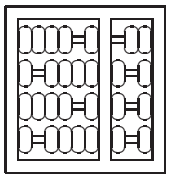

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Uma Arquitetura de Serviços Web Tolerante a Falhas para Sistemas de Gerência de Processos de Negócio

D. Z. G. Garcia *M. B. F. de Toledo*

Technical Report - IC-06-018 - Relatório Técnico

October - 2006 - Outubro

Uma Arquitetura de Serviços Web Tolerante a Falhas para Sistemas de Gerência de Processos de Negócio

Diego Zuquim Guimarães Garcia*
Maria Beatriz Felgar de Toledo†

Resumo

Sistema de Gerência de Processos de Negócio (SGPN) apóia a realização de processos de negócio. Serviços Web estão sendo indicados como a tecnologia mais adequada para SGPN. Conseqüentemente, a incorporação de tolerância a falhas na arquitetura de serviços Web é essencial para a continuidade de processos. O objetivo deste artigo é propor uma arquitetura de serviços Web tolerante a falhas para utilização com SGPN. Ela emprega camadas de mediação e monitoramento, provendo transparência, interoperabilidade e privacidade. As principais contribuições deste artigo são: uma abordagem para oferecer tolerância a falhas para serviços Web considerando SGPN, uma arquitetura baseada na abordagem proposta e sua implementação parcial.

1 Introdução

Tipicamente, Sistemas de Gerência de Processos de Negócio (SGPN's) são utilizados para atividades críticas por diversas organizações. Recentemente, serviços Web estão se tornando componentes importantes para SGPN's. Dessa forma, a disponibilidade de um SGPN passa a depender da disponibilidade dos serviços Web utilizados ao longo do processo de negócio.

Entretanto, apesar de garantia de disponibilidade e de qualidade de serviço (QoS - *Quality of Service*) em geral serem requisitos essenciais para sistemas críticos, não existe atualmente padrão para tolerância a falhas em serviços Web, como existe em outras tecnologias de *middleware* como, por exemplo, CORBA.

*Instituto de Computação, Universidade Estadual de Campinas, Caixa Postal 6176 CEP 13081-970 Campinas, SP, Brasil, e-mail: diego.garcia@ic.unicamp.br

†Instituto de Computação, Universidade Estadual de Campinas, Caixa Postal 6176 CEP 13081-970 Campinas, SP, Brasil, e-mail: beatriz@ic.unicamp.br

Tendo em vista tal deficiência, propostas para tolerância a falhas em serviços Web estão sendo desenvolvidas. Entretanto, nos trabalhos, de modo geral, serviços Web não são considerados como tecnologia para SGPN e, portanto, requisitos importantes não são tratados, tais como, transparência de tolerância a falhas para clientes, simplicidade, privacidade de dados, autonomia e compatibilidade com os padrões de serviços Web.

Este artigo foca na área de tolerância a falhas para serviços Web. O objetivo é propor uma arquitetura que, ao garantir alta disponibilidade de serviços Web, ofereça a continuidade de processos de negócio mesmo em presença de falhas.

A arquitetura estende o modelo tradicional de serviços Web, incorporando dois componentes. O componente Mediador é responsável por interagir com o repositório de serviços Web e gerenciar réplicas. O componente Monitor é encarregado de monitorar serviços e detectar erros.

Na arquitetura proposta, SGPN's permitem a utilização de serviços Web em processos de negócio para apoiar a cooperação entre organizações. Além disso, ela utiliza características da arquitetura básica de serviços Web para garantir simplicidade e compatibilidade com os padrões de serviços Web.

Outras características da arquitetura incluem o apoio de autonomia organizacional, transparência de tolerância a falhas e privacidade de dados críticos. Tais características são oferecidas pela definição da arquitetura, que engloba monitoramento do lado cliente e mediação remota ao cliente.

A seguir, na Seção 2 são introduzidos conceitos básicos, incluindo Sistemas de Gerência de Processos de Negócio, Serviços Web e Tolerância a Falhas. Em seguida, a abordagem proposta para uma arquitetura tolerante a falhas para SGPN's baseados em serviços Web é apresentada na Seção 3. As Seções 4 e 5 descrevem a arquitetura e aspectos de implementação, respectivamente. Na Seção 6, uma avaliação de desempenho do sistema que implementa a arquitetura é descrita. A Seção 7 expõe trabalhos de pesquisa na área. Finalmente, conclusões e trabalhos futuros são apresentados na Seção 8.

2 Conceitos Básicos

Nesta seção são introduzidos conceitos básicos para a compreensão dos temas envolvidos neste trabalho, incluindo Sistemas de Gerência de Processos de Negócio, Serviços Web e Tolerância a Falhas.

2.1 Sistemas de Gerência de Processos de Negócio

Considerando o contexto de negócio atual, um novo sistema de *middleware*, chamado de Sistema de Gerência de Processos de Negócio (SGPN), foi introduzido com

o objetivo de prover controle para definir e coordenar a execução de processos de negócio [11].

Diferentemente dos Sistemas de Gerência de *Workflow* (SGWf's) tradicionais, os SGPN's enfatizam a gerência de processos que cruzam fronteiras organizacionais e o aspecto da dinamicidade de processos [13, 20].

SGPN's oferecem automatização de processos de negócio. De modo geral, o ciclo de vida de automatização (Figura 1) inicia com a definição do processo. Em seguida, a definição do processo é registrada em um SGPN. Para executar um processo, o sistema cria uma instância de processo e, então, coordena, monitora e registra sua execução. O registro pode ser analisado, podendo gerar uma definição aprimorada do processo de negócio [19].

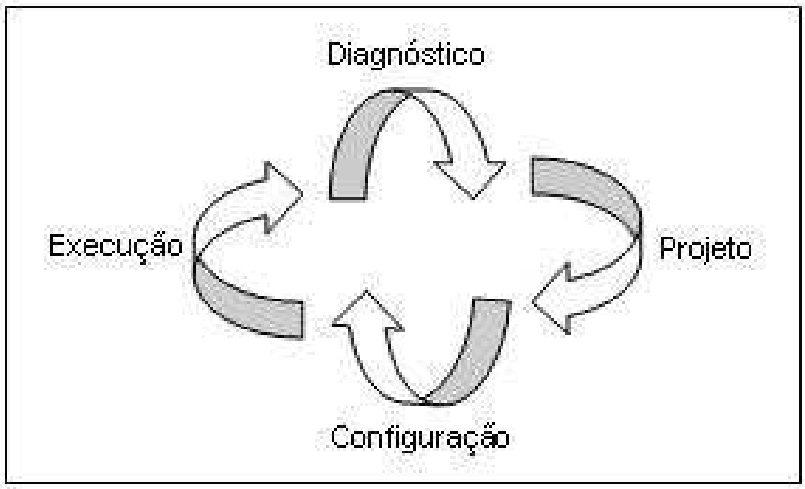

Figura 1: Ciclo de vida de gerência de processo de negócio

2.2 Serviços Web

A aplicação da Computação Orientada a Serviços na Web é realizada pela tecnologia de Serviços Web [17]. Um serviço Web é um tipo de serviço, identificado por um URI (*Uniform Resource Identifier*), que expõe funcionalidades accessíveis através da Internet ou de *intranets* e cuja descrição e transporte utilizam padrões da Web abertos [1, 5].

A Figura 2 ilustra a arquitetura básica de serviços Web. Os principais padrões empregados no âmbito da tecnologia de serviços Web são descritos a seguir.

- *Web Services Description Language* (WSDL). É um formato *Extensible Markup Language* (XML) que proporciona uma descrição de serviço Web compreensível por computador [6].

- *Universal Description Discovery & Integration* (UDDI). Provê um repositório de descrições de serviço Web [7].
- SOAP. É um protocolo baseado em XML que possibilita a comunicação entre serviços Web [16].

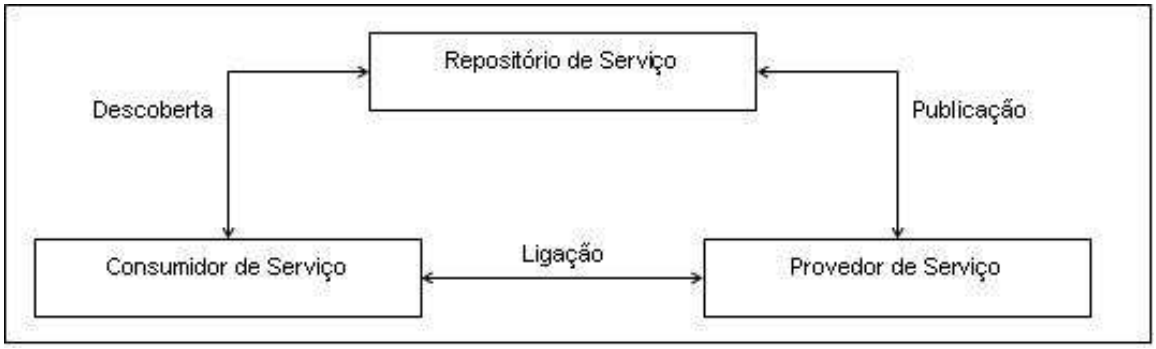

Figura 2: Arquitetura básica de serviços Web

2.3 Tolerância a Falhas

Segurança no funcionamento [1] de um sistema é a habilidade de evitar defeitos de serviço que são mais freqüentes e graves do que é aceitável.

Segurança no funcionamento indica a qualidade de serviço oferecida pelo sistema e envolve os seguintes atributos: disponibilidade, confiabilidade, segurança crítica [2], integridade e manutenibilidade [3] [3].

Tolerância a falhas é um dos meios para alcançar segurança no funcionamento e significa evitar defeitos de serviço na presença de falhas. Redundância é intensamente aplicada para implementar técnicas de tolerância a falhas. Para um sistema ser capaz de tolerar uma falha, uma forma de redundância deve ser empregada [12].

Técnicas de tolerância a falhas podem ser classificadas de acordo com as fases de aplicação: detecção de erro, confinamento, recuperação de erro e tratamento de falha [2].

3 Abordagem

Nesta seção a abordagem proposta para uma arquitetura para SGPN's (Seção 2.1) é apresentada. A arquitetura é baseada na tecnologia de serviços Web (Seção 2.2).

[1] Do inglês *dependability*.
[2] Do inglês *safety*.
[3] Do inglês *maintainability*.

Ela oferece aspectos fundamentais para SGPN's, tais como, cooperação, privacidade, autonomia e tolerância a falhas (Seção 2.3) transparente para clientes.

A arquitetura de tolerância a falhas faz com que os clientes percebam como um serviço Web único um serviço implementado com múltiplas réplicas desenvolvidas para plataformas computacionais diversas e executadas em localizações diferentes.

Os passos para descoberta e execução de serviços Web são apresentados na Figura 3. A seguir, esses passos são descritos.

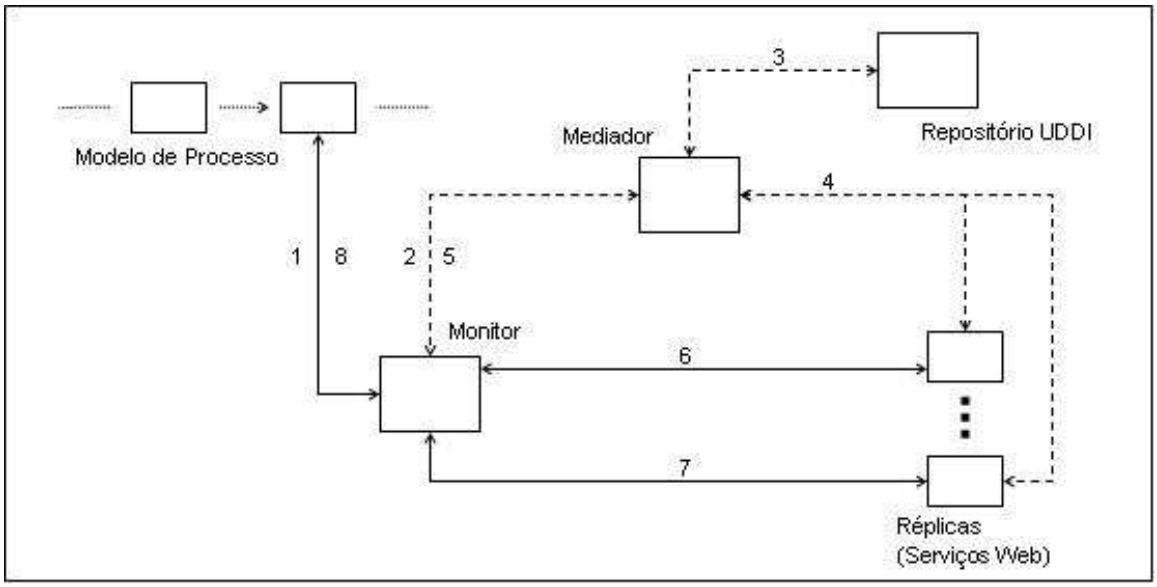

Figura 3: Abordagem para serviços Web tolerantes a falhas para utilização com SGPN.

A cada cliente corresponde um Monitor. O Mediador é um componente remoto e, assim como o Repositório UDDI, uma quantidade variada de Mediadores pode ser utilizada e instalada em diversos locais, por exemplo, junto com Monitores, Repositórios UDDI ou em máquinas independentes. Essas características garantem a autonomia organizacional em relação à capacidade de definir e executar processos e empregar Mediadores.

SGPN's permitem a utilização de serviços Web como atividades de processos de negócio. Com isso é possível incorporar ao processo de negócio de uma organização particular serviços executados por outras organizações, apoiando a cooperação.

Em tempo de definição de processo, modelos de atividades podem corresponder a serviços Web, referenciando tipos de serviços por meio do conceito de Modelo Técnico (*tModel*) do padrão UDDI (Seção 3.1).

Em tempo de execução de processo, quando atividades realizadas por serviços Web são atingidas, Máquinas de Automatização de Processos solicitam a Mediadores,

via Monitores, instâncias de serviços correspondentes às atividades (Passos 1 e 2 na Figura 3).

Mediadores localizam instâncias dos tipos de serviços solicitados em Repositórios UDDI (Passo 3) e executam testes para a seleção de serviços Web (Passo 4). Em seguida, retornam endereços de serviços Web a Monitores (Passo 5). SGPN's podem então consumir serviços por meio dos pontos de acesso retornados.

Em tempo de execução de atividades, Monitores verificam o andamento de execuções de serviços (Passo 6). Caso ocorram erros, Mediadores são contatados para que réplicas de serviços Web possam ser utilizadas (Passo 7). Concluídas as execuções de serviços, resultados gerados são retornados a clientes (Passo 8).

3.1 Modelos Técnicos do UDDI

O foco do padrão UDDI é oferecer descoberta de provedores de serviços Web, serviços disponibilizados e interfaces técnicas utilizadas para acessá-los.

Um repositório UDDI possui quatro tipos de dados principais. A estrutura *businessEntity* provê informação sobre um provedor. Informações descritivas para serviços são definidas em estruturas *businessServices* e, informações necessárias para ligar e interagir com serviços Web são definidas em estruturas *bindingTemplates* relacionadas. Cada *bindingTemplate* pode conter referências para diversas estruturas *tModels*, que são utilizadas para representar conceitos únicos e provêem uma estrutura que permite reuso e, conseqüentemente, padronização [7].

Para auxiliar na publicação de descrições WSDL em UDDI, documentos WSDL são divididos em dois grupos. A Interface de Serviço, composta de elementos *portType* e *binding*, contém uma definição de serviço utilizada para implementar diversos serviços. A Implementação de Serviço, composta de elementos *service* e *port*, contém uma descrição de serviço que implementa uma interface de serviço [6].

A Tabela 1 mostra a sugestão de mapeamento entre os elementos do WSDL e as estruturas do UDDI para a publicação de serviços Web [8].

Elemento do WSDL	**Estrutura do UDDI**
service	*businessService*
port	*bindingTemplate*
portType	*tModel*
binding	*tModel*

Tabela 1: Mapeamento WSDL-UDDI

Para descrever um serviço compatível com um conjunto de especificações, referências aos *tModels* que representam esses conceitos são colocadas na estrutura *bindingTemplate*. Estruturas *bindingTemplates* que se referem ao mesmo conjunto de *tModels* são do mesmo tipo. Dessa forma, *tModels* podem ser utilizados como

base para a definição de grupos de réplicas para proporcionar tolerância a falhas em arquiteturas orientadas a serviços Web, oferecendo simplicidade e, em conjunto com as demais características da abordagem proposta, compatibilidade com os padrões de serviços Web.

4 Arquitetura

Esta seção trata da arquitetura que oferece a abordagem proposta para serviços Web tolerantes a falhas (Seção 3), considerando as necessidades de evolução de SGWf's para a incorporação de funcionalidades de gerência de processos de negócio interorganizacionais dinâmicos. Os componentes que constituem a arquitetura e suas interações são descritos enfocando as alterações incorporadas à arquitetura de serviços Web atual.

A arquitetura de tolerância a falhas é composta de elementos que são responsáveis por funções de gerência de réplicas (serviços Web com funcionalidade idêntica), detecção de erros e confinamento.

O estágio de configuração envolve a publicação de serviços Web. Estruturas do UDDI são utilizadas por provedores para registrar réplicas. Além disso, a API (*Application Program Interface*) do UDDI é utilizada para fornecer os dados necessários para o registro de réplicas de serviços Web no repositório, conforme descrito na Seção 3.1.

No estágio de execução, o Mediador é utilizado para descobrir grupos de réplicas de serviços, durante a fase de busca de serviços. Após a criação de um grupo de réplicas, ele realiza testes para determinar qual serviço será utilizado. O Mediador interage com o Monitor, configurando-o para a verificação de estado do serviço durante a sua execução. Nesse estágio, o Monitor é responsável pela detecção de erros. Caso isso ocorra, ele interage com o Mediador para a obtenção de uma réplica do serviço.

A seguir, os principais componentes da arquitetura são descritos.

4.1 Monitor

O componente Monitor é responsável pela detecção e notificação de erros e confinamento. Esse componente foi estabelecido como uma camada de interceptação localizada no lado do cliente.

Ele monitora a execução de serviços Web, realiza testes de disponibilidade e limite de tempo e analisa respostas processadas para detectar erros nos serviços. Se um serviço Web apresentar um erro durante sua invocação ou execução, por exemplo, se ele não aceitar uma requisição, não completar a execução ou finalizar com uma exceção, o Monitor notifica o Mediador.

Em casos de erros de serviços após a seleção dos mesmos pelo Mediador, o Monitor

pode transferir requisições para réplicas obtidas por meio do Mediador, a fim de garantir a não-interrupção do processo.

O componente Monitor, ao intermediar as interações do cliente com o Mediador e o provedor de serviço Web, garante transparência de tolerância a falhas e privacidade de dados críticos para o cliente. Ele é encarregado de obter do cliente o tipo de serviço Web desejado, juntamente com o método a ser executado e os parâmetros exigidos para sua execução. Além disso, esse componente invoca o Mediador para realizar o mapeamento do tipo de serviço em uma instância de serviço Web disponível. O Monitor também é responsável por retornar as respostas obtidas para o cliente e pode atualizar o Repositório UDDI com as medidas de qualidade de serviço dos serviços monitorados.

4.2 Mediador

O componente Mediador é responsável pela gerência de réplicas. Ele cria grupos de réplicas de serviços Web utilizando o conceito de Modelo Técnico do padrão UDDI (Seção 3.1). Serviços Web diversos são agregados por meio do compartilhamento de especificações, incluindo descrições de interface WSDL. Grupos são estabelecidos dinamicamente, refletindo a própria dinamicidade típica do ambiente Web, em que novos serviços são continuamente oferecidos e ofertas são interrompidas.

Propriedades de gerência de réplicas podem ser definidas no Mediador. Por exemplo, é possível definir a quantidade de réplicas que compõem um grupo, tanto em relação ao número de serviços de provedores diferentes quanto em relação ao número de pontos de acesso de um serviço de um mesmo provedor. Os tipos de teste que serão aplicados para a seleção de réplicas também podem ser definidos.

O Mediador também é responsável por testar serviços Web durante a fase de seleção de serviços e pode utilizar as medidas de qualidade de serviço mantidas no Repositório UDDI para a seleção.

4.3 Repositório UDDI

O Repositório UDDI será estendido para incluir medidas de qualidade de serviço (por exemplo, tempo de resposta, taxa de serviço, confiabilidade e disponibilidade) que serão atualizadas pelo Monitor e apoiarão o Mediador no momento da seleção de serviços.

5 Implementação

Nesta seção aspectos de implementação das extensões propostas para tolerância a falhas em arquiteturas orientadas a serviços Web são apresentados.

A arquitetura está sendo implementada na linguagem Java, utilizando o JDK (Java Development Kit) 1.5. O servidor de aplicação Apache Tomcat 4.1 está sendo utilizado. Além disso, para o estabelecimento da arquitetura básica de serviços Web dois pacotes estão sendo empregados: o Apache Axis 1.3, que oferece uma máquina SOAP, além de apoio para o WSDL, e o jUDDI 0.9, uma implementação do UDDI.

O componente Mediador foi desenvolvido como um serviço Web, utilizando a biblioteca Apache Axis. Essa abordagem permite que a arquitetura possa ser utilizada tanto em um ambiente mais restrito, por exemplo, em *intranets*, quanto em ambientes abertos, por exemplo, na Internet. Além disso, essa abordagem oferece ao cliente a possibilidade de utilizar mediadores com funcionalidades diferentes, de acordo com a necessidade.

As funcionalidades de gerência de réplicas e detecção de erros foram parcialmente implementadas. Apenas medidas de disponibilidade são consideradas e essas medidas são obtidas por meio de testes. Entretanto, testes adicionais e extensões do UDDI podem ser implementados para garantir a seleção e a verificação de réplicas mais apuradas.

Serviços Web devem implementar um método que quando invocado retorna um valor verdadeiro, indicando que o serviço está acessível. Esse método é necessário para o monitoramento de serviços, executado pelo Monitor, e para a seleção de réplicas, executada pelo Mediador.

Apesar de existir a necessidade de serviços Web implementarem uma operação adicional de acordo com a abordagem sugerida, a arquitetura de serviços Web tolerante a falhas pode existir em conjunto com a arquitetura tradicional de serviços Web.

A arquitetura está sendo implementada de modo compatível com o perfil básico versão 1.0 da WS-I (*Web Services Interoperability Organization*) [4], que determina regras de interoperabilidade para serviços Web.

6 Avaliação de Desempenho

Os experimentos desenvolvidos para avaliar o desempenho do sistema de tolerância a falhas e os resultados obtidos são discutidos nesta seção. A avaliação focou no impacto provocado no desempenho da arquitetura de serviços Web pela inclusão de funcionalidades de tolerância a falhas.

Além disso, para testar a eficácia da arquitetura de tolerância a falhas proposta, falhas foram introduzidas de forma controlada durante a realização dos experimentos.

Para avaliar o desempenho do sistema que implementa a arquitetura proposta, testes foram desenvolvidos em um ambiente computacional cuja configuração é apresentada na Tabela 2.

Em uma máquina foram instalados o cliente e o componente Monitor. O servidor

Sistema operacional	Linux Fedora Core 2.6.16-1.2066_FC4
Ferramenta de desenvolvimento	JDK v1.5.0_06
Servidor de aplicação	Apache Tomcat 4.1.24
Máquina SOAP	Apache Axis 1.3
Repositório UDDI	jUDDI 0.9rc4
Processador	Intel Pentium 4A / 2800 MHz / 133 MHz
Memória	512 MB / 266 MHz
Placa mãe	Intel Sea Breeze D845GVSR / Chipset Intel Brookdale-G i845GV
Disco rígido	Samsung SP0411N / 40 GB / 7200 RPM / Ultra-ATA 133
Rede	Ethernet / 100 Mbps

Tabela 2: Parâmetros do sistema

de aplicação foi instalado em duas máquinas, que foram utilizadas para a instalação do componente Mediador, do repositório UDDI e dos serviços Web. Os serviços Web empregados nos testes foram implementados utilizando a linguagem de programação Java e a biblioteca Apache Axis.

Experimentos foram realizados para avaliar o impacto no tempo de resposta provocado pelo sistema de tolerância a falhas.

Três situações foram testadas. Inicialmente, a arquitetura básica de serviços Web foi avaliada, ou seja, o sistema de tolerância a falhas proposto não foi empregado. Nesse caso, o cliente descobre um serviço Web por meio do repositório UDDI e o invoca diretamente. A segunda situação testada foi semelhante à primeira, exceto pelo fato de o sistema de tolerância a falhas ter sido utilizado. Para avaliar um caso caracterizado pela existência de erros, na terceira situação, testada foram injetadas falhas no sistema de forma controlada.

A partir da comparação dos resultados obtidos nos dois últimos casos com o resultado do primeiro caso testado, foi possível avaliar o impacto provocado no tempo de resposta pela arquitetura proposta.

A Figura 4 mostra o tempo de resposta, medido em milésimo de segundo (ms), para os três casos avaliados. O tempo de resposta médio foi de aproximadamente 68 ms quando as extensões da arquitetura de serviços Web não foram utilizadas. Quando as extensões para tolerância a falhas foram utilizadas, o tempo de resposta foi de 108 ms, para o caso livre de falhas, e 138 ms, para o caso em que falhas foram injetadas.

Um serviço Web executado com tolerância a falhas apresentou um custo adicional médio de 58,8 % no tempo de resposta, quando comparado com a execução do mesmo serviço sem tolerância a falhas. Quando falhas foram introduzidas, ocorreu um acréscimo no custo adicional, que foi de 27,8 % em relação ao tempo de resposta

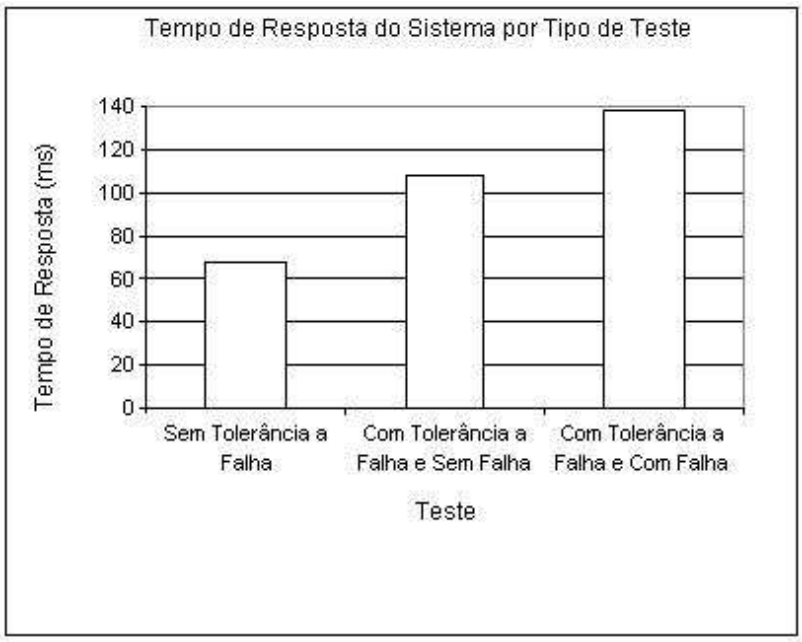

Figura 4: Tempo de resposta do sistema de tolerância a falhas

da execução do mesmo serviço no caso livre de falhas. Como o foco inicial do trabalho não foi em desempenho, melhorias serão implementadas a fim de reduzir o custo adicional futuramente.

Outros experimentos foram realizados para avaliar a sobrecarga proporcionada pelos componentes Monitor e Mediador.

A sobrecarga, medida em termos da utilização de CPU, foi avaliada em duas situações. A primeira situação avaliada foi caracterizada pela inexistência de falhas. Na segunda situação foram injetadas falhas no sistema.

A Figura 5 apresenta a porcentagem de utilização de CPU do sistema. A sobrecarga média gerada pelo Mediador foi de aproximadamente 13,8 %, e pelo Monitor, 13,5 %, em uma configuração sem falhas. Com a introdução de falhas, a sobrecarga média gerada foi de 16,3 % para o Mediador, e 13,8 % para o Monitor.

7 Trabalhos Relacionados

Esta seção apresenta trabalhos sobre tolerância a falhas em serviços Web. Em grande parte dos trabalhos a idéia de utilizar mediação na arquitetura de serviços Web é compartilhada. Entretanto, eles empregam técnicas e mecanismos variados de tolerância a falhas, incluindo, replicação passiva, replicação ativa, modelo de N-versões e *checkpointing/rollback*. Em geral, nesses trabalhos alguns requisitos de SGPN não são

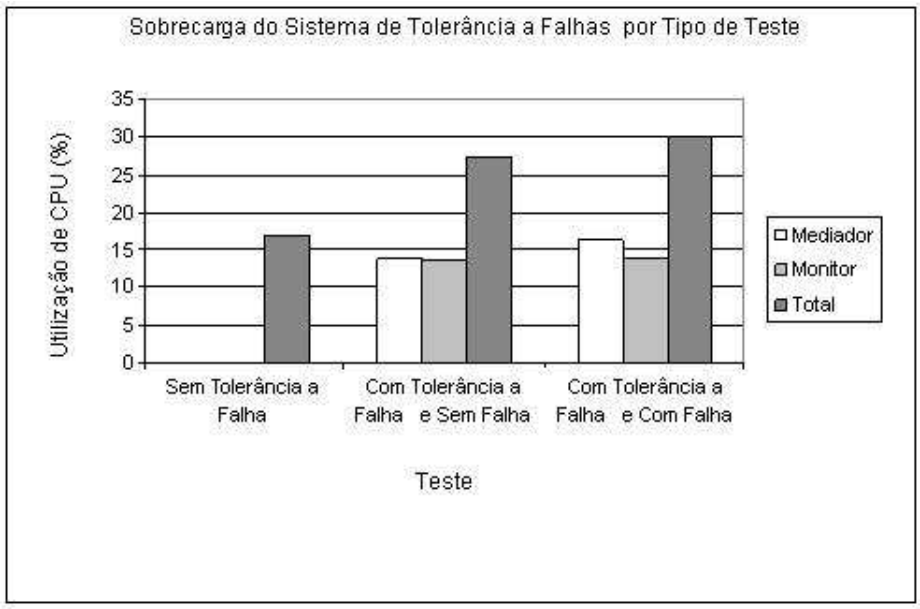

Figura 5: Sobrecarga do sistema de tolerância a falhas

considerados, tais como, interoperabilidade, transparência, autonomia e privacidade.

Em [10] é discutida a necessidade de aplicações de grade computacional e de computação distribuída em geral serem projetadas com tolerância a falhas para alcançar robustez. Entretanto, a comunidade Web não tem focado nesse aspecto. Essa deficiência influenciou a definição de modificações na camada SOAP com o objetivo de oferecer tolerância a falhas por mecanismos de *checkpointing* e *rollback*. A arquitetura para o desenvolvimento de serviços Web tolerantes a falhas proposta foca em recuperação de erros e, diferentemente da abordagem apresentada neste artigo, não trata de gerência de múltiplas cópias de serviços para tolerância a falhas.

Assim como na abordagem proposta neste trabalho, a técnica de replicação passiva também é explorada em [14]. Para alcançar tolerância a falhas, são propostas alterações nos padrões WSDL e SOAP para permitir, respectivamente, a especificação de réplicas de serviços Web e o direcionamento de requisições de serviços. A proposta aqui apresentada emprega camada de mediação na arquitetura de serviços Web, que oferece as funcionalidades necessárias para a gerência de réplicas e, portanto, não depende de alterações na linguagem para descrição de interface e no protocolo para intercâmbio de mensagem de serviços Web.

Em [18] a técnica de replicação ativa é adotada em serviços Web para garantir tolerância a falhas transparente para clientes. A arquitetura possui um componente central despachante que contém os mecanismos responsáveis pela gerência de réplicas e atua como um mediador entre clientes e provedores, assim como proposto na arquitetura apresentada neste artigo. Para tratar do ponto de falha representado pelo

despachante, a arquitetura possui um componente despachante *backup*. Na arquitetura aqui proposta, o Mediador é implementado como um serviço Web, permitindo ao cliente utilizar diferentes mediadores, do mesmo modo como pode utilizar qualquer outro serviço Web. Além disso, diferentemente da abordagem adotada aqui, que utiliza modelos técnicos para o estabelecimento de grupos de réplicas de serviços, o componente despachante contém um sistema de configuração para a criação de grupos.

Uma implementação do modelo de N-versões para serviços Web é apresentada em [15]. O modelo de N-versões é um padrão de projeto para tolerância a falhas já consolidado. Nesse modelo, com o intuito de evitar erros proporcionados, por exemplo, por problemas de especificação e implementação, réplicas são desenvolvidas como diferentes versões. Na proposta, o *stub* cliente é incrementado para permitir ao cliente fornecer uma lista de serviços. Semelhantemente à proposta apresentada neste artigo, réplicas são definidas por serviços Web de funcionalidade equivalente, e a exigência de N-versões é sustentada pela característica de fraco acoplamento da tecnologia de serviços Web. Entretanto, diferentemente da abordagem aqui proposta, a implementação do modelo de N-versões apresentada exige que o próprio cliente obtenha o estado do serviço Web.

8 Conclusões e Trabalhos Futuros

Neste artigo, uma arquitetura de serviços Web tolerante a falhas para utilização com SGPN foi descrita. A abordagem que orientou o projeto da arquitetura foi detalhada, e aspectos de implementação da arquitetura discutidos. Por fim, testes preliminares para a avaliação de desempenho do sistema que implementa a arquitetura proposta foram executados. A abordagem pode ser incorporada à arquitetura tradicional de serviços Web com alterações mínimas, a continuidade de processos de negócio é garantida pela tolerância a falhas oferecida pela arquitetura e as extensões introduzem um custo adicional aceitável se comparado com os benefícios proporcionados.

As principais contribuições deste artigo são:

- proposta de abordagem para prover tolerância a falhas para serviços Web, considerando SGPN's;
- arquitetura de serviços Web orientada pela abordagem sugerida;
- implementação parcial da arquitetura proposta.

Restrições da abordagem e possibilidades de trabalhos futuros são apresentadas a seguir:

- erros de um mesmo serviço Web são detectados antes de transferir a requisição para uma réplica. Um serviço de gerência de qualidade de serviço pode ser incorporado à arquitetura para que serviços incorretos possam ser desconsiderados;

- um fator que contribui para desestabilizar o sistema é a falta de coordenação de redirecionamento de requisições. Um mecanismo de balanceamento de carga pode ser empregado para evitar que vários clientes sejam direcionados para uma mesma réplica;

- integração com o SGPN WorkToDo [9].

Referências

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services: Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[2] T. Anderson and P. A. Lee. *Fault tolerance - principles and practice*. Prentice-Hall, Englewood Cliffs, 1981.

[3] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, 2004.

[4] Keith Ballinger, David Ehnebuske, Martin Gudgin, Mark Nottingham, and Prasad Yendluri. *Basic Profile Version 1.0, Final Material*. Web Services-Interoperability Organization, April 16, 2004. http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html, accessed on 05/2006.

[5] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. *Web Services Architecture, W3C Working Group Note*. W3C, February 11, 2004. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, accessed on 05/2006.

[6] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. *Web Services Description Language, Part 1: Core Language, Version 2.0, W3C Working Draft*. W3C, May 10, 2005. http://www.w3.org/TR/2005/WD-wsdl20-20050510/, accessed on 05/2006.

[7] L. Clement, A. Hately, C. von Riegen, and T. Rogers. *UDDI, Version 3.0.2, UDDI Spec Technical Committee Draft*. OASIS, October 19, 2004. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, accessed on 05/2006.

[8] J. Colgrave. *A new approach to UDDI and WSDL.* IBM, July 19, 2003. www.ibm.com/developerworks/ webservices/library/ws-udmod1/, accessed on 05/2006.

[9] Jackson da Cruz. WorkToDo Flex - Um Sistema de Gerenciamento de Workflows Flexíveis. Master's thesis, Universidade Estadual de Campinas, Instituto de Computação, 2005.

[10] Vijay Dialani, Simon Miles, Luc Moreau, David De Roure, and Michael Luck. Transparent fault tolerance for web services based architectures. In *Euro-Par '02: Proceedings of the 8th International Euro-Par Conference on Parallel Processing*, pages 889–898, London, UK, 2002. Springer-Verlag.

[11] M. Fantinato, M. B. F. de Toledo, and I. M. S. Gimenes. Arquiteturas de sistemas de gerenciamento de processos de negócio baseados em serviços. Technical Report IC-05-06, Universidade Estadual de Campinas, Instituto de Computação, Campinas, Brasil, 2005.

[12] Felix C. Gartner. Fundamentals of fault-tolerant distributed computing in asynchronous environments. *ACM Comput. Surv.*, 31(1):1–26, 1999.

[13] D. Hollingsworth. The workflow reference model 10 years on. In *Workflow Handbook 2004*. WfMC, Winchester, UK, 2004.

[14] Deron Liang, Chen-Liang Fang, Chyouhwa Chen, and Fengyi Lin. Fault tolerant web service. In *APSEC '03: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference*, pages 310–319, Washington, DC, USA, 2003. IEEE Computer Society.

[15] Nik Looker, Malcolm Munro, and Jie Xu. Increasing web service dependability through consensus voting. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 2*, pages 66–69, Washington, DC, USA, 2005. IEEE Computer Society.

[16] N. Mitra. *SOAP, Part 0: Primer, Version 1.2, W3C Recommendation.* W3C, June 24, 2003. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/, accessed on 05/2006.

[17] Mike P. Papazoglou and D. Georgakopoulos. Service-oriented computing - guest editorial. *Communications of the ACM*, 46(10):24–28, 2003.

[18] Giuliana Teixeira Santos, Lau Cheuk Lung, and Carlos Montez. Ftweb: A fault tolerant infrastructure for web services. In *EDOC '05: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05)*, pages 95–105, Washington, DC, USA, 2005. IEEE Computer Society.

[19] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In *BPM '03: Proceedings of the 1st International Conference on Business Process Management*, pages 1–12. Springer, 2003.

[20] Thomas Woodley and Stephane Gagnon. BPM and SOA: Synergies and challenges. In *WISE '05: Proceedings of the 6th International Conference on Web Information Systems Engineering*, pages 679–688. Springer, 2005.