rápido possível.

Os experimentos são conduzidos utilizando uma topologia formada por duas máquinas: uma para executar os clientes (tanto produtores quanto consumidores), e outra para executar o servidor (JMS Provider). Quanto à métrica utilizada para comparar os produtos, é avaliada a taxa de entrega de mensagens, conforme especificado nos trabalhos da Sonic Software.

6.1.5 Considerações sobre os trabalhos de medição

Ao longo dos trabalhos de medição pode-se observasr uma divergência de resultados, justificada pela falta de uma padronização na carga de trabalho

submetida aos produtos durante os testes. Essa padronização somente é conseguida através da definição de *Benchmarks* aceitos por todos os fabricantes e normalmente construída por entidades que possuem este papel institucional (ex. SPEC, TPC).

Apesar dos trabalhos de medição dos fabricantes possuírem essa falha de concepção e não contribuírem muito para a comparação entre os produtos, são inegáveis os avanços obtidos na área de avaliação de desempenho de MOM. Entre esses avanços destaca-se:

- Definição de métricas para avaliação, como Taxa de Entrega de Mensagens e Latência;
- Definição de duas dimensões para avaliar as plataformas de MOM: escalabilidade do servidor e do tópico;
- Definição de procedimentos de medição específicos em sistemas de MOM (esvaziamento da fila/tópico antes de cada medição, limpeza da base de dados de mensagens persistidas, utilização de uma única conexão por cliente, possibilidade de compartilhamento da mesma máquina para produtores e consumidores);
- Utilização de procedimentos gerais de medição (janela de medição, período de *ramp-up*, período de *ramp-down*, repetição das medições devido à necessidade de tratamento estatístico dos dados).
- Definição dos cenários NPNT (NonPersistent-NonTransact), PNT (Persistent-NonTransact), PT (Persistent-Transact), que identificam os níveis de serviço comumente utilizados por aplicações que utilizam os sistemas de MOM.

Todos esses trabalhos de medição demonstraram a necessidade de criação de ambiente de testes com características similares a um ambiente de produção, o que com certeza torna o processo caro e com elevado esforço.

Diante dessa dificuldade somada ao fato que o propósito desta dissertação é auxiliar no processo de planejamento e gerenciamento de capacidade, o que gera a necessidade de avaliar múltiplos cenários, o JMSCapacity adota técnicas de modelagem que permitem predizer o desempenho do sistema em cenários onde o custo de reprodução seria inviável.

Porém, durante a atividade de *Parametrizar, Validar e Calibrar o Modelo* faz-se necessário a realização de medições. O JMSCapacity apresenta uma ferramenta para conduzir as medições de forma a tornar-se independente de produto. A única exigência é que o MOM seja compatível com a especificação JMS.

6.2 Benchmarks

Na literatura, existem duas referências de benchmarks que se destacam para avaliação de MOM. O primeiro surgiu por iniciativa da indústria, e na verdade não se trata de um *benchmark* e sim de uma metodologia para construção de *benchmarks* para plataformas de MOM compatível com a especificação JMS [Sonic 2004] [Sonic 2007]. O segundo é resultado da iniciativa da *Standard Performance Evaluation Corporation* (SPEC), uma das mais respeitadas instituições responsáveis por criação de *Benchmarks*, e denomina-se

SPECjms2007 [SPEC 2007], *benchmark* para avaliação de desempenho de *JMS Providers*.

A importância desses trabalhos para o desenvolvimento do JMSCapacity está relacionado com a identificação e formalização de cenários de avaliação, a identificação e definição de métricas de desempenho, a caracterização da carga de trabalho, além de contribuir com a representação das interações entre as aplicações através de diagramas de sequência, o que facilita a documentação do *Plano de Integração*.

6.2.1 Progress/Sonic Software

A Sonic propôs em [Sonic 2004] a definição de um processo sistemático e estruturado de *benchmark* para sistemas de MOM. Esse trabalho apresenta cenários típicos de utilização de um sistema de entrega de mensagens, além de uma abordagem objetiva e direta para condução do processo, descrevendo as principais métricas a serem analisadas, uma metodologia para medição, assim como os fatores que as afetam.

Inicialmente, o trabalho define as seguintes métricas para realizar a avaliação do sistema de MOM: Taxa de Entrega de Mensagens (mensagens por segundo, mps) e Vazão do Sistema (*bytes* por segundo, bps). Em seguida, os autores elencam algumas perguntas que precisam ser respondidas durante o planejamento do *benchmark*:

- Como os resultados podem ser interpretados?
- Qual a duração mínima necessária dos experimentos?
- Com que freqüência as observações (medidas) devem ser coletadas?
- Como pode ser construído um cenário realístico?
- Quantos produtores e consumidores devem ser utilizados para que se obtenha um cenário realístico?

Para responder essas questões, o trabalho analisa um conjunto de características comuns em sistemas de entrega de mensagens (algumas específicas para os que adotam a especificação JMS), como: topologia de instalação, domínio de troca de mensagem, duração do *benchmark*, modos de entrega das mensagens, requisitos de escalabilidade, volume e tamanho das mensagens.

Quanto à topologia de instalação o trabalho destaca três opções: muitos produtores para poucos consumidores, poucos produtores para muitos consumidores e muitos produtores para muitos consumidores. Com certeza existem aplicações reais que utilizam cada um desses três modelos de topologia, sendo necessário, portanto que todos eles sejam utilizados para avaliar o MOM em um *benchmark*.

Quanto ao domínio do sistema de troca de mensagens, os autores destacam que tanto comunicações PTP quanto *Pub/Sub* são importantes para representar cenários reais. Quanto a desempenho, os dois domínios são bem diferentes, logo, um *benchmark* precisa avaliar ambos.

Quanto à duração do *benchmark* os autores afirmam que o ideal seria testa os *JMS Providers* por meses, porém devido à inviabilidade desse procedimento, eles defendem que no mínimo os experimentos devem durar 24 horas para sistemas críticos operando em um regime de 24x7. Ainda segundo eles, esse

seria o tempo mínimo para avaliar possíveis congestionamentos de mensagens, crescimento do consumo de memória.

Quanto aos modos de entrega de mensagens, os autores destacam a necessidade de avaliar o sistema de persistência da solução. Em alguns casos, os *JMS Providers* para acelerar esse mecanismo definem *cache* em memória, diminuindo a confiabilidade da solução.

Quanto à escalabilidade, o trabalho ressalta que é importante avaliar tanto a capacidade do sistema em receber um número crescente de usuários conectados a um mesmo canal (*destination*), quanto avaliar a manipulação de um número crescente de canais.

Quanto ao volume e ao tamanho das mensagens, os autores ressaltam que esses dois parâmetros do componente básico da carga de trabalho impactam diretamente no desempenho, e por isso devem ser definidos cenários variando-os para avaliar o seu impacto.

Entre as conclusões do trabalho, uma das que se destaca é a necessidade de definição de cenários (incluindo a definição do ambiente) que reflitam situações realísticas e que exercitem o *JMS Provider* em todos os aspectos destacados. O trabalho finaliza concluindo que para tornar os resultados relevantes é necessário o atendimento a algumas condições: ambientes de teste similares aos de produção; experimentos de longa duração; cenários com múltiplos produtores e consumidores; garantia que todos os *JMS Providers* serão testados nas mesmas condições e com a mesma carga.

Em [Sonic2007], a Sonic Software ampliou o escopo das recomendações quanto à elaboração de *benchmarks* para avaliação de *JMS Providers*. Nesse novo estudo o contexto é o mesmo utilizado nesta dissertação: a utilização do MOM JMS como a base para o ESB em processos de integração e implantação de uma arquitetura orientada a serviços (SOA).

Os autores definem três passos para realizar um *benchmarking* entre plataformas de integração: planejar a necessidade futura de carga; comparar a eficiência dos vários modelos de integração; comparar o custo/benefício das soluções de entrega de mensagens. Isso mais uma vez ressalta o que este trabalho já vem afirmando que as plataformas de MOM são a chave para os modelos de integração atuais e futuros.

Outra contribuição importante que esse estudo traz é, segundo os autores, a constatação que nenhum processo substitui a experiência dos avaliadores. Ainda segundo eles, o ideal é o envolvimento de profissionais com experiência tanto em avaliação de desempenho quanto em MOM, para garantir que os resultados obtidos sejam relevantes.

Os avaliadores apresentam que para definir a carga de trabalho a ser submetida ao MOM durante os experimentos, dois parâmetros são fundamentais: taxa de produção de mensagens e o tamanho das mensagens. Para definir avaliar cenários realísticos de integração, os autores sugerem que os cenários possuam taxas de produção e tamanho das mensagens definidas através de matrizes, permitindo assim que em um mesmo cenário se possa ter um mix de mensagens diferentes, assim como produtores com comportamento de geração de carga distinto.

Uma inovação desse estudo [Sonic 2007] em relação à versão anterior [Sonic 2004] é a definição de novas métricas para avaliar os sistemas. O estudo anterior praticamente definia apenas a vazão como métrica de avaliação. Nesse novo estudo, além da vazão é definida a Latência e o número de Exceções. A Latência mede o atraso médio inserido no processo de entrega de mensagens. O número de exceções determina a quantidade de falhas, mensagens rejeitadas ou transações abortadas.

Além dessas métricas, o trabalho define a importância de se monitorar o consumo de recursos importantes para o funcionamento do sistema, e que sejam finitos. Para tanto eles definem outras métricas para serem acompanhadas: utilização de CPU, utilização de memória, threads contidas, utilização da rede, taxa de E/S.

6.2.2 SPEC

A SPEC juntamente com a participação de centros de pesquisas acadêmicos e da indústria (ex. TU-Darmstadt, IBM, Sun, BEA, Sybase, Apache, Oracle, JBoss) desenvolveu um benchmark para avaliação de JMS Providers denominado SPECjms2007 [SPEC 2007].

As publicações [Sachs et al. 2007a] e [Sachs et al. 2007b] apresentam esse *benchmark* descrevendo principalmente a sua caracterização de carga, enquanto que em [Kounev e Sachs 2008] os autores descrevem suas características funcionais e algumas peculiaridades importantes para esta dissertação, como um *framework* para avaliação de desempenho de MOM.

Esse *framework* compõe um importante diferencial do SPECjms2007. Ele permite que os avaliadores customizem a carga de trabalho para suas necessidades, configurando o nível de estresse desejado para a infra-estrutura de MOM, sendo capaz de com isso representar cenários específicos de carga. Entretanto, os autores ressaltam que para a utilização dessas facilidades é necessário o entendimento da forma com que a carga de trabalho é decomposta em componentes básicos, e como cada um desses componentes básicos influencia no desempenho do MOM.

O objetivo do SPECjms2007 é prover uma carga de trabalho padrão e métricas para medir e avaliar o desempenho e a escalabilidade de uma plataforma de MOM baseada na especificação JMS. Para conseguir atingir esse objetivo a carga de trabalho do SPECjms2007 precisa preencher alguns requisitos importantes.

Primeiramente, ele deve ser baseado em um cenário de carga de trabalho que seja representativo em relação à utilização no mundo real desse tipo de plataforma. Com isso os usuários, ao analisar os resultados do *benchmark*, podem observar comportamentos similares aos seus ambientes reais.

O segundo requisito é quanto à capacidade da carga de trabalho explorar todas as funcionalidades normalmente utilizadas nas plataformas de MOM, incluindo ambos os estilos de comunicação PTP e *Pub/Sub*. As funcionalidades e serviços são estressados pelo *benchmark* de acordo com o peso que eles são utilizados em cenários reais.

O próximo requisito é que a carga deve realizar a medição do desempenho e da escalabilidade, tanto da plataforma de MOM quanto da plataforma de hardware que suporta o sistema de entrega de mensagens. Ou seja, o resultado do *benchmark* determina o desempenho desse conjunto que não deve ser separado, pois um influencia diretamente no outro. O importante desse requisito é que ele deixa de fora a avaliação de sistemas auxiliares como sistemas gerenciadores de banco de dados, que poderiam comprometer a comparação dos resultados.

Finalmente, a carga de trabalho do SPECjms2007 não deve ter qualquer limitação inerente ao *benchmark*, sendo necessário que ela possa ser escalada incrementando tanto o número de *Destinations* (filas e tópicos) assim como a quantidade de tráfego que passa por cada um deles.

O cenário escolhido para o SPECjms2007 modela um sistema de gerenciamento da cadeia de suprimentos (SCM) de um supermercado. Participam dessa cadeia as seguintes entidades: matriz (HQ), fornecedores (SP), centros de distribuição (DC) e lojas (SM).

A carga de trabalho gerada nesse cenário simula situações típicas de troca de mensagens para integração entre as aplicações dessas entidades. Para representar essas situações o *benchmark* define sete interações básicas envolvendo troca de mensagens:

- Interação 1 – solicitação e atendimento de pedidos entre as lojas (SM) e os centros de distribuição (DC) via comunicação PTP;
- Interação 2 – solicitação e atendimento de pedidos entre os centros de distribuição (DC) e os fornecedores (SP) via comunicação PTP;
- Interação 3 – atualização de preços comandada pela matriz (SM) para as lojas (SM) via comunicação *Pub/Sub*;
- Interação 4 – inventário das lojas (SM), permitindo que a matriz (HQ) seja informada das posições de estoque, via comunicação PTP;
- Interação 5 – envio de estatística de vendas das lojas (SM) para a matriz (HQ) via comunicação PTP;
- Interação 6 – anúncio de novos produtos por parte da matriz (HQ) para as lojas (SM) via comunicação *Pub/Sub*;
- Interação 7 – envio de informações de crédito por parte da matriz (HQ) para as lojas (SM) via comunicação *Pub/Sub*.

Em cada uma das interações é produzido um conjunto de mensagens que varia em termos de formato, estilo de comunicação (PTP ou *Pub/Sub*), *Destination*, modelo de confiabilidade (persistência, integridade transacional), modelo de assinatura (*Durable*, *NonDurable*). A Tabela 6-1 apresenta a lista de todas as mensagens trocadas em cada interação.

Tabela 6-1. Tipos de mensagens utilizadas em cada interação do SPECjms2007.

Int.	Mensagem	Destination	Formato	Pers	Trans	Dur
1	order	Queue (DC)	ObjectMsg	X	X	
1	orderConf	Queue (SM)	ObjectMsg	X	X	
1	shipDep	Queue (DC)	TextMsg	X	X	
1	statInfoOrderDC	Queue (HQ)	StreamMsg			

Int.	Mensagem	Destination	Formato	Pers	Trans	Dur
1	shipInfo	Queue (SM)	TextMsg	X	X	
1	shipConf	Queue (DC)	ObjectMsg	X	X	
2	callForOffers	Topic (HQ)	TextMsg	X	X	X
2	Offer	Queue (DC)	TextMsg	X	X	
2	pOrder	Queue (SP)	TextMsg	X	X	
2	pOrderConf	Queue (DC)	TextMsg	X	X	
2	Invoice	Queue (HQ)	TextMsg	X	X	
2	pShipInfo	Queue (DC)	TextMsg	X	X	
2	pShipConf	Queue (SP)	TextMsg	X	X	
2	statInfoShipDC	Queue (HQ)	StreamMsg			
3	priceUpdate	Topic (HQ)	MapMsg	X	X	X
4	inventoryInfo	Queue (SM)	TextMsg	X	X	
5	statInfoSM	Queue (HQ)	ObjectMsg			
6	productAnnouncement	Topic (HQ)	StreamMsg			
7	creditCardHL	Topic (HQ)	StreamMsg			

O SPECjms2007 oferece três topologias diferentes para estruturar a carga de trabalho: horizontal, vertical e livre. As duas primeiras topologias avaliam a escalabilidade de um MOM em dimensões distintas. Na primeira dimensão (topologia vertical), a avaliação se concentra na capacidade de um *Destination* encaminhar mensagens (escalabilidade do tópico). Essa avaliação é obtida aumentando-se a quantidade de tráfego gerado por localização (loja, centro de distribuição, etc.), simulando um aumento na quantidade de negociações envolvendo um número fixo de entidades (localizações).

Na segunda dimensão (topologia horizontal) a avaliação está focada na capacidade de entrega de mensagens do servidor como um todo (escalabilidade do *JMS Provider*), avaliando principalmente o gerenciamento de múltiplos *Destinations*. Para conseguir isso se incrementa a quantidade de centros de distribuição (DC), lojas (SM), fornecedores (SP), enquanto que o tráfego gerado por cada localização é mantido constante.

Já a topologia livre refere-se exatamente ao *framework* provido em conjunto com SPECjms2007, que permite a configuração da carga de trabalho, fazendo com que o avaliador possa configurar um *mix* entre as interações diferente do padrão estabelecido nas outras topologias. O *framework* permite a configuração personalizada dos seguintes parâmetros: quantidade de localizações, tráfego gerado por cada localização, tamanho das mensagens, modelo de confiabilidade. O usuário pode ainda escolher seletivamente as interações desejadas.

6.3 Modelagem

Os trabalhos de Menascé et al. [Menascé 2005] [Menascé et al. 1994] [Menascé et al. 1999] [Menascé et al. 2004] [Menascé e Almeida 1998] [Menascé e Almeida 2000] [Menascé e Almeida 2002] [Menascé e Gomma 2000] [Almeida

e Menascé 2002] utiliza a técnica de modelagem com o formalismo das redes de fila (*Queueing Networks, QNs*). As QNs são um formalismo gráfico que tanto podem ser utilizados para avaliação analítica quanto para simulação. Esses trabalhos apresentam também fundamentação da técnica de medição, quase sempre para validar e complementar a avaliação realizada pelos modelos.

Os trabalhos de Kounev et al. [Kounev 2005] [Kounev 2006] [Kounev e Buchmann 2002] [Kounev e Buchmann 2003a] [Kounev e Buchmann 2003b] [Kounev e Buchmann 2006] [Kounev e Buchmann 2007] [Kounev et al. 2008] também são na área de avaliação de desempenho de sistemas distribuídos utilizando a técnica de modelagem. O formalismo escolhido é baseado numa junção das GSPNs (utilizada nesta dissertação), com as redes de Petri coloridas e as redes de filas, e é denominado *Queueing Petri Net* (QPN). As QPNs têm como um dos pontos fortes o poder de representação elevado, com um aumento da abstração e com isso uma maior facilidade para modelar sistemas distribuídos.

As próximas subseções trazem uma coletânea de alguns desses trabalhos, além de outros trabalhos desenvolvidos utilizando modelos para avaliação de desempenho de sistemas distribuídos, e sempre que possível focando na modelagem de plataformas de MOM.

6.3.1 Modelagem Analítica

A modelagem analítica é uma técnica de representação de sistemas através de formulações matemáticas capazes de reproduzir o comportamento do sistema quanto aos aspectos modelados.

Em [Baldoni et al. 2003], os autores propõem um framework que inclui um modelo computacional *publish/subscribe* capaz de capturar o ponto de vista da aplicação quanto ao comportamento esperado do sistema de troca de mensagem com respeito a semântica de notificação da informação.

Em [Baldoni et al. 2005], os autores propõem outro modelo para o mesmo sistema, agora utilizando dois parâmetros relativos a atrasos. O primeiro é relativo ao atraso no processo de assinatura de um tópico, e o segundo está relacionado com o atraso de difusão de informação. Esse modelo permite que seja avaliado o comportamento de execução do MOM em um estado transiente.

Esse *framework* permite medir analiticamente a efetividade do sistema, o qual reflete o percentual garantido de notificações que são entregues aos assinantes. Os autores validam os resultados analíticos através de estudos de simulação.

Como os modelos possuem uma baixa granularidade (alta abstração), ele é mais adequado para projetistas de serviços de notificação, que podem estimar a probabilidade de entrega das notificações apenas estimando dois parâmetros. Além do mais, análises mais refinadas (assim como estudos de simulação) devem considerar muitos parâmetros (da rede para a aplicação), o que faz com que seja muito difícil de ter uma avaliação de desempenho precisa e ampla em um ambiente prático. Isso, segundo os autores, devido ao comportamento dinâmico do paradigma *Pub/Sub* e do ambiente de rede.

O modelo desenvolvido apresenta dois aspectos característicos: explicitamente introduz o sistema Pub/Sub no modelo, em vez de considerar apenas a

semântica dos processos; e considerar a noção de tempo na especificação do sistema. Essas características permitem aos autores apontar o não determinismo dos sistemas Pub/Sub no nível de especificação. Ou seja, independentemente de questões específicas de implementação, assim como topologia de implantação ou algoritmo de difusão de informação. Por outro lado, eles não exploraram a especificação para fornecer uma prova formal do sistema Pub/Sub, mas apenas proveram uma simples análise probabilística.

Na opinião dos autores, o modelo de desempenho analítico desenvolvido pode ser utilizado como regra por usuários e projetistas de sistemas Pub/Sub para prever o comportamento das suas aplicações, identificando facilmente aspectos críticos de implementação que influenciam o desempenho geral do sistema. Isso pode representar uma alternativa precisa e rápida para avaliação de desempenho quando comparado a outros métodos mais complexos que exigem mais tempo de análise.

O grupo do Projeto MTE, bastante abordado na discussão sobre trabalhos de medição, produziu o artigo [Liu e Gorton 2005], onde apresenta uma abordagem para predizer, durante a fase de projeto (*design*), o desempenho de aplicações *Java Enterprise Edition* (JEE) que utilizam o serviço de troca de mensagens compatível com a especificação JMS. Para isso, os autores propõem um modelo analítico que captura o comportamento de desempenho dessas aplicações, predizendo o desempenho delas antes mesmo delas serem desenvolvidas.

Esse artigo é uma extensão do trabalho publicado em [Liu et al. 2004b], onde os autores desenvolvem uma abordagem para predição de desempenho de aplicações JEE com comunicação síncrona. Nessa extensão, eles realizam uma predição de desempenho tanto síncrona quanto assíncrona. Nesses dois trabalhos o foco de avaliação está na aplicação e não na plataforma de *middleware*.

A abordagem proposta possui três aspectos chaves. Primeiramente, o modelo de desempenho deve explicitamente representar o componente *container*, o serviço de MOM e suas comunicações com os componentes da aplicação. Em segundo lugar, o tempo de serviço de uma requisição depende de atributos do *container* e do MOM, e precisam ser representados no modelo sempre que afetarem o desempenho do sistema. Por último, é necessário um perfil de desempenho do *container* e do MOM independente da aplicação. Ou seja, a abordagem necessita que tenha sido realizada uma avaliação prévia de desempenho do MOM e do *container* onde será hospedada a aplicação.

Para tanto, os autores modelam as interações entre o JEE e os componentes do MOM utilizando o formalismo das QNs. Em seguida eles calibram o modelo de desempenho com atributos da arquitetura associados a esses componentes, e realizam a definição dos parâmetros do modelo a partir de resultados de *benchmarks*. A utilização de um *benchmark* evita a necessidade de realizar os testes utilizando protótipos para obter os parâmetros do modelo, reduzindo assim o esforço para predição de desempenho.

O *benchmark* desenvolvido nesse trabalho disponibiliza além de um modelo da infra-estrutura de componentes do JEE (simulando a aplicação), um gerador de carga de trabalho, um utilitário para monitoramento e um *toolkit* de mercado para *profiling* da plataforma JEE. A ferramenta de monitoramento é utilizada

para coletar métricas de desempenho do servidor de aplicação em tempo de execução. O *toolkit* para *profiling* serve para realizar *tracing* da execução de uma aplicação no servidor de aplicação, com o intuito de medir os tempos gastos em partes internas da infra-estrutura, fornecendo importantes parâmetros para os modelos. Essa ferramenta é muito útil em situações onde avaliações caixa-preta precisam ser realizadas, por exemplo, quando não é possível a instrumentação direta do código-fonte.

Para validar a abordagem eles apresentam um estudo de caso de predição de desempenho de uma aplicação JEE com comunicação síncrona e assíncrona, e obtêm resultados com margens de erro em torno de 15% quando comparado com medições realizadas.

Os autores vêm trabalhando na geração automática de modelos de desempenho a partir de modelos de projeto da aplicação. Essa abordagem permitirá avaliar o requisito não-funcional de desempenho em tempo de projeto, antecipando possíveis problemas na arquitetura da aplicação.

[Menascé 2005] propõe um *framework* quantitativo para comparar soluções baseadas em MOM e RPC. O objetivo do autor, através do modelo matemático, é realizar uma avaliação comparativa da arquitetura de comunicação (síncrona ou assíncrona) das aplicações.

Este *framework* simples indica que ambos os modelos de comunicação (assíncrono através de MOM e síncrono representado pelo RPC) possuem vantagens em termos de desempenho. O RPC é a melhor escolha quando a resposta do provedor do serviço é pequena e imediata, levando a uma penalidade de tempo para o processo chamador por suspender a execução enquanto espera por uma resposta. O MOM trabalha melhor para transações que requerem longo tempo de execução do SP. A maioria das aplicações reais combina situações em que algums SPs respondem muito rápido com outras em que levam um tempo longo antes de responder ao APC.Desta forma, ambos os tipos de modelo de comunicação, RPC e mensagem assíncrona devem ser usados no contexto apropriado.

6.3.2 Modelagem de Carga de Trabalho

Em [Menascé et al.1999], o autor propôs uma metodologia para caracterizar e gerar modelos de carga de trabalho para servidores *e-commerce*. Primeiro, ele introduz um grafo de transição de estado denominado *Customer Behavior Model Graph* (CBMG), que é usado para descrever o comportamento de grupos de clientes que exibem padrão de navegação similar. Um conjunto de métricas, analiticamente derivado da análise do CBMG, é apresentado. Então, um modelo de carga de trabalho é definido e são apresentados os passos necessários para obter seus parâmetros. É proposto um algoritmo que dimensiona o tamanho do cluster para atender o volume de usuários de sítios de *e-commerce* com comportamento semelhante modelado em termos de CBMG.

Em [Menascé e Gomaa 2000], os autores afirmam a necessidade de integrar ambas as atividades de modelagem de projeto e desempenho, de forma que uma possa ajudar a outra. Eles descrevem uma abordagem iterativa para projetar sistemas e seu desempenho antes de sua implementação. Esse método foi desenvolvido e usado pelos autores no desenvolvimento de aplicações

cliente/servidor grandes e complexas. O objetivo é analisar o projeto através da perspectiva do desempenho, para comparar alternativas de projeto, e para comparar sua execução em diferentes configurações de sistemas. Para fazer isso, é necessário definir o modelo de projeto no nível de granularidade da comunicação de messagens entre cliente e servidor, e definir o modelo de funcionalidade da aplicação no lado cliente e servidor para capturar a lógica da aplicação e ao padrão de acesso aos recursos do sistema.

O método proposto é baseado em uma linguagem de engenharia de desempenho de software desenvolvida por um dos autores. Casos de Uso são desenvolvidos e mapeados em uma especificação de modelo de desempenho usando essa linguagem. Um compilador para a linguagem gera um modelo de desempenho analítico para o sistema. Desta forma, é possível predizer que o desempenho de um sistema em desenvolvimento requer que a demanda do serviço seja estimada por recursos tais como CPU, disco e rede.

Em [Singhera 2008], o autor propõe um modelo para representar carga em um sistema *Publish/Subscribe*, e demonstra como tal modelo pode ser usado em *benchmarking* e geração de carga. Também apresenta a automação de um *framework* para auxiliar na avaliação e projeto de novas arquiteturas para sistemas *Pub/Sub*.

6.3.3 Modelagem de Middleware

No artigo [Kounev et al. 2008], os autores propõem uma metodologia para carga de trabalho e modelagem de desempenho de DEBS. Um modelo de carga de trabalho de um DEBS genérico é desenvolvido e técnicas de análise operacional são usadas para caracterizar o tráfego do sistema, derivando uma aproximação para o valor médio da latência. Em seguida, ele apresenta uma técnica de modelagem que pode ser utilizada para predição de desempenho. Por fim, é apresentado um estudo de caso real demonstrando a efetividade e a praticidade da abordagem proposta.

Enquanto inúmeras abordagens para modelagem de sistemas distribuídos convencionais e avaliação de seu desempenho e escalabilidade estão disponíveis na literatura, nenhuma metodologia foi proposta para DEBS. Trabalhos atuais sobre modelagem e avaliação de DEBS visam implementações ou aplicações específicas e são altamente especializadas. Este artigo é o primeiro que provê uma metodologia para caracterização de carga de trabalho e modelagem de desempenho de DEBS que é aplicável a uma grande variedade de sistemas.

Esta metodologia auxilia na identificação e eliminação de gargalos e assegura que sistemas são projetados e dimensionados para alcançar seus requisitos de QoS. A metodologia é baseada na análise operacional e QPNS (*Queueing Petri Nets*). Inicialmente, técnicas de análise analítica são usadas para achar a utilização de componentes de sistemas e derivar uma aproximação para o valor médio da latência. Depois, são apresentados como modelos de desempenho baseados em QPNs podem ser construídos para prover uma predição de desempenho mais precisa. A vantagem da abordagem proposta é que é prática e genérica, podendo ser utilizada para avalivação de desempenho de DEBS.

[Liu et al. 2003] propõe um modelo para avaliação de desempenho de um Middleware para Integração de Processos de Negócio (BPI). Este modelo é baseado em *Layered Queuing Networks* (LQN) e os resultados são validados com medições obtidas utilizando o IBM *CrossWorlds InterChange Server* (ICS), através da parametrização da latência das aplicações a serem integradas com o ICS.

O modelo pode ser usado como uma ferramenta de *tuning*, usado para identificar a configuração apropriada para um *pool* de *threads* de vários componentes ICS, tornando o ICS escalável em diferentes cenários de integrações. Além disso, pode ser usado como uma ferramenta de planejamento de capacidade para predizer o desempenho de um sistema sob uma crescente carga de trabalho.

[Liu et al. 2004a] é uma continuação do trabalho apresentado em [Liu et al. 2003]. Ao invés de construir um LQM manualmente, os autores propõem neste artigo um método e uma ferramente para gerar um LQM a partir de um modelo de aplicação, um modelo de *middleware* e uma bibilioteca de configuração de hardware, todos descritos em xml.

Esta abordagem tem as seguintes vantagens: o modelo de aplicação pode ser facilmente criado com base em um dado BPI com requisitos específicos nos processos de negócio e aplicações corporativas a serem integradas; o modelo de *Middleware* é parametrizado em xml, sendo fácil de manter à medida que o desempenho do produto aumenta; sempre que um novo *hardware* torna-se disponível, a razão entre desempenho e custo podem ser facilmente incorporadas na ferramenta de dimensionamento; bibliotecas de configuração de *hardware* com diferentes estruturas de custo podem ser usadas no dimensionamento de uma aplicação para estudo de estratégias de precificação.

6.3.4 Modelagem de Middleware utilizando GSPN

A série de artigos [Souza et al. 2006a] [Souza et al. 2006b] [Souza et al. 2008] discute e propõe modelos GSPN para representar o mecanismo de *pool* de instâncias embarcado em servidores de aplicação JBoss e demonstra como estes modelos podem ser usados na avaliação de desempenho.

O primeiro artigo apresenta resultados inciais com o desenvolvimento de um modelo apenas representando o servidor. Em seguida o próximo artigo apresenta um modelo detalhado cliente/servidor, o qual deve-se usar em simulações; e por fim o útlimo artigo apresenta um modelo abstrato, o qual é solucionado analiticamente.

Para validar estes modelos, alguns cenários são propostos. Os dois primeiros representam uns poucos clientes demandantes, enquanto o terceiro representa uma situação mais realista, com um grande número de clientes submetendo requisições a uma taxa menor. Além disso, três métricas são consideradas: número de instâncias, vazão e utilização de CPU.

Geralmente, os resultados obtidos usando simulação e técnicas de análise são próximas daquelas obtidas a partir de experimentos de medição, dando uma indicação da precisão dos modelos propostos.

Uma vez que a execução dos experimentos de medição não é uma tarefa fácil, ter modelos precisos para predizer desempenho é, claramente, uma vantagem muito importante.

Apesar da relevância dos resultados obtidos, os modelos propostos têm algumas limitações. Primeiramente, estes modelos não podem ser usados para predizer desempenho em cenários envolvendo outros tipos de componentes (exemplo bancos de dados e filas). Segundo, o tempo gasto em simulação de experimentos depende fortemente do número de clientes considerados. Como conseqüência, tempo pode ser um fator limitante para simulações de cenários envolvendo um grande número de clientes. Finalmente, contenção de software não é totalmente representada e afetando a predição do ponto de saturação.

[Fernandes et al. 2004] apresenta uma avaliação do IBM WebSphere MQ v5.3 realizada através de modelos desenvolvidos em redes de Petri estocásticas. Os resultados foram bastante satisfatórios fazendo com que o modelo obtivesse um bom nível de precisão nos resultados.

A precisão do modelo é comprovada através de comparações com medições realizadas em um ambiente de medição real. O modelo GSPN demonstrou ser muito preciso e bem sucedido em obter as principais características de desempenho de um MOM. Além disso, algumas análises de desempenho comprovaram o comportamento de algumas métricas diante de mudanças de diferentes fatores e seus respectivos níveis. Em geral, os resultados apresentam a habilidade de modelagem e flexibilidade em avaliar MOM usando GSPN.

Em [Arteiro et al. 2007], o autor investigou a utilização de modelos de desempenho para MOMs, desenvolvidos em redes de Petri estocásticas, no auxílio ao planejamento de capacidade deste tipo de middleware. Em específico define como objetivo apresentar modelos para MOMs, compatíveis com o padrão JMS (Java Message Service), de forma a permitir a identificação da (1) capacidade máxima de entrega de mensagens mantendo um nível de serviço acordado, e (2) do ponto de saturação destes sistemas.

Os resultados obtidos da simulação do modelo GSPN se aproximaram de forma significativa dos resultados obtidos através da medição, validando ambos, a abordagem e o modelo. Como o processo de medição possui um alto custo associado, utilizar modelos para avaliar objetivos de desempenho é uma importante contribuição desse artigo. Outro importante aspecto é o desenvolvimento de um modelo de referência, o qual pode ser parametrizado para qualquer ambiente de MOM, tornando o modelo reusável.

Apesar da relevância dos resultados obtidos, o modelo proposto possui algumas limitações. Inicialmente, ele representa apenas um MOM PTP, com um modelo específico de QoS (NPNT). Outra limitação é que o modelo não consegue ainda representar o comportamento do sistema pós-saturação, gerando algumas distorções. Por último, o modelo representa apenas um destinatário (fila).

6.3.5 Considerações sobre os trabalhos de modelagem

Os trabalhos que apresentam modelos analíticos ou de simulações utilizam, em sua grande maioria, ou modelos analíticos puramente matemáticos, ou modelos que representam um sistema de MOM específico, ou focam na avaliação de uma métrica apenas, ou ainda utilizam um processo de parametrização (calibragem)

do modelo a partir de técnicas invasivas (instrumentação de código fonte). O JMSCapacity propõe um forma de avaliar o desempenho do MOM a partir da definição e formalização do serviço de integração, o que faz com que a avaliação seja realizada com o foco na aplicação e não no MOM em si. Isso foi possível devido ao foco do trabalho ser de planejamento de capacidade.

6.4 Considerações Finais

Ao longo desse capítulo foram apresentados vários trabalhos relacionados com o tema de avaliação de desempenho de *middleware*. Inicialmente, os trabalhos de medição foram detalhados e ao final comparados com as atividades de medição desenvolvidas durante o JMSCapacity, mais especificamente na atividade de *Parametrizar,Validar e Calibrar o Modelo*. O JMSCapacity inovou na definição de múltiplas métricas de avaliação, na junção de várias idéias desenvolvidas por trabalhos diferentes, além da construção de uma ferramenta de geração de carga artificial parametrizável e com sumarização dos resultados.

Em seguida, os trabalhos de *benchmarking* foram analisados e contribuíram com este trabalho principalmente na caracterização da carga de trabalho. A inovação do JMSCapacity foi utilizar o conceito da caracterização de carga utilizado nos *benchmarks* para realizar o planejamento de capacidade. Além disso, a sistematização na construção do *Plano de Integração*, acrescentando uma visão mais formal a atividade de Entender o Ambiente, faz com que a caracterização da carga de trabalho possa ser derivada facilmente, o que torna todo o resto do processo passível de uma automatização.

Por fim, os trabalhos de modelagem apresentam uma forte utilização de redes de fila (QNs) como formalismo. Nesse ponto, o JMSCapacity está aquém dos demais trabalhos, pois utiliza o formalismo das PNs o que dificulta em demasia a modelagem dos componentes de desempenho. Porém, uma inovação no JMSCapacity é a sistematização da construção do modelo de desempenho apartir do modelo de carga, fazendo desta camada intermediária uma importante aliada para a troca do formalismo utilizado (reimplementando a biblioteca), uma vez que o modelo de carga de trabalho é independente de formalismo.

Um aspecto que fica claro é que não foi encontrado durante este estudo trabalhos relacionados à sistematização do planejamento de capacidade de sistemas de MOM, tornando esse estudo inovador nessa área.

CAPÍTULO

7

Conclusão e Trabalhos Futuros

"Chegue a alguma conclusão, mesmo que seja a conclusão de não ter concluído nada ainda."

Kléber Novartes.

A necessidade crescente de integrar as aplicações dentro de uma organização, associada à importância dos sistemas de MOM nesse processo motivou o desenvolvimento deste trabalho. Este capítulo apresenta as suas principais contribuições e seus aspectos mais relevantes. As limitações dos modelos desenvolvidos e os trabalhos futuros a serem realizados dentro desta linha de pesquisa também são destacados.

7.1 Contribuições

Esta dissertação propõe o JMSCapacity, um *toolkit* formado por um processo, uma biblioteca de componentes e ferramentas para auxiliar no planejamento e gerenciamento de capacidade de MOMs baseados na especificação JMS. O processo adota uma abordagem baseada em modelos permitindo que especificações do serviço de integração sejam mapeadas em redes de Petri estocásticas. Isso viabiliza a realização de múltiplas avaliações de desempenho para diferentes cenários de carga, com o objetivo de determinar a capacidade máxima de um MOM JMS, mantendo os níveis de serviço desejados.

O processo proposto pelo *toolkit* é uma adaptação do trabalho [Menascé et al. 2004]. As intervenções realizadas no processo têm o objetivo de melhorar a sistematização das atividades, permitindo que algumas delas possam ser automatizadas, ou suportadas por ferramentas.

A primeira intervenção aglutina duas atividades do processo original (caracterização da carga de trabalho e obtenção dos parâmetros do modelo de carga) que, em conjunto, produziam um único artefato: *Modelo de Carga de Trabalho.* A nova atividade é denominada *Caracterizar Carga de Trabalho.*

Outra adaptação proposta é a representação explícita da atividade de parametrização do modelo de desempenho, que se encontrava embutida na atividade de desenvolvimento do modelo de desempenho, e agora foi incluída na atividade de validação e calibragem do modelo. A motivação para essa alteração é permitir que o desenvolvimento do modelo de desempenho seja passível de automatização.

A terceira intervenção modifica a característica seqüencial do fluxo de atividades, permitindo que a atividade responsável pela elaboração dos cenários futuros de carga de trabalho possa ser executada em paralelo com outas atividades do processo. A principal motivação para esta modificação é que esta atividade pode ser desempenhada por um profissional diferente dos que executam as atividades que a precedem no processo original.

Com o objetivo de tornar o processo sistematizado, a última intervenção modifica a notação utilizada originalmente, e propõe a criação explícita de papéis e artefatos.

Uma vez redefinido o processo, este trabalho, à medida que o detalha, o instancia para o domínio de um MOM, criando padrões de definições e recomendações que podem ser reutilizados quando da aplicação desse processo no planejamento de capacidade de MOMs baseados na especificação JMS.

Com o objetivo de facilitar o desenvolvimento dos modelos durante a execução do processo, este trabalho propõe uma biblioteca de componentes. Esta biblioteca é dividida em componentes de carga, para auxiliar na construção do *Modelo de Carga de Trabalho*; e em componentes de desempenho, que facilitam a construção do *Modelo de Referência do Sistema*.

Os componentes de carga são representados por uma notação abstrata, proposta por essa dissertação, que se caracteriza pela simplicidade com que o modelo de carga pode ser derivado da especificação do serviço de integração. Para realizar essa especificação, o processo propõe a adoção de diagramas de sequência e implantação, provenientes da notação UML.

Os componentes de desempenho são implementados através de modelos em redes de Petri estocásticas. A concepção desses modelos prevê mecanismos e regras para uma conexão simplificada entre os componentes.

Uma importante característica dessa biblioteca é a relação criada entre os componentes de carga e os componentes de desempenho, permitindo a geração sistematizada do *Modelo de Referência do Sistema* a partir do *Modelo de Carga de Trabalho*.

Para facilitar a adoção do processo proposto, este trabalho identifica, recomenda e adapta um conjunto de ferramentas que fornecem suporte a várias atividades do processo.

Além disso, uma ferramenta para geração artificial de carga é desenvolvida ao longo desta dissertação. A motivação para esse desenvolvimento é fornecer uma

ferramenta totalmente aderente a atividade *Parametrizar, Validar e Calibrar o Modelo*. Ela utiliza a mesma nomenclatura dos componentes da biblioteca para construção da topologia do experimento de medição, facilitando bastante o trabalho do *Avaliador*.

Por fim, este trabalho realiza a validação do processo, através da construção de um guia passo-a-passo, utilizando cenário ilustrativo. Durante a execução do processo, parte substancial da biblioteca de componentes é validada.

7.2 Relevância

A principal contribuição deste trabalho é a construção de um processo para planejamento de capacidade de MOM, utilizando uma abordagem baseada em modelos. A maioria das iniciativas de avaliação de desempenho, envolvendo MOMs, trata basicamente de medições e, em alguns poucos casos, da utilização de modelos analíticos voltados para predições de desempenho.

Outra contribuição importante é a possibilidade da geração sistematizada de modelos de desempenho a partir de especificações de alto nível do serviço de integração. Esse aspecto possibilita diminuir a carga de trabalho na construção dos modelos, além de reduzir o risco de erros de modelagem.

A biblioteca de componentes de desempenho representa um avanço na disponibilização de modelos formais para representação de sistemas *Publish/Subscribe*, uma vez que a maioria dos trabalhos que envolvem modelos para MOM tratam apenas do estilo PTP.

7.3 Limitações

A despeito da validade e relevância deste trabalho, é importante destacar algumas de suas limitações. Alguns aspectos tratados pela especificação JMS como controle transacional, mensagens *Pub/Sub* não duráveis, níveis de prioridade diferenciados e tempo de expiração para as mensagens não estão contemplados pela biblioteca de componentes.

A utilização de redes de Petri, uma representação de alta granularidade, para modelagem dos componentes de desempenho dificulta a construção e manutenção dos modelos de forma manual, exigindo bastante perícia do *Avaliador*.

Outra limitação relacionada às redes de Petri é o fenômeno da explosão do número de estados, que dificulta a avaliação dos modelos através de técnicas de resolução analítica, limitando a resolução através de simulação, consumindo, normalmente, bastante tempo e perdendo precisão nos resultados.

Durante a validação da biblioteca de componentes de desempenho, este trabalho encontra dificuldades para representação precisa da latência, uma vez que a avaliação através de simulação não conseguiu aproximar de forma satisfatória as medidas necessárias para o cálculo dessa importante métrica.

Ainda na atividade de validação, o fato da ferramenta não contemplar geração distribuída de carga (vários clientes distribuídos em máquinas distintas pela rede) dificulta a validação de cenários mais complexos.

7.4 Trabalhos Futuros

Trabalhos futuros devem considerar uma validação mais aprofundada da biblioteca de componentes de desempenho, realizando os ajustes necessários para uma boa representação de todas as métricas necessárias à avaliação de um MOM, especialmente a latência.

Outra importante complementação deste trabalho deve considerar o incremento da biblioteca de componentes com aspectos como prioridade, tempo de expiração e controle transacional.

Para facilitar a adoção do JMSCapacity é necessário o desenvolvimento de uma ferramenta para geração assistida de modelos de carga e sua derivação automática para modelos de desempenho. Além disso, seria interessante complementar a ferramenta JMSMeter no suporte à geração de carga distribuída, permitindo a parametrização e validação de cenários mais complexos.

Referências

[Almeida e Menascé 2002]
Almeida, V. e Menascé, D. (2002) "Capacity Planning: An Essential Tool for Managing Web Services", IT Professional Magazine, IEEE.

[Apache 2006]
Apache Foundation (2006) "ActiveMQ: JMeter Performance Test", disponível em http://incubator.apache.org/activemq/jmeter-performance-tests.html, acesso em 01/07/2008.

[Arteiro et al. 2007]
Arteiro, R., Souza, F., Rosa, N. e Maciel, P. (2007) "Utilizando Redes de Petri para Modelagem de Desempenho de Middleware Orientado a Mensagem", VI Workshop de Desempenho de Sistemas Computacionais e de Comunicação (WPerformance), Rio de Janeiro, Brasil.

[Avizienis 1976]
Avizienis, A. (1976) "Fault-Tolerant Systems", IEEE Transactions Computers, Vol. 25 (12), pp. 1304-1312 .

[Avizienis et al. 2001]
Avizienis, A., Laprie, J. e Randell, C. (2001) "Fundamental Concepts of Dependability", Report LAAS Nº 01145, pp. 1-19.

[BACEN 2007]
Banco Central do Brasil (2007) "Rede do Sistema Financeiro Nacional v6.0", Manual Técnico, Sistema de Pagamento Brasileiro, disponível em http://www.bcb.gov.br/htms/spb/relatoriosRede/SPB-Manual-Tecnico-RSFN.pdf, acesso em 01/08/2009.

[Balakrishnan e Trivedi 1995]
Balakrishnan, M. e Trivedi, K. (1995) "Componentwise Decomposition for an Efficient Reliability Computation of systems with Repairable Components"; Proc. Twenty-fifth International Symposium on Fault-Tolerant computing; Pasadena, CA.

[Baldoni et al. 2003]
Baldoni, R., Contenti, M., Piergiovanni, S. e Virgillito, A. (2003) "Modeling Publish/Subscribe Communication Systems: Towards a Formal Approach",

Proc. Eighth Intl. Workshop on Object-Oriented Real-Time Dependable Systems (WORDS), pp. 304–311, IEEE.

[Baldoni et al. 2005]
Baldoni, R., Beraldi, R., Piergiovanni, S. e Virgillito, A. (2005) “On the modelling of publish/subscribe communication systems”, Concurrency and Computation: Practice and Experience, Vol. 17 (12), pp. 1471–1495, John Wiley & Sons.

[Banavar et al. 1999]
Banavar, G., Chandra, T., Strom, R. e Sturman, D. (1999) “A Case for Message-Oriented Middleware”; Proc. 13th International Symposium on Distributed System, LNCS 1693, pp. 1-18.

[Bernstein 1996]
Bernstein, P. (1996) “Middleware: A Model for Distributed System Services”, Communications of the ACM, Vol.39(2), pp. 87-98.

[Bolch et al. 1998]
Bolch, G., Greiner, S., Meer, H. e Trivedi, K. (1998) “Queueing networks and Markov chains: modeling and performance evaluation with computer science applications”, ISBN 0-471-20058-1, John Willey & Sons,.

[Calabria 2004]
Calabria, E. (2004) “HERMES. Um Middleware Orientado a Mensagem para Ambientes Corporativos”,. Dissertação de Mestrado, Orientado por: Nelson Souto Rosa, Cin/UFPE/Brasil.

[Calzarossa e Serazzi 1993]
Calzarossa, M. e Serazzi, G. (1993) “Workload Characterization: A Survey,” Proc. IEEE, Vol. 81, No. 8, pp. 1136-1150.

[Campbell et al. 1999]
Campbell, A., Coulson, G. e Kounavis, M. (1999) "Managing complexity: middleware explained", IT Professional , Vol.1, No.5, pp.22-28.

[Carter 1979]
Carter, W. (1979) "Basic Concepts in Hardware Architecture for Reliable Computing", 2nd Advanced Course - Computing Systems Reliability, pp.10-21.

[Carter 2005]
Carter, M. (2005) “JMS Performance with WebSphere MQ for Windows V6.0”, Techinal Report v1.1, IBM UK Laboratories, Hursley Park, disponível em http://www-1.ibm.com/support/docview.wss?rs=171&uid=swg24010028, acesso em 01/07/2008, pp.1-32.

[Chappell 2004]
Chappell, D. (2004) “Enterprise Service Bus”. ISBN 0-596-00675-6, O'Reilly.

[Crimson 2003]
Crimson Consulting Group. (2003) "High-Performance JMS Messaging – A Benchmark Comparison of Sun Java System Message Queue and IBM WebSphere MQ", disponível em http://www.sun.com/software/products/messagequeue/wpJMSperformance.pdf, acesso em 01/07/2008.

[Denning e Buzen 1978]
Denning, P. e Buzen, J. (1978) "The Operational Analysis of Queuing Network Models"; Computing Surveys, Vol. 10, No. 3, pp. 225-261.

[Desrochers e Al-Jaar 1995]
Desrochers, A. e Al-Jaar, R. (1995) "Applications of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis"; ISBN 0-87942-295-5; IEEE Press.

[Emmerich 2000]
Emmerich, W. (2000) "Software engineering and middleware: a roadmap", Second International Workshop on Software Engineering and Middleware, Limerick, Ireland, pp. 119-129.

[Eugster et al. 2003]
Eugster, P., Felber, P., Guerraoui, R., e Kermarrec, A. (2003) "The Many Faces of Publish/Subscribe", ACM Computing Surveys, Vol. 35, No. 2, pp. 114-131.

[Fernandes et al. 2004]
Fernandes, S., Silva, W., Silva, M, Rosa, N., Maciel, P., e Sadok, D. (2004) "Performance Analysis of Message-Oriented Middleware Using Stochastic Petri Nets", 22nd Brazilian Symposium on Computer Networks (SBRC 2004), Anais do 22º Simpósio Brasileiro de Redes de Computadores.

[Gartner 2003]
Gartner Inc. (2003) "Predicts 2003: Enterprise Service Buses Emerge (DF-18-7304)", disponível em http://www.gartner.com.

[Gartner 2004]
Gartner Inc. (2004) "Predicts 2004: Enterprise Service Buses Are Taking Off", disponível em http://www.gartner.com.

[Gersting 2001]
Gersting, J. (2001) "*Fundamentos Matemáticos para a Ciência da Computação*"; 4a. Edição, LTC Editora.

[Girault e Valk 2003]
Girault, C. e Valk, R. (2003) "Petri Nets for Systems Engineering: A Guide to Modeling, Verification and Applications", Springer-Verlag, Berlin,Heidelberg.

[Gorton e Liu 2002]
Gorton, I. e Liu, A. (2002) "Streamlining the Acquisition Process for Large-Scale COTS Middleware Components", Proc. First international Conference

on Cots-Based Software Systems", J. C. Dean and A. Gravel, Eds. LNCS, Vol. 2255, Springer-Verlag, London, pp. 122-131.

[Hohpe e Woolf 2004]
Hohpe, G. e Woolf, B. (2004) "Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions", Addison-Wesley/Pearson Education, ISBN 0-321-20068-3.

[IBM 2000a]
Rindos, A., Kittel, A., Woolet, S. e Loeb, M. (2000) "A performance comparison of IBM MQSeries 5.1 and Microsoft Message Queue (MSMQ) 2.0 on Windows 2000: Express delivery mode", IBM Technical Report.

[IBM 2000b]
Rindos, A., Kittel, A., Woolet, S., Loeb, M e Maiolini;R. (2000) "A performance comparison of IBM MQSeries 5.1 and Microsoft Message Queue (MSMQ) 2.0 on Windows 2000: Transactional and persistent delivery mode (including a critique of Microsoft's recent NSTL performance report", IBM Technical Report TR29.3334.

[IBM 2001]
Rindos, A., Kittel, A., Woolet, S. e Loeb, M (2001) "A performance comparison of IBM MQSeries 5.2 and Microsoft Message Queue (MSMQ) 2.0 on Windows 2000: Express and persistent delivery modes", IBM Technical Report, TR29.3410.

[IBM 2005]
IBM Hursley (2005) "Performance Harness for Java Message Service", disponível em http://www.alphaworks.ibm.com/tech/perfharness.

[IDC 2003a]
IDC (2003) "The Enterprise Service Bus: Disruptive Technology for Software Infrastructure Solutions" (Document #29132). http://www.idc.com.

[IDC 2003b]
IDC (2003) "Integration Standards Trends in Program Development: It All Depends on What the Meaning of Open Is" (Document #30365). http://www.idc.com.

[Jain 1991]
Jain, R. (1991) "The Art of Computer Systems Performance Evaluation", Wiley Computer Publishing, ISBN: 0-471-50336-3.

[Jahming 2001]
Jahming Technologies (2001) "High Performance Messaging with JMS: A Benchmark Comparison of SonicMQ 4.0 vs. IBM MQSeries 5.2", Technical Report, Sonic Software, disponível em http://www.sonicsoftware.com.

[JBoss 2005]
JBoss (Red Hat) (2005) "JBoss JMS New Performance Benchmark", White Paper, disponível em http://www.jboss.org/community/docs/DOC-10476, acesso em 01/08/2009.

[Johnson 1989]
Johnson, B. (1989) “Design and Analysis of Fault Tolerant Digital Systems”.

[Kounev 2005]
Kounev S. (2005) “Performance Engineering of Distributed Component-Based Systems - Benchmarking, Modeling and Performance Prediction”. Shaker Verlag, ISBN: 3832247130.

[Kounev 2006]
Kounev, S. (2006) “Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets”, IEEE Transactions on Software Engineering, Vol. 32, No. 7.

[Kounev e Buchmann 2002]
Kounev, S. e Buchmann, A. (2002) “Improving Data Access of J2EE Applications by Exploiting Asynchronous Processing and Caching Services”, Proc. 28th International Conference on Very Large Data Bases - VLDB2002, Hong Kong, China.

[Kounev e Buchmann 2003a]
Kounev, S. e Buchmann, A. (2003) “Performance Modeling and Evaluation of Large-Scale J2EE Applications,” Proc. 29th Int’l Computer Measurement Groupt (CMG) Conf.

[Kounev e Buchmann 2003b]
Kounev, S. e Buchmann, A. (2003) “Performance Modelling of Distributed E-Business Applications Using Queueing Petri Nets”, Proc. 2003 IEEE Int’l Symp. Performance Analysis of Systems and Software.

[Kounev e Buchmann 2006]
Kounev, S. e Buchmann, A (2006) “SimQPN—A Tool and Methodology for Analyzing Queueing Petri Net Models by Means of Simulation”, Performance Evaluation, Vol. 63, Nos. 4-5, pp. 364-394.

[Kounev e Buchmann 2007]
Kounev, S., e Buchmann, A. (2007) “On the Use of Queueing Petri Nets for Modeling and Performance Analysis of Distributed Systems”, Book chapter to appear in Vedran Kordic (ed.) PetriNet, Theory and Application, Advanced Robotic Systems International, Austria, ISBN 978-3-902613-12-7.

[Kounev e Sachs 2008]
Kounev, S. e Sachs, K. (2008) “SPECjms2007: A Novel Benchmark and Performance Analysis Framework for Message-Oriented Middleware”, DEV2DEV Article, O'Reilly Publishing Group.

[Kounev et al. 2008]
Kounev, S., Sachs, K., Bacon, J., e Buchmann, A. (2008) “A Methodology for Performance Modeling of Distributed Event-Based Systems”, Proc.of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC). IEEE Computer Society, Washington, DC, pp.13-22.

[Krissoft 2004]
Krissoft Solutions (2004) “JMS Performance Comparison”, disponível em http://www.fiorano.com/comp-analysis/jmsperfreport.htm.

[Kruchten 1998]
Kruchten, P. (1998) “ The Rational Unified Process – An Introduction”, Addison-Wesley, ISBN 0-201-60459-0.

[Laprie 1985]
Laprie, J. (1985) “Dependable computing and fault tolerance: concepts and terminology”, Digest of FTCS-15, pp. 2-11.

[Lazowska et al. 1984]
Lazowska, E., Zahorjan J., Graham. G. e Sevcik K. C.(1984) “Quantitative System Performance, Computer System Analysis Using Queueing Network Models”, Prentice-Hall.

[Linthicum 2000]
Linthicum, D. (2000) “Enterprise Application Integration”, Addison-Wesley, ISBN 0-201-61583-5.

[Liu et al. 2003]
Liu, T., Behroozi, A. e Kumaran S. (2003) “A Performance Model for a Bussiness Process Integration Middleware”, Proc. IEEE International Conference on E-Commerce (CEC’03), IEEE Computer Society, pp. 191-198.

[Liu et al. 2004a]
Liu, T., Shen, H., and Kumaran, S. (2004) “A Capacity Sizing Tool for a Business Process Integration Middleware”, Proc. IEEE international Conference on E-Commerce Technology (CEC´04), IEEE Computer Society, pp.195-202.

[Liu et al. 2004b]
Liu, Y.; Fekete, A.; and Gorton, I. (204) “Predicting the performance of middleware-based applications at the design level”, Proc. Workshop on Performance and Software Engineering (WOSP), pp. 166-170.

[Liu e Gorton 2003]
Liu, A. e Gorton, I. (2003) “Accelerating COTS Middleware Acquisition: The i-Mate Process”, IEEE Software, IEEE Computer Society, pp. 72-79.

[Liu e Gorton 2005]
Liu, Y. e Gorton, I. (2005) “Performance Prediction of J2EE Applications using Messaging Protocols”, LNCS Component-Based Software Engineering (CBSE 2005), Springer Berlin, Heidelberg, ISBN 978-3-540-25877-3.

[Maciel et al. 1996]
Maciel, P., Lins, R. e Cunha, P. (1996) “Introdução às Redes de Petri e Aplicações”, X Escola de Computação, Campinas, São Paulo.

[Marsan et al. 1984]
Marsan, M., Balbo, G.e Conte, G.(1984) “A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems”, ACM Transactions on Computer Systems, Vol. 2, pp. 93-122.

[Marsan et al. 1995]
Marsan, M., Balbo, G., Conte, G., Donatelli, S. e Franceschinis, G. (1995) “Modelling with Generalised Stochastic Petri Nets”, John Wiley & Sons, ISBN: 0-471-93059-8.

[Menascé 2005]
Menascé, D. (2005) “MOM vs. RPC: Communication Models for Distributed Applications”, IEEE Internet Computing, IEEE Computer Society.

[Menascé et al. 1994]
D. Menasce´, V. Almeida, e L. Dowdy (1994) “Capacity Planning and Performance Modeling—From Mainframes to Client-Server Systems”, Englewood Cliffs, N.J.: Prentice Hall.

[Menascé et al. 1999]
Menasce, D., Almeida, V., Fonseca, R. e Mendes, M. (1999) “A Methodology for Workload Characterization of E-Commerce Sites”, Proc. First ACM Conf. Electronic Commerce, pp. 119-128.

[Menascé et al. 2004]
Menascé, D., Almeida, V. e Dowdy, L. (2004) “Performance by Design: Computer Capacity Planning by Example”, Prentice Hall, ISBN 0-13-090673-5.

[Menascé e Almeida 1998]
Menascé, D. e Almeida, V. (1998) “ Capacity Planning for Web Performance: Metrics, Models and Methods. Upper Saddle River”, N.J.: Prentice Hall.

[Menascé e Almeida 2000]
Menasce´, D. e Almeida, V. (2000) “Scaling for E-Business—Technologies, Models, Performance and Capacity Planning. Upper Saddle River”, N.J.: Prentice Hall.

[Menascé e Almeida 2002]
Menascé, D. e Almeida, V. (2002) “Planejamento de capacidade para serviços na Web: métricas, modelos e métodos”, Campus, ISBN 85-352-1102-0.

[Menascé e Gomaa 2000]
Menasce´, D. e Gomaa, H. (2000) “A Method for Design and Performance Modeling of Client/Server Systems”, IEEE Trans. Software Eng., Vol. 26, No. 11, pp. 1066-1085.

[Mohr e Penansky 1982]
Mohr, J. e Penansky, S. (1982) “A Forecasting Oriented Workload Characterization Methodology”, CMG Trans., Vol. 36.

[Murata 1989]
Murata, T.(1989) "Petri Nets: Properties, Analysis and Applications" Proceedings of the IEEE, Vol. 77, Nº 4.

[Neuts 1975]
Neuts, M. (1975) "Probability distributions of phase type", Liber Amicorum, Professor Emeritus H. Florin, University of Louvain, Belgium.

[NSTL 2000]
NSTL (2000) "Performance Evaluation of Microsoft Messaging Queue, IBM MQSeries and MQBridge", Report for Microsoft Corporation.

[Sachs et al. 2007a]
Sachs, K., Kounev, S., Carter, M. e Buchmann, A. (2007) "Designing a Workload Scenario for Benchmarking Message-Oriented Middleware", SPEC Benchmark Workshop.

[Sachs et al. 2007b]
Sachs, K., Kounev, S., Bacon, J. e Buchmann, A. (2007) "Workload Characterization of the SPECjms2007 Benchmark", Katinka Wolter, editor, "Formal Methods and Stochastic Models for Performance Evaluation", Proc. the 4th European Performance Engineering Workshop (EPEW 2007), Berlin, Germany, LNCS No. 4748, pp. 228-244, Heidelberg, Germany.

[Singhera 2008]
Singhera, Z. (2008) "Modeling Load on a Publish/Subscribe System" Database and Expert Systems Application, DEXA 19th International Conference , pp.706-710.

[Sonic 2003]
Sonic Software Corporation (2003) "JMS Performance Comparison: SonicMQ vs TIBCO Enterprise for JMS", White Paper, Sonic Software, disponível em http://www.onwhitepapers.com/redirect.php?wid=4A0D2BDBBE89205241534CCB0AA8ED56, acesso em 01/08/2009.

[Sonic 2004]
Sonic Software Corporation (2004) "Benchmarking e-Business Messaging Providers", White Paper, Sonic Software, disponível em http://www.sonicsoftware.com, acesso em 15/07/2008.

[Sonic 2007]
Hentchel, D. (20070 "Benchmarking Enterprise Messaging Systems", White Paper, Progress Software Corporation (Sonic Software), disponível em http://www.progress.com , acesso em 15/07/2008.

[Souto et al. 2004]
Souto, E., Guimarães, G., Vasconcelos, G., Vieira, M., Rosa, N., e Ferraz, C. (2004) "A message-oriented middleware for sensor networks". Proc. 2nd Workshop on Middleware For Pervasive and Ad-Hoc Computing (MPAC '04), Vol. 77, ACM, New York, NY, pp. 127-134.

[Souza et al. 2006a]
Souza, F., Arteiro, R., Rosa, N. e Maciel, P. (2006) “Using Stochastic Petri Nets for Performance Modelling of Application Servers”, Proc. 5th International Workshop on Performance Modeling, Evaluation and Optimization of Parallel and Distributed Systems (PMEO-PDS) on 20th IEEE International Parallel & Distributed Processing Symposium (IPDPS), Rhodes Island, Greece.

[Souza et al. 2006b]
Souza, F., Arteiro, R., Rosa, N. e Maciel, P. (2006) “Using Stochastic Petri Nets for Performance Modelling of JBoss Application Server”. V Workshop de Desempenho de Sistemas Computacionais e de Comunicação (WPerformance), Campo Grande, Mato Grosso do Sul.

[Souza et al. 2008]
Souza, F.; Arteiro, R.; Rosa, N.; e Maciel, P. (2008) “Performance Models for the Instance Pooling Mechanism of the JBoss Application Server”. Proc. 27th IEEE International Performance Computing and Communications Conference (IPCCC 2008), pp. 1-10.

[SPEC 2007]
Standard Performance Evaluation Corp. (SPEC) (2007) “SPECjms2007 Benchmark”, disponível em http://www.spec.org/jms2007/, acesso em 01/08/2009.

[Sun 2002]
Sun Microsystems, Inc. (2002) “Java Message Service (JMS) Specification - version 1.1”, disponível em. http://www.sun.com/products/jms/docs.html.

[Tran et al. 2001]
Tran, P., GreenField, P., Gorton, I. e Tran, H. (2001) “Performance Evaluation of Message-Oriented Middleware Technology - Report”, CSIRO Mathematical and Information Sciences, Australia.

[Tran et al. 2002]
Tran, P., Greenfield, P. e Gorton, I. (2002) "Behavior and performance of message-oriented middleware systems," Distributed Computing Systems Workshops, Proc. 22nd International Conference, pp. 645-650.

[Tran et al. 2003]
Tran, P., Gosper, J. e Gorton, I (2003) "Evaluating the sustained performance of COTS-based messaging systems". Proc. First International Workshop on Verification and Validation of Enterprise Information Systems on International Conference on Enterprise Information Systems (ICEIS 2003), John Wiley & Sons, France.

[Tran e Greenfield 2002]
Tran, P.e Greenfield, P. (2002) “Behaviour and performance of message-oriented middleware systems”, Proc. 2nd IEEE International Conference on Distributed Computing Systems Workshops.

[Vinoski 2002]
Vinoski, S.(2002) “Where is Middleware?”, IEEE Internet Computing, pp. 83-85.

[Vinoski 2003]
Vinoski, S. (2003) "The Performance Presumption", IEEE Internet Computing, Vol.07(2), pp. 88-90

[Watson e Desrochers 1991]
Watson III, J. e Desrochers, A. (1991) “Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Nonexponential Transition Functions”, IEEE Transactions on Systems, MAN, and Cybernetics, Vol.21(5), pp. 1008-1017.

[Woodside et al. 2007]
Woodside, M., Franks, G., and Petriu, D. (2007) “The Future of Software Performance Engineering”,2007 Future of Software Engineering. International Conference on Software Engineering, IEEE Computer Society, Washington, DC, pp.171-187.

[Zimmermann 2001]
Zimmermann, A. (2001) “TimeNET: A Software Tool for the Performability Evaluation with Stochastic Petri Nets”, Performance Evaluation Group, TU Berlin.

[Zimmermann et al. 1999]
Zimmermann, A., German, R., Freiheit, J. e Hommel, G. (1999) “TimeNET 3.0 Tool Description”, International Conference on Petri Nets and Performance Models (PNPM'99), Zaragoza, Spain.

APÊNDICE

A

Avaliação de Desempenho

"Nenhum experimento é sempre uma falha completa. Ele pode sempre servir como um exemplo negativo."

Arthur Bloch.

Os usuários, administradores e engenheiros de sistemas computacionais estão sempre interessados em avaliar o desempenho de seus ambientes tendo como objetivo obterem o melhor desempenho do sistema com o menor custo possível. Desta forma entendemos que desempenho é um critério chave na construção, compra ou uso de sistemas computacionais, fazendo com que os esforços dos engenheiros de software e pesquisadores concentrem-se, entre outras coisas, na capacidade de obter máximo desempenho do sistema para um dado custo pré-definido [Vinoski 2003].

Atualmente muitos estudos têm sido desenvolvidos no intuito de sistematizar processos e metodologias de avaliação de desempenho, formando uma área de pesquisa denominada *performance engineering*, que tem objetivos voltados para o *tuning* de sistemas, identificação de gargalos, caracterização de carga do sistema, planejamento e gerenciamento de capacidades (*capacity planning*), além de predição de futuros comprometimentos motivados pelo aumento de carga (*forecasting*).

De acordo com [Jain 1991], contrário a que comumente se espera a avaliação de desempenho é uma arte, e como tal, não pode ser produzida mecanicamente. Toda avaliação requer conhecimento íntimo com o domínio a ser analisado para que se possa definir metodologias, ferramentas e uma massa de teste adequada para efetuar a análise.

Sendo assim, para se obter um processo de avaliação de desempenho que determine com precisão aspectos quantitativos do sistema real, é necessário definir técnicas adequadas para realizar essa tarefa, além de um conjunto de critérios para efetuar a avaliação. Esses critérios são denominados de métricas do sistema.

Este apêndice tem como finalidade apresentar os conceitos básicos sobre avaliação de desempenho, uma vez que planejar e gerenciar a capacidade de sistemas é um dos objetivos dessa área do conhecimento. Inicialmente são detalhadas as técnicas para avaliação de desempenho, assim como um processo para definição da técnica mais adequada. Em seguida, é apresentada uma discussão sobre as métricas (critérios) utilizadas para avaliar os sistemas. Por fim, é apresentado um resumo dos resultados operacionais que relacionam as principais métricas.

A.1 Técnicas para Avaliação de Desempenho

As técnicas para avaliação de desempenho podem ser agrupadas em duas categorias: as baseadas em processo de medição e as baseadas em modelos. A abordagem de medição tem sido largamente utilizada para avaliação de desempenho de sistemas distribuídos, inclusive das plataformas de *middleware*. Essa abordagem possui importantes limitações. Primeiramente, ela é altamente sensível a variações nos parâmetros do ambiente, podendo comprometer a precisão dos resultados. Além do mais, experimentos de medição requerem a construção e configuração de ambientes separados aos de produção, elevando significativamente os custos com equipamentos, ferramentas e principalmente comprometendo um tempo razoável.

A abordagem baseada em modelos pode derivar para uma resolução analítica (matemática) dos modelos, ou ainda na execução de simulações destes. A modelagem analítica é capaz de produzir resultados de desempenho rapidamente, provendo informações valiosas sem ter custos adicionais com a replicação do ambiente real. Um argumento contrário a essa abordagem é o fato que muitas simplificações e suposições do sistema real são necessárias para a construção do modelo, dificultando a obtenção de resultados precisos. Essas simplificações precisam ser feitas devido ao problema da explosão do espaço de estados, que leva a um crescimento exponencial do tamanho do modelo, esgotando rapidamente os recursos computacionais durante sua resolução.

Uma vez que modelos para simulação são construídos para serem executados e não resolvidos, eles podem incorporar mais detalhes (com menos suposições) que os modelos analíticos. Então, os modelos para simulação podem, teoricamente, ser mais flexíveis e mais fiéis ao sistema real, porém possuem algumas limitações. Uma delas faz com que quanto mais complexos eles forem mais tempo é necessário para sua execução atingir um nível de precisão aceitável.

Cada uma das técnicas possui pontos fortes e fracos, sendo usualmente recomendada a utilização de mais de uma técnica pra validar os resultados de desempenho obtidos sobre um sistema. Nas próximas subseções cada uma das técnicas são apresentadas e discutidas.

A.1.1 Medição

As técnicas de medição podem ser classificadas em três áreas específicas: medição propriamente dita, *benchmark* e prototipação. As medições de fato são realizadas em um sistema real sob condições operacionais reais. Entretanto, os resultados medidos podem ser muito particulares, visto que eles são fortemente dependentes das características do sistema medido, levando em consideração a carga imposta, assim como os parâmetros de configuração que ele está submetido naquele momento e situação particular.

Para efetuar as medições, [Menascé e Almeida 2002] define um processo baseado em quatro etapas: especificar medições, especificar pontos de testes, instrumentar e coletar dados, analisar e transformar dados. Inicialmente é necessário determinar o que deve ser medido. Em seguida, determina-se onde esses dados de desempenho devem ser coletados.

A terceira etapa é a coleta propriamente dita dos dados. Em muitos casos, é possível identificar uma quantidade razoável de aplicações utilizando porções significativas de recursos do sistema. Logo, medições de carga devem ser realizadas em um ambiente controlado (ver Figura A-7-1) para que se possa identificar a contribuição de cada aplicação para o desempenho do sistema.

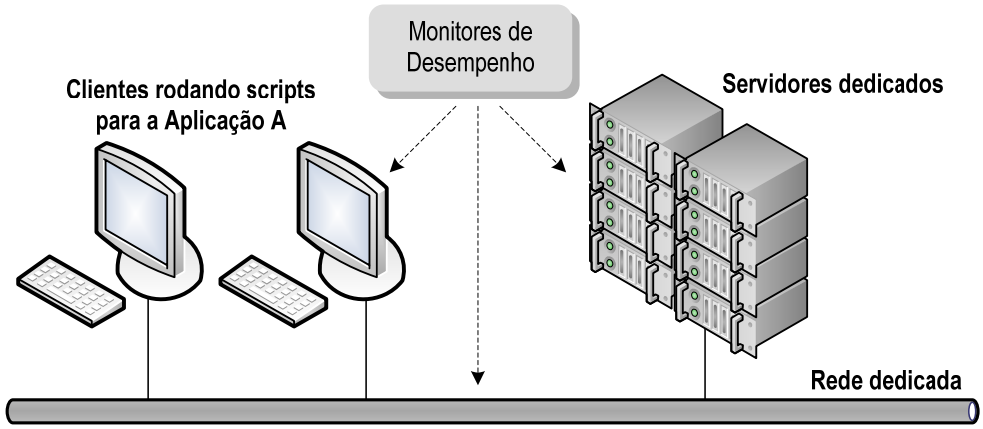


Figura A-7-1. Um típico ambiente de medição controlado.

Esse ambiente deve conter apenas os recursos (computadores, rede, software) envolvidos na coleta de informação, de preferência através de um segmento de rede isolado, e com cada uma das máquinas configurada de forma que apenas o essencial esteja consumindo recursos.

Outro aspecto que precisa ser levado em consideração durante a coleta de dados é quanto ao período de tempo escolhido para efetuar as medições. É bastante recomendável que essa janela de tempo possua um período inicial de estabilização (*ramp-up*), onde nenhum dado é coletado. Após esse período inicial, a expectativa é que os dados sejam coletados em um estado estacionário, não sofrendo interferência quanto a aspectos transientes Ao final é importante descartar um período da janela onde os recursos estão sendo liberados, e o sistema está sendo finalizado (*ramp-down*).

A última etapa para concluir a medição do sistema é a análise e transformação dos dados, com o intuito de estimar o valor dos parâmetros para cada componente básico. Em algumas situações, os valores medidos podem não ser suficientes para valorar os parâmetros, principalmente alguns parâmetros de

demanda de serviço, necessitando que os dados obtidos sejam transformados. Essa transformação normalmente utiliza as relações da análise operacional, como a Lei da Demanda de Serviço, ou a Lei da Utilização (ver Seção A.4).

Em muitas situações, quando dois sistemas precisam ser comparados (particularmente os sistemas computacionais de aplicação genérica) pode não ser suficiente realizar o processo de medição em ambos, considerando uma carga particular, pois muito provavelmente as condições operacionais aos quais estão submetidos são diferentes. Os *benchmarks* são alternativas que mitigam estes aspectos, pois definem um conjunto de cargas "artificiais" que exploram diversas características dos sistemas avaliados e apresentam resultados para as diversas situações particulares (cargas distintas) a qual devem ser submetidos ambos os sistemas, nas mesmas condições operacionais, fazendo com que a comparação possa ser estabelecida.

Uma terceira linha de pesquisa compreende o desenvolvimento de protótipos, ou no contexto de *middleware* a construção de laboratórios de testes e homologação, que visam o desenvolvimento e estudo de versões aproximadas do sistema que se pretende avaliar.

De forma geral, a avaliação das métricas do sistema é calculada a partir das observações do sistema real ou de seu protótipo. Estudos estatísticos das observações permitem inferir através da análise estatística (fundamentalmente a análise de valores centrais e dispersão) os resultados desejados.

A.1.2 Modelagem Analítica

Modelos analíticos são compostos por um conjunto de fórmulas ou algoritmos computacionais que fornecem o valor de uma métrica como função dos parâmetros de entrada do sistema. Como os modelos analíticos precisam ser matematicamente tratáveis, em muitas situações, adota-se um nível de abstração maior do sistema, considerando um menor detalhamento, principalmente se o compararmos com modelos voltados à simulação. Por outro lado, a avaliação baseada em técnicas analíticas, fornece resultados exatos. Quando se fala em exatidão é importante ressaltar que se está referindo ao tratamento matemático – obviamente os resultados obtidos através do modelo serão tão mais próximos dos valores reais quanto mais adequadas tenham sido as decisões de representação (abstração, representação de distribuições, etc) considerados no processo de modelagem.

É importante ressaltar que a utilização de modelos analíticos depende da possibilidade, e muitas vezes da habilidade do modelador/avaliador, de representar o objeto de estudo de maneira que a técnica possa ser aplicada. A aplicabilidade de muitos modelos analíticos fica, portanto, restrita a possibilidade de representar (de maneira adequada) o objeto, conforme uma estrutura conveniente.

Modelos analíticos podem ser classificados como combinatoriais ou baseados em espaço de estados [Balakrishnan e Trivedi 1995] [Bolch et al. 1998]. Alguns exemplos de técnicas de modelagem analítica combinatorial (não baseadas em espaço de estados) para avaliação de confiabilidade são: diagrama de blocos, grafo de confiabilidade e árvore de falhas [Gersting 2001].

Em projetos de avaliação de desempenho, os modelos são de natureza probabilística numa escala de tempo contínua. Para suportar esses modelos é necessário um arcabouço matemático baseado na teoria do processo estocástico, mais especialmente nas cadeias de Markov, que é uma técnica baseada na geração do espaço de estados. A seguir é apresentado um conjunto de definições e conceitos que auxiliará o entendimento deste tipo de técnica [Bolch et al. 1998].

Processo Estocástico. É um modelo matemático utilizado para representação de fenômenos de natureza probabilística como uma função de um parâmetro que usualmente possui significado temporal associado. O processo estocástico é baseado na noção de variáveis aleatórias e pode ser definido como uma família dessas variáveis definidas sobre o mesmo espaço de probabilidade, assumindo valores no mesmo espaço de estados e indexadas por um parâmetro. É normalmente conveniente para avaliação de desempenho a consideração do referido parâmetro como o tempo.

Processo Markoviano. É uma classe especial de processo estocástico que satisfaz as propriedades markovianas, ou seja, o comportamento futuro de um sistema é determinado apenas pelo seu estado atual, e não depende da história passada. A ausência de memória é o principal aspecto que caracteriza um processo markoviano. Em muitos casos esta propriedade não está presente no mundo real, no entanto, este tipo de processo estocástico é o mais utilizado para construção de sistemas de eventos discreto. O motivo para isso é a baixa complexidade computacional da avaliação desse tipo de processo.

Cadeias de Markov. Cadeias de Markov são processos markovianos com espaço de estados discreto. Estas cadeias podem possuir tempo contínuo (CTMC, *Continuous-Time Markov Chain*) ou discreto (DTMC, *Discrete-Time Markov Chain*). Nas CTMCs, o tempo é descrito por uma variável aleatória com função densidade de probabilidade exponencial negativa.

Processo Semi-Markoviano. É uma classe de processos estocásticos, também com espaço de estados discreto, mas que relaxa uma das pré-condições markovianas, onde o comportamento futuro depende apenas do estado atual – não importando a história passada, contudo o tempo de permanência no estado atual é relevante.

O principal problema da utilização das Cadeias de Markov é que para representação de sistemas complexos a tarefa de construção da cadeia torna-se difícil e propensa a erros. Outro ponto relevante quando a complexidade do sistema aumenta é a possível explosão do número de estados do modelo, o que eleva significativamente a complexidade computacional para realizar a sua avaliação precisa, ou até mesmo podendo inviabilizá-la.

Para resolver o problema da geração da cadeia de Markov, pode-se utilizar as redes de Petri [Murata 1989], que é um termo genérico associado a um conjunto de formalismos que possibilitam a representação de estados locais (lugares – que são graficamente representados por círculos) e ocorrência de eventos (transições - representadas por retângulos). Os lugares armazenam marcas que representam recursos ou condições específicas. Dentre os formalismos que compõe a família redes de Petri, as redes de Petri Estocáticas (*Stochastic Petri Nets* ou SPN) e suas extensões têm sido utilizadas como mecanismos de alto

nível para geração automática de CTMC, dado que o grafo de alcançabilidade (*Reachability Graph* ou RG) gerado por uma SPN é equivalente à CTMC [Marsan et al. 1984]. Uma discussão mais aprofundada das redes de Petri é apresentada no Apêndice B dessa dissertação.

Utilizando as redes de Petri, a modelagem dos sistemas é baseada em estados locais, ou seja, foca-se na modelagem dos componentes ao invés de necessariamente ter que observar o sistema como um todo, é possível para o modelador concentrar-se mais no problema a ser solucionado do que suplantar dificuldades e restrições associadas à técnica de representação. Outra vantagem das SPNs é que permitem a análise e verificação de propriedades qualitativas dos modelos.

A extensão *Generalized Stochastic Petri Net* (GSPN) introduz transições imediatas onde o tempo de permanência no estado é nulo. Isso faz com que existam marcações onde a descrição temporal não é capaz de determinar o próximo estado do sistema [Marsan et al. 1984]. Essas marcações no RG são independentes do tempo e ditas marcações *vanishing*. Para determinar que transição irá disparar a partir de uma marcação desse tipo, outros aspectos são necessários (como por exemplo pesos probabilísticos, prioridades). Com isso, o conjunto de alcançabilidade (*Reachability Set* ou RS) gerado pelo RG deixa de possuir um espaço de estado equivalente a uma CTMC. Contudo, a partir deste espaço de estado pode-se "extrair" o grafo de alcançabilidade reduzido (*Reduced Reachability Graph* ou RRG), onde as marcações *vanishing* são retiradas a partir de uma transformação que garante ao RRG um insomofirmo em relação à CTMC [Marsan et al. 1984.

Resumidamente, na modelagem analítica utilizando esse tipo de formalismo o cálculo das métricas do sistema passa necessariamente pela solução analítica das cadeias de Markov. As probabilidades estacionárias e transientes podem ser calculadas a partir da resolução de uma sistema de equações lineares formado por um número de equações igual a quantidade de estados na cadeia de Markov [Bolch et al. 1998], ou seja, a quantidade de estados no RRG da GSPN. A partir desta definição, verifica-se que a aplicação dessa técnica pressupõem a geração de um espaço de estados limitado (finito) e de escala reduzida, senão a solução desse sistema de equações torna-se impossível ou inviável computacionalmente, devido às restrições de memória.

É importante ressaltar que alguns aspectos importantes do comportamento de sistemas num modelo de desempenho não podem ser facilmente resolvidos por modelos markovianos. Estes modelos são chamados de não-markovianos, pois representam eventos (atividades) cujos tempos de ocorrência não são exponencialmente distribuídos. Para resolver esse problema, podem-se utilizar métodos de aproximação por distribuições de fases exponenciais, por meio de varáveis suplementares, ou por q*uantile matching*. Uma boa aproximação das distribuições não-markovianas por distribuições exponenciais pode ser obtida usando equalização de momento (*moment matching*). Nesse case, normalmente, utiliza-se a técnica de aproximação por fase por meio da equalização de momentos de primeira e segunda ordem (média e variância respectivamente).

A.1.3 Simulação

Na fase de projeto ou aquisição de um novo sistema, onde ele ainda não existe ou não está disponível, um modelo de simulação para o sistema facilita a previsão do seu comportamento, podendo ser utilizado para compará-lo com outros sistemas, assim como, propor alterações em seu projeto.

Mesmo nas situações onde um sistema está disponível para realização de medições, um modelo de simulação pode ser interessante pela flexibilidade de se avaliar o sistema sob várias situações distintas através de uma grande diversidade de cenários, o que via medição torna-se extremamente custoso, ou até mesmo impossível dependendo da situação.

Por outro lado, os modelos de simulação freqüentemente não produzem resultados ou produzem resultados imprecisos [Jain 1991]. Isso acontece ou porque os projetistas dos modelos de simulação são proficientes no domínio que querem modelar, mas normalmente carentes de uma formação estatística, ou proficientes no domínio da estatística e carentes no entendimento do domínio do sistema a ser modelado.

Alguns erros comuns na condução de projetos de avaliação de sistemas utilizando a técnica de simulação são:

- Nível de detalhamento inadequado;
- Formalismo utilizado inapropriado para modelagem;
- Modelos não validados ou inválidos;
- Simulações com execuções muito curtas; e/ou
- Mecanismo de geração de números aleatórios fracos ou mal inicializados.

[Jain 1991] define uma terminologia e alguns conceitos utilizados na modelagem de sistemas utilizando a técnica de simulação que se tornam importantes para o perfeito entendimento da área. Para contextualizar a definição dos conceitos foi escolhido, como exemplo, um componente básico de uma infra-estrutura de comunicação, mais especificamente um roteador.

Variáveis de Estados. As variáveis cujos valores definem o estado do sistema. No exemplo, uma das variáveis de estado seria o número de pacotes presentes no buffer de entrada do roteador.

Evento. Uma mudança no estado do sistema é provocada por um evento. No exemplo, a chegada de novos pacotes ou o encaminhamento de pacotes na fila são eventos do sistema.

Modelos com Tempo Contínuo ou com Tempo Discreto. Um sistema onde seu estado é definido para qualquer tempo possível é dito de tempo contínuo. Quando os estados de um sistema só estão definidos para instantes de tempo específicos este é dito de tempo discreto. No exemplo, o estado do roteador é válido para qualquer instante de tempo, logo se trata de um modelo de tempo contínuo.

Modelos com Estados Contínuos ou Estados Discretos. Os modelos são ditos de estado contínuo ou discreto se suas variáveis de estados são contínuas ou discretas, respectivamente. Uma variável é contínua quando ela possui valores não contáveis e não enumeráveis, por exemplo, o domínio dos números reais.

Uma variável é dita discreta se ela possui valores enumeráveis e contáveis mesmo sendo infinito. Como exemplo, pode-se utilizar o domínio dos números inteiros. Um modelo com estados discretos também é denominado de um *modelo de eventos discretos*, similarmente, os modelos com estados contínuos é dito um *modelo de eventos contínuo*.

É importante ressaltar que a continuidade do tempo não implica na continuidade do estado ou vice-versa. Logo, quatro possíveis modelos combinando as duas opções podem ser definidos: tempo discreto com estado discreto, tempo discreto com estado contínuo, tempo contínuo com estado discreto, e tempo contínuo com estado contínuo.

Modelos Probabilísticos ou Determinísticos. Se o resultado gerado por um modelo pode ser previsto de maneira absoluta, esse modelo é dito determinístico, ou seja, o comportamento é determinado, estabelecido para um determinado conjunto de entradas em um dado estado. Em um modelo probabilístico o resultado gerado poderá ser diferente cada vez que esse mesmo evento ocorra (conjunto de entradas) em um mesmo estado do sistema. No exemplo do roteador, a taxa de processamento de pacotes (vazão do sistema) atende a um modelo probabilístico, pois mesmo considerando que várias mensagens de um mesmo tamanho cheguem ao buffer de entrada a uma taxa constante, a vazão do sistema atenderá a uma distribuição probabilística não tendo um valor determinado.

Modelos Dinâmicos ou Estáticos. Um modelo onde o tempo não é uma variável é dito estático. Se o estado de um sistema varia com o tempo, este modelo é chamado dinâmico. O modelo do roteador é dinâmico, pois o seu estado varia com o tempo, ou seja, os valores de suas variáveis de estado se alteram com a variação do tempo.

Modelos Lineares e não-Lineares. Se a saída do sistema pode ser representada por uma função linear dos parâmetros de entrada, o modelo é dito linear, caso contrário é não-linear.

Modelos Estáveis ou Instáveis. Se o comportamento dinâmico de um modelo estabiliza em um estado estacionário, tornando-se independente do tempo, ele é dito estável. Mas se o comportamento do modelo faz com que o estado do sistema esteja em constante modificação, chama-se esse modelo de instável.

As GSPN possibilitam a representação de sistemas através de modelos com estados discretos, tempo contínuo, probabilístico ou ainda determinístico (no caso da inserção de transições temporizadas com tempos determinísticos, extensão DSPN, *Determinisc Stochastic Petri Nets*). Os modelos GSPN além de serem avaliados analiticamente, podem ser executados (simulados), fazendo desse formalismo um mecanismo tanto para modelagem analítica, quanto para simulação.

Dentre os vários tipos de algoritmos de simulação para avaliação de sistemas computacionais, pode-se destacar: simulação orientada a traces e simulação de eventos discretos probabilísticos [Jain 1991].

Simulação orientada por Traces. A característica básica desse tipo de simulação é o fato de que as entradas sob o qual o sistema é testado são *traces* obtidas do sistema real. *Traces* são conjuntos de registros (eventos) ordenados

no tempo que foram submetidos ao sistema real. Este tipo de simulação é normalmente utilizado para analisar novos algoritmos ou novas implementações do sistema ou ainda para realizar experimentos de *tuning* (ajuste fino dos parâmetros do sistema).

Simulação de Eventos Discretos Probabilísticos. Uma simulação que utiliza um modelo de estado discreto é dita ser uma simulação de eventos discretos. É importante ressaltar que o fato de possuir estados discretos não obriga a simulação a operar com tempo discreto, pelo contrário, esse tipo de simulação considera tempo contínuo. De uma forma geral, os simuladores orientados a eventos possuem um relógio da simulação o qual mantém a passagem do tempo do experimento, registrando o tempo e o estado do sistema (conjunto das variáveis de estados) [Lazowska 1984]. O Estado de um sistema normalmente inclui informações a respeito dos estados de todos os seus componentes. A simulação é orientada selecionando o próximo evento (de forma aleatória e normalmente probabilística de acordo com as dependências do modelo) que deverá ocorrer o mais breve possível. Para tanto, o relógio é alterado de em função da ocorrência do evento e a distribuição de probabilidade que representa o tempo associado desse evento. Para a geração das informações temporais em conformidade com as distribuições de probabilidades especificadas, os sistemas computacionais utilizam, como base, geradores de números pseudo-aleatórios. Após alterar o estado do sistema, na ocorrência do evento, o simulador determina os próximos possíveis eventos a serem executados e assim sucessivamente.

Durante a simulação, para cada iteração (a cada novo evento) as métricas a serem avaliadas são recalculadas e armazenadas (registrando-se aspectos de sua variabilidade, assim como o cálculo da variância). Normalmente, os algoritmos de simulação são parametrizados para que possam determinar uma condição de parada, comumente adota-se um nível de confiança e uma taxa máxima aceitável de erro estatístico das métricas. Uma vez que o simulador alcance a condição de parada, encerra o processo, e os valores das métricas são apresentados.

Existem vários algoritmos e técnicas para determinar o momento ideal da parada, entre eles pode-se ressaltar o método que avalia ou estima a variância. No método de avaliação da variância, para cada iteração o simulador calcula a variância das métricas. Se o tempo de simulação é curto a variância normalmente é elevada, enquanto que simulações longas tendem a calcular variâncias menores. A estatística provê suporte para cálculo tanto da variância de eventos independentes quanto de eventos correlacionados.

Para que se atinja a precisão desejada nas simulações, o tempo de simulação é o principal fator restritivo, em função do número de eventos necessários para que possa se alcançar um nível de precisão almejado. Esse efeito ainda torna-se pior quando na presença de eventos raros, ou seja, em situações onde a diferença entre as escalas temporais dos eventos do modelo é muito dispare. Nessas situações, mecanismos de aceleração desses eventos ou simulação transiente podem ser alternativas recomendáveis.

Resumindo, pode-se afirmar que as três técnicas de avaliação apresentadas são úteis em um processo de avaliação de desempenho. Entre as técnicas de

modelagem, o formalismo das GSPN se apresenta como uma alternativa interessante, pois permitem elaboração de um único modelo capaz de ser avaliado através das técnicas analíticas e de simulação. No caso das técnicas analíticas, utilizam-se as GSPN como um mecanismo de geração automático de cadeias de Markov, enquanto que no contexto de simulação se utiliza ferramentas (simuladores de eventos discretos) que utilizam a representação gráfica das GSPN para realizar a avaliação das métricas.

A.2 Processo de Escolha da Técnica de Avaliação

A seção anterior apresentou as técnicas utilizadas para avaliação quantitativa de sistemas. Esta seção detalha o processo de escolha da técnica apropriada e apresenta um conjunto de critérios (ver Tabela A-7-1), mostrados em ordem de prioridade, que balizam a escolha da técnica apropriada.

Tabela A-7-1 – Critérios sobre a Escolha da Técnica de Avaliação

Critério	Modelagem Analítica	Simulação	Medição
Estágio do Ciclo de Vida	Qualquer	Qualquer	Sistema em funcionamento
Tempo Disponível	Pequeno	Médio	Variável
Disponibilidade de Ferramentas	Ferramentas computacionais, modelos matemáticos.	Ferramentas computacionais, modelos de simulação.	Procedimento de Instrumentação
Precisão do Resultado	Moderada	Moderada	Elevada
Nível de Dificuldade	Moderado	Moderado	Elevado
Custo	Moderado	Moderado	Elevado

O critério chave para decidir qual técnica de avaliação deve ser utilizada é o *estágio do ciclo de vida* do sistema a ser avaliado. Se o sistema ainda não existe, ou não está disponível, a técnica de medição é prontamente descartada. Nesse cenário, apenas as técnicas de modelagem (analítica e simulação) podem ser adotadas. Vale também salientar que em sistemas já em estado de funcionamento, pode ser uma tarefa não trivial a criação de cenários complexos e diversos, pois a complexidade operacional para implementação e o processo de instrumentação/medição pode ser altamente sofisticado e caro. Em outras situações, as escalas de tempo podem inviabilizar o processo de medição de um sistema complexo para que se obtenham informações significativas.

Além do *estágio do ciclo de vida*, um segundo critério importante refere-se ao *tempo disponível* para avaliação do sistema. Em muitos casos, necessita-se de resultados com um nível de urgência elevado, sendo a técnica de modelagem analítica a única a responder rapidamente a tais requisitos. Simulações

requerem um tempo de processamento elevado para responderem dentro de níveis de confiança aceitáveis. Medições normalmente levam mais tempo que modelagem analítica, e são mais rápidos que simulações, contudo como mencionado anteriormente, nem sempre é possível ou viável a sua aplicação.

A *disponibilidade de ferramentas* é outro critério utilizado para a seleção da técnica de avaliação. Ferramentas de modelagem permitem, numa mesma plataforma, realizar experimentos tanto baseados em modelos analíticos quanto baseados em algoritmos de simulação. As redes de Petri possuem um grande arsenal de ferramentas que as tornam bastante atraentes como formalismo para modelagem de sistemas. No caso de medições, os dados dependem de softwares específicos, ou ainda de intervenção no sistema.

A *precisão do resultado* da avaliação quando se utiliza técnicas de modelagem (simulação ou modelagem analítica) está intimamente relacionada ao nível de abstração do modelo [Menascé e Almeida 2003]. A Figura A-7-2 apresenta um gráfico que mostra o compromisso entre a precisão dos resultados e o nível de abstração do modelo. Quanto maior precisão o avaliador necessitar introduzir, mais detalhes o modelo terá e maior será o espaço de estados gerados pelas GSPN, levando à escolha a técnica de simulação. Se o projetista pretende utilizar como técnica de avaliação a modelagem analítica (também através de modelos GSPN), porque necessita, por exemplo, de velocidade nos resultados, poderá necessitar elevar o nível de abstração do modelo, deixando de representar detalhes do sistema e com isso poderá reduzir a precisão dos resultados.

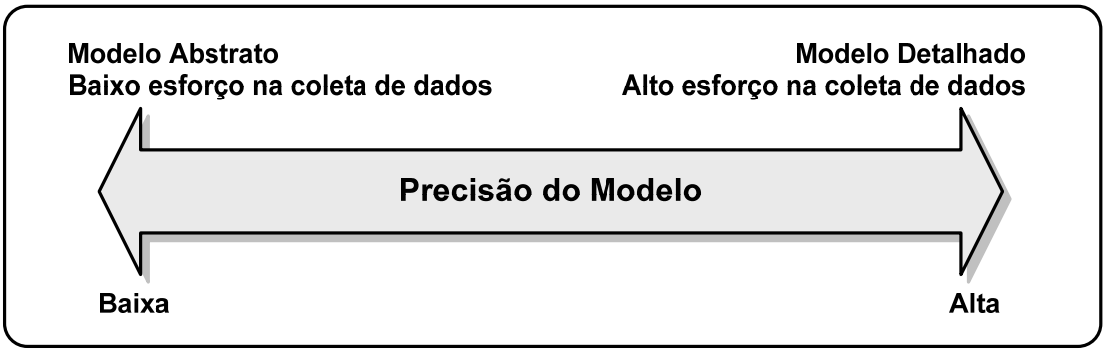


Figura A-7-2. Precisão dos modelos de avaliação

O critério do *nível de dificuldade* torna-se de fácil compreensão devido ao fato de que processos de medição necessitam quase sempre de intervenções no sistema, o que eleva a complexidade da avaliação. No caso de modelagem analítica, o processo torna-se mais simples devido ao suporte matemático inerentes aos modelos. A simulação exige uma cuidadosa parametrização dos experimentos, pois como mostrado na seção anterior, vários erros ocorrem a partir de falhas nesse quesito. Por último, o *custo* não poderia deixar de acompanhar o critério do nível de dificuldade, uma vez que quanto maior a dificuldade maior o custo para implementar a solução.

No contexto de avaliação de desempenho, as técnicas de modelagem são bastante utilizadas merecendo uma discussão mais aprofundada entre a escolha da modelagem analítica ou simulação. Considerando que são utilizadas as GSPN para modelagem do sistema, a escolha da técnica é baseada nas propriedades qualitativas do modelo GSPN. Esta escolha definirá se o avaliador

poderá resolver o modelo através de técnicas analíticas ou se dependerá do simulador provido pelo ambiente de modelagem. É importante ressaltar que o projetista do modelo pode, durante o processo de modelagem, orientar seus modelos para a técnica que deseje, representando, ou não, um nível maior de detalhamento do comportamento do sistema.

Na prática, a escolha da técnica de avaliação dependerá da análise de propriedades qualitativas do modelo GSPN gerado, apresentadas em detalhe no apêndice B dessa dissertação. De forma geral, é preciso observar se a cadeia de Markov gerada automaticamente pelo modelo apresenta propriedades e dimensões que permitam a sua avaliação analítica. [Marsan et al. 1984] mostra que as GSPNs são equivalentes às SPN e, assim possuem uma relação de equivalência com as CTMCs. Além disso, mostra que a complexidade computacional para se obter tais propriedades é menor nas redes GSPNs do que nas SPNs.

Segundo [Desrochers e Al-Jaar 1995], para realizar uma análise das propriedades qualitativas nas GSPNs, pode-se utilizar a rede de Petri básica do modelo considerando todas as transições iguais, assim como se faz para as redes de Petri mais simples e sem tempo (*Place/Transition* PN). Essas propriedades estão intimamente relacionadas à existência de uma distribuição de probabilidade estacionária da sua CTMC equivalente. Além disso, afirma que essa distribuição existe se:

- A rede é limitada;
- A marcação inicial é alcançável a partir de todas as marcações alcançáveis da rede; e
- O tempo de disparo das transições temporizadas é representado por uma variável aleatória exponencialmente distribuída.

Com isso, é preciso mostrar que os modelos gerados são limitados, reversíveis e que possuem todas as transições temporizadas com distribuição exponencial. A última propriedade é verdade (transições temporizadas exponenciais), pois os modelos GSPN por definição atendem esse critério. As duas outras propriedades (limitação e reversibilidade) são discutidas no apêndice B, assim como os métodos de verificá-las. Com essas propriedades atendidas restará definir se o tamanho do espaço de estado não é uma restrição, pois apesar de limitado nada garante que o espaço de estados seja tratável computacionalmente (problema da explosão de estados). Para determinar o tamanho do espaço de estado uma estimativa pode ser realizada baseada no número de lugares e no limite máximo de tokens num lugar qualquer da rede. Mais detalhes podem ser encontrados no apêndice B.

A.3 Métricas para Avaliação de Sistemas

Em cada estudo ou avaliação de um sistema, é necessária a escolha de um conjunto de critérios ou métricas para a avaliação do sistema sob análise (SUT, *System under Test*). Uma forma de identificar essas métricas é relacionar os principais serviços (funcionalidades) providos pelo SUT. Para cada serviço, solicitações são feitas para o sistema, levando a conjunto de possíveis respostas (ou saídas). Essas saídas podem ser classificadas segundo [Jain 1991] em três categorias. O sistema pode realizar o serviço corretamente, incorretamente, ou

não realizar o serviço (recusa de serviço, ou *Deny of Service*, DoS). As métricas associadas a essas saídas são respectivamente classificadas como métricas de desempenho, confiabilidade e disponibilidade.

Cada serviço provido por um sistema possui várias métricas de desempenho, de confiabilidade e de disponibilidade. Como os sistemas oferecem múltiplos serviços é comum que o número de métricas cresça proporcionalmente. A avaliação dessas métricas quase sempre é satisfeita pelo cálculo do valor médio, entretanto não se deve negligenciar o efeito da variabilidade de seus valores. Nesse caso, é recomendável a avaliação tanto do valor médio, quanto da variabilidade de cada uma das métricas.

Quando o sistema executa o serviço corretamente, o seu desempenho pode ser medido pelo tempo tomado para realizar tal serviço, pela taxa (quantidades por unidade de tempo) com que o serviço é realizado, ou ainda pelos recursos consumidos durante a realização do serviço. As métricas de desempenho relacionadas com esses aspectos de tempo, taxa e recurso são normalmente denominadas de métricas de resposta, produtividade e utilização, respectivamente.

Se o sistema realiza o serviço solicitado incorretamente, é dito que um erro ocorreu. Os erros precisam ser divididos em classes, e a probabilidade da ocorrência de cada classe de erro precisa apurada. Se o sistema não realiza o serviço solicitado, é dito que o sistema está fora de operação (*down*), em falha ou indisponível.

As métricas podem representar aspectos individuais ou globais do sistema. As métricas de utilização, confiabilidade e disponibilidade são exemplos de métricas globais, pois avaliam características gerais do sistema. As métricas individuais avaliam cada solicitação realizada individualmente. Algumas métricas, como as de produtividade (vazão do sistema) e de resposta (tempo de resposta) podem avaliar o sistema tanto de forma individual quanto global.

Para avaliar um sistema é fundamental a seleção de um subconjunto das métricas que possuam baixa variabilidade, especificidade (não redundância) e completude. A baixa variabilidade ajuda a reduzir o número de repetições requeridas para se alcançar certo nível de confiança estatística em processos de simulação ou medição. É importante evitar quanto possível métricas que sejam calculadas a partir da divisão de outras métricas, pois o grau de variabilidade do resultado tende a crescer substancialmente.

Se duas métricas representam a mesma informação, elas devem ser simplificadas em uma única para tornar a avaliação do sistema mais simples. Finalmente, o conjunto de métricas escolhidas deve ser completo, refletindo todas as possíveis saídas do sistema.

As próximas subseções apresentam uma discussão mais detalhada de cada uma das categorias de métricas, no contexto de sistemas distribuídos.

A.3.1 Métricas de Resposta

A capacidade de resposta de um sistema pode ser determinada pelo tempo transcorrido entre a solicitação do serviço e a sua efetiva realização. Esse tempo é normalmente denominado de tempo de resposta (*response time*), que é

formado pela composição de atrasos (*delay*) impostos por recursos do sistema responsáveis em realizar tal serviço.

Sistemas distribuídos compartilham seus recursos com múltiplas solicitações de forma simultânea (concorrentemente). Para utilizar cada recurso a solicitação precisa aguardar certo tempo na fila para ter acesso ao recurso, chamado de tempo de espera. Uma vez alocado o recurso, a solicitação passa outro tempo sendo servido pelo recurso, chamado de tempo de serviço. [Menascé e Almeida 2003] definem a somatória de todos os tempos de espera de uma solicitação para um dado recurso como o *tempo de fila*. Eles definem ainda a somatória de todos os tempos de serviço para um recurso específico como a *demanda de serviço*. A partir desses tempos surge o conceito de *tempo de residência* em um recurso como a soma do *tempo de fila* e da *demanda de serviço*.

Dessa forma, o *tempo de resposta* de um serviço pode ser sintetizado pela somatória do *tempo de residência* de cada um dos recursos utilizados pelo sistema para executar esse serviço.

A métrica *tempo de fila* é uma função da carga de trabalho submetida ao sistema, pois quanto mais solicitações concorrentes maior é a disputa pelos recursos, formando assim filas maiores. Por sua vez, a *demanda de serviço* depende fundamentalmente da complexidade do serviço realizado, podendo ser isolado e medido individualmente.

Em sistemas complexos os recursos são interdependentes e cada solicitação de serviço realiza uma composição não trivial desses recursos, utilizando intensamente dependências múltiplas, tornando assim a determinação do *tempo de resposta* uma atividade bastante complexa. Essa é a principal motivação da utilização de modelos probabilíticos para representar tais sistemas, principalmente na representação da carga a ser submetida ao sistema, do comportamento dos recursos e de suas dependências.

A.3.2 Métricas de Produtividade

Um sistema pode ser avaliado quanto a sua produtividade através da valoração da quantidade de solcitiações atendidas por unidade de tempo. Essa métrica é normalmente chamada de *vazão do sistema* ou *rendimento* (*throughput*). A produtividade de uma CPU é avaliada em milhões de instruções por segundo (MIPS). No caso de redes a vazão é medida em *bits* por segundo (bps), enquanto que em sistemas transacionais, transações por minuto (tpm).

O comportamento da vazão do sistema, e a sua relação com o tempo de resposta é mostrado na Figura A-7-3. Inicialmente, com o aumento da carga submetida ao sistema, a tendência é que a *vazão do sistema* se eleve proporcionalmente. Depois de atingido certo nível de carga a vazão passa a crescer numa proporção menor que a carga submetida, levando o sistema a acumular requisições nas filas esperando pelos recursos. Nesse momento o *tempo de resposta* passa a ser degradado, em função do aumento do *tempo de fila*. Essa degradação acontece, pois a capacidade do sistema é menor que a carga de solicitações submetida. Esse ponto de inflexão é denominado de capacidade do joelho. Se a carga continua a ser incrementada o sistema atinge um ponto de saturação, levando o *tempo de resposta* a degradação total.

A capacidade nominal de um sistema é uma medida teórica, e dificilmente é atingida em sistemas reais. Enquanto isso, a capacidade útil do sistema é a maior capacidade atingida antes do ponto de saturação. A relação entre a capacidade útil e a nominal leva a definição de uma nova métrica de produtividade denominada *eficiência do sistema*, expressa em percentual.

Se o objetivo é manter um *tempo de resposta* aceitável para o sistema, provavelmente a carga ideal para atingí-lo é próxima a capacidade do joelho, e um pouco menor que a capacidade útil. Encontrar a carga ideal para manter um *tempo de resposta* específico (nível de serviço) é uma das tarefas do planejamento e gerenciamento da capacidade de um sistema.

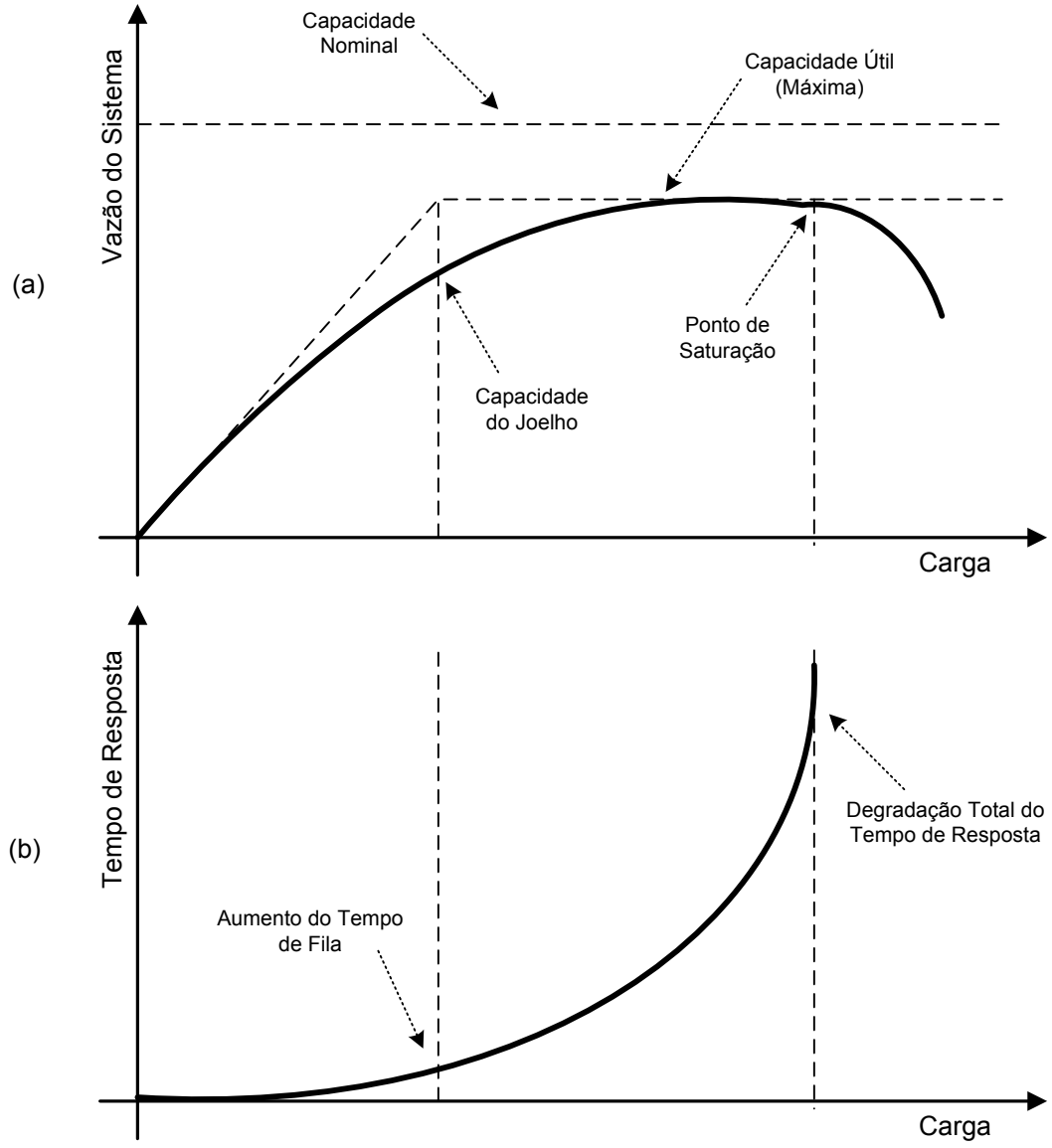


Figura A-7-3. Relação entre as métricas de resposta e produtividade em função da carga submetida ao sistema. (a) Vazão do Sistema (b) Tempo de Resposta.

A Figura A-7-4 apresenta a utilização do mecanismo de controle de admissão para administrar o nível de serviço prestado pelo sistema. Esse controle garante que as solicitações entregues ao sistema para processamento não são superiores a sua capacidade ideal, fazendo com que as demais solicitações sejam rejeitadas

(negação de serviço). Esse recurso é utilizado sempre que é necessário garantir um nível de serviço mínimo na prestação do serviço.

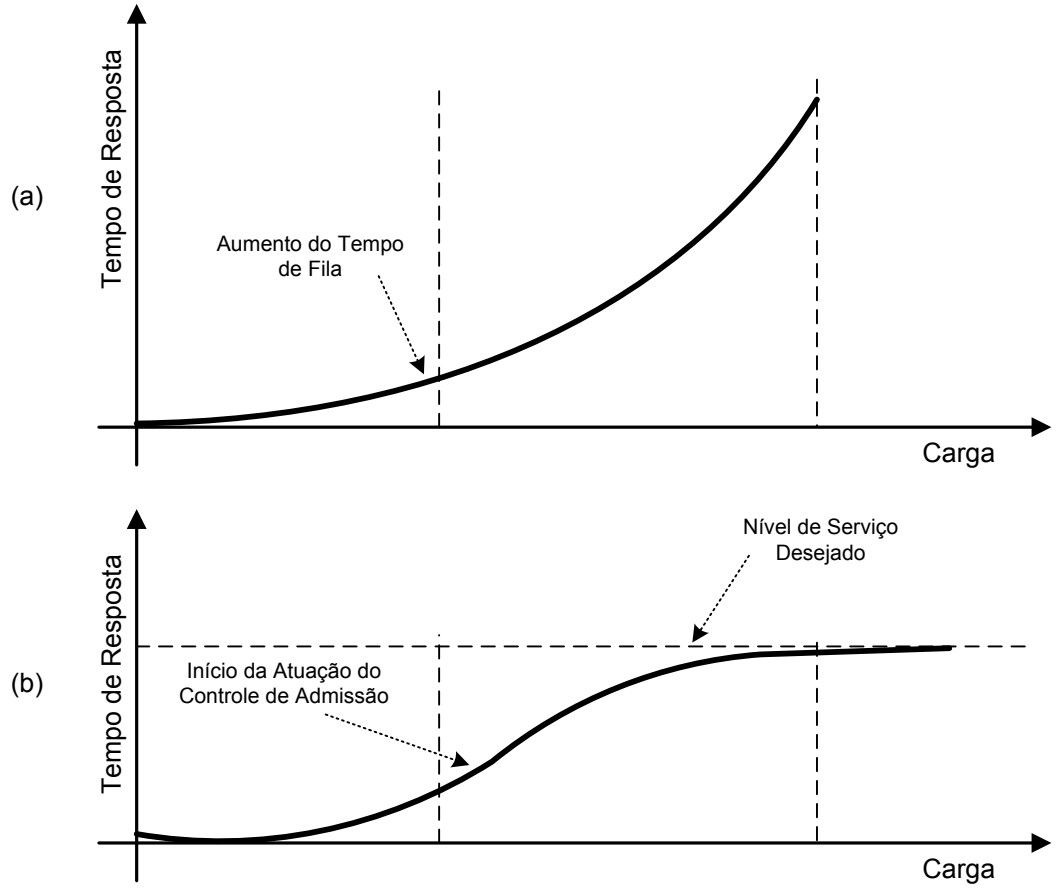


Figura A-7-4. Avaliação do tempo de resposta (a) com aumento da carga sem oontrole (b) e com controle de admissão.

Na Figura A-7-4a o *tempo de resposta* é mostrado sem controle algum, enquanto que na Figura A-7-4b pode-se avaliar o comportamento esperado para essa métrica com a adição do controle de admissão. Neste gráfico a atuação do controle de admissão garante que o tempo de resposta não degrade para níveis não aceitáveis, atendendo, assim, ao nível desejado.

A.3.3 Métricas de Utilização

A utilização de um recurso é medido como uma fração do tempo deste recurso em que ele está servindo ou processando solicitações. Ou seja, é a razão entre o tempo ocupado pelo tempo total em um dado período. O tempo em que o processo não está em uso é chamado de tempo ocioso (*idle time*).

Os gerentes dos sistemas estão freqüentemente interessados em equilibrar a carga entre seus recursos, fazendo com que todos sejam utilizados de forma homogênea. Obviamente nem sempre isso é possível, pois algumas dependências entre os recursos fazem com que essa missão seja muito complexa.

Alguns recursos, assim como processadores, possuem apenas dois estados: ocupado ou ocioso. Mas alguns outros, como as memórias, apenas uma fração do recurso pode ser utilizada em um dado instante de tempo. Nesses casos, a

utilização é medida como a média da fração utilizada sobre o intervalo de tempo.

As métrcias de utilização são definidas em função do sistema a ser avaliado. Em um MOM, algumas métricas de utilização são: tamanho da fila, utilização da fila, utilização da canal de comunicação, entre outros. Todas essas métricas possuem uma característica diferenciada em relação ao abjetivo do gestor do sistema. Enquanto que em outras métricas, quanto maior o valor melhor (ou em outros casos o menor), nas métricas de utilização não se deseja nem valores baixos, pois representam a subutilização do recurso, fazendo perceber que esse poderia ser dispensável, ou que está superdimensionado. Nem muito menos se deseja valores máximos, pois nesses casos o recurso está saturado, levando a uma degradação do nível de serviço prestado.

Segundo [Jain 1991] e [Menascé e Almeida 2003], o ideal é que a apuração do valor médio de utilização dos principais recursos do sistema gire em torno de 50% a 75%. Acima desse patamar o sistema fica vulnerável a picos de utilização dos recursos, podendo ser levado a saturação nesses instantes e comprometer o desempenho global do sistema.

A.3.4 Métricas de Confiabilidade e Disponibilidade

Em circunstâncias específicas, eventos externos naturais ou provocados podem fazer com que o comportamento previsto de um sistema não seja executado. Nestas circunstâncias, o sistema pode não fornecer os serviços para os quais foi especificado.

A habilidade para liberar serviço tal que confiança possa ser justificadamente depositada é denominada Segurança de Funcionamento (*Dependability*) [Avizienis et al. 2001] [Laprie 1985]. Este conceito é a integração de alguns outros conceitos, tais como: confiabilidade, disponibilidade, confidencialidade, integridade, segurança e manutenabilidade.

A confiabilidade de um sistema é uma medida da continuidade do serviço oferecido, a partir de um referencial inicial de tempo [Laprie 1985], ou uma medida da liberação contínua do serviço correto [Avizienis et al. 2001]. Ela usualmente define a probabilidade de ocorrer erros ou o tempo médio entre erros.

A disponibilidade de um sistema é uma medida da liberação de serviço correto com respeito à alternância de serviço correto e incorreto [Avizienis et al. 2001], ou uma medida do serviço realizado, levando-se em consideração a alternância entre os estados "em serviço" e "fora de serviço", isto é, de realizações e interrupções [Laprie 1985]. Ou seja, é a fração de tempo em que o sistema está disponível ou operacional.

Os conceitos de falha, erro e defeito são fundamentais para a área de confiabilidade e disponibilidade. Defeito ocorre quando o serviço executado é diferente do serviço especificado. O defeito deve ser visto como uma violação da especificação e não do projeto [Laprie 1985].

Erro é a parte do estado do sistema capaz de conduzir a um defeito [Laprie 1985], ou então, como aquela parte do estado do sistema a qual está incorreta [Carter 1979]. Diz-se que um sistema está em um estado errôneo, como

conseqüência de uma falha ocorrida, quando o seu estado interno é tal que o leva a apresentar um comportamento diferente do especificado.

Falha é o fenômeno que provoca o surgimento do erro, isto é, a causa direta, física ou algorítmica do erro [Carter 1979]. As falhas estão relacionadas a aspectos de hardware (componentes), de software (projeto), ou ainda operacionais (interação homem-máquina). De acordo com o seu modo de ocorrência, as falhas podem ser classificadas em falhas permanentes, intermitentes ou transitórias [Avizienis 1976].

O parâmetro MTTF (*Mean Time to Failure*) é definido como o tempo esperado até a ocorrência do primeiro defeito. O MTTF é um parâmetro útil para especificar a qualidade de um sistema. Outro parâmetro essencial para avaliação da dependabilidade de um sistema é o MTBF (*Mean Time Between Failure*). Este parâmetro fornece o tempo médio entre falhas. Esta medida inclui o tempo médio para restauração do sistema, conhecido como MTTR (*Mean Time To Repair/Recover/Restore*), outro importante indicador para a determinação da dependabilidade. Tanto o MTBF quanto o MTTR podem ser fornecidos pelos fabricantes dos recursos do sistema, assim como calculados estatisticamente apartir de dados históricos.

A métrcia de *disponibilidade* é formalmente definida em (0.1). Conforme definido anteriormente, a confiabilidade de um sistema depende do período de análise. Se este período tende a infinito, a confiabilidade é igual a zero, monstrando que o defeito é uma certeza futura de qualquer sistema.

$$\begin{aligned} &Disponibilidade = \frac{(MTTF - MTTR)}{MTTF} \\ &\text{onde:} \left\{ \begin{array}{l} MTTF = MTBF - MTTR \\ MTTF \text{ é o Tempo Médio até Falhar } (Mean\,Time\,To\,Failure) \end{array} \right\} \end{aligned} \tag{0.1}$$

Um sistema poderá ter alta disponibilidade, mesmo que venha a apresentar falhas com freqüência, desde que possa ser restaurado com rapidez. Caso as medidas de confiabilidade ou disponibilidade não satisfaçam os requisitos de projeto, pode-se tentar melhorá-las através da aplicação de técnicas de tolerância a falha. Estas técnicas fazem com que o sistema execute corretamente e continuamente os serviços para os quais foi projetado (sem qualquer assistência externa), mesmo na ocorrência de certo conjunto de falhas [Avizienis 1976].

Tolerância a falhas é um termo genérico associado às técnicas que, por redundância, possibilitam o provimento de serviços (conforme a especificação do sistema) mesmo na ocorrência de falhas [Avizienis et al. 2001].

Os mecanimos de redundância podem ser classificados como estáticos e dinâmicos [Laprie 1985]. Os mecanismos de redundância estática mantêm réplicas dos componentes permanentemente conectadas e energizadas. Estas técnicas têm como principal atrativo a ausência de período (tempo) para eliminação de elementos faltosos, sendo utilizadas em sistemas com restrições críticas de tempo. Os elementos redundantes são usados para anular os efeitos das falhas e tornam-se ativos a partir da inicialização do sistema. A ação desta técnica é transparente aos usuários e sua atuação é imediata.

A redundância dinâmica, também conhecida como redundância seletiva, de espera ou reserva, caracteriza-se pela detecção da falha seguida pela ação corretiva ou de recuperação. O uso desta técnica envolve duas etapas distintas: detectação da presença de falhas e eliminação das falhas por meio de ações de recuperação. A execução destas duas etapas exige do sistema um tempo extra de processamento, tempo este que deverá ser previsto no momento da especificação do sistema. Essa técnica ignora a ação humana no processo de recuperação, tornando o sistema auto-reparável.

Outro importante conceito relacionado à confiabilidade é a cobertura contra falhas de um sistema. Essa métrica mede a probabilidade de detecção de uma falha quando esta acontece [Johnson 1989]. Quando essas falhas ocultas ocorrem no sistema de redundância, a confiabilidade do sistema é fortemente afetada.

Na análise das métricas de resposta foi apresentado o mecanismo de controle de admissão para controlar o nível de serviço provido pelo sistema. O efeito negativo desse mecanismo é a diminuição da disponibilidade do sistema motivada pela recusa de serviço para algumas solicitações (ver Figura A-7-5). Essa recusa é necessária para impedir que a capacidade ideal do sistema seja superada, levando a uma degradação do nível de serviço.

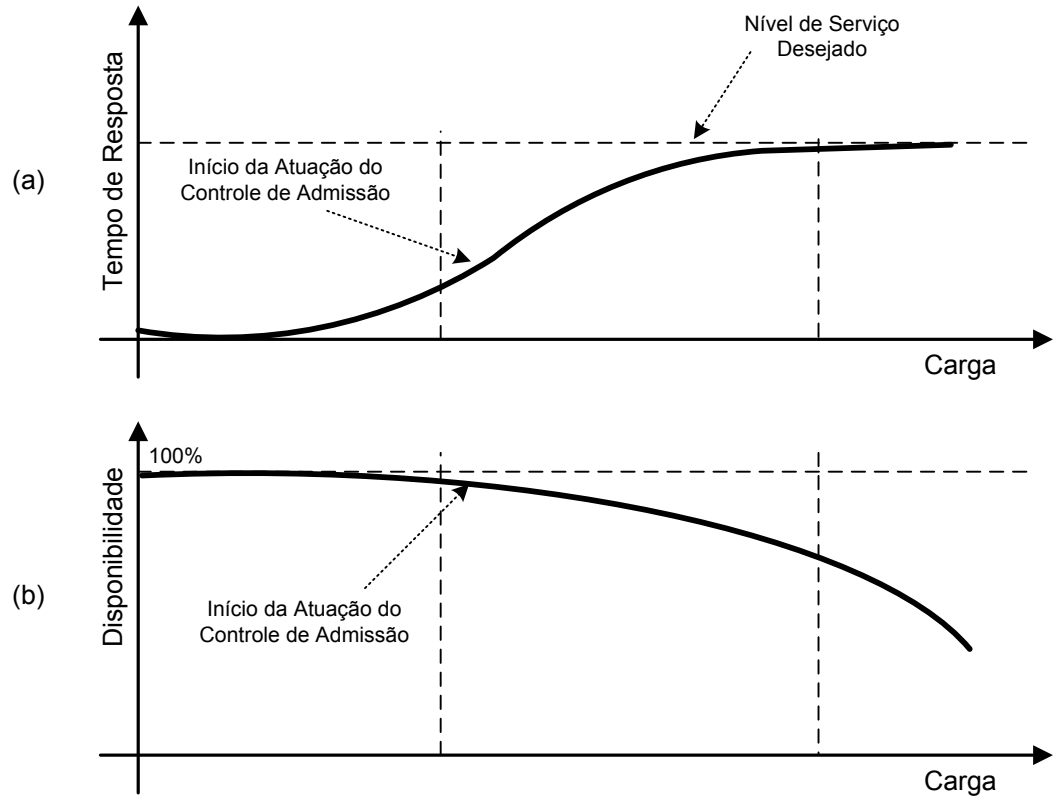


Figura A-7-5. Impacto do controle de admissão na disponibilidade do sistema.

Uma vez que disponibilidade é uma métrica que supõem a possibilidade de falha seguida de recuperação, a avaliação dessa métrica é realizada através de uma análise estacionária. Já a confiabilidade somente faz sentido de ser avaliada através de uma análise transiente, pois um sistema é confiável até que aconteça o primeiro defeito, após isso é dito que o sistema é não confiável. Logo o tempo é fundamental para a avaliação, por isso uma análise transiente.

A.4 Análise Operacional

A análise operacional introduzida por [Denning e Buzen 1978] cria uma relação entre as principais métricas de avaliação estudadas nas seções anteriores. Nem todas as métricas são facilmente obtidas diretamente por medição, ou até mesmo pela avaliação dos modelos. Mas através desses estudos pode-se obter, por exemplo, a *utilização de um recurso*, ou até mesmo o *tempo de residência* a partir de outros valores como a *vazão* e o *tempo de serviço*.

Sendo assim, o principal motivo dessa discussão sobre análise operacional é que eles são matematicamente embasados e ajudam a calcular métricas a partir de outras métricas. As próximas subseções apresentam algumas das Leis Fundamentais da Análise Operacional.

A.4.1 Lei da Utilização

Considere a existência de uma fila disputando o único recurso do sistema. A utilização U do recurso é definida como a fração do tempo em que o recurso está ocupado. Assim, monitorando a fila durante T segundos, obteve-se que o recurso estava ocupado B segundos. Logo, $U = B/T$. Suponha que durante o mesmo intervalo de tempo T, C solicitações foram atendidas pelo recurso. Logo obtem-se que a vazão do sistema, $X = C/T$. Combinando essas duas definições infere-se o relacionamento (0.2).

$$U = B/T = (B/C) \times X = X \times S \quad \text{onde, } S \text{ é igual ao tempo de serviço} \tag{0.2}$$

O *tempo de serviço* quando o sistema está em equilíbrio, ou seja, não encontra-se saturado pode ser expresso pela taxa média de chegada de solicitações na fila (λ). Combinando essa informação com a definição (0.2) obtém (0.3).

$$U = X \times S = \lambda \times S \tag{0.3}$$

A.4.2 Lei de Little

Considere um sistema representado por uma caixa preta (ver Figura A-7-6). Essa caixa pode representar qualquer coisa, desde um dispositivo muito simples, até mesmo o mais complexos dos sistemas distribuídos com todos os seus recursos.

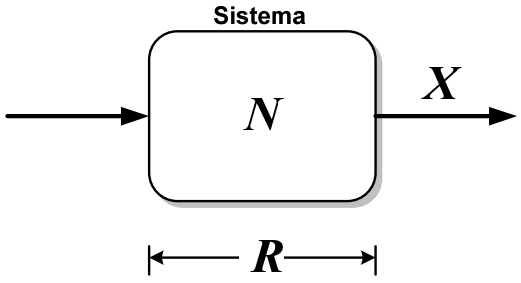


Figura A-7-6. Caixa preta para Lei de Little.

As solicitações para o sistema permanecem dentro da caixa por R segundos, ou seja, o sistema leva R segundos para responder a solicitação. A taxa média de processamento é de X solicitações/segundo e o número médio de solicitações dentro da caixa é N.

A Lei de Little mostra que

$$N = X \times R \tag{0.4}$$

Essa lei (0.4) é a mais importante entre todas definidas por [Denning e Buzen 1978] devido a sua larga aplicabilidade. Ela é bastante poderosa e pode ser aplicada a qualquer caixa preta, desde que não crie nem elimine solicitações que chegam ao sistema.

Com a Lei de Little pode-se obter facilmente o *tempo de residência* de uma solicitação em um recurso, métrica essa de difícil apuração em processos de avaliação. Isso é possível, pois nesses casos normalmente é fácil obter-se o número médio de solicitações (N) durante certo tempo, e a taxa média de processamento (X) nesse intervalo. Dividindo-se o primeiro pelo segundo obtem-se o *tempo de residência* (R).

A.4.3 Lei do Tempo de Resposta

Considere o cenário apresentado pela Figura A-7-7, onde um conjunto de M clientes realiza solicitações independentes para o mesmo sistema já apresentado pela Figura A-7-6, sendo Z o *tempo de pensar* (*think time*) ou *tempo de reação* de cada cliente. Esse tempo representa o intervalo médio entre um cliente receber a resposta de uma solicitação e submter outra ao sistema.

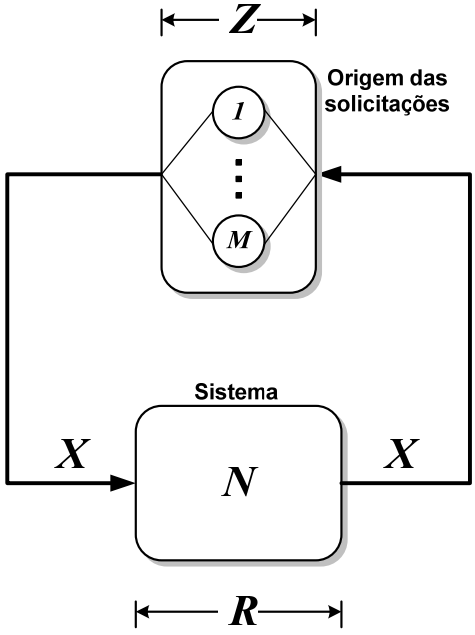


Figura A-7-7. Lei do tempo de resposta.

Nesse cenário o *tempo de residência* é igual ao *tempo de resposta*, pois todo o sistema está sendo considerado dentro da caixa. Utilizando a Lei de Little pode-se mostrar que a Lei do Tempo de Resposta é:

$$R = \frac{M}{X} - Z \tag{0.5}$$

APÊNDICE

B

Redes de Petri

"O trabalho se expande para preencher o tempo disponível para a sua conclusão"

C. Northcote Parkinson.

Atualmente o termo Redes de Petri (*Petri Nets* ou apenas PNs) se refere a um conjunto de formalismos gráficos largamente utilizado como mecanismo de representação de sistemas complexos caracterizados por aspectos de concorrência, sincronização, exclusão mútua e conflitos. Esses aspectos são largamente encontrados em ambientes distribuídos. As PNs incorporam o conceito de ação, de estados e as regras para mudança entre estados permitem representar as características estáticas e dinâmicas de um sistema real.

Este apêndice tem como objetivo fornecer um embasamento teórico profundo sobre as redes de Petri, fornecendo um importante complemento para o entedimento desta disertação, uma vez que todo o processo de modelagem do MOM é realizado utilizando PNs. Para tanto, inicialmente é apresentado uma definição informal sobre PNs, fazendo com que o leitor que não pretende se aprofundar no formalismo possa obter noções básicas. Em seguida, o formalismo das PNs é apresentado em detalhes, juntamente com: propriedades qualitativas e as técnicas utilizadas na sua análise; extensões temporais; além de técnicas para modelagem de atividades não-markovianas.

B.1 Definição Informal

Uma rede de Petri é um tipo particular de grafo dirigido (N) associado a um estado inicial chamado de marcação inicial (M_0). O grafo N de uma rede de Petri possui dois tipos especiais de nós (ou vértices) chamados de lugares e

transições. Os lugares são representados graficamente por círculos e as transições por retângulos. Os arcos do grafo interligam transições a lugares ou lugares a transições, e são rotulados com o seu peso (inteiro positivo). Um arco com peso *k* é equivalente a *k* arcos de peso unitário ligados em paralelo. Usualmente os rótulos unitários dos arcos são omitidos. Uma marcação M_0 (estado) atribui para cada lugar p_i da rede de Petri *N* um número inteiro não negativo *k* representando sua marcação (ou estado local). Graficamente a marcação de um lugar é representada por marcas (pontos pretos) que são chamadas de *tokens*. A Figura B-7-8 apresenta uma rede de Petri com dois lugares (P_{on} e P_{off}) e duas transições (T_{on} e T_{off}). O lugar P_{on} está marcado com um *token* e os arcos que conectam (T_{off} ,P_{off}) e (P_{off} ,T_{on}) possuem peso unitário.

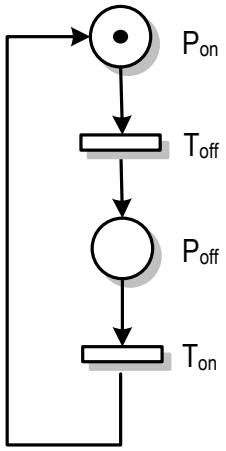


Figura B-7-8. Rede de Petri modelando uma chave inicialmente ligada.

A estrutura de uma rede de Petri é formada pelos seus lugares, transições e arcos. Os lugares representam possíveis estados locais do sistema modelado (condições ou situações), enquanto que as transições são utilizadas para representar eventos que podem modificar o estado do sistema. Os arcos especificam a relação entre os estados locais e os eventos de duas formas: eles podem indicar as pré-condições (conjunto de estados locais) para que ocorra um evento, ou as transformações dos estados locais após a ocorrência de um evento (pós-condições). Uma transição possui um conjunto de lugares como entrada (lugares conectados através de arcos com esta transição), e um conjunto de lugares como saída (lugares conectados através de arcos a partir da transição).

Como já foi dito, os *tokens* são marcas associadas aos lugares, se estes representam uma condição, a presença de tokens representa a condição satisfeita, enquanto a ausência representa o inverso. Se o lugar representa um tipo de objeto (por exemplo, um recurso específico), o número de *tokens* pode representar o número de recursos daquele tipo.

O comportamento de uma rede de Petri é definido pelas condições de habilitação (ou regras de disparo). Uma transição pode disparar se somente todos os lugares de entrada contém no mínimo um número de *tokens* igual ao peso dos arcos que conectam estes lugares a esta transição. Neste caso, a transição é dita habilitada. O disparo da transição (a ocorrência do evento representado pela transição) retira *tokens* dos lugares de entrada e coloca *tokens* nos lugares de saída. Ou seja, o disparo da transição *t* retirara do lugar p_i , que encontra-se conectado por um arco com peso *k* a esta transicão, *k tokens* e

armazena, no lugar p_j, o número de *tokens* representado pelo peso do arco que conecta esta transição (*t*) a este lugar.

A Figura B-7-8 modela uma chave *on-off* que possui dois estados possíveis *Ligado* (P_{on}) e *Desligado* (P_{off}). Os eventos que podem ocorrer neste sistema são *Ligar* (T_{on}) e *Desligar* (T_{off}). Os arcos definem as relações de dependência entre os estados e os eventos, de forma que só pode ocorrer um evento de *Ligar* quando a chave está no estado *Desligado* (lugar P_{off} com uma marca) e de forma equivalente só poderá acontecer o evento *Desligar* quando a chave estiver no estado *Ligado* (lugar P_{on} com uma marca). Uma vez ocorrendo o evento *Desligar*, a chave modifica seu estado *Ligado* (Figura B-7-9a) para *Desligado* (Figura B-7-9b). Ocorrendo o evento *Ligar*, a chave passa para o estado *Ligado* (Figura B-7-9c).

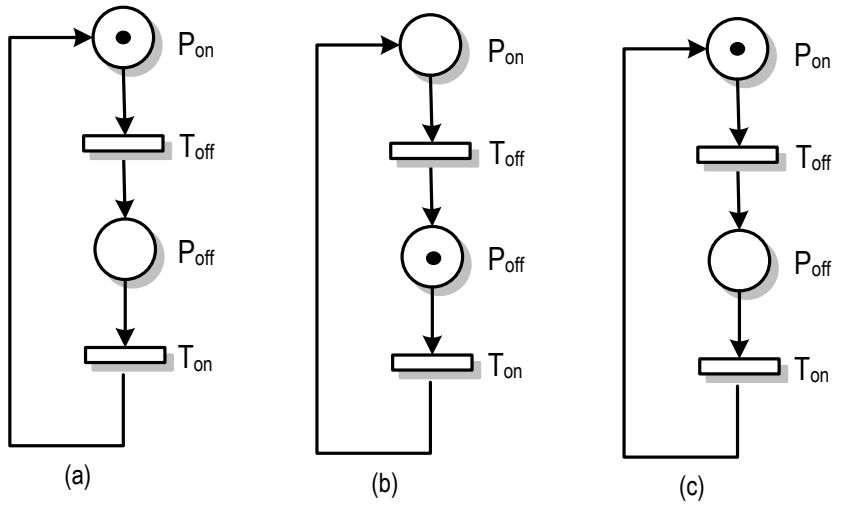


Figura B-7-9. Execução (*token game*) da rede de Petri da chave *on-off* (a) estado incial (b) após disparo de T_{off} (c) após o disparo de T_{on}.

Uma característica importante das redes de Petri é a capacidade de "*execução*" (simulação qualitativa, *token game*) da rede. Essa característica permite observar (e depurar) o comportamento dinâmico, através da simulação (ver Figura B-7-9). Entretanto, além de proporcionar uma linguagem gráfica e permitir execução da rede, as redes de Petri também possuem um forte embasamento matemático que permite realizar diversos tipos de análises qualitativas e quantitativas. Para demonstrar esta capacidade será necessário uma abordagem formal para as redes de Petri, que será realizada na próxima seção.

B.2 Formalização da Linguagem

A rede de Petri *Place/Transition* [Murata 1989] é uma linguagem de especificação formal para sistemas concorrentes. Esta seção apresenta algumas definições importantes relativas ao formalismo das redes de Petri possíveis de serem validadas e analisadas [Girauld e Vaulk 2003] [Marsan et al. 1995] [Maciel et al. 1996] [Murata 1989].

Definição 1. Uma rede de Petri (PN) *S* é uma tupla

$$S = (P, T, I, O, M_0)$$

Onde:

P é o conjutno de lugares;

T é um conjunto de transicões, $T \cap P = \varnothing$;

$I, O : T \rightarrow Bag(P)$, são as funções de *input* e *output* respectivamente, onde *Bag(P)* é um conjunto de multiconjuntos em P; Em notação matricial, pode-se definir a matrizes I e O como um mapeamento no naturais, ou seja: $I, O : \mathbf{T} \times \mathbf{P} \rightarrow \square$;

$M_0 : P \rightarrow \square$ é a função marcação inicial, que associa para cada lugar de P um número natural. Em notação vetorial $\mathbf{M_0} = [m_i], m_i \in \square \mid m_i = M_0(p_i), \forall p_i \in P$.

Conforme a Definição 1, a rede de Petri apresentada na Figura B-7-10 pode ser definida com P sendo o conjunto {P1, P2, P3}; T o conjunto {T1}; I a função com valores (T1,(P1,1)), (T1,(P2,2)); O representado por (T1,(P3,1)); e M_0 sendo a função com valores (1,2,0).

Definição 2. (Habilitação) A transição t está habilitada em uma marcação M se e somente se

$$\forall p \in I(t), M(p) \geq I(t, p)$$

Definição 3. (Disparo) O disparo da transição t, habilitada em uma marcação M, produz uma marcação M' tal que

$$M' = M + O(t) - I(t)$$

sendo válido afirmar que M' é diretamente alcançável a partir de M, ou sinteticamente M [t> M'.

A Figura B-7-10 apresenta a execução de uma rede de Petri com três lugares (P1, P2 e P3) e uma transição (T1) habilitada para M_0. Após o disparo de T1, ela retira os *tokens* dos lugares de entrada (P1 e P2) e coloca-os no lugar de saída. O número de *tokens* retirado e colocado é igual ao peso de cada arco. Logo a marcação M' (Figura B-7-10b) pode ser obtida conforme a Definição 3.

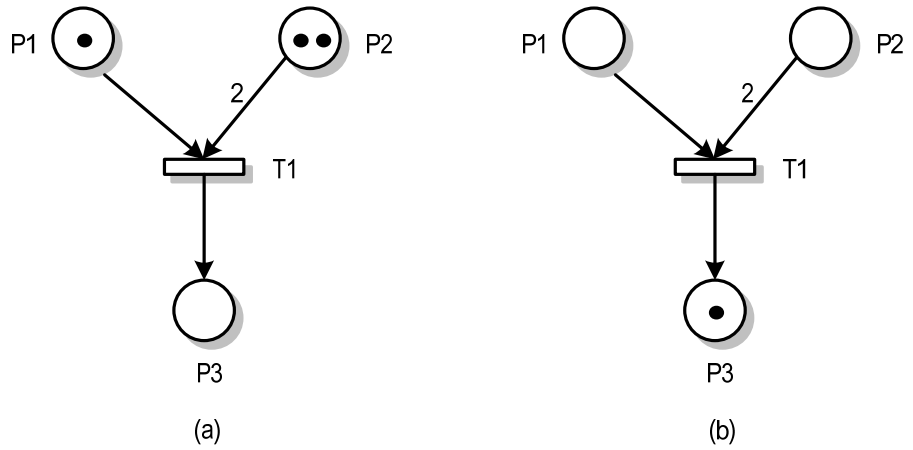


Figura B-7-10. Execução de um rede de Petri representando uma sincronização (a) estado inicial (b) após disparo de T1.

Definição 4. (Conjunto de Alcançabilidade) O Conjunto de Alcançabilidade de uma rede de Petri *N* com marcação inicial M_0 é descrito por $RS(M_0)$, e é definido como o menor conjunto de marcações tal que

- $M_0 \in RS(M_0)$
- $M_1 \in RS(M_0) \wedge \exists t \in T : M_1[t > M_2 \in RS(M_0)$

Definição 5. (Grafo de Alcançabilidade) Seja *N* uma rede de Petri, M_0 sua marcação inicial, e $RS(M_0)$ o conjunto de alcançabilidade da rede *N* na marcação inicial M_0 , o Grafo de Alcançabilidade $RG(M_0)$ é um grafo rotulado dirigido onde o conjunto de nós é *RS* e o conjunto de arcos *A* é definido como segue:

- $A \subseteq RS \times RS \times T$
- $\langle M_i, M_j, t \rangle \in A \Leftrightarrow M_i [t \rangle M_j$

e M_0 é configurado como o nó inicial do grafo.

O conjunto de alcançabilidade RS={(1,2,0),(0,0,1)} e o grafo de alcançabilidade da rede Petri apresentada na Figura B-7-10 é mostrado na Figura B-7-11. O grafo de alcançabilidade é de fundamental importância, pois representa o espaço de estados do sistema. Desta forma, várias propriedades serão verificadas através da verificação de propriedades deste grafo, assim como limitação e ausência de *deadlock*.

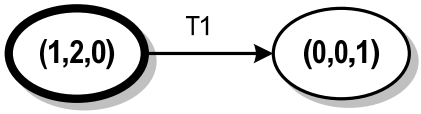


Figura B-7-11. Grafo de alcançabilidade da rede de Petri da Figura B-7-10.

As redes de Petri se configuram como uma boa solução para representar sistemas reais com aspectos de concorrências e conflitos, conforme já foi dito anteriormente. Estes mecanismos são parte natural da linguagem. A Figura B-7-12(a) apresenta as transições T1 e T2 habilitadas ao mesmo tempo tendo como pré-condição um lugar comun, a esta situação denominamos de conflito. Nesta situação uma escolha não determinística é a efetuada (*free-choice*).

Definição 6. (Grau de Habilitação) Para cada rede de Petri, o grau de habilitação é uma função $ED : T \times [P \rightarrow \square] \rightarrow \square$ tal que $\forall t \in T, \forall M : P \rightarrow \square, ED(t, M) = k$ se e somente se

$$\forall p \in I(t), \; M(p) \geq k \cdot I(t, p)$$

Na Figura B-7-12a o grau de habilitação de T1 é igual a um, já na Figura B-7-12c o grau de habilitação de T1 é igual a 2. Ou seja, o grau de habilitação representa a maior quantidade de disparos consecutivos que uma transição pode realizar numa dada marcação.

O conflito pode ser definido como simétrico ou assimétrico. No caso da Figura B-7-12a o conflito é simétrico, pois o disparo de qualquer uma das duas transições conflitantes diminui o grau de habilitação delas igualmente. No caso da Figura B-7-12b e Figura B-7-12c pode-se notar que o disparo de uma das transições poderá diminuir de forma não simétrica o grau de habilitação das transições.

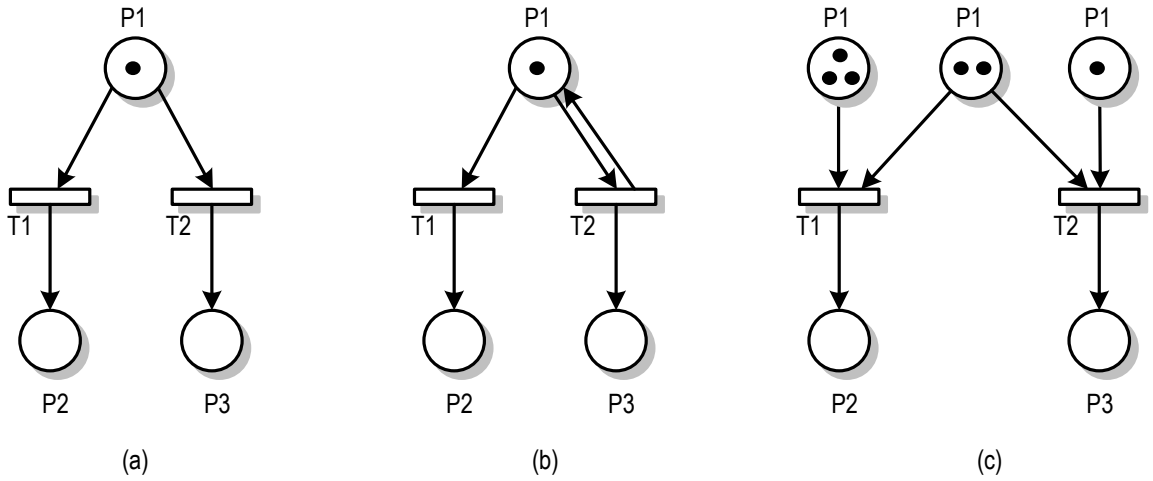


Figura B-7-12. Exemplos de conflitos.

Definição 7. (Conflito Efetivo) Para qualquer rede de Petri, $\forall t_l, t_m \in T$ tal que $t_l \neq t_m$, $\forall M : P \rightarrow \square$, a transição t_l está em conflito efetivo com t_m na marcação M (denotado t_l *EC(M)* t_m) se somente se

$$M\left[t_l\right\rangle M' \text{ e } ED(t_m, M) < ED(t_m, M')$$

Contrariamente ao conflito, a concorrência é caracterizada pelo paralelismo das atividades. Então, dizemos que duas transições quaisquer são concorrentes quando estão habilitadas simultaneamente numa dada marcação e não são conflitantes.

Definição 8. (Concorrência) Para qualquer rede de Petri, as transições t_l e t_m são ditas concorrentes numa marcação M se e somente se

$$t_m, t_l \in E(M) \;\Rightarrow\; not(t_l EC(M) t_m) \text{ and } not(t_m EC(M) t_l)$$

onde *E(M)* é o multiconjunto de todas as transições habilitadas na marcação *M*. A Figura B-7-13 apresenta um exemplo de concorrência entre as transições T1 e T2. Neste exemplo nenhuma das duas é conflitante com a outra, e ambas estão habilitadas na marcação mostrada na figura.

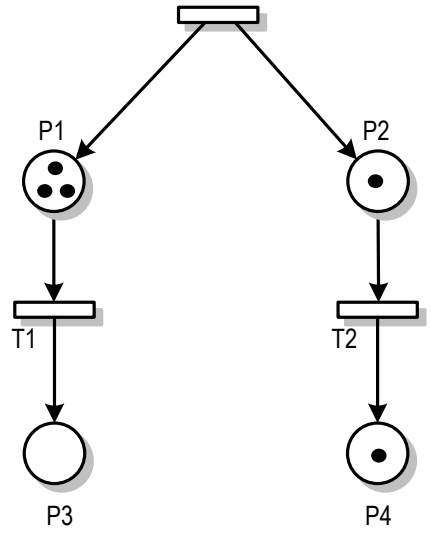


Figura B-7-13. Exemplo de concorrência.

Uma situação indesejada pode ser configurada se existirem simultaneamente conflito efetivo e concorrência no modelo. A Figura B-7-14 apresenta dois exemplos desta situação, chamada de *Confusão*. Na figura podemos ver claramente que dependendo da ordem de disparo as transições podem ser simplesmente concorrentes ou serem conflitantes. Mas o conflito efetivo nem sempre se configura na marcação apresentada e sim numa marcação posterior. Este tipo de situação pode provocar um comportamento do modelo diferente do projetado. Para resolver este tipo de situação ou se modifica o modelo, ou precisa-se determinar em tempo de projeto qual das duas transições deve disparar primeiro. Surge então a necessidade da definição de prioridades entre as transições. Por fim, outra importante característica que pode ser modelada em redes de Petri é a exclusão mútua. Ou seja, quando uma está habilitada a outra não poderá estar. Estas carcatrísticas serão mais bem descritas nas subseções seguintes.

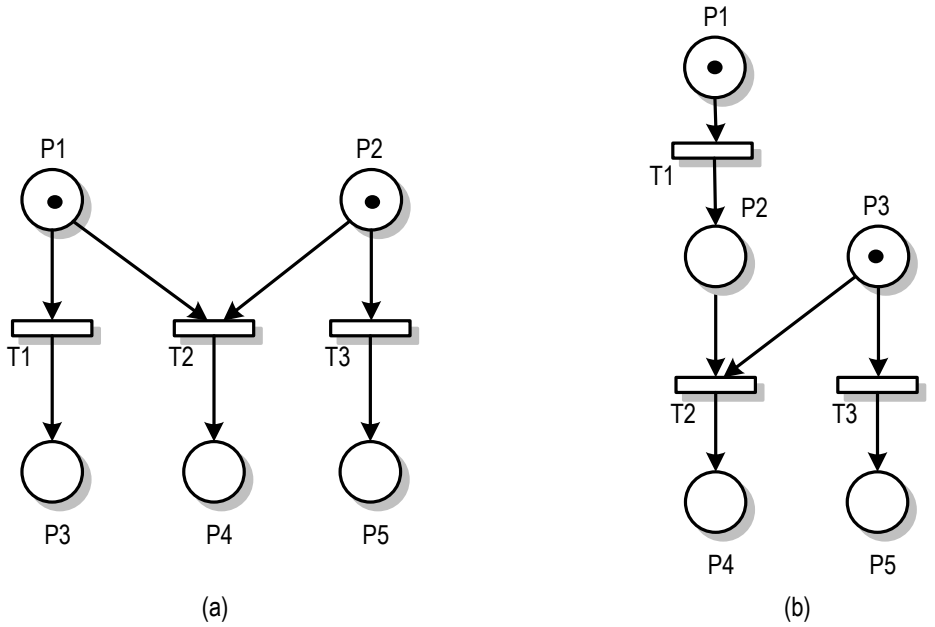


Figura B-7-14. Exemplos de confusão.

Algumas extensões da linguagem inicial foram propostas para resolver situações de comportamento indesejado do modelo, ou para aumentar a capacidade e poder de representação. As prioridades são exemplos da primeira situação, enquanto os arcos inibidores e as funções de guarda representam exemplos de melhoria na capacidade de expressão da linguagem.

Definição 9. (Prioridades) Para uma rede de Petri qualquer a função prioridade *Π* associa para cada transição *t* um número inteiro não negativo, $\Pi : T \rightarrow \square$. Desta forma, estarão efetivamente habilitadas a disparar as transições habilitadas segundo a Definição 2 e que tiverem o menor valor de *Π*.

Esta definição afeta quase todas as demais definições, podendo restringir o grau de habilitação, conflito, concorrência e até mesmo resolvendo possíveis confusões.

Definição 10. (Arco Inibidor) Para uma rede de Petri qualquer onde P é o conjunto dos lugares, a função de inibição *H* associa para uma transição $t \in T$ um *Bag(P)*, onde *Bag(P)* é um multiconjunto de *P*

$$H : T \rightarrow Bag(P)$$

Da mesma forma que a função prioridade Π, a função de inibição H afeta algumas definições como disparo, habilitação e grau de habilitação. Esta função tem importância vital quando precisamos modelar o teste a uma pré-condição zero (lugar de entrada sem marcas). Quando este teste precisa ser realizado sob um lugar ilimitado, é necessária a adição dos arcos inibidores.

A regra de habilitação determina que se existirem arcos inibidores como pré-condição para uma transição, esta só estará habilitada se os lugares de origem destes arcos tiverem o número de marcas menor que o peso do arco, ou seja, este arco funciona como o complemento da habilitação padrão. A Figura B-7-15 apresenta um exemplo da utilização deste (representado graficamente com o círculo vazio na ponta em vez da seta tradicional). Neste exemplo, o lugar P2 funciona como um *flag* que garante que a transição T1 só dispara novamente após o disparo da transição T2. Ou seja, supondo que P1 represente um *buffer* de peças do tipo *A* numa linha de montagem, e P3 represente um *buffer* de peças do tipo *B*. Suponha que P2 representa o esforço para montar a peça *A* e a peça *B* formando um conjunto *C* através do disparo da transição T3. Suponha que cada conjunto *C* tem que obrigatoriamente ser formado por uma peça *A* e uma uma peça *B*. O arco inibidor está forçando que a transição T2 dispare depois de T1, obrigatoriamente.

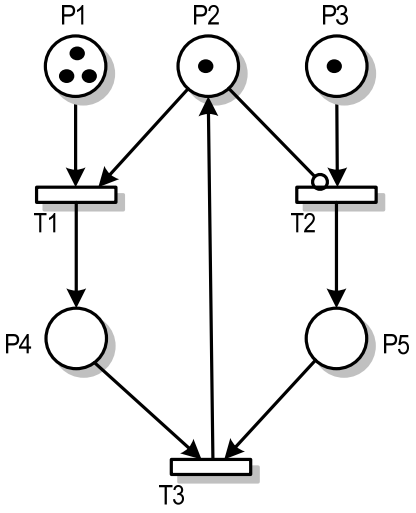


Figura B-7-15. Exemplo de arco inibidor.

Um outro recurso que aumenta a capacidade de expressão das redes de Petri é a inserção de funções de guarda. Assim como arcos inibidores, as funções de guarda modificarão a regra de habilitação das transições, sendo esta função de guarda representada por uma expressão booleana dependente ou não da marcação atual. Estas duas extensões tornam os modelos mais reduzidos e mais simples de entedimento, apesar de incrementarem complexidade ao processo de análise qualitativa e quantitativa da rede.

Definição 11. (Função de Habilitação ou Guarda) Para uma rede de Petri qualquer, a função de guarda *G* associa para uma transição *t* uma expressão booleana que pode ser avaliada em verdadeiro ou falso. Uma vez que esta expressão seja avaliada como falsa, a transição *t* não estará habilitada. Se a

avaliação for verdadeira, as demais condições já apresentadas para definir a habilitação de uma transição determinarão se *t* estará ou não habilitada.

$$\forall t \in T, g_t : \square^{|P|} \rightarrow \{Verdadeiro, Falso\}$$

B.3 Propriedades Qualitativas

As redes de Petri possuem propriedades qualitativas estruturais e comportamentais. As estruturais podem ser analisadas puramente pela avaliação da sua estrutura, enquanto que as comportamentais só podem ser analisadas uma vez que adiciona-se uma marcação inicial ao modelo. Todas as propriedades válidas através da análise estrutural são válidas para qualquer marcação, enquanto que as propriedades comportamentais são válidas apenas para a marcação inicial a qual foi verificada.

Na análise comportamental é preciso construir o grafo de alcançabilidade (ou o espaço de estados) para a rede ser analisada, considerando que a quantidade de estados seja finita. Uma vez construído este grafo, as propriedades comportamentais podem ser verificadas percorrendo-o (a teoria dos grafos fundamenta os algoritmos de verificação).

Os mecanismos de análise comportamental normalmente são mais conclusivos. Porém, mesmo para as redes limitadas, a complexidade para geração do espaço de estado cresce exponencialmente com o tamanho da rede. Considerando que problemas reais normalmente serão representados por redes complexas, pode-se inferir que a geração do espaço de estados nem sempre será viável computacionalmente. Logo, soluções de análise que não necessitem construir o grafo de alcançabilidade passam a ser fundamentais. Além disso, quando as redes são ilimitadas o espaço de estados será infinito.

Uma alternativa de abordagem para o problema de explosão de estados é utilização da árvore de cobertura, que proverá informações valiosas sobre o comportamento das redes, mesmo compondo parcialmente os estados alcancáveis. Esta abordagem tem limitação quanto a quantidade de propriedades que podem ser verificadas.

A seguir serão apresentadas algumas propriedades qualitativas que podem ser verificadas nas redes de Petri e técnicas para Análise tanto estrutural quanto baseada no espaço de estados (comportamentais).

B.3.1 Reversibilidade e *Home State*

Uma rede de Petri *S*, com marcação inicial M_0 é dita ser reversível se e somente se de qualquer estado alcançável a partir de M_0 é possível retornar para M_0

$$\forall M \in RS(M_0), M_0 \in RS(M)$$

no entanto, é possível que o estado para o qual a rede é reversível não seja a marcação inicial. Logo, define-se que $M \in RS(M_0)$ é um *home state* para a rede *S* se e somente se

$$\forall M' \in RS(M_0), M \in RS(M')$$

B.3.2 Ausência de *Deadlock*

Uma rede de Petri contém *deadlock* se numa dada marcação *M,* alcançável a partir de M_0, nenhuma transição estiver habilitada. A condição para ausência de *deadlock* é obtida de forma complementar.

Não existe nenhuma definição formal para ausência de *deadlock*, o que pode-se garantir é que existe potencialmente condições para se atingir um estado de *deadlock*. Na Seção B.3.7 serão apresentadas técnicas que permitem determinar com precisão a ausênciade *deadlock* a partir de uma análise estrutural.

B.3.3 Liveness

Uma transição é dita viva (*live*) numa rede de Petri se e somente se para cada marcação *M* alcançável a partir de M_0, existe uma marcação *M'*, alcançável a partir de *M*, tal que $t \in E(M')$, onde *E(M)* é o multiconjunto de todas as transições habilitadas na marcação *M*. Uma rede de Petri é chamada de viva (*live*) se todas as transições da rede forem vivas.

Se um modelo possui a propriedade de *liveness*, ele não terá bloqueio (*deadlock*), pois todas as ações do modelo devem ser potencialmente realizáveis a partir de qualquer estado do modelo. Desta forma, todo modelo *live* não possui *deadlock*, contudo nem todo modelo sem *deadlock* é *live*. Esta propriedade possui uma complexidade computacional elevada para ser calculada, e nem sempre é possível verificá-la. Além disso, pode ser que o modelo precise ter transições mortas para representar o sistema real e nem por isso ele entrará necessariamente em estado de *deadlock*. Na verdade, é condição suficiente para a ausência de *deadlock* a existência de pelo menos uma transição viva para qualquer marcação atingível a partir da marcação inicial (M_0).

B.3.4 Limitação

Um lugar *p* de uma rede de Petri é dito ser limitado a *k* se e somente se para cada marcação alcançável M o número de *tokens* no lugar *p* é menor ou igual a *k*. Uma rede de Petri é considerada limitada a *k* se e somente se todos os lugares da rede forem limitados a *k*.

Quando um modelo é limitado, o espaço de estados gerado é finito com um número de estados não superior a $(k+1)^N$, onde N é o número de lugares na rede limitados a *k*. Esta propriedade permite que estes modelos sejam analisados de maneira precisa, pois (potencialmente) é possível construir o espaço de estados e aplicar o conjunto de técnicas associadas à análise do espaço de estado. A limitação garante a realização de avaliações precisas das métricas. É importante observar, no entanto, que algumas técnicas possibilitam a representação finita de um número infinito de estados. Neste caso, o formalismo pode permitir a representação desta característica, mas a análise formal baseada no espaço de estados fica comprometida.

B.3.5 Segurança

Uma rede é dita ser segura (*safe*), se e somente se é limitada a um ($k = 1$), ou seja, se todos os lugares tiverem um ou nenhum *token* para qualquer marcação alcançável a partir da marcação inicial.

B.3.6 Conservação

Em redes de Petri, um modelo é dito conservativo quando o número de *tokens* (marcas) é constante durante toda execução do modelo, ou seja, quando não existe perda nem acréscimo de *tokens* ao modelo. Essa propriedade é importante, pois dado que pode ser analisada de maneira indireta através da estrutura do modelo (conservação estrutural e limitação estrutural), é possível avaliar *a priori* a possibilidade de construir o espaço de estados do modelo.

Todas estas propriedades são genéricas, sendo suas interpretações dependentes do sistema real que está sendo representado. Por exemplo, pode-se imaginar um sistema que tem o seu término normal, sendo este um estado de *deadlock*. Apesar disso, o que o projetista com certeza tentará verificar é se para qualquer marcação alcançável o estado de *deadlock* pode ser alcançado, significando que sempre o sistema pode ser encerrado normalmente.

B.3.7 Técnicas de Análise Estrutural

Em situações onde a geração do espaço de estado leva a uma explosão do número de estados, as análises estruturais se mostram como uma alternativa importante, pois não necessitam da construção do espaço de estados e analisam apenas a estrutura da rede. Para realizar as verificações são utilizadas técnicas baseada em conceitos de álgebra linear e de teoria dos grafos, representando as soluções baseadas na equação de estados e em objetos estruturais, respectivamente. Os *Siphons* e as *Traps*, que serão detalhados nas subseções seguintes, são exemplos de objetos estruturais.

A utilização de álgebra linear depende da resolução da equação de estados:

$$\mathbf{M}' = \mathbf{M} + \mathbf{C}(.,t)^{\mathbf{T}} \tag{1.1}$$

onde

M’ é a marcação alcançada a partir do disparo de *t* (vetor linha)*;*

M é uma marcação alcançável da rede (vetor linha);

C é a matriz de incidência ($\mathbf{C} = \mathbf{O}^{\mathbf{T}} - \mathbf{I}^{\mathbf{T}}$), e O^T e I^T são as matrizes transpostas de *O* e *I* respectivamente. A representação $\mathbf{C}(.,t)$ representa o vetor coluna da matriz **C** para a transição *t*.

Esta resolução nos permite inferir propriedades que independem da marcação inicial, como limitação estrutural, conservação estrutural, repetitividade, entre outras. A limitação deste aparato matemático é que quase sempre algumas condições precisam ser relaxadas, fazendo com que os resultados possibilitem determinar condições necessárias, mas muitas vezes não suficientes para a verificação de propriedades. Em alguns casos, podemos ter condições suficientes para o não atendimento da propriedade, pois as soluções encontram-se em um conjunto maior chamado de LRS (*Linear Reachability Set*), que por sua vez contém a RS (*Reachibility Set*). Logo, se a condição não for alcançada pela equação de estado não será pelo grafo de alcançabilidade.

Invariantes de Transição

Se em uma rede de Petri é possível, a partir de uma marcação *M'*, dispararmos cada transição *n* vezes e retornarmos a marcação inicial *M'*, ou seja, a marcação obtida após o disparo da seqüência de transições (σ) é a marcação de partida, diz-se que esta rede tem componentes repetitivos estacionários que correspondem a comportamentos cíclicos da rede. O vetor V_σ, chamado de de Invariante de Transição, corresponde ao vetor característico de números inteiros não negativos representando o número de vezes que cada transição disparou na seqüência σ [Maciel et al. 1996].

Através do cálculo dos invariantes de transição é possível verificar a existência dos componentes repetitivos estacionários (reversibilidade e repetividade), bem como verificar a consistência parcial ou total de uma rede.

Invariantes de Lugar

A obtenção dos Invariantes de Lugar de uma rede provê o que é denominado componentes conservativos do modelo. Estes componentes estão associados ao conjunto de lugares em que a soma ponderada de marcas (*tokens*) desses lugares permance constante quando uma seqüência de transições é disparada. Esta é a condição necessária e suficiente para uma rede ser dita conservativa.

Seja $\mathbf{Y}$ um vetor coluna de $|P|$ elementos com pesos, onde $\mathbf{Y} = \left[y_1, y_2, ..., y_{|P|} \right]^{\mathbf{T}}$ e as entradas y_n são números inteiros não negativos. Considere o produto escalar entre o vetor linha representando uma marcação arbitrária *M'* e $\mathbf{Y}$ (denotado $\mathbf{M}' \cdot \mathbf{Y}$). Se $M\,[t>M'$, então utilizando a Equação (1.1) chega-se a:

$$\mathbf{M}' \cdot \mathbf{Y} = \mathbf{M} \cdot \mathbf{Y} + \mathbf{C}(.,t)^{\mathbf{T}} \cdot \mathbf{Y} \tag{1.1}$$

Obviamente se $\mathbf{C}(.,t)^{\mathbf{T}} \cdot \mathbf{Y} = 0$, significa que o vetor de contador de número de *tokens* $\mathbf{Y}$ é invariante com respeito ao disparo de *t*. De forma mais genérica, se

$$\mathbf{C}^{\mathbf{T}} \cdot \mathbf{Y} = 0 \tag{1.2}$$

isto é, o vetor $\mathbf{Y}$ é uma solução para o conjunto de equações lineares

$$\forall t \in T: \ \mathbf{C}(.,t)^{\mathbf{T}} \cdot \mathbf{Y} = 0 \tag{1.3}$$

então não interessa a seqüência de disparos, o vetor $\mathbf{Y}$ não se altera e permanece o mesmo para qualquer marcação alcançável de uma dada marcação inicial *M*. Sendo assim, o vetor $\mathbf{Y}$ que satisfaz a equação (1.2) é chamado o *P-semiflows* da rede de Petri em análise. Esta relação de invariante é chamada *Invariante de Lugar*. Esta relação é uma importante técnica de análise qualitativa, pois uma vez que possamos provar que todos os lugares de uma rede de Petri são cobertos por algum invariante de lugar podemos afirmar que a rede possui limitação estrutural e com isso o espaço de estados é finito para qualquer marcação inicial.

Siphons e Traps

Os *Siphons* e as *Traps* são propriedades estruturais das redes de Petri que se caracterizam principalmente por auxiliarem na prova de teoremas e análises quanto à propriedades como ausência de *deadlock*.

Estas propriedades estruturais tem níveis de complexidade para serem calculadas bem menores do que os invariantes e o grafo de cobertura, podendo ser um forte aliado nas análises qualitativas.

Definição 12. (*Siphon*) Um *Siphon* é um conjunto de lugares *S*, onde $Pre(S) \subseteq Pos(S)$ e *Pre(S)* e *Pos(S)* são conjuntos de transições que fazem parte de *I* e *O* (ver Definição 1), respectivamente. Logo, para cada transição que adiciona um ou mais *tokens* para um lugar em *S*, ela também retira pelo menos um *token* de um lugar qualquer em *S*. Com esta propriedade, o Teorema 1 é definido como segue:

> ***Teorema 1.*** *Assuma uma rede marcada com um siphon S. Se S não está marcado na marcação inicial, então S não estará marcado em nenhuma marcação alcançável da rede.*

Teoremas como esta permite inferir aspectos como transições mortas e, conseqüentemente, características de *deadlock*. Um outro conceito estrutural é o *Trap* que funciona de forma inversa e complementar ao *Siphon*, fazendo com que este conjunto de lugares uma vez marcados permaneçam marcados durante toda a execução da rede.

Definição 13. (*Trap*) Um *Trap* é um conjunto de lugares *S* que nunca perdem todos os seus *tokens*, pois $Pos(S) \subseteq Pre(S)$. Uma condição suficiente é que cada transição que retirar um ou mais *tokens* de *S*, adiciona pelo menos um a *S*. Com esta propriedade podemos definir o teorema a seguir:

> ***Teorema 2.*** *Assuma uma rede marcada com um trap S. Se S está marcado na marcação inicial, então S estará marcado em todas as marcações alcançáveis da rede.*

Se for combinado os conceitos de *siphon* e *trap* pode-se produzir teoremas que mostram que se uma rede possui para cada *siphon* não vazio, sem lugares isolados, um *trap* marcado na marcação inicial, esta rede não possui *deadlock*.

Como os *traps* uma vez marcados permanecem marcados e os *siphons* uma vez desmarcados permanecem desmarcados na evolução comportamental da rede, pode-se dizer que estes predicados são predicados estáveis. Quando um conjunto de lugares é ao mesmo tempo um *trap* e um *siphon*, ou seja um suporte para invariantes de lugar, chama-se este conjunto de *trap-siphon component*.

Essas propriedades são utilizadas também para prova de teoremas, pois como formam predicados estáveis, permancem inalterados para todas as marcações.

Para resumir, a importância das técnicas de análise estrutural é que elas são calculadas apartir de mecanismos que se utilizam apenas da estrutura da rede, não sendo necessária a construção do espaço de estados, que muitas vezes é computacionalmente inviável ou impossível.

B.3.8 Técnicas de Análise baseadas em Espaço de Estados

Como já foi mencionado antes, existem propriedades das redes de Petri que dependem da marcação inicial. A técnica usada para verificar estas propriedades é chamada de técnicas de análise baseada em espaço de estado (ou alcançabilidade). Estas técnicas são baseadas na construção de um RG (grafo de alcançabilidade) da rede de Petri, e é válida apenas quando o RG e o RS

(conjunto de alcançabilidade) são finitos, ou seja, quando a rede de Petri é limitada.

Em muitos casos, o crescimento no número de de marcações é uma combinação do número de lugares e do número de tokens presentes na marcação inicial. As técnicas de análise de alcançabilidade são muito poderosas pois permitem provar a maioria das propriedades de interesse através da investigação do RG e do RS, visto que estes possuem toda as evoluções da rede. Entretanto, é comum que a complexidade de espaço e tempo exceda limites aceitáveis devido a alta complexidade dos algoritmos aplicados sobre o RG., associado a uma comum explosão do número de estados quando trata-se de um modelo para um sistema real.

Uma vez que o RG tenha sido construído, as propriedades podem ser verificadas utilizado-se algoritmos clássicos para análise de grafos, assim como explicado na seqüência para cada propriedade discutida no início desta seção.

B.3.9 Analisando as Propriedades Qualitativas

Esta subseção faz um resumo de como realizar análise das principais propriedades qualitativas utilizando tanto técnicas estruturais quanto comporatamentais (baseadas no espaço de estados).

Limitação

Um lugar $p \in P$ de uma rede de Petri é limitada a *k* se e somente se

$$k = \max_{M \in RS} M(p)$$

Sendo obviamente esta propriedade somente possível de ser calculada para RS finitos.

Limitação e Segurança

Para verificar a propriedade de limitação de uma rede de Petri é suficiente mostrar que todos os lugares da rede são cobertos por pelo menos um invariante de lugar. Para verificar a segurança de uma rede é suficiente que os lugares destes invariantes tenham todos uma única marca (*token*) para sua marcação inicial.

Se a rede não for coberta por invariantes de lugar, ou seja, nem todos os lugares são cobertos por invariantes de lugar, ela pode ainda ser limitada em situações especiais, onde prioridades, funções de guarda e arcos inibidores são utilizados. Contudo, na grande maioria dos casos, esta situação leva a redes ilimitadas. Recomenda-se ao modelador que analise cuidadosamente os lugares não cobertos por invariantes de lugar de maneira que limites “artificiais”, eventualmente, possam ser inseridos (lugares complementares, arcos inibidores) – obviamente sem deturpar de maneira significativa o caso em estudo – e que venha a tornar o modelo estruturalmente limitado e por conseqüência limitado.

Conservação

É condição necessária e suficiente que para verificação desta propriedade que todos os lugares da rede sejam cobertos por pelo menos um invariante de lugar. Uma rede conservativa é limitada.

Consistência

Uma rede é dita consistente quando existe uma sequência de transições *s* que permite $M_0 \lfloor s \rangle M_0$, e todas as transições $t \in T$ fazem parte desta sequência pelo menos uma vez. Para verificar esta propriedade é suficiente que exista um invariante de transição com o vetor característico V_σ sem valores nulos.

Alcançabilidade

Para verificar se uma dada marcação *M'* é alcançavel a partir de uma marcação *M* numa rede de Petri, é necessário apenas checar se o RG da rede em questão possui um caminho direto ligando *M* até *M'*. A concatenação dos rótulos de cada arco (que representam as transições na rede) formam a seqüência de disparos σ tal que $M[\sigma > M'$.

Reversibilidade

Para verificar se uma dada marcação *M* é um *home state*, é suficiente construir o conjunto de todas as marcações que podem ser alcançadas a partir de *M* em RG , e verificar que este conjunto é igual a RS(*M*). Uma rede de Petri é reversível se e somente se a marcação inicial é um *home state*. Isto é verdade se e somente se RG tem um componente fortemente conectado que contém a marcação inicial.

Reversibilidade e Repetitividade

Se uma rede é consistente, necessariamente é repetitiva, pois a propriedade de consistência é mais restritiva que a de repetitividade. Para que uma rede seja reversível a sua marcação inicial, a rede deverá ter pelo menos um componente repetitivo – condição necessária para reversibilidade.

Liveness

Uma transição *t* de uma rede de Petri é viva se e somente se de qualquer marcação $M \in RS$ é possível alcançar uma nova marcação *M'* tal que $t \in E(M')$, onde *E*(*M'*) é o multiconjunto de todas as transições habilitadas na marcação *M'*. Pode-se também definir que *t* é viva se e somente se o RG não contém marcações mortas e *t* faz parte de um componente fortemente conectado de RG.

Ausência de Deadlock

A presença de *deadlock* pode ser verificada em RG procurando-se por uma marcação morta, ou seja, um nó do grafo sem arcos de saída. A ausência de nós deste tipo é suficiente para garantir a ausência de *deadlock*.

Liveness e Ausência de Deadlock

Uma rede coberta por invariantes de transição tem uma propriedade necessária para *liveness*. Uma rede Pura é *live* e limitada se é coberta por invariante de lugar, todos os invariantes de lugar são inicialmente marcados e nenhum *siphon* nunca fica desmarcado (por exemplo se todos os *siphons* tiverem *traps* marcados).

Tamanho do espaço de estados

O tamanho do espaço de estados gerado por uma rede de Petri limitada pode ser estimado de forma preliminar a partir de dois parâmetros: o número total de tokens na marcação inicial (k) e o número de lugares (N). A fórmula básica para esta estimativa é $(k+1)^N$. Obviamente, esta estimativa é grosseira e pessimista, pois considera que cada lugar poderá obter todos os tokens da rede em dado instante.

Se a rede é limitada e todos os seus lugares (p_i) são cobertos por pelo menos um invariante de lugar, logo pode-se obter para cada lugar o número de tokens máximo permitido. Com isso uma estimativa mais precisa é representada pela expressão $\prod_{i=1}^{N}(k_i+1)$, onde k_i é o número de tokens no lugar p_i.

B.4 Extensão Temporal das Redes de Petri

Atualmente existem diversas variações do modelo original de redes de Petri. Dentre elas, encontram-se as redes de Petri temporizadas que serão utilizadas neste projeto, mas especificamente as redes GSPN (*Generalized Stochastic Petri Net*). As redes de Petri temporizadas permitem a análise de propriedades qualitativas, análise de desempenho e confiabilidade de sistemas.

O conceito de tempo foi desconsiderado nas pesquisas iniciais, devido aos efeitos da adição dessa semântica no modelo proposto. Estes efeitos poderiam modificar regras de disparos, regras de habilitações e principalmente destruir a importante característica de que o todos os possíveis comportamentos da rede poderiam ser expressos através da definição de sua estrutura.

A Figura B-7-16 apresenta uma classificação de algumas extensões temporizadas das redes de Petri. Nesta figura pode-se verificar que as extensões temporizadas são classificadas inicialmente pela forma como o tempo foi incorporado ao modelo, ou seja, o tempo pode ser associado aos lugares, aos *tokens*, aos arcos e finalmente às transições, sendo esta última a mais comum entre os modelos temporizados para rede de Petri.

Na classe *Places-Timed* o tempo está associado aos lugares, logo os *tokens* gerados em um lugar de destino só ficam disponíveis para dispararem novas transições após transcorrido o *delay* que está associado aquele lugar. Nas *Tokens-Timed*, o tempo está associado ao *token*, logo cada *token* possui uma etiqueta que determinará o tempo (*timestamp*) em que este deve disparar novas transições. No caso da classe *Arcs-Timed*, o tempo está associado aos arcos representando o *delay* para que os *tokens* viajem do lugar para a transição, fazendo com que a transição só possa disparar após este tempo. Por fim, na classe *Transitions-Timed* o tempo está associado à transições, fazendo com que o início da atividade esteja associado à habilitação da transição e o seu término, após transcorrido o *delay*, corresponda ao disparo da transição.

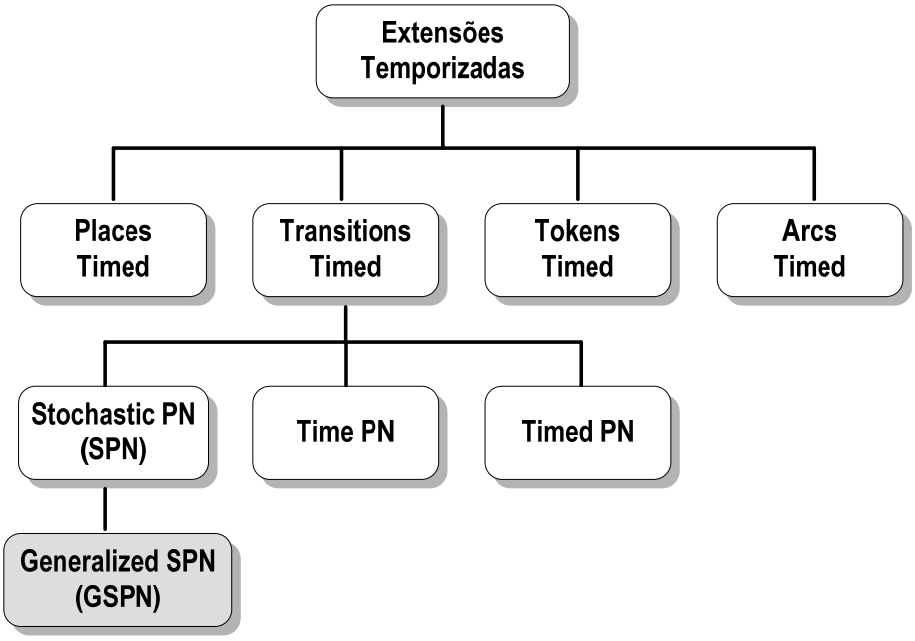


Figura B-7-16 - Taxonomia da família temporizada das redes de Petri.

O disparo de uma transição corresponde à mudança do estado do sistema real. Esta mudança ocorre, normalmente, ou pela satisfação de alguma pré-condição, ou pela finalização de alguma atividade. Desta forma, podemos analisar que transições podem ser utilizadas para modelar atividades realizadas pelo sistema, onde sua habilitação representa o início da atividade, que transcorre por um *delay* e tem seu término representado pelo disparo da transição.

B.4.1 Regras de Disparo

O tempo está naturalmente associado às transições, desta forma duas regras podem ser definidas para representar semanticamente os disparos. A primeira regra representa o disparo atômico, ou seja, o *token* fica no lugar durante o tempo (*delay*) necessário para a execução da atividade, e quando acontece o disparo o *token* é retirado do lugar de entrada e passa para o lugar de saída atomicamente (sem consumo de tempo). Esta regra preserva a semântica definida no modelo original (não temporizado) das redes de Petri, fazendo com que o estudo qualitativo das redes possa utilizar a mesma teoria desenvolvida para as redes não temporizadas, como: invariantes, conjunto de alcançabilidade, etc. Os tempos (*delays*) podem ser representados por constantes ou por intervalos, onde estes últimos representam uma faixa com possiblidade uniforme de disparo.

A segunda regra de disparo possível é conhecida como disparo em três fases, pois quando a transição está habilitada os *tokens* são retirados dos lugares de entrada e só são gerados nos lugares de saída após transcorrido o tempo associado à transição. Neste modelo os tempos são representados na forma de duração (*duration*). Desta forma, pode-se definir formalmente uma sequência de execução de uma rede de Petri temporizada como uma sequência de tuplas, *(t,T)*, formada pela transição que disparou (*t*) e o tempo de execução transcorrido até o momento (*T)*.

B.4.2 Regras de Seleção

A Figura B-7-17 apresenta uma situação de conflito entre as transições *T1* e *T2*, onde na abordagem não temporizada se configurava como uma situação onde uma escolha não determinística era a solução (*free-choice*).

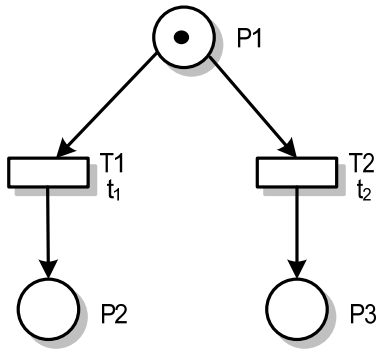


Figura B-7-17 - Transições conflitantes.

A semântica temporizada permite a solução destes conflitos utilizando-se duas regras de seleção existentes: pré-seleção e *race condition* (ou condição de corrida). Na regra de pré-seleção a escolha é realizada utilizando-se alguma métrica definida durante a construção da rede (ex.: prioridade). Na regra de seleção baseada em *race condition*, a seleção é realizada em tempo de execução de acordo com o *delay* de cada transição (t_1 e t_2), sendo escolhida a de menor valor.

No exemplo da Figura B-7-17, se considerar a regra de disparo atômica tem-se uma condição de conflito que poderá ser resolvida por uma das duas regras de seleção apresentadas. Considerando a regra de disparo em três fases, restará a regra de pré-seleção ou o comportamento será o mesmo atribuído a redes não temporizadas (escolha não determinística).

B.4.3 Semântica da Memória do Tempo

Um importante aspecto a ser considerado quando o foco são as redes temporizadas é o tratamento dispensado ao histórico de execução de cada transição quando acontece um disparo. Ou seja, o tempo já transcorrido de execução (no modelo em questão, representado pelo *delay*) de cada atividade (transição habilitada), quando uma outra transição dispara. Este aspecto conhecido como memória, pode ser tratado por dois mecanismos básicos: *continue* e *restart*.

No mecanismo *continue*, os relógios individuais de todas as transições são mantidos na situação atual do disparo de uma transição específica para que sejam continuados em um momento posterior. O único relógio que efetivamente sofre alteração é da transição que disparou, sendo o mesmo reinicializado.

No mecanismo *restart*, os relógios individuais de todas as transições são reinicializados no momento em que qualquer disparo na rede ocorra, desta forma a memória do tempo já transcorrido em alguma atividade é perdida.

Existem três modelos para tratamento de memória: *Resampling*, *Enabling Memory*, *Age Memory*. Os modelos temporais de redes de Petri normalmente disponibilizam estes modelos de tratamento de memória onde cada um deles se utiliza dos mecanismos *continue* e *restart* para definir suas semânticas.

No modelo *Resampling*, a cada transição que dispara o relógio de todas as transições temporizadas são descartados, e os seus valores reinicializados (mecanismo *restart*), ou seja, nenhuma memória das execuções das atividades são consideradas. Este modelo pode ser utilizado para representar, por exemplo, atividades que disputam um único resultado, como no caso de execuções paralelas de testes de hispóteses, onde uma vez que uma das execuções terminou as demais são irrelevantes (não precisam ser continuadas).

No modelo *Enabling Memory*, a cada transição que dispara o relógio de todas as transições temporizadas que se tornam desabilitadas são descartados e reinicializados (mecanismo *restart*). Por sua vez, os relógios das transições que permacem habilitadas são mantidos (mecanismo *continue*), com exceção da transição que disparou, ou seja, a quantidade de tempo já gasta na execução de uma atividade que permanece habilitada é mantida.

No modelo *Age Memory*, a cada transição que dispara o relógio de todas as transições temporizadas são mantidas em seus valores (mecanismo *continue*), ou seja, todo o tempo gasto na execução daquelas atividades será mantido e utilizado no futuro, mesmo que a transição seja temporariamente desabilitada. A única exceção é a transição que disparou, pois nesse caso o seu relógio é reinicializado.

B.4.4 Semântica de Temporização

Quando se tem modelos em redes de Petri que possuem grau de habilitação de suas transições maior que um, pode-se considerar três semânticas diferentes para determinar o disparo simultâneo ou não da mesma transição: *Single-Server*, *Infinite-Server* e *Multiple-Server*.

A semântica *Single-Server* é caracterizada pelo disparo singular de suas transições, ou seja, supondo que tem-se uma transição *t*, com *delay* associado *T*, e grau de habilitação três, tem-se três disparos sequenciais com cada um levando *T* unidades de tempo para acontecer. Isto acontece porque após cada disparo o relógio é reinicializado, logo levará *3T* para efetuar-se os três disparos.

Na semântica *Infinite-Server*, o mesmo exemplo apresentado levaria apenas *T* unidades de tempo para executar os três disparos, pois o comportamento desta semântica é equivalente a termos infinitos processadores executando a mesma tarefa em paralelo, logo os três disparos poderão acontecer simultaneamente.

Na semânica *Multiple-Server*, o comportamento é similar a termos uma semântica *infinite-server*, apenas com a limitação de só conseguirmos executar em paralelo *k* execuções da mesma atividade, e não mais infinitas execuções como no modelo *Infinite-Server*.

B.4.5 Redes de Petri Estocásticas (SPN)

Dentre os formalismos que compõe a família redes de Petri, as redes de Petri Estocásticas (SPN – *Stochastic Petri Nets*) e suas extensões têm sido utilizadas como mecanismos de alto nível para geração automática de cadeias de Markov [Bolch et al. 1998], permitindo que as dificuldades associadas à construção destas cadeias sejam reduzidas.

Além disto, dado que a modelagem dos sistemas é baseada em estados locais, ou seja, foca-se na modelagem dos componentes ao invés de necessariamente ter que observar o sistema como um todo, é possível para o projetista concentrar-se mais no problema a ser solucionado do que suplantar dificuldades e restrições associadas à técnica de representação. Outra vantagem das SPNs é que permitem a análise e verificação de propriedades qualitativas dos modelos.

A principal característica das SPNs é que o tempo associado as transições passam a serem considerados com distribuição exponencial, em vez de constantes ou uniformente distribuídos assim como nos modelos discutidos anteriormente. Esta propriedade é que possibilita a equivalência semântica destas redes com as cadeias de Markov, pela ausência de memória das distribuições exponenciais.

Generalized Stochastic Petri Nets (GSPN)

Em algumas redes temporizadas precisamos modelar atividades internas que não consomem tempo, ou que possuem uma escala de tempo desprezível comparada à demais transições. Para atender esta demanda uma nova extensão das redes SPN foi criada, as Generalized SPN (ou simplesmente GSPN) [Marsan et al. 1984], contemplando transições chamadas imediatas que possuem *delay* nulo, ou seja, disparam imediatamente após sua habilitação e possuem prioridade sobre transições temporizadas. As demais características ficam inalteradas, tornando esta extensão uma das mais utilizadas para análise de desempenho e confiabilidade.

Pode-se definir formalmente uma GSPN como uma tupla $(P, T, I, O, H, G, M_0, W, \Pi)$, onde P é o conjunto de lugares; T é o conjunto das transições; I, O e H são relações descrevendo as pré-condições, pós-condições e condições de inibição respectivamente; G representa as funções de habilitação (ou guarda) associadas as transições imediatas; M_0 é a marcação inicial da rede; W associa para cada transição temporizada um número real não negativo representando o tempo (ou taxa) exponencial da transição (na verdade o parâmetro da distribuição exponencial), e para cada transição imediata um número real não negativo representando o peso probabilístico da transição ocorrer; e Π associa para cada transição imediata um número natural que representa o nível de prioridade da transição.

B.5 Modelando Atividades Não-Markovianas

Uma variável aleatória é uma função que atribui um número real ao resultado de um experimento aleatório [Bolch et al. 1998], e.g., o número de requisições submetidas à uma aplicação de banco de dados e o número de tarefas submetidas para um sistema computacional. O intervalo de tempo gasto para enviar uma mensagem para o MOM também representa um exemplo de uma variável aleatória. Os valores destas variáveis (tempo) podem ser contínuos ou discretos. Em nosso contexto as variáveis analisadas são contínuas.

Definição 14. Uma variável aleatória contínua X que pode assumir todos os valores no intervalo [a,b], onde $-\infty \leq a < b \leq +\infty$ é chamada de uma variável aleatória contínua. Ela é descrita por uma função de distribuição (também chamada de função distribuição acumulada):

$$F_X(x) = P(X \leq x)$$

a qual especifica a probabilidade de que a variável aleatória *X* assuma valores menores ou iguais a *x*, para todo *x* [Bolch et al. 1998].

A distribuição exponencial é um exemplo de uma variável aleatória contínua onde a função distribuição acumulada é representada por

$$F_X(x) = \begin{cases} 1 - \exp\left(-\frac{x}{\overline{X}}\right) & , 0 \leq x < \infty \\ 0 & , \text{caso contrário} \end{cases}$$

com média $\overline{X} = \frac{1}{\lambda}$, variância $\text{var}(X) = \frac{1}{\lambda^2}$, coeficiente de variação igual a 1, e onde λ é o parâmetro da distribuição exponencial. Desta forma, a distribuição exponencial é completamente determinada pelo seu valor médio $\overline{X}$.

Uma atividade qualquer de um sistema que possa ser representada por uma variável aleatória é dita ser uma atividade markoviana quando ela atende ao princípio markoviano (ausência de memória). No domínio das variáveis aleatórias contínuas a distribuição exponencial é a única que atende a esta propriedade. Desta forma, a definição de uma atividade não-markoviana é toda e qualquer atividade que não pode ser representada por uma distribuição exponencial.

As redes GSPNs utilizadas para modelar os sistemas neste projeto exigem transições que sejam exponencialmente distribuídas, se configurando a princípio como um limitador para representar atividades não-markovianas.

É necessário ressaltar a importância do tratamento a ser considerado para especificação e avaliação de sistemas com características não-markovianas. O leitor pode se perguntar por que não especificar as atividades com características não-markovianas utilizando-se uma distribuição de probabilidade adequada? Uma resposta a esta preocupação pode ser dividida em duas partes, ambas com fortes apelos pragmáticos. Como primeiro aspectos, pode-se ressaltar o fator de que a distribuição de probabilidade dos tempos das atividades pode não ter sido especificada analiticamente, contudo obtida a partir de dados históricos ou mesmo por amostragem e nenhuma função analítica se apresentar como alternativa adequada de representação, restando-nos, portanto, representar tais distribuições empíricas a partir de resumos estatísticos. O segundo aspecto que deve ser ressaltado é a complexidade do tratamento analítico dos processos estocásticos (conjunto de variáveis aleatórias) não-markovianos. Esta não é uma tarefa trivial. Portanto, uma alternativa pragmática a tais restrições consiste em transformar (aproximar) distribuições analíticas de tratamento computacional e analítico complexos em distribuições poli-exponenciais (polinômios de exponenciais) e portanto utilizar a teoria markoviana (ou semi-markoviana) para tratar tais modelos.

Esta subseção tem o objetivo de apresentar uma abordagem alternativa para representar atividades não-markovianas utilizando as redes GSPN, ou seja, apenas transições exponenciais. Para permitir a explicação deste método de modelagem serão apresentadas duas outras distribuições: Erlang e Hiperexeponencial.

B.5.1 Distribuições Não-Exponenciais

A distribuição Erlang (E_k) pode ser utilizada para aproximar distribuições empíricas, onde se tenha apenas a média e o desvio padrão da amostra. A condição para que esta aproximação seja possível é que o inverso do coeficiente de variação da amostra seja maior que um e inteiro (quando este valor não é inteiro, a distribuição hipo-exponencial pode ser uma alternativa viável). Esta distribuição é caracterizada por um arranjo seqüencial de *k* (fases) exponenciais com média $\overline{X}/k$ (ver Figura B-7-18).

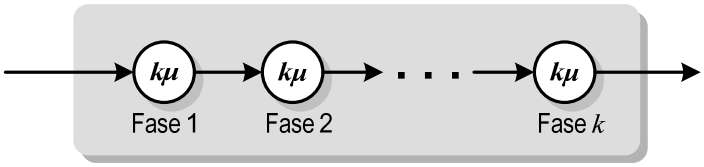


Figura B-7-18. Modelo de uma Erlang.

A função de distribuição da Erlang, a sua média, varância e coeficiente de variação podem ser definidos por:

$$F_X(x) = 1 - e^{-k\mu x} \cdot \sum_{j=0}^{k-1} \frac{(k\mu x)^j}{j!}, \quad x \geq 0, k = 1,2,...$$

$$\text{média} : \overline{X} = \frac{1}{\mu}$$

$$\text{variância} : \text{var}(X) = \frac{1}{k\mu^2}$$

$$\text{coef. variação} : c_X = \frac{1}{\sqrt{k}} \leq 1$$

A distribuição hiper-exponencial (H_k) pode ser utilizada para aproximar distribuições empíricas que possuem o inverso do coeficiente de variação menor que 1, onde *k* é o número de fases. A Figura B-7-19 apresenta um modelo para esta distribuição, que é representada por um arranjo paralelo de *k* exponenciais, onde apenas uma fase é ocupada em cada instante de tempo. Os parâmetros q_k determinam a probabilidade de ocorrência de cada uma das exponenciais, fazendo com que a probabilidade de uma fase *j* ser considerada é determinada pelo q_j.

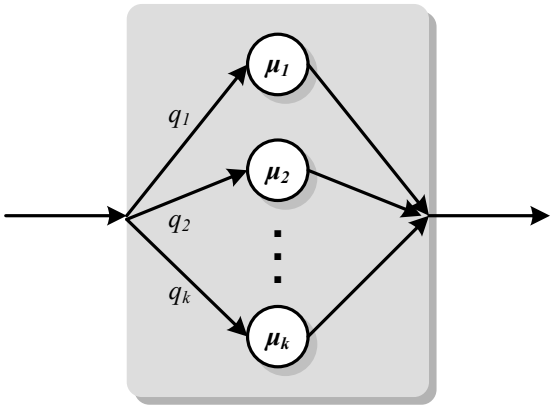


Figura B-7-19. Modelo de uma Hiperexponencial.

A função distribuição para a hiper-exponencial, sua média, a variância e o coeficiente de variação são definidos por:

$$F_X(x) = \sum_{j=1}^{k} q_j (1 - e^{-\mu_j x}), x \geq 0$$

$$\text{média} : \overline{X} = \sum_{j=1}^{k} \frac{q_j}{\mu_j} = \frac{1}{\mu}, x > 0$$

$$\text{variância} : \text{var}(X) = 2 \cdot \sum_{j=1}^{k} \frac{q_j}{\mu_j^2} - \frac{1}{\mu^2}$$

$$\text{coef. variação} : c_X = \sqrt{2\mu^2 \sum_{j=1}^{k} \frac{q_j}{\mu_j^2} - 1} \geq 1$$

B.5.2 Modelando Atividades Não-markovianas em GSPN

Os modelos GSPN são restritivos quanto aos tipos de transições que suportam e permitem apenas transições imediatas ou temporizadas com *delay* exponencialmente distribuído. O parâmetro que é fornecido para a rede é o *delay* (retardo) médio de uma distribuição exponencial (para cada transição temporizada).

Uma abordagem possível para aproximar distribuições não-exponencias é a utilização de um tipo especial de distribuição chamada *Phase Type Distribution* [Neuts 1975] ou distribuição por fases. Este tipo de distribuição utiliza uma composição de exponenciais para representar outras distribuições, dividindo o processo em várias fases e podendo para cada fase utilizar distribuições exponenciais com parâmetros específicos. Na subseção anterior foram apresentados dois tipos de distribuições (Erlang e hiper-exponencial) que podem ser caracterizadas como *Phase Type Distributions*, pois são composições de exponenciais distribuídas em fases.

Associado ao conceito de *Phase Type Distributions* foram definidos algoritmos que fazem mapeamentos de distribuições empíricas em combinações de exponenciais pela associação de momentos, chamados de algoritmos de casamento de momentos (*moment matching algorithms*). Este projeto utiliza o algortimo apresentado por [Desrochers e Al-Jaar 1995] que aproveita o fato de que distribuições de Erlang têm média maior que o desvio padrão, enquanto distribuições hiper-exponenciais possuem média menor que o desvio padrão. Esta observação é usada para propor a representação de uma atividade não-markoviana (com retardo seguindo uma distribuição empírica qualquer) através de sub-redes que representem distribuições de Erlang ou hiper-exponenciais, chamadas *s-transitions*.

Desta forma, de acordo com o coeficiente de variação (σ/μ) associado à duração de uma atividade, uma implementação de *s-transition* (Erlang ou hiper-exponencial) pode ser selecionada. Para cada implementação de *s-transition* há parâmetros que podem ser configurados de forma que o primeiro e o segundo momentos associados aos retardos de uma atividade empírica igualem-se ao primeiro e o segundo momentos da *s-transition* como um todo.

Para detalhar este algoritmo, suponha que uma dada atividade não-markoviana *D* com retardo médio μ_D e desvio padrão σ_D é representada em um modelo GSPN como uma transição temporizada T_D que tem como lugar de entrada *P1* e lugar de saída *P2*. Diante deste contexto, quatro situções podem configurar:

7. Se $\mu_D = \sigma_D$, então a atividade pode ser aproximada por uma transição exponencial com média igual a μ_D;
8. Se μ_D / σ_D é um inteiro *x*, e x ≠ 1, então a atividade pode ser aproximada por uma sub-rede Erlang com x^2 fases e média das transições exponencial igual a μ_D / x^2 ;
9. Para todos os demais casos onde $\mu_D > \sigma_D$, então a atividade pode ser aproximada por uma sub-rede Erlang antecedida por uma exponencial (ver Figura B-7-20) – rede hipo-exponencial, sendo o número de fases representado por γ, tal que $(\mu/\sigma)^2 - 1 \leq \gamma \leq (\mu/\sigma)^2$, e as taxas (inverso do retardo) das exponenciais $\lambda_1 = 1/\mu_1$ e $\lambda_2 = 1/\mu_2$ com:
10. $\mu_1 = \dfrac{\mu_D \mp \sqrt{\gamma(\gamma+1)\sigma_D{}^2 - \gamma \cdot \mu_D{}^2}}{\gamma+1}$ *e* $\mu_2 = \dfrac{\gamma \cdot \mu_D \pm \sqrt{\gamma(\gamma+1)\sigma_D{}^2 - \gamma \cdot \mu_D{}^2}}{\gamma+1}$
11. Para todos os demais casos onde $\mu_D < \sigma_D$, então a atividade pode ser aproximada por uma sub-rede hiper-exponencial (ver Figura B-7-21), sendo $r_1 = 2\mu_D{}^2/(\mu_D{}^2+\sigma_D{}^2)$, $r_2 = 1 - r_1$, e $\lambda = 2\mu_D/(\mu_D{}^2 + \sigma_D{}^2)$.

Aplicando o algoritmo descrito acima, as transições exponenciais utilizadas para representar atividades não-markovianas (ou seja, que não tenham natureza exponencial) podem ser substituídas por implementações de *s-transitions* correspondentes.

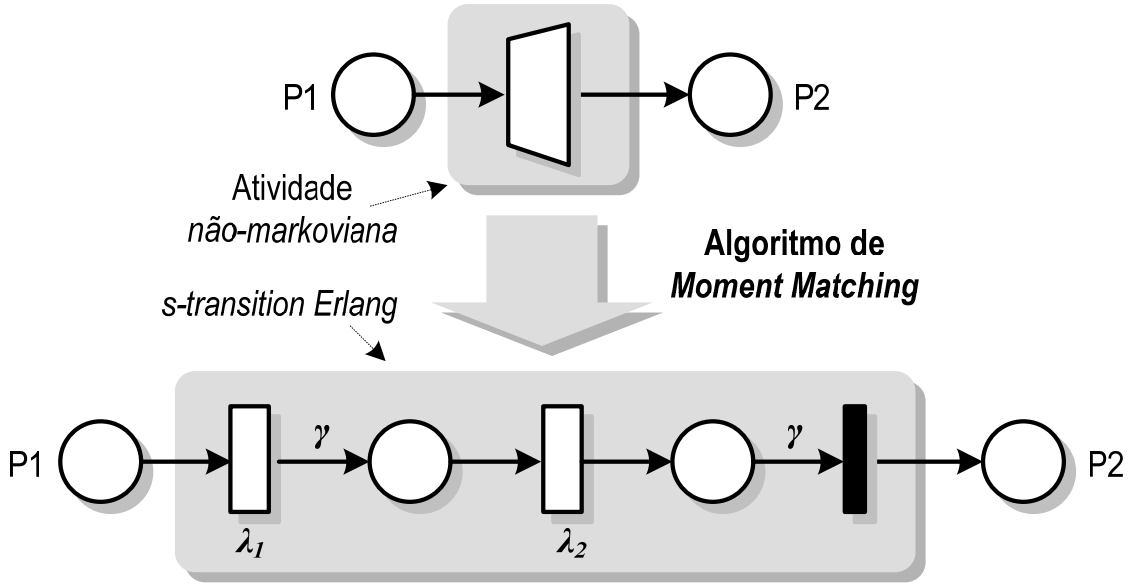


Figura B-7-20. Implementação de uma *s-transition* utilizando uma Erlang.

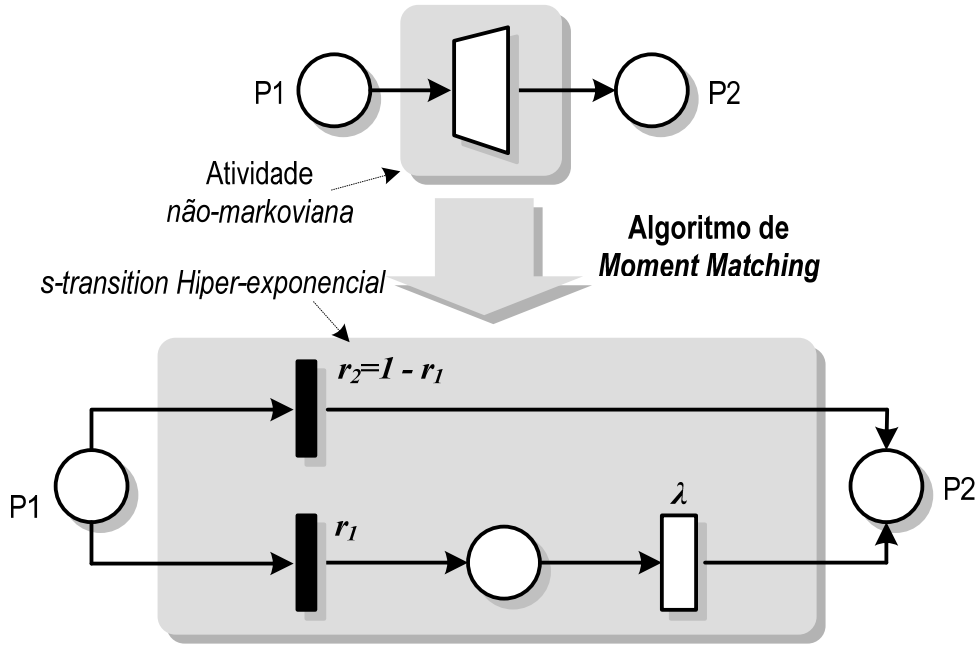


Figura B-7-21. Implementação de uma *s-transition* utilizando uma hiper-exponencial.

APÊNDICE

C

Validação da Biblioteca de Modelos

"O homem que não comete erros geralmente não faz nada"

Bishop W. C. Magee.

A biblioteca de componentes do JMSCapacity é formada por componentes de carga e de desempenho. Os componentes de desempenho são responsáveis em implementar as características de seus componentes de carga equivalentes em um formalismo capaz de ser resovido analiticamente ou através de simulação. No caso desta dissertação, o formalismo escolhido são as GSPNs.

Para permitir o uso em predições de desempenho esses componentes de desempenho precisam ser validados. Este apêndice tem o objetivo de efetuar a validação dos principais componentes de desempenho da biblioteca, em especial aqueles utilizados no cenário ilustrativo do Capítulo 5. Para isso, cinco experimentos são escolhidos de forma a exercitar os vários componentes envolvidos nesse cenário.

Ao longo deste apêndice são apresentadas as ferramentas utilizadas, informações gerais válidas para todos os experimentos, e em seguida o detalhamento de cada experimento.

C.1 Ferramentas Utilizadas

No Capítulo 4 (Seção 4.3) são apresentadas categorias de ferramentas recomendadas ou desenvolvidas pelo JMSCapacity. Esta seção apresenta detalhes sobre as ferramentas utilizadas durante a atividade *Parametrizar, Validar e Calibrar o Modelo*.

C.1.1 Ferramentas de Geração de Carga Artificial

Estas ferramentas geram carga de trabalho artificial para o MOM, simulando as cargas desejadas. Ao longo do desenvolvimento desta dissertação duas ferramentas foram utilizadas: *Apache JMeter* e *JMSMeter*.

Apache JMeter

Ferramenta desenvolvida em projeto no grupo Apache Jakarta, ela possui múltiplos mecanismos de geração de carga, para os mais diversos tipos de aplicação (JMS, HTTP, WebServices, FTP etc).

Ao longo deste trabalho, foram desenvolvidos *plugins* para esta ferramenta com o objetivo de melhorar a geração de carga para aplicações JMS. Foram desenvolvidos dois novos *Sampler* (geradores de carga) para mensagens PTP e *Pub/Sub*, permitindo a configuração de características como persistência, expiração, time-out, entre outras características que o original não permitia configurar.

Na construção do artigo [Arteiro et al. 2007] ela foi utilizada para geração de carga. Porém, durante esse trabalho ficaram nítidas algumas limitações no suporte à atividade de parametrização, onde se precisa ter maior controle sobre a carga de trabalho com o intuito de isolar cada componente básico de carga. Outra dificuldade está relacionada à apuração das métricas necessárias para a avaliação dos resultados.

JMSMeter

Ferramenta desenvolvida durante esta dissertação, ela permite a geração de carga de trabalho compatível com os componentes de carga especificados na biblioteca deste *toolkit*. Por conta disto, ela se constitui em uma contribuição deste trabalho.

C.1.2 Ferramentas de Monitoramento de Recursos

Estas ferramentas possibilitam a apuração de métricas de utilização do MOM durante o procedimeno de medição, como: *CPU Utilization* e *Destination Size*. O ambiente de medição utiliza o *JBoss Messaging* (*provider*) sob plataforma Microsoft, com as ferramentas de monitoramento dos próprios fabricantes.

Microsoft Performance & Reliability Monitor

Ferramenta de monitoração de desempenho da Microsoft, ela é capaz de monitorar a utilização de praticamente todos os objetos em execução no sistema operacional. A Tabela C-7-2 apresenta os objetos que podem ser utilizados para este efeito. Para cada objeto são apresentadas as métricas (objetos) utilizadas, indicando a sua aplicabilidade no processo.

Tabela C-7-2. *Performance Monitor*: objetos e métricas monitorados.

Objeto	Contador	Aplicabilidade
Processador	% tempo de processador	No cálculo do tempo de CPU do MOM. A instância a ser monitorada é a <_Total>.

Objeto	Contador	Aplicabilidade
Processo	% tempo de processador	No cálculo do tempo de CPU do MOM. As instâncias a serem monitoradas são: <java>, <idle> e <_Total>.
Processo	Bytes de disco/s	No cálculo do tempo de disco gasto pelo MOM. As instâncias <java> e <_Total> devem ser monitoradas.
PhysicalDisk	% tempo de disco	No cálculo do tempo de disco gasto pelo MOM. As instâncias dos discos utilzados pelo MOM devem ser monitoradas.

JBoss Logging Monitor

Esta ferramenta, na verdade, é um serviço de monitoramento de recursos habilitado no JBoss Messaging (JMS Provider), o qual pode monitorar qualquer recurso gerenciado pelo JBoss. No caso em questão, o interesse é pela métrica tamanho atual do *destination* (fila/tópico), sob estudo. A Tabela C-7-3 apresenta quais objetos são monitorados, suas métricas (atributos) e aplicabilidade no processo.

Tabela C-7-3. JBoss Logging Monitor: objetos e métricas monitorados.

Objeto	Atributo	Aplicabilidade
jboss.messaging.destination.Queue	MessageCount	Na monitoração da métrica *Destination Size* para filas.
jboss.messaging.destination.Topic	AllMessageCount	Na monitoração da métrica *Destination Size* para tópicos.

C.1.3 Ferramentas de Modelagem e Simulação

Na validação, essas ferramentas permitem avaliar os modelos PN para obter as métricas a serem comparadas com a medição do sistema real. Essas avaliações podem ser via resolução analítica ou simulação.

TimeNET

O TimeNet [Zimmermann 2001] [Zimmerman et al. 1999] é um pacote de software para modelagem e avaliação de redes de Petri estocásticas (SPNs) desenvolvido pelo *Institut für Technische Informatik da Technische Universität Berlin*.

A ferramenta suporta o desenvolvimento de vários modelos de redes de Petri estocásticas, dentre os quais se destacam: GSPN (*Generalised Stochastic Petri Nets*), DSPN (*Deterministic Stochastic Petri Net*) e eDSPN (*Extended Deterministic Stochastic Petri Net*). O modelo GSPN já foi detalhado no Apêndice B. O modelo DSPN estende o modelo GSPN introduzindo transições determinísticas que disparam em um tempo estabelecido. As eDSPN, ou

simplesmente EDSPN, são uma extensão das DSPN permitindo ainda a inclusão de transições poli-exponenciais, onde os tempos das transições podem ser estabelecidos por composição polinomial de várias exponenciais, arcos com expressões dependentes de marcação, expressões condicionais e lógicas associadas às transições imediatas e aos arcos. Como o modelo eDSPN compreende todos os demais modelos estocásticos permitidos pela ferramenta, todas as redes criadas são desse tipo.

O TimeNet possui um *framework* para análise qualitativa e avaliação que a torna uma ferramenta poderosa. Esse *framework* inclui desde algoritmos para cálculo de invariantes, passando pela função de *token game*, que permite a depuração *on-line* do modelo, até algoritmos sofisticados de simulação e resoluções numéricas (análise) estocásticas. As avaliações podem ser realizadas considerando a escala de tempo do modelo contínua ou discreta, além de permitir avaliar aspectos transientes ou estacionários do modelo.

O planejamento de experimentos torna-se mais fácil de ser implementado devido às rotinas que variam automaticamente um fator escolhido para o cálculo das métricas, segundo cada passo do intervalo de variação. Isso tudo associado a uma poderosa sintaxe para definição de métricas (fórmulas), condições de habilitação, valores dependentes de marcação, entre outros aspectos.

Um importante conjunto de funcionalidades da ferramenta é no suporte à avaliação, permitindo: avaliação estacionária ou transiente, análise (resolução analítica) ou simulação, tempo contínuo ou discreto, execução simples ou baseada em experimento.

C.2 Visão Geral dos Experimentos

Para efetuar a validação dos componentes de desempenho, cada experimento escolhido dá origem a um modelo GSPN que precisa inicialmente ser parametrizado para uma instalação específica (instanciação). Em seguida, eles são utilizados em um simulador para apuração das métricas de desempenho selecionadas para um MOM (Capítulo 3, Seção 3.2.1). Em contraponto é realizado um estudo de medição no sistema real definido como objeto do estudo (*System under Test*, SUT), apurando as mesmas métricas. Por fim, os resultados obtidos através da simulação são submetidos a um estudo comparativo com valores obtidos através de medições.

Como o objetivo desta biblioteca é auxiliar no planejamento de capacidade, logo, resultados com margem de erro de até 30% (medição x simulação) são considerados satisfatórios, e os modelos por conseqüência considerados válidos.

Antes de iniciar a descrição dos experimentos em si, é necessário descrever mais detalhes sobre o SUT, que é representado pelos modelos desta biblioteca durante os experimentos. Informações gerais que dizem respeito a todos os experimentos, como os parâmetros utilizados nas medições e simulações; e as fórmulas para cálculo das métricas também estão inclusos nesta seção.

C.2.1 Descrição do SUT

O sistema real escolhido para implementar o serviço de entrega de mensagens é composto pelos seguintes elementos: servidor, estação cliente e rede. A Figura

C-7-22 apresenta as configurações de hardware e os softwares que estão em execução em cada elemento do sistema.

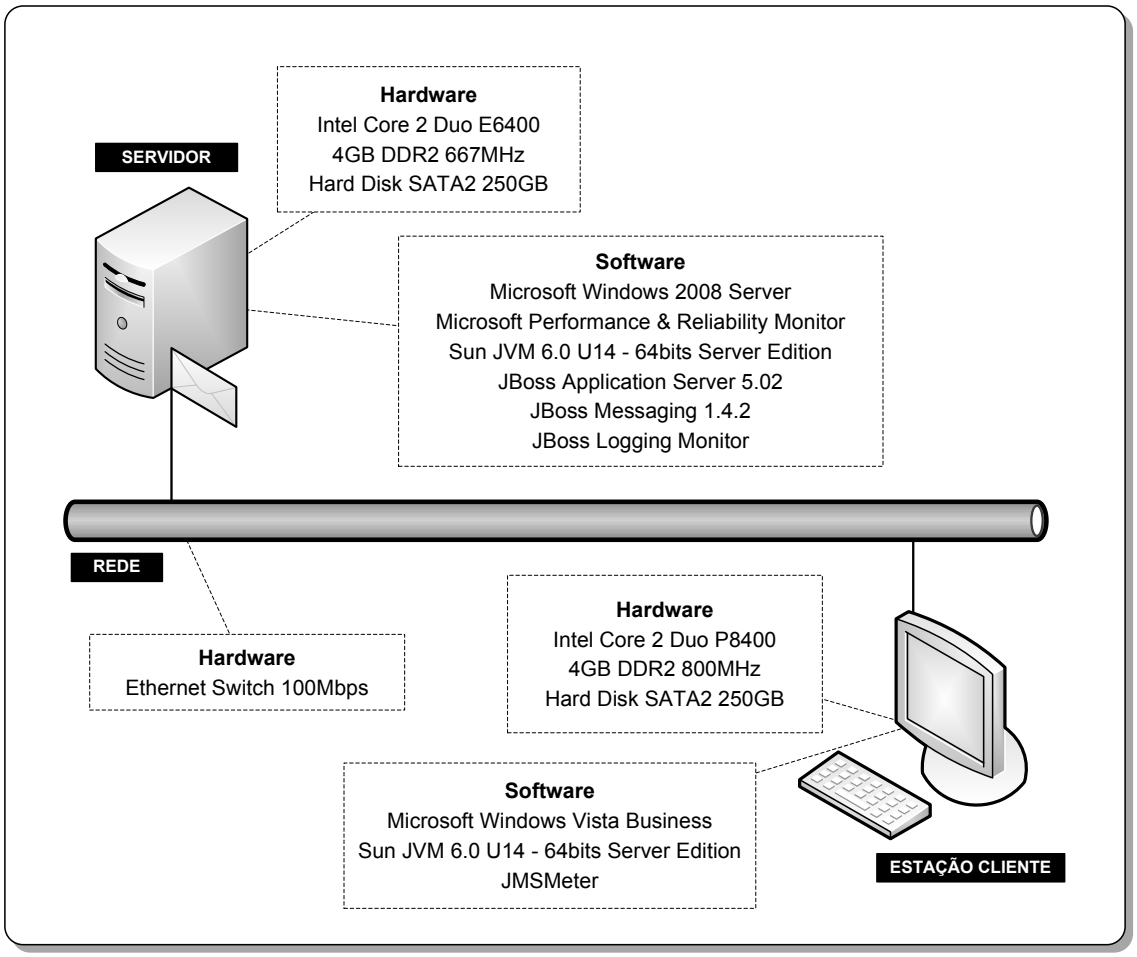


Figura C-7-22. Configurações do sistema real utilizado na validação.

A estação cliente possui configuração robusta para simular a execução de múltiplos produtores e consumidores, possibilitando a representação de cenários complexos. O gerador de carga artificial é o JMSMeter, executando sob uma JVM de 64bits.

O servidor é responsável pela execução do MOM, possuindo configuração adequada para este fim. O *provider* utilizado é desenvolvido em Java, e está executando numa JVM de 64 bits, garantindo alto desempenho ao sistema.

A rede é dedicada e isolada, suprimindo efeitos transientes externos ao ambiente, como tráfego *broadcast* de demais estações da rede. Ela possui vazão suficiente para não representar gargalo durante a execução do serviço de integração.

C.2.2 Parâmetros de Medição

A realização de medições de desempenho do SUT é realizada durante a parametrização dos modelos, e também para avaliação das métricas durante a atividade de validação. Para tanto, é necessário definir parâmetros de medição que guiam a realização desses procedimentos. É importante ressaltar que eles influenciam diretamente na qualidade estatística dos resultados.

Na parametrização, o JMSMeter realiza dois estágios de medição. Cada um deles segue o esquema apresentado na Figura C-7-23, composto de um período de *Ramp-Up*, seguido do período de medição, propriamente dito, e finalizando com um período de *Ramp-Down*. Esse esquema possibilita que fatores transientes sejam eliminados do período de medição, uma que no tempo de *ramp-up* o SUT é exercitado (*warm-up*) o suficiente para que valores estacionários sejam obtidos na medição.

Figura C-7-23. Definição de esquema para medição.

O primeiro estágio é destinado somente à produção de mensagens, enquanto que o segundo, ao consumo. Esse procedimento garante que o monitoramento capture de forma isolada as demandas de serviços no MOM para cada tipo de componente básico da carga de trabalho. A Tabela C-7-4 apresenta os valores para os parâmetros utilizados durante as parametrizações de todos os modelos.

Tabela C-7-4. Parâmetros de medição utilizados na parametrização.

Parâmetros	**Valores**
Número de produtores	1
Taxa de produção em *mps*	10.0
Número de consumidores	1
Taxa de recuperação de mensagens em *rps* (para experimentos com consumidores síncronos)	15
Tempo de *Ramp-Up* em *segundos*	180
Tempo de *Ramp-Down* em *segundos*	60
Tempo de Medição em *segundos*	600

Na validação, o JMSMeter realiza um único estágio de medição, pois neste momento o interesse não é pela apuração das demandas de serviço, e sim pela valoração das métricas do MOM. A Tabela C-7-5 define os parâmetros de medição utilizados durante o estágio de medição para todas as validações realizadas nos experimentos.

Tabela C-7-5. Parâmetros de medição utilizados na validação.

Parâmetros	**Valores**
Tempo de *Ramp-Up* em *segundos*	300
Tempo de *Ramp-Down* em *segundos*	60
Tempo de Medição em *segundos*	1.200

O fato dos períodos serem mais longo para a validação é motivado pela necessidade de uma maior precisão estatística dos valores das métricas, uma vez que nos experimentos nem sempre a taxa de produção é suficiente para exercitar o *provider* (*warm-up*) antes a medição.

C.2.3 Parâmetros de Simulação

A simulação de eventos discretos probabilísticos necessita de parâmetros que possibilitem orientar a qualidade estatística dos valores calculados para as métricas. A Tabela C-7-6 apresenta os valores para os parâmetros de simulação utilizados em todos os experimentos.

Tabela C-7-6. Parâmetros de simulação.

Parâmetros	Valores
Nível de confiança estatística	99%
Erro relativo máximo	1%
Método para análise estatística	Variância espectral

Os parâmetros definidos são fornecidos como entrada para o simulador, indicando, por exemplo, o momento de parada do algoritmo utilizado. As simulações realizadas capturam valores para o estado estacionário das métricas. Mais informações sobre simulação podem ser encontradas no Apêndice A (Seção A.1.3).

C.2.4 Fórmula para Cálculo das Métricas

O cálculo do valor das métricas de MOM durante a simulação dos modelos GSPN exige a captura de medidas do modelo (métricas PN), que são utilizadas como operandos nas fórmulas definidas nesta subseção. As métricas PN são apresentadas na Tabela C-7-7.

Tabela C-7-7. Metricas PN.

Métrica PN	Descrição
E_PRV_CPUs	Valor esperado de *tokens* no lugar *PRV_CPUs*, que representa o número de CPUs disponíveis no sistema.
E_DST1_InCPU_WPG1	Valor esperado de *tokens* no lugar *DST1_InCPU_WPG1*, que representa a quantidades de mensagens do grupo de produção *PG1* em processamento para escrita (*write*) no *destination.*
E_DST1_InCPU_RRPG1	Valor esperado de *tokens* no lugar *DST1_InCPU_RRPG1*, que representa a quantidades de mensagens do grupo de produção *PG1* em processamento para leitura e remoção (*read/remove*) no *destination.*
E_DST1_Delivered	Valor esperado de *tokens* no lugar *DST1_Delivered*, que representa a quantidade de mensagens que acabaram de ser entregues aos consumidores.
E_DST1_RTrash	Valor esperado de *tokens* no lugar *DST1_RTrash*, que representa a quantidade de chamadas do método *receive* sem êxito. Utilizada em

Métrica PN	Descrição
	consumidores síncronos com modelos de recuperação de mensagens do tipo *NoWait* e *TimeOut*.
E_DST1_Space	Valor esperado de *tokens* no lugar *DST_Space*, que representa o espaço livre no *destination*.
E_DST1_RepPG1	Valor esperado de *tokens* no lugar *DST1_RepPG1*, que representa a quantidade de mensagens armazenadas no *destination*, aguardando ser entregue ao consumidor.

A Tabela C-7-8 apresenta as fórmulas utilizadas para o cálculo das métricas de MOM a partir das métricas PN. As fórmulas utilizadas são baseadas nas leis fundamentais da análise operacional definidas no Apêndice A (Seção A.4).

Tabela C-7-8. Fórmulas para cálculo das métricas durante a simulação.

Métrica	Fórmula
Message Delivery Rate (mps)	*E_DST1_Delivered* * (1/*MeasurementTime*) * 1000
Receive Trash Rate (rps)	*E_DST1_RTrash* * (1/*MeasurementTime*) * 1000
CPU Utilization (%)	1 - ((2 - *E_DST1_InCPU_WPG1* - *E_DST1_InCPU_RRPG1*) / 2)
Destination Size (msg)	*DST1_Size* - *E_DST1_Space*
Latency (ms)	(1 / *MessageDeliveryRate* * 1000) * *DestinationSize*

O simulador ao calcular o valor médio para as métricas PN disponibiliza o intervalo de confiança associado. Esse intervalo é utilizado para efetuar a apuração do intervalo válido para as métricas de MOM.

C.3 Experimento 1: PTP-NPNT-M1024-JSC

O primeiro experimento tem o objetivo de validar os cenários que utilizam comunicação PTP, com nível de confiabilidade NPNT, mensagem de 1K e consumidor síncrono. Todos os cenários com essas características, em execução no SUT, podem utilizar a parametrização desenvolvida nesta seção.

Este experimento utiliza um cenário de carga definido pelos parâmetros apresentados na Tabela C-7-9.

Tabela C-7-9. Experimento 1: parâmetros do cenário de carga.

Parâmetros	Valores
Estilo de comunicação	PTP
Nível de confiabilidade do canal	NPNT
Tamanho da mensagem em *bytes*	1024

Parâmetros	Valores
Número esperado de mensagens (observações)	60.000
Número de produtores	50
Tipo dos produtores	JProd-Poisson
Taxa de produção em *mps*	1.0
Número de consumidores	4
Tipo dos consumidores	JSCons-NoWait-Poisson
Taxa de recuperação de mensagens em *rps*	15

C.3.1 Modelo GSPN

O modelo GSPN construído para implementar o modelo de carga de trabalho deste experimento é apresentado na Figura C-7-24.

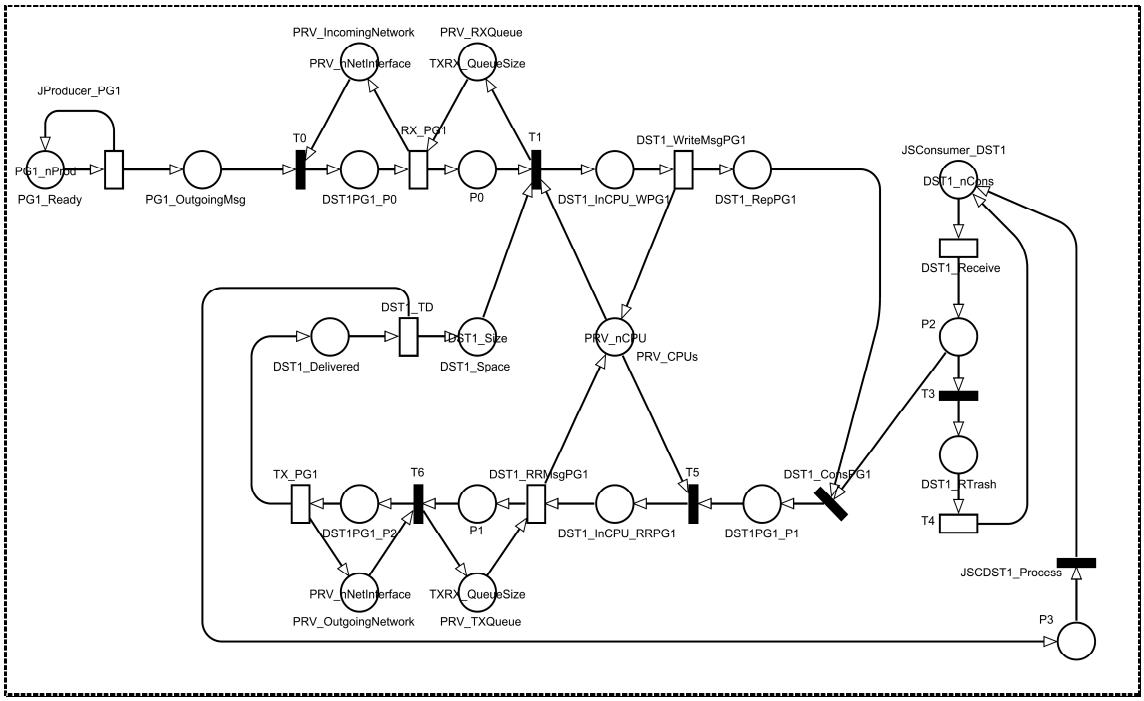


Figura C-7-24. Experimento 1: modelo GSPN.

C.3.2 Medição do SUT para Parametrização

Os resultados obtidos para as demandas de serviço (parametrização) através da medição do SUT são sumarizados na Tabela C-7-10. Também é calculado o valor médio das médias amostrais, e seus indicadores de variabilidade. Os resultados demonstram que houve pouca variabilidade entre as amostras.

Tabela C-7-10. Experimento 1: resultados da medição (parametrização).

Amostra	CPUTime_WritreMsg	CPUTime_RRMsg
P01	0,359078	0,527442
P02	0,345781	0,650549
P03	0,398858	0,582937
P04	0,420332	0,598154
P05	0,328888	0,550438
Média	**0,370587**	**0,581904**
Desvio Padrão	0,037949	0,047259
Variância	0,001440	0,002233
Coef. Variação	0,102401	0,081214

C.3.3 Avaliação via Simulação

Os resultados da simulação do Modelo GSPN parametrizado são apresentados na Tabela C-7-11. O simulador disponibiliza além do valor médio, o seu intervalo de confiança.

Tabela C-7-11. Experimento 1: resultados da simulação (métricas PN).

Métricas PN	Média	Intervalo de Confiança
E_PRV_CPUs	1,9636E+00	[1,9483E+00 , 1,9788E+00]
E_DST1_InCPU_WPG1	1,8882E-02	[1,8712E-02 , 1,9053E-02]
E_DST1_InCPU_RRPG1	2,9300E-02	[2,9053E-02 , 2,9546E-02]
E_DST1_Delivered	5,0797E-08	[5,0522E-08 , 5,1071E-08]
E_DST1_RTrash	9,5252E-09	[9,4388E-09 , 9,6115E-09]
E_DST1_Space	9,9699E+02	[9,8706E+02 , 1,0069E+03]
E_DST1_RepPG1	5,2038E+00	[5,1571E+00 , 5,2505E+00]

Com o valor das métricas PN e utilizando as fórmulas definidas na Tabela C-7-8, os valores calculados para as métricas do MOM são mostradas na Tabela C-7-13. Para cada métrica é apresentado o intervalo válido, indicando o limite inferior, valor médio e limite superior.

Tabela C-7-12. Experimento 1: resultados da simulação (métricas de MOM).

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Limite Inferior	50,52	9,44	2,39%	0,00	0,00
Valor Médio	50,80	9,53	2,41%	3,01	59,26
Limite Superior	51,07	9,61	2,43%	12,94	253,40

C.3.4 Medição do SUT para Validação

Os resultados obtidos para as métricas de MOM através da medição do SUT são sumarizadas na Tabela C-7-13. Para cada amostra da medição são apresentados os valores médios calculados de cada métrica. Além disso, é calculada a média das médias amostrais, assim com os indicadores de variabilidade. Os resultados demonstram uma variabilidade muito pequena, indicando a boa qualidade da medição.

Tabela C-7-13. Experimento 1: resultados da medição (validação).

Amostra	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
M01	49,85	10,44	2,02%	14,75	299,87
M02	49,91	10,01	2,14%	15,43	314,54
M03	50,22	9,68	2,23%	16,71	334,99
M04	50,31	9,23	2,16%	16,40	331,32
M05	50,19	9,79	2,26%	15,45	318,09
Média	**50,09**	**9,83**	**2,16%**	**15,75**	**319,76**
Desvio Padrão	0,20	0,44	0,09%	0,79	14,07
Variância	0,04	0,20	0,00%	0,63	197,85
Coef. Variação	0,00404	0,04517	0,04290	0,05043	0,04399

C.3.5 Validação

Os resultados do estudo comparativo entre simulação e medição, apresentados na Tabela C-7-14, indicam que as métricas *Message Delivery Rate*, *Receive Trash Rate* e *CPU Utilization* obtiveram excelente aproximação, validando esse modelo GSPN para representar de forma satisfatória o SUT.

Tabela C-7-14. Experimento 1: estudo comparativo entre medição e simulação.

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Medição (Média)	50,09	9,83	2,16%	15,75	319,76
Margem de 30%	[35,07 , 65,12]	[6,88 , 12,78]	[1,51 , 2,81]	[11,02 , 20,47]	[223,83 , 415,69]
Simulação (Média)	50,80	9,53	2,41%	3,01	59,26
Int. de Confiança	[50,52 , 51,07]	[9,44 , 9,61]	[2,39 , 2,43]	[0 , 12,94]	[0 , 253,4]
Erro % (Sim/Med)	**1,40%**	**-3,10%**	**11,33%**	**-80,89%**	**-81,47%**

A métrica *Destination Size* obteve um resultado razoável, pois o intervalo de confiança obtido pela simulação [0, 12,94] está apenas parcialmente compreendido dentro da margem aceitável de 30% (ver Seção 2.5.6). Essa imprecisão é repassada para a métrica *Latency*, uma vez que ela é calculada em função da *Destination Size*.

Considerando que todas as métricas obtiveram resultados da simulação dentro da margem de tolerância quando comparados aos valores obtidos via medição, o modelo de cenário PTP-NPNT-M1024-JSC está validado.

C.4 Experimento 2: PTP-PNT-M10240-JSC

O segundo experimento tem o objetivo de validar os cenários que utilizam comunicação PTP, com nível de confiabilidade PNT, mensagem de 10K e consumidor síncrono. Todos os cenários com essas características, em execução no SUT, podem utilizar a parametrização desenvolvida nesta seção.

Este experimento utiliza um cenário de carga definido pelos parâmetros apresentados na Tabela C-7-15. A principal diferença desse cenário em relação ao do primeiro experimento é a persitência das mensagens. Além disso, o volume de mensagens submetidas ao MOM é bastante superior, exigindo mais CPU, com isso reforçando a qualidade das parametrizações na captura das demandas desse importante recurso.

Tabela C-7-15. Experimento 2: parâmetros do cenário de carga.

Parâmetros	Valores
Estilo de comunicação	PTP
Nível de confiabilidade do canal	PNT
Tamanho da mensagem em *bytes*	10240
Número esperado de mensagens (observações)	216.000
Número de produtores	180
Tipo dos produtores	JProd-Poisson
Taxa de produção em *mps*	1.0
Número de consumidores	8
Tipo dos consumidores	JSCons-NoWait-Poisson
Taxa de recuperação de mensagens em *rps*	30

C.4.1 Modelo GSPN

O modelo GSPN construído para implementar o modelo de carga de trabalho deste experimento é apresentado na Figura C-7-25.

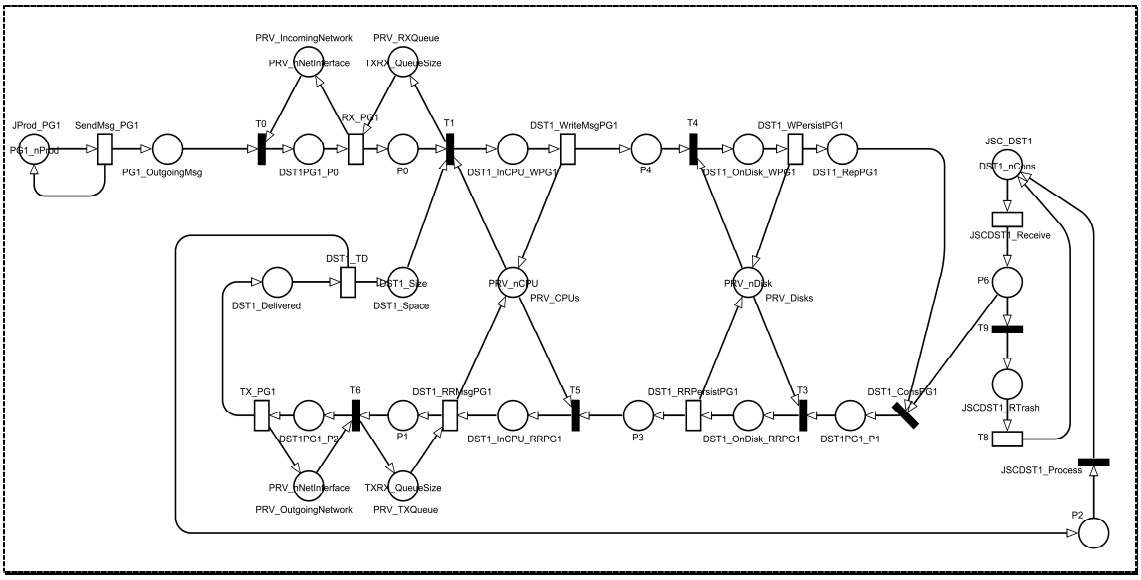


Figura C-7-25. Experimento 2: modelo GSPN.

C.4.2 Medição do SUT para Parametrização

Os resultados obtidos para as demandas de serviço (parametrização) através da medição do SUT são sumarizados na Tabela C-7-16. A média das médias amostrais demonstra pouca variabilidade, caracterizando a qualidade das amostras.

Tabela C-7-16. Experimento 2: resultados da medição (parametrização).

Amostra	CPUTime_WriteMsg	DiskTime_WritePersist	CPUTime_RRMsg	DiskTime_RRPersist
P01	1,736268	0,070900	1,060204	0,014383
P02	1,749774	0,074794	1,193225	0,013329
P03	1,747103	0,078573	0,959961	0,011518
P04	1,774105	0,067232	1,026515	0,015082
P05	1,807791	0,075131	1,044621	0,008811
Média	**1,763008**	**0,073326**	**1,056905**	**0,012624**
Desvio Padrão	0,028600	0,004358	0,085244	0,002520
Variância	0,000818	0,000019	0,007267	0,000006
Coef. Variação	0,016222	0,059431	0,080655	0,199599

C.4.3 Avaliação via Simulação

Os resultados da simulação do Modelo GSPN parametrizado são apresentados na Tabela C-7-17, com o valor médio e o seu intervalo de confiança.

Tabela C-7-17. Experimento 2: resultados da simulação (métricas PN).

Métricas PN	Média	Intervalo de Confiança
E_PRV_CPUs	1,5114E+00	[1,4987E+00 , 1,5240E+00]
E_DST1_InCPU_WPG1	3,2036E-01	[3,1752E-01 , 3,2320E-01]
E_DST1_InCPU_RRPG1	1,9096E-01	[1,8927E-01 , 1,9266E-01]
E_DST1_Delivered	1,8014E-07	[1,7849E-07 , 1,8178E-07]
E_DST1_RTrash	5,3915E-08	[5,3535E-08 , 5,4294E-08]
E_DST1_Space	9,7242E+01	[9,6888E+01 , 9,7596E+01]
E_DST1_RepPG1	3,3643E+00	[3,3434E+00 , 3,3853E+00]

Com o valor das métricas PN e utilizando as fórmulas definidas na Tabela C-7-8, os valores calculados para as métricas do MOM são mostradas na Tabela C-7-18, incluindo o valor médio e o intervalo de confiança obtido utilizando os limites definidos pelas métricas PN.

Tabela C-7-18. Experimento 2: resultados da simulação (métricas de MOM).

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Limite Inferior	178,49	53,54	25,34%	2,40	13,47
Valor Médio	180,14	53,91	25,57%	2,76	15,31
Limite Superior	181,78	54,29	25,79%	3,11	17,12

C.4.4 Medição do SUT para Validação

Os resultados obtidos para as métricas de MOM através da medição do SUT são sumarizadas na Tabela C-7-19, incluindo o valor médio para cada amostra, a média das médias amostrais e sua variabilidade. Os resultados demonstram uma alta representatividade do valor médio.

Tabela C-7-19. Experimento 2: resultados da medição (validação).

Amostra	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
M01	179,85	53,42	25,33%	21,28	127,33
M02	180,23	53,68	27,02%	20,82	125,66
M03	180,10	51,83	25,39%	20,72	129,16
M04	179,71	53,55	27,57%	21,35	125,22
M05	179,60	53,36	26,53%	20,35	124,14
Média	**179,90**	**53,17**	**26,37%**	**20,90**	**126,30**
Desvio Padrão	0,26	0,76	0,99%	0,41	1,97
Variância	0,07	0,57	0,01%	0,17	3,87
Coef. Variação	0,00146	0,01426	0,03766	0,01985	0,01557

C.4.5 Validação

Os resultados do estudo comparativo entre simulação e medição, apresentados na Tabela C-7-20, indicam que as métricas *Message Delivery Rate*, *Receive Trash Rate* e *CPU Utilization* obtiveram excelente aproximação, validando esse modelo GSPN para representar de forma satisfatória o SUT.

Considerando que esse cenário de carga possui uma maior complexidade que no primeiro experimento, com uma carga submetida bem maior (número de produtores, tamanho da mensagem, número de consumidores, taxa de recuperação de mesangens), os resultados mais aproximados para essas métricas demonstram a capacidade de representação dos modelos.

Tabela C-7-20. Experimento 2: estudo comparativo entre medição e simulação.

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Medição (Média)	179,90	53,17	26,37%	20,90	126,30
Margem de 30%	[125,93 , 233,87]	[37,22 , 69,12]	[18,46 , 34,28]	[14,63 , 27,17]	[88,41 , 164,19]
Simulação (Média)	180,14	53,91	25,57%	2,76	15,31
Int. de Confiança	[178,49 , 181,78]	[53,54 , 54,29]	[25,34 , 25,79]	[2,4 , 3,11]	[13,47 , 17,12]
Erro % (Sim/Med)	**0,13%**	**1,40%**	**-3,04%**	**-86,80%**	**-87,88%**

A métrica *Destination Size* obteve resultado insatisfatório na validação, os valores apresentam uma distorção substancial. Esse comportamento foi percebido no primeiro experimento, e provavelmente com o aumento da carga se intensificou. Essa imprecisão é repassada para a métrica *Latency* assim como no primeiro experimento.

Tentativas de tratamento dos dados da parametrização não obtiveram sucesso na solução dessa distorção. Na verdade, um retardo de 5.3ms adicionado serialmente a cada mensagem, formando uma nova fila faz com que os números da simulação se aproximem dos números da medição, tornando o modelo válido também para as métricas *Destination Size* e *Latency*. Porém essa alternativa precisa ainda ser generalizada para ser considerada válida.

Considerando que as métricas *Message Delivery Rate*, *Receive Trash Rate* e *CPU Utilization* obtiveram resultados da simulação dentro da margem de tolerância quando comparados aos valores obtidos via medição, o modelo de cenário PTP-PNT-M10240-JSC está validado para estas métricas. Quanto às demais métricas, o modelo não pode ser considerado validado, até que novos estudos em relação ao retardo adicional sejam finalizados.

C.5 Experimento 3: PTP-PNT-M1024-JSC

O terceiro experimento tem o objetivo de validar os cenários que utilizam comunicação PTP, com nível de confiabilidade PNT, mensagem de 1K e consumidor síncrono. Este experimento se diferencia do segundo pelo tamanho da mensagem, possibilitando a avaliação do seu impacto no desempenho. A Tabela C-7-21 apresenta os parâmetros que definem este cenário de carga.

Todos os cenários com essas características, em execução no SUT, podem utilizar a parametrização desenvolvida nesta seção. Um exemplo disso é a interação *Registro de Vendas*, no Capítulo 5, que se utiliza deste cenário de carga.

Tabela C-7-21. Experimento 3: parâmetros do cenário de carga.

Parâmetros	**Valores**
Estilo de comunicação	PTP
Nível de confiabilidade do canal	PNT
Tamanho da mensagem em *bytes*	1024
Número esperado de mensagens (observações)	216.000
Número de produtores	180
Tipo dos produtores	JProd-Poisson
Taxa de produção em *mps*	1.0
Número de consumidores	8
Tipo dos consumidores	JSCons-NoWait-Poisson
Taxa de recuperação de mensagens em *rps*	30

C.5.1 Modelo GSPN

O modelo GSPN construído para implementar o modelo de carga de trabalho deste experimento é apresentado na Figura C-7-26.

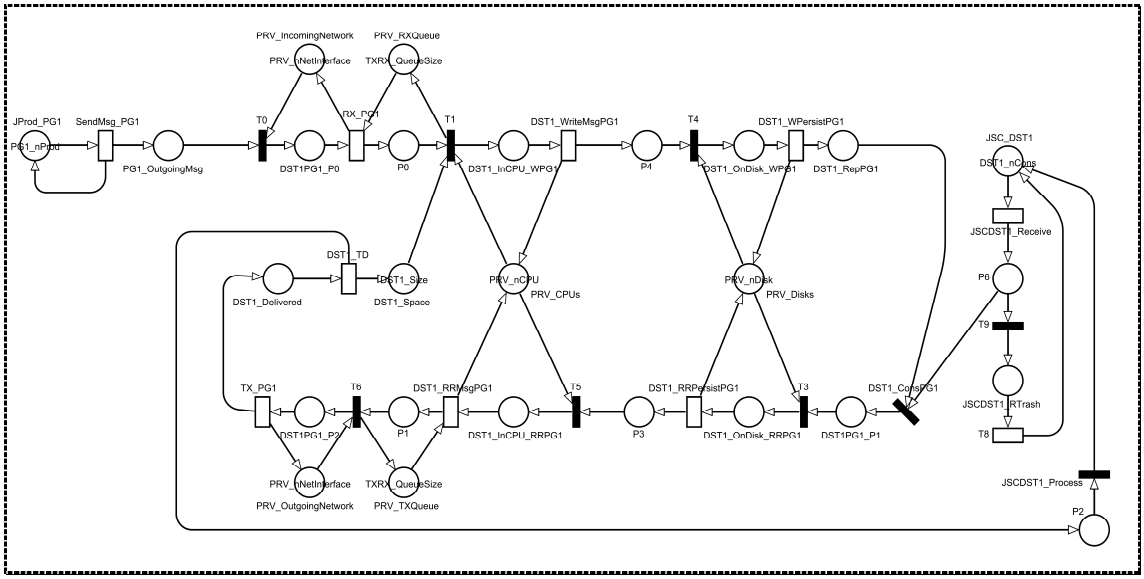


Figura C-7-26. Experimento 3: modelo GSPN.

C.5.2 Medição do SUT para Parametrização

Os resultados obtidos para as demandas de serviço (parametrização) através da medição do SUT são sumarizados na Tabela C-7-22. Pode-se perceber que as demandas diminuíram para praticamente todos os recursos se comparado com o segundo experimento (ver Tabela C-7-16). Isso era esperado uma vez que o tamanho da mensagem menor exige menos esforço do MOM, principalmente em um cenário com persistência.

Tabela C-7-22. Experimento 3: resultados da medição (parametrização).

Amostra	CPUTime_WriteMsg	DiskTime_WritePersist	CPUTime_RRMsg	DiskTime_RRPersist
P01	1,254938	0,023402	0,949626	0,024491
P02	1,321484	0,026451	1,084476	0,018058
P03	1,124904	0,025661	0,978329	0,016494
P04	1,207077	0,046891	0,868248	0,014728
P05	1,162500	0,022977	0,938786	0,016819
Média	**1,214181**	**0,029077**	**0,963893**	**0,018118**
Desvio Padrão	0,077244	0,010066	0,078630	0,003756
Variância	0,005967	0,000101	0,006183	0,000014
Coef. Variação	0,063619	0,346201	0,081575	0,207307

A única exceção é o tempo de leitura e remoção do disco, que praticamente ficou inalterado. Isso pode ser explicado devido a otimizações do JBoss Messaging que acentua a utilização de *cache* de memória para as filas persistidas. Na gravação ele precisa armazenar no disco, porém, na leitura utiliza apenas a memória.

C.5.3 Avaliação via Simulação

Os resultados da simulação do Modelo GSPN parametrizado são apresentados na Tabela C-7-23.

Tabela C-7-23. Experimento 3: resultados da simulação (métricas PN).

Métricas PN	Média	Intervalo de Confiança
E_PRV_CPUs	1,6158E+00	[1,6029E+00 , 1,6288E+00]
E_DST1_InCPU_WPG1	2,2029E-01	[2,1827E-01 , 2,2231E-01]
E_DST1_InCPU_RRPG1	1,7425E-01	[1,7294E-01 , 1,7557E-01]
E_DST1_Delivered	1,8002E-07	[1,7902E-07 , 1,8102E-07]
E_DST1_RTrash	5,3789E-08	[5,3275E-08 , 5,4302E-08]
E_DST1_Space	9,6278E+01	[9,5443E+01 , 9,7112E+01]
E_DST1_RepPG1	3,3085E+00	[3,2859E+00 , 3,3312E+00]

Com o valor das métricas PN e utilizando as fórmulas definidas na Tabela C-7-8, os valores calculados para as métricas do MOM são mostradas na Tabela C-7-24.

Tabela C-7-24. Experimento 3: resultados da simulação (métricas de MOM).

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Limite Inferior	179,02	53,28	19,56%	2,89	16,13
Valor Médio	180,02	53,79	19,73%	3,72	20,68
Limite Superior	181,02	54,30	19,89%	4,56	25,17

Esses resultados quando comparados aos do segundo experimento (ver Tabela C-7-18) apresentam uma redução do consumo de CPU como era de ser esperar, e a manutenção da taxa de entrega de mensagens, uma que a carga em termos de quantidade de mensagens continua igual. Porém, considerando que no segundo experimento o tamanho da mensagem é dez vezes maior, chega-se a conclusão que com 5% a mais de CPU consegue-se trafegar dez vezes mais informação.

Essa conclusão é utilizada no Capítulo 5 como alternativa para a interação de *Registro de Vendas*, permitindo que o aumento do volume de vendas seja tratável pelo MOM (ver Seção 5.8.2).

C.5.4 Medição do SUT para Validação

Os resultados obtidos para as métricas de MOM através da medição do SUT são sumarizadas na Tabela C-7-25. Os resultados da medição quando comparados com os do segundo experimento (ver Tabela C-7-19) mantém o mesmo comportamento da simulação.

Tabela C-7-25. Experimento 3: resultados da medição (validação).

Amostra	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
M01	180,10	58,92	15,09%	19,81	110,40
M02	179,72	60,65	15,77%	19,83	107,66
M03	179,98	59,83	14,48%	19,88	108,16
M04	179,71	60,23	15,61%	19,05	107,62
M05	180,99	58,76	15,26%	19,34	110,97
Média	**180,10**	**59,68**	**15,24%**	**19,58**	**108,96**
Desvio Padrão	0,53	0,82	0,50%	0,37	1,60
Variância	0,28	0,67	0,00%	0,14	2,57
Coef. Variação	0,00292	0,01375	0,03302	0,01886	0,01470

C.5.5 Validação

Os resultados do estudo comparativo entre simulação e medição, apresentados na Tabela C-7-26, indicam mais uma vez que as métricas *Message Delivery Rate*, *Receive Trash Rate* e *CPU Utilization* obtiveram aproximação dentro da margem de 30%, validando esse modelo GSPN para representar de forma satisfatória o SUT.

Tabela C-7-26. Experimento 3: estudo comparativo entre medição e simulação.

	DeliveryRate (mps)	RcvTrashRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Medição (Média)	180,10	59,68	15,24%	19,58	108,96
Margem de 30%	[126,07 , 234,13]	[41,77 , 77,58]	[10,67 , 19,81]	[13,71 , 25,46]	[76,27 , 141,65]
Simulação (Média)	180,02	53,79	19,73%	3,72	20,68
Int. de Confiança	[179,02 , 181,02]	[53,28 , 54,3]	[19,56 , 19,89]	[2,89 , 4,56]	[16,13 , 25,17]
Erro % (Sim/Med)	**-0,04%**	**-9,87%**	**29,43%**	**-80,99%**	**-81,02%**

A métrica *Destination Size* mais uma vez obteve resultado insatisfatório, assim como no segundo experimento. Essa imprecisão é repassada para a métrica *Latency*.

Considerando que as métricas *Message Delivery Rate*, *Receive Trash Rate* e *CPU Utilization* obtiveram resultados satisfatórios, o modelo de PTP-PNT-M10240-JSC está validado para estas métricas. Quanto às demais métricas, o modelo não atende dentro da margem de erro aceitável para estudos de planejamento de capacidade.

C.6 Experimento 4: PTP-NPNT-M1024-JASC

Este experimento tem o objetivo de validar os cenários que utilizam comunicação PTP, com nível de confiabilidade NPNT, mensagem de 1K e consumidor assíncrono. Ele se diferencia em relação ao primeiro experimento pela característica assíncronca do consumidor e pela quantidade de produtores que aumenta de 50 (no primeiro experimento) para 120.

A Tabela C-7-27 apresenta os parâmetros que definem este cenário de carga. Todos os cenários com essas características, em execução no SUT, podem utilizar a parametrização desenvolvida nesta seção.

Tabela C-7-27. Experimento 4: parâmetros do cenário de carga.

Parâmetros	Valores
Estilo de comunicação	PTP
Nível de confiabilidade do canal	NPNT
Tamanho da mensagem em *bytes*	1024
Número esperado de mensagens (observações)	144.000
Número de produtores	120
Tipo dos produtores	JProd-Poisson
Taxa de produção em *mps*	1.0
Número de consumidores	2
Tipo dos consumidores	JASCons

C.6.1 Modelo GSPN

O modelo GSPN construído para implementar o modelo de carga de trabalho deste experimento é apresentado na Figura C-7-27.

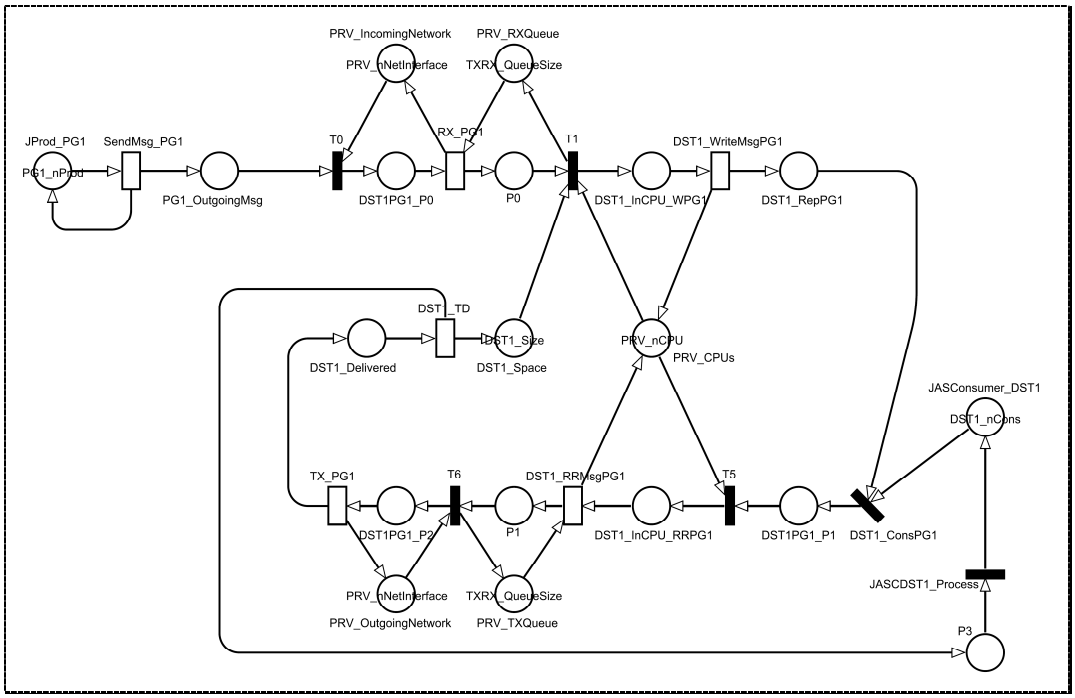


Figura C-7-27. Experimento 4: modelo GSPN.

C.6.2 Medição do SUT para Parametrização

Os resultados obtidos para as demandas de serviço (parametrização) através da medição do SUT são sumarizados na Tabela C-7-28. Apesar desse cenário possuir nível de confiabilidade NPNT, o que indica a não utilização da demanda de disco para parametrizar o sistema, o gasto deste recurso é contabilizado para comprovar a sua pouca expressividade, no contexto total.

Tabela C-7-28. Experimento 4: resultados da medição (parametrização).

Amostra	CPUTime_WriteMsg	DiskTime_WritePersist	CPUTime_RRMsg	DiskTime_RRPersist
P01	0,349714	0,002145	0,541319	0,012031
P02	0,324889	0,000534	0,499183	0,000313
P03	0,315245	0,004285	0,652965	0,000223
P04	0,252965	0,000505	0,630133	0,000319
P05	0,361069	0,002475	0,644849	0,000296
Média	**0,320776**	**0,001989**	**0,593690**	**0,002636**
Desvio Padrão	0,042151	0,001570	0,069161	0,005252
Variância	0,001777	0,000002	0,004783	0,000028
Coef. Variação	0,131403	0,789212	0,116494	1,991956

Esses resultados quando comparados aos do primeiro experimento (ver Tabela C-7-10 demonstram que o tipo do consumidor (síncrono ou assíncrono) afeta muito pouco a demanda de serviço no MOM.

C.6.3 Avaliação via Simulação

Os resultados da simulação do Modelo GSPN parametrizado são apresentados na Tabela C-7-29.

Tabela C-7-29. Experimento 4: resultados da simulação (métricas PN).

Métricas PN	Média	Intervalo de Confiança
E_PRV_CPUs	1,8952E+00	[1,8816E+00 , 1,9087E+00]
E_DST1_InCPU_WPG1	3,8632E-02	[3,8397E-02 , 3,8867E-02]
E_DST1_InCPU_RRPG1	7,0972E-02	[7,0336E-02 , 7,1609E-02]
E_DST1_Delivered	1,2030E-07	[1,1961E-07 , 1,2098E-07]
E_DST1_Space	9,8964E+00	[9,8246E+00 , 9,9681E+00]
E_DST1_RepPG1	6,1338E-05	[6,0845E-05 , 6,1832E-05]

Utilizando as fórmulas definidas na Tabela C-7-8, os valores calculados para as métricas do MOM são mostradas na Tabela C-7-30. Esses resultados demonstram um consumo maior de CPU quando comparados ao do primeiro experimento (ver Tabela C-7-12), uma vez que neste cenário a carga submetida é maior, devido ao número de produtores. O tamanho da fila (*Destination Size*) teve uma redução comprovando a melhor eficiência do modelo assíncrono versus o síncrono.

Tabela C-7-30. Experimento 4: resultados da simulação (métricas de MOM).

	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Limite Inferior	119,61	5,44%	0,03	0,27
Valor Médio	120,30	5,48%	0,10	0,86
Limite Superior	120,98	5,52%	0,18	1,45

C.6.4 Medição do SUT para Validação

Os resultados obtidos para as métricas de MOM através da medição do SUT são sumarizadas na Tabela C-7-31. As amostras *M01*, *M04* e *M05* apresentaram problemas durante a medição e foram expurgadas dos resultados. Para contornar o problema, e demonstrar tratarem-se apenas de imperícia na preparação da medição, duas outras amostras foram geradas (*M06* e *M07*).

A média das médias amostrais apresentou um valor com pouca variabilidade, demonstrando que essas amostras possuem um comportamento equivalente e reprodutível.

Esses resultados quando comparados com os do primeiro experimento (ver Tabela C-7-13) comprovam o comportamento que o modelo GSPN já apresenta, quanto ao tamanho da fila.

Tabela C-7-31. Experimento 4: resultados da medição (validação).

Amostra	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
M02	120,02	4,02%	0,78	10,78
M03	119,93	4,05%	0,83	10,69
M06	119,51	4,72%	0,76	10,63
M07	119,64	4,22%	0,80	10,63
Média	**119,77**	**4,25%**	**0,79**	**10,68**
Desvio Padrão	0,24	0,33%	0,03	0,07
Variância	0,06	0,00%	0,00	0,00
Coef. Variação	0,00198	0,07666	0,03600	0,00651

C.6.5 Validação

Os resultados do estudo comparativo entre simulação e medição, apresentados na Tabela C-7-32, indicam que o modelo GSPN para as métricas *Message Delivery Rate* e *CPU Utilization* pode ser considerado validado.

Tabela C-7-32. Experimento 4: estudo comparativo entre medição e simulação.

	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latency (ms)
Medição (Média)	119,77	4,25%	0,79	10,68
Margem de 30%	[83,84 , 155,71]	[2,98 , 5,53]	[0,55 , 1,03]	[7,48 , 13,89]
Simulação (Média)	120,30	5,48%	0,10	0,86
Int. de Confiança	[119,61 , 120,98]	[5,44 , 5,52]	[0,03 , 0,18]	[0,27 , 1,45]
Erro % (Sim/Med)	**0,44%**	**28,85%**	**-86,93%**	**-91,93%**

Apesar da métrica *Destination Size* continuar apresentando um erro elevado entre os valores medidos e simulados (repassando o erro para a *Latency*), o comportamento do modelo GSPN está compatível com os resultados apresentados na medição. Tanto o modelo quanto a medição reduziram o tamanho da fila com a utilização de consumidores assíncronos.

C.7 Experimento 5: PS-PNT-M10240

O quinto e último experimento apresenta um cenário com comunicação *Pub/Sub*, nível de confiabilidade do canal PNT e mensagem de 10K. Todos os cenários com essas características, em execução no SUT, podem utilizar a parametrização desenvolvida nesta seção.

Este tipo de cenário apresenta dificuldades inerentes ao mecanismo *Pub/Sub*, como no caso da parametrização, pois o isolamento do consumo de recursos por tipo de mensagem (componente básico de carga) é bastante complexa. A Tabela C-7-33 apresenta os parâmetros que definem este cenário de carga.

Tabela C-7-33. Experimento 5: parâmetros do cenário de carga.

Parâmetros	Valores
Estilo de comunicação	*Pub/Sub*
Nível de confiabilidade do canal	PNT
Tamanho da mensagem em *bytes*	10240
Número esperado de mensagens (observações)	1.200 (produzidas) 144.000 (consumidas)
Número de produtores	1
Tipo dos produtores	JProd-Poisson-Sync
Taxa de produção em *mps*	1.0
Número de consumidores	120
Tipo dos consumidores	JASCons
Modelo de assinatura	durable

C.7.1 Modelo GSPN

O modelo GSPN construído para implementar o modelo de carga de trabalho deste experimento é apresentado na Figura C-7-28.

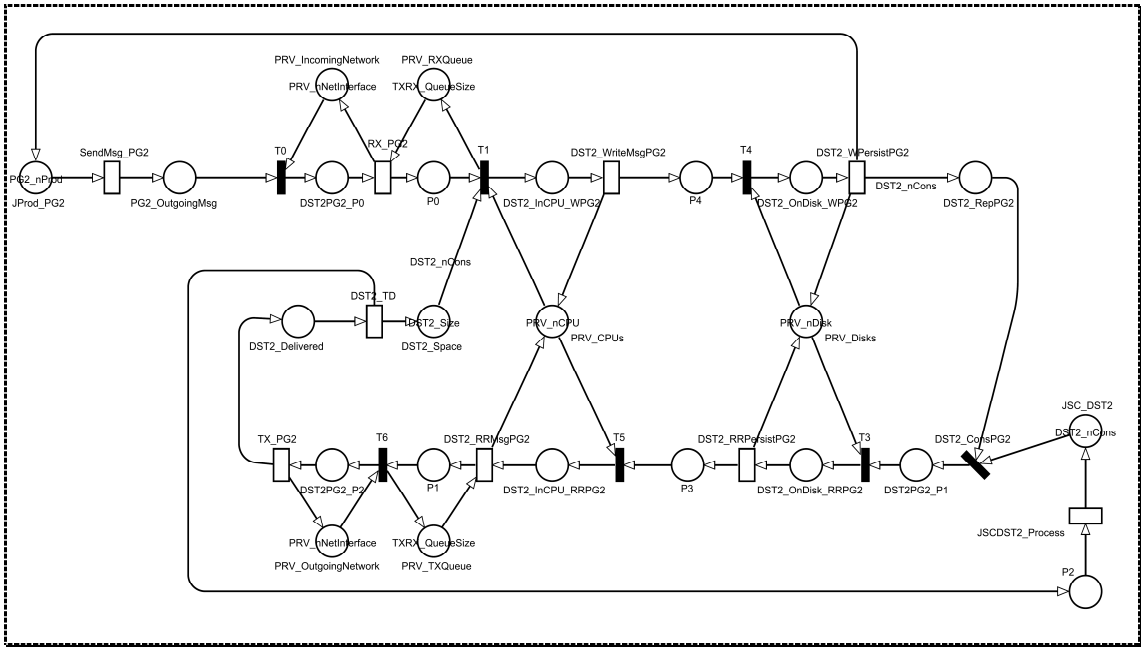


Figura C-7-28. Experimento 5: modelo GSPN.

C.7.2 Medição do SUT para Parametrização

Os resultados obtidos para as demandas de serviço (parametrização) através da medição do SUT são sumarizados na Tabela C-7-34. Os valores demonstram

mais uma vez que o nível de confiabilidade PNT consome muito mais recursos de CPU na recepção das mensagens por parte do MOM, do que no encaminhamento delas para os consumidores.

Tabela C-7-34. Experimento 5: resultados da medição (parametrização).

Amostra	CPUTime_WriteMsg	DiskTime_WritePersist	CPUTime_RRMsg	DiskTime_RRPersist
P01	1,690993	0,210615	1,560267	0,022443
P02	1,792156	0,361972	1,385266	0,075357
P03	1,661017	0,257153	1,667833	0,095256
P04	1,909248	0,254579	1,662177	0,024224
Média	**1,763354**	**0,271080**	**1,568886**	**0,054320**
Desvio	0,112285	0,064248	0,132015	0,036698
Variância	0,012608	0,004128	0,017428	0,001347
Coef. Variação	0,063677	0,237009	0,084146	0,675589

C.7.3 Avaliação via Simulação

Os resultados da simulação do Modelo GSPN parametrizado são apresentados na Tabela C-7-35. Apenas as métricas PN que estão em uso nesse cenário são medidas, para aumentar a velocidade da execução da simulação.

Tabela C-7-35. Experimento 5: resultados da simulação (métricas PN).

Métricas PN	Média	Intervalo de Confiança
E_DST1_InCPU_WPG1	1,7613E-04	[1,7462E-04 , 1,7764E-04]
E_DST1_InCPU_RRPG1	2,0408E-02	[2,0281E-02 , 2,0535E-02]
E_DST1_Delivered	1,1841E-08	[1,1769E-08 , 1,1912E-08]
E_DST1_Space	9,9762E+02	[9,8900E+02 , 1,0062E+03]

Com o valor das métricas PN e utilizando as fórmulas definidas na Tabela C-7-8, os valores calculados para as métricas do MOM são apresentados na Tabela C-7-36. O valor médio é acompanhado pelo limite inferior e superior calculados através do intervalo de confiança das métricas PN.

Tabela C-7-36. Experimento 5: resultados da simulação (métricas de MOM).

	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latência (ms)
Limite Inferior	11,77	1,02%	0,00	0,00
Valor Médio	11,84	1,03%	2,38	200,86
Limite Superior	11,91	1,04%	11,00	923,80

C.7.4 Medição do SUT para Validação

Os resultados obtidos para as métricas de MOM através da medição do SUT são sumarizadas na Tabela C-7-37. A baixa variabilidade das medidas obtidas indicam uma boa qualidade do conjunto de amostras. A média das médias amostrais apresenta uma latência alta, comprovando a tendência de sistemas Pub/Sub apresentarem um retardo elevado na entrega de mensagens.

Um dos motivos para isso, é a serialização da entrega de cada mensagem para cada um dos assinantes do tópico (*destination*). Como o cenário possui 120 assinantes para um único tópico, os valores da latência são baixos para alguns

consumidores e alto para outros, dependendo da ordem de atendimento que os consumidores receberam do *provider*.

Tabela C-7-37. Experimento 5: resultados da medição (validação).

Amostra	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latência (ms)
M01	11,91	1,50%	29,89	516,49
M02	11,97	1,42%	30,45	529,72
M03	11,53	1,28%	31,55	532,12
M04	11,88	1,43%	31,90	558,91
M05	11,60	1,48%	27,16	567,81
Média	**11,78**	**1,42%**	**30,19**	**541,01**
Desvio	0,20	0,09%	1,88	21,48
Variância	0,04	0,00%	3,52	461,54
Coef. Variação	0,01687	0,06066	0,06218	0,03971

C.7.5 Validação

Os resultados do estudo comparativo entre simulação e medição, apresentados na Tabela C-7-38, indicam que as métricas *Message Delivery Rate* e *CPU Utilization* obtiveram resultados dentro da margem de 30% aceitável.

Tabela C-7-38. Experimento 5: estudo comparativo entre medição e simulação.

	DeliveryRate (mps)	CPU Utiliz. (%)	DestSize (msg)	Latência (ms)
Medição (Média)	11,78	1,42%	30,19	541,01
Margem de 30%	[8,24 , 15,31]	[1 , 1,85]	[21,13 , 39,25]	[378,71 , 703,31]
Simulação (Média)	11,84	1,03%	2,38	200,86
Int. de Confiança	[11,77 , 11,91]	[1,02 , 1,04]	[0 , 11]	[0 , 923,8]
Erro % (Sim/Med)	**0,53%**	**-27,63%**	**-92,12%**	**-62,87%**

A métrica *Destination Size* mais uma vez apresentou distorções que influeciam o resultado da métrica *Latency*. Porém dessa vez a latência obteve uma distorção menor que em outras oportunidades.

Diante dos números pode-se considerar validado o modelo GSPN para representar as métricas *Message Delivery Rate* e *CPU Utilization.*

C.8 Considerações Finais

Este apêndice realizou a validação da biblioteca de componentes de desempenho do JMSCapacity. Para isso, foi apresentado um conjunto de ferramentas utilizadas, assim como um conjunto de informações gerais sobre os experimentos.

Em seguida, cinco experimentos utilizando diversos configurações de carga demonstraram a atividade de parametrização e validação dos modelos. A maioria dos resultados demonstram a validade dessa biblioteca na representação do sistema real. Porém, a métrica *Destination Size* apresentou um erro elevado quando comparado os resultados da medição com a simulação, influenciando a valor da latência.

Isso indica que algum possível recurso compartilhado não está capturado pelos modelos. Novos estudos precisam ser realizados para comprovar essa tese e identificar qual recurso está faltando.

APÊNDICE

D

Notação do Processo

"Os últimos serão os primeiros"

Autor desconhecido.

A notação universal introduzida por [Kruchten 1998] para definição do processo unificado de desenvolvimento de software (Rational Unified Process, RUP) possui um formalismo adequado para a definição de processos, pois representa explicitamente (e graficamente) elementos importantes de um processo. Por este motivo essa notação é adotada para representação do processo de planejamento de capacidade do JMSCapacity.

Segundo essa notação a representação de um processo está fundamentada em quatro elementos básicos que apresentam *quem* vai fazer *o que*, *quando* e *como*:

- Papel ou Função: *quem*
- Atividade: *como*
- Artefatos: *o que*
- Fluxo de Trabalho: *quando*

Este apêndice tem o objetivo de descrever sinteticamente a notação utilizada no JMSCapacity.

D.1 Papel ou Função

O papel ou função é um conceito chave dentro de um processo, pois define o comportamento e as responsabilidades de um indivíduo ou grupo de indivíduos. O comportamento é definido pelas atividades desempenhadas pelos papéis, e cada papel está associado a um conjunto coeso de atividades.

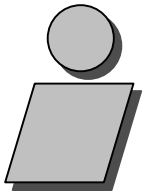


Figura D-7-29. Representação gráfica para papel.

Papéis não representam diretamente indivíduos, nem muito menos cargos. Um indivíduo pode desempenhar um ou mais papéis e um papel pode ser desempenhado por mais de um indivíduo. A Figura D-7-29 apresenta a representação gráfica utilizada para um papel.

É importante ressaltar que os papéis não identificam indivíduos, mas o seu comportamento e suas responsabilidades quando estão desempenhando cada papel. Apesar disto é comum expressar que "um *Avaliador* realizou uma atividade Y", quando na verdade o correto seria dizer "um indíviduo X desempenhando o papel de *Avaliador* realizou uma atividade Y".

D.2 Atividade

Os papéis desempenham atividades que definem o trabalho que eles realizam. Uma atividade é uma unidade de trabalho que um indíviduo atuando em um papel realiza produzindo resultados relevantes dentro do contexto do projeto. A atividade tem um objetivo claramente definido e usualmente é relacionado à produção ou atualização de um ou mais artefatos. Cada atividade é atribuída a um papel específico. A Figura D-7-30 apresenta a representação gráfica utilizada na metodologia para representar uma atividade.

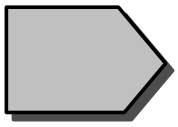


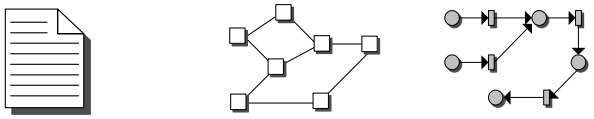
Figura D-7-30. Representação gráfica para atividade

Normalmente, as atividades podem ser descritas através de passos ou ações, que determinam como será realizada uma atividade. Essas atividades manipulam em geral apenas um ou um número reduzido de artefatos.

D.3 Artefato

Um artefato representa um conjunto de informações que é produzido, modificado ou utilizado por um processo. Os artefatos são produtos tangíveis em um projeto: resultados produzidos ou utilizados até a entrega do produto final.

Os artefatos são utilizados como insumos, pelos papéis, para realização das atividades, além de serem também resultados ou produtos das atividades. Alguns artefatos podem ser componentes de outros artefatos. A Figura D-7-31 apresenta algumas representações gráficas utilizadas no contexto do JMSCapacity para os artefatos.



(a) **(b)** **(c)**

Figura D-7-31. Representação gráfica para artefatos: (a) Relatórios (b) Modelos (c) Modelos em redes de Petri.

As responsabilidades de cada papel são usualmente representadas em relação aos artefatos que eles criam, consultam, modificam ou controlam. Nesse ponto é possível a definição dos artefatos que estão sob o controle de cada papel, definindo assim o escopo de suas responsabilidades.

D.4 Fluxo de Trabalho (*workflow*)

Um Fluxo de Trabalho é uma sequência de atividades que produz resultados. A representação gráfica desse fluxo conecta as atividades através, podendo ou não representar os artefatos de entrada e saída. Um possiiblidade é a utilização do Diagrama de Atividade do UML.

Dissertação de Mestrado apresentada por **Roberto Delgado Arteiro** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "JMSCAPACITY – Um *Toolkit* para Auxiliar no Planejamento de Capacidade de *Middlware* Orientado a Mensagem", orientada pelo **Prof. Nelson Souto Rosa** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Carlos André Guimarães Ferraz
Centro de Informática / UFPE

Prof. Glêdson Elias da Silveira
Departamento de Informática / UFPB

Prof. Nelson Souto Rosa
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 24 de agosto de 2009.

Prof. Nelson Souto Rosa
Vice-Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.