Avaliação de Desempenho da Composição de *Web Services* Usando Redes de Petri

Arnoldo N. da Silva[1], Fernando A. A. Lins[1], José C. S. Júnior[1], Nelson S. Rosa[1], Nívia C. Quental[2], Paulo R. M. Maciel[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

[2]Departamento de Sistemas Computacionais – Universidade de Pernambuco (UPE)
Recife – PE – Brasil

{ans2,faal2,jcsj,nsr,ncq,prmm}@cin.ufpe.br

Abstract. *Web Services have played an important role in the development of distributed systems. In particular, the possibility of composing already implemented Web Services in order to provide a new functionality is an interesting approach for building distributed systems. However, choosing the better composition still a challenger as different qualities may be observed in the composition, such as security, performance, fault tolerance, and so on. In this context, this paper proposes a methodology based on Stochastic Petri Nets to model, evaluate and help to choose Web Service compositions considering performance aspects. In order to illustrate the proposed methodology, we applied it to a classical case study of credit approving.*

Resumo. *Web Services tem desempenhado um importante papel no desenvolvimento de sistemas distribuídos. Particularmente, a possibilidade de se compor Web Services já implementados a fim de prover uma nova funcionalidade é uma abordagem interessante para construção de sistemas distribuídos. Contudo, escolher a melhor composição continua sendo um desafio à medida que diferentes qualidades podem ser observadas na composição, como segurança, desempenho, tolerância a falhas, e assim por diante. Neste contexto, este artigo propõe uma metodologia baseada em Redes de Petri Estocásticas para modelar, avaliar e apoiar a escolha das composições de Web Service considerando aspectos de desempenho. Com o intuito de ilustrar a metodologia proposta, ela foi aplicada em um estudo de caso clássico de aprovação de crédito.*

1. Introdução

A composição de *Web Services* tem surgido como uma importante estratégia para permitir a colaboração de aplicações entre empresas (*Business-to-Business*) [Benatallah 03, Zeng 04]. Além disto, a idéia de composição permite que aplicações complexas possam se construídas combinando-se serviços mais básicos. Em termos práticos, esta estratégia acelera o desenvolvimento de aplicações, permite um alto grau de reusabilidade de serviços, e facilita o desenvolvimento de sistemas complexos [Milanovic 04].

Neste contexto, é importante observar que não há uma padronização da composição de *Web Services* e nem existem propriedades já estabelecidas que a composição deva satisfazer [Milanovic 04]. Na prática, ainda não há regras padronizadas que definam as possíveis formas de combinações de *Web Services* (*e.g.*, paralelo, seqüencial, desabilitação). Ao mesmo tempo, poucas abordagens permitem algum tipo de verificação de propriedades da composição (*e.g.*, corretude, segurança e desempenho). Por exemplo, dado um conjunto de *Web Services* que podem ser combinados para realizar uma determinada funcionalidade, qual ou quais das possíveis composições é a mais segura ou a que possui um melhor desempenho é ainda uma questão não trivial a ser resolvida.

Diversas abordagens têm sido propostas para tratar os mais variados aspectos da composição de *Web Services*, tais como: linguagens de composição, formalização da composição, ambientes de suporte à composição. Entre as linguagens para composição de *Web Services*, BPEL4WS [Andrews *et al* 2003] tem se destacado pelo fato de ser uma linguagem baseada em XML e especificamente projetada para suportar a composição de serviços. Em termos de formalização, o objetivo básico é o uso de técnicas de descrição formal (rede de Petri [Hamadi *et al* 2003, Narayanan 02], álgebra de processos [Ferrara 04, Cardeli 99, Radetzki 04], lógica construtiva [Lammermann 01]) para a verificação de propriedades da composição. Por fim, os ambientes mencionados consistem de sistemas de *middleware* especificamente projetados para suportar a composição de *Web Services* [Benatallah 03, Zeng 04, Bottcher 05]. Comum a todas estas abordagens está ausência de tratamento de aspectos de desempenho da composição. Em particular, mesmo naquelas abordagens que usam redes de Petri para formalização da composição, o ponto central é a modelagem, sem preocupação com o desempenho.

Este trabalho tem como objetivo propor uma metodologia baseada em redes de Petri estocásticas [Marsan 95] que possa ser utilizada para verificação de propriedades de desempenho de composições de *Web Services*. Na prática, a metodologia consiste de um conjunto de passos que quando aplicados geram modelos em redes de Petri que permitem a simulação e obtenção de resultados sobre métricas de desempenho da composição de *Web Services*.

Este artigo está organizado da seguinte forma: a Seção 2 introduz os principais conceitos básicos ligados à composição de *Web Services* e Redes de Petri; na Seção 3 é apresentada a metodologia proposta; a aplicação da metodologia a um estudo de caso é mostrada na Seção 4; finalmente, na Seção 5 são apresentados as conclusões e os trabalhos futuros.

2. Conceitos Básicos

Antes de apresentarmos a metodologia proposta, esta seção introduz os conceitos básicos relacionados à composição de serviços e redes de Petri estocásticas.

2.1 Composição de *Web Services*

Dada a ploriferação atual dos *Web Services*, o paradigma da composição de serviços surge como uma possibilidade de abordagem de implementação de aplicações, onde a funcionalidade final das mesmas é obtida através da combinação de diversos serviços.

Como exemplo de aplicação, pode-se imaginar um tradutor de línguas capaz de traduzir textos do Português para o Árabe. Se não houver disponível um tradutor direto para as duas línguas, mas sim um tradutor de Português para Inglês e outro de Inglês para Árabe, cada um deles implementado em um *Web Service*, o serviço pode ser criado através da composição dos outros dois pré-existentes.

Os serviços podem ser compostos de maneira estática ou dinâmica [Benatallah *et al* 2002]. Esta escolha depende do tipo do processo que está sendo composto. Se os serviços que serão compostos têm uma natureza fixa, ou mudam raramente as suas interfaces e semântica, a composição estática satisfaz as necessidades. Contudo, um processo pode conter um conjunto de funções fracamente definidas a ser realizado, ou então ele tem que ter a possibilidade de se adaptar dinamicamente a mudanças não previstas no ambiente. Para estes casos, a composição estática pode não ser a abordagem mais adequada, pois alterações em um sistema baseado em composição estática requerem a interrupção da continuidade do processo. Composição dinâmica é uma alternativa importante neste caso, dado que existem processos críticos que não podem sofrer interrupções.

Além da natureza estática e dinâmica da composição, os *Web Services* podem ser compostos de diversas formas [Hamadi *et al* 2003]:

- Composição em seqüência: executa, de forma serial, um serviço (S1) seguido de outro (S2). A principal característica deste tipo de composição é que, para se iniciar o serviço S2, S1 deverá terminar a sua execução;
- Composição em paralelo: executa, simultaneamente e independentemente, dois serviços arbitrários (S1 e S2). Como variação, este tipo de composição pode se dar de forma com ou sem comunicação entre os serviços envolvidos;
- Composição com escolha: serviços compostos desta forma resultam na execução de apenas um dos mesmos, descartando a execução do restante. A escolha é feita de maneira aleatória;
- Composição seletiva: similar a composição com escolham, diferindo na forma da escolha. Aqui, a mesma é realizada baseada em critérios e informações, e não de maneira aleatória;
- Composição iterativa: neste caso, um mesmo serviço S1 é invocado diversas e consecutivas vezes.

Estas diferentes formas de composição podem ser incorporadas a linguagens de composição através da definição de operadores ou combinadores [Cardelli 99].

2.2 Redes de Petri e extensões Estocásticas

Modelagem analítica é uma abordagem de avaliação de desempenho que procura modelar um sistema como um processo estocástico, onde variáveis aleatórias são indexadas no tempo. Os modelos mais comumente usados são os modelos de espaço de estados, como as cadeias de Markov [Markov 1971]. Entretanto, quando problemas do mundo real são estudados, a análise em cadeias de Markov torna-se mais complexa, devido à tendência da geração de uma grande quantidade de estados. É importante especificar tais sistemas por caminhos compactos que reduzam o risco de erros e a

dificuldade na criação dos modelos. É de grande importância, portanto, separar a descrição dos modelos de alto nível dos modelos computacionais de baixo nível, procurando tornar sua geração automática.

Redes de Petri constituem um formalismo matemático que permite representação gráfica, e possui métodos poderosos de análise formal, bem como de desempenho (utilizando suas extensões temporizadas) de um sistema. Os lugares e transições das Redes de Petri são usados para modelar uma visão lógica de pontos dos sistemas. A idéia de associar variáveis aleatórias de atraso exponencialmente distribuídas às transições foi explorada pela primeira vez por Natkin [Natkin 1980] e Molloy [Molloy 1981], permitindo o surgimento das Redes de Petri Estocásticas (*Stochastic Petri Nets* - SPNs) e suas extensões. As SPNs possuem a propriedade de serem isomorfas às Cadeias de Markov de Tempo Contínuo, permitindo que as mesmas análises estacionárias e transientes já conhecidas destas possam podem ser aplicadas àquelas redes. A extensão GSPN (*Generalized Stochastic Petri Nets*), originalmente proposta por Marsan [Marsan 1995], é uma das extensões mais conhecidas.

Uma GSPN é definida por uma 8-upla GSPN=(P,T,I,O,H,Π,W,M0) [Marsan, 1995]. O conjunto P de lugares representa a disponibilidade de recursos (marcas na rede) do sistema, estados locais e variáveis de sistema. O conjunto T de transições representa o conjunto de ações que podem provocar mudança de estado. Esse conjunto é dividido no subconjunto de transições temporizadas e no subconjunto de transições imediatas que descrevem ações instantâneas, possuindo maior prioridade de disparo que a anterior. A função W associa um número real não negativo que denote a taxa da distribuição exponencial a cada transição temporizada; para cada transição imediata atribui-se um número natural para determinar o peso que será utilizado no cálculo de probabilidade de seu disparo. A função Π define o nível de prioridade de cada transição imediata (a prioridade de uma transição temporizada é zero). Temos I e O como as funções de mapeamento de lugares a transições e transições a lugares, respectivamente, enquanto a função H representa os arcos inibidores. A marcação M0 representa o estado inicial do sistema. As GSPNs têm sido amplamente utilizadas na modelagem de diversos tipos de sistemas, desde sistemas de manufatura a redes sem fio [Heindl 2001].

3. Metodologia Proposta

Nesta seção será abordado como redes de Petri estocásticas podem ser utilizadas tendo em vista a avaliação de desempenho da composição de *Web Services*. A metodologia proposta consiste de quatro passos: o primeiro consiste na construção da topologia da rede de Petri (condizente com a possibilidade de composição a ser modelada); o segundo passo visa fazer um tratamento dos dados de entrada que serão utilizados no modelo; o terceiro trata do aspecto de modelar o fato de os *Web Services* estarem ou não sendo executados no mesmo servidor e, por último, o quarto passo destina-se a orientar o processo de simulação do modelo desenvolvido nas etapas anteriores.

Passo I. Definição da topologia da Rede de Petri

De início, o analista possui uma visão abstrata das possibilidades de composição da sua aplicação. Neste primeiro passo, este modelo abstrato será usado como entrada. Como saída, espera-se um modelo em redes de Petri simples, que modele a(s)

possibilidade(s) de composição desejadas. Por definição, a composição de vários serviços se constitui em um outro serviço (ver Figura 1).

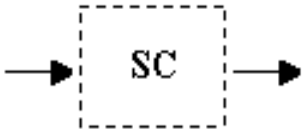

Figura 1: Notação genérica de um serviço composto

Para a definição da rede de Petri estocástica, é prudente transformar o modelo em um modelo estacionário [Marsan 1995] para facilitar a análise do modelo final na ferramenta de redes de Petri estocástica. Para isso, usamos uma iteração na composição, conforme mostrado na Figura 2(a).

A partir desta transformação, pode-se pensar na construção da topologia da composição propriamente dita. Na Subseção 2.1 foram mostradas diversas possibilidades de composição. A partir delas, foram gerados modelos de rede de Petri para ilustrar a composição de *Web Services*. Na Figura 2(b) está modelada uma composição em seqüência, onde o serviço S2 só irá disparar após a execução S1 e da transição imediata que os separa. Na figura 2(c) está modelada a composição em paralelo, enquanto na figura 2(d) encontra-se o modelo correspondente à composição com escolha.

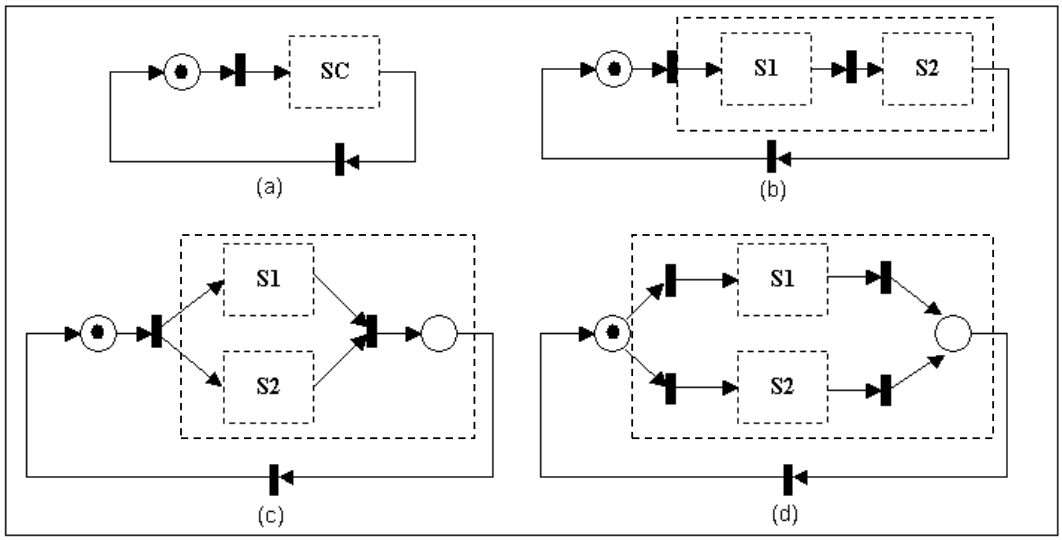

Figura 2 : Composição de Serviços e seus Modelos de Rede de Petri

Passo II. Modelagem dos Dados de Entrada

Com a topologia da rede de Petri construída, faz-se necessário alimentá-la com dados que representem os atributos que caracterizam o sistema modelado. No caso específico deste trabalho, onde os objetivos concentram-se na avaliação da composição de *Web Services*, torna-se imprescindível a caracterização dos tempos individuais de

cada um deles. Neste passo, é ilustrado como modelar o comportamento estocástico de tais elementos através do uso de distribuições de probabilidade [Desrochers 1994].

Como entradas deste passo, serão utilizadas as médias e os desvios padrões dos tempos dos *Web Services*, enquanto na saída são esperados os valores de μ_1, μ_2 e γ, elementos que serão detalhados mais tarde e irão alimentar o modelo. Os dados coletados constituirão uma amostra com distribuição desconhecida de média μ_D e desvio padrão σ_D. Deve-se calcular o coeficiente de variação (CV) :

$$CV = \frac{\sigma_D}{\mu_D} \quad (1)$$

Uma amostra empírica de distribuição desconhecida deverá ser representada em modelo de Redes de Petri. A Figura 3 expõe a representação abstrata deste modelo.

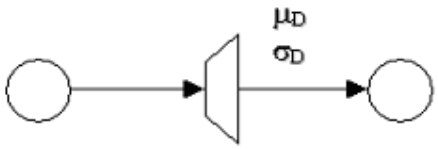

Figura 3: Abstração do modelo de Rede de Petri com transição de distribuição desconhecida

Dependendo do valor de CV, esses dados serão aproximados a uma das distribuições: Erlang, Hiperexponencial ou Hipoexponencial. Isto torna possível representar a questão probabilística envolvida no problema no modelo da Rede de Petri, uma vez que estas distribuições derivam de uma associação de transições exponenciais.

Caso o coeficiente de variação seja maior do que 1 (CV > 1) e o mesmo seja um valor inteiro, a amostra empírica deverá ser aproximada à distribuição Erlang, que é representada por uma seqüência de γ transições exponenciais, cuja taxa será λ. O modelo em redes de Petri para a aproximação de uma distribuição Erlang está representado na Figura 4.

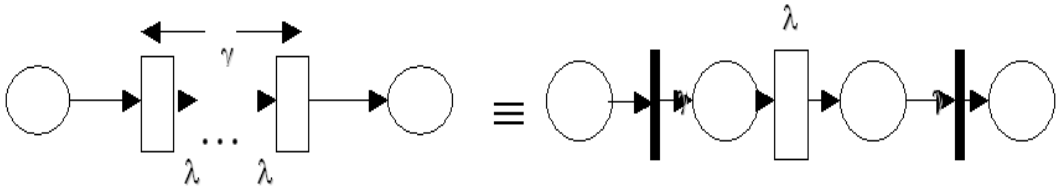

Figura 4: Modelo em Redes de Petri para aproximação de distribuição Erlang

Os parâmetros λ e γ são calculados de acordo com as equações (2) e (3):

$$\gamma = \left(\frac{\mu_D}{\sigma_D} \right)^2 \quad (2)$$

$$\lambda = \frac{\gamma}{\mu_D} \quad (3)$$

Caso CV>1 (sendo CV um número inteiro ou não), a distribuição deverá ser aproximada à Hiperexponecial, cujos parâmetros são: λ_h - Taxa da transição exponencial, e r_1 e r_2, pesos das transições imediatas. Tais parâmetros são calculados através das equações (4), (5) e (6):

$$\lambda_h = \frac{2\mu_D}{(\mu_D^2 + \sigma_D^2)} \tag{4}$$

$$r_1 = \frac{2\mu_D^2}{(\mu_D^2 + \sigma_D^2)} \tag{5}$$

$$r_2 = 1 - r_1 \tag{6}$$

O modelo de redes de Petri para esta aproximação é apresentado na Figura 5.

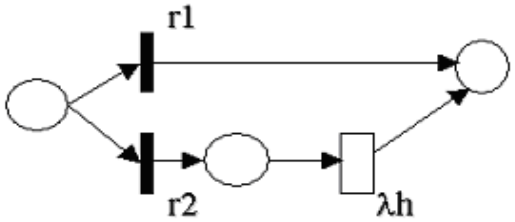

Figura 5: Modelo de rede de Petri para representar a distribuição Hiperexponencial

Para o caso de CV < 1, a distribuição deverá ser aproximada à distribuição Hipoexponencial, composta de uma exponencial com taxa λ_1 e uma Erlang formada por γ exponenciais de taxas λ_2. Os parâmetros λ_1, λ_2 e γ são calculados a partir das equações seguintes:

$$\left(\frac{\mu_D}{\sigma_D}\right)^2 - 1 \le \gamma < \left(\frac{\mu_D}{\sigma_D}\right)^2 \tag{7}$$

$$\lambda_1 = \frac{1}{\mu_1}, \mu_1 = \frac{\mu_D \pm \sqrt{\gamma(\gamma+1)\sigma_D^2 - \gamma\mu_D^2}}{(\gamma+1)} \tag{8}$$

$$\lambda_2 = \frac{1}{\mu_2}, \mu_2 = \frac{\gamma\mu_D \mp \sqrt{\gamma(\gamma+1)\sigma_D^2 - \gamma\mu_D^2}}{(\gamma+1)} \tag{9}$$

Em (8) e (9), μ_1 e μ_2 são, respectivamente, os tempos (*delays*) que devem ser associados às transições da rede de Petri, que está apresentada na Figura 6. Esta figura ilustra a representação do modelo empírico em uma aproximação para um modelo Hipoexponencial.

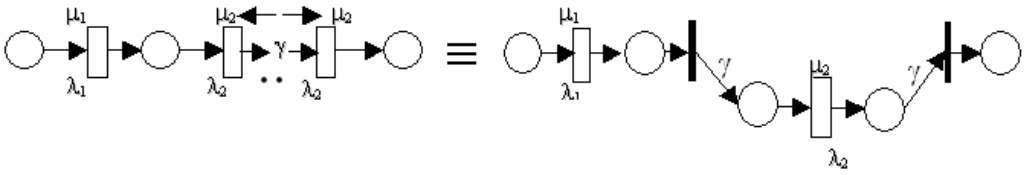

Figura 6. Modelo de rede de Petri para representar distribuição Hipoexponencial

Identificadas as distribuições mais adequadas para caracterizar os *Web Services*, é possível traduzi-las para a rede de Petri Estocástica equivalente. Com este passo, poderão ser substituídos os serviços S1 e S2, da Figura 2, pela rede de Petri encontrada.

Passo III. Recursos

Em se tratando de composição, *Web Services* podem estar localizados em um mesmo servidor ou estar em servidores diferentes (BPEL4WS oferece esta flexibilidade). Em outras palavras, eles podem ou não estar compartilhando recursos.

Como entrada deste passo, será utilizado o modelo gerado no Passo I. Como saída, é esperado este mesmo modelo contemplando o aspecto relacionado a recursos.

Para um ambiente onde há compartilhamento, o modelo de Redes de Petri terá um lugar representando este recurso, onde a presença de uma *marca* simboliza a sua disponibilidade. Para o caso em que os *Web Services* disporem de recursos próprios representados no modelo, um lugar com uma *marca* indicará que o *Web Service* equivalente estará em execução.

IV. Refinamento e Execução do Modelo

Até este ponto foram apresentados passos que possibilitam ao projetista chegar a um modelo da composição dos serviços de forma fácil e prática. No entanto, características adicionais podem ser necessárias ao modelo, dependendo da aplicação e dos objetivos definidos no projeto. Este passo foi denominado de refinamento, cuja realização pode não ser necessária.

Existem diversas ferramentas disponíveis de Redes de Petri.A ferramenta aqui adotada é o TimeNet [Zimmerman 2000]. Esta ferramenta trabalha tanto com taxas (λ) como com tempos (μ). Tendo em vista o foco deste trabalho, recomenda-se a utilização da variável tempo (μ). Adicionalmente, a depender do modelo, pode ser utilizada tanto a análise estacionária como a análise transiente. A diferença fundamental se baseia em que, na primeira, todas as possibilidades são alcançadas para se chegar ao resultado final. Caso o modelo desenvolvido não alcance todas as possibilidades, esta análise não pode ser utilizada, pois ele não terminará sua execução. Neste caso, deverá ser utilizada a análise transiente.

Este passo toma como entrada os artefatos gerados nos passos anteriores e gera como saída resultados obtidos através da execução da simulação.

4. Estudo de Caso: Sistema de Aprovação de Empréstimos

Com a metodologia proposta e definida, esta seção apresenta um estudo de caso prático com o objetivo de verificar a usabilidade e a validade da mesma.

4.1 Descrição da aplicação

Uma determinada empresa está interessada em avaliar se seus clientes devem ou não possuir o direito de contrair empréstimos. Desta forma, foi implementado, baseado na noção de orientação a serviço [Pallos 2001], um sistema para realizar esta análise. A aplicação escolhida é o "Aprovador de Empréstimo". Esta aplicação é composta basicamente por dois *Web Services*: "aprovador" e "assessor" (ver Figura 7).

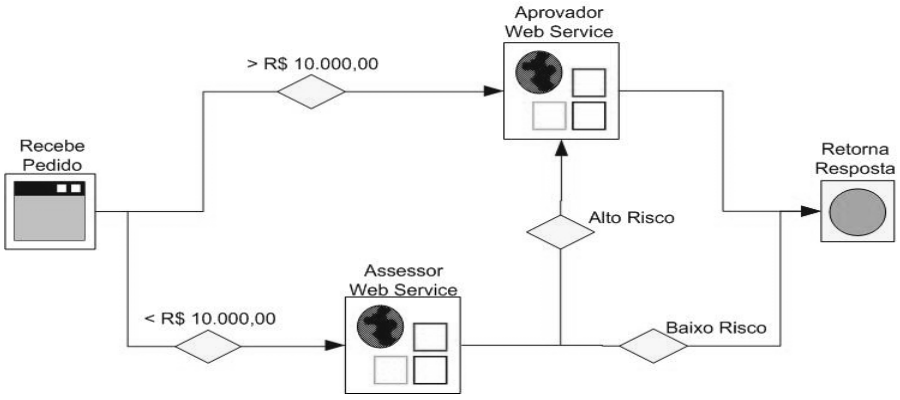

Figura 7: Semântica da aplicação

A depender do valor pedido pelo cliente e da resposta do serviço assessor, será definido se a intervenção do serviço aprovador será necessária. No caso da necessidade da intervenção do serviço assessor e do serviço aprovador, uma composição em seqüência será realizada. Uma alternativa que poderia ser avaliada pela loja seria a de re-implementar esta aplicação de tal forma que os dois serviços sempre fossem executados, e de forma paralela, reduzindo-se assim o tempo total de execução caso fosse necessária à intervenção do aprovador (se não fosse, a resposta do mesmo seria descartada).

4.2 Ferramental utilizado e Medições

Tendo em vista a execução da aplicação, vários artefatos de software foram utilizados. Inicialmente, a aplicação foi executada em uma máquina AMD Sempron 2600+, com placa mãe *on-board* e com 512MB de memória RAM. O sistema operacional instalado é o *Windows XP Professional*.

Em termos de software, os *Web Services* (escritos em Java) foram disponibilizados em um servidor *Tomcat*, versão 5.0.28. Adicionalmente, foram utilizados o *Ant* (versão 1.6), o módulo SOAP [Hendricks 2002] para o Apache (versão 2.3.1) e o *Axis* (Versão 1.4). Todos estes softwares são necessários para a execução de *Web Services*, e informações adicionais sobre a necessidade e a forma de utilização dos

mesmos podem ser encontradas em [Hendricks 2002]. Para possibilitar a composição, uma *engine* de BPEL deve ser utilizada. Pela razão de ser *Open-Source*, a *engine ActiveBPEL* [ActiveBPEL 2005], versão 1.2, foi adotada.

Foram medidos os tempos de execução de toda a composição e dos *Web Services* de forma individual. O tempo da composição compreende desde o tempo no qual a primeira invocação é feita até a emissão da resposta, incluindo o tempo de execução dos próprios *Web Services*. Os tempos dos *Web Services* foram colhidos de forma individual, não se considerando o overhead causado pela *engine*. A Tabela 1 apresenta os tempos totais das composições (tanto serial quanto paralela) obtidos através da medição do tempo de execução da composição.

Tabela 1: Média e desvio padrão do tempo de execução das composições em série e paralela

	Média	Desvio Padrão
Tempo total da composição serial (em ms)	199,3690282	75,51175395
Tempo total da composição paralela (em ms)	191,0529125	75,62969962

Foram também obtidos os tempos de execução relativos à execução individual dos *Web Services*. A Tabela 2 apresenta o tempo de execução destes serviços, em milissegundos (ms).

Tabela 2: Média e desvio padrão do tempo de execução de cada *Web Service*

	Média	**Desvio Padrão**
Tempo de execução do *Web Service* aprovador	7,091404112	4,754760842
Tempo de execução do *Web Service* assessor	6,819372061	3,722488371

Independente da forma como é realizada a composição, o tempo de execução individual de cada um não sofreu variações significativas, dado que o tempo de execução individual dos mesmos não tem forte dependência da forma com que é feita a composição.

Ao se analisar os resultados das medições, pode-se inferir que o fator determinante na diferença do tempo total de execução das composições, explicitados na Tabela 1, foi à topologia das composições. Como se pode constatar, a diferença entre os dois valores se aproxima bastante do tempo de execução do *Web Service* aprovador, o que sugere que o tempo total de execução (neste caso) é influenciado diretamente, e quase exclusivamente, pela topologia da composição. Desta forma, apenas os tempos

relevantes para a configuração da topologia serão considerados para a construção do modelo de simulação.

4.3 Modelagem e simulação da aplicação

A partir da média e do desvio padrão calculados com base na amostra coletada de cada *Web Service*, foram obtidos os coeficientes de variação mostrados na Tabela 3, a fim de verificar a qual distribuição a amostra empírica deve ser aproximada. Os cálculos dos mesmos se basearam na metodologia exposta na seção anterior. Os coeficientes de variação calculados possuíam valores maiores do que 1, indicando que as amostras dos *Web Services* deveriam ser aproximadas a distribuições Hipoexponenciais.

Tabela 3. Coeficiente de Variação do tempo de execução de cada *Web Service*

	Coeficiente de Variação (CV)	**Aproximação**
Web Service aprovador	0,67	Hipoexponencial
Web Service assessor	0,54	Hipoexponencial

Para a modelagem dos dados de entrada no modelo, serão necessários os valores relativos a μ_1, μ_2 e γ, apresentados na Seção 3.2. Estes valores deverão ser calculados tendo como base a média e o desvio padrão dos *Web Services* aprovador e assessor, apresentados anteriormente na Tabela 2. A Tabela 4 apresenta os valores obtidos dos mesmos, em milissegundos (ms).

Tabela 4. Dados calculados para a inserção no modelo de redes de Petri

	μ_1	μ_2	γ
Web Service aprovador	4,337812499	2,753592	2,000000
Web Service assessor	2,998373314	3,820999	3,000000

Utilizando os dados da Tabela 4 e a metodologia proposta neste trabalho, os modelos da composição em seqüência e da composição em paralelo foram desenvolvidos. Os mesmos se encontram, respectivamente, nas Figuras 9 e 10. Eles foram baseados na noção de recurso compartilhado, também abordada na metodologia.

O modelo em redes de Petri apresentado na Figura 8 mostra a composição seqüencial dos *Web Services* em um mesmo servidor. O lugar P1 representa a chegada do pedido do cliente e o lugar recurso representa o processador. As transições imediatas *Less10000* e *More10000* possuem, respectivamente, pesos 1 e 0 para que o serviço assessor possa ser executado e, da mesma forma, a transição imediata *high* possui peso 1 para permitir a execução do *Web Service* aprovador. Esta abordagem foi utilizada para

possibilitar a execução dos dois *Web Services*, caracterizando assim a composição em seqüência. Os lugares P8, P9, P12 e P14 controlam o uso do recurso, tomando-o do processador quando do início de uma operação e devolvendo-o ao processador ao fim da operação.

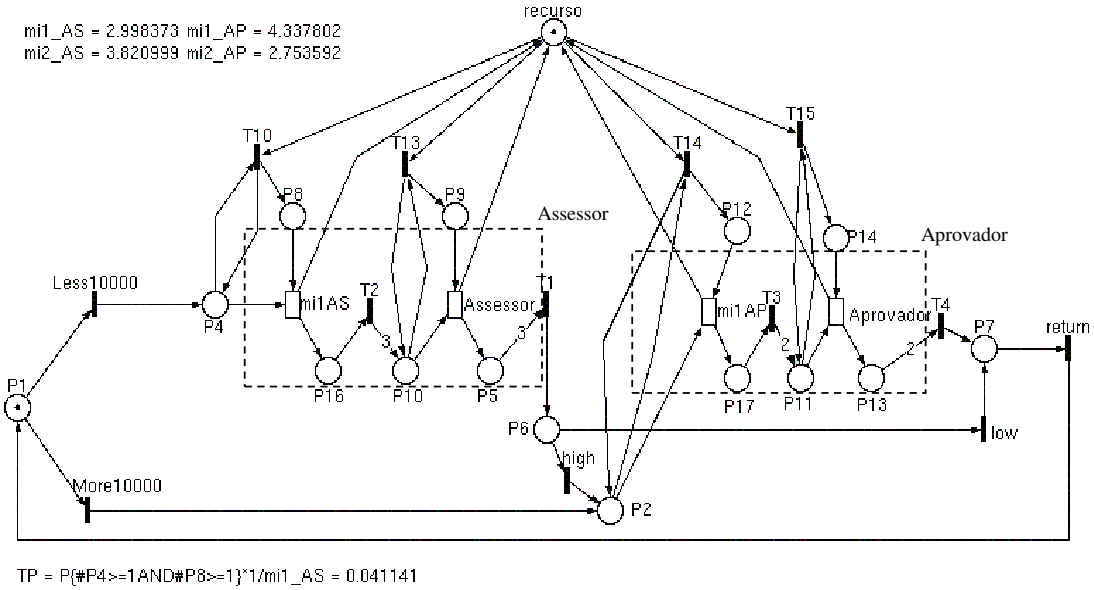

Figura 8: Modelo de Redes de Petri da composição em seqüência

O *throughput* do sistema pode ser obtido via análise estacionária da rede através da fórmula de TP (Figura 8). O valor obtido depende da probabilidade de P4 e P8 possuirem uma marca e da taxa associada à transição *miAS* e é utilizado para inferir o tempo médio total do sistema, uma vez que os lugares P4 e P8 recebem uma marca ao fim de um "ciclo" (fim marcado pela transição *return*).

$$TP = \frac{P(\#P4 \geq 1 AND \#P8 \geq 1).1}{mi1\_AS} \qquad (10)$$

O modelo de composição em paralelo é mostrado na Figura 9. Os *Web Services* assessor e aprovador executam em uma mesma máquina e concorrem com o recurso processador. Os arcos que entram no lugar recurso permitem que o mesmo fique novamente disponível durante a execução dos *Web Services* para a representação da política de *time sharing* no modelo. Da mesma forma que no modelo seqüencial, os lugares P7, P8, P9 e P12 controlam o uso do recurso, P1 representa o cliente que fará a requisição do serviço e o lugar P6 recebe os *tokens* resultantes do término das execuções dos *Web Services*. Assim como no exemplo anterior, o *throughput* TP medido em *miAS* e *miAP*, cujos lugares de entrada são P4 e P3 , respectivamente, é calculado para obter o tempo total do sistema.

$$TP = \frac{P(\#P4 \geq 1 AND \#P7 \geq 1).1}{mi1\_AS} + \frac{P(\#P3 \geq 1 AND \#P9 \geq 1).1}{mi1\_AP} \qquad (11)$$

Vale ressaltar que estes modelos desenvolvidos dizem respeito apenas à forma com que a composição pode ser realizada no problema. Foi visto, nas medições, que a variação dos valores entre a composição em seqüência e a em paralelo se deve fundamentalmente a mudança da topologia da composição.

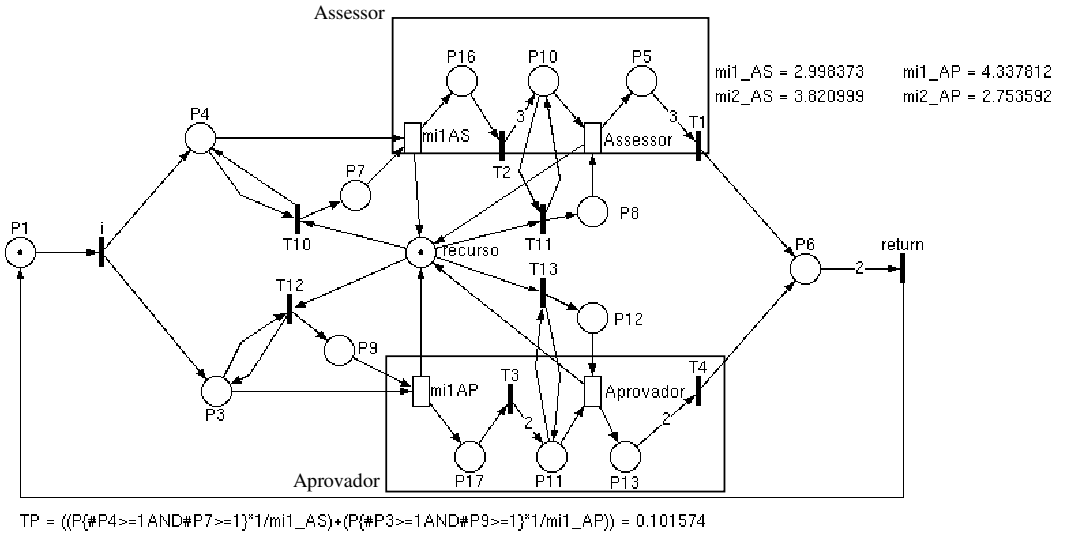

Figura 9: Modelo de Redes de Petri da composição em paralelo

Tabela 5: Tempo de simulação das composições

	Valor Medido (em ms)
Tempo simulado da composição serial	209,7583
Tempo simulado da composição paralela	189,2921

4.4 Avaliação dos resultados obtidos

Conforme abordado anteriormente nas medições, foi verificado que o tempo total de execução da composição, em termos comparativos, apresentava uma forte ligação com o modo de se fazer à composição (topologia). Na verdade, foi visto que outros fatores como chamadas SOAP e *engine* de BPEL não influenciavam diretamente no resultado da comparação das possibilidades de composição. Por exemplo, em comparação com os modelos em seqüência e paralelo, o número de invocações (primitiva *invoke* do BPEL) continua o mesmo, se existiam dois serviços em seqüência, no modelo paralelo continuarão existindo dois serviços, e será gasto aproximadamente duas vezes o tempo relativo à utilização desta primitiva.

A partir deste ponto, uma proporção foi feita, com o intuito de comparar os desempenhos obtidos na execução (através de medições) e na simulação. A estratégia de análise adotada consiste na divisão dos tempos médios obtidos na execução (paralelo /

seqüencial) e na simulação (paralelo / seqüencial). Ao se dividir o tempo da composição paralela com o tempo da composição seqüencial, é esperada uma quantificação do impacto da utilização da primeira.

$$Razão(paralelo/seqüencial) = \frac{tempo\_composicao\_paralela}{tempo\_composição\_sequencial} \qquad (12)$$

Com o intuito de subsidiar a avaliação de resultados, pode-se inferir qual o ganho percentual da utilização da abordagem da composição paralela. Para isso, basta realizar uma subtração simples, subtraindo de 1 (correspondente a 100%) o valor da razão. Esta diferença, se multiplicada por 100, indicará o ganho percebido pela adoção da primeira possibilidade de composição.

$$Ganho\_percebido = 1 - Razão(paralelo/seqüencial) \qquad (13)$$

Através do emprego da metodologia proposta, não é esperado gerar resultados equivalentes ao visto no mundo real, mas sim obter uma aproximação válida deste ganho percebido, com objetivo de fornecer indicações para uma possível escolha dentre as várias possibilidades de composição. A Tabela 6 resume os resultados obtidos através da aplicação das fórmulas (12) e (13) para os casos medidos (Subseção 4.2) e simulados (Subseção 4.3).

Tabela 6. Razões e ganhos percebidos para medição e simulação

	Medição (em ms)	**Simulação com GSPN (em ms)**
Razão (paralelo/seqüencial)	0,958	0,902
Ganho Percebido	0,042 (4,2%)	0,098 (9,8%)

Os resultados obtidos mostram que, dentro de uma margem de erro aceitável (5,6%), uma decisão estratégica acerca de qual possibilidade de composição usar poderia ser tomada com auxílio da metodologia proposta.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi proposta uma metodologia baseada em modelos analíticos de redes de Petri estocásticas generalizadas com o intuito de se avaliar possibilidades de composição de *Web Services*, tendo como foco principal o desempenho (tempo de execução) da mesma. Um estudo de caso bastante conhecido na literatura foi utilizado de forma a mostrar a utilização prática da metodologia e sua validade no reconhecimento do melhor tipo de composição, tendo em vista a aplicação em questão.

O uso de GSPN para modelagem confirmou que este tipo de formalismo pode ser utilizado não apenas em sistemas de pequena escala, mas também em aplicações reais, tendo como vantagem a possibilidade de análise formal, bem como de aspectos quantitativos (foco deste trabalho), pois, sendo isomorfa às cadeias de Markov, é capaz

de gerar estados equivalentes, facilitando a tarefa de modelagem pelo usuário, que não perde as vantagens da utilização de modelos analíticos para avaliação de desempenho, além de não ser penalizado pelas dificuldades em trabalhar com as cadeias de Markov.

Como trabalho futuro, é desejado dar continuidade às questões vistas neste trabalho, em especial um tratamento mais abrangente a questão dos refinamentos, tendo em vista a modelagem de outros aspectos que não foram contemplados neste artigo, como, por exemplo, influência da *engine* em aplicações de maior complexidade. Deseja-se também aplicar a metodologia a outros estudos de casos mais complexos, envolvendo diversas possibilidades de composição. Vislumbra-se também a extensão deste trabalho para dar suporte a *Grid Services* [Foster 2002], tecnologia recente que adiciona novas características ao projeto de *Web Services*.

Referências Bibliográficas

ActiveBPEL, LLC (2005) "The Open Source BPEL Engine", disponível em http://www.activebpel.org. Acesso em 20/15/2005.

Andrews *et al* (2003) "Business Process Execution Language for Web Services (Version 1.1.)", disponível em http://www.ibm.com/developerworks/library/ws-bpel/. Acesso em 23/12/2005.

Benatallah, B., Dumas, M., Fauvet, M., and Rabhi, F. (2002) "Towards Patterns of Web Services Composition", em *Patterns and Skeletons for Parallel and Distributed Computing,* Springer Verlag, UK.

Bottcher, S., Dannewitz, C. (2005) "Service Composition on Top of Exchangable Protocols System", Proceedings of the 38th Annual Hawaii International Conference on Sciences (HICSS '05), pp. 282a - 292a.

Cardelli, L., Davies, R. (1999) "Service combinators for Web computing", em *IEEE Transactions on Software Engineering*, Volume 25 (3), pp 309 – 316.

Desrochers, A. A. (1994) "Applications of Petri Nets em Manufacturing Systems: Modeling, Control and Performance Analysis ", IEEE Press.

Ferrara, Andrea (2004) "Web Services: a Process Algebra Approach", em Proceeding of 2nd International Conference on Service Oriented Computing (ICSOC'04), pp. 242-251.

Foster, I. *et al*. (2002) "Grid Services for Distributed System Integration", em IEEE Press (Computer), vol. 35, no. 6, pp. 37-46. Junho.

Hamadi, R. e Benatallah, B. (2003) "A Petri Net-based Model for Web Service Composition", em *Fourteenth Australasian Database Conference (ADC2003)*, Adelaide, Australia.

Heindl, A. e German, R. (2001) "Performance modeling of IEEE 802.11 wireless LANs with stochastic Petri nets", em *Performance evaluation: an international journal*. Elsevier.

Hendricks, M. *et al* (2002) "*Java Web Services*", Alta Books.

Lammermann, S. eTyugu, E. (2001) "A specification logic for dynamic composition of services", em Prooceeding of International Conference on Distributed Computing Systems Workshop, Abril, pp 157 – 162.

Markov, A. (1971) "Extension of the limit theorems of probability theory to a sum of variables connected in a chain", reimpresso em Appendix B de: R. Howard. Dynamic Probabilistic Systems, vol. 1: Markov Chains. John Wiley and Sons.

Marsan, M.A. *et al*. (1995) "*Modelling with generalized stochastic Petri Nets*", Università degli studi di Torino, Dipartimento di informática.

Molloy, M.(1981) "On the Integration of Delay and Throughput Measures inDistributed Processing Models", Ph.D.Thesis, UCLA.

Natkin, S. (1980) "Les Réseaux de Petri Stochastiques et Leur Application a L'evaluation des Systemes Informatiques", Thèse de Docteur Ingegneur, CNAM, Paris, France.

Pallos, M. S. (2001) "Service-Oriented Architecture: a Primer", disponível em http://www.bijonline.com/PDF/SOAPallos.pdf. Acesso em 10/11/2005.

Radetzki, U., Cremers, A.B (2004) "IRIS: a framework for mediator-based composition of service-oriented software", Proceedings of IEEE International Conference on Web Services, 2004, pp. 752 – 756.

Zeng, Liangzhao, Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H. (2004) "QoS-aware middleware for Web Services composition", *IEEE Transactions on Software Engineering,* Volume 30 (5), pp. 311 – 327.

Zimmermann, A., Freiheit, J., German, R., Hommel, G. (2000) "Petri net modeling and performability evaluation with TimeNET 3.0", em Lecture Notes in Computer Science, Vol. 1786: Computer Performance Evaluation: Modeling Techniques and Tools, pages 188-202. Springer-Verlag.