Desafios na construção e validação da robustez de aplicações orientadas a serviços

Eliane Martins[1], Regina Moraes[2], Taisy Weber[3]
Cecília Mary Fisher Rubira[1], Ana Maria Ambrósio[4], Magnos Martinello[5]

[1]Instituto de Computação - Universidade Estadual de Campinas (Unicamp)
[2]CESET - Universidade Estadual de Campinas (Unicamp)
[3]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
[4]Instituto Nacional de Pesquisas Espaciais (INPE)
[5]Universidade Federal do Espírito Santo (UFES)

eliane@ic.unicamp.br, regina@ceset.unicamp.br, taisy@inf.ufrgs.br
cmrubira@ic.unicamp.br, ana@dss.inpe.br, magnos@inf.ufes.br

Abstract. *Service-orientation is a new paradigm that is emerging for the development of distributed and scalable applications. It is based on object-oriented and component-based paradigms, in which the services (autonomous platform-independent computational elements that can be described, published, discovered and accessed over the Internet using standard protocols) are the new elements for development and reuse. Service oriented applications are largely distributed and dynamic, structured as a set of services interfaces with little knowledge about the quality attributes of their implementations. Therefore, the probability of concurrency of service failures is high. In this paper we discuss the challenges related to the development and validation of service oriented applications based on Web Services considering that these applications should guarantee a desired high level of dependability and robustness typical of e-business and mission critical applications.*

Resumo. *Um novo paradigma emergiu para o desenvolvimento de aplicações distribuídas e de grande escala – o desenvolvimento orientado a serviços, uma evolução da orientação a objetos e do desenvolvimento de sistemas baseados em componentes em que a entidade básica de desenvolvimento passa a ser o serviço em si, e não quem presta esse serviço. Aplicações orientadas a serviços são altamente distribuídas e dinâmicas e usam serviços dos quais só se conhece a interface de utilização, sem informações sobre a sua qualidade. Portanto, as oportunidades de mau funcionamento são inúmeras, podendo acarretar em grandes perdas para as empresas ou organizações. Neste artigo discutimos os desafios a serem enfrentados não somente para a criação de aplicações orientadas a serviço baseadas em Web Services que sejam robustas, mas também para validar tanto os serviços usados quanto a composição dos mesmos na determinação se a aplicação final possui os atributos de qualidade desejados.*

1 Introdução

A Computação Orientada a Serviços é um novo paradigma no ambiente de desenvolvimento de aplicações distribuídas e de *e-business*. Ela é baseada nos paradigmas de orientação a objetos e de sistemas baseados em componentes, em que os serviços são os novos elementos para desenvolvimento e reuso. Serviços neste contexto são elementos computacionais de plataformas independentes e autônomas que podem ser descritos, publicados, revelados e alcançados pela Internet usando protocolos padrão. Este novo paradigma tem modificado a maneira como um sistema de software é desenvolvido e utilizado. Esta maneira de estruturar uma aplicação em um conjunto de serviços interconectados é chamada de Arquitetura Orientada a Serviços (*Service-Oriented Architectures* – SOA).

Nos últimos anos, várias formas de arquiteturas orientadas a serviço têm surgido e, entre elas, Web Services (WS) têm sido os mais comumente utilizados. WS é um tipo de sistema cliente-servidor que foi especialmente estruturado para fazer o melhor uso possível de padrões Web. Não apenas sistemas convencionais usuais na Web podem se beneficiar desses serviços, mas todos os tipos de dispositivos desde o menor sensor ao mais sofisticado equipamento doméstico, comercial ou industrial podem aproveitar as vantagens de sua forma padrão de comunicação. Sua arquitetura nos parece extremamente promissora para o desenvolvimento de sistemas onivalentes, como os mencionados no quinto desafio da SBC [SBC 2006].

Fundamental para integrar aplicações baseadas na Web e uma rede Internet é a interoperabilidade, que é alcançada através do uso de normas baseadas em XML (eXtensible Markup Language) como SOAP, WSDL e UDDI. O protocolo SOAP (*Simple Object Access Protocol*) é usado para a transferência de dados entre os serviços. A linguagem WSDL (*Web Service Description Language*) define um esquema XML para descrever os serviços disponíveis. A especificação UDDI (*Universal Description, Discovery, and Integration*) define a forma de publicar e descobrir informações sobre um serviço específico em um diretório ou registro de serviços.

Embora SOA e WS não representem o mesmo conceito, o relacionamento entre estas duas tecnologias é importante. Como colocado por Y. Natis em uma nota do Gartner [Natis 2003], o sucesso do WS pode auxiliar na definição de uma maneira prática de se utilizar SOA e as boas práticas arquiteturais do SOA irão auxiliar para que as iniciativas no desenvolvimento de serviços Web sejam bem sucedidas. O autor também afirma que, a partir de 2008, SOA irá prevalecer como prática de Engenharia de Software e finalmente terminar com o domínio que exerceu a arquitetura de software monolítica nos últimos 40 anos. Ainda coloca que a partir de 2008, SOA e WS serão implementados em conjunto em mais de 75% dos novos projetos baseados em SOA ou em WS. No que se refere ao seu uso, de acordo com um relatório do IDC (baseado em [Boulton 2005]), 2,3 bilhões de dólares foram gastos em software para WS ao redor do mundo em 2004, representando mais do que o dobro do montante gasto em 2003.

Atualmente, empresas de software estão lançando suas próprias soluções. Apenas para mencionar algumas iniciativas temos a Microsoft com a plataforma .NET e a Sun com a iniciativa Sun One, essa última prevendo a integração de serviços de Web Service com a especificação do J2EE (*Java 2 Enterprise Edition*). A IBM também está ativamente envolvida neste domínio com as ferramentas para ambientes de desenvol-

vimento e execução WSDL. É também importante mencionar iniciativas de grandes corporações, tais como SAP e Oracle, que prometem usar WS para disponibilizar suas aplicações; e a eBay que está expandindo sua atuação na linha de API para WS.

Padrões de WS são importantes para que se permita a disseminação de SOA. Várias iniciativas existem nesse sentido, tais como *Web Service Interoperability Organization* (WS-I), que tem por objetivo promover a interoperabilidade entre Web Services com base em definições e padrões comuns usados na indústria. A WS-I cria orientações e ferramentas para auxiliar os desenvolvedores a construir WSs interoperacionais. Outra iniciativa é a *Organization for the Advancement of Structured Information Standards* (OASIS) que, entre outras coisas, define um modelo de referência para SOA.

Estas iniciativas representam uma etapa essencial para a adoção e difusão de WS e SOA. No entanto, compor uma solução robusta baseada em SOA, especialmente para o desenvolvimento de aplicações de negócios ou de missão crítica, ainda se encontra nos estágios iniciais de desenvolvimento. Segundo a IEEE [IEEE 1990], a robustez é definida como o grau segundo o qual um sistema ou componente pode funcionar corretamente em presença de entradas inválidas ou de condições ambientais estressantes. As possibilidades de entradas inválidas ou de condições ambientais estressantes para aplicações distribuídas altamente dinâmicas e escaláveis são inúmeras. Adicionalmente, os aspectos de segurança, neste contexto distribuído e colaborativo, são também relevantes e não podem ser desconsiderados.

Este artigo é resultado de discussões acadêmicas conduzidas pelos autores em diversas oportunidades. Os desafios a serem enfrentados para a criação de aplicações SOA robustas baseadas em Web Services e para validar tanto os serviços utilizados quanto a composição dos mesmos buscando determinar se a aplicação final possui as qualidades desejadas são o nosso foco. Nas próximas seções discutiremos como a arquitetura orientada a serviços pode auxiliar na solução de alguns dos problemas relacionados ao quinto dos grandes desafios da SBC e como, para servir ao quinto desafio, a área tem que vencer os seus próprios grandes desafios.

2 Computação orientada a serviços e o quinto grande desafio

Aplicações desafiadoras atuais e futuras são geralmente distribuídas, baseadas fortemente em comunicação. São ubíquas, presentes em todo lugar, controlando, apoiando ou suportando inúmeras atividades humanas. Em várias dessas aplicações um defeito pode levar a catástrofes com conseqüências severas para a qualidade de vida, do ambiente ou das atividades industriais e comerciais.

O quinto desafio foca o desenvolvimento dessas aplicações, denominadas onivalentes [SBC 2006], e as dificuldades para garantir disponibilidade, correção, segurança e persistência para essas aplicações. A esses atributos poderiam ser acrescidos vários outros como confiabilidade, integridade, confidencialidade e segurança funcional (*safety*), dependendo do nicho específico da aplicação e de eventuais requisitos necessários para a área [Moreira *et al* 2007].

Aplicações onivalentes distribuídas na Web operam sobre uma infra-estrutura de comunicação pouco robusta e estão sujeitas a inúmeros problemas devido à baixa disponibilidade, baixa confiabilidade, quase total ausência de privacidade, múltiplos domí-

nios administrativos, mau funcionamento de componentes de hardware e software com possíveis efeitos severos sobre a qualidade do serviço oferecido. Várias técnicas, métodos e paradigmas vêm sendo desenvolvidos e empregados para aumentar confiabilidade, disponibilidade, integridade, robustez e segurança de aplicações na Web. Por ser uma evolução de técnicas modernas de grande aceitação e popularidade e por contar com o apoio e interesse de fornecedores e desenvolvedores de software, o desenvolvimento orientado a serviços parece-nos um paradigma promissor para garantir o desenvolvimento de sistemas mais robustos, confiáveis e seguros para ambientes de larga escala.

Não se deve esperar que novas tecnologias, que venham a ser desenvolvidas, evitem totalmente a ocorrência de falhas de hardware ou software e o conseqüente mau funcionamento das aplicações. Novas tecnologias poderão diminuir a probabilidade de ocorrência de falhas devidas à fadiga de componentes físicos, erro de projeto ou bugs de programação, mas seguramente vão introduzir novas suscetibilidades a falhas e novos modos de defeito ainda não previstos.

Naturalmente o desenvolvimento de sistemas denominados fidedignos no quinto desafio da SBC, ou seja, sistemas computacionais de qualidade, que operem adequadamente mesmo na presença de falhas e no qual, mesmo em situações críticas, se possa justificadamente colocar confiança na correta prestação do serviço, é um desafio para ser vencido no longo prazo. Desenvolvimento e experimentação com novos paradigmas e tecnologias de construção de sistemas, aproveitamento de padrões existentes e largamente aceitos e considerável investimento em verificação e validação devem permitir sistemas muito próximos às necessidades de qualidade, robustez e fidedignidade dos usuários das próximas décadas de aplicações Web de larga escala.

3 Desafios na construção e aplicação de arquiteturas de software robustas

Para a criação de aplicações SOA baseadas em Web Services que sejam robustas é importante seguir diretrizes de como utilizar as técnicas disponíveis para o projeto de arquiteturas confiáveis buscando uma maneira de compor os serviços dos usuários. O estabelecimento dessas diretrizes é um dos inúmeros desafios a serem vencidos.

Assegurar que as políticas de segurança sejam respeitadas para que serviços de diferentes fornecedores possam ser utilizados também é um desafio. Neste aspecto é necessário lidar com segurança e confidencialidade de mensagens e preservar identidades em seus respectivos domínios na troca de informação de segurança. Cada uma dessas preocupações representa um desafio.

Validação é outro aspecto essencial e um enorme desafio a ser vencido. Uma investigação aprofundada de como utilizar as técnicas de testes disponíveis, no contexto de WS e SOA, assim como a modelagem e avaliação de desempenho neste contexto seria útil para assegurar que o nível desejado da qualidade seja atingido e mantido, apesar das constantes mudanças que ocorrem neste contexto. Antes de disponibilizar serviços na Internet, é preciso testá-los rigorosamente. Um serviço defeituoso na Web não é desejável devido à quantidade enorme de recursos da rede pública que ele consome. Como WSs são uma nova infra-estrutura para aplicações de negócios e também de missão crítica, faz-se necessário testá-los na presença de falhas para determinar se o seu comportamento em um ambiente hostil não trará conseqüências catastróficas aos negócios e aos serviços públicos.

Não menos importante é a arquitetura da aplicação que utiliza esses serviços: por mais robustos que estes sejam individualmente, sua composição pode não ser robusta, devido a conflitos na interação entre eles. Portanto, o desenvolvimento e popularização de diretrizes e técnicas para a construção de arquiteturas robustas também são desafios importantes a serem vencidos para alcançar a qualidade necessária da aplicação final.

Outro grande desafio é adaptar as técnicas, métodos e mesmo ferramentas existentes ao contexto de aplicações SOA, para as quais as mudanças são contínuas e imprevisíveis. Em um primeiro momento é possível que os serviços Web sejam predominantemente estáticos, criados a partir de aplicações já existentes, podendo a busca por novos serviços ser dinâmica já neste primeiro momento. Mas, na medida em que experiência com o uso do paradigma é acumulada, os serviços não serão mais necessariamente estáticos, podendo, estes também, serem dinâmicos, diferentes a cada invocação por uma aplicação. Além disso, os papéis de provedor e usuário de serviços também não serão muito claros, devido à complexidade da orquestração entre os serviços.

Para estabelecer diretivas que facilitem o desenvolvimento e validação de aplicações robustas, distribuídas, dinâmicas de larga escala e enderecem os desafios mencionados, devem ser investigadas respostas às seguintes questões:

1. Como projetar uma arquitetura que ofereça o nível desejado de confiabilidade, disponibilidade e desempenho, dado que os serviços podem mudar durante a execução e a única informação que temos dos mesmos é a especificação de sua interface pública?
2. Como as técnicas de testes, em especial aquelas baseadas em modelos, podem ser adaptadas para testar serviços e aplicações nesse ambiente em contínua mudança?
3. Como testar a robustez dos serviços sem alterar a operação dos mesmos?
4. Qual arquitetura de testes deve ser utilizada de forma a permitir testes dos diferentes serviços em diferentes plataformas, sendo que, em muitos casos, os serviços deverão ser testados remotamente?
5. Como avaliar o risco associado ao uso de um serviço em uma aplicação crítica, dado o contexto dinâmico e em constante evolução?
6. Como garantir que o nível de desempenho desejado seja alcançado e que se mantenha nesse ambiente em constante evolução?

Nas próximas seções, um aprofundamento da discussão de cada uma dessas questões permitirá um maior conhecimento dos desafios e um esboço de caminhos investigativos para possíveis respostas a cada uma dessas questões.

3.1 Arquiteturas orientadas a serviços robustas

O principal desafio do projeto de uma arquitetura robusta é adaptar técnicas já usadas para a composição tolerante a falhas de componentes estáticos, ao contexto de aplicações baseadas em Web Services. Quando se integra um serviço Web a uma aplicação, essa integração é geralmente dinâmica e sua implementação pode evoluir durante a execução. Geralmente o desenvolvedor da aplicação (usuário do WS) só possui como

informação a especificação da interface pública do serviço, sem indicações da qualidade deste serviço. Para que a aplicação, que integra esses serviços, possa ter a qualidade desejada, deve-se focar em soluções arquiteturais que permitam garantir a sua robustez.

Uma abordagem já proposta para a composição tolerante a falhas de aplicações concorrentes é baseada na composição de contratos [Silva Jr. *et al* 2003]. Um contrato de composição estende a noção de coordenação de contratos de forma a incluir o tratamento de exceções baseado em ações atômicas coordenadas [Xu *et al* 1995]. Um contrato de coordenação é uma conexão que pode ser estabelecida entre um grupo de objetos, a qual é regida por regras e restrições necessárias para que ocorra a colaboração. O aprofundamento e a aplicação desta abordagem pode ser a resposta desejada para o projeto de arquiteturas robustas auxiliando a vencer este desafio.

Os trabalhos de Tartanoglu *et al.* [Tartanoglu *et al* 2003] e Gorbenko *et al.* [Gorbenko *et al* 2007] analisam várias abordagens existentes para incorporar medidas de tolerância a falhas em composições de serviços Web, incluindo recuperação de erros por avanço e retrocesso. As soluções apresentadas permitem que a composição de serviços seja estruturada em termos de ações atômicas coordenadas que tem um comportamento bem definido considerando a presença de falhas e também a sua ausência.

Chen e Romanovsky [Chen and Romanovsky 2008] descrevem uma arquitetura de serviços web que utiliza um mediador para prover confiabilidade na integração de serviços web. O mediador é responsável por aplicar as políticas, monitorar os serviços web e gerar metadados representando seus atributos de confiabilidade, como tempo de resposta e taxa de falha. Em particular, o uso de recuperação de erros por avanço através de mecanismos de tratamento de exceções pode ser bastante efetiva para obtenção de tolerância a falhas para aplicações baseadas em serviços Web que não podem ser recuperadas por retrocesso, onde tratadores específicos do domínio da aplicação podem ser mais eficientes e flexíveis. Estudos mais aprofundados são necessários para avaliar a efetividade dessas propostas usando-se estudos de casos mais complexos e reais.

3.2 Testes baseados em modelos

O principal desafio para a realização de testes de conformidade no contexto de SOA e Web Services é estabelecer quais técnicas são mais promissoras e adequadas.

Testes de conformidade visam determinar se uma implementação está de acordo com o que foi especificado. Determinar se um serviço faz o que é esperado é o primeiro passo para se ter uma aplicação que funcione, mas deve-se testar também se a composição dos serviços também atende ao que é esperado para a aplicação. Testes de conformidade baseiam-se em modelos da especificação, que podem representar os dados ou o comportamento de um sistema. Esses modelos podem ser usados de duas formas diferentes. Como testes ativos, o modelo serve de base para a geração de testes. Como testes passivos, o modelo é usado para determinar se um traço de execução, obtido ao se monitorar a execução da implementação durante certo tempo, é conforme ao que foi especificado.

Uma abordagem de testes ativos propõe que os testes de conformidade sejam derivados a partir da modelagem de serviços na forma de Máquinas Finitas de Estado (MEF) [Ambrosio *et al* 2005]. A abordagem se apóia no uso de ferramentas de geração

de testes a partir deste tipo de modelo [Martins, Sabião and Ambrosio 1999] e foi proposta para um contexto em que os serviços são estáticos [Ambrosio *et al* 2006]. Uma extensão desta abordagem para serviços dinâmicos seria uma contribuição valiosa para a área de testes de conformidade para aplicações SOA / WS.

Em um primeiro momento, deve-se determinar a adequação para a representação de serviços que evoluem dinamicamente. No contexto de WSs, são utilizados padrões de especificação, como WSDL, para descrever a interface de serviços e BPEL (linguagem baseada em XML, padronizada pela OASIS), para a composição dos mesmos. É interessante, portanto, determinar como mapear tais padrões para o tipo de modelagem proposto. Algumas soluções já foram propostas, baseadas no uso de máquinas finitas de estado estendidas temporizadas [Kazhamiakin *et al* 2006], ou de RT-UML [Cambronero *et al* 2006], uma extensão da UML para representação de sistemas de tempo real. Tais soluções precisam ser consolidadas, aprofundadas e validadas.

3.3 Testes de robustez

Em aplicações baseadas em WS, pode acontecer de um serviço ser incorporado dinamicamente à aplicação, estando em uso em outros contextos. O desafio nesse caso é como aplicar os testes de robustez sem prejudicar sua operação nesses outros contextos. Deve-se evitar ou minimizar a interferência da ferramenta de teste sobre o sistema. Vamos considerar que os testes de robustez têm por objetivo verificar a capacidade de um sistema, ou componente, de funcionar de forma *aceitável* (ao invés de correta) em presença de falhas ou de condições ambientais estressantes [Castanet and Waeselynk 2003].

Existem vários tipos de testes de robustez, mas consideramos como as mais promissoras duas abordagens sistemáticas: uma baseia-se em métodos formais, outra, no uso de injeção de falhas. A abordagem baseada em métodos formais inspira-se nas técnicas e modelos utilizados nos testes de conformidade para a realização dos testes de robustez [Fernandez, Mounier and Pachon 2005, Khorcheff *et al* 2006]. Na abordagem baseada em injeção de falhas, a robustez é validada em presença de falhas introduzidas de forma deliberada (e, se possível, controlada) sobre o sistema.

Existem diversas técnicas para injeção de falhas [Hsueh *et al* 1997], entre elas a injeção de falhas por software ou SWIFI (*Software-Implemented Fault Injection*). Ferramentas SWIFI podem injetar uma vasta gama de falhas simuladas desde falhas em memória, registradores, barramentos, recursos de armazenamento, comunicação [Drebes *et al* 2005] até erros de software. O uso de SWIFI para testes de robustez já foi objeto de vários trabalhos, inclusive a metodologia Ballista [Kropp *et al* 1998] e a metodologia CoFI (*Conformance and Fault Injection testing*) [Ambrosio *et al* 2005] que combina injeção de talhas e métodos formais [Avresky *et al* 1992, Suri and Sinha 1998] numa abordagem híbrida.

As abordagens mencionadas precisam ser adaptadas ou aprofundadas para realização de testes de robustez em projetos de SOA ou Web Service. Um cuidado especial deve ser tomado em relação ao controle e minimização da interferência das ferramentas de teste por injeção de falhas na funcionalidade do sistema.

3.4 Arquitetura de testes

O padrão ISO 9646 (*Conformance Testing Methodology and Framework*) define, entre outros, arquiteturas para os testes de sistemas de comunicação. Um elemento desta arquitetura é o *testador*, i.e., a entidade de testes, o qual interage com a implementação em testes através de pontos de controle e observação. A partir da ISO 9646 diversas arquiteturas foram definidas, de acordo com a localização dos *testadores* em relação à implementação em testes, ou seja, se residem na mesma máquina ou são remotos.

Para vencer o desafio de testes de sistemas SOA/Web Services, a arquitetura de testes deve levar em conta não somente o fato de que os testes deverão ser realizados remotamente, mas também que estes deverão causar um mínimo possível de interferência na execução do serviço em testes. Este é um problema crucial especialmente nos testes por injeção de falhas, pois a inserção do injetor sempre causa interferência, por mínima que seja. Um desafio é também o fato de que nem sempre se pode inserir o injetor no ambiente de execução do serviço em teste. Outro desafio a ser considerado é o fato de que a implementação em testes pode evoluir durante a execução dos testes.

Mais um desafio, não menos importante, é a definição de uma linguagem para especificar os testes, de forma que a arquitetura de testes seja o mais independente possível dos serviços e das aplicações em teste. As linguagens para especificação de testes são uma preocupação antiga. Por exemplo, o padrão ISO 9646 já citado contém a definição de uma linguagem para especificação dos testes de protocolos, denominada TTCN (*Tree and Tabular Combined Notation*). A OMG também se preocupou em definir um perfil para a especificação de testes (*UML Test Profile*), que é baseado no TTCN. Uma possibilidade a ser investigada para vencer os desafios de arquitetura de teste é o uso de padrões utilizados no contexto de Web Services para descrever os testes.

3.5 Avaliação de risco de uso de um serviço

Na composição de Web Services, o desenvolvedor de uma aplicação pode eventualmente ter a escolha entre diversos serviços semelhantes ou equivalentes que atendam suas necessidades. Para guiar a seleção destes serviços, uma análise de riscos poderia ser realizada para se determinar qual o risco de utilização de um ou outro serviço visando à maior robustez da aplicação desenvolvida.

Alguns trabalhos propõem uma abordagem para avaliação de risco em sistemas baseados em componentes utilizando-se modelos estatísticos e injeção de falhas [Moraes *et al* 2007]. Um desafio considerável é adaptar essa abordagem, ou outras semelhantes, para a avaliação do risco de serviços que evoluem ao longo do tempo, e não são estáticos, como é o caso de componentes.

3.6 Avaliação do desempenho

Aplicações baseadas em serviços Web se apóiam em uma infra-estrutura distribuída composta por várias camadas de software e uma ampla variedade de componentes de hardware. Tradicionalmente, a avaliação de desempenho consiste em obter medidas quantitativas que permitem ao projetista analisar eventos aleatórios (chegada, serviço, perda de pacotes, falhas e reparações relacionadas ao hardware, software). A avaliação pode ser efetuada baseada essencialmente em duas abordagens complementares: (i)

modelagem analítica e (ii) medições. Medições provêem informação para caracterizar o comportamento atual e passado dos sistemas já existentes. Por outro lado, a modelagem permite guiar o desenvolvimento durante a fase de projeto, a fase operacional e prever futuras demandas.

Um dos desafios na avaliação de desempenho consiste em criar abordagens analíticas capazes de capturar o comportamento probabilístico desta infra-estrutura extremamente complexa. Além disso, é preciso que uma nova abordagem seja i) computacionalmente viável, ou seja, há solução para o modelo, ii) tenha seus parâmetros calibrados a partir de valores obtidos de aplicações reais, e iii) seja validada comparando resultados analíticos com experimentais.

Um caminho que merece ser explorado para avaliar serviços Web distribuídos em larga escala é utilizar-se dos conceitos de web *crawler*. Trata-se de um robô que percorre a estrutura de sítios Web [Marcondes *et al* 2008]. Este conceito poderia ser usado para obter um diagnóstico detalhado de um grande número de serviços Web que compõem aplicações típicas. A partir dos resultados obtidos no percurso exploratório, relatorios estatísticos seriam elaborados incluindo medidas quantitativas como disponibilidade do serviço, capacidade do sítio, largura de banda disponível, probabilidade de perda e tempo de resposta. Com base em resultados experimentais poderiam ser construídos modelos analíticos hierárquicos, permitindo descrever a aplicação alvo em diversos níveis de abstração, com sub-modelos associados a cada nível.

4 A busca por estudos de caso significativos

Novos paradigmas, diretrizes de projeto, técnicas e ferramentas de desenvolvimento de software sendo propostas precisam passar por uma fase de avaliação experimental. Essa avaliação permite refinar novas metodologias e validá-las para casos representativos de uma gama de aplicações usuais no mundo real. A escolha desses casos representativos é vital para comprovação dos benefícios de novos paradigmas e sua aplicabilidade. Mas essa escolha sozinha já representa um grande desafio. Casos elementares, fáceis de serem construídos, geralmente mostram benefícios e aplicabilidade pontuais, sem endereçar à vasta gama de problemas que surgem com a interação de várias características funcionais e não funcionais. Casos completos e úteis, típicos das necessidades atuais e precursores de necessidades futuras são extremamente difíceis de serem encontrados.

A ausência da disponibilidade de casos representativos para a demonstração de novos paradigmas e técnicas prejudica tanto o seu desenvolvimento e sua consolidação como sua popularização entre desenvolvedores. Soluções possíveis para cada uma das seis questões levantadas na seção anterior precisam ser avaliadas através de estudos de casos representativos, que aliem utilidade, relevância e necessidade de atender critérios de qualidade típicos de sistemas onivalentes para aplicações de missão crítica com alto grau de fidedignidade.

O Brasil possui, em universidades e centros de pesquisa, um grupo ativo de pesquisadores ligados às áreas de engenharia de software, teste e tolerância a falhas. Esse grupo, distribuído geograficamente, mas conectado graças a interesses comuns e oportunidades proporcionadas pelos eventos da SBC, identifica aplicações de interesse de instituições nacionais que poderiam proporcionar excelentes estudos de caso. Um

exemplo é o INPE, uma instituição disposta a prover interessantes casos de estudo e se beneficiar dos desenvolvimentos na área de desenvolvimento orientado a serviços. Um desafio lançado à comunidade acadêmica é o Sistema de Apoio à Construção de Plano de Vôo (SAPV), uma aplicação real desenvolvida pelo INPE como parte dos serviços oferecidos pelo Segmento Solo em aplicações espaciais. O objetivo do sistema é permitir que pesquisadores da área espacial possam colaborar remotamente na construção do Plano de Vôo do Satélite através da Internet, elaborando telecomandos para a carga útil do satélite. O sistema permite a edição, organização e agendamento de telecomandos, via páginas Web, os quais seriam transmitidos nas visadas do satélite pelas estações solo e executados a bordo mais tarde [Mattiello-Francisco *et al* 2006].

A transformação dessa aplicação Web em um serviço proporcionaria um excelente caso de estudo. Essa aplicação poderia servir como piloto para no futuro se ter o desenvolvimento de um sistema de Monitoração e Controle de satélites e outros artefatos espaciais, totalmente baseados em serviços, conforme vem sendo recomendado pelo grupo de trabalho em Monitoração e Controle do Consultative Committee for Space Data Systems [CCSDS 2006] que reúne as principais agências espaciais.

5 Trabalhos relacionados

Padrões como o XML, SOAP, UDDI, WSDL já mencionados abordam os conceitos básicos de serviços interoperáveis, mas para WS-SOA outras normas devem ser adicionadas e aprovadas. Alguns exemplos são [Ort 2005]: (i) WS-Security, um padrão da OASIS visando à segurança de mensagens SOAP e que prevê integridade e confidencialidade. (ii) SAML (*Security Markup Assertion Language*), outro padrão OASIS baseado em XML para troca de informações de segurança. (iii) WS-BPEL (*Web Services Business Process Execution Language*), ou simplesmente BPEL, uma linguagem baseada em XML, que é usada para coordenar serviços Web em um único processo empresarial. BPEL é uma combinação de linguagens definidas pela IBM e Microsoft que agora faz parte do esforço de padronização OASIS.

Em sites de fornecedores e de grupos de interesse, pode ser encontrada uma grande quantidade de trabalhos relativos às técnicas de modelagem para SOA, alguns desses trabalhos parecem promissores para solução de alguns dos desafios mencionados aqui. O SCA - *Service Component Architecture* é um conjunto de especificações que descrevem um modelo para a construção de aplicações e sistemas. SCA amplia e complementa abordagens anteriores para a execução de serviços e baseia-se em padrões abertos, como os serviços Web. Seu objetivo é fornecer um conjunto de especificações e uma referência de implementação de um modelo de programação SOA, e tem um forte envolvimento dos principais fornecedores entre eles a IBM [IBM 2008]. A OMG está atualmente definindo um padrão *UML profile* para SOA chamado UPMS (*UML Profile and Metamodel for Services*).

Um grande número de iniciativas proprietárias existe nesta área, como a solução específica da IBM (*Enterprise Architecture tool*). Na França, encontramos uma iniciativa aberta, chamada Praxeme, que engloba várias empresas de consultoria, e é apoiada por grandes organizações como companhias de seguros e alguns setores públicos. Esta iniciativa visa proporcionar uma metodologia aberta cobrindo as necessidades da arquitetura (*Enterprise Architecture*) para o desenvolvimento de software. Ainda assim, esse desafio não pode ser considerado vencido. É necessário que uma solução completa

seja finalizada e difundida. Esta solução deveria apoiar a modelagem da arquitetura lógica, ser baseada em UML2 e centrar-se em SOA. Além disso, ela deveria ser capaz de suportar metas de teste e definição da especificação.

O crescente uso de Web Services faz com que seja necessário garantir que o seu comportamento esteja correto. A prática comum é gerar testes de unidade e testes de integração com ferramentas, tais como SOAPUI [SOAPUI 2008], uma aplicação *open source* para inspeção, invocação, desenvolvimento e teste funcional/carga/cumprimento de Web Services sobre protocolo HTTP que são baseados em abordagens empíricas.

A natureza dos Web Services requer, primeiro, uma validação e teste individual de todos os serviços isolados e um novo teste quando já estão integrados. Na indústria já existem alguns instrumentos para realizar esses testes. Quando são considerados serviços Web complexos, também temos de lidar com a sua composição (chamada orquestração). Atualmente, na literatura, encontram-se vários trabalhos que abordam a especificação formal com BPEL e modelos para os serviços Web. Wombacher [Wombacher *et al* 2004] propõe uma transformação de BPEL num autômato determinístico de estado finito. Este formalismo não captura aspectos de tempo de algumas atividades do BPEL e não considera as variáveis BPEL. *Extended Finite State Automaton* [Nakajima 2005] foi proposto para lidar com variáveis, mas também não são consideradas restrições de tempo. Kazhamiakin [Kazhamiakin *et al* 2006] propõe um formalismo que leva em conta restrições de tempo, o WSTTS. No entanto, este formalismo usa apenas marcações de tempo, mas não variáveis. Alguns trabalhos usam redes de Petri para especificar WS [Hinz*et al* 2005, Yang *et al* 2005]. Este formalismo não está bem adaptado para testes de composição de WS, pois resulta numa descrição global que carece de estrutura tornando impossível diferenciar componentes do serviço. RT-UML também foi usada para modelar aspectos de tempo real da coreografia de WS [Cambronero *et al* 2006], mas não considerou a orquestração nem a interoperabilidade de serviços e componentes. Outro trabalho digno de menção também propôs uma metodologia baseada em BPEL [Farahbod *et al* 2004] onde a formalização da semântica da linguagem BPEL é necessária para eliminar as ambigüidades e tornar especificações operacionais abstratas executáveis. Aqui um modelo baseado em *Abstract State Machine* (ASM) é utilizado, mas ao contrário de modelos baseados em *Finite State Machines* (FSM), não são diretamente executáveis e requerem traduções não tão evidentes, tornando mais difícil o desenvolvimento de ferramentas de testes. Também a ASM precisa igualmente ser completada com restrições de tempo e variáveis. Outra abordagem baseada no BPEL faz a sua tradução para Promela, que é a linguagem de entrada do modelo SPIN (*SPIN Model Checker*), permitindo modelos de verificação, bem como geração de testes [Fanjul, Tuya and de La Riva 2006]. Essas abordagens devem ainda ser desenvolvidas e implementadas como ferramentas onde escalabilidade e usabilidade possam ser avaliadas.

As abordagens mencionadas até agora são para testes funcionais relacionadas ao testes de conformidade do WS e sua composição. Outra abordagem, mencionada anteriormente, é baseada na injeção de falhas [Hsueh *et al* 1997]. Um método promissor de teste, apresentado por Looker e Xu [Looker and Xu 2003], visa avaliar a dependabilidade das interfaces de um WS, e principalmente o middleware de WS e os protocolos RPC baseado em SOAP. Aqui falhas são injetadas na fronteira entre a aplicação e a pilha de protocolo. O modelo considera apenas falhas de comunicação, tais como a

supressão, duplicação, reordenação ou corrupção de mensagens. Embora essas falhas sejam pouco significativas para um middleware que execute sobre uma LAN confiável são essenciais para o teste de WS, uma vez que operam em WANs, muito mais sujeitas a falhas do que LANs. Outro trabalho interessante usa agentes móveis para testar WSs candidatos, bem como a interação entre os WS selecionados [Zhang 2004]. Os agentes móveis são usados para testar os WS em seus respectivos sítios (*sites*). Esses agentes também implementam assertivas para permitir a detecção de erros durante a execução. Falhas afetam a interface entre os WSs, especificado no WSDL. O agente transporta dados normais, tal como definido pela especificação WSDL, bem como dados corrompidos, ao WS que está sendo testado. Para ajudar na geração de dados de falhas, o sistema é especificado na forma de redes de Petri coloridas. Neste sentido, é possível não só testar um WS de forma isolada, mas também integrados com outros WSs candidatos.

6 Enfrentando os desafios

Muitos são os desafios identificados pela comunidade acadêmica. Entretanto, a elaboração de propostas de soluções sem implementação e validação apontam caminhos, mas não solucionam problemas reais. O desenvolvimento tecnológico de qualidade, contemplando disponibilidade, correção, segurança, escalabilidade, persistência e ubiqüidade, mencionado no quinto grande desafio da SBC, vai além da elaboração de propostas de modelos, métodos e arquiteturas, exigindo uma fase consolidada de implementação, experimentação e validação junto de parcerias que vivenciam o problema real e são possíveis usuários das soluções.

Uma forma de atacar o quinto grande desafio proposto pela SBC e mais especificamente, os desafios relacionados ao desenvolvimento orientado a serviços discutidos neste artigo, é reunir competências de pesquisadores de diferentes áreas da computação. Esses pesquisadores, especialistas em diferentes etapas do desenvolvimento de aplicações que vão desde o projeto até a sua validação, devem unir esforços para integração e validação de suas propostas aproveitando suas experiências de sucesso e fracasso atualmente dispersas. A discussão de avanços e técnicas promissoras de cada área e formas viáveis de integração é apenas um passo inicial. Desenvolvimento, validação, medições sobre aplicações que atendam necessidades reais e que não estejam suficientemente atendidas por paradigmas e plataformas atualmente disponíveis é o próximo passo essencial para que novas propostas possam ser efetivamente aceitas e aplicadas por desenvolvedores de software.

Um olhar menos distante da academia sobre as necessidades reais de desenvolvedores de sistemas dinâmicos na Web e uma maior aproximação com desenvolvedores e usuários de sistemas de missão crítica abriria portas de comunicação vantajosas para os dois mundos.

7 Conclusões

Este trabalho teve como objetivo levantar os desafios a serem enfrentados para o desenvolvimento de aplicações onivalentes no que se refere à garantia de disponibilidade, correção, segurança e principalmente robustez. Os desafios mais significativos foram apresentados e discutidos, assim como foram sugeridos caminhos viáveis de investigação para o seu enfrentamento. Estamos convencidos que SOA e Web Sevices representam um caminho viável e promissor para o desenvolvimento de aplicações que

exigem escalabilidade, confiabilidade e segurança e que forneçam sobre a Internet todo o tipo de serviço, do mais simples sensor e atuador ao mais completo cluster. A par do desenvolvimento de recursos para o projeto e implementação eficiente de aplicações orientadas a serviços, identificamos como essencial o desenvolvimento de métodos de validação que permitem fornecer garantias mínimas de qualidade, correção e fidedignidade a essas aplicações.

Agradecimentos: agradecemos a Fátima Mattiello por todo esforço que tem despendido para estreitar os laços de cooperação entre universidades e o INPE.

8 Referências bibliográficas

Ambrosio, A.M., Martins, E., Vijaykumar, L.N. and Carvalho, S.V. (2005) "A Methodology for Designing Fault Injection Experiments as an Addition to Communication Systems Conformance Testing", In: Proc. of the First Workshop on Dependable Software – Tools and Methods in the IEEE Conference on Dependable System and Network, 28 June – 1 July 2005, Yokohama, Japan.

Ambrosio, A.M., Martins, E., Vijaykumar, L.N. and Carvalho, S.V. (2006) "A Conformance Testing Process for Space Applications Software Services", Journal of Aerospace Computing, Information, and Communication (JACIC) /AIAA, vol. 3, no. 4, pp. 146-158, April.

Avresky, D., Arlat, J., Laprie, J.-C. and Crouzet, Y. (1992) "Fault injection for the formal testing of fault tolerance", In: Proc. Of the 22nd Int. Symp. on Fault-Tolerant Computing (FTCS-22), IEEE, Boston, MA, pp. 345-354, July.

Boulton, C. (2005) "IDC: Web Services Consumption to Hit Stride", July 14. Available at http://www.internetnews.com/dev-news/article.php/3520271, último acesso em 03/2008.

CCSDS (2006) - Consultative Committee for Space Data System - 520.0-G-2, "Mission Operations Services Concept", Green Book, Issue 2, August, disponível em http://public.ccsds.org/publications/GreenBooks.aspx, último acesso em 03/2008.

Cambronero, M.E., Diaz, G., Pardo, J.J., Valero, V. and Pelayo, F. L. (2006), "RT-UML for modeling Real-Time Web Services", SCW Journal, Volume 0, pp. 131-139, IEEE Computer Society, Los Alamitos, CA, USA.

Castanet, R. and Waeselynk, H. (2003), "Techniques avancées de test de systèmes complexes: test de robustesse", relatório, CNRS-AS23.

Chen, Y. and Romanovsky, A. (2008), "Improving dependability of web services integration", In: ITPro, January/February, pp. 20-26, IEEE Computer Society Press.

Drebes, R. J., Silva, G. J., Trindade, J. M. F. and Weber, T. S. (2006), "A Kernel-based Communication Fault Injector for Dependability Testing of Distributed Systems.", Hardware and Software, Verification and Testing, First International Haifa Verification Conference, LNCS, Berlin- Heidelberg, Springer-Verlag, v. 3875, pp. 177-190.

Fanjul, J. G., Tuya, J. and de La Riva, C. (2006), "Generating Test Cases Specifications for BPEL Compositions of Web Services Using SPIN", In: Proc. of the International Workshop on Web Services Modeling and Testing - WS-MaTe.

Farahbod, R., Glasser, U. and Vajihollahi, M. (2004), "Specification and Validation of the Business Process Execution Language for Web Services", In: Abstract State Machines, pp. 78-94.

Fernandez, J. C., Mounier, L. and Pachon, C. (2005), "A Model-based Approach for robustness testing". In: Proc. of TESTCOM 2005, The 17th IFIP Int. Conf. on Testing of Communicating Systems, Montreal (Québec), Canada, Springer Verlag Heidelberg ISSN: 0302-9743 LNCS.

Gorbenko, A., Kharchenko, V.S. and Romanovsky A.B. (2007), "On composing dependable Web Services using undependable web components", Int. Journal on Simulation and Process Modelling, Vol. 3, Issue 1/2, pp. 45-54, Interscience Publishers.

Hinz, S., Schmidt, K. and Stahl, C. (2005), "Transforming BPEL to Petri Nets", In: Business Process Management, pp. 220-235.

Hsueh, M.C., Tsai, T. K. and Iyer, R. K. (1997), "Fault Injection Techniques and Tools", In: IEEE Software, April, pp. 75-82.

IBM (2008), "Service Component Architecture", disponível em http://www.ibm.com/-developerworks/library/specification/ws-sca/, último acesso 03/2008.

IEEE Std Glossary of Software Engineering Terminology - 610.12, (1990).

Kazhamiakin, R., Pandya, P. and Pistore, M. (2006), "Timed Modelling and Analysis in Web Service Compositions", In: Ares Journal, Volume 0, pp. 840-846, IEEE Computer Society, Los Alamitos, CA, USA.

Khorcheff, F. S., Berrada, I., Rollet, A. and Castanet, R. (2006), "Automated Robustness Testing for Reactive Systems: Application to Communicating Protocols", In: Proc. of the 6th International Worhshop on Innovative Internet Community Systems - I2CS, LNCS, Neuchâtel, Switzerland, June 26-28.

Kropp, N.P., Koopman, P.J. and Siewiorek, D.P. (1998), "Automated Robustness Testing of Off-the-Shelf Software Components", In: Proc. of the 28th. Fault Tolerant Computing Symposium, June 23-25, Munich, Germany.

Looker, N. and Xu, J. (2003), "Assessing the Dependability of SOAP RPC-Based Web Services by Fault Injection", In: Proc. of the Ninth IEEE Int. Workshop on Object-Oriented Real-Time Dependable Systems - WORDS'03, Guadalajara, Mexico.

Moraes, R., Durães, J., Barbosa, R., Martins, E. and Madeira, H. (2007), "Experimental Risk Assessment and Comparison Using Software Fault Injection", In: Proc. of the 37th Int Conf on Dependable Systems and Networks - DSN07, Edimburg, UK.

Marcondes, C., Martinello, M., Schwartz, R., Santos, R., Gerla, M. and Sanadidi, M.Y. (2008), "PathCrawler: Automatic Harvesting Web Infra-Structure", IEEE/IFIP Network Operations and Management Symposium (NOMS), Salvador.

Martins, E., Sabião, S.B. and Ambrosio, A. M. (1999), "ConData: a Tool for Automating Specification-based Test Case Generation for Communication Systems. Software", Quality Journal, v.8, n. 4, pp. 303-319.

Mattiello-Francisco, M.F., Sakugawa, B.M and Yano E.T. (2006), “Safety in a Web-based Satellite Flight Supporting System”, SPACEOPS-2006, 19-23 june, Rome, Italy.

Moreira, A. F., Cota, E., Ribeiro, L., Gaspary, L., Carro, L. , Ritt, M. and Weber, T. S. (2007), “Em Direção a um Modelo para Desenvolvimento de Sistemas Computacionais de Qualidade para Aplicações Onivalentes”. In: XXXIV Seminário Integrado de Software e Hardware, Rio de Janeiro, SBC, v. 1, pp. 2262-2276.

Nakajima, S. (2005), “Lightweight formal analysis of Web Service flows”. Journal Progress in Informatics, Volume 2, pp. 57-76.

Natis, Y.V. (2003), “Service-Oriented Architecture Scenario”, Gartner Research Note ID Number: AV-19-6751, 16 April, disponível em http://www.gartner.com/it/-products/research/, último acesso 03/2008.

Ort, E. (2005), “Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools”, april, disponível em http://java.sun.com/developer/-technicalArticles/WebServices/soa2/, último acesso 03/2008.

Sociedade Brasileira de Computação (2006), “Grandes Desafios da Computação”, disponível em http://www.sbc.org.br, último acesso 03/2008.

Silva Jr., M.C., Guerra, P. A. C., Rubira, C.M.F. (2003), "A Java Component Model for Evolving Software Systems", In: Proc. of 18th IEEE Int. Symposium on Automated Software Engineering, Vol. 1, pp. 327-330, Montreal, Canadá.

SoapUI (2008), “Web Services Testing Tool”, disponível em http://www.soapui.org/, último acesso 03/2008.

Suri, N. and Sinha, P. (1998), “On the Use of Formal Techniques for Validation”, Proc. of FTCS-28, pp. 390-399.

Tartanoglu, F., Issarny, V., Romanovsky, A., Levy, N. (2003), “Dependability in the Web Services Architecture”, In: Architecting Dependable Systems, de Lemos, R., Gacek, C., Romanovsky, A., Editors, pp. 90-109, LNCS 2677, Springer.

Wombacher, A., Fankhauser, P. and Neuhold, E. (2004), “Transforming BPEL into Annotated Deterministic Finite State Automata for Service Discovery”. In: Proc. of the IEEE International Conference on Web Services - ICWS'04, Washington-DC, USA .

Xu,J., Randell,B., Romanosky,A., Rubira-Calsavara,C.M.F., Stroud,R.J. and Wu, Z. (1995), “Fault Tolerance in Concurrent Object-oriented Software through Coordinated Error Recovery”, In: Proc. of 25th Int. Symposium on Fault-tolerant Computing - FTCS-25, Pasadena, California, USA.

Yang, Y. P., Tan, Q. P., Yu, J. S., Liu, F. (2005), “Transformation BPEL to CP-Nets for Verifying Web Services Composition”. In: Proc. of the Int. Conference on Next Generation Web Services Practices - NWESP '05.

Zhang, J. (2004), “An approach to facilitate reliability testing of Web Services components”, In: Proc. of the 15th. Int. Symposium on Software Reliability Engineering - ISSRE, Saint-Malo, France, November.