abnt-nbr10520=2002

Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pós-Graduação em Engenharia de Teleinformática

Avaliação de Desempenho de *Web Services* Orquestrados com BPEL4People

Henrique Jorge Amorim Holanda

Fortaleza – Ceará
Março 2012

Universidade Federal do Ceará
Departamento de Engenharia de Teleinformática
Programa de Pós-Graduação em Engenharia de Teleinformática

Avaliação de Desempenho de *Web Services* Orquestrados com BPEL4People

Autor
Henrique Jorge Amorim Holanda

Orientador
Giovanni Cordeiro Barroso
Co-orientador
Antônio de Barros Serra

Tese de Doutorado apresentada à Coordenação do Curso de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de ***Doutor em Engenharia de Teleinformática****.*

Fortaleza – Ceará
Março 2012

Henrique Jorge Amorim Holanda

Avaliação de Desempenho de *Web Services* Orquestrados com BPEL4People

Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia de Teleinformática e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará.

Nome do Aluno

Banca Examinadora:

Prof. Dr. Giovanni Cordeiro Barroso
(Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. Antônio de Barros Serra
(Co-orientador)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Francisco Milton Mendes Neto
Universidade Federal Rural do Semi-Árido - UFERSA

Prof. Dr. Pedro Fernandes Ribeiro Neto
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Rossana Maria de Castro Andrade
Universidade Federal do Ceará - UFC

Prof. Dr. Prof. Dr. José Marques Soares
Universidade Federal do Ceará - UFC

Fortaleza, 01 de março de 2012

Resumo

Web Services (WS) são pilares para a construção de aplicações orientadas a serviços. Uma série de linguagens para a composição de serviços web têm sido propostas, sendo formado um consenso em torno da linguagem de execução de processos de negócio (BPEL). BPEL centra-se em processos de negócio que orquestram interações de WS. No entanto, em geral, processos de negócio são compostos por um amplo espectro de atividades que exigem muitas vezes a participação humana para executar tarefas, rever ou aprovar medidas e inserir dados. Essas interações humanas são abordadas em uma nova especificação do BPEL denominada de BPEL4People. BPEL4People introduz a atividade humana para BPEL. Com o uso do BPEL4People, modelos formais (como as redes de Petri) de BPEL4People têm sido propostos. Com base em modelos formais é possível a realização de análises formais, tais como análise de desempenho de modelos para descobrir possíveis problemas em WS orquestrados com o BPEL4People. Há duas dimensões importantes para o desempenho de WS: tempo de resposta e escalabilidade. O tempo de resposta é a capacidade de um sistema de prover tempos aceitáveis para suas atividades e a escalabilidade é a capacidade de um sistema de continuar a cumprir seus objetivos de tempo de resposta quando a demanda pelo mesmo aumenta. Alguns trabalhos têm sido propostos acerca da análise do desempenho de WS orquestrados com o BPEL. Nesta tese, é proposta uma arquitetura denominada "SOASPE" (SOA + SPE) para a transformação de códigos BPEL4People em redes de Petri estocásticas generalizadas - *Generalized Stochastic Petri Nets* (GSPN) e redes de Petri coloridas - *Coloured Petri Net* (CPN). Através dos modelos GSPN e CPN de BPEL4People é possível avaliar o desempenho dos WS orquestrados com o BPEL4People através da comparação dos seus tempos de resposta reais quando submetidos a uma quantidade de requisições e

dos tempos de resposta dos modelos GSPN e CPN dos WS gerados pela arquitetura "SOASPE" quando realizadas simulações com a mesma quantidade de requisições. Durante a transformação de códigos BPEL4People, as redes de Petri são usadas para modelar atividades BPEL e as atividades humanas. Então, pela simulação dos modelos gerados, problemas potenciais com o desempenho de WS orquestrados com o BPEL4People podem ser detectados.

Palavras-chave: Desempenho, SOA, Web Service, BPel4People e Redes de Petri.

Abstract

Web Services (WS) are pillars for the construction of service-oriented applications. A number of languages for web service composition have been proposed, formed a consensus on the language of business process execution (BPEL). BPEL focuses on business processes that orchestrate WS interactions. However, in general, business processes are composed of a broad spectrum of activities that often require human involvement to perform tasks, review or approve steps and enter data. These human interactions are discussed in a new specification of BPEL called BPEL4People. BPEL4People introduces human activity to BPEL. With the use of BPEL4People, formal models (such as Petri nets) of BPEL4People have been proposed. Based on formal models is possible to perform formal analysis, such as performance analysis of models to discover potential problems with the WS orchestrated with BPEL4People. There are two dimensions important to the performance of WS: response time and scalability. The response time is the ability of a system to provide an acceptable time for their activities and scalability is the ability of a system to continue to fulfill its goals of response time when demand for it increases. Some works have been proposed about the performance analysis of WS orchestrated with BPEL. In this thesi, it is proposed an architecture called "SOASPE" (SOA + SPE) for the transformation of the BPEL4People code in generalized stochastic Petri nets (GSPN) and colored Petri nets (CPN). Through the GSPN and CPN models of BPEL4People is possible to evaluate the performance of WS orchestrated with BPEL4People by comparing their real response times when subjected to a number of requests and response times of the GSPN and CPN models generated by the architecture "SOASPE" when simulations with the same amount of requests. During the transformation of BPEL4People code, Petri nets are used to model BPEL activities and human activities. Then, by simulation of

generated models, potential problems with the performance of WS orchestrated with BPEL4People can be detected.

Keywords: Performance, SOA, Web Service, BPel4People and Petri Nets.

Dedico esta tese a meu pai Pedro Jorge Holanda (em memória), a minha mãe Francisca Amorim, a minha esposa Carla Monteiro, aos meus filhos Pedro Henrique e Juliana e a todos meus familiares e amigos que de forma imprescindível e amorosa contribuiram para a realização desta.

"Não basta que seja pura e justa a nossa causa. É necessário que a pureza e a justiça existam dentro de nós."Augustino Neto

Agradecimentos

Gostaríamos de agradecer aos colegas do Centro Politécnico Superior (CPS) da Universidade de Zaragoza (UNIZAR) - Espanha e do Laboratório de Computação do Departamento de Física da Universidade Federal do Ceará, na ajuda e colaboração no desenvolvimento do arquitetura "SOASPE". Gostaríamos também de agradecer ao Professor Giovanni Cordeiro Barroso por nossas interessantes discussões sobre vários aspectos deste trabalho. Um obrigado especial ao Professor José Merseguer Hernáiz por acreditar na minha capacidade e por estar presente no desenvolvimento deste trabalho e por suas freqüentes revisões nas várias etapas da concepção, implementação e publicações acerca desta tese. Finalmente, agradeço a todos aqueles que direta e indiretamente estiveram envolvidos na execução deste trabalho. Também agradeço o apoio e o estímulo oferecidos por minha esposa Carla Monteiro Marques que me acompanhou em todos os momentos difíceis por que passei.

"As veredas dos justos são como a luz da aurora, que vai brilhando mais e mais até ser dia perfeito", Pv. 4:18

Publicações do Autor

0.1 Publicações Nacionais

- Henrique J. Holanda, G. Barroso, Antonio de B. Serra and José Merseguer Hernáiz, "Performance Evaluation of BPEL4People Specifications Integrate Human Interactions Into Business Process", SBSI, Marabá-Pa, Junho, 2010.
- Henrique J. Holanda, G. Barroso and Antonio de B. Serra, "SOASPE: a Framework for the Performance Analysis of Service Oriented Software", SBSI, pp.204-215, May 2009, Brasilia.
- Holanda, Henrique Jorge A., Barroso, Giovanni Cordeiro, Serra, Antonio de Barros (2009) "Performance Analysis of Service Oriented Software", In: iSys - Revista Brasileira de Sistemas de Informação, Volume II, ISSN eletrônico 1984-2902, 2009.

0.2 Publicações Internacionais

- Henrique J. Amorim Holanda, Giovanni Cordeiro Barroso, Carla K. Monteiro Marques, Antonio de Barros Serra, Vando A. S. Alves, "Human Interactions Formalization in Web Services Orchestrated with BPEL4People in the Perspective of Performance Evaluation", Journal of Information Sciences - Elsevier, 2012. (Obs.: artigo aceito a ser publicado com alterações a menor).
- Henrique J. Holanda, G. Barroso, Antonio de B. Serra and José Merseguer Hernáiz, "Performance Analysis of Web Services Orchestrated with WS-BPEL4People", International journal of Computer Networks & Communications - IJCNC, volume 2, number 5, ISSN 0974 - 9322 (Online), 2010.
- Henrique Jorge A. Holanda, Giovanni Cordeiro Barroso, A.B. Serra, "Model Method for the Transformation of BPEL4People into Generalized

Stochastic Petri Nets," iccgi, pp.146-151, Fifth International Multi-conference on Computing in the Global Information Technology, Valencia, Spain, September 20-September 25, ISBN: 978-0-7695-4181-5, 2010. doi = http://doi.ieeecomputersociety.org/10.1109/ICCGI.2010.41, 2010.

- Henrique Jorge A. Holanda, Giovanni Cordeiro Barroso, Antonio de Barros Serra, "SPEWS: A Framework for the Performance Analysis of Web Services Orchestrated with BPEL4WS", ICIW, pp.363-369, Fourth International Conference on Internet and Web Applications and Services, ISBN: 978-0-7695-3613-2, doi = http://doi.ieeecomputersociety.org/10.1109/ICIW.2009.60. Venice/Mestre, Italy, May 24 -May 28, 2009.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Acrônimos

BPEL	*Business Process Execution Language* - Linguagem de Execução de Processos de Negócio
BPEL4People	Linguagem de Execução de Processos de Negócio para pessoas
BP	*Business Process* - Processos de Negócios
BPM	*Business Process Management* - Gerenciamento de Processos de Negócios
CPN	*Colored Petri Net* - Redes de Petri coloridas
GSPN	*Generalized Stochastic Petri Nets* - Rede de Petri Estocástica Generalizada
LQN	Teoria das listas
OASIS	*Organization for the Advancement of Structured Information Standard*
PASA	*Performance Assessment of Software Architectures* - Método de avaliação de desempenho de arquiteturas de software
QN	*Queuing netwoks*
RP	Redes de Petri
SOA	*Service Oriented Architectures* - Arquitetura Orientada a Serviço
SPE	*Software Performance Egineering* - Engenharia de Performance de Software
SOASPE	Junção das siglas SOA e SPE (SOA + SPE)
SOA	*Service-Oriented Architecture* - Arquitetura Orientada a Serviço
SOAP	*Simple Object Access Protocol*
SPE	*Software Performance Engineering* - Engenharia de performance de software
SPN	*Stochastic Petri Nets* - Redes de Petri Estocásticas
RP	Redes de Petri
TI	Tecnologia da informação

UDDI	*Universal Description, Discovery, and Integration*
UML	*Unifield Moleling Language*
XML	*Extensible Markup Language* - Linguagem Extensível de Formatação
WSDL	*Web Services Description Language* - Linguagem de Descrição de Serviços Web
WS	*Web Service*
WSTC	*Web Service Trust Center*

Capítulo 1

Introdução

Em um mundo competitivo, as empresas estão naturalmente interessadas em tecnologia da informação para ganhar vantagem competitiva. Dado que a cooperação se torna cada vez mais importante para as empresas, surgem novos desafios para o apoio às empresas em cenários de negócios com tecnologia da informação. Embora as empresas já utilizem sistemas tradicionais de gerenciamento de *workflow*, a linguagem padrão de execução de processo BPEL (*Business Process Execution Language*) permite a especificação de processos e permite que as empresas colaborem umas com as outras através da interação de processos de negócios. BPEL pode ser usada em processos automatizados entre empresas que utilizam serviços respectivos (SELVAGE D. WOLFSON, 2005). No entanto, o cenário óbvio de um processo de negócio que depende de uma pessoa para cumprir uma determinada tarefa humana como um tipo de atividade do processo não é abrangido pelo BPEL.

O BPEL vem se tornando o padrão para a especificação e a execução de composição de *Web Services* (WS). A maior desvantagem do BPEL é a falta do chamado "fluxo de trabalho humano". A especificação BPEL4People (*Business Process Execution Language for people*) tenta minimizar a presente desvantagem, adicionando suporte à tarefa humana para BPEL (ADOBE et al., 2007).

Como já mencionado, o BPEL não trata do fluxo de trabalho humano. Nos processos de negócio de longa duração, as tarefas que requerem o envolvimento humano são muito frequentes. Um processo pode esperar para a entrada de participantes humanos ou WS, e a entrada deve ser coletada dentro de um determinado intervalo de tempo. Quando se excede o tempo limite, o usuário

deve ser notificado para decidir como o processo deve proceder. Como citado em (HOLANDA G.C. BARROSO, 2010a): "a interação do usuário humano não está atualmente abrangida pelo BPEL, que é principalmente destinado a apoiar processos de negócios automatizados baseados em WS". Na prática, porém, cenários de vários processos de negócio requerem interação do usuário. A introdução da interação humana também leva a outros conceitos, tais como papéis e permissões, que fazem o projeto e a verificação de fluxos de trabalho humano ainda mais difíceis (HOLANDA G.C. BARROSO, 2010b).

Para apoiar e padronizar as atividades humanas em BPEL, BPEL4People é proposto pela IBM-SAP, que descreve os recursos para apoiar as atividades humanas dentro de padrões existentes no BPEL, e introduz o princípio de tarefas manuais executadas por participantes humanos. Alguns problemas como autorizações de tarefas, também são descritos na especificação BPEL4People.

O BPEL4People inclui duas especificações: WS-BPEL extensão para as pessoas e WS-*HumanTask* (ADOBE et al., 2007). A primeira estende a linguagem BPEL com atividade humana e a torna uma invocação de serviço da Web normal. A última define conceitos, tais como papéis humanos, tarefas e permissões usadas no âmbito das atividades de pessoas. Uma vez que ambas devem ser usadas em conjunto para compor um fluxo de trabalho humano, elas são referidas como um todo neste trabalho.

Com o uso do BPEL4People, modelos formais (como redes de Petri) de códigos-fonte BPEL4People têm sido propostos. Estes modelos não só ajudam a compreender melhor a especificação, mas também oferecem subsídios para análises mais importantes no BPEL4People. Além disso, com base no modelo formal pode-se simular o desempenho de modelos e assim descobrir possíveis problemas de gargalos em códigos-fonte BPEL4People (HOLANDA G.C. BARROSO, 2010a; HOLANDA; BARROSO; SERRA, 2010; HOLANDA G.C. BARROSO, 2009a).

Redes de Petri estocásticas generalizadas - *Generalized Stochastic Petri Nets* (GSPN) e redes de Petri coloridas - *Coloured Petri Net* (CPN) são extensões de redes de Petri clássicas. Existem várias razões para a escolha das GSPN e das CPN como linguagens de modelagem de distribuição de trabalho no contexto da BPEL4People. Primeiro, elas têm semântica formal e permitem diferentes tipos de análises, por exemplo, análise de espaço de estado e invariantes. Em segundo lugar, elas são executáveis e permitem a prototipagem rápida e simulações. Em terceiro

lugar, elas são gráficas e suas notações são semelhante às linguagens de fluxo de trabalho existentes. Finalmente, a linguagem GSPN é apoiada pela ferramenta GREATSPN[1] e a linguagem CPN é apoiada pela ferramenta CPN tools[2], que são ambientes gráficos para modelar e analisar GSPN e CPN, respectivamente.

Esta tese motiva-se por questões relacionadas com a definição de uma arquitetura denominada "SOASPE" (SOA + SPE) para a transformação da BPEL4People em GSPN e em CPN e desta forma avaliar o seu desempenho.

1.1 Justificativas e Motivações

O desenho e implementação de sistemas de computadores é uma tarefa difícil. Por esta razão, nos últimos anos as tarefas de modelagem, validação, avaliação de desempenho e implementação destes sistemas têm sido abordadas juntamente com propostas de modelos formais.

Rede de Petri (RP) é um modelo formal que pode auxiliar no desenvolvimento de todo o ciclo de vida dos sistemas (DYER; ZULAUF, 2005). As RP têm sido usadas para modelar e avaliar sistemas de manufatura (LAW; KELTON, 2000), arquiteturas multi-processadas (KELTON; SADOWSKI; SADOWSKI, 2002), sistemas de comunicação (BILLINGTON; DIAZ; ROZENBERG, 1999) e também para avaliar a eficiência de sistemas de computador (AALST, 1998).

Desde que a análise de desempenho de *software* não é uma prática que vem sendo aplicada usualmente no desenvolvimento de *software*, torna-se necessária a existência de uma integração de modelos de análise de desempenho com metodologias de desenvolvimento de sistemas e ferramentas que as implementam. Cadeias de Markov, álgebra estocástica e Redes de Petri Estocástica - *Stochastic Petri Nets* (SPN) são bons paradigmas de modelos de desempenho. As redes de Petri estocásticas se destacam pela sua adequação com a modelagem de sistemas paralelos e distribuídos, simplicidade matemática, seu poder de generalização, sua adequação de expressar todas as semânticas básicas de concorrência, sua fácil representação gráfica, sua fácil detecção de estados e ações, sua boa quantidade de técnicas de análise quantitativa e qualitativa e mais a existência de ferramentas de análise ao seu dispor (MARSAN et al., 1998).

[1] `http://www.di.unito.it/~greatspn`
[2] `http://www.daimi.au.dk/CPNtools/`

Esta tese se justifica no desenvolvimento de sistemas de *software* que preencham os objetivos de desempenho.

Desempenho é o grau em que um sistema de *software* ou componente deste satisfaz a seus objetivos temporais de forma aceitável. Para atingir estes objetivos muitos autores defendem o princípio de que a análise de desempenho deve ser incluída no princípio do projeto de desenvolvimento do *software*. Como o desenho e implementação de sistemas de computadores é uma tarefa difícil, nos últimos anos as tarefas de modelagem, validação, avaliação de desempenho (*performance*) e implementação destes sistemas têm sido abordadas juntamente com propostas de modelos formais. Neste contexto, redes de Petri (RP) é um modelo formal que pode auxiliar no desenvolvimento de todo o ciclo de vida dos sistemas (BRINKSMA; HERMANNS; KATOEN, 2002).

Há duas dimensões importantes para o desempenho de *softwares*: tempo de resposta e escalabilidade.

O tempo de resposta é a capacidade de um sistema de prover tempos aceitáveis para suas atividades. Em sistemas de tempo real, o tempo de resposta é uma medida de quão rápido o sistema responde a um evento, ou o número de eventos que podem ser processados em um determinado instante.

A escalabilidade é a capacidade de um sistema de continuar a cumprir seus objetivos de tempo de resposta quando a demanda pelo mesmo aumenta. Escalabilidade é uma propriedade muito importante para os sistemas de *softwares* (BRINKSMA; HERMANNS; KATOEN, 2002).

É baseada na satisfação destas duas dimensões que este trabalho de tese propõe a arquitetura "SOASPE".

1.2 Hipótese e Objetivos do Trabalho

Para a concretização desta tese parte-se da seguinte hipótese: possibilidade de formalização das interações humanas (WS-HumanTask) nos Web Services orquestrados com o Bpel4Peple na perspectiva de avaliação de desempenho.

1.2.1 Objetivos

- averiguação do estado da arte;

- desenvolvimento e especificação de uma arquitetura "SOASPE" para avaliação de desempenho de *web services* orquestrados através de códigos BPEL (extensão para pessoas - BPEL4People);
- especificar regras e estruturas para conversão de códigos:
  - BPEL4People (Bpel e WS-HumanTask) em GSPN; e
  - BPEL4People (Bpel e WS-HumanTask) em CPN.
- desenvolver algoritmos para atribuição de tempo de atraso das transições através de técnicas de análise de funções de distribuição de probabilidade;
- desenvolver um módulo de captura de tempo de resposta de serviços automatizados e serviços com a participação humana;
- avaliação de desempenho de WS orquestrados com o BPEL4People através da simulação em modelos (GSPN e CPN) obtidos a partir da arquitetura "SOASPE";
- análise comparativa dos resultados de avaliação de desempenho dos modelos em GSPN e dos modelos em CPN de WS orquestrados com o BPEL4People; e
- desenvolver uma ferramenta e uma *interface* para conversão de BPEL4People em GSPN.
- estudos de casos.

1.3 Trabalhos Relacionados

Engenharia de *performance de software* - *Software Performance Engineering* (SPE) e qualidade de servivos (QoS) no contexto de *web service* é o tema de muitos estudos.

Em (MENASCé; ALMEIDA, 2001) foi desenvolvida uma metodologia de avaliação de desempenho de *web services*. Sua metodologia é focada no planejamento de capacidade usando teoria das listas - *Queuing netwoks* (QN). A metodologia de (MENASCé; ALMEIDA, 2001) difere da metodologia desta tese que se utiliza de redes de Petri para a avaliação de desempenho de códigos BPEL4People.

O *Web Service Trust Center* (WSTC) é uma plataforma para desenvolvimento e avaliação de ferramentas de medição e técnicas na área de arquitetura orientada a serviço - *Service Oriented Architectures* (SOA) e serviços web. Uma de suas publicações, intitulada "*Performance modeling of WS-BPEL based web service compositions*" (RUD; SCHMIETENDORF; DUMKE, 2006), aborda aspectos da qualidade de serviço de orquestrações de serviços da Web criados usando o WS-BPEL do ponto de vista de um integrador de serviços web. Um modelo matemático baseado em técnicas de pesquisa operacional e semântica formal de WS-BPEL é proposto para estimar e prever a influência da execução de processos orquestrados sobre a utilização e rendimento de cada um dos nós envolvidos e do sistema como um todo. Este modelo é aplicado para a otimização de processos, de acordo com os níveis de serviço entre as partes envolvidas.

O trabalho aqui proposto se diferencia daquele apresentado em (RUD; SCHMIETENDORF; DUMKE, 2006), pelo fato de usar RP para avaliar o desempenho da orquestração de WS orquestrados com BPEL4People e não um modelo matemático puro como os autores dessa proposta. A diferença de usar RP é que elas também são modelos matemáticos, com a vantagem de proporcionar uma boa visão do modelo do sistema.

Em (SILVA; LINS, 2006), serviços web têm desempenhado um papel importante no desenvolvimento de sistemas distribuídos. Em particular, a possibilidade de composição de *web services* já implementados, a fim de proporcionar uma nova funcionalidade é uma abordagem interessante para a construção de sistemas distribuídos. No entanto, a escolha da melhor composição ainda é um desafio, como qualidades diferentes a serem observadas na composição, tais como segurança, desempenho e tolerância a falhas. Neste contexto, o referido artigo propõe uma metodologia baseada em redes de Petri estocásticas para modelar, avaliar e ajudar a escolher composições *web services*, considerando os aspectos de desempenho.

O trabalho aqui proposto se diferencia daquele apresentado em (SILVA; LINS, 2006), por estar interessado em avaliar o desempenho dos WS orquestrado pelo BPEL4People e não apenas na composição dos serviços web orquestrados pelo BPEL. O maior interesse do trabalho aqui proposto é definir uma arquitetura que possa ser usada para análise de desempenho de WS orquestrados com o BPEL4People.

Outros trabalhos relacionados com obras de interesse são citadas a seguir.

Em (CHANDRASEKARAN et al., 2003), os autores propõem uma técnica de simulação para analisar o desempenho de serviços web compostos, a fim de obter processos web eficientes. O referido artigo descreve a composição e execução da ferramenta (SCET) e diferentes metodologias que poderiam ser adotadas para avaliar o desempenho de um processo de web. A ferramenta (SCET) permite compor serviços estaticamente usando seus modelos (desenhos) e armazenando-os como *Web Service Flow Language* (WSFL). A execução de um processo permite perceber a sua funcionalidade e também analisar o seu desempenho.

Em (NARAYANAN; MCILRAITH, 2003), Narayanan e seu grupo utilizam-se da ontologia (DAML-S) para descrever os recursos de WS e definir a semântica de um subconjunto relevante de (DAML-S) em termos de uma linguagem de lógica de primeira ordem. Com o uso da referida semântica, eles codificam descrições de serviços em redes de Petri e oferecem procedimentos de decisão para simulação, verificação e composição de WS.

Em (GÓMEZ-MARTÍNEZ; MERSEGUER, 2006), os autores fazem, a partir da literatura, um estudo do desempenho de WS. Este estudo, baseado na teoria das listas - *Queuing Networks* (QN), é abordado na sequência da abordagem (PUMA) para obter um novo modelo de desempenho, neste caso em termos de redes de Petri.

Menasce, em seu trabalho (MENASCE, 2004), pressupõe que a arquitetuta orientada a serviços (SOA) permite uma infinidade de prestadores de serviços na prestação de serviços de baixo acoplamento. Seu trabalho considera processos de negócio compostos por atividades que são suportados por provedores de serviços. A estrutura de um processo de negócio pode ser expressa por linguagens como BPEL e permite construções como: sequência, enquanto, fluxo, e escolha. Assim, seu artigo aborda o problema de encontrar o conjunto de prestadores de serviço que minimizam o tempo total de execução do processo de negócios sujeitos a restrições de tempo e custo de execução. Este problema é claramente NP-difícil. No entanto, o trabalho apresenta um algoritmo otimizado que encontra a solução ideal sem ter de explorar o espaço amostral de toda a solução. Este algoritmo pode ser utilizado para encontrar a solução ótima em problemas de tamanho moderado. Uma solução heurística também é apresentada e os estudos experimentais que comparam a melhor solução e a heurística mostram que o tempo de execução médio obtido com uma dotação heurística dos prestadores de serviços não ultrapa 6% do tempo da solução ótima.

Em (DUN; XU; WANG, 2008), a composição de WS envolve a combinação de um conjunto de serviços existentes para criar um serviço de valor agregado. BPEL é uma linguagem promissora que descreve a composição de *web service* em forma de processos de negócios. O BPEL é uma linguagem baseada em XML e faz-se necessário analisar os processos de negócios especificados em BPEL em uma ferramenta formal. Em seu artigo (DUN; XU; WANG, 2008), os autores apresentam uma abordagem para o modelo e o verifição com base em BPEL ServiceNet, uma classe especial de redes de Petri. Eles apresentam algumas regras de transformação dos processos de negócio em BPEL *ServiceNet*. Em seguida, a análise de um processo de negócio BPEL pode ser verificada através da redução da *ServiceNet* correspondente com base em algumas regras de redução.

1.4 Organização do Trabalho

Esta tese encontra-se organizada, além deste capítulo introdutório, da seguinte forma. No Capítulo 2 são introduzidos conhecimentos preliminares em Engenharia de *Performance de Software* (SPE), Arquitetura Orientada a Serviços (SOA), *Web Services* e Modelos Formais, que são necessários para a compreensão da tecnologia da arquitetura "SOASPE". No Capítulo 3 são abordos BPEL e BPEL4People que são as linguagens de orquestração de WS utilizadas pela arquitetura "SOASPE". No Capítulo 4 é apresentada a arquitetura "SOASPE". O Capítulo 5 apresenta a modelagem das interações humanas em GSPN e em CPN. Uma ferramenta automática de conversão de BPEL4People em GSPN, denominada de "TPeople4PN", é apresentada no Capítulo 6. São apresentadas no Capítulo 7 as conclusões e as sugestões de trabalhos futuros para a continuação dos estudos na linha de trabalho aqui proposta.

Capítulo 2

Conhecimentos Preliminares

Neste capítulo são apresentados os assuntos que formam a base do desenvolvimento deste trabalho. Inicialmente, na Seção 2.1 é introduzida a Engenharia de *Performance de Software* - *Software Performance Egineering* (SPE). Na Seção 2.2 é conceituada Arquitetura Orientada a Serviços - *Service Oriented Architecture* (SOA) referente a negócios e requisitos tecnológicos. A tecnologia de *Web Wervices* (WS) é descrita na Seção 2.3 como uma implementação de SOA. Na Seção 2.4 é apresentada uma visão geral sobre modelos formais com enfoque em redes de Petri - *Petri Nets* (PN) e suas extensões GSPN e CPN.

2.1 Engenharia de Performance de *Software* - SPE

Desempenho (*performance*) é fundamental para o sucesso de um projeto de *software*. No entanto, muitos *softwares* não satisfazem os objetivos de desempenho quando da sua concepção inicial. A solução destas insatisfações é cara e causa atrasos de cronograma, perda de produtividade, e uma série de outras dificuldades (WESTERGAARD; LASSEN, 2006). Nesta seção é apresentada a Engenharia de *Performance* de *Software* (SPE), uma abordagem sistemática quantitativa para construir sistemas de *software* que preencham os objetivos de desempenho. A SPE deve começar a ser utilizada no início do processo de desenvolvimento de *software* para uma proposta de arquitetura de desenho de alto nível. Nesta tarefa os modelos ajudam a identificar potenciais problemas de desempenho quando eles podem ser corrigidos rapidamente e economicamente.

2.1.1 Definição de SPE

Embora a funcionalidade de uma aplicação de *software* seja obviamente importante, esta não deve ser a única preocupação. Durante a vida útil de um *software*, seu custo deve ser determinado mais pela maneira como ele deve atingir os seus objetivos em matéria de qualidade como desempenho, confiabilidade, disponibilidade e manutenibilidade, do que por sua funcionalidade (CHEN; CHEN; SHAO, 2003).

Esta seção centra-se na conceituação de desenvolvimento de sistemas de *software* que preencham os objetivos de desempenho. Desempenho é o grau em que um sistema de *software* ou componente deste satisfaz a seus objetivos temporais de forma aceitável. Assim, o desempenho é qualquer característica de um produto de *software* que você poderia, em princípio, medir com um cronômetro na mão (SMITH, 1989).

Há duas dimensões importantes para o desempenho de *software*: "tempo de resposta" e "escalabilidade". Tempo de resposta é a capacidade de um sistema de prover tempos aceitáveis para suas atividades. Em sistemas de tempo real, o tempo de resposta é uma medida de quão rápido o sistema responde a um evento, ou o número de eventos que podem ser processados em um determinado instante. A escalabilidade é a capacidade de um sistema de continuar a cumprir seus objetivos de tempo de resposta quando a demanda pelo mesmo aumenta. Escalabilidade é uma propriedade muito importante para os sistemas de *software* (SMITH, 1989).

Desempenho é fundamental para o sucesso dos sistemas de *software*. No entanto, a maioria dos projetos de *software* não tem a preocupação de avaliar o desempenho do seu produto na fase inicial. A avaliação de desempenho somente é feita quando o *software* já está implementado e em produção. Neste caso a correção destes problemas é onerosa e provoca atrasos de cronograma, perda de produtividade, e uma série de outras dificuldades. Em casos extremos, pode não ser possível corrigir problemas de desempenho sem uma extensa reforma do projeto lógico e consequente re-implementação. Nesses casos, o projeto ou se torna um consumidor infinito de tempo e dinheiro, ou é incondicionalmente cancelado (SMITH, 1989).

O desempenho deve ser objeto de análise desde o início do projeto do *software*. O "fazer executar, fazê-lo funcionar direito e após torná-lo rápido" é uma abordagem perigosa que deve ser substituída por uma abordagem baseada nos princípios da

SPE.

A SPE é uma abordagem sistemática quantitativa para construir sistemas de *software* que preencham objetivos de desempenho. Ela define princípios para a criação de *software* com bom desempenho, define também parâmetros necessários para a avaliação de desempenho de *software* e orientações para os tipos de avaliação de desempenho que devem ser realizadas em cada fase do desenvolvimento do sistema de *software*. A SPE incorpora a modelos de representação, seus princípios de avaliação de desempenho, bem como um conjunto de métodos de análise (SMITH, 1989).

Devido à importância da arquitetura do *software* na determinação de seu desempenho, a SPE toma uma perspectiva arquitetônica. Os princípios e técnicas de SPE formam a base para o método de avaliação de desempenho de arquiteturas de *software* - *Performance Assessment of Software Architectures* (PASA) (SMITH, 1989).

A PASA foi desenvolvida segundo a experiência na condução de avaliação de desempenho em múltiplas arquiteturas de *software* em vários domínios, incluindo aplicativo baseado em sistemas Web e sistemas em tempo real. Ela usa os princípios e as técnicas citadas adiante nesta seção para determinar se uma arquitetura é capaz de suportar seus objetivos de desempenho. Quando um problema é encontrado, PASA também identifica estratégias para reduzir ou eliminar esses riscos.

O método PASA pode ser aplicado no desenvolvimento de um projeto novo para descobrir possíveis problemas quando são mais fáceis e menos dispendiosos para corrigir. Também pode ser utilizado quando da atualização de sistemas legados, para se decidir se quer continuar a empenhar recursos na atual arquitetura ou migrar para uma nova, ou em sistemas existentes que possuam um fraco desempenho e que exigem rápida correção.

2.1.2 Visão Geral da SPE

A SPE é uma metodologia cuja abordagem é a utilização de métodos de transformação de *software* em modelos formais, com o objetivo de se utilizar estes modelos para que se possa identificar problemas com a arquitetura do sistema, desenho ou implementação deste. Esses modelos são construídos e analisados para fornecer *feedback* sobre se a proposta de *software* é susceptível de permitir alcançar suas metas de desempenho. Conforme o processo de desenvolvimento do *software*

avança, os modelos são refinados para melhor representar o desempenho destes novos estágios do *software*.

A precisão dos resultados da análise do modelo depende da qualidade dos valores dos parâmetros de estimativas estabelecidos. Estes valores são difíceis de estimar para as arquiteturas de *software*. A SPE utiliza estratégias adaptativas, tais como a definição de fronteiras superior e inferior, que representam o melhor e pior caso. Usando estas estimativas, os analistas produzem previsões do melhor e do pior caso de desempenho. Se os resultados forem insatisfatórios então tenta-se identificar componentes críticos cujas estimativas têm o maior efeito sobre o desempenho e centra-se na obtenção de valores mais precisos para eles. Uma variedade de técnicas pode fornecer mais precisão, incluindo: refinar ainda mais a arquitetura do sistema e construção mais detalhada de modelos formais destes ou ainda a construção de protótipos para medição de desempenho dos componentes críticos (SMITH, 1989).

Dois tipos de modelos fornecem informações acerca da avaliação de desempenho da arquitetura de *software*: o modelo de execução do *software* e o modelo de execução do sistema. O modelo de execução do *software* é obtido a partir de modelos UML do *software* e representa aspectos de comportamento da execução deste *software*. Ele é construído utilizando grafos executáveis para representar a atividade dos diversos cenários. Nestes grafos, "nós" representam componentes funcionais do *software* e "arcos" representam controle de fluxo. Os gráficos são hierárquicos e devem conter informações completas acerca da estimativa temporal dos recursos necessários (SMITH, 1989).

A construção do modelo de execução do *software* proporciona uma análise estática da arquitetura do *software*. Esse modelo analisa o desempenho dos recursos da proposta de *software* isoladamente, na ausência de outros. Se esta análise é satisfatória, então não há necessidade de construir modelos de análise mais sofisticados. O modelo de execução do *software* é geralmente suficiente para identificar problemas devido ao mau desempenho da arquitetura do *software*.

Se o modelo de execução do *software* indica que não há problemas, então os analistas podem continuar a análise de desempenho com a definição do modelo de execução do sistema. Este modelo dinâmico caracteriza o desempenho do *software* na presença de fatores, tais como: existência de vários cenários e usuários, o que poderia afetar o desempenho do mesmo.

Os resultados obtidos através do modelo de execução do *software* fornecem

parâmetros de entrada para o modelo de execução do sistema. O modelo de execução do sistema prevê a análise de desempenho levando em consideração as seguintes informações adicionais (SMITH, 1989):

- refinamento dos requisitos de desempenho;
- métricas mais precisas dos recursos que representam contensão;
- métricas de desempenho mais sensíveis às variações de composição de carga;
- identificação dos recursos gargalo;
- dados comparativos para melhorar o desempenho através de mudanças de *workload*, alterações de *software*, atualizações de *hardware* e várias combinações de cada um destes;
- escalabilidade da arquitetura: o efeito do crescimento futuro no desempenho;
- identificação dos elementos cruciais da concepção; e
- concepção de testes de desempenho.

2.1.3 O Processo de Modelagem da SPE

O processo de modelagem da SPE, centrado em diagramas de caso de uso do modelo UML do sistema e em seus cenários, é descrito por Smith (SMITH, 1989). O Desenvolvimento a partir de uma perspectiva de utilização de diagramas de caso de uso e seus cenários fornece um meio de compreensão dos requisitos do sistema e de sua arquitetura, possibilitando assim a construção de modelos formais para cada cenário e sua efetiva avaliação de desempenho.

O processo SPE inclui os seguintes passos que estão esquematizados no diagrama de atividades mostrado na Figura 2.1 (SMITH, 1989).

i. identificar os casos críticos;

ii. verificar e validar os modelos;

iii. selecionar os cenários chaves de desempenho;

iv. estabelecer objetivos de desempenho;

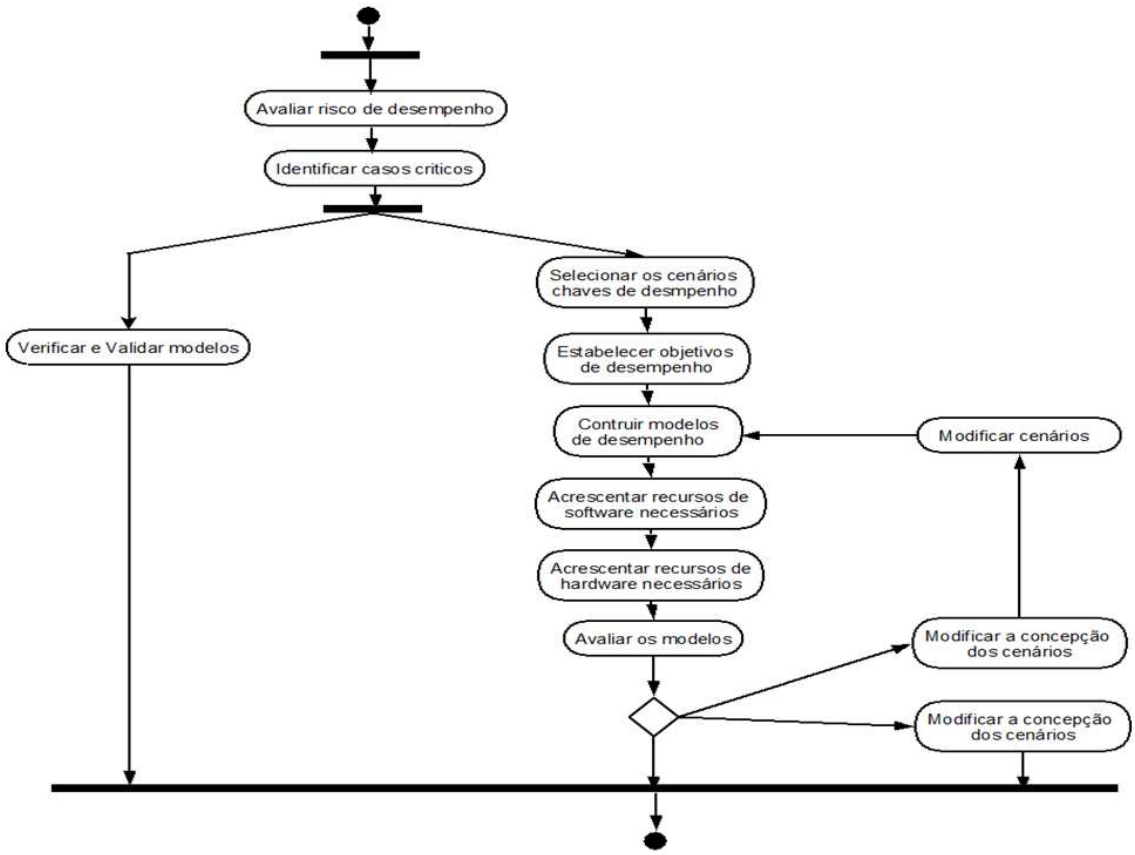

Figura 2.1: O processo de modelagem SPE (SMITH, 1989).

v. construir modelos de desempenho;

vi. acrescentar recursos de *softwares* necessários;

vii. acrescentar recursos de *hardware* necessários;

viii. avaliar os modelos;

ix. revisar objetivos de desempenho; e

x. modificar a concepção dos cenários.

2.2 Arquitetura Orientada a Serviços (SOA)

Service-Oriented Architecture (SOA) - em português, Arquitetura Orientada a Serviços - é um termo que descreve duas coisas muito diferentes. As duas primeiras palavras expressam uma metodologia para desenvolvimento de *software*. A terceira palavra é um panorama de todos os ativos de *software* de uma empresa, assim como

uma planta arquitetônica é uma representação de todas as peças que, juntas, formam uma construção. Portanto, "*service-oriented architecture*" é uma estratégia que proclama a criação de todos os ativos de *software* de uma empresa via metodologia de programação orientada a serviços (ERL, 2005)

A arquitetura SOA torna possível transformar aplicações monolíticas rígidas em componentes de *software* flexíveis, permitindo a interoperabilidade entre diferentes tecnologias. O alavancar das aplicações legadas e enfoque na reutilização, a redução dos custos de desenvolvimento e manutenção tornam as soluções de negócio mais ágeis e fáceis de implementar, viabilizando as oportunidades de negócio em tempos antes considerados impossíveis de realizar (ARSANJANI, 2004).

O motivo para se migrar para o SOA é a reutilização de serviços de negócio (ARSANJANI, 2004). Os programadores dentro e fora da organização podem alavancar o código desenvolvido pelas aplicações existentes, expondo-o como um *web service*, e podem reutilizá-lo para satisfazer novos requisitos de negócio. A reutilização de funcionalidades que já existem na própria organização ou no exterior pode resultar na redução significativa do esforço de desenvolvimento do aplicativo e, portanto, dos custos. Os benefícios de reutilização crescem dramaticamente à medida que mais serviços de negócio são construídos e incorporados nas diferentes aplicações.

A abordagem SOA das interações entre os pedidos de clientes e os serviços fracamente acoplados (*loosely-coupled*) traduz-se tão somente num conceito: interoperabilidade vasta e abrangente. Por outras palavras, pretende-se que os clientes e os serviços possam comunicar-se e "compreender-se" mutuamente, independentemente da plataforma que os suporta. Este objetivo somente será tangível se tanto os clientes como os serviços fizerem uso de uma forma padrão e bem definida de comunicação entre si - uma forma consistente, definida e transparente que seja reconhecida através das plataformas, sistemas e linguagens atuais. De fato, os *web services* são os "suportes" que permitem exatamente isto. Os *web services* providenciam um conjunto robusto e consistente de protocolos, regras e tecnologias que são completamente independentes das plataformas, sistemas ou linguagens de programação (FARREL, 2003).

Os serviços *loosely-coupled* são tipicamente muito mais flexíveis que as aplicações baseadas nas arquiteturas distribuídas. Os componentes deste tipo de aplicações estão intrinsecamente ligados entre si, combinando semânticas, utilizando bibliotecas

comuns e partilhando até estados de execução. Este fato torna difícil e pesado todo o trabalho de evolução e adaptação destas aplicações às constantes mutações do negócio, das suas regras e das suas necessidades e requisitos. Em contraponto, tem-se a arquitetura SOA, que utilizando serviços de natureza "*loosely-coupled*" possibilita às aplicações grande flexibilidade e facilidade de evolução e rápida adaptação às constantes mutações do negócio, das suas regras e requisitos (ERL, 2005).

2.2.1 Definição de SOA

O termo *Service-Oriented Architecture* foi inicialmente proposto por SCHULTE & NATIS, então analistas do *Gartner Group*. Eles definiram SOA como "uma configuração de computação multicamadas que ajuda as organizações compartilhar lógica e dados através de múltiplas aplicações e modelos de uso" (SCHULTE; NATIS, 2006).

Estar preparado para atender às demandas das áreas de negócios com mais agilidade e flexibilidade. Ter como cenário um ambiente baseado em uma arquitetura de *software* modular, onde todos os aplicativos são acessados por uma única *interface web* e os sistemas utilizam dados uns dos outros e "conversam" indiscriminadamente. Isto é o pano de fundo de uma arquitetura orientada a serviços (SOA).

Neste novo modelo arquitetural, não há mais uma *interface*, um banco de dados e um sistema de integração para cada *software*. Na web, em uma única tela, os usuários acessam todos os programas de forma transparente. Além de mais flexível e mais amigável, a interface única dá muito mais agilidade e transparência ao usuário. Os aplicativos, por sua vez, utilizam qualquer informação, independentemente de onde ela tenha sido gerada, graças ao mapeamento lógico dos dados. É como se todos estivessem em uma única base, ajudando a reduzir a inconsistência.

A *Organization for the Advancement of Structured Information Standards* define SOA da seguinte forma: Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de competências distribuídas, as quais estão sob o controle de diferentes domínios proprietários. Ela fornece um meio uniforme de oferecer, descobrir, interagir e usar capacidades para produzir efeitos desejáveis consistentes com pré-condições e expectativas mensuráveis (SCHULTE; NATIS, 2006).

2.2.2 Benefícios e Desafios de uma Arquitetura SOA

Os benefícios para os interessados em implementar uma SOA para gerenciamento da informação são significantes. Os seguintes benefícios são alcançados (SCHULTE; NATIS, 2006):

- **permite a reutilização dos ativos de TI**: este é o primeiro e o mais importante benefício de implementar SOA. Pode-se construir um serviço de negócio como uma agregação de componentes existentes. Para utilizar este novo serviço é exigido somente saber seu nome e *interface*. A implementação de um serviço específico, bem como sua arquitetura e as complexidades do fluxo de dados que compõem este serviço, são transparentes para quem o chama. Isto permite às organizações aproveitar investimentos atuais, construir serviços de um conglomerado de componentes construídos em diferentes plataformas, que rodam em diferentes sistemas operacionais e desenvolvidos em diferentes linguagens de programação. Sistemas legados podem ser encapsulados e acessados utilizando as *interfaces web services*;

- **diminui o tempo de desenvolvimento e reduz os custos de desenvolvimento e manutenção**: como as demandas do negócio crescem e novos requisitos são introduzidos a todo o momento, o custo de desenvolver e criar novos serviços através da arquitetura SOA e biblioteca de serviços é bastante reduzido. A curva de aprendizagem de uma equipe de desenvolvimento é também reduzida, pois a mesma pode estar familiarizada com os componentes existentes;

- **diminui o risco**: reusar componentes existentes reduz o risco de se introduzir novas funcionalidades dentro de processos de melhoria ou criar novos serviços do negócio. O risco de manutenção e gerenciamento da infra-estrutura dos serviços de suporte também será reduzido; e

- **protege investimento**: protege o investimento do cliente no gerenciamento da informação que é altamente volátil num mercado onde as fusões e aquisições são frequentes.

Atualmente a Tecnologia da Informação (TI) tem sofrido várias modificações, impulsionada pela necessidade de melhoria no relacionamento que existe entre as

áreas que suportam tecnologicamente os negócios e os negócios propriamente ditos. Seus gestores estão se dando conta da necessidade e da conveniência do enfoque nos serviços, como um fim, sendo vitais para os negócios em um futuro que já se tornava atualidade. A SOA trouxe à tona a necessidade de fortalecer o enfoque no cliente e tornar a gestão de serviços como uma atividade produtiva, que agregue valor à empresa (BOTTO, 2004).

2.2.3 Principais Termos e Definições de SOA

A visibilidade, interação e efeitos são os conceitos chaves para descrever o paradigma SOA (SCHULTE; NATIS, 2006). A visibilidade refere-se à capacidade para aqueles com necessidades e aqueles com competências estarem aptos a se verem mutuamente. Isto é possível pelo oferecimento de descrições acerca destes aspectos como as funções e requisitos técnicos, restrições e políticas relacionadas, e mecanismos para acesso e resposta. Enquanto a visibilidade introduz a possibilidade de compatibilizar as necessidades com as competências (e vice-versa), a interação é a atividade que usa a competência mediada por troca de mensagens. Uma interação prossegue através de uma série de ações de troca de informações e invocações. O propósito de usar as competências é realizar um ou mais efeitos no mundo real. Uma interação é "um ato" em oposição a "um objeto" e o resultado de uma interação é um efeito (ou um conjunto/série de efeitos). Este efeito pode ser o retorno de uma informação ou a mudança no estado de entidades (conhecidas ou desconhecidas) que estão envolvidas na interação (SCHULTE; NATIS, 2006).

A descrição de SOA tem ainda que mencionar o que é usualmente considerado o conceito central de SOA: o serviço. O termo "serviço" pode ser definido como "a realização de trabalho (uma função) por alguém para outro." Contudo, serviço, como o termo é geralmente entendido, também combina as idéias relacionadas com:

- a competência de executar o trabalho para outro;
- a especificação do trabalho oferecido para outro; e
- a oferta para executar trabalho para outro.

Serviço é uma função independente, sem estado, que aceita uma ou mais requisições e retorna uma ou mais respostas através de uma *interface* padronizada e bem definida. Serviços podem também realizar partes discretas de um processo tal

como editar ou processar uma transação. Serviços não devem depender do estado de outras funções ou processos. A tecnologia utilizada para prover o serviço, tal como uma linguagem de programação, não pode fazer parte da definição do serviço (SCHULTE; NATIS, 2006).

A *ERL-2005* estabelece uma relação entre serviço e operações, como ilustrado na Figura 2.2, onde pode-se observar que um serviço representa um conjunto logicamente agrupado de operações capaz de executar unidades de trabalho relacionades (ERL, 2005).

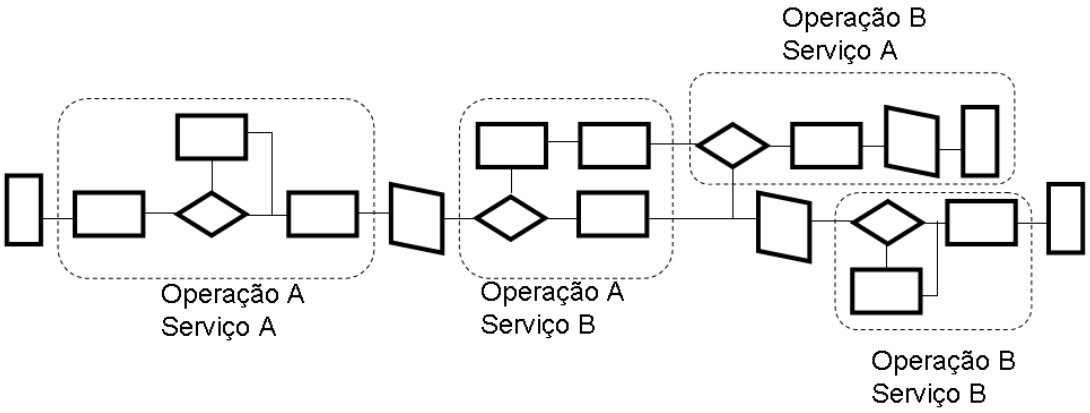

Figura 2.2: Operações pertencentes a diferentes serviços representam várias partes de um processo de negócios lógico.

Na Tabela 2.1 são mostrados alguns dos principais termos utilizados por SOA e suas respectivas definições.

2.2.4 Princípios de SOA

A ERL-2005 define os seguintes princípios de uma arquitetura orientada a serviços (ERL, 2005):

- **reusabilidade do serviço**: é a divisão em serviços com a intenção de promover o seu reuso;
- **fraco acoplamento**: serviços mantêm um relacionamento que minimiza dependências e somente requer que eles mantenham o conhecimento da existência um do outro;
- **contrato de serviço**: os serviços aderem a um acordo da comunicação, como definidos coletivamente por um ou mais documentos de descrição do serviço;

Tabela 2.1: Principais definições de SOA (SCHULTE; NATIS, 2006).

Termo	**Definição / Comentário**
Serviço	É uma função independente, sem estado (*stateless*) que aceita uma ou mais requisições e retorna uma ou mais respostas através de uma interface padronizada e bem definida. Serviços podem também realizar partes discretas de um processo tal como editar ou processar uma transação. Serviços não devem depender do estado de outras funções ou processos. A tecnologia utilizada para prover o serviço, tal como uma linguagem de programação, não pode fazer parte da definição do serviço.
Orquestração	Processo de sequenciar serviços e prover uma lógica adicional para processar dados. Não inclui uma representação de dados.
Stateless	Não depende de nenhuma condição pré-existente. Os serviços não devem depender de condições de outros serviços. Eles recebem todas as informações necessárias para prover uma resposta consistente. O objetivo de buscar a característica de *stateless* dos serviços é possibilitar que o consumidor do serviço possa sequenciá-los, ou seja, orquestrá-los em vários fluxos (algumas vezes chamados de *pipelines*) para executar a lógica de uma aplicação.
Provedor	O recurso que executa o serviço em resposta a uma requisição de um consumidor
Consumidor	É quem consome ou pede o resultado de um serviço fornecido por um provedor.
Descoberta	SOA se baseia na capacidade de identificar serviços e suas características. Consequentemente, esta arquitetura depende de um diretório que descreva quais os serviços disponíveis dentro de um domínio.

- **abstração**: além do que está descrito no contrato de serviços, serviços escondem a lógica para quem está olhando de fora;
- **composabilidade**: coleções de serviços podem ser coordenadas e montadas para formar o menu de serviços;
- **autonomia**: serviços têm o controle sobre a lógica que eles encapsulam;
- **sem estado**: serviços minimizam a retenção da informação específica para uma atividade; e
- **descobertabilidade**: os serviços são projetados para serem descritivos, de modo que possam ser achados e possam ser avaliados via mecanismos disponíveis de descoberta.

Todos os princípios são bastante importantes, mas o da reusabilidade passa a ser o principal. Quando um serviço encapsula a lógica que é utilizada por mais de uma requisição de serviço, este pode ser considerado reusável. O conceito de reuso é baseado em um número de princípios complementares, tais como (MARTENS, 2003):

- autonomia do serviço estabelece um ambiente de execução que facilita o reuso porque o serviço tem independência e auto-governança;
- sem-estado do serviço suporta o reuso porque ele maximiza a disponibilidade de um serviço e em geral promove uma especificação genérica do serviço que difere do processamento de atividades específicas fora das fronteiras lógicas do serviço;
- abstração do serviço permite o reuso porque estabelece o conceito de caixa preta, onde os detalhes do processamento são completamente escondidos do requisitante. Isto permite que um serviço expresse de uma forma simples uma *interface* pública genérica;
- descobertabilidade do serviço promove o reuso, e isto permite a requisitantes procurar e descobrir serviços reusáveis; e
- baixo acoplamento dos serviços estabelece uma independência intrínseca que liberta um serviço das amarras imediatas de outros. Isto tem uma grande influência em deixar um serviço reusável.

2.2.5 As Camadas de uma Arquitetura SOA

ERL-2005 apresenta um alto nível de abstração da camada de serviços posicionada entre as camadas dos processos de negócio e de aplicação, que demonstra a conectividade aberta entre as camadas de serviço. No nível físico, serviços são desenvolvidos em ambientes proprietários, onde eles são individualmente responsáveis pelo encapsulamento de uma aplicação lógica específica (ERL, 2005). Na Figura 2.3 é mostrado como serviços individuais, dentro da camada lógica de serviços, representam aplicações lógicas originadas em diferentes plataformas (ERL, 2005).

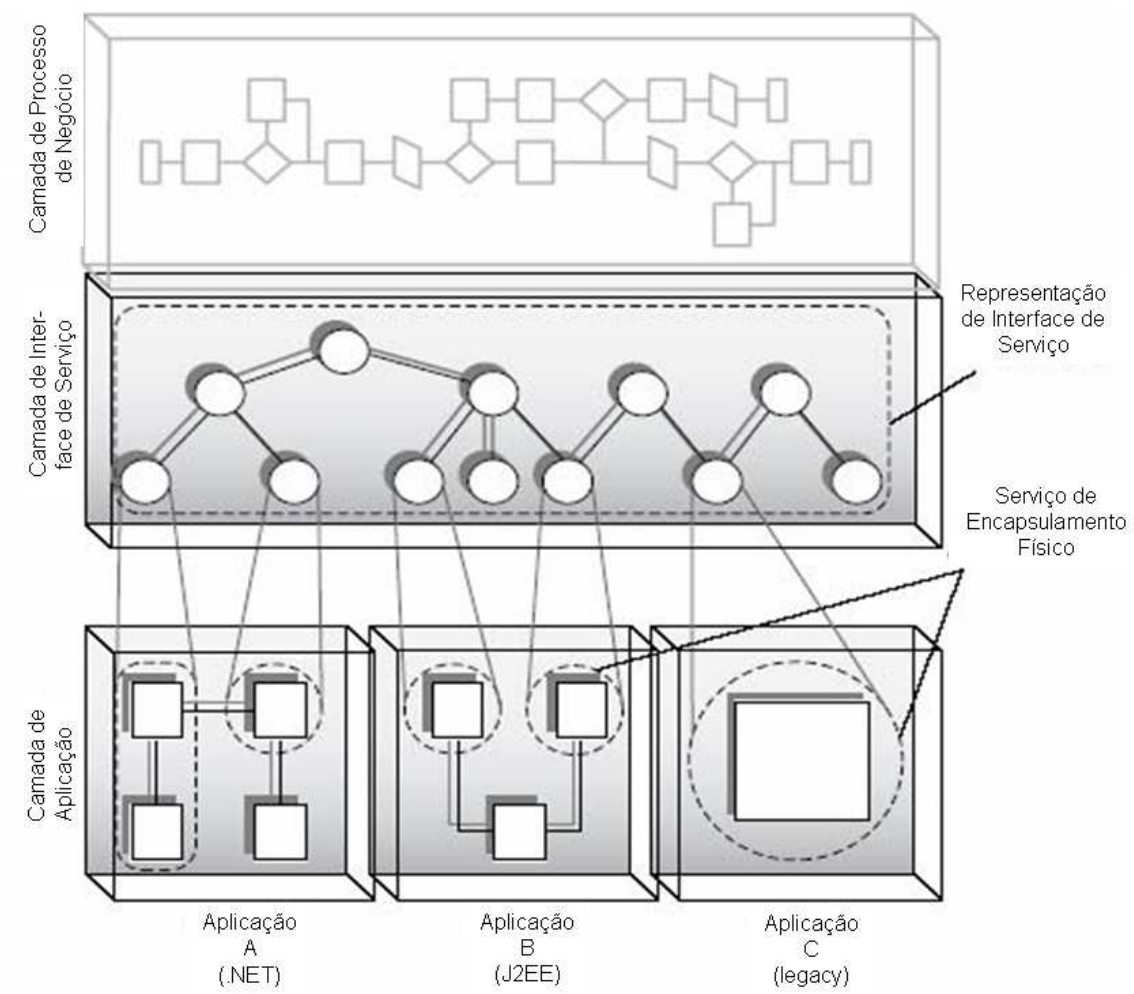

Figura 2.3: Camada de serviços posicionada entre as camadas de processos de negócio e de aplicação (SCHULTE; NATIS, 2006).

2.2.6 SOA e a Tecnologia *Web Services*

SOA expressa um conceito em que aplicativos ou rotinas são disponibilizados como serviços em uma rede de computadores (Internet ou Intranets) de forma independente e se comunicando através de padrões abertos. A maior parte das implementações de SOA se utilizam de *web services*. Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em web (CURBERA et al., 2002).

A base do padrão *web services* relevantes ao SOA incluem:

- **XML (*eXtensible Markup Language*)**: é uma linguagem de marcação para descrever dados em cargas de mensagens em um formato de documento;
- **SOAP (*Simple Object Access Protocol*)**: é um protocolo baseado em XML utilizado para trocas de informações num ambiente distribuído. A utilização de XML é uma forma eficiente para trocas de informações, mas não é suficiente para trocas de dados via Web. Ainda precisamos de um protocolo em comum para envio de documentos XML pela Web de tal forma que o receptor entenda o que está recebendo e o que deve ser respondido. É neste ponto que entra o SOAP;
- **WSDL (*Web Services Description Language*)**: é um documento XML que descreve um conjunto de mensagens SOAP e a forma como essas mensagens são trocadas. Como o WSDL é XML, ele é legível e editável, mas na maioria dos casos, ele é gerado e consumido pelo *software*; e
- **UDDI (*Universal Description, Discovery, and Integration*)**: é uma das formas de localizar um *Web Service* num registro, como se fosse um catálogo de páginas amarelas, de tal forma que um programa em busca de um determinado serviço possa facilmente localizar e entender o que ele faz.

Segundo ARSANJANI (ARSANJANI, 2004), a arquitetura orientada a serviços (SOA) pode ser bem representada a partir do seguinte modelo envolvendo três partes principais: o fornecedor do serviço, o usuário e o registro do serviço, também chamado de "*find-bind-execute paradigm*" o que significa paradigma de "procurar-consolidar-executar" conforme melhor demonstrado na Figura 2.4.

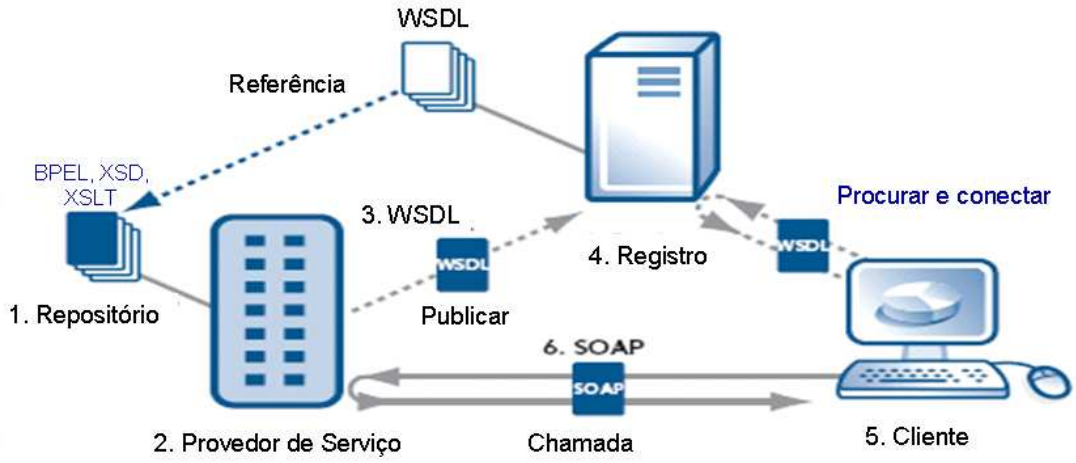

Figura 2.4: Paradigma *procura-consolida-executa*, adaptado de (ARSANJANI, 2004).

Tecnicamente falando, o processo preconiza que os provedores de serviços registrem informações em um registro central, com suas características, indicadores, e aspectos relevantes às tomadas de decisões. O registro é utilizado pelo cliente para determinar as características dos serviços necessários, e se o mesmo estiver disponível no registro central, como por exemplo, por um catálogo de serviços, o cliente poderá utilizá-lo, sendo este oficializado através de um contrato de serviço que enderece este serviço.

2.2.7 Estratégias de Desenvolvimento e Implementação SOA

ERL-2005 define as seguintes estratégias para desenvolvimento e implementação SOA (ERL, 2005):

- **a estratégia em top-down**: esta estratégia de cima para baixo requer não somente que os processos de negócio se tornem orientados a serviço, mas também promove a criação (ou realinhamento) de todo o modelo de negócios da organização. Os seguintes passos estão previstos nesta abordagem:
  - definição de processos de negócio;
  - composição SOA;
  - definição do modelo de serviço;
  - análise orientada a serviço;
  - especificação orientada a serviço;

- desenvolvimento dos serviços;
- testes; e
- implementação.

- **a estratégia *bottom-up***: esta abordagem de baixo para cima encoraja primeiramente a criação das aplicações de serviços servindo como base para as demais fases, e a integração passa a ser o motivador principal desta estratégia. Os seguintes passos estão previstos nesta abordagem:

  - modelar as aplicações de serviços;
  - especificar as aplicações;
  - desenvolver as aplicações;
  - testar; e
  - implementar.

A escolha de uma ou outra estratégia vai depender de cada organização considerando as variáveis tempo e recursos financeiros que dispõem. A abordagem de cima para baixo claramente exige mais tempo e tem um custo bem mais alto que a abordagem de baixo para cima, pois esta última se utiliza das configurações e processos que se encontram na organização, não sendo necessário rever todos os processos e pode-se também escolher determinadas aplicações para torná-las orientadas-a-serviço.

- **a estratégia mista**: pode-se também aplicar uma estratégia mista, ou seja, uma combinação entre as abordagens *top-down e bottom-up*, onde as fases de modelagem dos processos e do negócio se dão concomitantemente com as fases de especificação e implementação, necessitando um re-alinhamento com os processos em cada uma das fases.

2.3 *Web Services* (WS)

Web services podem ser definidos como componentes, ou unidades lógicas de aplicação, acessível através de protocolos padrões da Internet, possuindo uma funcionalidade que pode ser reutilizada sem a preocupação de como é implementada. A W3C define *web services* como uma aplicação identificada por uma URI (*Uniform*

Resource Identifier), cujas *interfaces* e ligações são definidas, descritas e descobertas utilizando-se como padrão a linguagem XML (GARCIA; TOLEDO, 2006).

Segundo (LUDWIG, 2003), a tecnologia *web services* poderá ser tão importante para as interações entre aplicação-aplicação (B2B), quanto a web foi para as interações entre aplicação-usuário (B2C). *Web services* possibilitam empresas reduzirem custos de *e-bussiness*, fornecem soluções mais rápidas e criam novas oportunidades de negócios.

Além disso, *web services* permitem que empresas desenvolvam funções que podem ser utilizadas sob demanda ou utilizadas em conjunto para prover um serviço de negócio (LUDWIG, 2003). *Web services* estão emergindo para fornecer uma infraestrutura sistemática e extensível para interações aplicação-aplicação, construída sobre protocolos web já existentes e baseada em padrões XML abertos (LUDWIG, 2003). A arquitetura adotada pelos *web services* é mostrada na Figura 2.5.

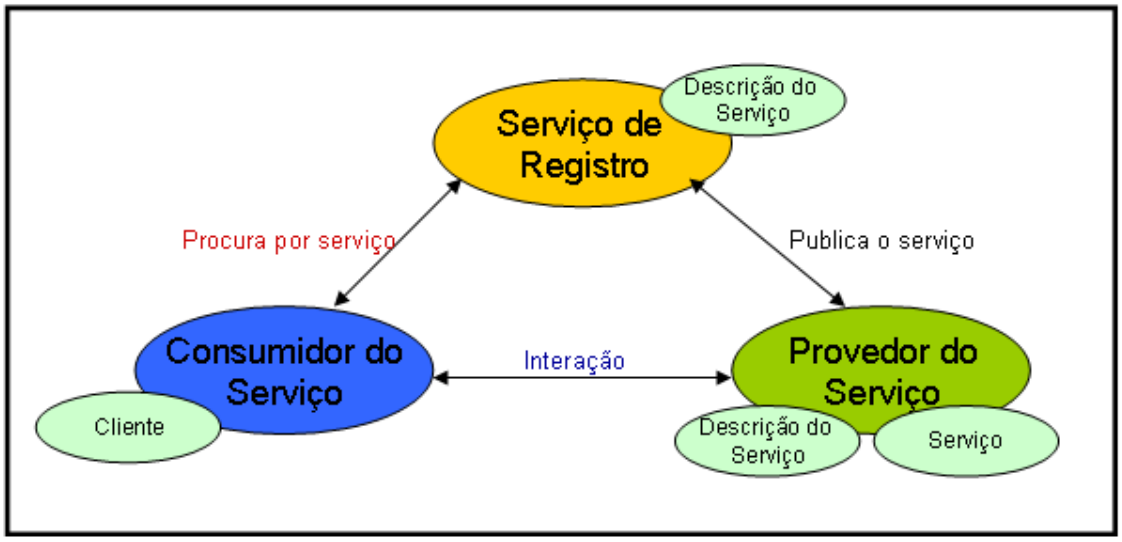

Figura 2.5: Arquitetura adotada pelos WS (LUDWIG, 2003)

A arquitetura, mostrada na Figura 2.5, é formada por três entidades participantes: provedor de serviço, consumidor de serviço e registro de serviços. O provedor de serviço é o repositório do serviços. É função do provedor de serviço descrever de forma padronizada cada serviço criado e publicá-lo em registros de serviços. O consumidor de serviço ou cliente é a entidade que vai utilizar um serviço criado por um provedor de serviço. O cliente é capaz de utilizar o serviço por meio das informações encontradas na descrição realizada pelo provedor. A busca

pelo serviço adequado é feita através de uma consulta a um registro de serviços. O registro de serviços é a entidade com a qual ambos, o provedor e o cliente, interagem. Os provedores publicam seus serviços no registro para que os clientes consigam encontrá-los e utilizá-los. O registro de serviços é essencialmente um repositório de dados baseado em XML.

Vale ressaltar que os *web services* são baseados em três padrões web fundamentais: SOAP, WSDL e UDDI. O *Simple Object Access Protocol* (SOAP) (SCHULTE; NATIS, 2006) é um protocolo de comunicação baseado em XML para a interação de aplicações. Nos *web services*, o SOAP é utilizado para definir a estrutura das mensagens trocadas entre as entidades participantes. A linguagem *Web Services Description Language* (WSDL) (SCHULTE; NATIS, 2006) é um padrão da W3C utilizado para descrever as interfaces dos *web services*. O (UDDI) - *Universal, Discovery, Description and Integration* (SCHULTE; NATIS, 2006) é um padrão da OASIS[1] que fornece aos usuários uma forma unificada e sistemática para descobrir serviços por meio de um registro centralizado. Em resumo, um *web service* pode ser definido como um serviço de *software* publicado na web, que troca mensagens utilizando o protocolo SOAP, é descrito por um arquivo na WSDL e registrado em um repositório conhecido como UDDI.

Ainda que a *interface* WSDL permita a descrição de atributos funcionais dos *web services* (LUDWIG, 2003), há uma carência em permitir que outros atributos não-funcionais e transacionais também sejam anexados. Permitir que atributos não funcionais (tempo de resposta, vazão, segurança, etc) sejam anexados à WSDL seria uma possibilidade para melhorar a provisão de QoS em *web services*. Como um quesito essencial para a computação orientada a serviço, QoS tem-se demonstrado um fator difícil de se gerenciar devido às diferenças entre as organizações e localização dos *web services* (LUDWIG, 2003). Nesse sentido, QoS e avaliação de desempenho para *web services* são assuntos pesquisados na atualidade, atraindo tanto pesquisadores da academia quanto da indústria.

2.4 Modelos Formais

Várias técnicas formais de modelagem de comportamento de *software* têm sido objeto de pesquisa e desenvolvimento. Esta seção apresenta uma introdução sobre

[1]Organization for the Advancement of Structured Information Standards - `http://www.oasis-open.org`

algumas técnicas formais que vêm sendo referenciadas nas pesquisas relacionadas à verificação de sistemas ou programas de um modo geral.

2.4.1 Redes de Petri

As redes de Petri (RP) (MURATA, 1989) são uma ferramenta matemática e gráfica de uso geral que permite: modelar o comportamento de sistemas dinâmicos a eventos discretos; descrever as relações existentes entre condições e eventos; e visualizar propriedades tais como paralelismo, sincronização, compartilhamento de recursos e desempenho. Elas foram criadas em 1962 por Carl Adam Petri (PETRI, 1962).

As redes de Petri são utilizadas para representar modelos conceituais de um sistema real. Conforme (HEUSER, 1990), um modelo conceitual é um modelo de uma área de uma organização, o qual não envolve detalhes de implementação e descreve tanto as propriedades estáticas quanto as propriedades dinâmicas do sistema modelado.

Conforme (CARDOSO; VALETTE, 1997), redes de Petri são aplicadas para avaliação de desempenho, análise e verificação formal de sistemas discretos, tais como: protocolos de comunicação, controle de oficinas de fabricação, concepção de software em tempo real e/ou distribuído, sistemas de informação (organização de empresas), sistemas de transporte, logística, gerenciamento de base de dados, interface homem-máquina e multimídia.

Na etapa de modelagem e análise da presente tese, foram utilizadas:

- **As redes de Petri Coloridas**, usando a ferramenta CPNTools [2];
- **As redes de Petri Estocásticas Generalizadas**, usando a ferramenta GreatSPN [3].

A seguir, serão detalhados os conceitos e extensões das redes de Petri.

O modelo original das redes de Petri é limitado quando se deseja representar duas importantes características: aspectos funcionais complexos, tais como condições que determinam o fluxo de controle e informação; e os aspectos de temporização. Para enfrentar estas duas limitações, duas classes de extensões às RP foram desenvolvidas:

[2] (`http://wiki.daimi.au.dk/cpntools/cpntools.wiki`)
[3] (`http://www.di.unito.it/~greatspn`)

- As RPs de alto-nível (aqui são enfatizadas as Redes de Petri Coloridas, que formam a categoria mais representativa e mais usada das Redes de Petri de alto-nível);
- As RPs com restrições de tempo (dentre elas, as redes de Petri temporizadas: determinísticas e estocásticas generalizadas) (PENHA; FREITAS; MARTINS, 2004).

2.4.1.1 Redes de Petri Coloridas

As redes de Petri são uma excelente ferramenta para modelagem de sistemas. Entretanto em sistemas complexos, onde existem vários produtos distintos com processos semelhantes, elas apresentam algumas limitações. Nestes casos, ao se utilizar redes de Petri, o projetista tem duas escolhas, como apresentado em (CARDOSO; VALETTE, 1997):

- Modelar o comportamento geral, sem indicar a identidade de cada processo, mas somente seu número; e
- Modelar, individualmente, cada um dos processos que constituem o sistema e modelar a interação existente entre eles, o que consiste, muitas vezes, em desdobrar o modelo que representa o comportamento geral.

No primeiro caso, é evidente a falta de precisão do modelo criado, pois muito do comportamento do sistema é perdido. Já no segundo, o modelo obtido tende a ser demasiadamente grande e complexo para ser entendido. Para resolver tal problema, foram propostas várias extensões às Redes de Petri. Estas abordagens recebem o nome de Redes de Petri de Alto Nível. Tais redes têm sua notação gráfica e definições padronizadas na norma ISO 15909[4].

Uma destas extensões são as Redes de Petri Coloridas (RPC), descritas em (JENSEN; KRISTENSEN; WELLS, 2007). O principal objetivo das RPC é a redução do tamanho do modelo, permitindo que fichas individualizadas (coloridas) representem diferentes processos ou recursos numa mesma subrede. Elas recebem este nome porque as fichas contêm dados capazes de distingui-las umas das outras em contraponto com às redes de Petri padrão, onde as fichas são indistinguíveis.

[4][ISO/IEC 15909 2002]

Em RPC, as fichas são representadas por estruturas de dados complexas. Desse modo, as fichas podem conter informações. Além disso, cada lugar armazena fichas de um certo tipo definido e arcos realizam operações sobre elas. As transições determinam a dinâmica da RPC e podem apresentar "expressões de guarda". Estas, por sua vez, indicam os tipos de fichas que possibilitam habilitar uma transição. Para que uma transição esteja habilitada em uma RPC, devem existir fichas suficientes nos lugares de entrada da transição. Além disso, estas fichas devem possuir valores que sejam correspondentes às expressões associadas aos arcos que ligam estes lugares à transição.

Uma RPC é composta por três partes: estrutura, inscrições e declarações.

- A estrutura é um grafo direcionado, com dois tipos de nós (lugares e transições), com arcos interconectando nós de tipos diferentes. Graficamente, os lugares são representados por círculos ou elipses, e transições, simbolizadas por retângulos.
- As inscrições são associadas aos lugares, transições e arcos.
- As declarações envolvem tipos, funções, operações e variáveis. Quando a expressão do arco é avaliada, ela gera um multi-conjunto (conjunto com várias cópias de um mesmo elemento) de fichas coloridas.

As RPC suportam hierarquia. Dessa forma, pode-se construir um modelo mais abstrato e, através da hierarquia, refinar o modelo. As RPC usam transições especiais, denominadas transições de substituição, para implementar hierarquias.

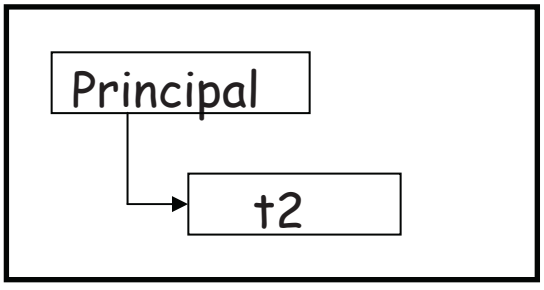

Figura 2.6: Hierarquia em RPC de um sistema de manufatura.

Na Figura 2.6, é apresentada a hierarquia do modelo RPC de um sistema de manufatura apresentado na Figura 2.7. Nessa hierarquia, são apresentadas redes

que compõem o modelo, as quais são: a rede Principal e a sub-rede t2.

Na Figura 2.7, é apresentada a página Principal do modelo e, na Figura 2.8, é apresentada a sub-rede associada à transição de substituição t2.

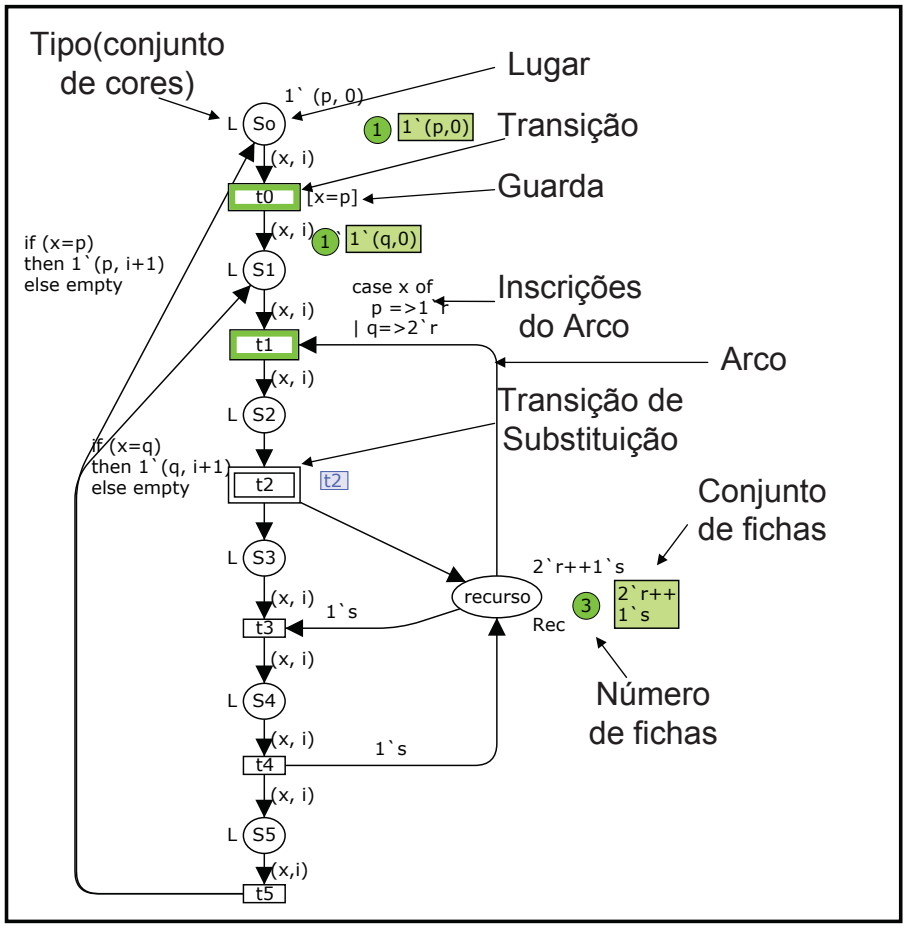

Figura 2.7: Descrição em RPC da página Principal de um sistema de manufatura.

A seguir, são descritos os elementos de uma RPC (Figura 2.7):

- *Lugares*, representados graficamente por um círculo ou uma elipse. Os mesmos contêm as seguintes inscrições:
  - *Nome*: para identificação (por exemplo, o lugar s_0 na RPC apresentada na Figura 2.7);
  - *Tipo* (conjunto de cores): especificando os tipos de fichas que podem residir no lugar (por exemplo, o tipo L associado ao lugar s_0);
  - *Número de fichas*: quantidade de fichas coloridas em um determinado lugar em um dado instante (por exemplo, o lugar *recurso* (Figura 2.7)

possui 3 fichas do tipo *Rec*);

- *Conjunto de fichas*: exemplo de conjunto de cores: 2'r++1's são fichas do tipo *Rec*, duas fichas de valor r e uma de valor s;
- *Portas de Entrada* (In): lugares de entrada das transições de substituição; e
- *Portas de Saída (Out)*: lugares de saída das transições de substituição.

▶ *Transições*, representadas graficamente por retângulos ou seguimentos de retas, contêm as seguintes inscrições:

- *Nome*: para identificação (por exemplo, a transição t_0 na RPC apresentada na Figura 2.7); e
- *Expressões de Guarda*: por exemplo, a condição $[x = p]$ associada â transição t_0.

▶ *Arcos*, ligando transições a lugares e lugares a transições. Os arcos contêm *Inscrições do arco* que definem a habilitação de uma transição e as fichas que serão removidas e adicionadas após o disparo dessa transição (por exemplo, a inscrição *case x of p=>1´r | q=>2´r* associada ao arco que liga o lugar *recurso* à transição t_1);

▶ *Transição de substituição*, representada graficamente por um duplo retângulo. Esta transição representa um modelo que deve ser expandido via uma sub-rede associada à essa transição (por exemplo, a transição t_2 da RPC apresentada na Figura 2.7); e

▶ *Marcação Inicial*, representando a distribuição inicial de fichas nos lugares da rede.

No modelo apresentado na Figura 2.7, os estágios de produção são representados pelos lugares de s_0 a s_5. As fichas de cores (tipos) distintas diferenciam os processos. Os recursos compartilhados são representados por fichas de cores distintas no lugar *recurso*.

A marcação inicial da rede é:

▶ M(s_0) = 1'(p,0);

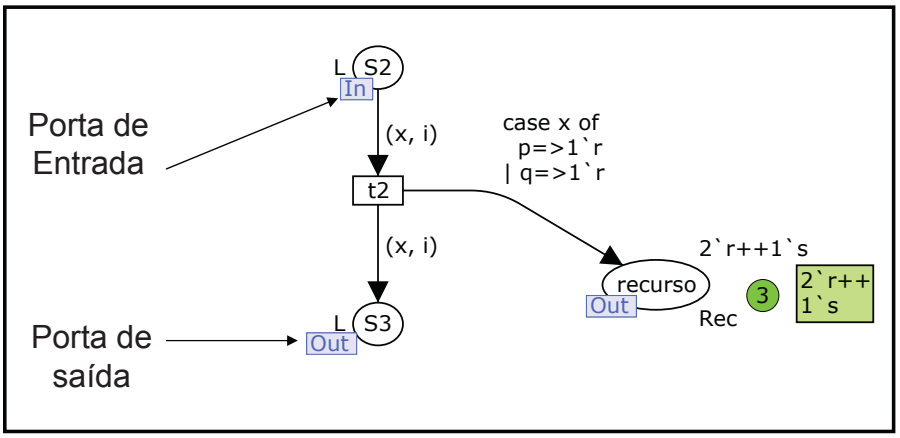

Figura 2.8: Descrição em RPC da página t2 de um sistema de manufatura.

- M(s_1) = 1'(q,0); e
- M(recurso) = 2'r + 1's.

Desse modo, o lugar s_0 possui uma ficha do tipo Proc*I, ou seja, 1'(p,0), onde o tipo do processo é p e o zero corresponde ao número de iterações realizadas. No lugar *recurso*, existem inicialmente duas fichas de valor r e uma de valor s.

Inicialmente, as transições t_0 e t_1 estão habilitadas. Para que a transição t_1 seja disparada, a variável x do arco que liga o lugar *recurso* à transição t_1 deve receber uma ficha de valor q. O disparo da transição t_1 remove uma ficha de valor (q,0) do lugar s_1 e duas fichas de valor r do lugar *recurso* e coloca uma ficha de valor (q,0) em s_2.

2.4.1.2 Redes de Petri Temporizadas Determinísticas

O conceito de tempo não foi contemplado na definição original das redes de Petri. Contudo, para avaliação de desempenho, é necessária a introdução de retardos de tempo (RAMCHANDANI, 1974). Conforme a proposta de Ramchandani, as formas pelas quais o tempo pode ser especificado são:

- **Associado aos lugares**: onde as fichas armazenadas nos lugares de saída, após o disparo de uma transição, só estarão disponíveis para disparar uma nova transição após um determinado tempo;
- **Associado às fichas**: as fichas possuem uma informação que indica quando esta estará disponível para disparar a transição; e

- **Associado à transição**: que impõe a cada transição um retardo (ou duração) fixo entre o tempo em que ela está habilitada e o tempo de disparo.

Um exemplo de rede de Petri temporizada determinística é apresentado na Figura 2.9.

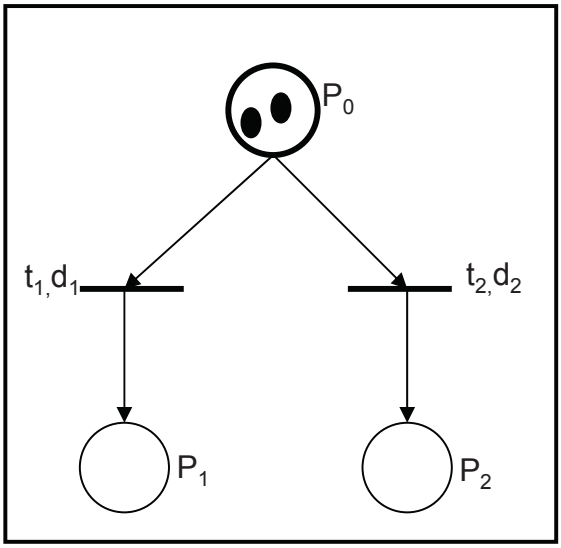

Figura 2.9: Exemplo de Rede de Petri Temporizada.

Nela, as transições t_1 e t_2 possuem tempos associados diferentes (respectivamente, d_1 e d_2). Supondo que $d1 < d2$, então dada a marcação apresentada na Figura 2.9 (2 fichas em P_0), t_1 irá disparar primeiro e uma ficha chegará primeiro ao lugar P_1. Assim, de maneira determinística, pode-se estabelecer a ordem em que os eventos devem ocorrer.

2.4.1.3 Redes de Petri Estocásticas Generalizadas

As Redes de Petri Estocásticas Generalizadas (GSPN) (CHIOLA et al., 1993) incluem transições temporizadas e não-temporizadas, particionando as transições em dois conjuntos:

- **Transições Imediatas**: uma vez habilitadas, disparam em tempo zero; e
- **Transições Temporizadas**: disparam com tempos aleatórios, descritos por distribuições exponenciais. Nas GSPN, as taxas de disparo estão associadas somente às transições temporizadas.

A definição formal das GSPN é semelhante à das Redes de Petri Estocásticas (BERNARDI; DONATELLI, 2004), diferenciando somente no conjunto das taxas de disparo ou atrasos (λ). Nessa extensão, as expressões de eventos são acrescidas de um intervalo [1, λ], que determina as fronteiras temporais mínima e máxima do disparo da transição. Os parâmetros temporais associados à transição determinam o tempo durante o qual o sistema permanecerá no conjunto de estados de origem, bem como o atraso permitido para que a transição dispare após a fronteira mínima ter sido alcançada. Nas GSPN, esses atrasos são associados somente às transições temporizadas, e não mais a todas as transições, como na redes de Petri Estocásticas.

Segundo Chiola e outros, a diferença fundamental entre as redes de Petri estocásticas e as redes de Petri estocásticas generalizadas está em admitir que as transições também podem ser não-estocásticas, isto é, uma transição também pode ser imediata, como nas redes de Petri. Chiola et al (1993) definiram que as transições imediatas deveriam ter atraso de disparo igual a zero, e que somente as transições estocásticas tinham atrasos associados diferentes de zero.

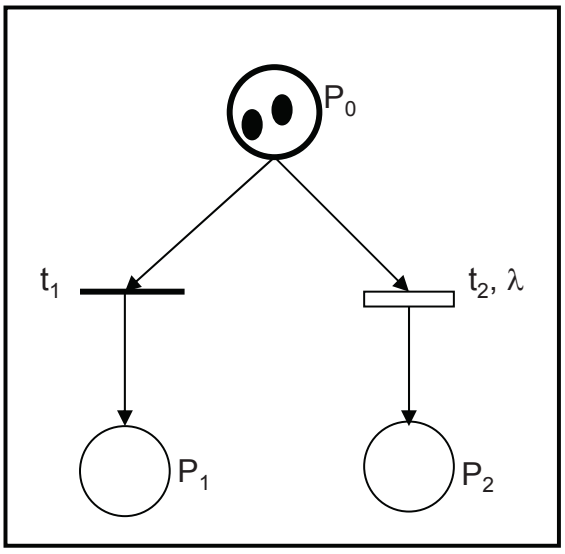

Figura 2.10: Exemplo de Rede de Petri Estocástica Generalizada.

Um exemplo de rede de Petri estocástica generalizada é apresentado na Figura 2.10. Nela, a transição t_1 é imediata e a transição t_2 é temporizada, com atraso de λ.

2.4.2 Álgebras de Processos

Álgebras de processos são linguagens formais usadas para especificar sistemas de *software* (BERA; GHOSH, 2009), principalmente aqueles constituídos de componentes concorrentes que se comunicam. Essas linguagens provêem um meio para descrever interações em alto nível, comunicações e sincronizações entre uma coleção de agentes ou processos independentes. São definidas leis que permitem manipular e analisar descrições algébricas dos processos como, por exemplo, verificar equivalências usando bissimulação. Exemplos dessas álgebras incluem CSP (*Communication Sequential Processes*), CCS (*Calculus of Communicating Systems*), ACP (*Algebra of Communicating Process*), *Lotos*, *Promela*, *Pi-Calculus* e *Ambient Calculus*.

Além de existir uma variedade de álgebras de processo, essas também têm variantes que incorporam comportamentos estocásticos, informações de tempo, etc. Porém, as principais funcionalidades comuns a todas elas são (BERA; GHOSH, 2009):

- representar interações entre processos independentes através de comunicação (mensagens), ao invés de variáveis compartilhadas;
- descrever processos e sistemas usando um conjunto pequeno de primitivas e operadores que combinam essas primitivas; e
- definir leis algébricas para os operadores de processos permitindo que expressões de processos sejam manipuladas algebricamente.

Existem algumas áreas em que as pesquisas em álgebra de processos são aplicadas: aquelas relacionadas ao estudo de comportamento dos processos e técnicas de modelagem. A primeira trata de, a partir de uma descrição do processo, efetuar a análise e verificação do seu comportamento. A segunda é empregada para modelar processos a partir de suas especificações.

A álgebra de processos inicia-se com um conjunto de nomes (canais) que servem como meio de comunicação. Em algumas propostas, canais têm estruturas mais complexas, mas isso é abstraído nos modelos teóricos. Adicionalmente, há a necessidade de se formar novos processos. As principais operações comuns presentes na maioria das álgebras são (BERA; GHOSH, 2009):

- composição de processos paralelos;

- envio e recepção de mensagens através de canais;
- composição sequencial;
- pontos de interações locais (sem interferência de outros processos); e
- recursão e replicação de processos.

O uso de canais de comunicação é uma das características que distingue álgebra de processos de outros modelos formais como rede de Petri, tratada na seção 2.4.1.

Pi-Calculus é a álgebra de processo usada para a verificação de processos. Caracteriza-se principalmente por permitir a mobilidade dos processos de um sistema.

2.4.3 Linguagem Z

A linguagem Z é uma notação de especificação formal usada para descrever e modelar sistemas computacionais, baseada em teoria dos conjuntos e lógica predicada de primeira ordem. A linguagem foi desenvolvida pelo *Programming Research Group* da Universidade de Oxford na década de 1970. A linguagem foi padronizada pela ISO (*International Standard Organization*), sob o código ISO/IEC 13568 (primeira edição em 2002) (YUAN et al., 2007).

Uma característica da linguagem Z é o uso de tipos de dados matemáticos para modelar dados em um sistema. Esses tipos não são orientados às representações em computadores; eles obedecem a uma rica coleção de leis matemáticas que tornam possível descrever efetivamente o comportamento de um sistema.

A linguagem Z é definida de maneira a permitir a decomposição de uma especificação em partes menores chamadas esquemas. Os esquemas descrevem os aspectos estáticos e dinâmicos de um sistema. Os aspectos estáticos incluem estados e relações invariantes que são mantidas quando um sistema muda de um estado para outro. As propriedades dinâmicas são: operações possíveis, relações entre entradas e saídas e mudanças de estado.

A linguagem Z permite que diferentes aspectos de um sistema sejam descritos separadamente e posteriormente relacionados e combinados. Por exemplo, o funcionamento de um sistema, ao receber uma entrada válida, pode ser primeiramente descrito e depois ser estendido para mostrar o tratamento de erros. Os esquemas para definição das extensões não alteram os já construídos.

Esquemas podem ser usados para descrever uma transformação de uma perspectiva para outra, permitindo explicar porque uma especificação abstrata está corretamente implementada por uma outra que contém mais detalhes de um projeto mais concreto. Ao se construir uma sequência de especificações em que cada uma contém mais detalhes que a anterior, obtém-se uma maior confiabilidade no programa final no que se refere à sua especificação (FARREL, 2003).

Por exemplo, pode-se considerar a especificação do funcionamento de uma base de dados que tem operações de leitura de uma página, escrita de uma página, *backup e restore*. Em um primeiro nível, pode-se descrever essas operações considerando-se que a base de dados tem duas estruturas iguais: uma para trabalho e outra para *backup*; num segundo nível, reescrevem-se todas as operações definindo um esquema em que se tem uma cópia mestre e um registro das mudanças. O segundo nível é uma especialização do primeiro que adiciona uma propriedade de funcionamento em direção à implementação. A linguagem Z permite provar que o funcionamento das operações do segundo nível é matematicamente equivalente ao primeiro.

O uso de tipos de dados matemáticos para modelar dados em um sistemade é uma das características que distingue a linguagem Z de outros modelos formais como rede de Petri, tratada na seção 2.4.1.

2.5 Conclusões do Capítulo

Neste capítulo foram apresentados os principais conceitos utilizados no desenvolvimento da arquitetura "SOASPE" (objetivo deste trabalho): SPE, SOA, *web services* e modelos formais. A SPE é uma metodologia cuja abordagem é a utilização de métodos de transformação de *softwares* em modelos formais, com o objetivo de utilizar estes modelos para testar a arquitetura do sistema, design, desempenho ou implementação deste. Esses modelos são construídos e analisados, dentre outros objetivos, para fornecer um *feedback* sobre se as propostas de *software* são susceptíveis de permitir alcançar suas metas de desempenho. SOA é uma arquitetura que tem como objetivo alinhar o mundo dos negócios às soluções em tecnologia da informação. Foram descritos os diversos requisitos para se implantar uma arquitetura SOA utilizando *web services*. A pilha de especificações da arquitetura SOA inclui a camada de composição de *web services*. É nessa camada que a linguagem BPEL é aplicada. Foram apresentados também os principais modelos de descrição de comportamento de *software*, como a base teórica para a modelagem

formal de processos, ou seja, redes de Petri, álgebra de processos e linguagem Z.

O Capítulo 3 contém uma descrição sobre BPEL e BPEL4People, complementando os conceitos necessários para o desenvolvimento da arquitetura "SOASPE".

Capítulo 3

BPEL e BPEL4People

Processos de negócios foram concebidos para tornar a coordenação de atividade mais fácil e mais rentável. BPEL, apresentado na Seção 3.1, e sua extensão BPEL4People, apresentada na Seção 3.2, juntos coordenam os WS e atividades humanas dos processo de negócio. Com a globalização, as organizações tornam-se mais dinâmicas e os processos de negócios subjacentes são frequentemente mais otimizados no mundo empresarial de hoje. Adaptar os processos de negócio às mudanças do mercado e serviços prestados às empresas e automatizar sob demanda são as necessidades principais para facilitar a colaboração entre parceiros de negócios existentes e potenciais (AALST; HEE, 2004).

No ambiente de negócios orientado por processos, uma linguagem de especificação unificada de processo é significativamente importante em termos de colaboração. BPEL é uma linguagem que fornece a sintaxe e notações para a especificação de processos de negócios baseados em WS. Além de WS automáticos, a maioria dos processos de negócios requerem interações humanas. BPEL4People aborda este aspecto importante de prover suporte a ações humanas nos WS (HOLMES; VASKO; DUSTDAR, 2008; ADOBE et al., 2007; BERTINO, 2006).

3.1 Conceitos Fundamentais de BPEL

A linguagem de execução de processos de negócio - *Business Process Execution Language* (BPEL) é um padrão criado pela *Organization for the Advancement of Structured Information Standards* (OASIS) para a execução de processos de negócios, descrevendo como ocorre o relacionamento entre os diversos serviços Web

participantes de uma composição. Ela permite a composição de serviços web com o objetivo de realizar funcionalidades mais complexas (BERTINO, 2006).

A primeira versão de BPEL foi desenvolvida em agosto de 2002 pela IBM, BEA e Microsoft, após a SAP e a Siebel se juntarem ao consorcio que estava desenvolvendo o BPEL e foi lançado em março de 2003 a versão 1.1. Em abril de 2003, a OASIS recebeu uma versão para padronização e em abril de 2007 foi lançada oficialmente a versão 2.0 com o nome de WS-BPEL 2.0 (FU; BULTAN; SU, 2004; BIANCULLI; GHEZZI; SPOLETINI, 2007).

A programação com BPEL é parecida com a programação das linguagens de programação já existentes, na medida que ela oferece determinados tipos de construções como estruturas de repetição, estruturas condicionais, variáveis e atribuições. Este fato possibilita que o processo de negócio seja visto como um algoritmo. Porém, o uso desses recursos é bem simples e limitado, objetivando a facilidade de uso e aprendizagem da mesma (ZHAO; KRISHNAN, 2006).

Uma composição de serviços especificada em BPEL é vista como serviço, ou seja, quem usou um processo de negócio definido em BPEL não tem a ciência que na verdade está usando vários serviços para realizar uma determinada operação. Por ser um serviço web, a composição necessita ter sua interface baseada no padrão WSDL, troca de mensagens com outros serviços web através do padrão SOAP e pode ser cadastrado em um repositório UDDI para fins de consulta. O padrão BPEL assume que os serviços participantes da composição estejam descritos usando WSDL (MOLLOY, 1981).

Um processo BPEL pode ser síncrono ou assíncrono (ZHAO; KRISHNAN, 2006). Um processo BPEL síncrono bloqueia o cliente (aquele que está usando o processo) até que o processo termine e retorne um resultado para o cliente. Um processo assíncrono não bloqueia o cliente. Em vez disso, ele usa um retorno de chamada para retornar o resultado (se houver). Geralmente, os processos assíncronos são usados para processos de longa duração, e os síncronos são usados para processos que retornam um resultado em um tempo relativamente curto. Se um processo BPEL usa serviços web assíncronos, o próprio processo geralmente também é assíncrono (ZHAO; KRISHNAN, 2006).

A propriedade mais importante do BPEL está relacionada à invocação de serviços web. BPEL permite que as operações de um serviço web sejam invocadas sincronamente ou assincronamente de maneira mais fácil. Estas operações podem ser

invocadas sequencialmente ou paralelamente. As funcionalidades mais importantes do BPEL são listadas a seguir (KOSHKINA et al., 2003):

- descrever a lógica dos processos de negócio através da composição de serviços;
- tratar as invocações de operações de modo síncrono ou assíncrono;
- invocar serviços de maneira sequencial ou paralela;
- prover mecanismos para tratar erros durante a invocação de serviços;
- manter atividades de longa duração (transacionais), assim como interrompê-las;
- reiniciar uma composição que foi interrompida ou apresentou erros; e
- agendar a execução de atividades e definir a ordem em que elas irão executar.

A estrutura geral de um processo de negócio especificado em BPEL é formada por quatro seções: *partnerLinks*, variáveis, *faultHandlers* e atividades, conforme descritas a seguir (ARIAS-FISTEUS; FERNÁNDEZ; KLOOS, 2004):

- ***partnerLinks***: seção que define os diferentes parceiros (partes envolvidas) que interagem com o processo de negócio durante toda sua execução. Eles são usados para identificar a funcionalidade que deve ser oferecida por cada serviço parceiro. As ligações de parceiros (*partnerLink*) devem estar associadas a um tipo de ligação entre parceiros (*parterLink type*) definidos na especificação de serviços web em WSDL;
- **variáveis**: seção que define as variáveis de dados usadas pelo processo de negócio. As definições são feitas em termos de tipos de mensagem WSDL, elementos ou tipos simples de esquemas XML. Elas são usadas para manter os dados de estado e o histórico do processo com base nas mensagens trocadas. As variáveis devem estar associadas a tipos de mensagens (*messages*) definidos na especificação WSDL;
- ***faultHandlers***: seção que contém os tratadores de falhas que definem as atividades a serem executadas em resposta às falhas resultantes da invocação de serviços de avaliação e de aprovação;

Tabela 3.1: Primitivas básicas usadas em uma composição de serviço BPEL.

PRIMITIVA	SIGNIFICADO
<Invoke>	Invoca um Serviço Web.
<Receive>	Aguarda a resposta de um cliente.
<Reply>	Gera resposta síncrona.
<Assign>	Manipula dados
<Wait>	Espera por certo tempo.
<Terminate>	Finaliza um processo.
<Sequence>	Sequência de atividades a serem invocadas.
<Flow>	Usada para definir um conjunto de atividades que podem ser invocadas em paralelo.
<while>	Repetição.
<Pick>	Aguarda por um evento.

- **atividades**: seção que contém a descrição do comportamento normal para a execução do processo de negócio. Existem basicamente dois tipos de atividades:

  - atividade básica: tipo de atividade usado para executar alguma operação. Algumas dessas atividades básicas envolvem a interação com algum parceiro, as quais são: *<Invoke>*, *<Receive>* e *<Reply>*. Outras dessas atividades básicas são executas sem a interação com qualquer parceiro, as quais são: *<Terminate>*, *<Assign>*, *<Empty>*, *<Compensate>*;
  - atividade estruturada: tipo de atividade usado para agrupar atividades básicas dentro de algumas estruturas de fluxo. Tais atividades são: *<While>*, *<Pick>*, *<Flow>*, *<Sequence>* e *<Switch>*.

As atividades (primitivas XML) que formam a gramática da linguagem BPEL são interpretadas e executadas por um mecanismo de orquestração (KOSHKINA et al., 2003).

Na Tabela 3.1 estão descritas de maneira sucinta as atividades mais usadas numa composição de serviço BPEL.

3.1.1 Síntese do Funcionamento do BPEL

Conforme (VERBEEK, 2005), pode-se resumir o funcionamento do BPEL da seguinte maneira, Figura 3.1:

- *partnerlinks* são partes que interagem com o processo BPEL; e
- *portTypes* são usados para acesso a serviços do próprio processo BPEL.

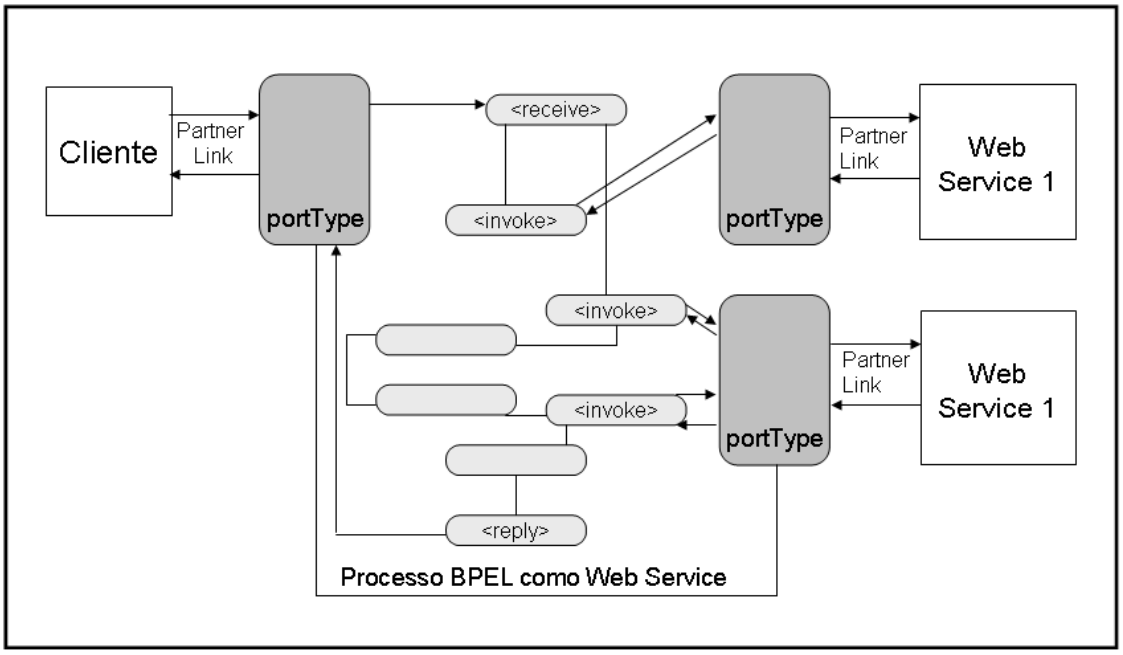

Figura 3.1: Funcionamento do BPEL (VERBEEK, 2005).

3.2 Conceitos Fundamentais de BPEL4People

BPEL orquestra os processos de negócio que compõem as interações *web service*. No entanto, em geral, os processos de negócios são compostos por um amplo espectro de atividades que exigem muitas vezes a participação de pessoas para executar tarefas, tais como, revisão ou aprovação medidas e inserir dados. Essas interações humanas são abordadas por um novo tipo de espeficicação a BPEL4People (HOLMES; VASKO; DUSTDAR, 2008).

Interações humanas não estão atualmente abrangidas pelo BPEL, que é principalmente destinado a apoiar processos de negócios automatizados baseados em

WS. Na prática, porém, muitos cenários de processos de negócio requerem interação humana.

Pode-se verificar que a interação humana nos processos de negócio pode se tornar bastante complexa. Embora a especificação BPEL 1.1 (e sua atualização BPEL 2.0) não abranjam especificamente a interação humana, BPEL é adequado para fluxos de trabalho humano. Serviços de fluxo de trabalho que utilizam o suporte BPEL para serviços assíncronos são criados hoje em dia. Desta forma, interações humanas se tornam apenas mais um serviço assíncrono a partir da perspectiva do processo de orquestração e assim sendo tornando-os compatíveis com os processos BPEL (HOLMES; VASKO; DUSTDAR, 2008).

Vemos agora a próxima geração das especificações de fluxo de trabalho emergentes do BPEL, com o objetivo de padronizar a inclusão explícita das tarefas humanas em processos BPEL. Esta proposta é chamada BPEL4People e foi originalmente apresentada pela IBM e SAP, em Julho de 2005. Outras empresas, como Oracle, também manifestaram o interesse em participar e apoiar esse esforço.

As extensões mais importantes introduzidas no BPEL4People foram: *People activities* e *People links* (DYER; ZULAUF, 2005). *People activities* é uma nova atividade BPEL usada para definir as interações humanas, em outras palavras, as tarefas que um usuário tem para executar. Para cada *People activities*, o servidor BPEL deve criar itens de trabalho e distribuí-los aos usuários qualificados para executá-los. *People activities* podem ter variáveis de entrada e de saída e pode especificar prazos.

Para especificar a implementação das *People activities*, BPEL4People introduz tarefas (*tasks*). Tarefas especificam ações que o usuário pode invocar. Tarefas podem ter descrição, prioridades, atrasos e outras propriedades. Para apresentar as tarefas para os usuários, precisa-se de uma aplicação cliente que forneça uma interface de usuário com a qual ele interaja com as tarefas. Com esta interface o usuário pode reivindicar consulta a tarefas disponíveis, revogá-las e completá-las ou não.

Para associar as *People activities* e as tarefas relacionadas com os usuários ou grupos de usuários, BPEL4People introduz o *People links*. *People links* são semelhantes aos *partner links*. Eles associam usuários com uma ou mais *People activities* (ADOBE et al., 2007).

BPEL4People amplia os recursos do BPEL para suportar uma ampla gama de

padrões de interação humana, permitindo a expansão da modelagem de processos de negócio dentro da linguagem BPEL.

BPEL4People é composto de duas especificações (ADOBE et al., 2007):

- *WS-BPEL Extension for People*: que possui camadas em cima de BPEL para descrever tarefas humanas como atividades que podem ser incorporadas como componentes de primeira classe em definições de processos BPEL; e
- *WS-Human Task*: apresenta a definição de tarefas humanas autônomas, incluindo as propriedades de comportamento, e as operações usadas para manipulá-las. Capacidades fornecidas por *WS-Human Task* podem ser utilizadas por WS orquestrados pelo BPEL.

Nos Capítulos 4 e 5 serão utilizadas estas tecnologias (BPEL e BPEL4People) na definição da arquitetura "SOASPE" para a transformação de códigos BPEL4People em GSPN e em CPN e assim se possa avaliar seu desempenho, que é um dos objetivos deste trabalho de tese.

Capítulo 4

A arquitetura "SOASPE"

Os trabalhos relacionados citados na Seção 1.3 mostraram que houve uma série de estudos e pesquisas acerca da avaliação de desempenho de *web services*. Contudo, a maioria desses estudos e pesquisas promovem a avaliação do desempenho de *web services* centrado na otimização de sua composição.

Nesse trabalho de tese pretende-se abordar a questão relativa à avaliação de desempenho dos *web services* baseada na execução de processos orquestrados onde exista a inclusão explícita das tarefas humanas em processos BPEL (extensão para pessoas, chamada de BPEL4People) e sendo estes processos transformados em duas extensões de redes de Petri as GSPN e as CPN.

Uma arquitetura denominada de "SOASPE" (HOLANDA G.C. BARROSO, 2009b), a origem do termo "SOASPE" vem da junção das siglas SOA e SPE, é proposta com os fins de análise de desempenho de *web services* orquestrados com o BPEL4People. A estrutura desta arquitetura (ver Figura 4.1) é composta por cinco camadas:

- camada SOA;
- camada BPEL;
- camada de transformação;
- camada rede de Petri; e
- camada de desempenho.

A arquitetura "SOASPE" é baseada nos princípios da SOA, onde um processo de negócio - *Business Process* (BP) é composto de um ou mais serviços e cada serviço, por sua vez, pode ser composto de vários subserviços. A execução dos processos de negócio, que executam nos provedores de serviços - *Services Providers* (SP), são coordenados por um Processo de Negócio Integrador (PNI) - *Business Process Integrator* (BPI).

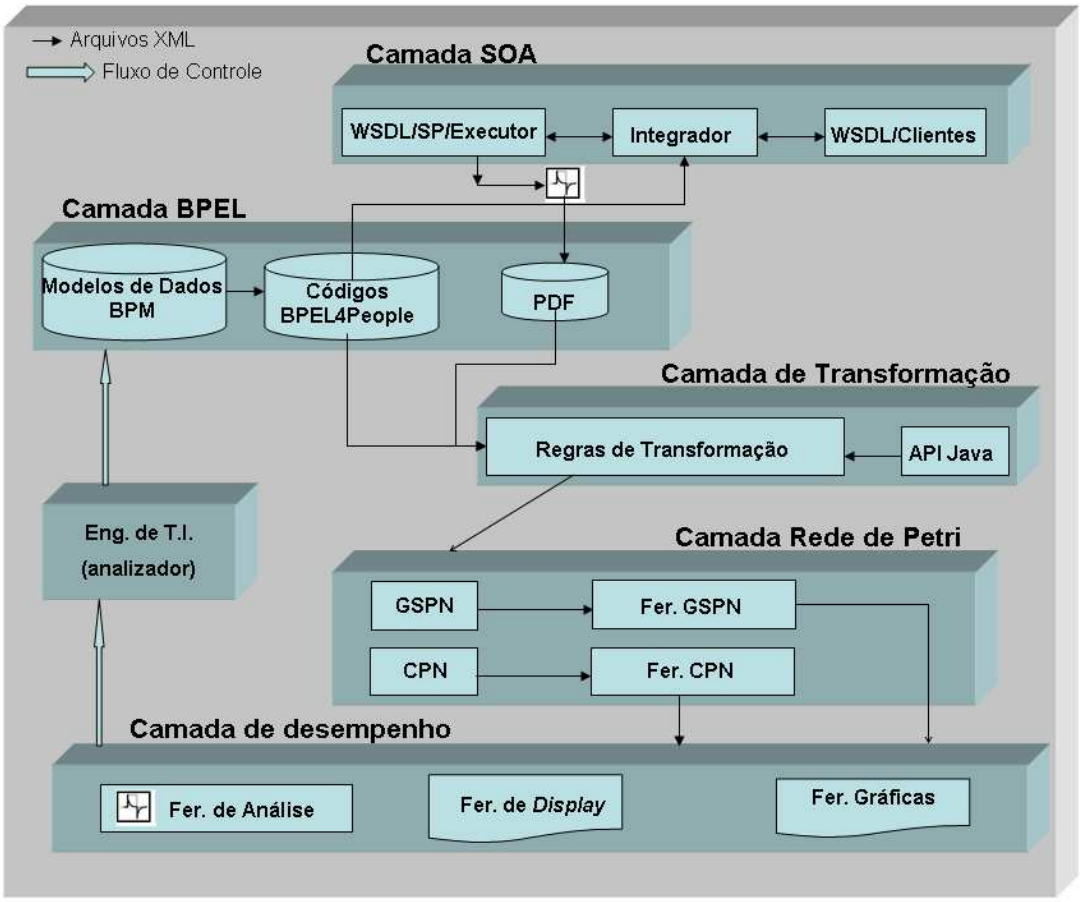

Figura 4.1: Arquitetura "SOASPE" (HOLANDA G.C. BARROSO, 2009b).

Os serviços que compõem os processos de negócios dos WS podem ser de dois tipos: básicos e orquestrados (HOLANDA G.C. BARROSO, 2009b).

Os serviços básicos são os serviços que são processados pelos sistemas informáticos pertencentes a um provedor de serviços que retornam uma mensagem XML (*Extensible Markup Language*) como um resultado do processamento.

Os serviços orquestrados são serviços que são códigos (BPEL ou BPEL4People) que servem para orquestrar processos de negócios que compõem o processo de negócio integrador.

A composição dos processos de negócio é mostrada na Figura 4.2.

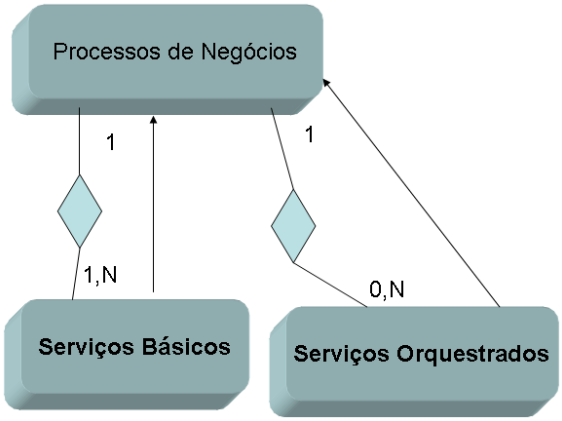

Figura 4.2: Composição dos processos de negócios (HOLANDA G.C. BARROSO, 2009b).

Os serviços estão disponíveis nos provedores de serviços (SP) e são descritos por um arquivo WSDL (*Web Services Description Language*) e registrados em um repositório conhecido como UDDI (*Universal Description, Discovery, and Integration*). Na sua maioria estes serviços são orquestrados com o BPEL4People para compor *web services*. Por ser importante a avaliação do desempenho destes WS baseada na execução de processos orquestrados com o BPEL4People, neste trabalho de tese foi concebida a arquitetura "SOASPE" com este propósito (HOLANDA G.C. BARROSO, 2010b, 2009b).

4.1 A Camada SOA

Um dos cenários possíveis de implementação de SOA é um sistema composto de um conjunto de provedores de serviços, um integrador que é um operador de um mecanismo de orquestração dos serviços e um conjunto de clientes para estes serviços, ou seja, consumidores de serviços. Este cenário é utilizado como base para a camada SOA da arquitetura "SOASPE", Figura 4.1.

A missão do integrador é executar o código BPEL4People do BPI dos WS. Na execução de um serviço de um processo de negócio, faz-se necessário o fornecimento, ao processo de execução do serviço, de informações necessárias para iniciá-lo, tais como, *links* de parceiros, endereços e finalmente fornecer o serviço para os clientes. Este código do BPI será escrito com BPEL4People e para poder avaliar o desempenho de sua orquestração, o mesmo será transformado em um modelo formal (redes de Petri).

A relação entre o integrador e prestadores de serviços, bem como entre o integrador e os clientes é baseada em *Service Level Agreements* (SLA) que, em particular, determina: preços e qualidade de serviço (QoS) como por exemplo desempenho.

4.2 A Camada BPEL

Na camada BPEL da arquitetura "SOASPE" mostrada na Figura 4.1, são encontrados:

- os modelos *Business Process Management* (BPM) de dados: uma vez que códigos BPEL4People não são muito amigáveis para os desenvolvedores, no que se refere à codificação, isto faz com que a maioria deles prefira modelar suas aplicações utilizando ferramentas de BPM;
- os códigos BPEL4People: estes códigos fazem a orquestração de *web services* onde exista interação humana. O padrão para modelagem de processos de negócio em SOA são as notações *Business Process Management Notation* (BPMN) para modelagem gráfica de processos de negócio e os códigos baseados em XML, como o BPEL4People, para a sua execução. A abordagem assim descrita é denominada de orquestração de *web service*. O principal componente de infra-estrutura de uma orquestração é o motor (*engine*) de BPEL4People que impulsiona a realização de processos de negócio através da execução das construções lógicas (algoritmos) determinadas nos códigos BPEL4People e possibilita também a comunicação dos serviços *web* com clientes. Os códigos BPEL4People são também usados para serem transformados em redes de Petri, possibilitando assim se fazer análise de desempenho, baseada na execução dos códigos BPEL4People de processos de negócios; e
- arquivos PDF (Função Distribuição de Probabilidade): estes arquivos contêm os valores dos tempos de resposta, de cada um dos provedores de serviços dos WS e os tempos de resposta de cada pessoa participante das interações humanas do WS, para serem aproximados por funções PDF.

4.3 A Camada de Transformação

Na camada de transformação estão contidas as regras de transformação de códigos BPEl4People em rede de Petri (GSPN e CPN).

Por ser BPEL4People uma extensão do BPEL, tratar-se-á da sua transformação em redes de Petri em dois momentos. Nas Seções 4.3.1 e 4.3.2 serão apresentadas as transformações de códigos BPEL em rede de Petri GSPN e em rede de Petri CPN, repectivamente, e no Capítulo 5 tratar-se-á especificamente da transformação da extensão humana "WS-HumanTask" de códigos BPEL4People nas duas extensões de rede de Petri (GSPN e CPN).

A funcionalidade da camada de transformação é ilustrada na Figura 4.3.

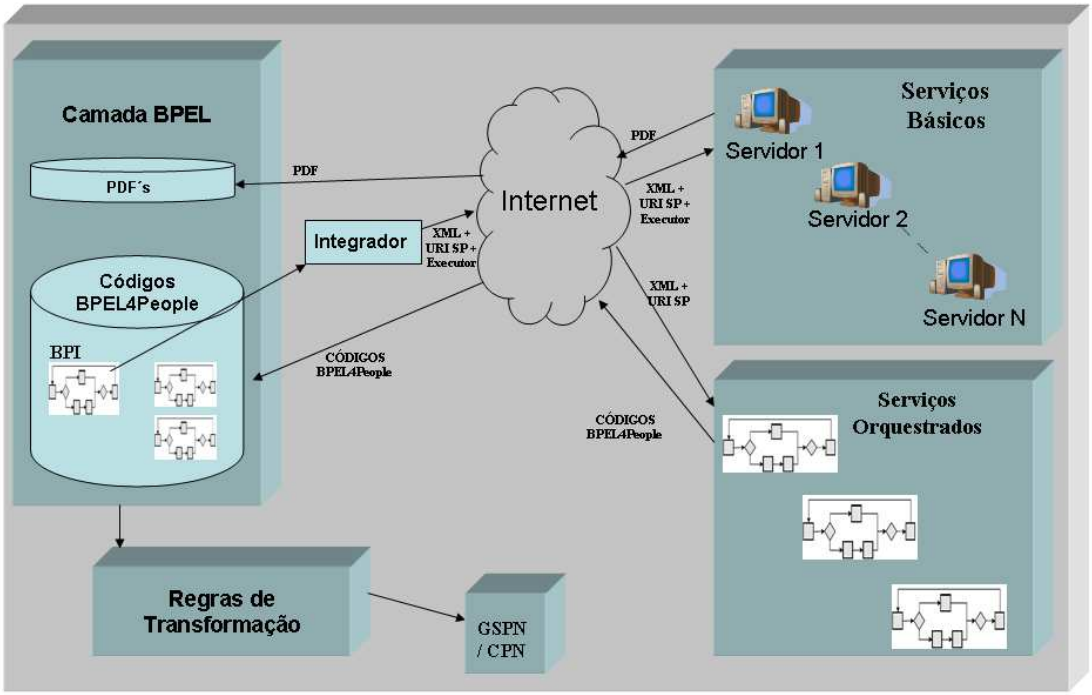

Figura 4.3: A funcionalidade da camada de transformação (HOLANDA G.C. BARROSO, 2009b).

A funcionalidade da camada de transformação estabelece que em todos os provedores de serviços que possuam serviços básicos que possam ser invocados pelo processo de negócio integrador (PBI), exista um aplicativo executando nestes provedores que guarde (em arquivos XML) todos os tempos de resposta desses provedores e os tempos de resposta dos possíveis executores destes serviços, quando estes serviços envolverem a participação humana.

Os arquivos de tempos de resposta dos diversos provedores de serviços e dos possíveis executores dos serviços serão enviados para compor a camada BPEL (arquivos PDF) e serão utilizados no cálculo, através de Função Distribuição de Probabilidade (PDF), de seus respectivos tempos de atraso nos modelos GSPN e

CPN.

Quando se tratar de serviços orquestrados, os códigos BPEL4People destes serviços serão enviados para comporem juntamente com o código BPI a base de códigos da camada BPEL.

Os arquivos PDF e os códigos BPEL4People serão utilizados pelas regras da camada de transformação para gerar a rede de Petri do WS orquestrado pelo BPI.

O outro componente da camada de transformação são as API Java. As API são adicionadas aos códigos JAVA produzidos a partir das regras de transformação para gerar os códigos fontes das redes de Petri (GSPN ou CPN).

4.3.1 Regras de Transformação de BPEL em GSPN

O objetivo desta seção é fornecer as regras de transformação do BPEL em GSPN. São apresentadas as representações das atividades básicas e das atividades estruturadas do BPEL em GSPN. Nesta transformação, o código BPEL que está presente no BPI e os códigos BPEL dos serviços orquestrados são transformados em GSPN.

Na transformação, os serviços básicos, que são executados nos provedores de serviços, são modelados em GSPN como mostrado na Figura 4.4.

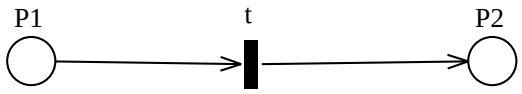

Figura 4.4: Representação dos serviços básicos e atividades básicas em GSPN.

A representação dos serviços básicos em GSPN é modelado por uma transição "t", de dois lugares "P1" e "P2", e dois arcos ligando cada um destes lugares a uma transição "t", como mostrado na Figura 4.4. A existência de fichas no lugar "P1" representa que o serviço básico modelado pela transição "t" esta apto a executar. Após a execução (disparo) da transição "t", o lugar "P2" conterá fichas, e isto representa que o serviço básico foi executado.

A transformação dos serviços orquestrados em GSPN dar-se-á a partir da transformação das atividades básicas e atividades estruturadas que compõem o seu código BPEL.

4.3.1.1 Transformação de Atividades Básicas

As atividades básicas são aquelas que descrevem os passos de uma atividade elementar. BPEL define as seguintes atividades básicas: <*Process*>, <*Invoke*>, <*Receive*>, <*Reply*>, <*Wait*> e <*Empty*>.

A representação em GSPN das atividades básicas é o mesmo que dos serviços básicos mostrados na Figura 4.4.

4.3.1.2 Transformação de Atividades Estruturadas

As atividades estruturadas prescrevem a ordem na qual um conjunto de atividades básicas é executado. Para permitir a representação de estruturas complexas, BPEL define as seguintes atividades estruturadas: <*Sequence*>, <*Switch*>, <*While*>, <*Pick*>, <*Flow*>, <*fault handlers*>, <*Compensation handlers*> e <*Compensate*>.

A seguir são apresentadas as suas transformações em GSPN.

- **Estrutura de Sequência <*Sequence*>**: essa estrutura contém uma ou mais atividades que são realizadas sequencialmente. Sua representação em GSPN é mostrada na Figura 4.5;

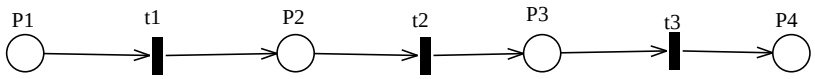

Figura 4.5: Representação da estrutura de sequência em GSPN.

- **Estrutura de Condição <*Switch*>**: esta estrutura suporta escolhas condicionais. Sua representação em GSPN é mostrada na Figura 4.6. Nesta estrutura apenas uma das transições, de "t2" até "tn", é disparada quando da chegada de uma ficha em "P1". É importante salientar que são atribuídos valores probabilísticos para cada ocorrência de disparo das transições, de "t2"

até "tn", ou seja, se houver duas transições para serem escolhidas para disparar, pode-se atribuir a ambas a mesma probabilidade de disparo, ou seja, atribui-se 50% para cada uma delas;

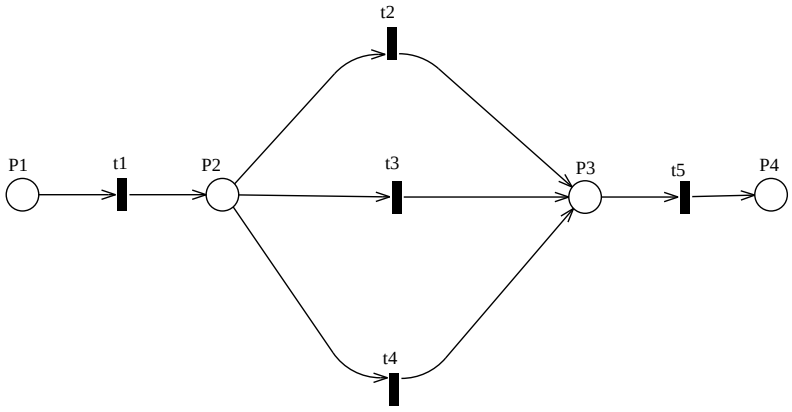

Figura 4.6: Representação da estrutura de condição em GSPN.

- **Estrutura de Repetição - Enquanto <*While*>**: essa estrutura permite que uma ou uma série de atividades sejam executadas: nenhuma, uma ou mais vezes. A Figura 4.7 mostra a representação em GSPN dessa estrutura. As transições, de "t3" até "tn", podem disparar de maneira repetitiva, até que a transição "t2" dispare e encerre o ciclo de repetições. É importante salientar que da mesma forma da estrutura de condição <*Switch*>, atribuem-se valores probabilísticos para cada ocorrência de disparo das transições "t2" e "t3". Se ambas, "t2" e "t3", possuem a mesma probabilidade de disparo, existirá a mesma probabilidade de se permanecer no laço de repetições ou sair dele;

- **Estrutura <*Pick*>**: a estrutura <*Pick*> aguarda a ocorrência de um ou de um conjunto de eventos e, em seguida, ela executa a atividade associada com o evento ou conjunto de eventos que ocorreu. A Figura 4.8 mostra a representação em GSPN dessa estrutura; e

- **Estrutura <*Flow*>**: a estrutura <*Flow*> em BPEL permite especificar que uma ou mais atividades sejam executadas concorrentemente. A estrutura <*Flow*> é encerrada quando todas as atividades que compreendem a estrutura <*Flow*> forem completadas. A Figura 4.9 mostra a representação em GSPN dessa estrutura.

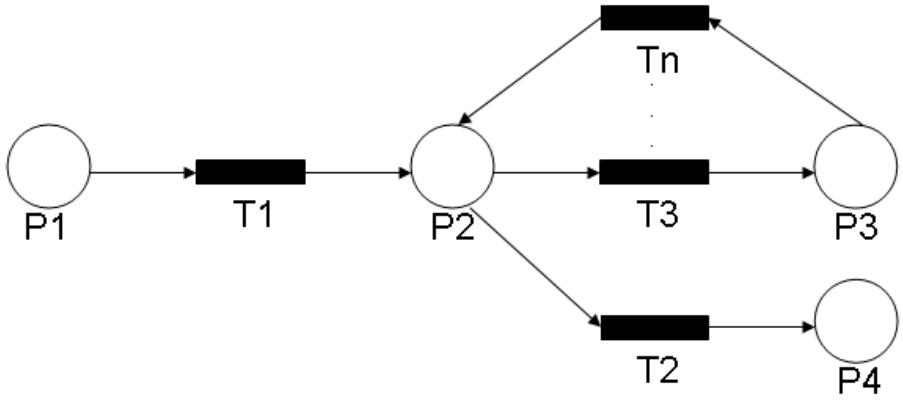

Figura 4.7: Representação da estrutura de repetição em GSPN.

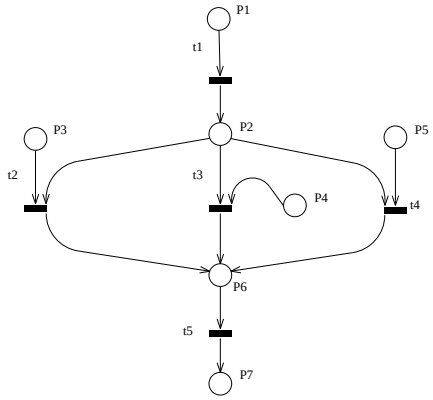

Figura 4.8: Representação da estrutura <*Pick*> em GSPN.

Note-se que os <*fault handlers*> não serão considerados para efeitos do cálculo de desempenho, pois as exceções não fazem parte do comportamento normal da execução dos *web services*. <*Compensation handlers*> e atividades <*Compensate*> serão também ignoradas, porque elas só podem ser ativadas a partir de falhas ou de outras <*Compensation handlers*>.

Na Seção 4.3.1.3 são apresentadas as regras para calcular o tempo de atraso das transições temporizadas das GSPN.

4.3.1.3 Atribuições Temporais à GSPN

Na transformação de códigos BPEL em GSPN, faz-se uso de transições imediatas, exceto as transições que representam a atividade básica <*Invoke*> que são modeladas por transições temporizadas e recebem como tempo de atraso, o valor

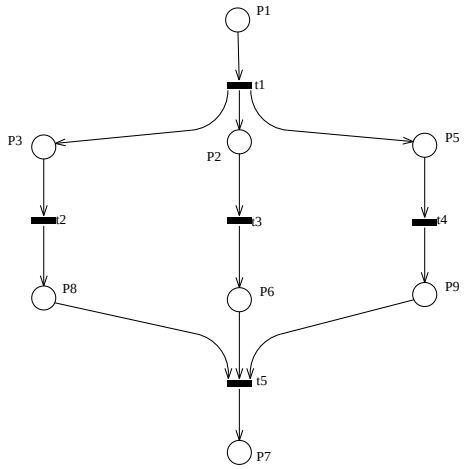

Figura 4.9: Representação da estrutura <*Flow*> em GSPN.

da resposta da Função Distribuição de Probabilidade (PDF) do tempo de resposta de cada um dos provedores de serviços, onde os serviços invocados pela atividade básica <*Invoke*> são executados.

Como já foi mencionado, para modelar o comportamento estocástico do tempo de resposta dos provedores de serviços, faz-se-á uso da Função Distribuição de Probabilidade (PDF).

Como entradas da PDF, serão utilizadas a média aritmética e o desvio padrão dos tempos de resposta dos provedores de serviços, enquanto como saída é esperado o valor do tempo de atraso - *delay time* da transição, representado pela letra grega (λ).

Os tempos de resposta dos provedores de serviços oferecem uma amostra com distribuição desconhecida, média (μ) e desvio padrão (σ).

A média (μ) é calculada através da média aritmética dos tempos de resposta dos provedores de serviços e o desvio padrão (σ) é calculado como mostrado na Figura 4.10(A).

Dependendo do valor do coeficiente de variação (CV), que é calculado como mostrado na Figura 4.10(B), os tempos de resposta dos provedores de serviços são aproximados por uma das distribuições: hiperexponential ou hipoexponential (SILVA; LINS, 2006). Isto torna possível representar a questão estocástica (aleatória) envolvida na aproximação destes tempos de resposta dos provedores de serviços ao

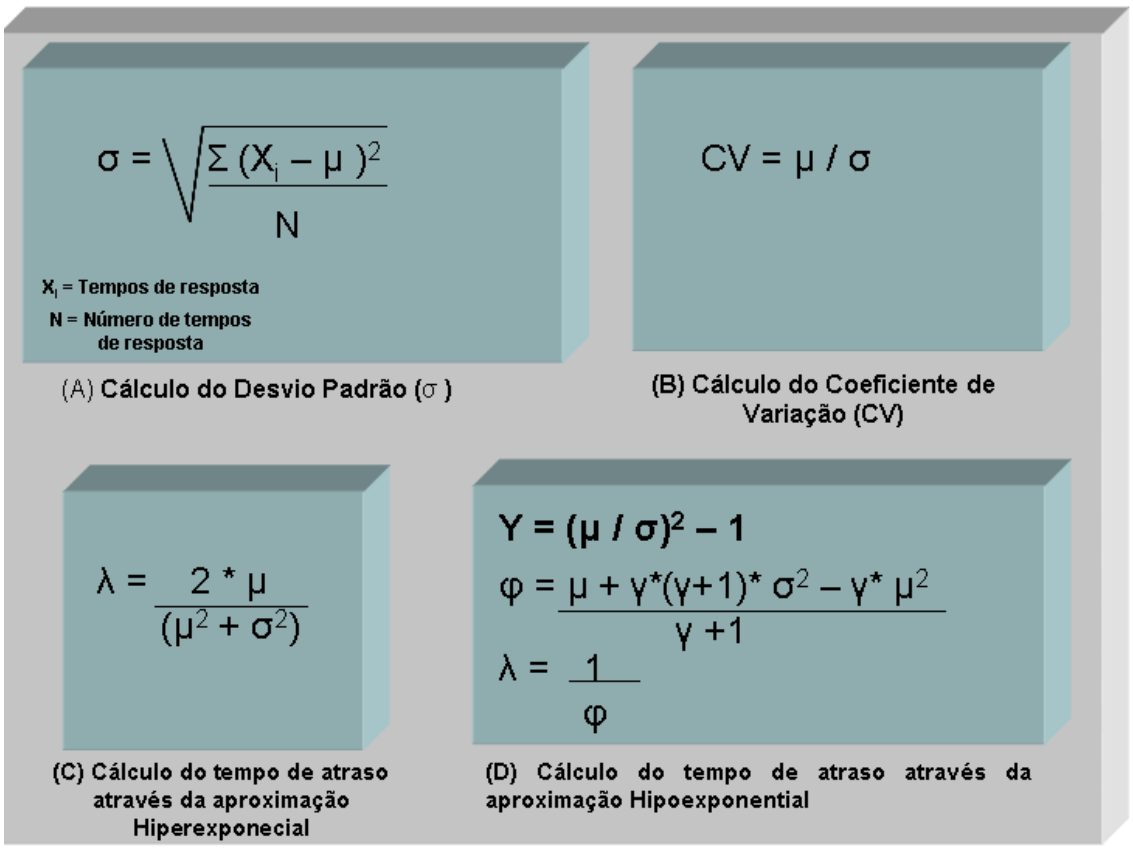

Figura 4.10: Cálculo do σ , CV e λ (SILVA; LINS, 2006).

tempo de atraso (λ) das transições que os modelam.

Se CV $>$ 1, a distribuição deve ser aproximada pela distribuição hiperexponecial. Neste caso o tempo de atraso (λ) da transição que modela a execução da atividade básica <*Invoke*> para o provedor de serviço será calculado conforme demonstrado na Figura 4.10(C).

Se CV $<$ 1, a distribuição deve ser aproximada pela distribuição hipoexponential e o tempo de atraso (λ) da transição que modela a execução da atividade básica <*Invoke*>, para o provedor de serviço, será calculado conforme demonstrado na Figura 4.10(D).

Para ilustrar a transformação de BPEL em GSPN suponha o seguinte código BPEL, como mostrado na Figura 4.11.

No código BPEL da Figura 4.11 as atividades básicas: <*Process*>, <*Receive*>, <*Switch*>, <*Sequence*>, <*While*>, <*Flow*>, <*Reply*> serão modeladas com transições imediatas, representadas no modelo da Figura 4.12 como retângulos

```
<Process>
/** PartnerLink
/** Variables
/** Flow Logic  ----
 <Receive createInstance="yes"
/>
<Seguence>
       <Invoke Serv_1 />
       <Invoke Serv_2 />
</Sequence>
<Switch>
       <Case OPT = "CHT_1">
          <Invoke Serv_3 />
       </Case>
       <Case OPT = "CHT_2">
          <Invoke Serv_4 />
       </Case>
</Switch>
```

```
<while Con = "NOk">
     <Invoke Serv_5 />
     <Invoke Serv_6 />
</while>
<Flow>
     <Invoke Serv_6 />
     <Invoke Serv_7 />
     <Invoke Serv_8 />
</Flow>
<Reply variabele="status" />
/** ------------
/** Fault Handler
/** Compensation Handler
</Process>
```

Figura 4.11: Código BPEL.

escuros. Já as transições que modelam as atividades <*Invoke*> (no modelo da Figura 4.12 representadas por retângulos claros) serão modeladas com transições temporizadas e receberão como tempos de atrasos os valores das saídas das PDF, do tempo de resposta de cada um dos provedores de serviços, onde os serviços invocados serão executados.

Na Seção 4.5.1.2 será feita a análise de desempenho para o código BPEL da Figura 4.11, através do modelo apresentado na Figura 4.12, utilizando a ferramenta GREATSPN[1].

4.3.2 Regras de Transformação de BPEL em CPN

O objetivo desta seção é fornecer as regras de transformação do código BPEL em rede de Petri colorida - *Colored Petri Net* (CPN). Será apresentada a representação das atividades básicas e das atividades estruturadas presentes nos códigos BPEL, que estão presentes no BPI e nos códigos BPEL dos serviços orquestrados, em CPN.

Dois principais requisitos para o uso de modelos formais, como as GSPN e as

[1] `http://www.di.unito.it/~greatspn`

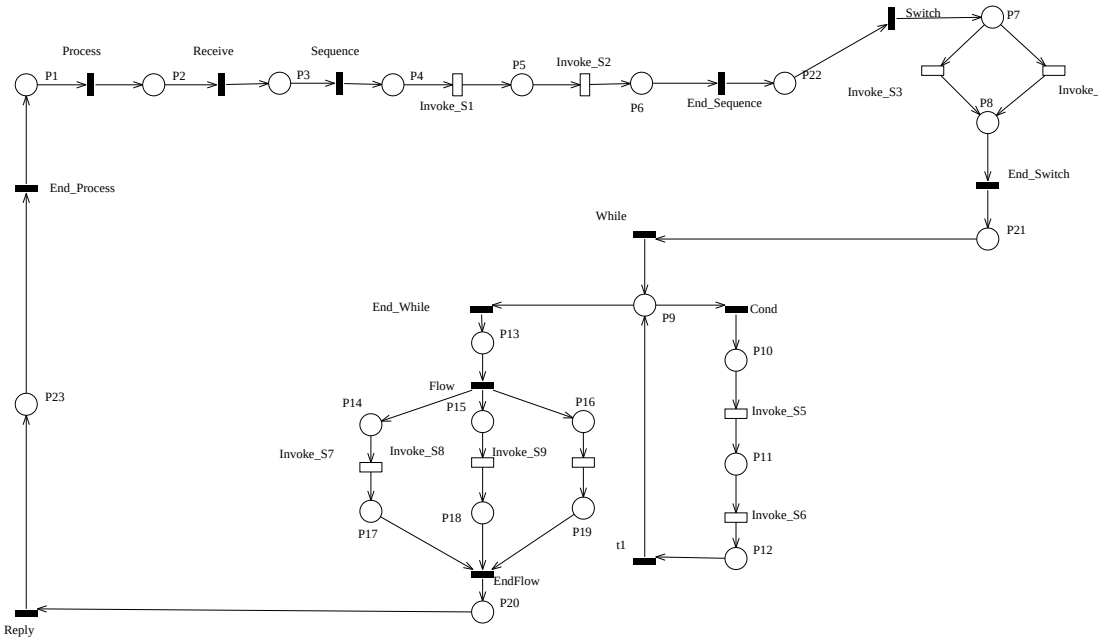

Figura 4.12: Rede GSPN do Código BPEL da Figura 4.11.

CPN, para analisar códigos BPEL são: i) a possibilidade de se poder modelar a concorrência, comunicação e sincronização dos serviços; ii) a possibilidade de se fazer uma análise de desempenho dos WS orquestrados com o BPEL.

A Figura 4.13 apresenta a sequência do processo de transformação e análise de desempenho realizadas através de simulações nos modelos CPN obtidos.

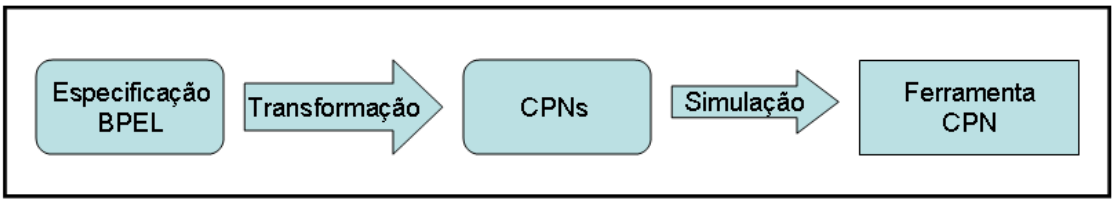

Figura 4.13: Sequência de transformação e análise de BPEL em CPN.

A mesma sequência foi aplicada ao processo de transformação e análise de desempenho das GSPN, com a utilização da ferramenta GREATSPN na realização de simulações.

4.3.2.1 Transformação de Atividades Básicas

A transformação das atividades básicas em CPN é relativamente simples.

- A atividade <*Receive*> especifica o *link* de parceiro que espera receber e o tipo de porta e operação que espera que o parceiro invoque. Sua representação em CPN é mostrada na Figura 4.14.

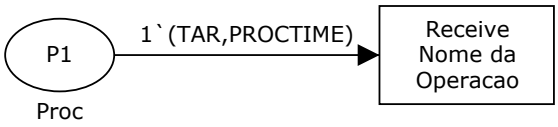

Figura 4.14: Atividade <*Receive*> modelada em CPN.

- A atividade <*Reply*> é usada para enviar uma resposta a um pedido previamente aceito através de um atividade <*Receive*>. Tais respostas só são significativas para as interações síncronas. A atividade <*Reply*> pode especificar uma variável que contém os dados da mensagem a ser enviada em uma resposta. Sua representação em CPN é mostrada na Figura 4.15.

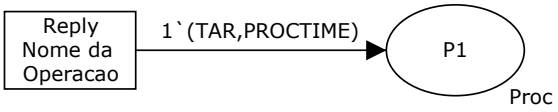

Figura 4.15: Atividade <*Reply*> modelada em CPN.

- A atividade <*Invoke*> pode ser uma operação "pedido/resposta" síncrona ou uma operação assíncrona *one-way*. BPEL usa a mesma sintaxe básica para ambas com opções adicionais para o caso síncrono. Sua representação em CPN é mostrada na Figura 4.16.

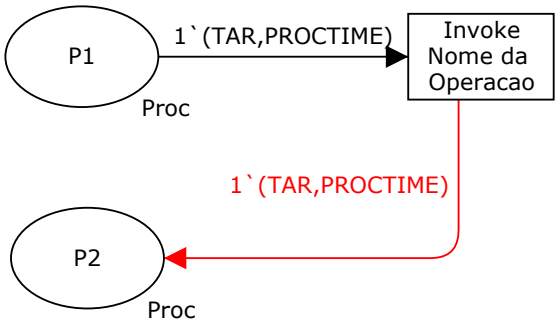

Figura 4.16: Atividade <*Invoke*> modelada em CPN.

- A atividade <*Empty*> permite que se insira uma instrução em branco "*no-op*" em um código BPEL. Sua representação em CPN é mostrada na Figura 4.17.

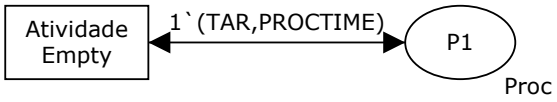

Figura 4.17: Atividade <*Empty*> modelada em CPN.

4.3.2.2 Transformação de Atividades Estruturadas

Para permitir a apresentação de estruturas complexas em BPEL, as seguintes atividades estruturadas foram definidas: <*Sequence*>, <*Switch*>, <*While*>, <*Pick*>, <*Flow*>, <*fault handlers*>, <*Compensation handlers*> e <*Compensate*>. A seguir são apresentadas a sua transformação em CPN.

Na transformação de códigos BPEL4People em CPN os seguintes tipos coloridos foram definidos, conforme mostrado na Figura 4.18.

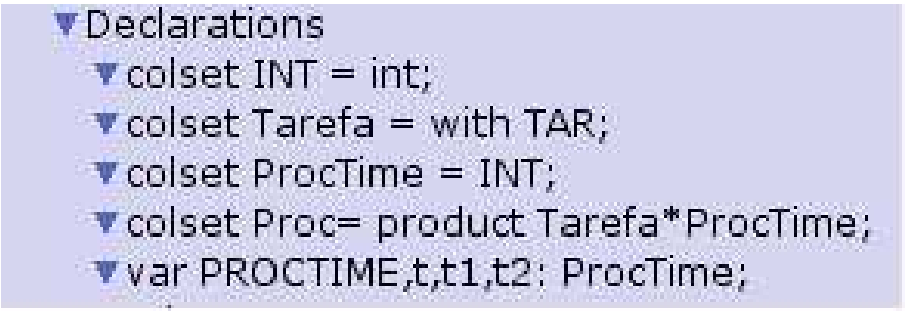

Figura 4.18: Tipos coloridos das redes CPN dos códigos BPEL4People.

- A estrutura <*Sequence*> do BPEL contém uma ou mais atividades que são executadas sequencialmente, na ordem em que são listadas na estrutura <*Sequence*>. Sua representação em CPN é mostrada na Figura 4.19.

- A estrutura <*Switch*> do BPEL aceita o comportamento condicional. A estrutura consiste em uma lista ordenada de um ou mais elementos condicionais. Os elementos condicionais <*Case*> do <*Switch*> são considerados na ordem em que aparecem. O primeiro elemento condicional <*Case*>, cuja condição é verdadeira é selecionado e contém as atividade a serem realizadas pelo <*Switch*>. Se nenhum elemento condicional <*Case*> é selecionado, então o elemento <*Otherwise*> é selecionado. Sua representação em CPN é mostrada na Figura 4.20.

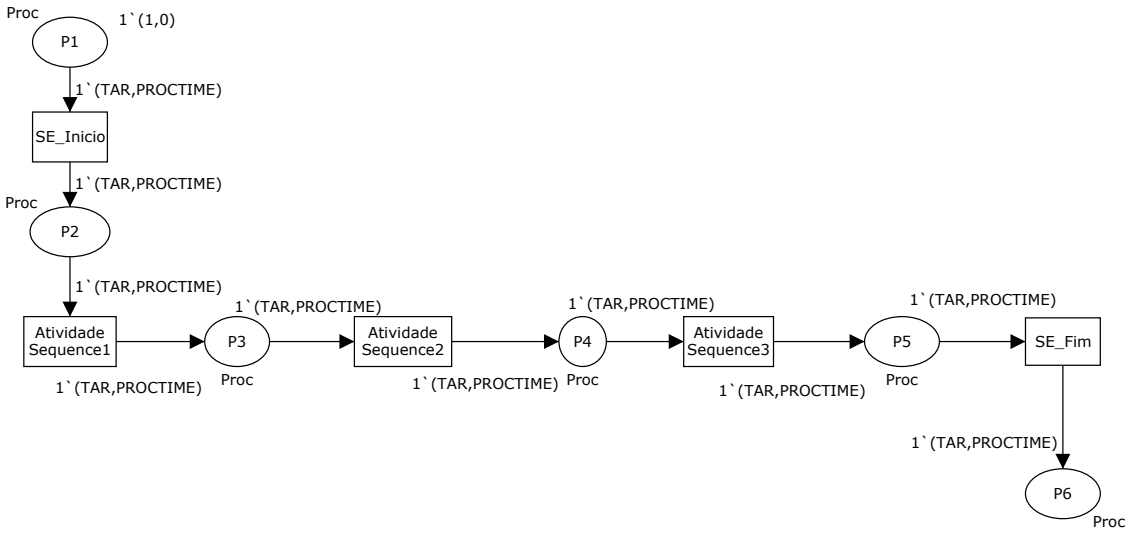

Figura 4.19: Estrutura <*Sequence*> modelada em CPN.

- A estrutura <*While*> do BPEL suporta a repetição de uma determinada atividade ou conjunto de atividades até que a condição booleana dada não seja mais verdadeira. Sua representação em CPN é mostrada na Figura 4.21.

- A estrutura <*Pick*> do BPEL aguarda a ocorrência de um ou de um conjunto de eventos e, em seguida, executa a atividade associada com o evento que ocorreu. A ocorrência dos eventos é, muitas vezes, mutuamente excludente. Sua representação em CPN é mostrada na Figura 4.22.

- A estrutura <*Flow*> do BPEL permite especificar uma ou mais atividades a serem realizadas concorrentemente. A estrutura <*Flow*> é concluída quando todas as atividades no fluxo foram concluídas. Sua representação em CPN é mostrada na Figura 4.23.

Note-se que, como já foi mencionado anteriormente, os <*fault handlers*> não serão considerados para efeitos do cálculo de desempenho, pois as exceções não fazem parte do comportamento normal da execução dos *web services*. <*Compensation handlers*> e atividades <*Compensate*> serão também ignoradas, porque elas só podem ser ativadas a partir de falhas ou de outras <*Compensation handlers*>.

4.3.2.3 Atribuições Temporais à CPN

Na transformação de códigos BPEL em CPN, faz-se uso de transições que passarão como valor do tipo colorido "*proctime*" o valor 0 (zero), exceto as transições

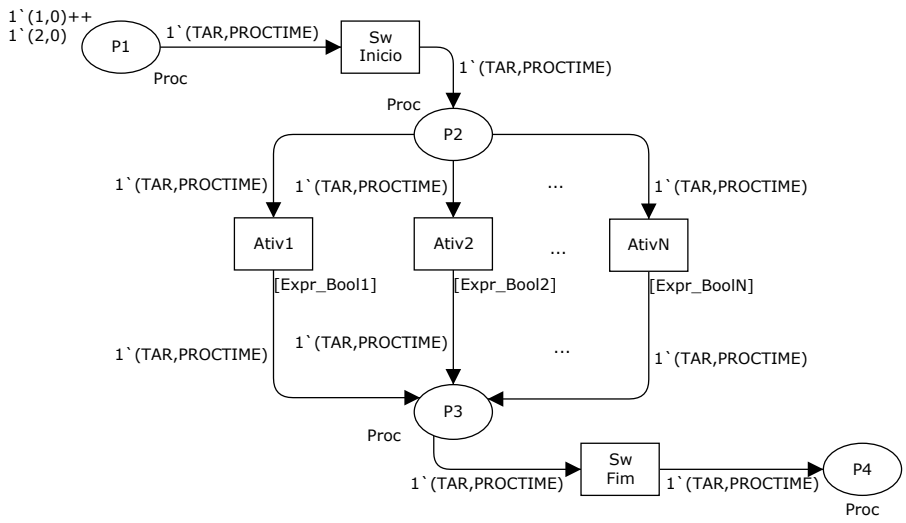

Figura 4.20: Estrutura <*Switch*> modelada em CPN.

que modelam as atividades <*Invoke*> que serão modeladas por transições que passarão como valores do tipo colorido "*proctime*" os valores das saídas das PDF dos tempos de resposta de cada um dos provedores de serviços, onde os serviços invocados serão executados.

Para modelar o comportamento estocástico dos tempos de respostas dos provedores de serviços, faz-se uso da Função Distribuição de Probabilidade (PDF), como já apresentado na Seção 4.3.1.

Como entradas da PDF, serão utilizados a média aritmética e o desvio padrão dos tempos de resposta dos provedores de serviços, enquanto como saída da PDF é esperado um valor numérico para ser atribuído ao tipo colorido "*proctime*", que tem a função de guardar o atraso provocado pela invocação do serviço em um provedor de serviço.

Para ilustrar a transformação de BPEL em CPN suponha o código BPEL, como mostrado na Figura 4.11.

No código BPEL da Figura 4.11 as atividades básicas: <*Process*>, <*Receive*>, <*Switch*>, <*Sequence*>, <*While*>, <*Flow*>, <*Reply*> serão modeladas por transições que passarão como valor do tipo colorido "*proctime*" o valor 0 (zero), como mostrado na Figura 4.24. Já as transições que modelam as atividades <*Invoke*> serão modeladas por transições que passarão como valores do tipo colorido "*proctime*" os valores das saídas das PDF do tempo de resposta de cada um dos provedores de serviços, onde o serviço invocado será executado, como também

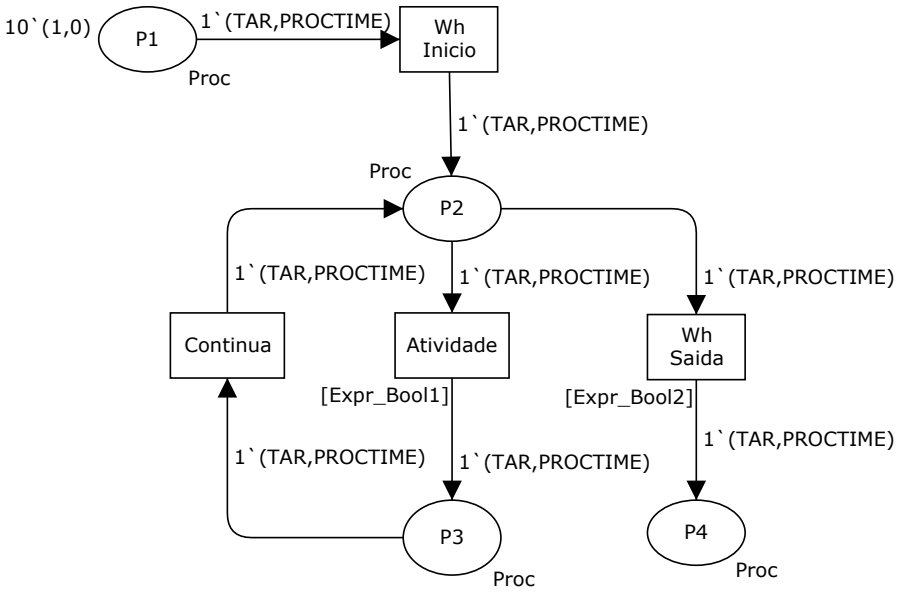

Figura 4.21: Estrutura <*While*> modelada em CPN.

mostrado na Figura 4.24.

Na Seção 4.5.1.3 será feita a análise de desempenho para o código BPEL da Figura 4.11 através do modelo apresentado na Figura 4.24, utilizando a ferramenta CPN-*tools* (HUBER; JENSEN; SHAPIRO, 1991).

4.4 A Camada Rede de Petri

Como resultado das atividades realizadas na camada de transformação da arquitetura "SOASPE", apresentada na Seção 4.3, foram gerados os arquivos XML, códigos fontes das redes de Petri GSPN ou CPN, necessários para carregar estas redes em ferramentas GREATSPN ou CPN-*tools*, respectivamente.

Na atual versão da ferramenta de transformação, implementada a partir das regras de transformação especificadas nesta tese e denomidana de "TPeople4PN", somente foi implementada a transformação automática de códigos BPEL e de sua extensão BPEL4People em GSPN. As regras de transformação destes códigos em CPN, somente estão descritas teoricamente nas Seções 4.3.2 e 5.3. A implementação da transformação automatica de códigos BPEL e de sua extensão BPEL4People em CPN fica como sugestão de trabalho futuro.

A Camada de rede de Petri é composta pelos códigos fontes das GSPN ou CPN obtidos por meio das regras da camada de transformação. Estes códigos fontes

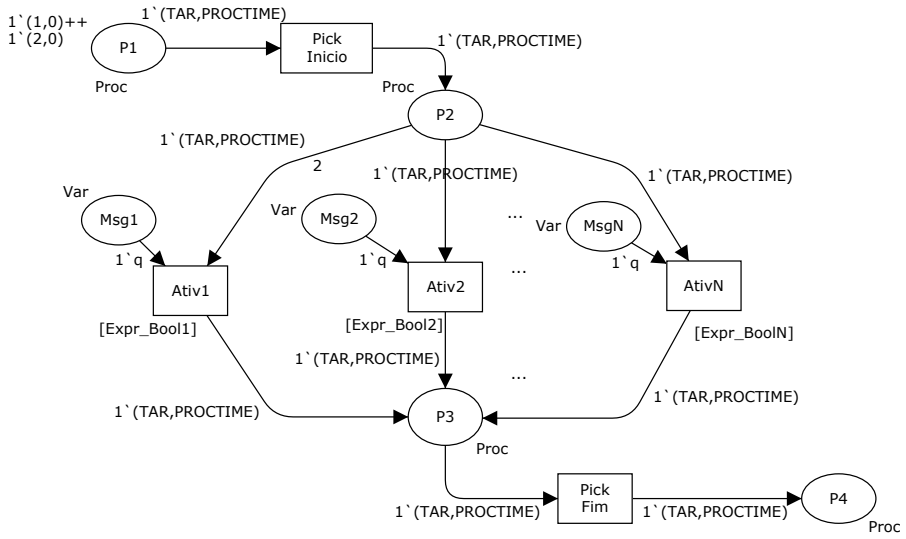

Figura 4.22: Estrutura <*Pick*> modelada em CPN.

gerados devem ser carregados na ferramenta GREATSPN ou CPN-*tools* (HUBER; JENSEN; SHAPIRO, 1991) respectivamente, assim sendo, os outros componentes desta camada são as ferramenta GREATSPN ou CPN-*tools*.

4.5 A Camada de Desempenho

A camada de desempenho é definida como sendo a camada responsável pelos testes de análise de desempenho que os modelos serão submetidos para averiguar a eficiência dos WS orquestrados com o BPEL4People. Esta camada é também responsável pela visualização das simulações submetidas a estas redes GSPN ou CPN e dos gráficos obtidos destas referidas simulações.

A camada de desempenho se fundamenta na SPE, uma vez que ela é uma metodologia cuja abordagem é a utilização de métodos de transformação de *softwares* em modelos formais, com o objetivo de se utilizar destes modelos para que se possa identificar problemas com a arquitetura do sistema, *design*, implementação ou desempenho destes.

4.5.1 Análise de Desempenho de Códigos BPEL Modelados em GSPN e em CPN

Na verificação da usabilidade e a validade da arquitetura "SOASPE" na análise de desempenho, foi adotada a metodologia de se medir os tempos de resposta reais dos

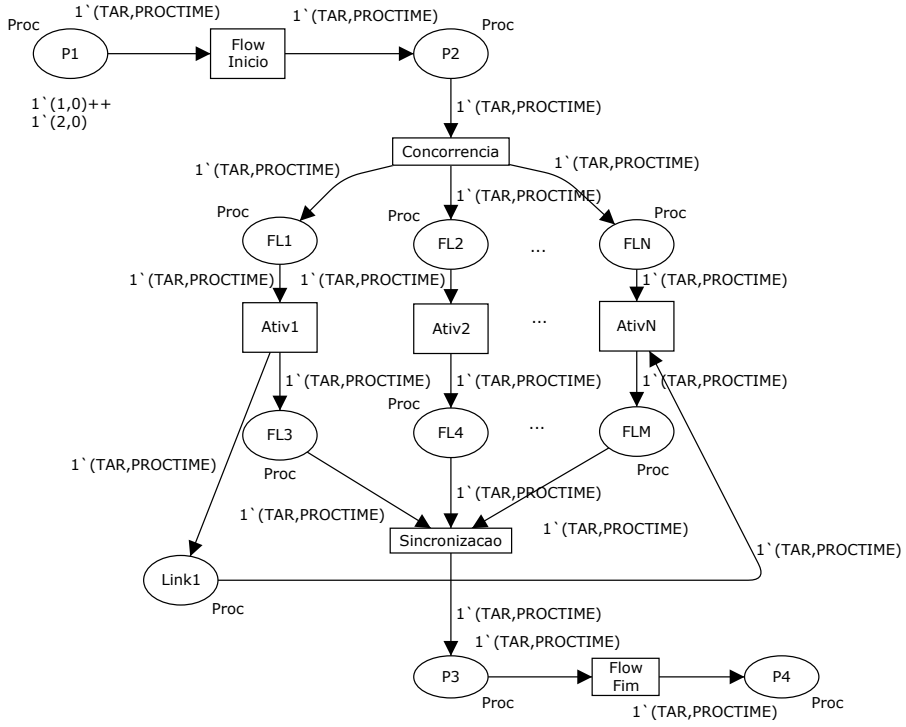

Figura 4.23: Estrutura <*Flow*> modelada em CPN.

WS e em seguida compará-los com os tempos de resposta obtidos das simulações dos modelos gerados (GSPN e CPN) para os mesmos através da arquitetura "SOASPE".

Com a arquitetura "SOASPE" definida, esta seção apresenta a análise de desempenho de um estudo de caso prático através do *web services* - "WS SodaSys" com o objetivo de verificar a usabilidade e a validade da mesma para modelos GSPN e CPN.

O *web service* "WS SodaSys" é um WS orquestrado com o código BPEL mostrado na Figura 4.25 e foi implementado no centro politécnico superior da Universidade de Saragoza (Espanha).

A análise do estudo de caso do *web service* - "WS SodaSys" foi executada em máquinas Intel Duo Core de 1.86 GHz, com placa mãe *on-board* e com 2 GB memória RAM. O sistema operacional instalado foi o *Windows XP Professional*.

Na implementação do *web service* "WS SodaSys", vários artefatos de *software* foram utilizados. Os *web services* (escritos em Java) foram disponibilizados em servidores *Tomcat*, versão 5.0.28. Adicionalmente, foram utilizados o Ant (*Apache AntUnit* - versão 1.6), o módulo SOAP para o *Apache (Java Web Services* - versão 2.3.1) e o *Apache Axis* (Versão 1.4). Todos estes *softwares* são necessários para

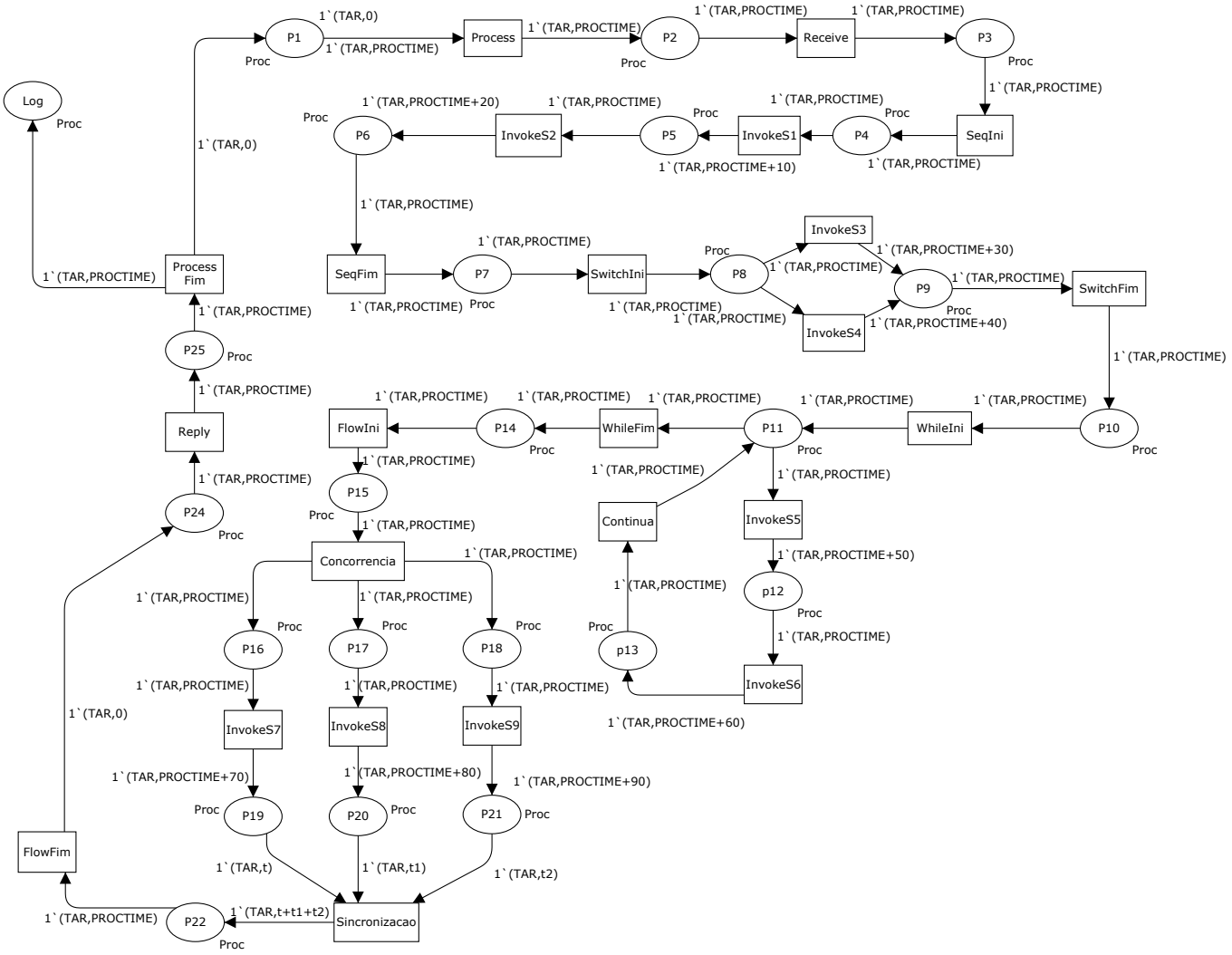

Figura 4.24: Rede CPN do Código BPEL da Figura 4.11.

a execução de *web services* escritos em Java. Informações adicionais sobre a necessidade e a forma de utilização dos mesmos podem ser encontradas em "*Java Web Services*" (YU; LIN, 2006).

Para possibilitar a orquetração do *web service* "WS SodaSys", um motor (*engine*) de BPEL foi utilizado. Pela razão de ser *Open-Source*, o motor ActiveBPEL (versão 1.2) foi adotado.

4.5.1.1 Desempenho do "WS SodaSys"

O "WS SodaSys" é orquestrado por um Processo de Negócio Integrador (PNI), código BPEL apresentado na Figura 4.11, que invoca os serviços de nove provedores de serviços: Serv-1, Serv-2, ... , Serv-9.

```
<Process>
/** PartnerLink
/** Variables
/** Flow Logic  ----
 <Receive createInstance="yes"
/>
<Seguence>
       <Invoke Serv_1 />
       <Invoke Serv_2 />
</Sequence>
<Switch>
       <Case OPT = "CHT_1">
          <Invoke Serv_3 />
       </Case>
       <Case OPT = "CHT_2">
          <Invoke Serv_4 />
       </Case>
</Switch>
```

```
<while Con = "NOk">
     <Invoke Serv_5 />
     <Invoke Serv_6 />
</while>
<Flow>
     <Invoke Serv_6 />
     <Invoke Serv_7 />
     <Invoke Serv_8 />
</Flow>
<Reply variabele="status" />
/** ------------
/** Fault Handler
/** Compensation Handler
</Process>
```

Figura 4.25: Código BPEL de orquetração do *web service* - "WS SodaSys".

Foram medidos os tempos de resposta do Processo de Negócio Integrador (PNI) e dos provedores de serviços de forma individual. O tempo de resposta do PNI compreende desde o tempo no qual uma invocação é feita até a recepção de sua resposta, incluindo o tempo de execução dos próprios provedores de serviços.

Os tempos dos provedores de serviços foram colhidos de forma individual. A Tabela 4.1 apresenta os tempos de resposta dos provedores de serviços do "WS SodaSys".

Na Tabela 4.2 são apresentados os tempos de resposta do código BPEL do Processo de Negócio Integrador do "WS SodaSys", quando este encontra-se com um número de requisições variando de 130 a 290 requisições.

4.5.1.2 Desempenho do Modelo GSPN Gerado pela Arquitetura "SOASPE"

Nesta seção será abordada a utilização da arquitetura "SOASPE" para simular a avaliação de desempenho do "WS SodaSys" a partir do modelo GSPN gerado a partir do código BPEL do PNI do "WS SodaSys" pela arquitetura "SOASPE".

Tabela 4.1: Tempos de resposta dos provedores de serviços do "WS SodaSys".

	Tempo de resposta (em ms)
Serv-1	5,191
Serv-2	5,293
Serv-3	4,891
Serv-4	5,351
Serv-5	5,061
Serv-6	4,754
Serv-7	5,097
Serv-8	5,196
Serv-9	5,392

Tabela 4.2: Tempos de resposta do código BPEL do PNI do "WS SodaSys".

Qtd. Requisições	**Tempo de Resposta (em ms)**
130	5200,079
150	5212,406
165	5206,748
170	5221,571
190	5219,313
200	5217,848
210	5231,001
230	5218,148
240	5226,833
255	5208,288
270	5212,121
290	5225,143

Como já mencionado, a camada de transformação recebe como entrada: o código BPEL do Processo de Negócio Integrador (PNI), os códigos BPEL (Serviços Estruturados) e as Funções Distribuição de Probabilidade (FDP) dos tempos de respostas dos provedores de serviços que executarão os serviços básicos. A partir de então, as regras de transformação da camada de transformação converterão estas entradas em um modelo GSPN.

No estudo de caso em questão, a camada de transformação receberá o código BPEL do PNI do "WS SodaSys" e os tempos de resposta dos nove provedores de serviços. Com estes dados de entrada, as regras de transformação gerarão a GSPN para o "WS SodaSys" de acordo com as diretrizes definidas na Seção 4.3.1.

O cálculo do tempo de atraso das transições que modelam as atividades $<Invoke>$ nas chamadas de serviços nos provedores de serviços do "WS SodaSys" é mostrado a seguir.

Cada provedor de serviço (SP) invocado envia uma mensagem SOAP contendo uma lista de tempos de resposta deste servidor, como ilustrado na Figura 4.3, para o integrador. A partir desta lista, as regras de transformação calculam o valor do coeficiente de variação (CV) para que se possa fazer uma aproximação destes valores por uma Função de Distribuição de Probabilidade (FDP) e, assim, seja obtido um valor aproximado do tempo de atraso da transição que modelará a chamada deste serviço no provedor de serviço respectivo no modelo GSPN.

No estudo de caso em questão, o "WS SodaSys" invoca nove provedores de serviço: Serv-1, Serv-2, ... , Serv-9. Seus tempos de atrasos para as transições que os modelarão em GSPN serão calculados a seguir.

A Tabela 4.3 mostra a Média (μ) e o Desvio Padrão (σ) para os provedores de serviços: Serv-1, Serv-2, ... , Serv-9.

Com os valores μ e σ são calculados os coeficientes de variação (CV) de cada um dos provedores de serviços, conforme a fórmula especificada na Figura 4.10.

A Tabela 4.4. mostra os valores dos coeficientes de variação para cada um dos provedores de serviços e o tipo de aproximação utilizada pela FDP.

Como os coeficientes de variação calculados na Tabela 4.4 foram menores que 1 (CV<1), as distribuições deverão ser aproximadas através da distribuição hipoexponencial e os tempos de atrasos (λ) das transições que modelam estes

Tabela 4.3: Média e desvio padrão dos provedores de serviços: Serv-1, Serv-2, ... , Serv-9.

	μ	σ
Serv-1	5,351	3,37
Serv-2	4,819	2,505
Serv-3	5,162	3,17
Serv-4	4,912	2,575
Serv-5	5,372	3,229
Serv-6	4,823	2,495
Serv-7	5,051	3,09
Serv-8	4,768	2,523
Serv-9	5,359	3,35

Tabela 4.4: CV e tipo da aproximação.

	Coeficiente de Variação (CV)	**Aproximação**
Serv-1	0,63	Hipoexponencial
Serv-2	0,52	Hipoexponencial
Serv-3	0,59	Hipoexponencial
Serv-4	0,55	Hipoexponencial
Serv-5	0,61	Hipoexponencial
Serv-6	0,59	Hipoexponencial
Serv-7	0,65	Hipoexponencial
Serv-8	0,57	Hipoexponencial
Serv-9	0,61	Hipoexponencial

Tabela 4.5: Tempo de atraso dos SP´s: Serv-1, Serv-2, ... , Serv-9.

	λ **em ms**
Serv-1	4,33785699
Serv-2	2,998373314
Serv-3	4,577812434
Serv-4	3,948373314
Serv-5	4,581249943
Serv-6	3,788373314
Serv-7	4,007812499
Serv-8	3,238373314
Serv-9	4,161249987

serviços nos provedores de serviços deverão ser calculados como mostrado na Figura 4.10.

A Tabela 4.5 mostra os valores calculados para os tempos de atrasos das transições que modelam os provedores de serviços: Serv-1, Serv-2, ... , Serv-9.

Finalizadas as atividades realizadas pela camada de transformação, obtém-se os arquivos necessários para a carga da espeficicação em GSPN para a ferramenta GREATSPN.

Com a rede GSPN carregada (aberta) na ferramenta GREATESPN, iniciam-se as atividades da camada de desempenho.

A análise de desempenho do modelo GSPN do "WS SodaSys" é realizada a partir de simulações com a mesma quantidade de requisições realizadas com o "WS SodaSys".

A Tabela 4.6 apresenta os tempos de resposta do modelo GSPN do "WS SodaSys" quando este encontra-se com um número de requisições variando de 130 a 290 requisições

A Figura 4.26 mostra o gráfico dos tempos de resposta reais do "WS SodaSys" e dos tempos de resposta do modelo GSPN do "WS SodaSys" gerado a partir da arquitetura "SOASPE".

A partir dos resultados obtidos no estudo de caso, observa-se que os tempos de resposta do modelo GSPN gerado pela arquitetura "SOASPE" e os tempos

Tabela 4.6: Tempos de resposta da simulação no modelo GSPN do "WS SodaSys".

Qtd. Requisições	Tempo de Resposta (em ms)
130	5291,079
150	5362,406
165	5421,731
170	5391,571
190	5715,313
200	5699,848
210	5491,001
230	5578,148
240	5596,833
255	5598,288
270	5602,121
290	5635,143

de resposta do "WS SodaSys" não diferem em mais de 4,1%, validando assim a arquitetura "SOASPE" na análise de desempenho de WS orquestrados com o BPEL e modelados com GSPN.

A Figura 4.27 apresenta o percentual comparativo dos tempos de resposta do modelo GSPN gerado pela arquitetura "SOASPE" para o "WS SodaSys"e dos tempos de resposta do reais do "WS SodaSys".

4.5.1.3 Desempenho do Modelo CPN Gerado pela Arquitetura "SOASPE"

Nesta seção será abordada a utilização da arquitetura "SOASPE" para simular a análise de desempenho do "WS SodaSys" a partir do modelo CPN gerado pela arquitetura "SOASPE".

As regras de transformação de códigos BPEL em modelos CPN estão descritas teoricamente na Seção 4.3.2. O processo de geração automática de BPEL em CPN será contemplado em um versão futura da ferramenta "TPeople4PN". A atual versão da ferramenta "TPeople4PN" é uma implementação de uma ferramenta automática de conversão de BPEl4People em GSPN, seguindo as regras de transformação da arquitetura "SOASPE".

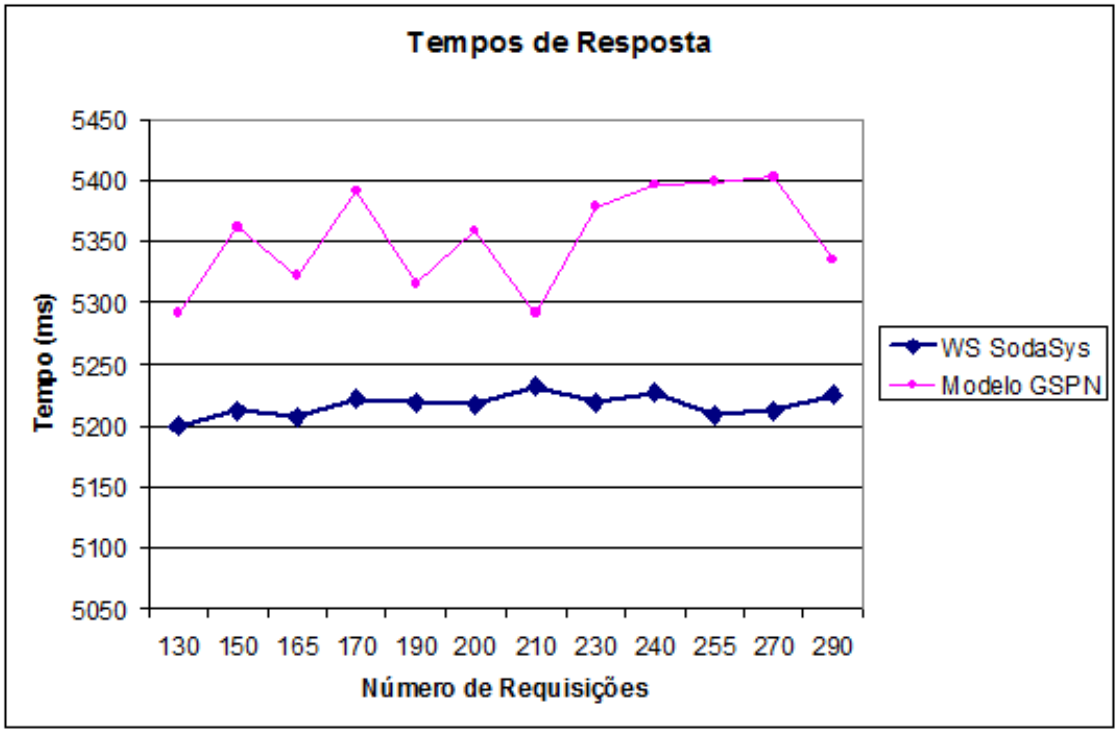

Figura 4.26: Tempos de resposta do "WS SodaSys" e do modelo GSPN gerado a partir da arquitetura "SOASPE" para o "WS SodaSys".

O desempenho do "WS SodaSys" já foi avaliado na Seção 4.5.1.1. Serão usados os mesmos valores dos tempos de resposta do "WS SodaSys" para confrontá-los com os tempos de resposta do modelo CPN do "WS SodaSys" obtidos a partir da arquitetura "SOASPE".

Da mesma forma como já foi mencionado na Seção 4.5.1.2, finalizadas as atividades realizadas pela camada de transformação, obtêm-se os arquivos necessários para a carga da espeficicação da CPN para a ferramenta CPN-*tools*.

Com a rede CPN carregada (aberta) na ferramenta CPN-*tools*, iniciam-se as atividades da camada de desempenho.

A análise de desempenho do modelo CPN do "WS SodaSys" é realizada a partir de simulações com a mesma quantidade de requisições realizadas com o "WS SodaSys".

A Tabela 4.7 apresenta os tempos de resposta do modelo CPN do "WS SodaSys" quando este encontra-se com um número de requisições variando de 130 a 290 requisições

A Figura 4.28 mostra o gráfico dos tempos de resposta reais do "WS SodaSys" e dos tempos de resposta do modelo CPN do "WS SodaSys" gerado a partir da

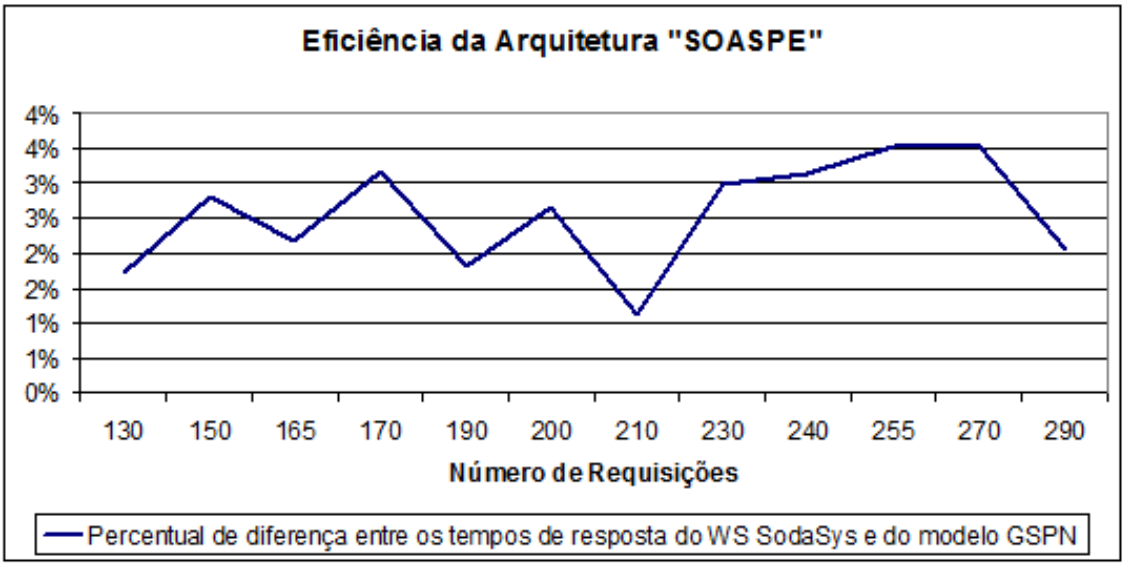

Figura 4.27: Eficiência da arquitetura "SOASPE" na análise de desempenho de WS orquestrados pelo BPEL4 em GSPN.

arquitetura "SOASPE".

A partir dos resultados obtidos no estudo de caso, observa-se que os tempos de resposta do modelo CPN gerado pela arquitetura "SOASPE" e os tempos de resposta do "WS SodaSys" não diferem em mais de 5,5%, validando assim a arquitetura "SOASPE" na análise de desempenho de WS orquestrados com o BPEL e modelados com CPN.

A Figura 4.29 apresenta o percentual comparativo dos tempos de resposta do modelo CPN gerado pela arquitetura "SOASPE" para o "WS SodaSys"e dos tempos de resposta do reais do "WS SodaSys".

4.6 Conclusões do Capítulo

Neste capítulo foi apresentada a arquitetura "SOASPE" e um estudo de caso do *web service* "WS SodaSys". Neste estudo de caso, o código BPEL do Processo de Negócio Integrador (PNI) do *web service* "WS SodaSys" é transformado em rede de Petri (GSPN e CPN) e assim permitindo se fazer um análise de desempenho do mesmo.

A análise de desempenho dos modelos em rede de Petri (GSPN e CPN) do *web service* "WS SodaSys" mostrou-se satisfatória, pois os tempos de resposta dos modelos (GSPN e CPN) gerados pela arquitetura "SOASPE" e os tempos de resposta

Tabela 4.7: Tempos de resposta da simulação no modelo CPN do "WS SodaSys".

Qtd. Requisições	Tempo de Resposta (em ms)
130	5391,319
150	5382,246
165	5491,631
170	5406,531
190	5436,343
200	5378,848
210	5531,213
230	5438,134
240	5456,753
255	5504,728
270	5412,119
290	5535,323

reais do "WS SodaSys" não diferem em mais de 5,5%, validando assim a usabilidade da arquitetura "SOASPE" na análise de desempenho de WS orquestrados com o BPEL.

A modelagem em GSPN mostrou-se um pouco mais precisa do que a modelagem CPN, pois as redes de Petri (GSPN) modelam com maior precisão o comportamento estocástico do tempo de resposta de cada um dos serviços nos provedores de serviços, do que as redes CPN que tem a limitação de definir os tempos de resposta (atrasos das transições) com valores inteiros.

No Capítulo 5 será apresentada a transformação da extensão humana "WS-HumanTask" do BPEL4People em GSPN e CPN.

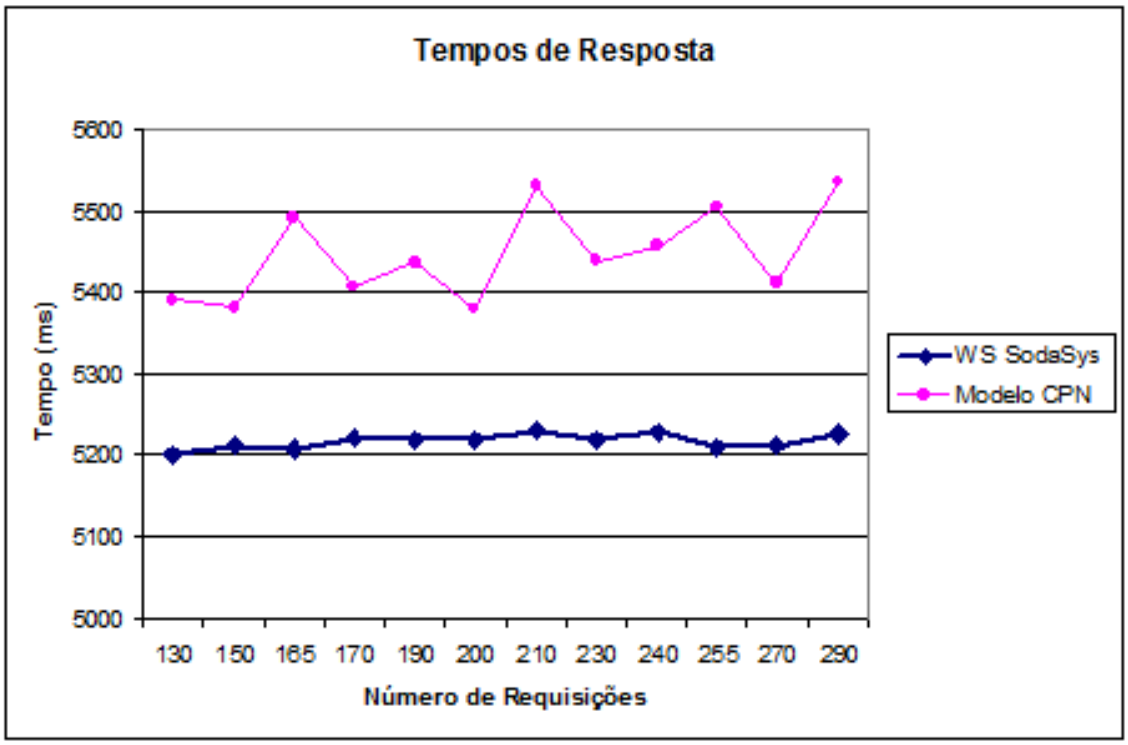

Figura 4.28: Tempos de resposta do "WS SodaSys" e do modelo CPN gerado a partir da arquitetura "SOASPE" para o "WS SodaSys".

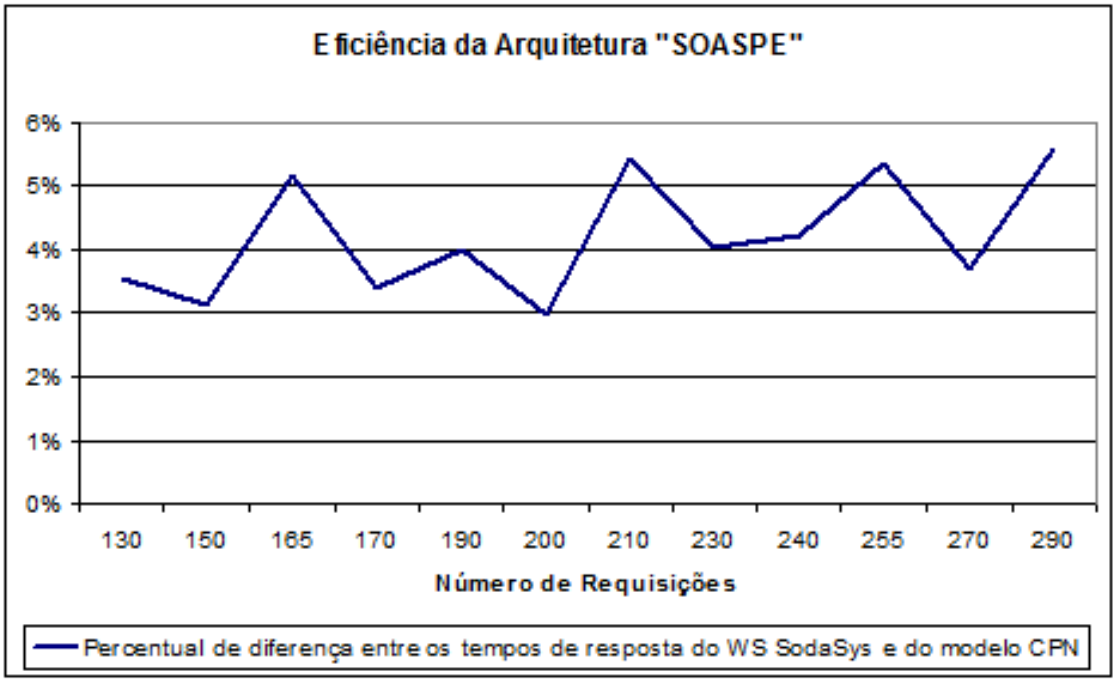

Figura 4.29: Eficiência da arquitetura "SOASPE" na análise de desempenho de WS orquestrados pelo BPEL4 em CPN.

Capítulo 5

SOASPE: Modelagem das Interações Humanas em GSPN e em CPN

Como complemento do BPEL, BPEL4People introduz novos conceitos para suportar a tarefa humana. No BPEL4People *PeopleLinks* são usados para ligar grupos de pessoas a processos de negócio semelhante ao modo que os *PartnerLinks* são usados para ligar serviços web dos WS. Quando o motor (*engine*) de processos de negócios encontra uma atividade de pessoas, ele (motor) suspende o processo de negócio até que uma pessoa do grupo conclua a tarefa associada. Os detalhes das interações humanas, ou seja, quem pode executar que atividade, são definidos na especificação de *WS-HumanTask*.

Para definir um processo de negócio com participação humana completo, ambas as especificações devem ser usadas (BPEL e WS-HumanTask), onde BPEL4People pode ser visto como uma ponte entre BPEL e *WS-HumanTask*. Termos como *HumanTask* e *People Activity* são sinônimos utilizados por estas duas especificações. Nesta tese usa-se o termo tarefa humana - *HumanTask* de modo uniforme. *WS-HumanTask* introduz também a idéia de grupos lógicos de pessoas. Cada grupo inclui um conjunto de pessoas, e cada tarefa humana só pode ser executada por alguns grupos específicos.

Conforme mostrado na Figura 5.1, uma tarefa humana envolve uma atividade que é semelhante a uma invocação de serviço normal em BPEL. A *WS-HumanTask* define a permissão da tarefa com o elemento "*PotentialOwner*", que remete para um "*PeopleLink*" previamente definido. Para finalizar o processo de negócio

de uma tarefa *WS-HumanTask* a pessoa que executa a tarefa (em terminologia BPEL4People, o atual dono da tarefa - *GetActualOwner*) deve notificar o mecanismo de processos de negócios quando a tarefa é finalizada com sucesso ou sem sucesso.

```
<b4p:peopleActivity
<htd:task name="votingTask">
<htd:interface operation="vote“
                portType="el:votingPT"/>
<htd:peopleAssignments>
<htd:potentialOwners>
<htd:user> NmUser1 </htd:user>
<htd:user> NmUser2 </htd:user>
<htd:user> NmUser3 </htd:user>
</htd:potentialOwners>
<htd:excludedOwners>
<htd:user> NmUser3 </htd:user>
</htd:excludedOwners>
</htd:peopleAssignments>
</htd:task>
</b4p:peopleActivity>
```

Figura 5.1: Exemplo de tarefa humana - *WS-HumanTask*.

Na seção 5.1 a seguir, serão abordadas as restrições de autorização.

5.1 Restrições de Autorização

BPEL4People possui suporte para excluir alguns usuários de executar uma tarefa por causa de algumas tarefas que tenham executado antes, ou então força um usuário a executar uma sequência de tarefas. Nesta tese, essas exigências são chamadas "restrições de autorização", assim também é usado este termo na literatura. Nas Seções 5.2 e 5.3 são usadas GSPN e CPN, respectivamente, para expressar as restrições de autorização através de modelos formais.

"*4-eyes principle*" é um tipo de restrição de autorização proposto na especificação BPEL4People. O "*4-eyes principle*", também conhecido como separação de direito - *separation of duty* (SoD), é um cenário comum em muitas áreas de aplicação, quando uma decisão deve ser tomada por duas ou mais pessoas independentemente uma da outra, às vezes por razões de segurança estabelecidas pelas regras do negócio.

5.1.1 Separação de Direito - *Separation of Duty*

A separação de direito - *Separation of Duty* (SoD) é um princípio bem conhecido na autorização para impedir fraudes ou erros, exigindo que pelo menos duas pessoas

estejam envolvidas em algum trabalho específico. A "SoD" também é útil quando duas pessoas têm de cooperar em um trabalho, mas nenhuma delas deve saber todos os detalhes relativos à execução do mesmo.

A forma básica da "SoD" atesta que dados dois tipos distintos de tarefas "T1" e "T2", estas devem ser realizadas por diferentes pessoas. Isto pode ser verificado pela afirmação que uma pessoa "P0" não pode realizar ambas as tarefas "T1" e "T2". Pode-se definir variações desta mesma forma, por exemplo, as tarefas "T1" e "T2" devem ser realizadas por diferentes pessoas. Pode-se ainda definir restrição "SoD" para uma pessoa específica, por exemplo, a mesma pessoa não pode executar a tarefa "T1" e a tarefa "T2".

5.1.2 Vinculação de Direito - *Binding of Duty*

Vinculação do direito - *Binding of Duty* (BoD) é uma forma dupla de "SoD", que afirma que algumas tarefas distintas devem ser realizadas por uma pessoa. A "BoD" é usada para definir a responsabilidade de uma pessoa, por exemplo: afirma-se que se a pessoa "P0" realiza a tarefa "T1", então "P0" também deve executar "T2", e vice-versa.

A "*Separation of Duty*" (SoD) e a "*Binding of Duty*" (BoD) podem ser combinadas para definir restrições mais complexas.

5.2 Modelagem de BPEL4People em GSPN

Como já mencionado, por ser BPEL4People uma extensão do BPEL, a sua transformação em redes de Petri será tratada em dois momentos. No Capítulo 4, as Seções 4.3.1 e 4.3.2 trataram da transformação do BPEL em GSPN e em CPN, repectivamente. Neste quinto capítulo será tratada a transformação da extensão BEL4People e, mais especificamente, da transformação da especificação "WS-HumanTask" nas duas extensões de redes de Petri: GSPN e CPN.

As atividades básicas dos códigos BPEL4People são aquelas que descrevem os passos de uma atividade elementar. Códigos BPEL4People definem além das atividades básicas do BPEL (mostradas na Seção 4.3.1.1) as seguintes atividades básicas: *<PeopleActivity>*, *<TaskName>*, *<PeopleAssignmets>*, *<PotencialOwners>*, *<humanInteractions>*, *<Tasks>* e *<User>*.

5.2.1 Transformação da Extensão Humana "WS-HumanTask" do BPEL4People em GSPN

A representação da transformação de cada uma das atividades básicas e de cada um dos usuários <User> da cláusula <PotentialOwner> em GSPN é modelada por uma transição "t", por dois lugares "P1" e "P2", e dois arcos ligando estes lugares à transição, como mostrado na Figura 5.2. Um ficha (*token*) no lugar "P1" representa que a atividade básica ou o usuário modelado pela transição "t" esta apto a executar o serviço. O lugar "P2" conterá ficha após o disparo da transição "t" e este disparo representa que a atividade básica foi executada ou o usuário executou o serviço.

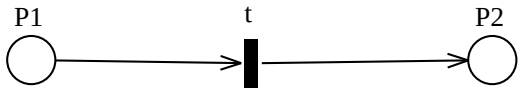

Figura 5.2: Representação das atividades básicas e dos usuários potenciais de um serviço em GSPN.

A transformação da extensão humana "WS-HumanTask" em GSPN segue os passos estabelecidos pelo fluxo de transformação da extensão humana "WS-HumanTask" em GSPN, mostrado na Figura 5.3.

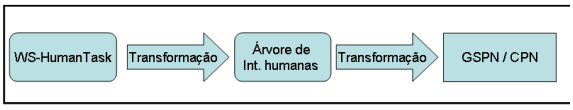

Figura 5.3: Fluxo de transformação da extensão humana "WS-HumanTask" em GSPN.

No fluxo da Figura 5.3, a transformação da extensão humana "WS-HumanTask" em GSPN é realizada pelo algoritmo mostrado na Figura 5.4.

O algoritmo de transformação da extensão humana "WS-HumanTask" em GSPN converte inicialmente a parte do código da extensão humana "WS-HumanTask" em uma estrutura de dados do tipo árvore. Esta árvore é construida obedecendo os critérios de que cada um dos usuários <User> da cláusula <PotentialOwner> de um serviço seja representado por um "nó" da árvore e que em cada nível de profundidade

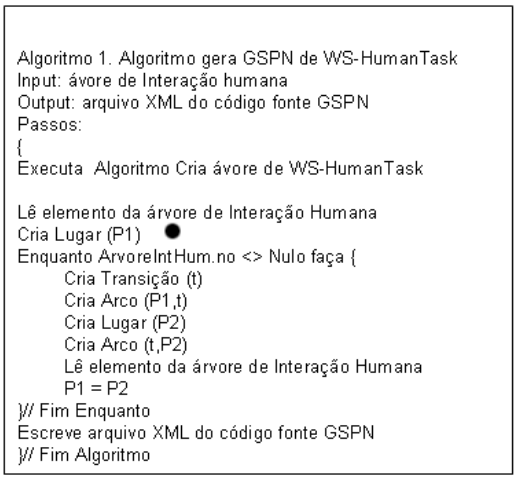

Figura 5.4: Algoritmo de transformação da extensão humana "WS-HumanTask" em GSPN.

desta árvore estejam representadas as possibilidades de execução de um serviço da orquestração do código BPEL4People por seus <*PotentialOwner*>.

Na existência da cláusula <*ExcludedOwners*> para um serviço, deve-se excluir os usuários presentes na cláusula <User> para o referido serviço e para os serviços nos quais existam a cláusula de exclusão do atual dono de um serviço anteriormente executado com a cláusula <GetActualOwner> deve-se consultar todos os "nós" ancestrais na profundidade da árvore que correspondam àquele serviço e excluí-lo da execução do serviço em questão.

O algoritmo de criação da árvore de interações humanas é mostrado na Figura 5.5

Definida a árvore, esta será utilizada para a construção da GSPN. Esta construção é feita através da leitura da árvore anteriormente definida em pré-ordem e da transformação de cada um dos "nós", usuários <User> da cláusula <PotentialOwner>, em GSPN como mencionado no início desta secção. A estrutura de filiação da árvore se mantém na GSPN através de arcos ligando as diversas estruturas GSPN modeladoras dos usuários, <User> da cláusula <PotentialOwner>, dos serviços na orquestração do código BPEL4People.

A Figura 5.6 apresenta a estrutura de dados do tipo árvore para a extensão humana "WS-HumanTask" do código BPEL4Peole da Figura 5.7.

```
Algoritmo 2. Algoritmo Cria ávore de WS-HumanTask
Input: arquivo XML do código fonte WS-HumanTask
Output: estrutura de dados árvore
Passos:
{
P. Parse = Arquivo XML do código fonte WS-HumanTask
Lê elemento da função Dom Parcer
Enquanto P.elem <> Null faça {
...
   Se P.Tag = 'PotencialOwners'  {
    Enquanto PilhaTarefaAnterior <> Nulo faça {
        Enquanto P.Tag = 'User' faça {
            Lê (ArvoreIntHum)
            Enquanto ArvoreIntHum.no <> Nulo faça {
                Cria (ArvoreIntHum.no)
             }// Fim Enquanto
        }// Fim Enquanto
   }// Fim Enquanto
   }// Fim Se
Se P.Tag = 'ExcludedOwners'  {
    Enquanto PilhaTarefaAnterior <> Nulo faça {
        Enquanto P.Tag = 'User' faça {
            Lê (ArvoreIntHum)
            Se ArvoreIntHum.no = P.NomeDono faça {
                Exclui (ArvoreIntHum.no)
             }// Fim Enquanto }// Fim Enquanto
    }// Fim Enquanto
   }// Fim Se
}// Fim Enquanto
Escreve arquivo XML do código fonte GSPN
}// Fim Algoritmo
```

Figura 5.5: Algoritmo de criação da árvore de interações humanas.

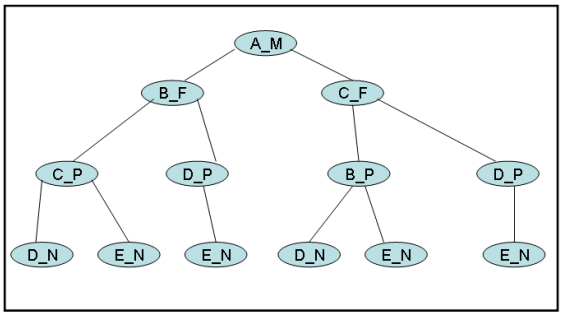

Figura 5.6: Árvore representativa da extensão humana "WS-HumanTask".

5.2.2 Atribuições Temporais ao modelo GSPN da extensão humana "WS-HumanTask" de Códigos BPEL4People

Na transformação da extensão humana "WS-HumanTask" de códigos BPEL4People em GSPN, faz-se uso de transições imediatas, exceto as transições que representam a atividade básica <User> que são modeladas por transições temporizadas e estas recebem como tempo de atraso, os valores das respostas da Função Distribuição de Probabilidade (PDF) do tempo de resposta de cada um dos usuários nos provedores de serviços, onde o serviço invocado será executado.

Para modelar o comportamento estocástico do tempo de resposta de cada um dos usuários nos provedores de serviços, esta tese faz uso da Função Distribuição de Probabilidade (PDF), como já apresentado na Seção 4.3.1.3.

```
<process name="purchasing">
<b4p:humanInteractions><htd:tasks>
<htd:task name="manager_approve">
<htd:peopleAssignments><htd:potentialOwners>
<htd:user>Alan</htd:user></htd:potentialOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="finance_approve">
<htd:peopleAssignments><htd:potentialOwners>
<htd:user>Ben</htd:user>
<htd:user>Cindy</htd:user>
</htd:potentialOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="purchase">
<htd:peopleAssignments>
<htd:potentialOwners>
<htd:user>Ben</htd:user>
<htd:user>Cindy</htd:user>
<htd:user>Diana</htd:user>
</htd:potentialOwners>
<htd:excludedOwners>
<htd:getActualOwner("finance_approve")
</htd:excludedOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="notify_staff">
<htd:peopleAssignments>
<htd:potentialOwners>
```

```
<htd:user>Diana</htd:user>
<htd:user>Edward</htd:user>
</htd:potentialOwners>
<htd:excludedOwners>
<htd:getActualOwner("purchase")
</htd:excludedOwners>
</htd:peopleAssignments>
</htd:task>
</htd:tasks>
</b4p:humanInteractions>
</process>
```

Figura 5.7: Código BPEL4Peole.

Como entradas da PDF, serão utilizados a média aritmética e o desvio padrão dos tempos de resposta de cada um dos usuários nos provedores de serviços (SP), enquanto como saída é esperado o valor do tempo de atraso (*delay time*) da transição, representado pela letra grega (λ). O cálculo do tempo de atraso (*delay time*) da transição segue as mesmas regras mostradas na Seção 4.3.1.3.

5.2.2.1 Estudo de caso de Transformação de BPEL4People em GSPN

Esta seção aborda um estudo de caso de transformação de um código BPEL4People, mostrado na Figura 5.8, para GSPN e em seguida será apresentada sua análise de desempenho.

O estudo de caso trata de um código fonte BPEL4People para a orquestração de um WS denominado de "WS PurchSys". O "WS PurchSys" é um processo de compra cujo código fonte BPEL4People que é mostrado na Figura 5.8. No "WS PurchSys", quatro serviços são definidos: *manager approve, finance approve, purchase e notify staff*. Os potenciais proprietários de cada um destes serviços são: *manager approve* (Alan); *finance approve* (Ben,Cindy); *purchase* (Ben, Cindy, Diana); *notify staff* (Diana, Edward). O proprietário excluído do serviço *purchase* é o usuário <GetAtualOwner> do serviço *finance approve* e o proprietário excluído do serviço *notify staff* é o usuário <GetAtualOwner> do serviço *purchase*.

```
<process name="purchasing">
<b4p:humanInteractions><htd:tasks>
<htd:task name="manager_approve">
<htd:peopleAssignments><htd:potentialOwners>
<htd:user>Alan</htd:user></htd:potentialOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="finance_approve">
<htd:peopleAssignments><htd:potentialOwners>
<htd:user>Ben</htd:user>
<htd:user>Cindy</htd:user>
</htd:potentialOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="purchase">
<htd:peopleAssignments>
<htd:potentialOwners>
<htd:user>Ben</htd:user>
<htd:user>Cindy</htd:user>
<htd:user>Diana</htd:user>
</htd:potentialOwners>
<htd:excludedOwners>
<htd:getActualOwner("finance_approve")
</htd:excludedOwners>
</htd:peopleAssignments></htd:task>
<htd:task name="notify_staff">
<htd:peopleAssignments>
<htd:potentialOwners>
<htd:user>Diana</htd:user>
<htd:user>Edward</htd:user>
</htd:potentialOwners>
<htd:excludedOwners>
<htd:getActualOwner("purchase")
</htd:excludedOwners>
</htd:peopleAssignments>
</htd:task>
</htd:tasks>
</b4p:humanInteractions>
</process>
```

Figura 5.8: BPEL4People para a orquestração do "WS PurchSys".

Na implementação do "WS PurchSys", vários artefatos de *software* foram utilizados. Os *web services* (escritos em Java) foram disponibilizados em servidores *Tomcat*, versão 5.0.28. Adicionalmente, foram utilizados o Ant (*Apache AntUnit* - versão 1.6), o módulo SOAP para o *Apache (Java Web Services* - versão 2.3.1) e o *Apache Axis* (Versão 1.4). Todos estes *softwares* são necessários para a execução de *web services* escritos em Java. Informações adicionais sobre a necessidade e a forma de utilização dos mesmos podem ser encontradas em "*Java Web Services*" (YU; LIN, 2006).

Para possibilitar a orquestração do "WS PurchSys", um motor (*engine*) de BPEL4People foi utilizado. Pela razão de ser *Open-Source*, a *engine* ActiveBPEL (versão 2.0) foi adotada.

5.2.2.2 Modelo GSPN para o *Web Service* "WS PurchSys"

No estudo de caso do "WS PurchSys" em questão, a camada de transformação da arquitetura "SOASPE" receberá o código BPEL4People do "WS PurchSys" mostrado na Figura 5.8 e os tempos de resposta de cada um dos usuários nos provedores de serviços. Com estes dados de entrada, as regras de transformação gerarão a GSPN para o "WS PurchSys" de acordo com as diretrizes definidas na Seção 4.3.1 (transformação BPEL para GSPN) e na Seção 5.2 (transformação BPEL4People para GSPN).

A Figura 5.9 mostra o modelo GSPN do código BPEL4People para o "WS PurchSys", apresentado na Figura 5.8.

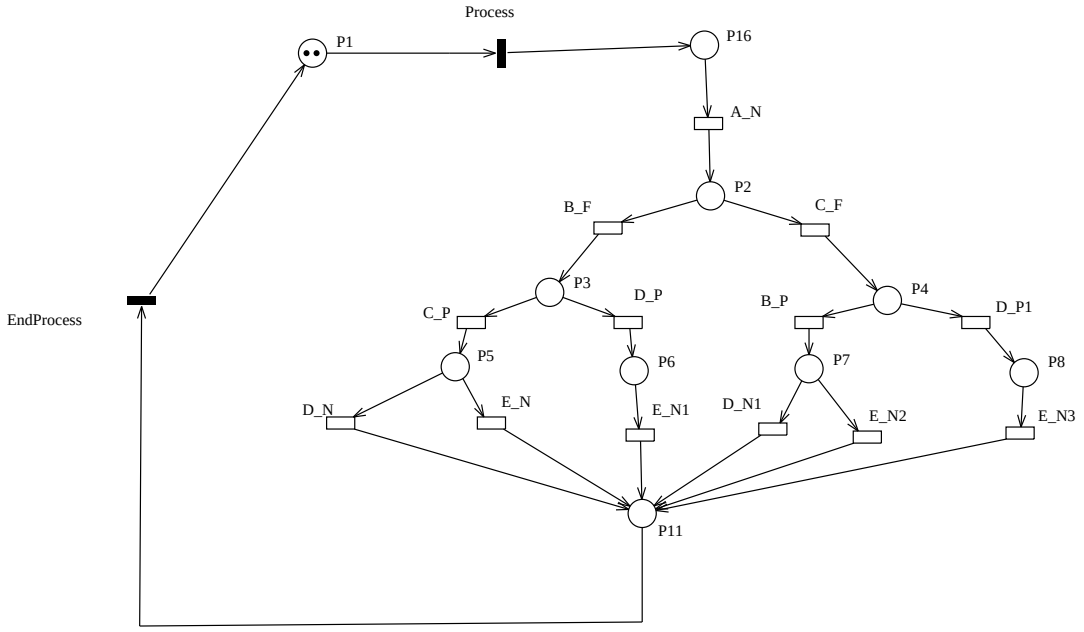

Figura 5.9: Modelo GSPN para Código BPEL4People do "WS PurchSys".

5.2.2.3 Análise de Desempenho do Modelo GSPN para o "WS PurchSys"

Nesta seção será abordada a utilização da arquitetura "SOASPE" para simular a avaliação de desempenho do "WS PurchSys" a partir do modelo GSPN gerado a partir do código BPEL4People mostrado na Figura 5.8.

Com o modelo GSPN para código BPEL4People para o "WS PurchSys" carregado na ferramenta GREATSPN começam as atividades de avaliação de desempenho. A análise do desempenho do modelo é feita a partir de simulações com a mesma quantidade de requisições feitas no "WS PurchSys". O número de requisições para a análise do desempenho do estudo de caso em questão variou de 10 a 30 requisições.

A Figura 5.10 mostra o gráfico dos tempos de resposta reais do "WS PurchSys" e dos tempos de resposta do modelo GSPN do "WS PurchSys" gerados a partir da arquitetura "SOASPE".

Os resultados mostram que os tempos de resposta do modelo GSPN gerado pela arquitetura "SOASPE" e os tempos de resposta do "WS PurchSys" não diferem

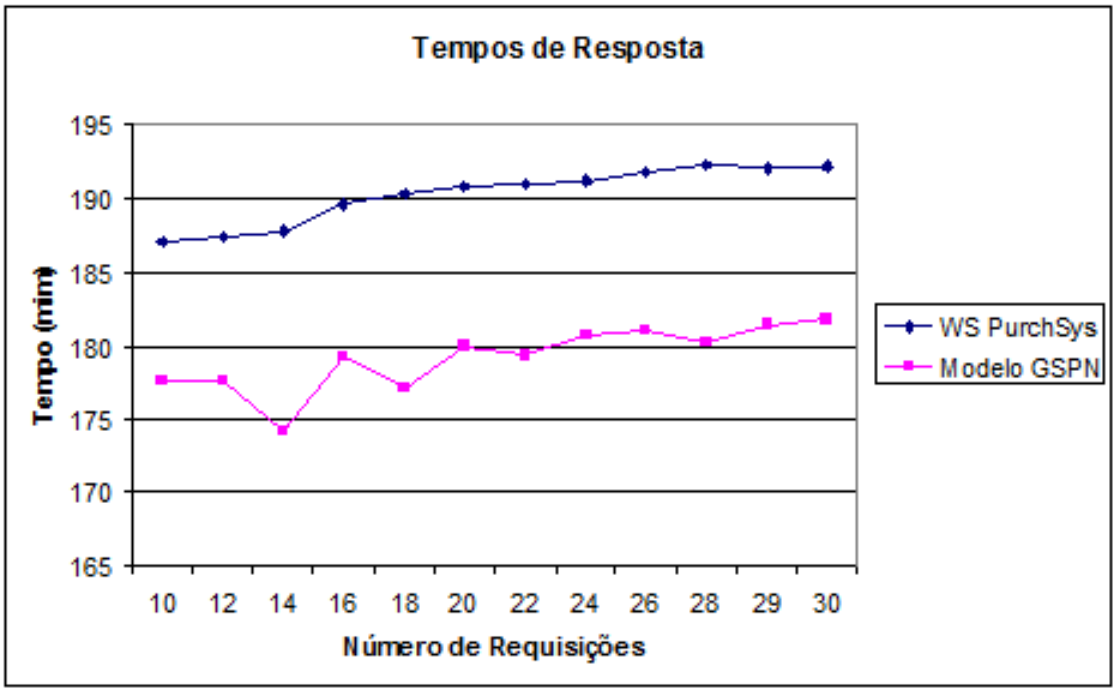

Figura 5.10: Tempos de resposta do "WS PurchSys" e do modelo GSPN gerado a partir da arquitetura "SOASPE" para o "WS PurchSys".

em mais de 8,0%, comprovando-se assim a validade da arquitetura "SOASPE" na análise de desempenho de *web services* orquestrados pelo BPEL4People, ou seja, de *web services* onde existem interações humanas.

A Figura 5.11 apresenta o percentual comparativo dos tempos de resposta do modelo GSPN gerado pela arquitetura "SOASPE" para o "WS PurchSys" e dos tempos de resposta reais do "WS PurchSys".

5.3 Modelagem de BPEL4People em CPN

Como já foi mencionado anteriormente, as atividades básicas dos códigos BPEL4People são aquelas que descrevem os passos de uma atividade elementar. Códigos BPEL4People definem além das atividades básicas do BPEL (mostradas na Seção 4.3.2.1) as seguintes atividades básicas: *<PeopleActivity>*, *<TaskName>*, *<PeopleAssignmets>*, *<PotencialOwners>*, *<humanInteractions>*, *<Tasks>* e *<User>*.

Na transformação de códigos BPEL4People em CPN os seguintes tipos coloridos foram definidos, conforme mostrado na Figura 5.12.

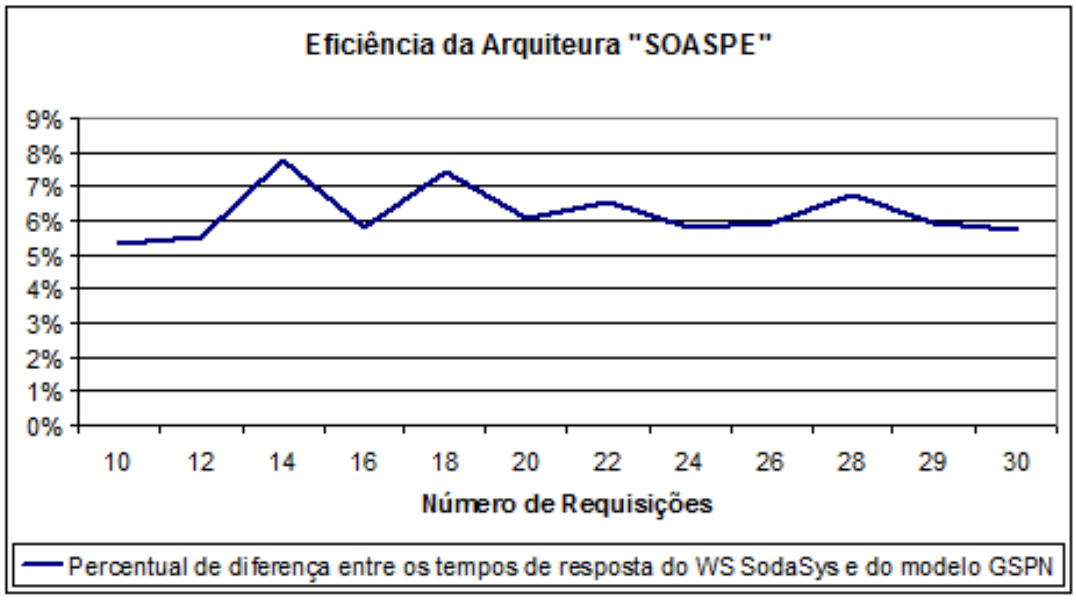

Figura 5.11: Eficiência do framework "SOASPE" para análise de desempenho de WS orquestrados pelo BPEL4People em GSPN.

```
Declarations
  colset INT = int;
  colset Tarefa = with TAR;
  colset ProcTime = INT;
  colset Proc= product Tarefa*ProcTime;
  var PROCTIME,t,t1,t2: ProcTime;
```

Figura 5.12: Tipos coloridos das redes CPN dos códigos BPEL4People.

5.3.1 Transformação da Extensão Humana "WS-HumanTask" de Códigos BPEL4People em CPN

A representação da transformação de cada uma das atividades básicas e de cada um dos usuários <User> da cláusula <PotentialOwner> em CPN é modelada por uma transição "t", por dois lugares "P1" e "P2", e dois arcos ligando estes lugares à transição, como mostrado na Figura 5.13. Uma ficha (*token*) no lugar "P1" representa que a atividade básica ou o usuário modelado pela transição "t" esta apto a executar. O lugar "P2" conterá ficha após o disparo da transição "t" e este disparo representa que a atividade básica foi executada ou o usuário executou o serviço.

A transformação da extensão humana "WS-HumanTask" em CPN, da mesma

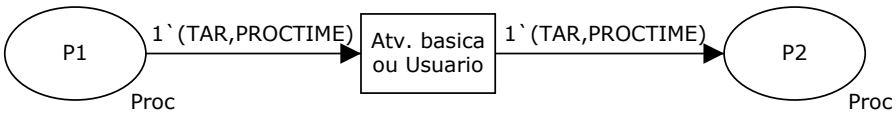

Figura 5.13: Representação das atividades básicas e dos usuários potenciais de um serviço em CPN.

forma que o algoritmo de transformação da extensão humana "WS-HumanTask" em GSPN, converte inicialmente a parte do código da extensão Humana "WS-HumanTask" em uma estrutura de dados do tipo árvore. Esta árvore é contruida obdecendo os critérios de que cada um dos usuários <User> da cláusula <PotentialOwner> de um serviço seja representado por um "nó" da árvore e que em cada nível de profundidade desta árvore, estejam representados as possibilidades de execução de um serviço da orquestração do código BPEL4People por seus *<PotentialOwner>*.

Na existência da cláusula *<ExcludedOwners>* para um serviço, devem-se excluir os usuários presentes na cláusula <User> para o referido serviço e para os serviços nos quais existam a cláusula de exclusão do atual dono de um serviço anteriormente executado com a cláusula <GetActualOwner>, deve-se consultar todos os "nós" ancestrais na profundidade da árvore que correspondam à aquele serviço e excluí-lo da execução do serviço em questão.

Definida a árvore, esta será utilizada para a construção da CPN. Esta construção é feita através da leitura da árvore anteriormente definida em pré-ordem e da transformação de cada um dos "nós", usuários <User> da cláusula <PotentialOwner>, em CPN como mencionado no início desta seção. A estrutura de filiação da árvore se mantém na CPN através de arcos ligando as diversas estruturas CPN modeladoras dos usuários, <User> da cláusula <PotentialOwner>, dos serviços na orquestração do código BPEL4People.

5.3.2 Atribuições Temporais ao modelo CPN da extensão humana "WS-HumanTask" de Códigos BPEL4People

Na transformação da extensão humana "WS-HumanTask" de códigos BPEL4People em CPN, faz-se uso de transições que passarão como valor do tipo colorido '*proctime*' o valor 0 (zero), exceto as transições que modelam as atividades <User>. Estas serão modeladas por transições que passarão como valores do tipo colorido '*proctime*' os valores das PDF do tempo de resposta de cada um dos usuários

nos provedores de serviços, onde o serviço invocado será executado.

Para modelar o comportamento estocástico do tempo de resposta de cada um dos usuários nos provedores de serviços, faz-se uso da Função Distribuição de Probabilidade (PDF), como já apresentado na Seção 4.3.1.3.

Como entradas da PDF, serão utilizados a média aritmética e o desvio padrão dos tempos de resposta de cada um dos usuários nos provedores de serviços (SP), enquanto como saída da PDF é esperado um valor numérico para ser atribuído ao tipo colorido "*proctime*", que tem a função de guardar o atraso provocado por estes usuários quando da resposta pela invocação do serviço em um provedor de serviço. O cálculo do tipo colorido "*proctime*" segue as mesmas regras mostradas na Seção 4.3.2.3.

5.3.3 Estudo de caso de Transformação de BPEL4People para CPN

Nesta seção será abordado um estudo de caso de transformação de um código BPEL4People, mostrado na Figura 5.8, para CPN e, em seguida, será apresentado sua análise de desempenho.

Este estudo de caso também utiliza o código fonte BPEL4People para a orquestração do "WS PurchSys". O WS "WS PurchSys", como já foi mencionado, é um WS de um processo de compra que é mostrado na Figura 5.8. Quatro serviços são definidos: *manager approve, finance approve, purchase e notify staff*. Os potenciais proprietários de cada tarefa são: *manager approve* (Alan); *finance approve* (Ben,Cindy); *purchase* (Ben, Cindy, Diana); *notify staff* (Diana, Edward). O proprietário excluído do serviço *purchase* é o usuário <GetAtualOwner> do serviço *finance approve* e o proprietário excluído do serviço *notify staff* é o usuário <GetAtualOwner> do serviço *purchase*.

5.3.3.1 Modelo CPN para o "WS-PurchSys"

No estudo de caso em questão, a camada de transformação do *framework* "SOASPE" receberá o código BPEL4People do *web services* - "WS PurchSys" mostrado na Figura 5.8 e os tempos de resposta de cada um dos usuários nos provedores de serviços. Com estes dados de entrada, as regras de transformação gerarão a CPN para o "WS PurchSys" de acordo com as diretrizes definidas na Seção 5.2 (transformação BPEL para CPN) e na Seção 5.3 (transformação BPEL4People para CPN).

A Figura 5.14 mostra o modelo CPN do código BPEL4People para o "WS PurchSys", apresentado na Figura 5.8.

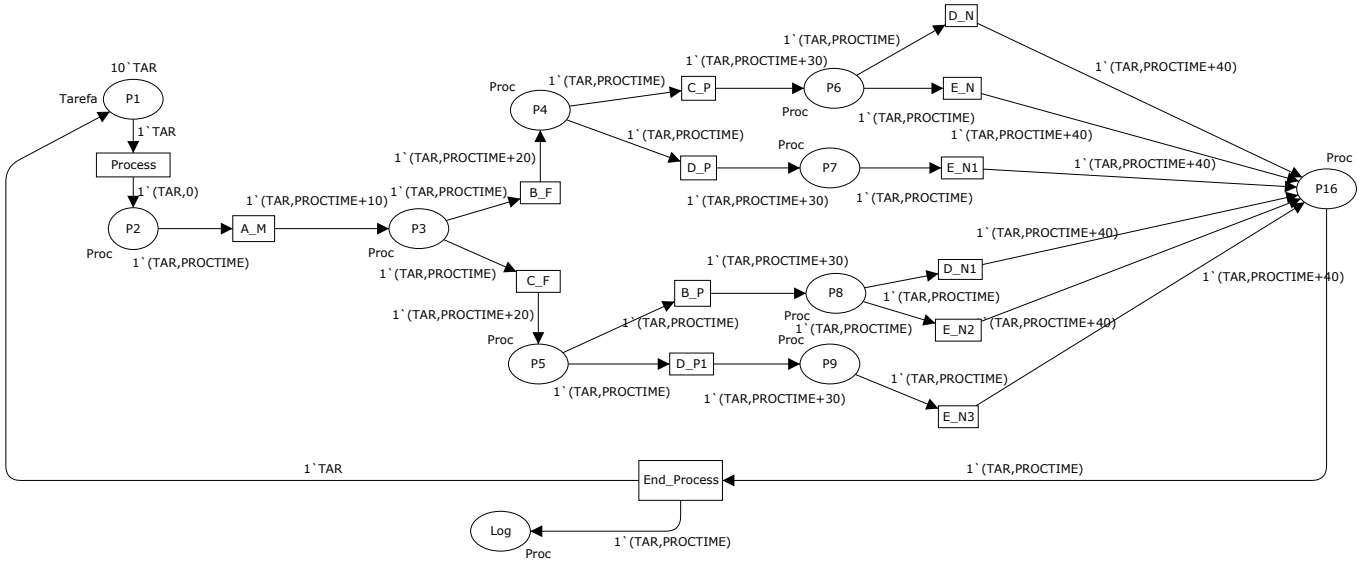

Figura 5.14: Modelo CPN para Código BPEL4People do "WS-PurchSys".

5.3.3.2 Análise de Desempenho do Modelo CPN para o "WS PurchSys"

Nesta seção será abordada a utilização da arquitetura "SOASPE" para a avaliação de desempenho do "WS PurchSys" no modelo CPN gerado a partir do código BPEL4People mostrado na Figura 5.8.

Com o modelo CPN para código BPEL4People para o "WS PurchSys" carregado na ferramenta CPN-*tools*, começam as atividades de avaliação de desempenho. A análise do desempenho do modelo é feita a partir de simulações com a mesma quantidade de requisições feitas no "WS PurchSys". O número de requisições para a análise do desempenho do estudo de caso em questão variou de 10 a 30 requisições.

A Figura 5.15 mostra o gráfico dos tempos de resposta reais do "WS PurchSys" e dos tempos de resposta do modelo CPN do "WS PurchSys" gerado a partir da arquitetura "SOASPE".

Os resultados mostram que os tempos de resposta do modelo CPN gerado pela arquitetura "SOASPE" e os tempos de resposta do "WS PurchSys" não diferem em mais de 9,0%, validando assim a arquitetura "SOASPE" na análise de desempenho de *web services* orquetrados pelo BPEL4People, ou seja, de *web services* onde existem interações humanas.

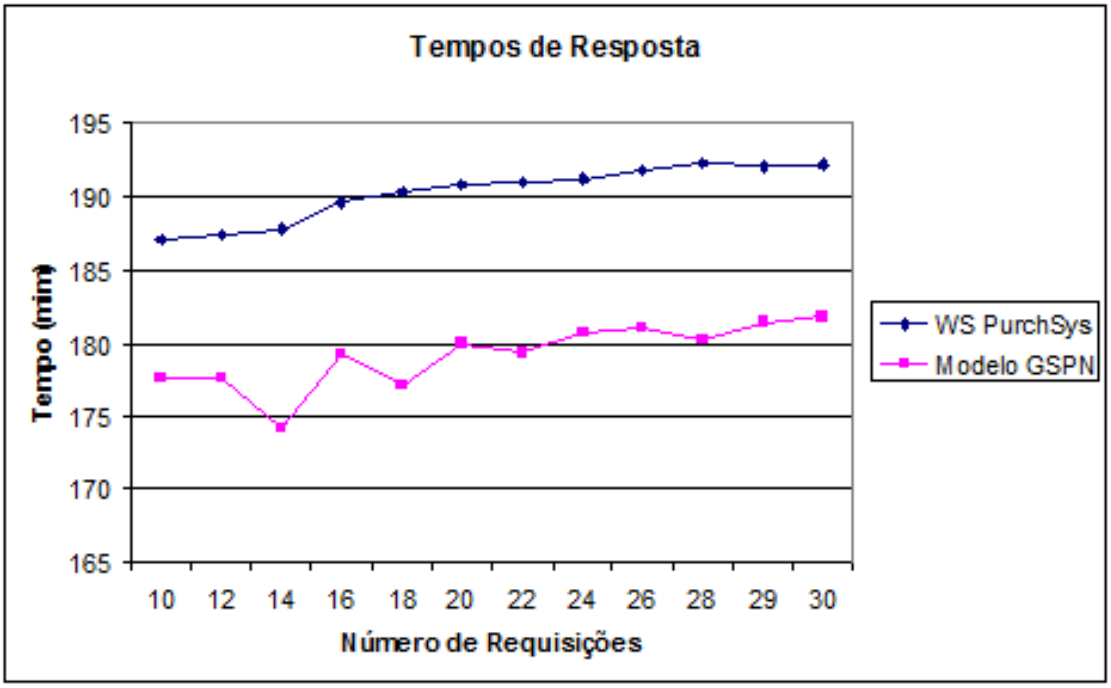

Figura 5.15: Tempos de resposta do "WS PurchSys" e do modelo CPN gerado a partir da arquitetura "SOASPE" para o "WS PurchSys".

A Figura 5.16 apresenta o percentual comparativo dos tempos de resposta do modelo CPN gerado pela arquitetura "SOASPE" para o "WS PurchSys" e dos tempos de resposta do reais do "WS PurchSys".

5.4 Conclusões do Capítulo

Neste capítulo foi tratada a transformação da extensão BPEL4People e mais especificamente da transformação da especificação "WS-HumanTask", nas duas extensões de redes de Petri: GSPN e CPN. Foi apresentada também a questão relativa à avaliação de desempenho dos WS baseada na execução de processos orquestrados onde exista a inclusão explícita das tarefas humanas (*WS-HumanTask*) em processos BPEL (extensão para pessoas, chamada de BPEL4People) e sendo estes processos transformados em duas extensões de redes de Petri, as GSPN e as CPN.

As análises de desempenho dos modelos GSPN e CPN do *web service* - "WS-PurchSys" mostraram-se satisfatórias, pois os tempos de resposta dos modelos (GSPN e CPN) gerados pela arquitetura "SOASPE" e os tempos de resposta do "WS PurchSys" não diferem em mais de 9,7%, validando assim a arquitetura "SOASPE"

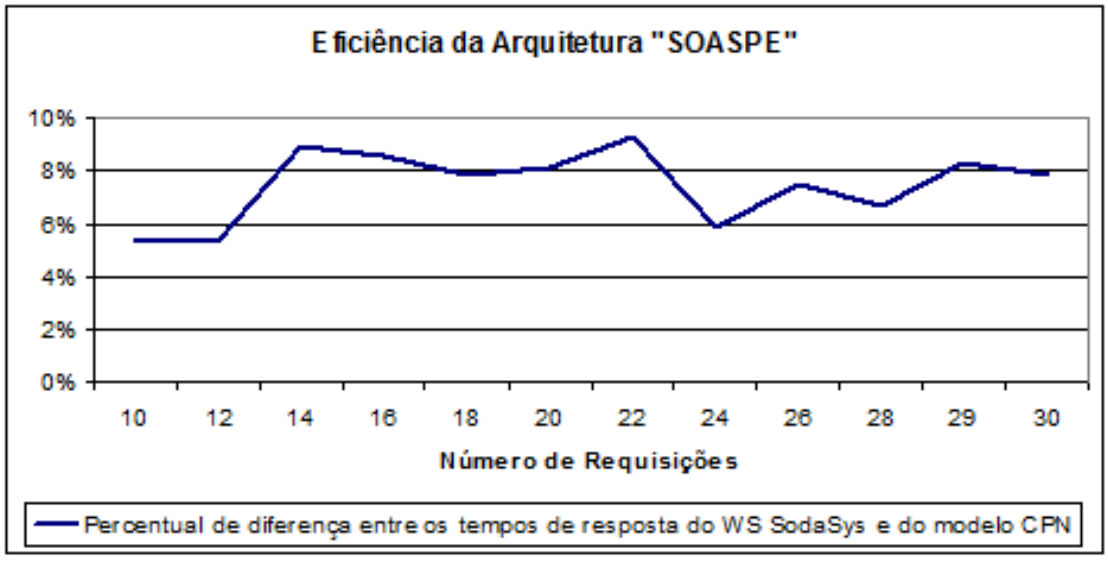

Figura 5.16: Eficiência do framework "SOASPE" para análise de desempenho de WS orquestrados pelo BPEL4People em CPN.

na avaliação de desempenho de WS orquestrados com o BPEL4People, existindo nestes a inclusão explícita das tarefas humanas (*WS-HumanTask*).

A modelagem em GSPN mostrou-se um pouco mais precisa do que a modelagem CPN, pois as redes de Petri (GSPN) modelam com maior precisão o comportamento estocástico do tempo de resposta de cada um dos serviços nos provedores de serviços e também o tempo de resposta de cada um dos usuários nos provedores de serviços, onde o serviço invocado será executado, do que as redes CPN que tem a limitação de definir os tempos de resposta (atrasos das transições) com valores inteiros.

No Capítulo 6 serão apresentados o projeto e a implementação de uma ferramenta automática de conversão de BPEl4People em GSPN, a ferramenta "TPeople4PN" .

Capítulo 6

SOASPE: Projeto e Implementação de uma Ferramenta "TPeople4PN"

No Capítulo 4, nas Seções 4.3.1 e 4.3.2, foram apresentadas as regras de transformação de estruturas BPEL em GSPN e em CPN. No Capitulo 5 foram apresentadas as regras da transformação da extensão humana "WS-HumanTask" do BPEL4People em GSPN e CPN. Neste capítulo serão apresentados o projeto e a implementação de uma ferramenta automática de conversão de BPEl4People em GSPN, a ferramenta "TPeople4PN" .

Na atual versão da ferramenta de transformação "TPeople4PN" somente foi implementada a transformação automática de códigos BPEL e de sua extensão BPEl4People em GSPN. As regras de transformação destes códigos em CPN, somente estão descritas teoricamente nas Seções 4.3.2 e 5.3.

6.1 O Projeto e a Implementação da Ferramenta "TPeople4PN"

A ferramenta de transformação de BPEL4People para GSPN foi desenvolvida utilizando a plataforma Java. O processo de geração automática de BPEL4People em CPN será contemplado em um versão futura da ferramenta "TPeople4PN".

O projeto da ferramenta "TPeople4PN" é composta pelas classes:

- JMenuDemo;
- DomParse;

- CreatePetriNet;
- GreatSPNFileWriter
- Temporização; e
- Estatística.

6.1.1 A Interface da Ferramenta "TPeople4PN"

A classe JMenuDemo define a *interface* da ferramenta "TPeople4PN" para a interação com os usuários. Esta *interface* é apresentada na Figura 6.1 com o arquivo XML do código BPEL4People a ser transformado em GSPN.

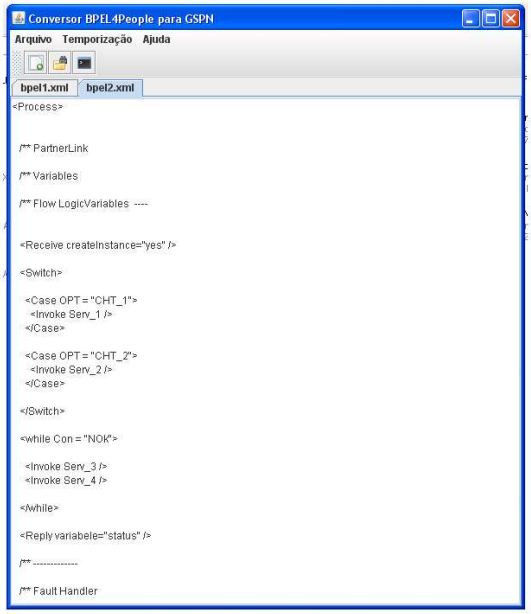

Figura 6.1: *Interface* da ferramenta "TPeople4PN" com arquivo XML do BPEL4People.

A Figura 6.2 mostra a *interface* da ferramenta "TPeople4PN" após a axecução da opção "Converter" de seu *Menu*. A opção "Converter" utiliza as regras definidas pela arquitetura "SOASPE" para transformar códigos BPEL4People em códigos fonte de GSPN.

A *interface* da ferramenta "TPeople4PN" apresenta também um *Menu* composto pelas opções: "Abrir" e "Converter". A opção "Abrir" solicita a localização e nome do arquivo XML do código BPEL4People a ser transformado em GSPN. A partir do momento que este arquivo é localizado e identificado, o mesmo é mostrado na *interface*, Figura 6.1.

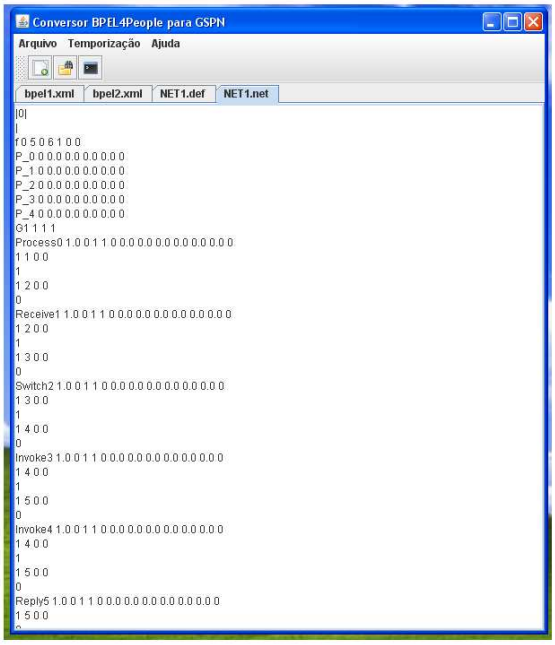

Figura 6.2: *Interface* da ferramenta "TPeople4PN" com arquivo XML da GSPN.

A opção "Converter" do *Menu* chama as demais classes do projeto da ferramenta "TPeople4PN" que seguirão as regras estabelecidas pela arquitetura "SOASPE" para transformação de código BPEL4People em GSPN. A finalização das tarefas da opção "Converter" são a geração do arquivo XML do código fonte da rede GSPN e a apresentação deste referido arquivo na *interface* da ferramenta "TPeople4PN", Figura 6.2.

6.1.2 A Classe "DomParse"

Na transformação do código BPEL4People (XML) em GSPN inicialmente deve-se converter o arquivo XML do código BPEL4People em uma árvore com a classe "DomParser".

O *Parsing* é o processo de leitura e divisão do documento em elementos, atributos, entidades, comentários e outros tipos de dados, por meio do qual poderão ser analisados e validados. O *Parsers XML* é uma biblioteca de *software* (uma classe em Java) que lê o documento XML e verifica se ele é bem formado e válido. O DOM (*Document Object Model*) é um conjunto de *interfaces* que decompõe um documento XML em uma árvore transversal hierárquica de nós/objetos. A Figura 6.3 mostra a classe "DomParser".

Para trabalhar com a informação de um arquivo XML, é aconselhável primeiro

```
package parser;import java.io.File;import java.io.IOException;import
javax.xml.parsers.DocumentBuilderFactory;import
javax.xml.parsers.ParserConfigurationException;import
org.w3c.dom.Document;import org.w3c.dom.Node;import
org.w3c.dom.NodeList;import org.xml.sax.SAXException;
public class DOMParser{
        private Document doc = null;
        public DOMParser(){
                try{
                        doc = parserXML(new File("parser/file.xml"));
                        visit(doc, 0);          }
                catch(Exception error) {
                                error.printStackTrace(); }
        }
       public void visit(Node node, int level){
                NodeList nl = node.getChildNodes();
                for(int i=0, cnt=nl.getLength(); i<cnt; i++){
                    System.out.println("["+nl.item(i)+"]"); visit(nl.item(i), level+1);}
        }
       public Document parserXML(File file) throws SAXException,
                                IOException, ParserConfigurationException
        { return
                DocumentBuilder
                Factory.newInstance().newDocumentBuilder().parse(file); }
       public static void main(String[] args){
                new DOMParser();}
}
```

Figura 6.3: A classe "DomParser".

fazer o *parse* para criar um objeto (*Document*).

A Figura 6.4 exemplifica o processo de parsear um documento XML com a API DOM.

A API DOM é uma *interface* de programação, baseada no modelo de objetos, que permite a manipulação e transformação de documentos em XML. A *interface* DOM manipula documentos XML na forma de uma estrutura em árvore. Quando um documento é carregado na memória do computador, suas estruturas podem ser lidas e manipuladas através do objeto DOM (YU; LIN, 2006).

A *interface* DOM faz a representação do documento XML na forma de uma árvore de objetos, onde cada nó da árvore representa um componente do documento.

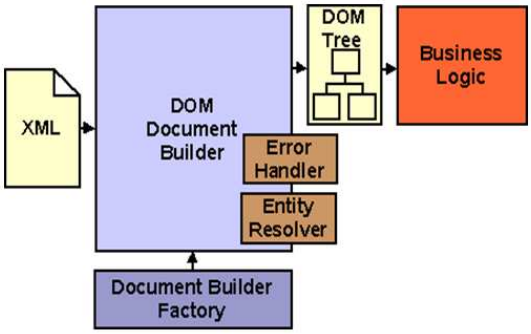

Figura 6.4: O processo de parsear um XML com a API DOM.

Após o processamento completo do documento XML, a memória deverá conter a árvore de objetos DOM, a qual disponibiliza para as aplicações qualquer informação relacionada à estrutura e ao conteúdo do documento.

O modelo implementado pela *interface* DOM é a forma mais flexível para manipulação do conteúdo de um documento XML. Este modelo não é limitado pela ordem em que as estruturas do documento são processadas, logo é possível navegar pelo documento em qualquer ordem. Contudo, como a utilização do objeto DOM requer a leitura de toda estrutura XML em uma árvore na memória, pode acontecer um alto consumo de recursos da máquina (YU; LIN, 2006).

A Figura 6.5 representa o mapa DOM (YU; LIN, 2006).

Figura 6.5: O mapa DOM.

A Figura 6.6 mostra algumas *interfaces* importantes da API DOM (YU; LIN, 2006).

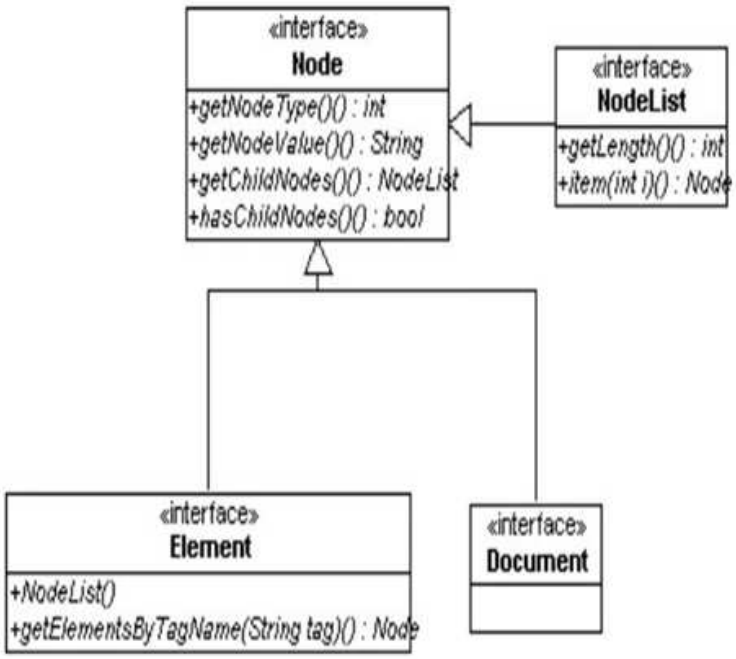

Figura 6.6: Interfaces da API DOM.

6.1.3 As Classes "CreatePetriNet" e "GreatSPNFileWriter"

As classes "CreatePetriNet" e "GreatSPNFileWriter" são as principais classes do projeto da ferramenta de transformação de códigos BPEL4People em GSPN, "TPeople4PN". Suas importâncias se devem ao fato de que nelas é que é feito todo o processo de leitura dos códigos (XML) BPEL4People, definidas as regras de conversão e a geração dos arquivos contendo os códigos (XML) das redes GSPN.

6.1.3.1 A Classe "CreatePetriNet"

A função principal da classe "CreatePetriNet" é a leitura da árvore gerada pela classe "DomParser" mostrada na Figura 6.7, em seguida a comparação de cada "tag" de cada nó da árvore com os elementos da sintaxe do BPEL4People e dando sequência com a transformação em seus respectivos modelos em GSPN, seguindo as regras da camada de transformação da arquitetura "SOASPE", apresentadas na Seção 4.3.1 e na Seção 5.2.

```
public void ReadWhile(Element root){
            System.out.println("ReadWhile =>"+root.getTagName());
            String s = root.getTagName();

            NodeList a;
            Node node = null;

            if(root.hasChildNodes()){
                  a = root.getChildNodes();
                  for(int i = 0;i < a.getLength();i++){
                              node = a.item(i);
                              if(node.getNodeType() == Node.ELEMENT_NODE){

                                          root = (Element)node;
                                    ReadWhile(root);
                              }
                  }

            }
}
```

Figura 6.7: Função de leitura da árvore gerada pela classe "DomParser".

A função "OperaçãoBasica", mostrada na Figura 6.8, utiliza um tipo estruturado que auxilia na comparação de cada "tag" de cada nodo da árvore com os elementos da sintaxe do BPEL4People.

Identificado o tipo de "tag" de cada nodo da árvore com os elementos da sintaxe do BPEL4People, a sua transformação em seus respectivos modelos em GSPN segue utilizando as funções de criação de transições, lugares e arcos da classe "CreatePetriNet". As funções de criação de transições, lugares e arcos são mostradas nas Figuras 6.9, 6.10 e 6.11, respectivamente.

Como já definido pelo *framework* "SOASPE" na Seção 4.3.1.3 e na Seção 5.2.2, na transformação de códigos BPEL4People em GSPN, faz-se uso de transições imediatas, exceto:

- as transições que representam a atividade básica <*Invoke*> que são modeladas por transições temporizadas e recebem como tempo de atraso, os valores das respostas da Função Distribuição de Probabilidade (PDF) do tempo de resposta de cada um dos provedores de serviços, onde os serviços invocados pela atividade básica <*Invoke*> são executados; e

```
public boolean OperacaoBasica(String s){
        if (   s == "Process"
               || s == "Invoke"
               || s == "Receive"
               || s == "Reply"
               || s == "Wait"
               || s == "Empty")
               || s == "PeoleActivity"
               || s == "Taskname"
               || s == "PeopleAssignmets"
               || s == "PotencialOwner"
               || s == "User")
           return true;
        else return false;
    }
```

Figura 6.8: Função "OperaçãoBasica".

```
public GSPNTransition CreateTrans(String s){

        GSPNTransition prob_trans = new GSPNTransition (s+trans++,new
RealExpression(new Float(1.0)),
                        0,NO_TIMED,0,0,ZERO,ZERO,ZERO,ZERO,ZERO,ZERO);

        net.addTransition(prob_trans);
        return prob_trans;
    }
```

Figura 6.9: Função de criação de transições.

- as transições que representam a atividade básica <User> que são modeladas por transições temporizadas e estas recebem como tempo de atraso, os valores das respostas da Função Distribuição de Probabilidade (PDF) do tempo de resposta de cada um dos usuários nos provedores de serviços, onde o serviço invocado será executado.

As classes "Temporização" e "Estatistica" são as responsáveis pela temporização das GSPN.

- A Classe "Temporização"

  Na Seção 4.3 (A Camada de Transformação) a funcionalidade da camada de transformação estabelece que, em todos os provedores de serviços que possuam

```
public GSPNPlace CreatePlace(){

        GSPNPlace init_place = new GSPNPlace("P_"+lugar++,0,ZERO,ZERO,ZERO,ZERO);
        net.addPlace(init_place);
        return init_place;
    }
```

Figura 6.10: Função de criação de lugares.

```
public void CreateArc(GSPNPlace P,GSPNTransition T,boolean direction){
        net.createArc("arc",P,T,1,direction,false);
    }
```

Figura 6.11: Função de criação de arcos.

serviços básicos que possam ser invocados pelo processo de negócio, exista um aplicativo executando neste provedor que guarde (em arquivos XML) todos os tempos de resposta desse provedor e dos usuários (potenciais executores dos serviços) nos provedores de serviços.

Estes arquivos de tempos de resposta dos diversos servidores e dos usuários (potenciais executores dos serviços) são utilizados no cálculo, através de funções Distribuição de Probabilidade, dos tempos de atraso das transições que modelaram a invocação de cada um destes seviços e dos usuários (potenciais executores dos serviços) nos provedores de serviços, no modelo GSPN.

A classe "Temporização" lê os arquivos XML contendo tempos de resposta dos diversos servidores e dos usuários (potenciais executores dos serviços), através da função "ReaderFile", mostrada na Figura 6.12, em seguida calcula o tempo de atraso para a transição que modela a invocação do referido serviço no provedor de serviço. As diretrizes para o cálculo do tempo de atraso, encontram-se detalhadas na Seção 4.3.1.3 e na Seção 5.2.2.

- A Classe "Estatistica"

  Para modelar o comportamento estocástico do tempo de resposta dos provedores de serviços e do tempo de resposta dos usuários (potenciais executores dos serviços), faz-se uso da Função Distribuição de Probabilidade (PDF). Como entradas da PDF, serão utilizados a média aritmética e o desvio

```
import java.io.*;

   public class ReaderFile {

       static double [] Tempos = new double[100];
       static int num=0;
       static Estatistica e = new Estatistica();
              public static void main(String[] args){
           try {
               BufferedReader in = new BufferedReader(new FileReader("/home/vando/servidor1.txt"));
               String str;

               while((str = in.readLine()) != null){
                       Tempos[num] = Double.parseDouble(str);
                       num++;
                       e.setArray(Tempos);
                  System.out.println(str);
               }
               in.close();
           }
           catch (IOException e){
               // possiveis erros são tratatos aqui
           }

           System.out.print("\n mediana: "+ e.getMediana());

           System.out.print("\n Soma Elementos: "+e.getSomaDosElementos());

           System.out.print("\n Media Aritmética: "+e.getMediaAritmetica());

           System.out.print("\n SomaDosElementosAoQuadrado: "+e.getSomaDosElementosAoQuadrado());

           System.out.print("\n Variância Amostral: "+e.getVariancia());

           System.out.print("\n Desvio Padrão: "+e.getDesvioPadrao());

           System.out.print("\n Coeficiente Variação de Pearson: "+e.getPearson());

           System.out.print("\n Moda: "+e.getModa());

           System.out.print("\n Coeficiente de Assimetria: "+e.getCoefAssimetria());

           System.exit(0);
         }
       }
```

Figura 6.12: Função de leitura de arquivos de tempos: dos provedores de serviços e dos usuários (potenciais executores dos serviços).

padrão dos tempos de resposta, obtidos através da função "ReadFile" da classe "Temporização", enquanto como saída é esperado o valor do tempo de atraso - *delay time* da transição que modela a invocação de cada um destes seviços nos provedores de servidores no modelo em rede de Peri (GSPN), representado pela letra grega (λ).

Dependendo do valor do coeficiente de variação (CV), que é calculado, como mostrado na Figura 4.10, esses tempos de resposta são aproximados por uma das distribuições: Hiperexponential ou Hipoexponential (SILVA; LINS, 2006). Isto torna possível representar a questão estocástica (aleatória) envolvida na aproximação dos tempos de resposta ao tempo de atraso (λ) da transição que o modela. A Figura 6.13 mostra a função estatística que faz a aproximação dos tempos de resposta por uma das distribuições: hiperexponential ou hipoexponential.

```
public class Estatistica {
        double u,Teta,Lambda,Upsilon,CV,Sigma,Fi;
    double desvioPadrao;

    public void HiperExponencial(){
       Lambda = 2 * u;
       Lambda = Lambda / u*u + Teta*Teta;
    }

    public void HipoExponencial(){
        Upsilon = (u/2) * (u/2) -1;
        Sigma = u + Upsilon * (Upsilon + 1) * Teta - Upsilon * (u*u);
        Sigma = Sigma / Upsilon + 1;
        Lambda = 1 / Sigma;
    }
}
```

Figura 6.13: Função de aproximação dos tempos de resposta dos SP.

6.1.3.2 A Classe "GreatSPNFileWriter"

Realizadas as taferas das classes: CreatePetriNet, Temporização e Estatistica. A classe "GreatSPNFileWriter" está apta a criar os arquivos XML de especificação das redes GSPN para um código BPEL4People.

A Figura 6.14 mostra a função "CreateGreatSPNFiles" que cria os arquivos de extenções (.net, .def e .eti) que especificam as redes GSPN na formatação da ferramenta GreatSPN.

No próximo capítulo serão apresentadas as Conclusões obtidas neste trabalho de tese e sugestões de trabalhos futuros.

```
private void createGreatSPNFiles (String p_name, File p_directory) throws ProgramError {

        // Primeiro, nós criamos o arquivo .net .
        debugMessage_stdout("[createGreatSPNFiles] Directory path = " +
p_directory.getAbsolutePath());
        try {
           FicheroNet = new FileWriter(p_directory.getAbsolutePath() + "/" +
                        p_name + ".net");

            salidaNet = new PrintWriter(FicheroNet);

        } catch (IOException ex) {
            errorMessage("[createGreatSPNFiles] Exception: " + ex.getMessage());

            throw new FileOperationError("File '" + p_name +
                    ".net' could not be created.");
        }
        // depois o arquivo .def
        try {
            FicheroDef = new FileWriter(p_directory.getAbsolutePath() + "/" +
                        p_name + ".def");
            salidaDef = new PrintWriter(FicheroDef);
        } catch (IOException ex) {
            errorMessage("[createGreatSPNFiles] Exception: " + ex.getMessage());

            throw new FileOperationError("File '" + p_name +
                    ".def' could not be created.");
        }
        // depois o arquivo .eti
        try {
            FicheroEti = new FileWriter(p_directory.getAbsolutePath() + "/" +
                        p_name + ".eti");
            salidaEti = new PrintWriter(FicheroEti);
        } catch (IOException ex) {
            errorMessage("[createGreatSPNFiles] Exception: " + ex.getMessage());

            throw new FileOperationError("File '" + p_name +
                    ".eti' could not be created.");
        }
    }
```

Figura 6.14: Função de criação dos arquivos de definição das GSPN.

Capítulo 7

Conclusões e Trabalhos Futuros

O modelo SOA traz vários novos benefícios para a arquitetura e *design* de *software*, permitindo a reutilização e compartilhamento de componentes através da descoberta dinâmica. Orquestração de serviço permite que serviços sejam colocados juntos de várias maneiras na formação de aplicações complexas. Cada seleção de serviço possível traz diferentes níveis de QoS. Assim, há a necessidade de criar mecanismos rápidos e eficientes que possam ser usados para análise de desempenho de WS entre um conjunto de prestadores de serviços. Esta tese apresentou um mecanismo eficiente que, em todos os testes realizados, possui tempos de resposta comparáveis aos tempos de resposta reais de WS (menos de 10% o pior caso) depois de ser comparada com o tempo do modelo gerado pela arquitetura "SOASPE".

7.1 Contribuições do Estudo

Neste trabalho de tese abordou-se a questão relativa à avaliação de desempenho de WS baseada na execução de processos orquestrados onde exista a inclusão explícita das tarefas humanas em processos BPEL (extensão para pessoas, chamada de BPEL4People) e sendo estes processos transformados, seguindo as regras da arquitetura "SOASPE", em duas extensões de redes de Petri as GSPN e as CPN. Para a concretização deste trabalho de tese foi especificada uma arquitetura "SOASPE" para transformação de códigos BPEL4People de orquestração de WS em redes GSPN e CPN. Com as redes GSPN e CPN de códigos BPEL4People é possível fazer a avaliação de desempenho de WS orquestrados através de códigos BPEL (extensão para pessoas - BPEL4People). Dentro da arquitetura "SOASPE" foram especificadas regras e estruturas para conversão de códigos: BPEL4People

(Bpel e WS-HumanTask) em GSPN e BPEL4People (Bpel e WS-HumanTask) em CPN. Foi desenvolvido um módulo de captura de tempo de resposta de serviços automatizados e serviços com a participação humana e também foram desenvolvidos algoritmos para atribuição de tempo de atraso das transições, através de técnicas de análise de funções de distribuição de probabilidade. Para a automatização do processo de transformação de códigos BPEL4People em redes GSPN, seguindo as regras de transformação da arquitetura "SOASPE", foi desenvolvida uma ferramenta denominada de "TPeople4PN".

7.2 Limitações e proposições

Note-se que os <*fault handlers*> não são considerados para efeitos do cálculo de desempenho, pois as exceções não fazem parte do comportamento normal da execução dos *web services*. <*Compensation handlers*> e atividades <*Compensate*> são também ignoradas, porque elas só podem ser ativadas a partir de falhas ou de outras <*Compensation handlers*>.

Na atual versão da ferramenta de transformação "TPeople4PN" somente foi implementada a transformação automática de códigos BPEL e de sua extensão BPEl4People em GSPN. As regras de transformação destes códigos em CPN somente estão descritas teoricamente nas Seções 4.3.2 e 5.3.

7.3 Considerações finais

A modelagem em rede de Petri (GSPN) mostrou-se um pouco mais precisa do que a modelagem em rede de Petri (CPN), pois as redes de Petri (GSPN) modelam melhor o comportamento estocástico do tempo de resposta de cada um dos serviços e dos tempos de resposta de cada um dos usuários nos provedores de serviços, onde exista a participação humana.

7.4 Trabalhos Futuros

Como trabalho futuro, sugere-se a continução das pesquisas relativas a este trabalho de tese, particularmente refinamentos mais profundos com vista à modelagem de outros aspectos que não foram incluídos nesta tese, tais como a extensão deste trabalho para suportar serviços em Grid, tecnologia recente que adiciona novos recursos para o projeto de *web services*.

Como já mencionado nesta tese, os <*fault handlers*> não foram considerados para efeitos do cálculo de desempenho, pois as exceções não fazem parte do comportamento normal da execução dos *web services*. <*Compensation handlers*> e atividades <*Compensate*> foram também ignoradas, porque elas só podem ser ativadas a partir de falhas ou de outras <*Compensation handlers*>. Fica como sugestão de trabalho futuro a modelagem destas claúsulas para análise de qualidade dos WS.

Pretende-se também expandir a ferramenta "TPeople4PN" para contemplar o processo de geração automática de BPEL4People em CPN.

Referências Bibliográficas

AALST, W. M. P. van der. The application of Petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, n. 1, p. 21–66, 1998.

AALST, W. van der; HEE, K. van. *Workflow Management: Models, Methods, and Systems (Cooperative Information Systems)*. [S.l.]: The MIT Press, 2004. Paperback. ISBN 0262720469.

ADOBE et al. *Web Services Human Task (WS-HumanTask), Version 1.0*. June 2007. "IBM: `http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf`.

ARIAS-FISTEUS, J.; FERNÁNDEZ, L. S.; KLOOS, C. D. Formal verification of bpel4ws business collaborations. In: *EC-Web*. [S.l.: s.n.], 2004. p. 76–85.

ARSANJANI, A. Service-oriented modeling and architecture: How to identify, specify, and realize services for your soa. IBM developerWorks, November 2004.

BERA, P. D. P.; GHOSH, S. K. Formal analysis of security policy implementations in enterprise networks. *International Journal of Computer Networks and Communications*, v. 1, n. 2, p. 117–134, july 2009. ISSN 0974 - 9322.

BERNARDI, S.; DONATELLI, S. Stochastic petri nets and inheritance for dependability modelling. In: *PRDC '04: Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04)*. [S.l.: s.n.], 2004. p. 363–372.

BERTINO, E. Access control and authorization constraints for ws-bpel. In: *In ICWS*. [S.l.]: IEEE Computer Society, 2006. p. 275–284.

BIANCULLI, D.; GHEZZI, C.; SPOLETINI, P. A model checking approach to verify bpel4ws workflows. In: *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*. Washington, DC, USA: IEEE Computer Society, 2007. p. 13–20. ISBN 0-7695-2861-9.

BILLINGTON, J.; DIAZ, M.; ROZENBERG, G. (Ed.). *Application of Petri Nets to Communication Networks, Advances in Petri Nets*. London, UK: Springer-Verlag, 1999. ISBN 3-540-65870-X.

BOTTO, R. *Arquitetura Corporativa de Tecnologia da Informação*. Rio de Janeiro: Brasport, 2004.

BRINKSMA, E.; HERMANNS, H.; KATOEN, J.-P. (Ed.). *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*. New York, NY, USA: Springer-Verlag New York, Inc., 2002. ISBN 3-540-42479-2.

CARDOSO, J.; VALETTE, R. *Redes de Petri*. [S.l.]: Ed. da UFSC, 1997.

CHANDRASEKARAN, S. et al. Performance Analysis and Simulation of Composite Web Services. *Electronic Markets*, v. 13, n. 2, 2003.

CHEN, M.; CHEN, A. N. K.; SHAO, B. B. M. The implications and impacts of web services to electronic commerce research and practices. *J. Electron. Commerce Res.*, v. 4, n. 4, p. 128–139, 2003.

CHIOLA, G. et al. Generalized stochastic petri nets: A definition at the net level and its implications. *IEEE Transactions on Software Engineering*, v. 19, n. 2, p. 89–107, 1993.

CURBERA, F. et al. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, IEEE, v. 6, n. April, p. 86–93, 2002.

DUN, H.; XU, H.; WANG, L. Transformation of bpel processes to petri nets. In: *Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2008. p. 166–173. ISBN 978-0-7695-3249-3.

DYER, J. M. S.; ZULAUF, J. Motion capture white paper. December 2005. Disponível em: <http://reality.sgi.com/employees/jam$_s$b/*mocap*/$MoCapWP_v$2.0.*html*>.

ERL, T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005. ISBN 0131858580.

FARREL, C. F. J. What are web services? *Communications of the ACM*, v. 46, n. 6, p. 31, 2003.

FU, X.; BULTAN, T.; SU, J. Analysis of interacting bpel web services. In: . [S.l.]: ACM Press, 2004. p. 621–630.

GARCIA, D. Z. G. a.; TOLEDO, M. B. F. de. Semantics-enriched qos policies for web service interactions. In: *Proceedings of the 12th Brazilian Symposium on Multimedia and the web*. New York, NY, USA: ACM, 2006. (WebMedia '06), p. 35–44. ISBN 85-7669-100-0.

GÓMEZ-MARTÍNEZ, E.; MERSEGUER, J. Impact of soap implementations in the performance of a web service-based application. In: *ISPA Workshops*. [S.l.: s.n.], 2006. p. 884–896.

HEUSER, C. A. *Modelagem Conceitual de Sistemas: Redes de Petri*. [S.l.]: Material de apoio utilizado no curso de pós-graduação em Ciência da Computação da UFRGS., 1990.

HOLANDA G.C. BARROSO, A. S. H. Performance analysis of service oriented software. *iSys - Revista Brasileira de Sistemas de Informação*, v. 2, 2009. ISSN 1984-2902.

HOLANDA G.C. BARROSO, A. S. H. Soaspe: a framework for the performance analysis of service oriented software. *Simpósio Brasileiro de Sistemas de Informação*, p. 204–215, maio 2009.

HOLANDA G.C. BARROSO, A. S. H. Performance evaluation of bpel4people specifications integrate human interactions into business process. *Simpósio Brasileiro de Sistemas de Informação*, junho 2010.

HOLANDA G.C. BARROSO, A. S.-J. H. H. Performance analysis of web services orchestrated with ws-bpel4people. *International Journal of Computer Networks and Communications*, v. 2, n. 6, p. 117–134, november 2010.

HOLANDA, H. J. A.; BARROSO, G. C.; SERRA, A. Model method for the transformation of bpel4people into generalized stochastic petri nets. *Computing*

in the Global Information Technology, International Multi-Conference on, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 146–151, 2010.

HOLMES, T.; VASKO, M.; DUSTDAR, S. Viebop: Extending bpel engines with bpel4people. In: *16TH EUROMICRO INTERNATIONAL CONFERENCE ON PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING*. [S.l.]: IEEE Computer Society, 2008. p. 547–555.

HUBER, P.; JENSEN, K.; SHAPIRO, R. M. Hierarchies in coloured petri nets. In: *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets 1990*. London, UK: Springer-Verlag, 1991. p. 313–341. ISBN 3-540-53863-1. Disponível em: <http://portal.acm.org/citation.cfm?id=647736.735468>.

JENSEN, K.; KRISTENSEN, L.; WELLS, L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, v. 9, n. 3, p. 213–254, June 2007.

KELTON, W. D.; SADOWSKI, R. P.; SADOWSKI, D. A. *Simulation with Arena*. 2. ed. New York, NY, USA: McGraw-Hill, Inc., 2002. ISBN 0071122397.

KOSHKINA, M. et al. *Verification of business processes for Web services*. [S.l.], 2003.

LAW, A. M.; KELTON, D. W. *Simulation Modelling and Analysis*. [S.l.]: McGraw-Hill Education - Europe, 2000. Paperback. ISBN 0071165371.

LUDWIG, H. Web services qos: External slas and internal policies - or: How do we deliver what we promise? In: . [S.l.: s.n.], 2003.

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *SIGMETRICS Perform. Eval. Rev.*, ACM, New York, NY, USA, v. 26, p. 2–, August 1998. ISSN 0163-5999.

MARTENS, A. *Distributed Business Processes — Modeling and Verification by help of Web Services*. [S.l.]: Humboldt-Universität zu Berlin, 2003. PhD Thesis.

MENASCé, D. A.; ALMEIDA, V. F. *Capacity Planning for Web Services: metrics, models, and methods*. [S.l.]: Prentice Hall., 2001. 608P p.

MENASCE, D. A. Composing web services: A qos view. *IEEE Internet Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 8, p. 88–90, November 2004. ISSN 1089-7801.

MOLLOY, M. *On the Integration of Delay and Throughput Measures In Distributed Processing Models*. [S.l.]: UCLA, 1981. Ph.D.Thesis.

MURATA, T. Petri nets: Properties, analysis and applications. In: *Proceedings of the IEEE*. [S.l.: s.n.], 1989. p. 541–580. Proceedings of the IEEE, volume 77, number 4.

NARAYANAN, S.; MCILRAITH, S. Analysis and simulation of web services. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 42, p. 675–693, August 2003. ISSN 1389-1286.

PENHA, D. O.; FREITAS, H. C.; MARTINS, C. A. Modelagem de sistemas computacionais usando redes de petri: aplicação em projeto, análise e avaliação. IV Escola Regional de Informática RJ/ES, Novembro 2004.

PETRI, C. A. *Kommunikation mit Automaten*. Tese (Doutorado) — Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, 1966, Pages: Suppl. 1.

RAMCHANDANI, C. *Analysis of asynchronous concurrent systems by timed Petri nets*. Cambridge, MA, USA, 1974.

RUD, D.; SCHMIETENDORF, A.; DUMKE, R. Performance modeling of ws-bpel-based web service compositions. In: *Proceedings of the IEEE Services Computing Workshops*. Washington, DC, USA: IEEE Computer Society, 2006. p. 140–147. ISBN 0-7695-2681-0.

SCHULTE, W. R.; NATIS, Y. V. N. *Organization for the Advancement of Structured Information Standards*. maio 2006. OASIS Committee Draft.

SELVAGE D. WOLFSON, J. H.-B. M. *Information management in Service-Oriented Architecture, Part 1: Discover the role of information management in SOA*. [S.l.], 2005.

SILVA, A. N.; LINS, F. A. Avaliação de desempenho da composição de web services usando redes de petri. In: *In: SBRC, 24o. Simpósio Brasileiro de Redes de Computadores*. Curitiba: SBC, 2006.

SMITH, C. U. Performance engineering of software systems. In: *Computer Measurement Group Conference*. [S.l.: s.n.], 1989. p. 1354–1355.

VERBEEK, H. M. W. Analyzing bpel processes using petri nets. In: *Florida International University*. [S.l.: s.n.], 2005. p. 59–78.

WESTERGAARD, M.; LASSEN, K. The britney suite animation tool. In: DONATELLI, S.; THIAGARAJAN, P. (Ed.). *Petri Nets and Other Models of Concurrency - ICATPN 2006*. [S.l.]: Springer Berlin / Heidelberg, 2006, (Lecture Notes in Computer Science, v. 4024). p. 431–440. 10.1007/11767589-26.

YU, T.; LIN, K.-J. Qcws: an implementation of qos-capable multimedia web services. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 30, p. 165–187, August 2006. ISSN 1380-7501.

YUAN, C.-Y. et al. A three-layer model for business processes: process logic, case semantics and workflow management. *J. Comput. Sci. Technol.*, Institute of Computing Technology, Beijing, China, v. 22, p. 410–425, May 2007. ISSN 1000-9000.

ZHAO, A. C. X.; KRISHNAN, P. Verifying bpel work?ows under authorisation constraints. In: *Business Process Management Lecture Notes in Computer Science*. [S.l.]: Springer-Verlag, 2006. v. 4102, p. 439–444.