ANDRÉ PEREIRA PIAZZA

UMA ABORDAGEM PARA INTEROPERABILIDADE ENTRE PLATAFORMAS HETEROGÊNEAS DE SERVIÇOS WEB PARA REDES COLABORATIVAS DE ORGANIZAÇÕES

FLORIANÓPOLIS
2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

UMA ABORDAGEM PARA INTEROPERABILIDADE ENTRE PLATAFORMAS HETEROGÊNEAS DE SERVIÇOS WEB PARA REDES COLABORATIVAS DE ORGANIZAÇÕES

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

ANDRÉ PEREIRA PIAZZA

Florianópolis, Maio de 2007.

UMA ABORDAGEM PARA INTEROPERABILIDADE ENTRE PLATAFORMAS HETEROGÊNEAS DE SERVIÇOS WEB PARA REDES COLABORATIVAS DE ORGANIZAÇÕES

André Pereira Piazza

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

_____
Prof. Ricardo José Rabelo, Dr.
Orientador

_____
Profª. Kátia Campos de Almeida, Dra.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____
Prof. Ricardo José Rabelo, Dr.
Presidente

_____
Prof. Carlos Barros Montez, Dr.

_____
Prof. Frank Augusto Siqueira, Dr.

_____
Prof. Mario Dantas, Dr.

Aos meus amados pais, Almerinda e João Batista,
pelo amor e dedicação.

Agradecimentos

Gostaria de iniciar esse trabalho agradecendo a Deus, por ter me dado a dádiva da vida, e por sempre ter me aberto as portas e me guiado pelos caminhos que escolhi traçar.

Aos meus pais, Almerinda Pereira Piazza e João Batista Piazza, por tudo que fizeram por mim em toda a minha vida, pelo amor, carinho, dedicação e educação. Sem vocês eu não teria chegado até aqui. Amo muito vocês!

Ao professor Ricardo José Rabelo, por ter acreditado em mim e em meu trabalho, aceitando ser meu orientador, sempre me auxiliando com direcionamentos e sugestões em minhas dúvidas e questionamentos.

À minha namorada Gláucia Cavalcanti, que sempre me apoiou e me agüentou nos meus momentos de stress, dando palavras de conforto e motivação, mesmo de longe!

A todas as pessoas que tive a oportunidade de conhecer no decorrer do mestrado, colegas do LCMI, colegas do Grupo GSIGMA, professores do DAS e funcionários da PGEEL.

Aos membros da banca cujas sugestões e críticas contribuíram para o engrandecimento deste trabalho.

Ao Projeto ECOLEAD pelo suporte financeiro.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

UMA ABORDAGEM PARA INTEROPERABILIDADE ENTRE PLATAFORMAS HETEROGÊNEAS DE SERVIÇOS WEB PARA REDES COLABORATIVAS DE ORGANIZAÇÕES

André Pereira Piazza

Maio, 2007

Orientador: Prof. Ricardo José Rabelo, Dr.
Área de Concentração: Automação e Sistemas.
Palavras-chave: Interoperabilidade, Serviços Web, Invocação Dinâmica, Colaboração, Integração, Redes Colaborativas de Organizações
Número de Páginas: 150

RESUMO: Cada vez mais as organizações têm enfrentado um ambiente de intensa concorrência e mudanças de mercado. Para se manterem competitivas e melhorar a colaboração entre parceiros e clientes, as organizações vêm participando de redes colaborativas mais complexas e menos rígidas, como, por exemplo, cadeias de suprimento e organizações virtuais. A Tecnologia da Informação e Comunicação (TIC) vem sendo cada vez mais utilizada como forma de suportar essa colaboração. Porém, isso cria um cenário extremamente heterogêneo, onde organizações diferentes e distribuídas utilizam diversos sistemas de TIC, providos por fabricantes variados e desenvolvidos em diversas plataformas e tecnologias. Portanto, é preciso que sejam oferecidos meios de alcançar uma maior interoperabilidade e colaboração entre as organizações, de maneira padronizada, aberta e dinâmica. Para tal, faz-se necessário utilizar uma abordagem que siga um modelo descentralizado e de baixo acoplamento, permitindo que as organizações possam participar de maneira transparente e interoperável nos processos colaborativos. O paradigma de orientação a serviços, que pode ser implementado através da tecnologia e padrões dos serviços Web, vem sendo largamente utilizado para atingir tal interoperabilidade entre organizações e seus sistemas de TIC. Apesar dos serviços Web serem amparados por padrões, quando as organizações desejam fazer seus serviços, que normalmente são desenvolvidos em diferentes plataformas, linguagens e sistemas operacionais, se comunicarem plenamente entre si, diversos problemas de interoperabilidade surgem na prática. Esse trabalho apresenta uma pesquisa acerca da forma como se aumentar a interoperabilidade e a colaboração entre organizações através de serviços Web, propondo uma abordagem para ocultar a complexidade tecnológica e atingir uma maior interoperabilidade, independentemente da plataforma computacional, das tecnologias e linguagens de implementação, e da localização geográfica do serviço.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

AN APPROACH FOR INTEROPERABILITY AMONG HETEROGENEOUS WEB SERVICES PLATFORMS FOR COLLABORATIVE NETWORKED ORGANIZATIONS

André Pereira Piazza

May, 2007

Advisor: Prof. Ricardo José Rabelo, Dr.
Area of Concentration: Automation and Systems.
Keywords: Interoperability, Web Services, Dynamic Invocation, Collaboration, Integration, Collaborative Networked Organizations
Number of Pages: 150

ABSTRACT: Nowadays, organizations have to deal with a more competitive and changing market. To remain competitive and improve collaboration relationships between partners and clients, organizations are joining more complex but flexible collaborative networks, like, for example, supply chains and virtual organizations. The Information and Communication Technology (ICT) is been used to support this collaboration. However, this is generating a very heterogeneous scenario, where different and distributed organizations use several ICT systems, provided by various vendors, using different technologies and frameworks. Therefore, it is imperative to provide ways to reach a better interoperability and collaboration between organizations, in a standard, open and dynamic way. To accomplish that, it is necessary to apply an approach that follows a decentralized and loosely-coupled model, allowing that the organizations can participate in a transparent and interoperable way in the collaborative processes. The service-oriented paradigm, that can be implemented using the Web services technology and standards, are been broadly adopted to achieve such interoperability between organizations and ICT systems. Despite the Web services been supported by standards, when organizations try to use each others services, that are normally developed in different frameworks, languages and operating systems, several interoperability problems arise in practice. This work presents a deep research in the use of Web services in a way to improve the interoperability and collaboration between organizations, and proposes an approach to achieve this interoperability in a seamless way, independently of the computational platform and framework, operating system, programming languages and geographic location of the services.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1
Introdução

1.1 Cenário geral

A Tecnologia da Informação e Comunicação (TIC) vem tendo um papel de extrema importância em diversas áreas de negócios. Organizações e empresas em um constante ambiente de competição vêm utilizando a TIC para melhorar seus processos de negócios de forma a reduzir custos e riscos, e aumentar a produtividade, qualidade e dinamicidade dos processos. Empresas devem integrar e permitir que seus sistemas e processos de negócio se tornem visíveis e acessíveis globalmente, para dessa forma se tornarem mais hábeis a colaborar, lidarem mais facilmente com mudanças do mercado, e consequentemente sobreviverem no mercado.

Uma nova geração de TICs vem surgindo, como por exemplo as Arquiteturas Orientadas a Serviço, Computação em Grade, Redes P2P, com a promessa de possibilitar um acesso mais direto e ágil a informações e processos de negócio, e compartilhando um número maior de recursos computacionais, permitindo então uma maior colaboração entre diferentes organizações.

Porém, com a velocidade em que novas tecnologias são lançadas e utilizadas, surge o problema de sistemas e infra-estruturas que acabam se tornando obsoletas, por não suportarem os atuais processos de negócio. Os processos de negócios das organizações têm sido alterados com cada vez mais freqüência, resultado tanto de melhorias e adaptações internas, como de requisitos de clientes e do reflexo de inovações. Isto faz com que constantes alterações e adaptações tenham que ser feitas nos sistemas existentes/legados, o que muitas vezes é extremamente difícil, custoso, moroso e, em alguns casos, até mesmo inviável, dependendo da qualidade e da flexibilidade do projeto original do sistema, assim como das TICs usadas na sua implementação (BISBAL *et al.*, 1999).

Muitos destes sistemas legados são desenvolvidos internamente na organização, e lidam com atividades e processos de negócio muito específicos, e de maneira independente e *ad-hoc*. Com isso, surge o problema da falta de integração e interoperabilidade entre

esses diferentes sistemas heterogêneos, resultando em uma dificuldade (técnica, custos e tempo) de automação dos processos de negócio entre diferentes setores dentro da empresa. Grande parte dessa integração acaba sendo feita de maneira manual, através de adaptações e interfaceamentos entre diferentes sistemas.

Visando minimizar este problema, um esforço grande vem sendo feito no sentido de se criar novos conceitos e tecnologias para suportar essa necessidade de integração e colaboração entre diferentes sistemas, de maneira padronizada e universal.

No que toca a integração entre sistemas intra-organizacionais, abordagens iniciais para se resolver este problema basearam-se na de EAI[1], que possibilita a automatização e integração desses diferentes sistemas e processos de negócio da organização em uma única infra-estrutura centralizada. Os sistemas de supervisão, ERP[2], SCM[3], CRM[4], por exemplo, antes aplicações independentes, tendem a se tornar uma única infra-estrutura de serviços de alto nível, interoperável e "sem costuras"[5]. Através dessa integração, é possível que os diferentes processos de negócios suportados pela organização e desenvolvidos em sistemas independentes possam se comunicar de maneira transparente, entre diferentes níveis dentro de uma organização (PUSCHMANN *et al.*, 2004).

Porém, soluções baseadas em EAI apresentam diversas desvantagens. Os custos de implantação de soluções de EAI são altos e, além disso, são soluções majoritariamente monolíticas, pouco modulares, e, assim, pouco flexíveis para ágeis mudanças nos processos e novas integrações de sistemas (CAPE-CLEAR, 2005). Um outro problema é que as soluções de EAI são voltadas para a integração intra-organizacional, não se adequando a questões de interoperabilidade inter-organizacional.

Já entre sistemas inter-organizacionais a necessidade da integração e interoperabilidade se dá em decorrência da crescente globalização e do aumento da necessidade de alianças e terceirizações, onde diferentes organizações tendem a trabalhar

[1] EAI (*Enterprise Application Integration*) é o processo de interconexão entre diferentes sistemas dentro de uma organização.
[2] ERP (*Enteprise Resource Planning*) são sistemas computacionais que integram os processos de negócios de diversas áreas de uma organização.
[3] SCM (*Supply Chain Management*) é o sistema pelo qual a organização provê os serviços e produtos aos seus clientes e parceiros.
[4] CRM (*Customer Relashionship Management*) são sistemas que automatizam informações de contato com os clientes, bem como armazenam suas atividades e interações com a organização.
[5] Do inglês, *seamless*. No restante do documento, será utilizada sempre em português: sem costuras.

de maneira interligada e organizada, colaborando umas com as outras. As organizações têm criado alianças estratégicas de negócio cada vez mais complexas e mais dinâmicas, de forma a aumentar a colaboração.

Um exemplo disso são as Redes Colaborativas de Organizações (RCO), que são constituídas de várias entidades - p.ex.: organizações ou pessoas - que são autônomas, geograficamente distribuídas e heterogêneas, em termos de arquitetura operacional, cultura e objetivos (CAMARINHA-MATOS *et al.*, 2005). Essas entidades colaboram entre si de forma a atingir um objetivo comum ou compatível com as suas necessidades. Como exemplo de uma manifestação de RCO pode-se citar uma Organização Virtual (OV). Uma OV é uma agregação lógica, dinâmica e temporária de organizações autônomas que cooperam umas com as outras, de modo que possam atender a uma determinada oportunidade de negócios, ou lidar com um objetivo específico. Em uma OV, todas as operações são executadas através da troca coordenada de habilidades, recursos e informações, totalmente habilitadas pela rede de computadores (RABELO *et al.*, 2004).

A formação, operação e dissolução de RCO pode ser muito dinâmica ou ser relativamente estável; sua criação pode ter objetivo de curto prazo (*demand-oriented*) ou de longo prazo (*product-oriented*); sua composição pode ser com parceiros fixos ou pode ser muito volátil; e sua estrutura de coordenação pode ser desde o tipo estrela (com uma empresa dominante) ou de um tipo descentralizado onde todos têm os mesmos direitos.

No entanto, independentemente disso, o estabelecimento de alianças visa prover às organizações maior flexibilidade de reação e adaptação e lhes permitir estender suas capacidades e seu alcance no mercado. Para tal, precisam igualmente estender o tipo de colaboração que tradicionalmente têm, se baseando na troca de informações e serviços entre seus sistemas de TIC, de maneira interoperável e com muito mais agilidade.

Esta interoperação entre sistemas e serviços das OVs, se torna possível com a utilização da Internet como meio de comunicação, e com o auxílio de uma infra-estrutura de TIC que seja transparente, de fácil utilização e de baixo custo (CAMARINHA-MATOS, 2003). Essa infra-estrutura de TIC serve como uma camada intermediária, um *middleware* de suporte a colaboração (COULOURIS *et al.*, 2005), devendo ser implementada com a utilização de tecnologias e padrões abertos, de forma que permita a interoperabilidade entre os diversos componentes - organizações membros da OV - independentemente do sistema computacional utilizado por cada um deles (RABELO *et al.*, 2007).

No lado técnico, tentativas anteriores de integração, como o EDI (*Electronic Data Interchange*) e técnicas similares, são muito caras e altamente inflexíveis (FEUERLICHT, 2006). As organizações precisam implementar estruturas de TIC que suportem interações com outras organizações, porém, reutilizando a arquitetura já existente – sistemas e aplicações legadas.

O paradigma de orientação a serviços, que é uma evolução das abordagens baseadas em objetos e componentes, tem o potencial para tratar os problemas apresentados acima, principalmente no provimento de integração, reusabilidade e interoperabilidade entre os atuais ambientes computacionais heterogêneos (LI *et al.*, 2006). A Arquitetura Orientada a Serviços, ou SOA (*Service Oriented Architecture*), é um estilo arquitetural que permite a interação entre diversas aplicações, independentemente da plataforma, através da utilização de serviços genéricos e padronizados (GOLD-BERNSTEIN *et al.*, 2006). Dentre as vantagens da SOA estão a possibilidade de reuso de aplicações de software já existentes, e a flexibilidade na interação entre diferentes atores envolvidos nos processos, que podem pertencer a diferentes organizações (FEUERLICHT, 2006).

Numa filosofia SOA, uma aplicação não é mais desenvolvida como um software monolítico, como um "pacote", mas sim como uma *composição* de serviços, interligados de acordo com a lógica do processo. Existem três tendências principais no desenvolvimento orientado a serviços, que são os serviços Web, serviços de Grade e serviços P2P (LI *et al.*, 2006). Os serviços Web, que são o foco dessa dissertação, utilizam o conceito de serviços do SOA, e implementam seus serviços na Web, utilizando padrões abertos.

Um serviço Web é uma aplicação modular, independente, e que possui uma interface aberta, padrão e baseada na Internet (OASIS-UDDI, 2001). Essa interface pode ser descrita, publicada, descoberta e invocada utilizando protocolos padrão baseados na linguagem XML, como o WSDL, SOAP e UDDI (AUSTIN *et al.*, 2004). Na Arquitetura Orientada a Serviços, um serviço Web pode ser definido como um componente autônomo e reutilizável para ser empregado em uma função ou processo de negócios. No decorrer dessa dissertação, os serviços Web e os principais protocolos e padrões que os constitui serão apresentados em mais detalhes.

Um aspecto da tecnologia de serviços Web é que toda a funcionalidade ou processo de negócio que uma organização deseja disponibilizar será exposto na forma de um

serviço, isto é, terá uma interface publicada em um formato padrão, que pode ter suas operações invocadas por diferentes clientes (FEUERLICHT, 2006). Nesse cenário, diversos serviços Web são definidos, desenvolvidos e gerenciados por diferentes organizações autônomas e independentes, tornando-os uma tecnologia de baixo acoplamento[6], e por utilizar padrões abertos, também permite uma grande interoperabilidade entre diferentes organizações.

A existência na Internet de serviços baseados no padrão XML, interoperando entre diferentes aplicações de software e processos de negócios de empresas, e sendo executados em uma variedade de plataformas e servidores de aplicação, gerou um grande interesse por parte dos fabricantes de software e da indústria em geral (BOOTH *et al.*, 2004). Os fabricantes vêm adicionando cada vez mais funções e novos padrões em suas plataformas, habilitando uma comunicação mais robusta entre sistemas. Já as organizações vêm adotando os serviços Web para expor suas aplicações de negócios externamente, permitindo a colaboração entre membros da organização e também com parceiros de negócio de outras organizações, criando uma cadeia virtual de parceiros de negócios que colaboram entre si.

1.2 Motivação e descrição do problema

A forma de interação entre organizações vem evoluindo rapidamente, ficando claro que as empresas precisam adaptar cada vez mais agilmente a mudanças e rapidamente aproveitar novas oportunidades de negócios que surgem. A comunicação sem obstáculos, sem costuras, transparente e dinâmica entre sistemas computacionais de diversas organizações facilita a colaboração, tornando a execução dos processos de negócio mais ágeis.

A interoperabilidade dos sistemas é uma área de grande interesse, devido ao fato da constante necessidade de integração tanto de sistemas totalmente novos, como dos sistemas legados já existentes, em particular no contexto de redes colaborativas voltadas para negócios. Os sistemas e aplicações das organizações devem ser interoperáveis, de forma

[6] Baixo acoplamento, do termo em inglês *loose coupling*, descreve uma relação entre dois ou mais sistemas que podem interagir entre si, porém de forma independente, i.e., uma mudança em um sistema não interfere ou impede o funcionamento de outro sistema.

que seja possível fazer negócios externos à organização e concretizar as RCOs (FISCHER *et al.*, 2005b).

Os serviços Web foram criados com o intuito de atingir essa transparência e interoperabilidade sem costuras, utilizando a Web como ambiente de execução de seus processos de negócio em forma de serviços, se tornando acessíveis e invocáveis por qualquer organização (ANDRADE ALMEIDA *et al.*, 2003).

Essa visão da Web como um grande ambiente de execução, contendo diversos serviços que utilizam padrões abertos para publicar a descrição de sua interface, podendo assim ser invocados por qualquer tipo de plataforma, de diferentes fabricantes, é sem dúvida uma grande evolução, comparados com os problemas de tecnologias anteriores, baseadas em objetos e componentes (LI *et al.*, 2006).

Isso traz muitas vantagens. Porém, alguns problemas e limitações também surgem. Apesar de estarem trabalhando com tecnologias padrão, quando as organizações tentam interoperar com serviços de diferentes plataformas, sistemas de informação ou processos de negócios, diversos problemas de interoperabilidade podem surgir (EGYEDI, 2006). A especificação dos padrões de serviços Web é muito vasta, fazendo com que fabricantes implementem diferentes versões do mesmo padrão, impedindo que a interoperação entre implementações de diferentes fabricantes (ANDRADE ALMEIDA *et al.*, 2003), criando assim “silos tecnológicos” isolados, e fazendo com que projetos de integração se tornem mais complexos e custosos (LI *et al.*, 2006).

Um outro problema é a forma como os serviços são descobertos e invocados. Em uma organização que realiza negócios em tempo real, é importante se identificar rapidamente o serviço necessário para um determinado processo de negócio, e invocá-lo de maneira automática e dinâmica. Porém, diversos problemas, como a necessidade de se conhecer previamente a localização do serviço, a criação estática dos *proxies* no cliente para a posterior invocação, entre outros, ainda fazem com que os serviços Web tenham diversos problemas de interoperabilidade.

Esse problema é ainda mais crítico em uma OV, onde os membros, por definição, não se conhecem à priori e assim, nenhum deles têm uma integração entre si, previamente definida e estabelecida. A forma de interação entre esses membros é de alta dinamicidade, pois membros entram e saem de uma OV de acordo com a necessidade do negócio. Os serviços disponibilizados pelos membros, conseqüentemente, também são temporários,

fazendo com que haja a necessidade de se criar uma forma de invocar os serviços automaticamente e de maneira dinâmica, sem necessidade de conhecimento prévio do serviço.

Portanto, em vista da problemática apresentada acima, este trabalho visa responder a seguinte pergunta de pesquisa: Como se pode garantir uma maior interoperabilidade sem costuras entre diferentes organizações pertencentes a uma rede colaborativa, de forma que estas possam atuar com mais agilidade e ao mesmo tempo seus sistemas e serviços possam atuar com mais flexibilidade quando da mudança de processos?

1.3 Objetivos da pesquisa

A utilização da TIC como forma de melhorar a agilidade dos processos de negócio contribui efetivamente para o aumento da produtividade e dinamicidade de uma organização. Cada vez mais as organizações têm exposto seus processos de negócio além de suas fronteiras, se tornando hábeis a colaborar, lidar mais facilmente com mudanças do mercado e, conseqüentemente, sobreviver no mercado.

Organizações em um cenário de RCO, aberto e de larga escala, estão em constante interação com diferentes tipos de sistemas e serviços que são providos por seus membros. Por diferente, quer-se dizer que a linguagem de programação, a plataforma de disponibilização e o sistema operacional são heterogêneos, pois cada organização tem um tipo particular e específico de metodologia, plataformas e arquiteturas para desenvolver, disponibilizar e publicar seus serviços.

Isso faz com que a interoperabilidade em serviços Web se torne um grande campo de pesquisa, onde técnicas, ferramentas e recomendações devem ser criadas, com o objetivo de suportar de forma plena e sem costuras a interoperabilidade em todos os níveis das organizações pertencentes a uma rede colaborativa (PAGANELLI *et al.*, 2005).

1.3.1 Objetivo geral

O objetivo geral desse trabalho é o de prover um modelo de interoperabilidade que oculte a complexidade tecnológica e automatize a localização e invocação de diferentes serviços Web, disponibilizados por diversas organizações, em diferentes plataformas,

sistemas operacionais e linguagens de programação, de maneira interoperável, transparente e dinâmica, tornando-a simples e ágil para o consumidor de serviços.

Na abordagem do modelo, a invocação de serviços é feita sem a necessidade de intervenções manuais ou de construções de interfaces caso-a-caso por parte do consumidor de serviços. Dessa forma, é possível se alcançar um aumento na colaboração entre diferentes organizações, de forma que estas possam funcionar mais eficientemente, contribuindo para o aumento da agilidade dos processos de negócio hoje impostos pelo mercado.

1.3.2 Objetivos específicos

- Estudo detalhado dos padrões e especificações dos serviços Web, principalmente no que toca a interoperabilidade desses padrões.
- Avaliação de diversas plataformas de disponibilização de serviços Web e suas linguagens de implementação, desenvolvimento de diferentes serviços em cada uma destas plataformas, e avaliação da interoperabilidade de cada um deles.
- Concepção de um modelo que oculte a complexidade tecnológica e automatiza a localização e invocação de serviços Web disponibilizados em diferentes arquiteturas, plataformas e redes, sem a necessidade de intervenções manuais ou de construções de interfaces caso-a-caso.
- Desenvolvimento de uma biblioteca, a ser utilizada pelos consumidores e provedores de serviços Web, com a finalidade de publicar, localizar e invocar serviços Web em tempo de execução, sob-demanda, de maneira interoperável, dinâmica e sem o uso de *stubs*, ou *stubless*[7], provendo alto nível de transparência e flexibilidade ao consumidor, que faça uso de um único código para invocar diferentes serviços, em diferentes plataformas, utilizando diferentes protocolos.

A hipótese-base de pesquisa assenta-se na observação de que:

[7] No restante desta dissertação, será utilizado o termo *stubless*, que se refere à não geração prévia e uso de *stubs* na invocação de serviços Web.

- Há um número significativo de empresas que tem vários de seus sistemas sendo desenvolvidos numa filosofia de serviços e fazendo uso de plataformas SOA para negócio eletrônico (B2B);
- Embora ainda predominante nas grandes empresas, uma série de *roadmaps* vem claramente apontando que as pequenas e médias empresas passarão a usar sistemas baseados em serviços. Isto se dá por pressão a se integrarem aos sistemas das grandes empresas, pela maior flexibilidade que isso dá aos processos, pela redução dos custos de aquisição e manutenção de software, e por buscarem modelos de negócios mais viáveis.

Estas duas hipóteses permitem o desenvolvimento de uma abordagem de interoperação, que suporte a composição e execução de serviços que tenham sido desenvolvidos e disponibilizados em diferentes plataformas SOA.

1.4 Justificativa

Atualmente, as organizações precisam colaborar, de forma a sobreviver no mercado. Organizações, de grande e pequeno porte, devem ser focadas na inovação e no aproveitamento de informações, através da união com outras organizações, de forma a responder de maneira rápida e flexível às mudanças de mercado, e até mesmo na criação de novos mercados (LI *et al.*, 2006).

Diversos padrões e tecnologias foram desenvolvidas na área de colaboração inter-organizacional, com o intuito de suportar uma melhor interoperabilidade entre organizações. Os serviços Web se encaixam nesse grupo, e têm sido amplamente adotados por diversas organizações, justamente por sua característica de prover uma maior interoperabilidade entre diferentes plataformas, graças ao uso de padrões abertos.

Porém, o desenvolvimento da indústria acerca dos serviços Web tem sido mais focado na criação de ferramentas para projetar, criar e disponibilizar os serviços Web. Infelizmente, pouco tem sido feito em relação ao consumo, ou *invocação* desses serviços, voltados para a aplicação cliente.

A forma tradicional de invocação dos serviços Web, independentemente da plataforma na qual ele está disponibilizado, é feita de maneira manual, onde a localização do serviço é descoberta em tempo de projeto e os *stubs* são criados a priori, para então

invocar o serviço. Manter o código dessas aplicações de invocação pode se tornar uma atividade trabalhosa e que consome muito tempo, devido ao fato de que os serviços são dinâmicos, e podem sofrer mudanças, ou até mesmo deixar de existir.

A necessidade de suportar diferentes tipos de protocolos de acessos a serviços é essencial em um ambiente heterogêneo como uma RCO. Em uma oportunidade de negócios entre organizações, é preciso que as partes envolvidas sejam capazes de interoperar de maneira transparente, automatizada e sem costuras, onde seja possível a invocação de diferentes serviços Web de maneira ágil, simples e dinâmica, independente dos sistemas e arquiteturas em que foram desenvolvidos e disponibilizados. O foco principal desse trabalho é dado na interoperabilidade entre diferentes plataformas de serviços Web, e na invocação dinâmica e *stubless* desses serviços disponibilizados, independente da forma como esse serviço é disponibilizado e do protocolo de acesso com o qual esse serviço é acessado.

Uma necessidade crescente por interoperabilidade também vem surgindo dentro das PMEs (Pequenas e Médias Empresas). As PMEs são empresas com possibilidades limitadas de terem soluções mais avançadas de TIC, devido à falta de cultura organizacional, recursos financeiros, de pessoal qualificado e de conhecimento. Porém, a sua necessidade por negócios se iguala a de grandes organizações. Para CAMARINHA-MATOS et al. (2002), as infra-estruturas que têm surgido possuem uma configuração complexa, onde a customização dos processos é difícil de ser gerenciada pelas PMEs, assim como a manutenção das TICs envolvidas é cara.

Um estudo recente reportou que existem aproximadamente 25 milhões de PMEs nos Estados Unidos, que retêm 53% dos empregos e 51% da movimentação de vendas. Na Europa, existem 18 milhões de PMEs, retendo 66% dos empregos e 55% da movimentação de vendas (BOSE *et al.*, 2006). Já no Brasil, as PMEs foram responsáveis por 77% dos números de exportação em 2005 (ZACHARIAS *et al.*, 2007). Com base nessas estatísticas, é possível visualizar a necessidade de soluções de interoperabilidade voltadas para PMEs, de forma que estas possam colaborar com organizações de grande porte, e se manterem competitivas no mercado.

Portanto, o que se propõe é uma abordagem que visa ocultar a complexidade da aplicação cliente (por exemplo, uma organização) ao se invocar um serviço Web dentro de uma federação de serviços, e ao mesmo tempo aumentar a interoperabilidade entre

organizações e a reusabilidade dos serviços de forma transparente e automatizada. Com isso, aumenta-se a colaboração inter-organizacional, através de uma invocação flexível e dinâmica.

1.5 Metodologia

De acordo com (KAZDIN, 2003 apud MARCZYK *et al.*, 2005), metodologia se refere aos princípios, procedimentos e práticas que dirigem uma pesquisa, e abrange todos os processos, ou seja, o planejamento e desenvolvimento, as conclusões feitas, e por fim a disseminação do conhecimento adquirido. A metodologia é a base que define todos os processos de pesquisas científicas.

Em uma pesquisa, a investigação da realidade é feita sob os mais diversos aspectos e dimensões, através de diferentes níveis de aprofundamento e enfoques específicos, de acordo com o objeto de pesquisa. Então, torna-se necessária a classificação da pesquisa, baseado em alguns critérios. Essa seção tem como objetivo enquadrar o presente trabalho quanto à sua natureza, aos objetivos gerais, à abordagem do problema, e às modalidades de pesquisa (SILVA *et al.*, 2005).

1.5.1 Classificação quanto à Natureza da Pesquisa

A natureza da pesquisa está ligada às motivações do pesquisador e com o destino, ou a apropriação social dos frutos de seu trabalho (SCHWARTZMAN, 1979). A pesquisa, quanto a sua natureza, pode ser classificada como: acadêmica, aplicada e básica.

A pesquisa aplicada é aquela que tem um resultado prático visível em termos econômicos ou de outra utilidade que não seja o próprio conhecimento. Objetiva gerar conhecimentos para aplicação prática e que são dirigidos à solução de problemas específicos.

Essa dissertação deve ser classificada como **aplicada**, pois tem o intuito de discutir teoricamente a interoperabilidade dos serviços Web e a colaboração entre organizações, e então apresentar uma abordagem que visa aumentar essa colaboração através do aumento da dinamicidade e interoperabilidade entre diferentes serviços Web. Os resultados serão aplicados em ambientes práticos e reais, de forma que se facilite a invocação de serviços por parte dos clientes dos serviços.

1.5.2 Classificação quanto aos Objetivos Gerais

É usual a classificação de pesquisas com base em seus objetivos gerais, podendo ser classificadas em três grandes grupos: exploratória, descritiva e explicativa (LAKATOS *et al.*, 1985).

A pesquisa exploratória é vista como o primeiro passo de todo o trabalho científico, e tem como finalidade proporcionar maior entendimento sobre determinado assunto, e contribuir para sua solução. Essa dissertação deve então ser classificada como **exploratória**, pois tem como objetivo se aprofundar no assunto da interoperabilidade em serviços Web, de forma a garantir que estes serviços Web, disponibilizados em diferentes plataformas distribuídas, sejam interoperáveis entre si, utilizando os padrões disponíveis, visando suportar um maior nível de colaboração entre organizações, através da partilha de recursos/serviços entre estas.

1.5.3 Classificação quanto à Abordagem do Problema

Quanto à abordagem do problema, é possível se classificar uma pesquisa como quantitativa e qualitativa.

Na pesquisa quantitativa, os resultados podem ser quantificados, ou seja, traduzir em números as opiniões, amostras e informações, e depois analisá-las e classificá-las utilizando técnicas estatísticas. Já na pesquisa qualitativa, não se faz uso de métodos e técnicas estatísticas, é uma pesquisa mais descritiva, onde o processo e seu significado são os focos principais da abordagem (SILVA *et al.*, 2005).

Essa dissertação possui as características de uma pesquisa **qualitativa**, pois foca mais na análise do ambiente como fonte de dados, não fazendo uso de técnicas estatísticas. É uma pesquisa que foca principalmente no processo e na sua descrição detalhada, ou seja, na interoperabilidade entre diferentes serviços Web.

1.5.4 Modalidades de pesquisa utilizadas

Quanto às modalidades, uma pesquisa pode ser classificada como: bibliográfica, documental, ex-post-facto, baseada em levantamento, com *survey*, estudo de caso, pesquisa participante, pesquisa ação, experimental e etnográfica (MATOS *et al.*, 2002, SILVA *et al.*, 2005).

Abaixo, são apresentadas e detalhadas as modalidades que foram utilizadas neste trabalho:

- **Bibliográfica**: Após a identificação e definição do tema do trabalho, foi realizada uma pesquisa bibliográfica elaborada a partir de material já publicado, como consulta a artigos e periódicos nacionais e internacionais, livros, e materiais disponibilizados na Internet.
- **Estudo de caso**: O presente trabalho pode ser classificando também como um estudo de caso, não no que se toca ao estudo de caso aplicado à organizações, mas sim envolvendo um estudo profundo sobre a implementação de serviços Web em diferentes plataformas, caso-a-caso, procurando descobrir o que há de mais essencial e característico, e identificando os problemas de interoperabilidade que possam existir. Visa também apresentar uma perspectiva global do objeto de estudo, do ponto de vista do investigador.

1.5.5 Procedimentos para a elaboração do trabalho

Esta seção apresenta a metodologia que foi utilizada para a realização do trabalho.

1. Revisão do estado da arte
    1.1. Revisão bibliográfica da literatura, por meio de artigos, livros, anais e páginas da Web, de forma a fundamentar todos os assuntos tratados nessa dissertação, como por exemplo, serviços Web e suas plataformas, interoperabilidade e integração de serviços.
    1.2. Revisão de projetos de pesquisa e trabalhos científicos correlatos, onde foram analisados artigos em anais e revistas.
    1.3. Revisão de servidores de aplicação e plataformas para SOA/B2B de diferentes fabricantes.
2. Identificação de requisitos de interoperabilidade entre plataformas SOA/B2B.
3. Concepção de uma abordagem que oculta a complexidade tecnológica e automatiza a localização e invocação de serviços Web disponibilizados em diferentes arquiteturas, plataformas e redes, sem a necessidade de intervenções manuais ou de construções de interfaces caso-a-caso.

4. Elaboração do protótipo

    4.1. Instalação e configuração de diferentes plataformas para serviços Web, que foram selecionados baseando-se em sua utilização e aceitação no mercado.

    4.2. Implementação de diferentes serviços Web nas plataformas citadas, previamente instaladas.

    4.3. Implementação dos clientes dos serviços em todas as plataformas de serviços Web, de maneira distribuída (redes diferentes), fazendo com que os clientes invoquem os serviços de outras plataformas, monitorando e analisando as interações entre os serviços, suas funcionalidades e limitações, de forma a tentar localizar problemas relacionados com a interoperabilidade.

    4.4. Análise dos resultados do passo 4.3, e criação de uma ferramenta de invocação dinâmica e *stubless*, de forma que a localização e invocação de serviços Web seja simples e transparente para o usuário.

5. Validação e análise dos resultados alcançados no trabalho

    5.1. Invocação de diferentes serviços que foram implementados nas diferentes plataformas instaladas no passo 4, utilizando a ferramenta descrita acima

    5.2. Analise e avaliação dos possíveis problemas de interoperabilidade, através do uso das ferramentas e metodologias propostas por organizações voltadas para a interoperabilidade de serviços Web.

6. Escrita da dissertação e descrição dos resultados

1.5.6 Meios para desenvolvimento da dissertação

Esta dissertação foi desenvolvida no laboratório GSIGMA – Grupo de Sistemas Inteligentes de Manufatura – onde seu autor teve acesso a uma infra-estrutura adequada, incluindo recursos de hardware, software e bibliografia necessários para o desenvolvimento de um bom trabalho.

O trabalho foi desenvolvido no âmbito de um projeto internacional, o que significa que as contribuições devem satisfazer não apenas os requisitos necessários para obtenção do grau de mestre, mas também apresentar resultados relevantes a este projeto.

O projeto ECOLEAD – *European Collaborative Networked Organizations Leadership Initiative* – é um Projeto Integrado do 6º Programa Quadro da Comissão Européia, iniciado em abril de 2004 com duração de 48 meses, que conta com a participação de 20 organizações distribuídas em 14 países, sendo 18 instituições Européias e 2 da América Latina. O projeto visa criar fundamentos teóricos e mecanismos de tecnologia da informação para auxiliar no estabelecimento de uma avançada sociedade colaborativa entre organizações. A idéia principal é causar impacto substancial na forma de materialização de redes colaborativas através de uma abordagem holística compreensível (ECOLEAD, 2007).

1.6 Adequação às linhas de pesquisa do curso

O trabalho descrito nesta dissertação está inserido no contexto da Área de Concentração em Automação e Sistemas do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina. Este trabalho está perfeitamente integrado com os demais trabalhos de pesquisa sobre Integração de Sistemas e Organizações Virtuais, e com as atividades do Curso de Pós-Graduação em Engenharia Elétrica desta Universidade.

1.7 Organização do documento

Esse trabalho está organizado da seguinte maneira:

O capítulo 1 descreveu a problemática geral e de enquadramento do trabalho executado, assim como sua metodologia, objetivos e justificativa.

No capítulo 2, serão apresentados os trabalhos relacionados, e também serão introduzidos os conceitos iniciais sobre Arquitetura Orientada a Serviços, tecnologia de serviços Web e os principais padrões que os suportam.

No capítulo 3, será apresentada uma investigação mais detalhada sobre interoperabilidade, seus problemas, soluções já propostas, se focando dentro do âmbito dos serviços Web.

Após apresentados os conceitos necessários, o capítulo 4 detalhando o modelo conceitual da abordagem proposta, apresentando uma arquitetura independente de tecnologia, e em seguida aplicando essa arquitetura dentro do modelo de serviços Web, porém, sem apresentar nenhum detalhe de implementação.

No capítulo 5 serão apresentados os detalhes da implementação, baseando-se no modelo conceitual apresentado no capítulo anterior. São apresentados também a validação da abordagem proposta e os resultados obtidos.

O capítulo 6 conclui o estudo com as principais contribuições da pesquisa e trabalhos futuros.

CAPÍTULO 2
Revisão Bibliográfica

Neste capítulo são abordados conceitos importantes para o trabalho descrito nesta dissertação. As seções 2.1 e 2.2 apresentam os conceitos da Arquitetura Orientada a Serviços, descrevendo mais detalhadamente a tecnologia de serviços Web e seus principais padrões e metodologias de desenvolvimento, sempre levando em conta questões de interoperabilidade. A seção 2.3 apresenta alguns projetos de pesquisa e trabalhos acadêmicos relacionados com o tema de pesquisa deste trabalho.

2.1 Arquitetura Orientada a Serviços

Atualmente, uma abordagem que vem adquirindo grande suporte na indústria é baseada na visão de que as soluções e processos de negócio das organizações são vistos como serviços, que são conectados por contratos bem definidos que definem suas interfaces. Essa abordagem é conhecida como Arquitetura Orientada a Serviços, ou SOA (*Service-Oriented Architecture*) (BOOTH *et al.*, 2004).

A SOA surgiu de uma evolução das abordagens baseadas em componentes e objetos, com a promessa de superar suas deficiências. Porém uma variedade e diversidade de implementações e interpretações da SOA causam controvérsias e ceticismo entre desenvolvedores e arquitetos de sistemas. Existe uma grande quantidade de padrões, que em muitas vezes sobrepõem a aplicabilidade de outros padrões, fazendo com que se torne mais difícil de se compreender e utilizar o potencial da SOA (BERRE *et al.*, 2004). Essa seção apresenta uma visão geral dos principais padrões e tecnologias da SOA.

2.1.1 Definição de SOA

Existem diversas definições para o que constitui uma SOA. De acordo com BOOTH *et al.* (2004), SOA pode ser definido como "um conjunto de componentes que podem ser invocados, e que a descrição de suas interfaces pode ser publicada e descoberta". No entanto, essa definição é muito técnica, pois descreve a arquitetura apenas

em termos de implementação, e não o sentido no qual o termo "arquitetura" é normalmente usado, ou seja, para se descrever um estilo ou um conjunto de práticas bem definidas.

Uma outra definição de SOA, de uma perspectiva mais arquitetural e de negócios, é dada por IBM-SOFTWARE-GROUP (2005), que define que SOA é uma arquitetura para aplicações que utiliza as aplicações de negócios do cotidiano e as divide em funções e processos de negócio individuais, compartilhados e reutilizáveis, chamados de serviços. A SOA permite a construção, disponibilização e integração desses serviços, independentemente das aplicações e plataformas computacionais nas quais estas irão executar.

2.1.2 Motivações da SOA

Atualmente, organizações têm gasto muito tempo e recursos tentando atingir uma integração rápida e flexível de seus sistemas de TIC dentre todos os elementos de seu ciclo de negócios. Através do uso da SOA é possível definir uma abordagem arquitetural que permita essa integração dos sistemas de maneira rápida e flexível (DUDLEY *et al.*, 2007). As motivações para tal incluem:

- Aumento na agilidade em que organizações e empresas implementam novos produtos e processos, alteram processos existentes, ou compõem diferentes processos, formando um novo processo.
- Redução dos custos de integração e implementação dos sistemas, tanto intra quanto inter-organizacionais.
- Possibilidade de reuso de sistemas e aplicações legadas.
- Alcançar uma melhor utilização da TIC e do retorno do investimento.

A SOA prescreve uma abordagem arquitetural, se baseando em serviços, para que se possa atingir essa integração de forma rápida e flexível.

2.1.3 Descrição da arquitetura

A Arquitetura Orientada a Serviços (SOA) suporta diversas atividades básicas, que são descritas abaixo (TSALGATIDOU *et al.*, 2002):

- Criação do serviço;

- Descrição do serviço;
- Publicação do serviço em repositórios de forma que usuários em potencial do serviço possam localizá-lo;
- Descoberta do serviço pelos usuários em potencial
- Invocação do serviço
- Remoção do serviço do repositório caso não esteja mais disponível ou não seja necessário

Existem outras atividades que fazem parte da SOA, como por exemplo, composição, gerenciamento, monitoramento e segurança.

A arquitetura da SOA consiste fundamentalmente de três componentes, ou entidades, que são o Provedor do Serviço, um Diretório de Serviços, e um Consumidor do Serviço, que serão descritos abaixo, e apresentados na Figura 1(BOOTH *et al.*, 2004):

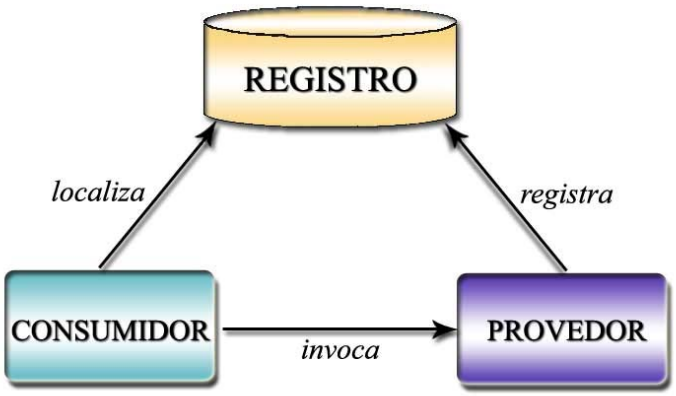


Figura 1: Modelo conceitual de uma SOA

- **Provedor do serviço**: O provedor do serviço é a parte que provê as aplicações de software e processos de negócio, em forma de serviços. Os provedores de serviço publicam, removem e atualizam seus serviços de forma que fiquem disponíveis aos consumidores dos serviços. O provedor é o dono do serviço e mantêm sua lógica de negócios e implementação invisíveis para o consumidor.
- **Consumidor do serviço**: O consumidor do serviço é a parte que possui uma necessidade que pode ser satisfeita por um serviço disponível. O consumidor do serviço pode ser um usuário, uma aplicação ou até mesmo um outro serviço,

que localiza e invoca serviços de acordo com sua necessidade. A localização é feita pelo consumidor através de um diretório de serviços, que ao localizar, invoca o serviço diretamente do provedor.

- **Diretório de serviços**: O diretório de serviços provê um repositório onde é possível se localizar descrições e interfaces de serviços. Os provedores de serviço publicam seus serviços no diretório, e os consumidores de serviço localizam e obtêm as informações necessárias para a invocação do serviço.

2.1.4 O que é um serviço

De acordo com O'BRIEN *et al.* (2005), um serviço é uma implementação bem definida de uma funcionalidade de negócios, com uma interface publicada que pode ser descoberta por consumidores de serviços para a criação de diferentes aplicações e processos de negócio, através de um modelo de comunicação de baixo acoplamento baseado na troca de mensagens.

Os serviços podem ter diversas formas, as quais são relacionadas com a tecnologia em que são implementados. Um tipo específico de serviço é o serviço Web, que é definido por BOOTH *et al.* (2004) como "uma aplicação de software identificada por uma URI, cujas interfaces são capazes de serem definidas, descritas e descobertas por artefatos XML, e que suporta interações diretas com outras aplicações de software que utilizam trocas de mensagens baseadas no XML através de protocolos da Internet".

As duas descrições de serviços, a primeira focando mais no estilo arquitetural da SOA e a segunda nos serviços Web, mostram um conjunto de características relacionadas com a natureza e aplicabilidade dos serviços. Abaixo, serão apresentadas algumas características dos serviços, de forma mais detalhada (BROWN *et al.*, 2005, O'BRIEN *et al.*, 2005):

- **Granularidade**: As operações sobre os serviços são freqüentemente implementadas para atingir uma maior funcionalidade e operar em maiores quantidades de dados, em comparação ao projeto baseado em componentes.
- **Definição baseada em Interface**: A definição da interface dos serviços é feita de forma que uma entidade de software implemente e exponha uma parte fundamental de sua definição de forma padrão. Uma aplicação ou sistema cuja

estrutura é baseada na SOA é criada como uma coleção de serviços que são interligados utilizando a descrição das suas interfaces.

- **Descoberta**: Os serviços precisam ser localizados, tanto no desenvolvimento quanto em tempo de execução, e não somente por uma identidade única, mas também pela identidade da interface e também pelo tipo e nome do serviço. Isso é feito através de um servidor de diretório, ou no caso do endereço do serviço já ser conhecido durante a implementação, o endereço pode ser inserido manualmente dentro do software do usuário (*hard-coded*).
- **Instanciação única**: Cada serviço é uma instância única, que está executando constantemente, com o qual um número de clientes se comunicam.
- **Baixo acoplamento**: A SOA é uma arquitetura de baixo acoplamento, pois separa estritamente a interface da implementação. Outra característica é que a descoberta e invocação do serviço em tempo de execução reduzem a dependência entre os provedores e consumidores de serviço.
- **Reusabilidade**: A lógica da aplicação encapsulada como um serviço pode ser reutilizada em diferentes aplicações. Na SOA, os componentes se tornam reutilizáveis, pois sua interface é definida de forma padrão, fazendo com que um serviço em C# possa interoperar com uma aplicação em Java, por exemplo.
- **Abstração da lógica**: A única parte do serviço que é visível para os usuários é o que é exposto através da descrição do serviço em sua interface. A lógica da aplicação abaixo do serviço é invisível e irrelevante aos consumidores desse serviço.
- **Autonomia**: Os serviços são autônomos, ou seja, o serviço é independente de outros serviços para sua execução.
- **Não gerenciam estados**: Os serviços não gerenciam informações de estado (*stateless*), pois isso impediria sua habilidade de baixo acoplamento.
- **Capacidade de Composição**: Serviços podem compor outros serviços, promovendo a reusabilidade dos serviços em diversas aplicações, processos e lógicas distintas.

Dos princípios descritos acima, o baixo acoplamento, autonomia e abstração podem ser considerados os princípios básicos que formam os fundamentos da SOA. Cada serviço implementa uma função de negócios[8], e qualquer aplicação que precisar executar essa função utiliza esse serviço.

É possível também que uma aplicação seja criada através da composição e coordenação das atividades entre diversos serviços, que são necessários para se executar um dado processo de negócios. Conforme novas oportunidades de negócio vêm surgindo, os desenvolvedores podem implementar ou adicionar novos serviços que estejam disponíveis.

Um Sistema Orientado a Serviços é um sistema baseado nos princípios da SOA (O'BRIEN *et al.*, 2005). Um serviço Web é a forma mais utilizada de se implementar a SOA, apesar de existirem outras formas de implementação, como os serviços em Grade e os serviços P2P. Porém, a tecnologia de serviços Web vem adquirindo uma grande aceitação da indústria como sendo um padrão para se construir aplicações baseadas na arquitetura SOA, sendo a base de estudo dessa dissertação. Na seção abaixo, serão descritos os serviços Web, bem como as tecnologias e padrões que os constituem.

2.2 Serviços Web

O termo *serviços Web* é muito utilizado atualmente, porém nem sempre com o mesmo significado. No entanto os conceitos e tecnologias fundamentais são em grande parte independentes de como serão interpretadas (ALONSO *et al.*, 2004).

De acordo com OASIS-UDDI (2001), "os serviços Web são aplicações modulares, independentes e auto-descritas que podem ser publicadas, localizadas e invocadas na Web utilizando padrões baseados na linguagem XML". Os serviços são descritos através de uma interface com um formato processável por máquina, e são invocados utilizando troca de mensagens sobre um protocolo de transporte, como, por exemplo, o HTTP ou SMTP. Esses serviços podem ser aplicações ou processos de negócios que são encapsulados e disponibilizados na Web, através de uma URI única, que é publicada em um repositório de serviços.

[8] Função de negócios também referenciada como "processo de negócios" (ou *business process*).

Um serviço Web é uma instância do paradigma de orientação a serviços, descrito na seção anterior, onde as entidades e operações especificadas pelo modelo são presentes, e possuem uma pilha de protocolos padronizados, que serão descritos nas seções seguintes. A Figura 2 abaixo ilustra o modelo de SOA aplicando a tecnologia de serviços Web.

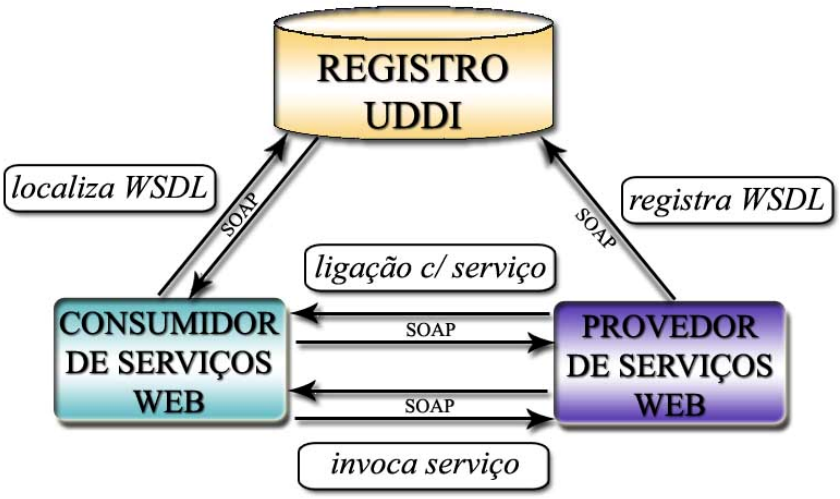


Figura 2: Cenário de serviços Web

Em um típico cenário de serviços Web, uma organização cliente precisa de um serviço específico, por exemplo, de consulta de crédito. É feita então uma consulta em um repositório de serviços UDDI, que então retorna o serviço, ou uma lista de serviços que estão disponíveis para serem invocados. A partir daí o cliente irá, utilizando o documento WSDL que contém a descrição do serviço, criar toda a infra-estrutura necessária para invocar o serviço. O cliente então envia uma solicitação a uma determinada URI utilizando o protocolo SOAP, normalmente sobre HTTP. O serviço recebe a requisição, processa e retorna o resultado.

A comunidade dos serviços Web vem constantemente promovendo a abordagem de orientação a serviços, produzindo protocolos e padrões abertos e baseados na Internet. Tem contribuído também para outras propostas de protocolos, de forma a estender o modelo básico, através da composição, transações, segurança, confiança, entre outros (BOOTH *et al.*, 2004). Todos esses protocolos serão descritos nas seções a seguir.

2.2.1 Padrões dos serviços Web

Os serviços Web são constituídos de um conjunto de padrões, que permitem a localização e invocação dos serviços, independentemente da plataforma e sistema utilizado para criar e publicar esse serviço, contanto que todos sigam aos padrões definidos.

Diversos órgãos padronizadores e da indústria estão envolvidos na especificação e padronização dos serviços Web. Existem atualmente três entidades principais responsáveis pelos padrões dos serviços Web, que são a W3C (W3C, 2007), a OASIS (OASIS, 2007), e a WS-I (WS-I, 2007). Este último não é um órgão padronizador, mas sim um órgão que determina como implementar e utilizar os padrões de forma a garantir uma melhor interoperabilidade entre eles. O capítulo 3 trata do problema da interoperabilidade entre serviços Web de forma mais detalhada.

A infra-estrutura tecnológica dos serviços Web é estruturada sobre padrões baseados na linguagem XML e nos protocolos da Internet. Os principais padrões dos serviços Web são o XML, o SOAP (*Simple Object Access Protocol*) (GUDGIN *et al.*, 2003a), o WSDL (*Web Services Description Language*) (CHRISTENSEN *et al.*, 2001) e o UDDI (*Universal Description, Discovery and Integration*) (BRYAN *et al.*, 2002).

Nas seções abaixo, serão apresentados os principais padrões e especificações dos serviços Web em detalhes. Na seção 2.2.6 são apresentadas outras especificações estendidas, conhecidas como WS-* [9].

2.2.2 XML

Histórico

O XML (*Extensible Markup Language*) é uma linguagem de marcação, que tem sua origem vinculada à SGML, que é um padrão de marcações criado pela IBM para representação de informação de múltiplos propósitos. Em 1989, foi criado a HTML, que é uma versão simplificada da SGML, e que se tornou o formato padrão para informação na Web até os dias atuais. Porém, a HTML possui uma grande limitação que é o número fixo de marcadores para expressar informações. Pensando nisso, em 1996 a W3C tentou

[9] WS-* lê-se *WS-Star* ou *WS-Splat*.

introduzir na Web os benefícios da SGML, juntamente com a simplicidade da HTML, lançando então a especificação 1.0 da XML em 1998 (BRAY *et al.*, 2006).

Definição

A estrutura da XML é similar à HTML, porém suas características principais são derivadas do SGML, como por exemplo:

- **Extensibilidade**: É possível definir novos marcadores, de acordo com as necessidades da aplicação.
- **Validação**: É possível validar a estrutura de um documento XML através das definições da estrutura, que são feitas ou via DTD (*Document Type Definition*) (MALER, 1998) ou via XML *Schema* (PETERSON *et al.*, 2006, THOMPSON *et al.*, 2006).
- **Semântica**: É possível interpretar semanticamente os elementos, atributos e conteúdos de um documento XML.

A XML, devido à sua natureza ilimitada de marcações, pode ser definida como uma meta-linguagem, ou seja, uma linguagem para definição e descrição de outras linguagens. Isso faz com que a XML seja utilizada em uma variedade de aplicações e cenários diferentes, incluindo os serviços Web.

Programas e ferramentas capazes de ler XML[10] mapeiam e transformam os tipos de dados genéricos do XML em tipos específicos dentro de um domínio de uma aplicação (por exemplo, mapear XML para Java e vice-versa). Transformar uma representação genérica de dados em XML em uma representação específica de um domínio de aplicação é um dos aspectos essenciais dos serviços Web.

Sintaxe

A sintaxe de um documento XML é constituída de marcadores e pelos dados propriamente ditos. A Figura 3 ilustra um documento XML básico. Os principais marcadores do XML 1.0 são (BRAY *et al.*, 2006):

[10] Programas que lêem XML são conhecidos como XML *Parsers*

- **Elementos ou *Tags*:** Um elemento é a marcação mais comum em um documento XML. Similarmente ao HTML, o elemento possui um *tag* de abertura e um *tag* de fechamento do elemento. O nome do elemento deve ser utilizado para identificar seu conteúdo, criando uma semântica no elemento. Por exemplo, dentro de um elemento <endereço> sabe-se que o valor será um endereço.
- **Entidades:** Uma entidade é uma representação simbólica de uma informação. Normalmente é utilizado quando se tem uma informação que é utilizada frequentemente, onde esta entidade irá referenciar essa informação, como se fosse um "atalho", ou *alias*.
- **Declaração do tipo de documento (*DOCTYPE*):** Permite que um documento XML declare ao seu *parser* meta-informações, tais como seqüência dos elementos, tipos de dados e atributos. Essa declaração é feita através de uma DTD (*Document Type Definitions*) diretamente no documento XML ou através de uma referência a um entidade de DTD externa.

```
<?xml version="1.0" encoding="UTF-8" ?>
<agenda>
  <endereco tipo="comercial">
     <rua>Rua Central</rua>
     <cidade>Florianopolis</cidade>
     <estado>SC</estado>
     <cep>98765-100</cep>
  </endereco>
</agenda>
```

Figura 3: Exemplo de um documento XML

O documento XML exemplificado acima possui 6 elementos (agenda, endereco, rua, cidade, estado, cep). Cada elemento agenda possui a sua marcação de inicio <agenda> e sua marcação final </agenda>, contendo os outros 5 elementos, sendo então chamado de elemento raiz.

Um nome de elemento deve sempre ser iniciado com uma letra ou um sublinhado, e não pode conter alguns caracteres reservados como /, <, >, ?, entre outros (BRAY *et al.*, 2006). Existem elementos que não possuem nenhum dado associado, conhecidos como **marcadores vazios**, que são representados por uma única marcação do tipo <elemento/>, que é o equivalente a <elemento></elemento> (um par de inicio e fim, sem conteúdo).

Um elemento pode conter um ou mais atributos. Os atributos são utilizados para prover informações adicionais sobre os dados contidos em um elemento. No exemplo da Figura 3, o elemento “endereco” possui um atributo chamado “tipo”, para descrever que tipo de endereco é representado pelo elemento (por exemplo, residencial ou comercial).

De acordo com o *Basic Profile* da WS-I, que será apresentado na seção 3.4.2, os documentos XML usados em serviços Web devem ter a codificação de caracteres UTF-8 ou UTF-16 (BALLINGER *et al.*, 2004). Portanto, ao se padronizar a forma de codificar os caracteres, não existirão problemas de interoperabilidade entre diferentes formas de codificação, tornando o trabalho dos desenvolvedores mais fácil, e também tornando os serviços providos mais interoperáveis.

Espaço de Nomes (Namespaces)

Um espaço de nomes XML provê um nome qualificado para um elemento ou atributo XML, da mesma forma que um *package* Java provê um nome qualificado para uma classe Java.

Um espaço de nomes permite uma melhor organização e distinção entre os elementos e atributos XML, evitando assim conflitos de nomes quando, por exemplo, dois documentos XML de diferentes organizações são trocados. A Figura 4 ilustra um documento XML mais completo, contendo marcadores, dados e definição de espaço de nomes.

O atributo *xmlns* declara um espaço de nomes XML específico, na forma de *xmlns=“URI”*, onde a URI deve estar em conformidade com a especificação de URI dada pela IETF em BERNERS-LEE *et al.*, (1998). Porém, na maioria dos casos, existem mais de um espaço de nomes a ser declarado. Para esses casos, o espaço de nomes XML possui uma notação para associar elementos e atributos com espaços de nomes (MONSON-HAEFEL, 2003).

Isso é feito através da associação do espaço de nomes a um prefixo da forma *xmlns:prefixo=”URI”*. Esse prefixo será usado em todos os elementos que pertencerem ao espaço de nomes relativo ao prefixo, onde o formato dos elementos será na forma de <*prefixo:elemento*>. Essa combinação de prefixo e elemento é conhecido como QName (*Qualified Name*). No exemplo da Figura 4, existem dois espaços de nome, *oc* e *ender*, que

são utilizados para separar cada elemento em seu respectivo espaço de nomes, como, por exemplo, *oc:OrdemDeCompra* e *ender:cep*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<oc:OrdemDeCompra data="2006-12-25"
  xmlns:oc="http://www.gsigma.ufsc.br/compra/OC"
  xmlns:ender="http://www.gsigma.ufsc.br/compra/ENDER">
  <oc:nomeCliente>Submarino</oc:nomeCliente>
  <oc:codCliente>100</oc:codCliente>
  <ender:endereco>
    <ender:nome>Submarino.com.br</ender:nome>
    <ender:rua>Av. Central</ender:rua>
    <ender:cidade>Rio de Janeiro</ender:cidade>
    <ender:estado>RJ</ender:estado>
    <ender:cep>12345-100</ender:cep>
  </ender:endereco>
  <oc:livro>
    <oc:titulo>Web Services</oc:titulo>
    <oc:quantidade>100</oc:quantidade>
    <oc:valor>40</oc:valor>
  </oc:livro>
  <oc:total>4000.00</oc:total>
</oc:OrdemDeCompra>
```

Figura 4: Documento XML com espaço de nomes

DTD (*Document Type Definitions*)

Em um DTD, é possível definir quase todos os aspectos de um conjunto de marcadores. Dessa maneira, um documento XML que declara esse DTD deve estar de acordo com a estrutura e restrições que são definidas neste, de forma que seja considerado válido (TRAMONTIN JR., 2004).

Os DTDs podem ser úteis em diversas aplicações, porém são limitados, devido ao fraco sistema de dados, que não permite a declaração de diferentes tipos de dados (PETERSON *et al.*, 2006). Em uma DTD é possível apenas definir os elementos como contendo nada, ou outros elementos, ou texto, não suportando outros tipos como inteiro, booleano e decimal, por exemplo.

XML Schema

Para solucionar as limitações de DTDs apresentadas acima, a W3C, órgão que coordena os padrões fundamentais do XML, criou uma nova forma de descrever e validar

linguagens de marcação através do uso de um esquema, chamado XML *Schema* (THOMPSON *et al.*, 2006). Um XML *Schema* provê um sistema para tipos de dados mais robusto, pois permite tipos primitivos de dados – *integer, double, boolean,* etc. – e também permite a criação de novos tipos de dados mais complexos.

Um esquema descreve uma linguagem de marcação XML, isto é, define quais elementos e atributos serão utilizados na linguagem de marcação, como eles são ordenados e aninhados e quais são os tipos de dados de cada um deles (MONSON-HAEFEL, 2003).

Um XML *Schema* descreve a estrutura de um documento XML em termos de:

- **Tipos simples:** São os tipos de dados primitivos, que não permitem a existência e elementos filho.
- **Tipos complexos**: São tipos que podem possuir elementos filho e atributos, descrevendo a forma como os elementos são organizados e aninhados.

Tipos Simples

Um esquema pode declarar um tipo chamado de **tipo simples**, que é um elemento atômico, isto é, não pode conter outros elementos, apenas dados. A especificação do XML *Schema* define 44 tipos simples como padrão, chamados de tipos embutidos - *built-in types* (PETERSON *et al.*, 2006). Os tipos embutidos são os tipos básicos para se construir um documento de XML *Schema* e são membros do espaço de nomes específico do XML *Schema*. Alguns exemplos de tipos simples mais usados são: *string*, *integer*, *decimal*, *boolean*, *date* e *time*.

É possível também criar outros tipos simples a partir dos tipos embutidos já existentes, através da derivação por restrição, onde o novo tipo aceita um subconjunto dos valores que o tipo original aceita (GESSER, 2006). Por exemplo, deseja-se restringir o formato do CEP de acordo com os padrões brasileiros, que contém 5 números de 0 a 9, seguido de um hífen, e seguido de mais 3 números de 0 a 9. Isso é feito, conforme ilustrado na Figura 5:

```
<!-- ... -->
<xs:simpleType name="cep">
     <xs:restriction base="xs:string">
          <xs:pattern value="[0-9]{5}-[0-9]{3}"/>
    </xs:restriction>
</xs:simpleType>
<!-- ... -->
```

Figura 5: Esquema XML – Tipo Simples

Tipos Complexos

Um esquema pode declarar, além do tipo simples, os chamados **tipos complexos**, que definem como os elementos contidos em outros elementos devem ser organizados (MONSON-HAEFEL, 2003). Um tipo complexo declara os nomes e tipos dos elementos, e os atributos que um elemento pode conter. A Figura 6 ilustra a declaração de um tipo complexo OrdemDeCompra:

```
<!-- ... -->
<xs:complexType name="OrdemDeCompra">
     <xs:sequence>
          <xs:element name="nomeCliente" type="string" minOccurs="0" maxOccurs="1"/>
          <xs:element name="codCliente"  type="integer" minOccurs="1" maxOccurs="unbounded"/>
          <xs:element name="total"       type="float" minOccurs="1" maxOccurs="1"/>
          <!-- ... -->
     </xs:sequence>
</xs:complexType>
<!-- ... -->
```

Figura 6: Esquema XML – Tipo Complexo

A declaração de tipos complexos em um esquema, os elementos podem ser agrupados utilizando as seguintes construções:

- ***sequence***: os elementos devem aparecer na seqüência em que são declarados. Esse é normalmente o tipo mais utilizado, devido a sua natureza mais restritiva.
- ***choice***: um, e apenas um dos elementos deve aparecer.
- ***all***: todos os elementos devem aparecer, porém em qualquer ordem.

É possível verificar nos elementos exemplificados na Figura 6 a definição de atributos do tipo *minOccurs* e *maxOccurs*, indicando, respectivamente, o número mínimo e máximo de repetições do elemento no documento XML. É importante notar que quando o

valor de *minOccurs* é zero, o elemento não é obrigatório. Já quando o valor de *maxOccurs* é *unbounded*, o elemento pode ser repetido indefinidas vezes.

Considerações sobre interoperabilidade e XML

Uma das formas de se atingir a interoperabilidade é no seu nível sintático, onde existe um acordo em relação ao formato dos dados a ser utilizado. O XML é o formato básico que é utilizado nos serviços Web , ou seja, é o formato **padrão,** utilizado em todas as implementações, promovendo então o aumento da interoperabilidade (RAY *et al.*, 2006).

Porém, problemas de interoperabilidade ainda podem surgir quando sistemas de diferentes organizações têm entendimentos diferenciados em relação ao significado dos termos do XML, ou seja, a semântica relacionada aos itens descritos via XML. Quando a interação ocorre entre organizações do mesmo setor, o significado dos termos não é um grande problema, pois todas compartilham um mesmo entendimento (RAY *et al.*, 2006). No entanto, em um ambiente heterogêneo, como, por exemplo, uma OV, organizações de diferentes setores podem interagir, levando então a problemas de interoperabilidade, principalmente semânticos.

Existem atualmente diversos esforços de pesquisa na área de interoperabilidade semântica, que promovem o uso de ontologias e mapeamento entre ontologias, conforme apresentados na seção 3.3.

2.2.3 SOAP

Histórico

O SOAP, originalmente chamado de *Simple Object Access Protocol* é um protocolo leve, voltado para a troca de informações estruturadas em um ambiente distribuído e descentralizado (GUDGIN *et al.*, 2003a). O SOAP faz uso da tecnologia XML, sendo independente de qualquer modelo de programação em particular, definindo assim uma estrutura de transferência de mensagens extensível. A idéia por trás da criação do SOAP era o de utilizar tecnologias já existentes e manter o máximo de simplicidade possível (SWITHINBANK *et al.*, 2005).

Um dos objetivos dos idealizadores do SOAP era o de fazer com que fosse possível a troca de mensagens de aplicações desenvolvidas em diferentes linguagens, rodando em diferentes plataformas, e se comunicando com diferentes protocolos através da Internet, dessa forma facilitando a interoperabilidade.

Estrutura das mensagens SOAP

Para se obter a interoperabilidade desejada utilizando o SOAP, é preciso que a mensagem SOAP seja estruturada e processada de maneira padrão.

O SOAP consiste de uma mensagem baseada no XML, possuindo o seu próprio espaço de nomes, XML *Schema* e regras de processamento. A representação da estrutura do SOAP é apresentada na Figura 7:

Figura 7: Estrutura de uma mensagem SOAP

Conforme visto anteriormente, todo documento XML deve conter um elemento raiz. No SOAP, o elemento raiz é o *Envelope*. O *Envelope* pode conter um elemento opcional de cabeçalho chamado *Header*, e deve obrigatoriamente conter um elemento *Body*, que é o corpo da mensagem SOAP, contendo os dados de aplicação. O elemento opcional *Fault* é apresentado apenas em mensagens que apresentam alguma exceção de processamento. A Figura 8 apresenta o formato da mensagem SOAP em termos de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
  <soap:Body>
    <oc:OrdemDeCompra data="2006-12-25" xmlns:oc="http://www.gsigma.ufsc.br/compra/OC">
        <oc:nomeCliente>Submarino</oc:nomeCliente>
        <oc:codCliente>100</oc:codCliente>
        <oc:livro>
            <oc:titulo>Servicos Web GSIGMA</oc:titulo>
            <oc:quantidade>5</oc:quantidade>
        </oc:livro>
    </oc:OrdemDeCompra>
  </soap:Body>
</soap:Envelope>
```

Figura 8: Exemplo de mensagem SOAP

A mensagem SOAP apresentada na figura acima contém um elemento XML chamado de *OrdemDeCompra*, onde a loja Submarino encomenda 5 livros com título "Serviços Web GSIGMA". Todos os elementos são totalmente qualificados (*fully qualified*), isto é, utilizam prefixos referentes ao seu espaço de nomes, evitando assim conflito de nomes de elementos.

Métodos de interação do SOAP

As mensagens SOAP são fundamentalmente *one-way*. Porém, elas podem ser combinadas para formar padrões de interações do tipo requisição-resposta, porém isso é feito baseando-se na WSDL e no mapeamento para um protocolo de transporte (SWITHINBANK *et al.*, 2005).

Não existe, na estrutura da mensagem SOAP, uma forma de correlacionar as respostas com as requisições. Porém, uma nova especificação que faz parte das WS-*, chamada de *WS-Addressing* introduz cabeçalhos especiais para auxiliar e gerenciar estilos mais sofisticados de interações. Mais detalhes sobre a pilha WS-* são apresentados na seção 2.2.6.

Os estilos de interação do SOAP são apresentados na seção 2.2.4. Os estilos de mensagens do SOAP, bem como detalhes sobre a interoperabilidade de cada um deles, são apresentados na seção 2.2.4.

Considerações sobre interoperabilidade e SOAP

O SOAP é considerado um padrão *de facto* para serviços Web (MONSON-HAEFEL, 2003), e vem sendo amplamente adotado por fabricantes de ferramentas e plataformas para serviços Web. Porém, nem sempre as implementações da especificação do SOAP são criadas igualmente (KUMAR *et al.*, 2004). No entanto, os principais fabricantes têm seguido as recomendações do *Basic Profile*, o que faz com que esse problema de interoperabilidade seja praticamente solucionado. Na seção 5.1 são apresentadas diversas plataformas de serviços Web que foram testadas e avaliadas em relação a sua interoperabilidade.

Porém, problemas de interoperabilidade podem surgir quando fabricantes passam a inserir em seus motores SOAP detalhes e implementações proprietárias. Um exemplo disto é o cabeçalho do SOAP, que é um elemento extensível, que permite modificar a mensagem SOAP e, dessa forma, prendê-la[11] a uma implementação específica de um fabricante.

2.2.4 WSDL

Histórico

O WSDL – *Web Service Description Language* – define uma gramática XML para descrever os serviços de rede como uma coleção de pontos de acesso[12] de comunicação, capazes de trocar mensagens (CHRISTENSEN *et al.*, 2001). A descrição dos serviços via WSDL provê uma documentação para sistemas distribuídos e serve como um recipiente para automatizar a comunicação entre aplicações, pois contém todos os detalhes sobre o serviço, como por exemplo, sua localização e forma de invocação.

O conceito de uma linguagem para a definição de interfaces já era utilizado anteriormente pelo CORBA e DCOM, através da IDL – *Interface Definition Language* (LINDEMANN *et al.*, 2006). Porém, o WSDL é uma instância de um documento XML, e é passível de uma validação por um processador XML, que se baseia em um esquema, também escrito em XML, para tal.

---

[11] Referenciado na literatura como *vendor lock-in* .
[12] Também referenciado em inglês como e*ndpoint* ou *port* para serviços Web. Nesse trabalho será utilizado "ponto de acesso".

Estrutura de um documento WSDL

A estrutura de um documento WSDL é separada em especificações abstratas e em implementações concretas dessas especificações abstratas. Isso reflete a separação entre a definição da interface do serviço – interface abstrata – e da definição da implementação do serviço – implementação concreta de um ponto de acesso (CHRISTENSEN *et al.*, 2001).

Abaixo são descritos os elementos que são utilizados pela especificação do WSDL para a definição dos serviços (SWITHINBANK *et al.*, 2005). A Figura 9 ilustra a forma como estes elementos são relacionados logicamente.

- **Types:** um container para definição dos tipos de dados que serão trocados – utilizando, por exemplo, o XML *Schema*.
- **Message:** uma definição abstrata das mensagens que serão trocadas tanto no pedido quanto na resposta, incluindo os parâmetros e seus tipos – através dos *Parts.*
- **Operation:** uma descrição abstrata de uma operação suportada pelo serviço.
- **Port Type:** um conjunto de operações abstratas suportadas por um ou mais pontos de acesso.
- **Binding:** um protocolo e especificação de formato de dados concreto para um *Port Type* particular.
- **Binding Operations:** operações que são suportadas por um determinado *binding* (referenciam as operações de um *portType*).
- **Port:** um ponto de acesso único para o serviço, que é definido como a combinação de um *Binding* e um endereço de rede.
- **Service:** uma coleção de pontos de acesso relacionados.

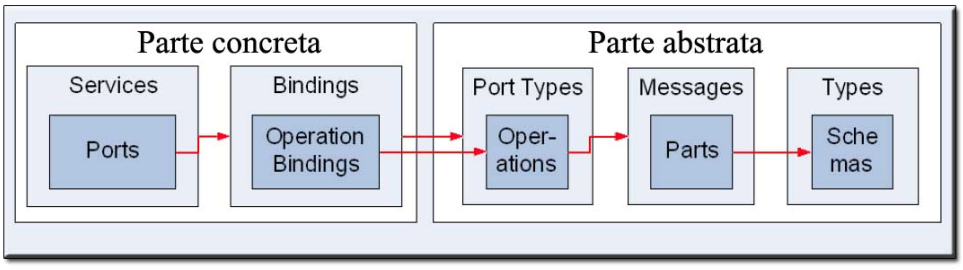
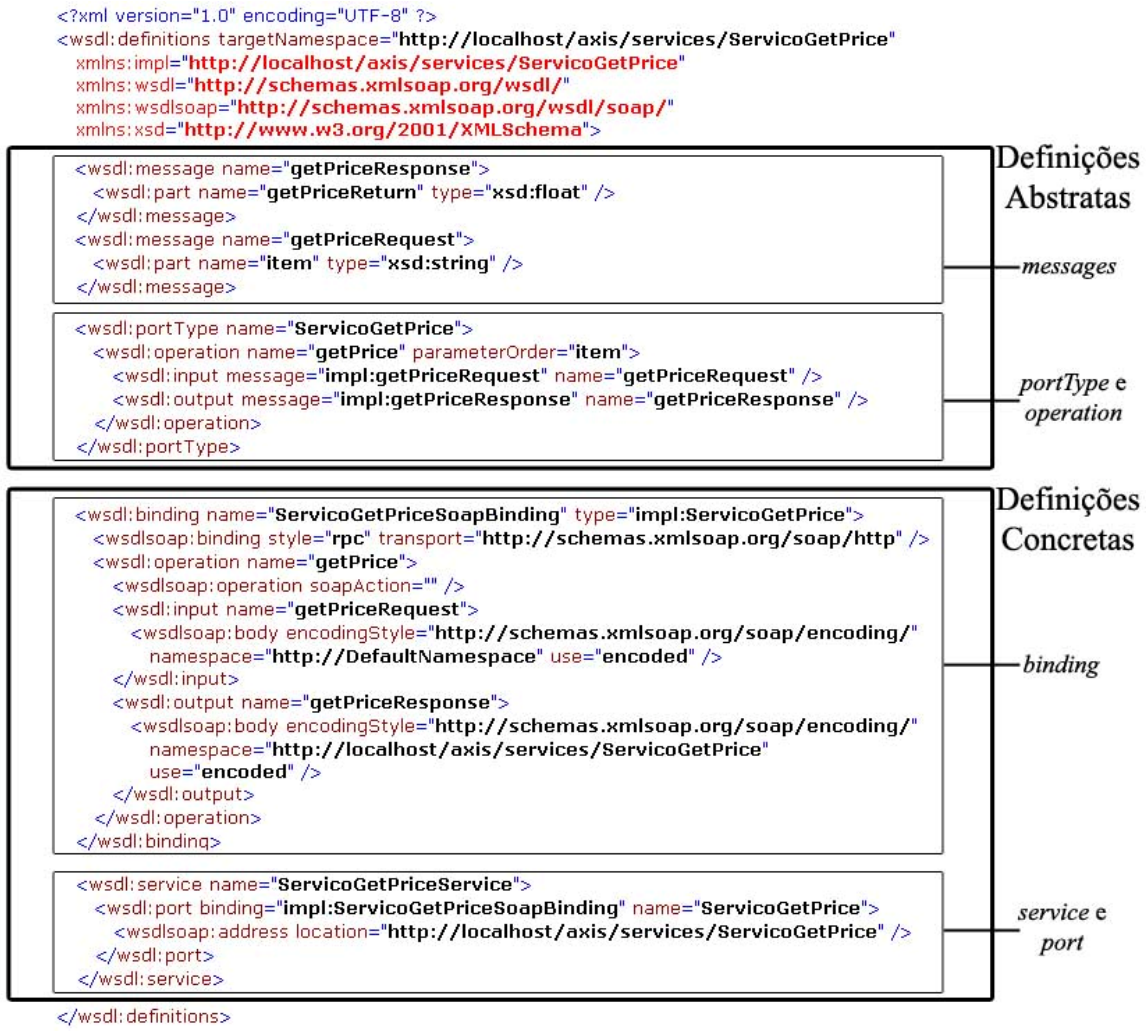


Figura 9: Relacionamento entre os elementos de um documento WSDL

A Figura 10 ilustra um documento WSDL real, separando cada uma das partes descritas, para uma melhor compreensão:




```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://localhost/axis/services/ServicoGetPrice"
  xmlns:impl="http://localhost/axis/services/ServicoGetPrice"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:message name="getPriceResponse">
    <wsdl:part name="getPriceReturn" type="xsd:float" />
  </wsdl:message>
  <wsdl:message name="getPriceRequest">
    <wsdl:part name="item" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="ServicoGetPrice">
    <wsdl:operation name="getPrice" parameterOrder="item">
      <wsdl:input message="impl:getPriceRequest" name="getPriceRequest" />
      <wsdl:output message="impl:getPriceResponse" name="getPriceResponse" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ServicoGetPriceSoapBinding" type="impl:ServicoGetPrice">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="getPrice">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="getPriceRequest">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://DefaultNamespace" use="encoded" />
      </wsdl:input>
      <wsdl:output name="getPriceResponse">
        <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="http://localhost/axis/services/ServicoGetPrice"
          use="encoded" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ServicoGetPriceService">
    <wsdl:port binding="impl:ServicoGetPriceSoapBinding" name="ServicoGetPrice">
      <wsdlsoap:address location="http://localhost/axis/services/ServicoGetPrice" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Figura 10: Exemplo de um documento WSDL e suas partes

Considerações sobre interoperabilidade e WSDL

De acordo com YE (2004), uma grande fonte de problemas de interoperabilidade nos serviços Web é o seu documento WSDL, mais especificamente na escolha do tipo de *binding* SOAP e na codificação a ser utilizada em seu envelope.

Um documento WSDL, conforme já visto anteriormente, é uma estrutura XML, que descreve os serviços Web e define suas interfaces utilizando elementos XML padrão, como o *portType*, *message*, *types*, *binding* e *service*. O elemento *binding* do WSDL permite que sejam declarados no documento, detalhes do protocolo a ser utilizado e os requisitos da mensagem, de forma que a aplicação possa se comunicar corretamente com o

serviço, isto é, interoperar (MCCARTHY, 2002). Os protocolos utilizados pelo *binding* podem ser o SOAP, HTTP GET/POST ou MIME (CHRISTENSEN *et al.*, 2001).

Um *binding* WSDL do tipo SOAP permite a definição de **estilos** – atributo *style* – da mensagem SOAP, e da forma de **codificação** dessa mensagem – atributo *use*. Isso irá ditar a forma como o envelope SOAP é construído e transmitido na rede. Os estilos podem ser *Document* ou *RPC* (*Remote Procedure Call*).

O estilo *Document* indica que o corpo da mensagem SOAP contém um documento XML simples, carregando dados. O remetente e o receptor da mensagem devem concordar previamente no formato do documento, semelhante aos sistemas tradicionais de troca de mensagens. Esse acordo é normalmente feito através das definições do *XML Schema*.

Já o estilo RPC indica que o corpo da mensagem SOAP contém uma representação em XML de uma chamada de método, utilizando o nome do método e seus parâmetros para gerar uma estrutura que representa a pilha de chamada do método.

A forma de **codificação** pode ser de dois tipos: *literal*, ou seja, a informação nas mensagens é literalmente o que o *XML Schema* define para os tipos (*types* no WSDL), ou *encoded*, que implica que a informação nas mensagens é codificada e serializada pelas regras de codificação do SOAP (GUDGIN *et al.*, 2003b). Essa especificação de codificação serve para informar como mapear estruturas de dados comuns em um formato XML. Porém, a codificação *encoded* não é recomendada pela WS-I, devido ao fato de ser limitada em relação à quantidade de tipos de dados disponíveis, o que pode gerar diversos problemas de interoperabilidade.

Então, combinando os estilos e formas de codificação, têm-se quatro possibilidades, *RPC/Encoded*, *RPC/Literal*, *Document/Encoded*, *Document/Literal*. Um outro modelo conhecido como *Document/Literal Wrapped* é uma quinta forma de codificação que também é muito utilizada (SWITHINBANK *et al.*, 2005).

O *RPC/Encoded* foi a primeira forma de se tratar as mensagens SOAP. O documento WSDL é simples, e a operação a ser executada aparece no corpo da mensagem SOAP, facilitando o despacho da mensagem para a implementação da operação. As desvantagens do *RPC/Encoded* são o *overhead* de informações de codificação que causam problemas com o desempenho, o não suporte a validação, e também a sua não compatibilidade com as especificações da WS-I (BUTEK, 2005).

O *RPC/Literal* é semelhante ao *RPC/Encoded* com a diferença que as informações adicionais de codificação são excluídas. Está de acordo com as especificações da WS-I, porém essa forma de invocação é raramente utilizada devido a sua limitação em questões de validação do documento (AKRAM *et al.*, 2006).

O *Document/Encoded* não é mais utilizado, por não ser aceito pelas especificações da WS-I e por não existir uma real utilidade na utilização dessa forma de invocação (BUTEK, 2005).

O *Document/Literal* define os tipos no próprio WSDL, e que são seguidos e utilizados na mensagem SOAP. Isso aumenta o desempenho, pois não existem informações de codificação, e todo o conteúdo do corpo da mensagem SOAP está definido no XML *Schema*[13], fazendo com que seja possível validar a mensagem, o que é uma grande vantagem em cenários de troca de mensagens. Porém, o nome da operação, antes presente, agora não existe mais, podendo causar problemas na hora do despacho da mensagem para sua implementação (BUTEK, 2005). É recomendado pela WS-I, porém com a restrição de que só pode haver um único nó filho dentro do corpo da mensagem SOAP, ficando fora das especificações quando se necessita mais de um nó filho, no caso de dois parâmetros por exemplo.

O *Document/Literal Wrapped* segue o estilo anterior, onde as definições dos tipos são feitas no próprio WSDL. Os parâmetros da mensagem são empacotados – ou *wrapped* - dentro de um nó declarado como um tipo no WSDL, que representa o nome da operação (SWITHINBANK *et al.*, 2005). As informações de codificação não são enviadas, porém o nome da operação volta a ser enviado no envelope SOAP. Também é recomendado pela WS-I, e agora com o empacotamento, é possível ter mais de um parâmetro, e mesmo assim o corpo da mensagem SOAP ainda terá somente um nó filho. A desvantagem desse estilo é o aumento a complexidade do WSDL.

Em uma organização, um dos motivos da utilização e adoção dos serviços Web como forma de expor os seus processos é o de que estes processos sejam acessíveis por diferentes organizações e plataformas, sem se preocupar com a maneira na qual o serviço

[13] A versão 1.2 do SOAP suporta o uso de outros tipos de esquema além do XML *Schema*, como por exemplo, o RELAX NG (*Regular Language Description for XML, Next Generation*) (OASIS, 2007). A utilização de diferentes esquemas não influencia na interoperabilidade, pois os *namespaces* referentes aos esquemas são claramente identificados na seção de *namespaces do* WSDL.

foi implementado. Portanto, ao eleger a forma de se expor os serviços, é preciso levar em conta a questão da interoperabilidade e se a forma como o serviço será exposto está em conformidade com as especificações da WS-I. O estilo *Document/Literal Wrapped*, apesar de sua complexidade, tem sido o estilo mais utilizado, por sua grande interoperabilidade, por ser um estilo que tenta suprir todos os problemas e desvantagens dos outros estilos, e também por ser aceito pela WS-I (BUTEK, 2005).

2.2.5 UDDI

A UDDI (*Universal Description, Discovery and Integration*) é uma especificação da OASIS que define uma forma padrão de publicar e localizar negócios e interfaces de seus serviços em um registro (BRYAN *et al.*, 2002).

Um registro UDDI oferece um mecanismo padrão de classificação, catalogação e gerenciamento de serviço, de forma que esses serviços possam ser descobertos e consumidos por outras aplicações. Dessa maneira uma organização que deseja publicar e disponibilizar seus serviços, isto é, um provedor de serviços, pode utilizar o UDDI para tal. Da mesma forma, uma organização (o consumidor do serviço) que deseja utilizar algum serviço, pode utilizar o UDDI para descobrir os serviços que se adaptam as suas necessidades e então obter os metadados necessários para a invocação do serviço.

Um registro UDDI pode ser dividido em dois tipos: público e privado (BRYAN *et al.*, 2002). Grandes corporações como IBM, Microsoft e SAP disponibilizaram registros UDDI públicos, porém atualmente estão descontinuados (MICROSOFT-UDDI, 2007). Os registros UDDI privados são geralmente aplicados à integração de aplicações entre diferentes organizações.

O UDDI é baseado em padrões já existentes como o XML, para descrever sua estrutura, e o SOAP, para a comunicação cliente-registro-cliente (SWITHINBANK *et al.*, 2005). Pode-se dizer que o UDDI é uma ferramenta de busca para “máquinas” ao invés de “seres humanos”, no entanto, existem navegadores UDDI com interfaces visuais, voltados para usuários (UDDI-BROWSER, 2007).

As informações providas por um registro UDDI podem ser utilizadas para prover três tipos de pesquisas:

- Uma pesquisa em "paginas brancas"[14], que retorna informações básicas como endereço, contatos e identificadores de uma organização e seus serviços.
- Uma pesquisa em "páginas amarelas", que retorna informações de acordo com uma dada categorização e taxonomia.
- Uma pesquisa em "páginas verdes" que retorna informações técnicas sobre os serviços Web, e também informações descrevendo a forma como executar esses serviços.

Estrutura de um Registro UDDI

Uma estrutura de um registro UDDI é composta por cinco tipos de estruturas de dados. Essa divisão por tipo de informação provê uma forma simples de auxiliar na rápida localização e entendimento dos diferentes tipos de informação que estão contidas no registro (BELLWOOD *et al.*, 2002).

Cada uma das estruturas de dados contém um número de campos que são utilizados tanto para o propósito técnico quanto de negócios, e são acessíveis por identificadores únicos, chamados de *keys*. Abaixo serão descritas cada uma dessas estruturas, apresentando na Figura 11 as relações entre essas estruturas (ORT *et al.*, 2002):

- **businessEntity:** Representa todas as informações conhecidas sobre o negócio, inclusive os serviços que são oferecidos.
- **businessService:** Contém informações descritivas em termos de negócios (não-técnicos), descrevendo os tipos de serviços oferecidos. Cada *businessService* é filho de um único *businessEntity*, que é identificado pela *businessKey*.
- **bindingTemplate:** Contêm informações sobre o acesso ao serviço, ou seja, as descrições técnicas relevantes aos desenvolvedores de aplicações que desejam localizar e invocar um serviço Web.
- **tModel:** Tem a finalidade de prover informações de categorização. Descreve também como um serviço se comporta, quais convenções utiliza, e com quais especificações ou padrões é compatível. Existe também um tipo de *tModel* que

[14] O conceito de páginas com cores faz referência a uma lista telefônica, contendo diferentes cores para cada tipo de consulta a lista.

normalmente é referido como *service type*, e aponta, através de uma URL, para o documento WSDL que descreve o serviço.

- **publisherAssertion:** Mecanismo básico para permitir que um ou mais elementos *businessEntity* sejam interligados, de forma que seja criado um relacionamento entre esses elementos.

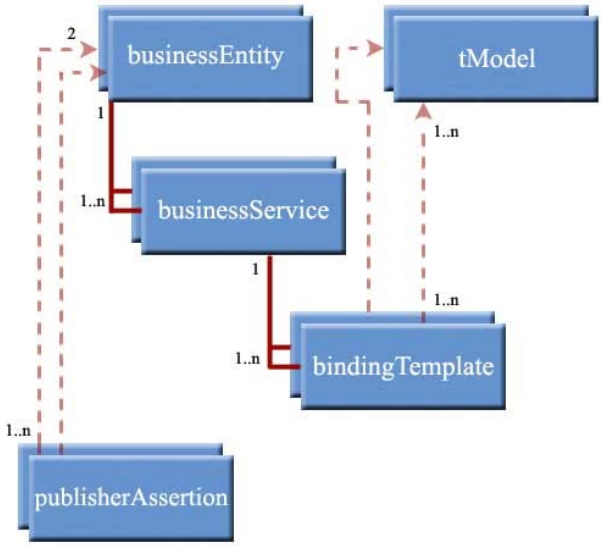


Figura 11: Relação entre estruturas de um UDDI (adaptado de (ORT *et al.*, 2002))

Mapeamento WSDL em um Registro UDDI

Conforme visto na seção 2.2.4, o arquivo WSDL foi projetado para suportar definições modulares, separadas em abstratas e concretas, onde cada definição tem um tipo de relação com outra definição.

A organização OASIS (OASIS, 2007), que é o órgão responsável pelas especificações da UDDI, define mapeamentos entre um documento WSDL e um registro UDDI, de forma a possibilitar consultas mais flexíveis e precisas, baseadas nas definições do documento WSDL.

A metodologia proposta por COLGRAVE *et al.* (2004), mapeia cada artefato WSDL para uma entidade de UDDI, representando as descrições contidas no documento WSDL de forma precisa. A Figura 12 apresenta esse mapeamento de forma detalhada:

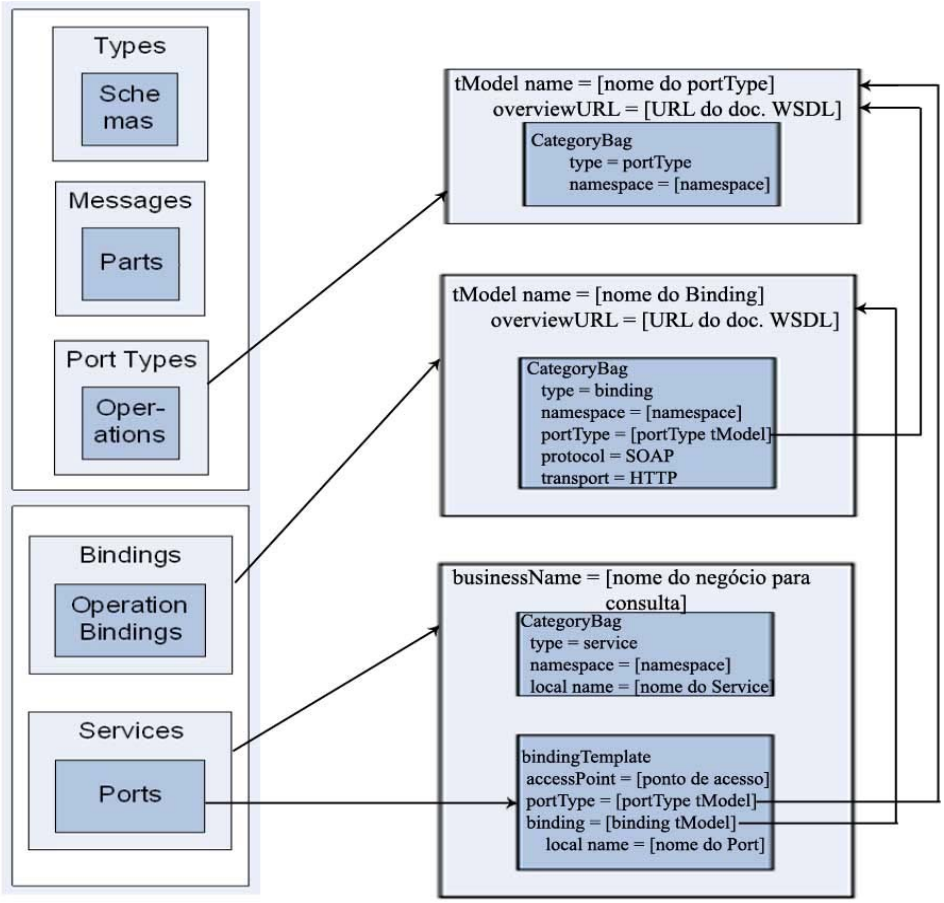


Figura 12: Mapeamento entre os elementos WSDL e UDDI

Pode-se observar que os elementos *wsdl:portType* e *wsdl:binding* são mapeados para entidades *uddi:tModel*, os elementos *wsdl:service* são mapeados para uma entidade *uddi:businessName*, e os elementos *wsdl:port* são mapeados para uma entidade *uddi:bindingTemplate*. Mantendo esse mapeamento padrão, torna-se possível que diferentes clientes possam automatizar a procura por serviços, a localização da interface do serviço – documento WSDL – para então executar os procedimentos de invocação, de maneira dinâmica e interoperável.

Motivações para uso de Registros UDDI

Um registro UDDI não descreve ou aponta para um serviço Web individual e sim para a WSDL do serviço, que contém todas as informações necessárias para a localização e invocação do serviço. Isto significa que, ao contrário do XML, WSDL e o SOAP, um registro UDDI não é uma parte essencial para o funcionamento dos serviços Web.

Porém, quando se tem uma grande quantidade de organizações interagindo em uma oportunidade de negócio, expondo seus serviços e também procurando por novos parceiros de negócios, se torna mandatário que exista uma forma de tornar os serviços públicos e visíveis. Portanto, um registro UDDI se torna aplicável principalmente em cenários onde existe uma grande quantidade de serviços disponibilizados por diferentes organizações, das mais diferentes áreas de negócios. Os registros UDDI privados têm se tornado uma solução viável nesse cenário, onde apenas as organizações com acesso definido pelo administrador do registro UDDI podem publicar seus negócios e serviços, evitando redundâncias, inconsistências e também algumas questões de segurança e integridade.

Considerações sobre interoperabilidade e UDDI

Um registro de serviços de negócios como o UDDI possibilita uma maior visibilidade, gerenciamento, adaptabilidade e reusabilidade de serviços de uma organização, se tornando uma das fundações para a infra-estrutura de uma SOA (MORELAND, 2004).

A especificação da UDDI é vasta, onde as organizações podem registrar seus serviços de maneiras distintas, causando problemas de interoperabilidade na localização e invocação por parte dos consumidores desses serviços. Portanto, a padronização na publicação em registros UDDI e mapeamento do documento WSDL se tornam essenciais para que as funcionalidades e possibilidades de um registro UDDI sejam alcançadas.

Um dos módulos implementados no presente trabalho tem a função de publicar e localizar serviços em um UDDI de maneira padronizada, desta forma, mantendo a localização dos serviços por parte de possíveis consumidores o mais interoperável possível.

2.2.6 Extensões para serviços Web

Os protocolos básicos apresentados acima – SOAP, WSDL e UDDI – são utilizados atualmente por diversas organizações para disponibilização de serviços Web. Porém, com o intuito de melhorar questões de segurança, disponibilidade e composição de serviços Web, e dessa forma suportando cenários de negócios mais complexos e confiáveis, uma grande variedade de protocolos adicionais vêm sendo propostos (WEERAWARANA *et al.*, 2005). A Figura 13 apresenta a pilha de protocolos e especificações para serviços Web:

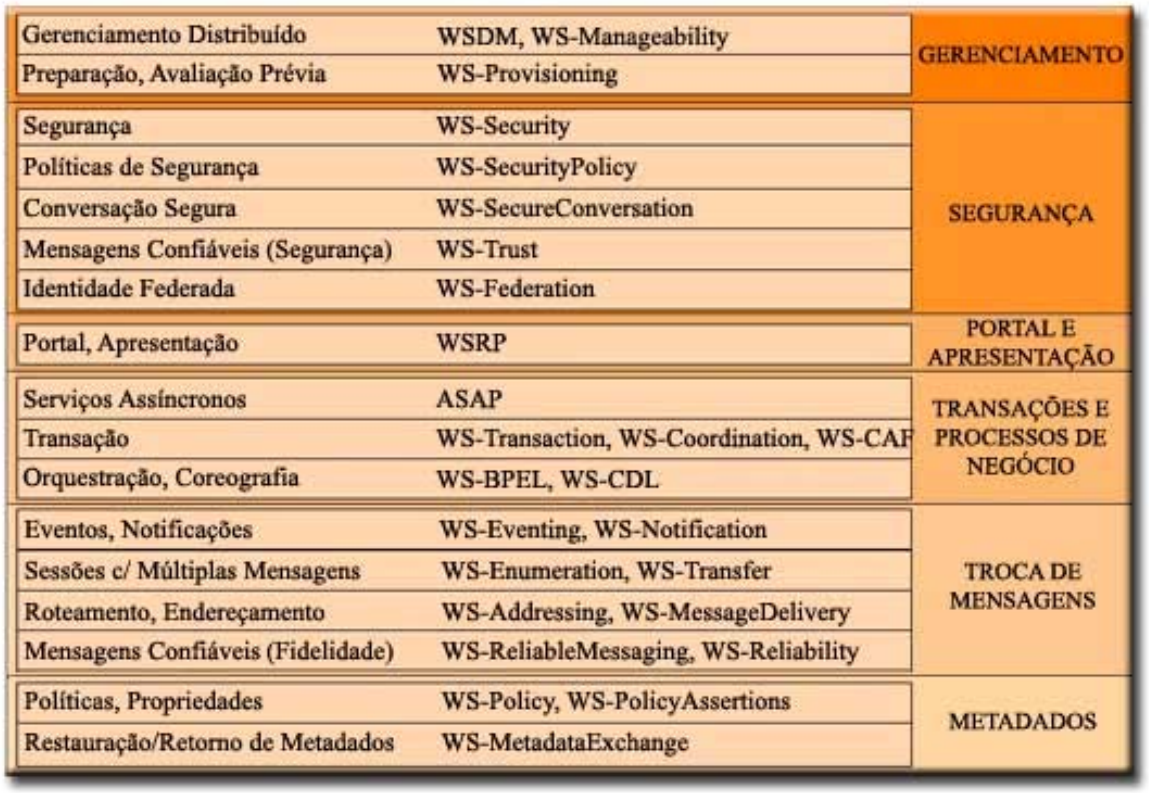


Figura 13: Pilha de protocolos WS-* (adaptado de (WILKES, 2005))

Este conjunto de especificações é chamado de WS-*, onde se lê *WS-Splat* ou *WS-Star*. Algumas destas especificações já foram aprovadas como padrão, como o *WS-Security* e o *WS-Reliability*, enquanto que outras especificações ainda não foram padronizadas, como, por exemplo, o WS-BPEL.

2.2.7 Metodologias de desenvolvimento de serviços Web

Quando se fala em interoperabilidade entre diferentes serviços Web, não se pode desconsiderar a metodologia de desenvolvimento desses serviços. Portanto, ao se escolher a metodologia de desenvolvimento mais adequada para um determinado cenário, é preciso também levar em conta questões e formas de garantir uma maior interoperabilidade entre os serviços Web. Essa seção apresenta detalhes das metodologias de desenvolvimento de serviços Web existentes, e considerações sobre a interoperabilidade de cada uma delas.

Ciclo de vida de serviços Web

O ciclo de vida de um serviço Web consiste de diferentes fases. Essas fases podem ser definidas como (BRITTENHAM, 2001):

- **Construção:** Essa fase do ciclo de vida inclui o planejamento, desenvolvimento e teste da implementação do serviço e a definição da interface do serviço. A implementação de um serviço Web pode ser feita através da criação de um novo serviço Web, da transformação de uma aplicação existente em um serviço Web, e da composição de serviços Web existentes em um novo serviço. As metodologias de desenvolvimento de serviços Web são descritas na subseção seguinte.
- **Disponibilização:** Essa fase inclui a publicação do serviço e de sua interface em um registro de serviços – UDDI, por exemplo – e inclusão dos executáveis do serviço em uma plataforma de execução de serviços Web.
- **Execução:** Durante essa fase, o serviço Web já está disponível e operacional. Os consumidores do serviço podem então localizar a sua interface e invocar todas as operações disponibilizadas pelo serviço Web.
- **Gerenciamento:** Essa fase inclui o gerenciamento e administração do serviço Web, que inclui segurança, desempenho, disponibilidade, etc.

Modelagem de serviços Web

Existem quatro métodos que um provedor de serviços pode utilizar para desenvolver um serviço Web, que são apresentados na Tabela 1. A escolha da utilização do método resulta da existência ou não de uma aplicação ou interface que irá se tornar um serviço Web.

	Nova interface de serviço	Interface de serviço existente
Novo serviço Web	*green-field*	*top-down*
Aplicação existente	*bottom-up*	*meet-in-the-midle*

Tabela 1: Métodos para desenvolvimento de serviços Web (BRITTENHAM, 2001)

Green-Field

Nesse método, primeiramente a aplicação é criada, depois exposta como um serviço Web, e então uma nova definição de interface é gerada a partir desse novo serviço.

Após isso, o serviço é então disponibilizado em um servidor de aplicações, e a interface publicada em um registro.

Top-Down

Nesse método, o serviço Web é criado a partir de uma interface de serviço antes da criação da lógica da aplicação. Essa interface pode ser também um padrão da indústria, onde diversos provedores de serviço podem implementá-la (TAO *et al.*, 2006). O provedor do serviço primeiramente cria ou localiza a descrição da interface do serviço, em seguida implementa as operações dessa interface, para então disponibilizar o seu serviço em um servidor de aplicação e publicá-lo em um registro.

Bottom-Up

Ao contrário do *top-down*, esse método é usado para criar uma nova interface de serviços a partir de uma aplicação ou função de negócio já existente, que pode ter sido desenvolvida em diferentes linguagens de programação em sistemas legados da organização (PEREPLETCHIKOV *et al.*, 2005). Esse método se assemelha ao do *green-field*, porém já existe uma aplicação previamente desenvolvida e funcional, que será exposta como um serviço Web através da criação e publicação da sua interface. Esse método permite que o desenvolvedor reutilize componentes já existentes.

Meet In The Middle

Esse método é uma combinação dos métodos *top-down* e *bottom-up*, que consiste na utilização de uma interface e de uma aplicação já existente, de forma que o desenvolvedor implemente um suporte adicional entre estes, através da utilização de um adaptador ou um invólucro[15] (TAO *et al.*, 2006). Nesse método, uma funcionalidade de processos negócios de alta granularidade[16] é utilizada, e adaptada, de forma que se forneça um serviço com uma granularidade mais baixa (PEREPLETCHIKOV *et al.*, 2005).

[15] Invólucro também é referenciado na literatura como *wrapper*

[16] A granularidade de um serviço pode ser baixa (*fine-grained*) ou alta (*coarse-grained*), e se refere ao escopo ou funcionalidade que um serviço expõe (LEYMANN, 2003). Um serviço de granularidade baixa provê uma função de negócios mais simples (como acesso a dados, por exemplo) e que oculta seus detalhes de implementação, enquanto que um serviço de granularidade alta normalmente provê funções de negócios mais complexas, expondo as funcionalidades e os detalhes de implementação.

Considerações sobre a metodologia de desenvolvimento e interoperabilidade

De acordo com LEHMANN (2005), a escolha do método para desenvolvimento de serviços Web pode ter influência direta em sua interoperabilidade.

Foram apresentados na seção anterior quatro métodos para o desenvolvimento de serviços Web, porém os métodos *top-down* e *bottom-up*[17] são os mais comumente utilizados para desenvolvimento de serviços Web (LEHMANN, 2005, PEREPLETCHIKOV *et al.*, 2005), e por isso foram escolhidos para uma análise mais detalhada a respeito da interoperabilidade.

Conforme visto, no método *top-down* existe uma análise prévia dos processos de negócio e a identificação das atividades desse processo, para então definir as operações que podem ser expostas na especificação do serviço. No método *bottom-up,* é feita uma análise das aplicações já existentes em um sistema legado, identificando aqueles que podem ser expostos como serviços (PEREPLETCHIKOV *et al.*, 2005).

O método de *top-down* torna os serviços mais interoperáveis em comparação aos serviços gerados pelo método de *bottom-up* (LEHMANN, 2005, PEREPLETCHIKOV *et al.*, 2005). Essa maior interoperabilidade é devida ao fato de que no método *top-down* o desenvolvimento é iniciado a partir da definição da interface e mensagens do serviço, evitando assim tipos de dados específicos de uma linguagem de programação em particular. Por exemplo, um problema de interoperabilidade pode ocorrer quando um cliente .NET invoca um serviço Java que utiliza algum tipo de *Collections* (*Vectors*, *HashMap*, etc), que é específico da linguagem Java (SWITHINBANK *et al.*, 2005).

2.2.8 Invocação dos serviços Web

Um dos quesitos necessários aos serviços Web, de forma que satisfaça as necessidades de agilidade e eficiência em um negócio, é o de rapidamente identificar o(s) serviço(s) necessário(s) para executar um determinado processo de negócios, e invocá-lo(s), sempre que houver necessidade, de forma transparente e fácil para o usuário.

[17] Também referenciados como *contract-first* e *code-first*, respectivamente.

Os serviços Web permitem que aplicações heterogêneas invoquem métodos publicados através do uso de protocolos padrão. A forma de invocação dos serviços Web, segue o modelo de integração tradicional de invocação remota de procedimento (HOHPE *et al.*, 2003), o qual permite a integração Aplicação-para-Aplicação, onde uma aplicação "A" invoca um método disponibilizado por uma aplicação "B", como se fosse um método local de "A".

O mecanismo que fornece essa transparência é o modelo de *Proxy*, que envolve a criação de um *stub*, que tem a função de aceitar requisições de chamadas de métodos, e encaminhá-las para outro serviço que irá realmente executar a operação. O *stub* se encarrega de encontrar o serviço, serializar os argumentos e enviá-los a real implementação, e deserializar a resposta do serviço, e encaminhá-lo ao cliente.

Existem duas formas de invocação de serviços Web:

- **Invocação Estática**: Utiliza *stubs* que são gerados previamente à execução do cliente.
- **Invocação Dinâmica**: Gera os *stubs* no momento da invocação do método, em tempo de execução do cliente.

Nas seções seguintes é apresentada a definição de cada um dos tipos de invocação, onde são apresentadas as utilizações de cada uma delas.

Invocação Estática

Na invocação estática, o desenvolvedor de um cliente para serviços Web primeiramente obtém o documento WSDL do serviço que deseja invocar, e então gera as classes *stub* referentes a esse serviço utilizando alguma ferramenta, por exemplo, o WSDL2Java (APACHEWEBSERVICES, 2007), que mapeia todos os *bindings* e *portTypes* necessários para o cliente. Após isso, utilizando as classes *stub* geradas, são criados então os procedimentos de invocação – chamado de *Proxy* – que serão utilizados pelo cliente para invocar o serviço (BUSCHMANN *et al.*, 1996).

A grande vantagem desse tipo de invocação é que exige um menor processamento no tempo de execução, pois o cliente não precisa executar algumas tarefas prévias, como localizar o serviço, fazer a análise completa do documento WSDL e construir o *proxy* para a invocação do serviço, que é construído na fase de desenvolvimento do cliente.

Porém, existem diversas limitações desse modelo de invocação. Em um ambiente dinâmico e de constante evolução, como uma OV, organizações entram e saem de um processo de negócio constantemente, ou seja, serviços ora estão disponíveis, ora não. Quando isso acontece, o cliente precisa alterar seus procedimentos de invocação, de forma a atender as mudanças necessárias, como por exemplo, a escolha de um mesmo serviço, porém provido por outra organização. Ou seja, se um cliente utiliza 'n' serviços Web diferentes, todos os 'n' *stubs* e *proxies* precisam ser mantidos no lado do cliente e alterados quando o serviço é modificado.

Outra desvantagem desse tipo de invocação está relacionado com a escolha dinâmica de serviços em um registro UDDI (YU *et al.*, 2004), onde os serviços são descobertos e invocados em tempo de execução, de acordo com as necessidades do negócio, tornando esse tipo de invocação inviável.

Invocação Dinâmica

Na invocação dinâmica, ao contrário da estática, o cliente não precisa criar previamente – em tempo de projeto[18] – os *stubs* e o *proxy* para a invocação, que são gerados em tempo de execução, utilizando APIs de invocação dinâmica. Isso é feito através da análise do documento WSDL provido pelo serviço Web.

A vantagem desse tipo de invocação é que mudanças no serviço – e na sua descrição – ou a sua exclusão não irão afetar a invocação do serviço Web, pois o WSDL é lido em tempo de execução. Uma questão importante também é que o cliente não precisa manter os *stubs* do serviço localmente.

Existem diferentes maneiras de implementar uma invocação dinâmica. Porém, algumas abordagens utilizadas se tornam ineficientes e não interoperáveis dentro de alguns cenários. Os detalhes e formas de implementação são detalhados na seção 5.3.2.

2.3 Redes Colaborativas de Organizações

Uma Rede Colaborativa de Organizações (RCO), é um conceito utilizado para representar todas as diferentes formas de colaboração entre grupos de entidades autônomas

[18] *Design time* ou *Development time*

estruturadas em rede (CAMARINHA-MATOS *et al.*, 2004). Este conceito agrupa todas as formas existentes de colaboração, bem como outras emergentes, sobre o mesmo domínio de pesquisa. Em decorrência da crescente globalização e do aumento da necessidade de alianças e terceirizações, as organizações tendem a trabalhar de maneira interligada e organizada, colaborando umas com as outras.

Este trabalho é voltado principalmente para a interoperabilidade entre diferentes organizações pertencentes a uma Rede Colaborativa. Portanto, nesta seção, serão apresentados alguns conceitos fundamentais e características relacionadas ao estado da arte em redes colaborativas.

2.3.1 Organização Virtual

O termo Organização Virtual (OV) representa um dos mais difundidos tipos de RCOs (BALDO, 2006). Uma organização virtual representa um conjunto de entidades ou organizações independentes e geograficamente distribuídas, conectadas através de uma infra-estrutura de comunicação, onde os participantes estão comprometidos a alcançar um objetivo comum através do compartilhamento de seus recursos e competências (CAMARINHA-MATOS *et al.*, 2004). Os serviços e funcionalidades disponibilizados pelas OVs são vistos como oferecidos por uma única organização.

2.3.2 Empresa Virtual

Existem diferentes conceituações para Empresas Virtuais. Porém, a definição mais completa e que mais se encaixa ao tema deste trabalho é a de CAMARINHA-MATOS *ei al.* (1999), que diz que uma Empresa Virtual é uma aliança temporária de empresas que se unem para compartilhar competências e recursos para melhor responder a uma oportunidade de negócios, e essa cooperação é suportada por redes de computadores. As empresas virtuais são enquadradas como um caso particular de uma Organização Virtual.

2.4 Trabalhos Relacionados

O atual estado da arte em interoperabilidade envolve diversas áreas de pesquisa, como serviços Web, infra-estruturas inteligentes, projetos de padronização e plataformas para negócios eletrônicos (ATHENA-IP, 2005). Essa seção tem como finalidade apresentar

alguns trabalhos e projetos de pesquisa relacionados com a interoperabilidade, principalmente aqueles que são focados na questão da interoperabilidade em sistemas computacionais, serviços e arquiteturas de suporte, que é o foco desse trabalho.

Os projetos foram selecionados baseando-se em sua relevância e contribuição científica, inclusive contribuindo com material bibliográfico – artigos, relatórios técnicos, *deliverables*, etc. – para o desenvolvimento dessa dissertação.

2.4.1 Projetos de Pesquisa

Projeto ATHENA

O ATHENA – *Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications* – é um projeto de pesquisa integrado, voltado para a área de integração e interoperabilidade em aplicações empresariais, com o objetivo de remover as barreiras para a troca de informações entre diferentes organizações (ATHENA-IP, 2004). Este projeto, por ser o mais representativo em relação à interoperabilidade, será apresentado em maiores detalhes, apresentando cada área de atuação, de forma que seja possível enquadrar de maneira mais clara o presente trabalho.

Sua principal área de pesquisa é a de interoperabilidade entre aplicações, por ser considerado um assunto importante, devido à tendência das empresas de tornar visíveis externamente suas aplicações e parte de seus processos de negócios de forma a se manterem competitivas (ATHENA-IP, 2004). A missão que é definida pelo projeto é a de contribuir para que empresas possam atingir uma interoperabilidade plena e sem costuras, através da definição dos requisitos de interoperabilidade para aplicações, dados e comunicações, e prover soluções que satisfaçam esses objetivos.

O projeto ATHENA define um *framework*, que é um modelo conceitual que engloba diferentes aspectos da interoperabilidade, como Negócios, Conhecimento, Plataformas de TIC e Semântica. Para que um negócio possa ser efetuado entre duas organizações é preciso que haja uma interdependência entre todos esses aspectos, conforme apresentado na Figura 14.

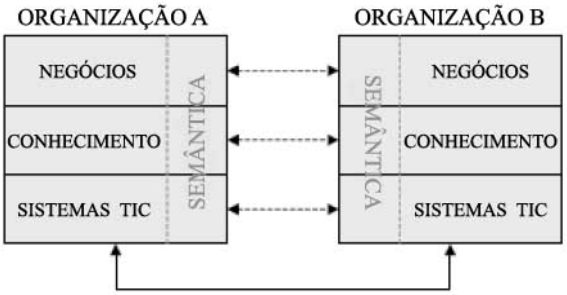


Figura 14: Modelo ATHENA

A camada de Negócios, localizada no topo, está relacionada com a organização e com suas operações, como por exemplo, operações de produção, tomada de decisões e gerenciamento de relacionamentos com parceiros. A camada de Conhecimento inclui funções organizacionais, as habilidades e competências da organização. A camada de Plataformas de TIC é focada em soluções de TIC que permitem que uma organização funcione, tome decisões, troque informações dentro e fora da organização e invoque serviços. A camada Semântica sobrepõe todas as outras camadas, de forma a capturar e representar os conceitos e significados, promovendo um melhor entendimento em todas as camadas (ATHENA-IP, 2004).

A execução completa de uma aplicação empresarial é orquestrada pelo modelo de processos de negócio identificada na camada do topo e representada e armazenada formalmente na camada intermediária. A camada mais baixa é vista como a habilidade dos sistemas TIC da organização de colaborarem com outros sistemas, de outras organizações.

O projeto ATHENA é considerado um dos maiores projetos de pesquisa que tem como foco a interoperabilidade. Seu objetivo principal é o de produzir resultados que cubram todos os aspectos da interoperabilidade: aspectos tecnológicos, aplicações e serviços, pesquisa, desenvolvimento, testes e treinamento. Diversos documentos desenvolvidos pelo ATHENA foram utilizados para consultas bibliográficas para o desenvolvimento dessa dissertação.

Projeto INTEROP-NOE

O projeto INTEROP-NOE – *Interoperability Research for Networked Enterprises Applications and Software, Network of Excellence* – tem como objetivo criar condições para uma pesquisa inovadora e competitiva no domínio da interoperabilidade para sistemas e aplicações empresariais (INTEROP-NOE, 2007). Isso é feito através de uma abordagem

multidisciplinar, que abrange três áreas de pesquisa de suporte a interoperabilidade (VALLESPIR *et al.*, 2004):

- Arquitetura e Plataformas: prover *frameworks* e estruturas de suporte para a implementação.
- Modelagem Empresarial/Organizacional: definição de requisitos de interoperabilidade e suporte às soluções de implementação
- Ontologias: identificar e prover interoperabilidade semântica em empresas.

Esse projeto trabalha em conjunto com o projeto ATHENA, de forma a garantir um impacto mais significativo na indústria.

Projeto ABILITIES

O projeto de pesquisa ABILITIES – *Application Bus for InteroperabiLITy In enlarged Europe SMEs* – tem como objetivo estudar, projetar e desenvolver uma arquitetura federada, que será implementada por um grupo de mensagens UBL (OASIS, 2007) e serviços básicos de interoperabilidade, de forma a suportar EAI nas PMEs no contexto de negócio eletrônico, especificamente em países pouco desenvolvidos (ABILITIES-PROJECT, 2007). A pesquisa é dividida em três áreas:

- Orientação para a ampliação das PMEs na Europa: definição de um formato XML genérico para documentos de negócios, “do pedido ao faturamento”, baseando-se nas especificações UBL da OASIS.
- Arquitetura federada e documentos inteligentes de negócios: prover suporte para arquiteturas federadas, utilizando-se de SOA e Sistemas Multiagentes.
- Suporte a orquestração de serviços e processos de negócios: estudar e prover pesquisa e padrões para gerenciamento de processos de negócios (BPM).

Projeto GENESIS

O projeto de pesquisa GENESIS tem como objetivo a pesquisa, desenvolvimento de infra-estruturas, *middlewares* e metodologias que permitam que PMEs conduzam suas transações via Internet (GENESIS-PROJECT, 2007). Isso é feito através de:

- Modelagens das aplicações e transações mais típicas das PMEs.

- Projeto e desenvolvimento de protocolos e formatos de dados para interoperabilidade dessas aplicações.
- Desenvolvimento de uma infra-estrutura central de suporte as aplicações.
- Disseminação dos resultados para fabricantes de software, empresas, governos e universidades.

MEL

O MEL (*Manufacturing Engineering Laboratory*) é um laboratório de pesquisa pertencente ao instituto governamental americano NIST (*National Institute of Standards and Technology*), e tem como finalidade prover padrões, tecnologias e serviços para a indústria da manufatura (MEL-NIST, 2007).

Dentro desse laboratório, existe uma área de pesquisa voltada exclusivamente para interoperabilidade na manufatura, que possui ferramentas de teste de interoperabilidade entre padrões da indústria da manufatura, incluindo serviços Web, semântica, ebXML, entre outros (RAY *et al.*, 2006).

SICoP

O SICoP (*Semantic Interoperability Community of Practice*) é um grupo especial de pesquisa em interoperabilidade semântica, que faz parte do KMWG (*Knowledge Management Working Group*), que é um projeto de pesquisa voltado para Semântica no Governo Eletrônico, patrocinado pelo governo dos Estados Unidos (SICOP-RESEARCH, 2007). O objetivo do SICoP é o de alcançar a **interoperabilidade semântica** e a **integração de dados semântica**, focando-se em serviços de Governo Eletrônico.

Projeto SATINE

O projeto de pesquisa SATINE – *Semantic-based interoperability infrastructure for integrating Web service platforms to peer-to-peer networks* – foi finalizado em 2006, porém sua área de pesquisa, os serviços Web semânticos, ainda são de grande interesse no meio científico (BATTLE *et al.*, 2005, BURSTEIN *et al.*, 2005, LILLENG, 2005, LI *et al.*, 2006).

O objetivo do projeto era o de desenvolver um *framework* de interoperabilidade baseado em semântica. Esse *framework* tem a finalidade principal de permitir que serviços Web, cadastrados em um registro ebXML ou UDDI possam interoperar de maneira sem costuras com serviços Web em redes P2P, principalmente na indústria do turismo (SATINE-PROJECT, 2007).

Isso foi feito através de um aperfeiçoamento dos registros ebXML e UDDI, utilizando ontologias e informações semânticas que são armazenadas nos registros.

Projeto SemanticGov

O projeto de pesquisa SemantiGov, é um projeto voltado para a interoperabilidade governamental. O objetivo do projeto é o de construir uma infra-estrutura – software, modelos, serviços, etc. – necessária para capacitar a disponibilização de serviços Web semânticos para a administração pública. Com isso, o projeto pretende prover uma interoperabilidade entre diferentes agências governamentais, tanto internas ao país, como inter-países, facilitando também a descoberta de serviços - tanto por usuários/cidadãos quanto por outras agências - e a execução de serviços mais complexos, como por exemplo, em *workflows* envolvendo diferentes agências (SEMANTICGOV-PROJECT, 2007).

O projeto se baseia no paradigma de Orientação a Serviços, implementado através da tecnologia de serviços Web semânticos, suportados por uma análise e modelagem rigorosa do domínio de administração pública na Europa.

Projeto Tango

O projeto Tango é uma iniciativa da *Sun*, focado na interoperabilidade de tecnologias de serviços Web, principalmente no nível de mensagens, segurança, disponibilidade e qualidade de serviços. Para tal, desenvolveu o WSIT (*Web Services Interoperability Technology*), que é uma implementação de código aberto, que visa prover a interoperabilidade entre serviços Web desenvolvidos nas plataformas Java e .NET (CARR, 2006).

Projeto TERREGOV

O Projeto de pesquisa TERREGOV é voltado para a interoperabilidade de serviços de Governo Eletrônico (*e-Government*), com foco nos *frameworks* de interoperabilidade

em e-Government na Europa (TERREGOV-PROJECT, 2007). A interoperabilidade entre serviços de governo é uma das mais importantes linhas de ação estratégicas no desenvolvimento do governo eletrônico e tem sido adotada pelos países mais avançados nessa área (SANTOS, 2004).

O objetivo do projeto é o de permitir, através de uma infra-estrutura que faz uso de serviços Web e semânticas, que administrações locais, intermediárias e regionais possam prover uma variedade de serviços *online*, tornando-os disponíveis de uma forma transparente e simples.

2.4.2 Trabalhos Acadêmicos

Web Services Interoperability: A Practitioner's Experience

Neste artigo, SIDDHARTHA *et al.* (2003) fazem um levantamento prático dos problemas relacionados com a interoperabilidade em serviços Web. São levantados problemas na arquitetura e implementação dos serviços Web, principalmente relacionados com as inconsistências e incompatibilidades entre plataformas de serviços Web pertencentes a diferentes fabricantes. A análise do problema é feita através de um exemplo prático, onde se tenta acessar um serviço desenvolvido utilizando o motor SOAP[19] da Fundação Apache através de um cliente que utiliza o motor SOAP da Microsoft.

SIDDHARTHA *et al.* (2003) tentam prover soluções que contornem os problemas encontrados, através da alteração manual de códigos e dos documentos WSDL e do XML *Schema*. Porém, atualmente esses problemas já foram sanados através do *Basic Profile* da WS-I (BALLINGER *et al.*, 2004), que provê direcionamentos e recomendações para o desenvolvimento de serviços Web interoperáveis. Grandes fabricantes, como a IBM, BEA, Microsoft e Sun, já desenvolvem seus produtos em conformidade com a WS-I.

Os problemas apresentados por esse artigo, foram utilizados para avaliar a interoperabilidade dos serviços Web desenvolvidos no presente trabalho, através do monitoramento das trocas de mensagens e utilização de ferramentas providas pela WS-I

[19] Motor SOAP ou SOAP *Engine* é uma entidade que gerencia o protocolo SOAP, incluindo a serialização, troca de mensagens com outros motores SOAP, através de protocolos de transporte, como o HTTP.

como forma de verificar a conformidade dos serviços Web e eventuais problemas de interoperabilidade.

WS-I Basic Profile: A Practitioner's View

Neste artigo, KUMAR *et al.* (2004) descrevem os problemas mais comuns de interoperabilidade relacionados com serviços Web, mais especificamente com o SOAP e o WSDL. O artigo também apresenta uma avaliação mais detalhada do *Basic Profile* da WS-I (BALLINGER *et al.*, 2004), detalhando as suas limitações, e também descreve como este pode ser efetivamente empregado no desenvolvimento dos serviços Web, de maneira que se atinja um nível considerável de interoperabilidade.

Esse trabalho pode ser considerado um complemento do trabalho de SIDDHARTHA *et al.* (2003), onde são apresentados problemas de interoperabilidade entre serviços Web relacionados com os padrões, como por exemplo, tipo de codificação do SOAP e posicionamento e ordem dos elementos em um arquivo WSDL, e também foram utilizados para avaliação e análise de interoperabilidade dos serviços Web desenvolvidos no presente trabalho.

Towards an Interoperability Framework for Model-Driven Development of Software Systems

Nesse artigo, ELVESÆTER *et al.* (2005) propõen um *framework* de interoperabilidade para sistemas e aplicações de software, baseando-se no paradigma de Desenvolvimento Orientado por Modelos[20] (MELLOR *et al.*, 2003) para projetar e desenvolver os sistemas de software, suportando as necessidades de negócio de cada organização. Utilizando modelos, esse paradigma provê uma melhor forma de tratar e resolver os problemas de interoperabilidade, comparando-se com as abordagens tradicionais (FISCHER *et al.*, 2005a).

A proposta deste *framework* é de prover direcionamentos a respeito de como o Desenvolvimento Orientado por Modelos pode e deve ser aplicado de maneira específica,

[20] *Model-Driven Development*: No paradigma de desenvolvimento orientado por modelos é baseado em modelos refinados entre diferentes níveis de abstração. Nesse paradigma de desenvolvimento, os desenvolvedores não ditam exatamente como um sistema é implementado, mas permite que o desenvolvedor utilize modelos para especificar qual funcionalidade de sistema é necessária e qual arquitetura será utilizada.

de forma a tratar os problemas de interoperabilidade entre organizações. Isso é alcançado através de um conjunto de modelos de referência que tratam de problemas de interoperabilidade em diferentes domínios, que são definidos como integração conceitual, integração técnica e integração prática de sistemas de software (ELVESÆTER *et al.*, 2005).

O *framework* proposto neste artigo não lida com problemas específicos de serviços Web, tratando o problema de uma forma mais macro, provendo modelos genéricos e direcionamentos, que podem ser utilizados para construção de metodologias e tecnologias de software, inclusive no provimento de serviços Web.

A Composite Design Pattern for Stubless Web Service Invocation

Neste artigo, BUHLER *et al.* (2004) dissertam sobre os problemas da invocação de serviços Web, principalmente em relação ao problema de interoperabilidade devido à rápida evolução dos serviços Web, e também devido às diferentes implementações e invocações providas pelos fabricantes das plataformas de serviços Web. Os autores afirmam que a invocação de serviços por parte dos consumidores de serviços deve ser genérica e simples, dessa forma aumentando a flexibilidade. Para tal, propõem a aplicação de padrões de projeto e modelagem de software[21] em serviços Web, onde através da composição desses padrões, se oculta do cliente as peculiaridades das plataformas de serviços Web de diferentes fabricantes.

O artigo descreve também a implementação de um invocador *stubless*, que segue o modelo arquitetural baseado na composição de padrões de modelagem, porém sem apresentar muitos detalhes de implementação. Algumas limitações do protótipo proposto é que o invocador funciona apenas para serviços Web baseados em SOAP, e restritos a serviços do tipo *rpc/literal*, não sendo capaz de invocar serviços baseados no tipo *document/literal*. Os tipos e estilos de serviços Web serão apresentados na seção 2.2.4.

On-Demand Invocation of Web Services

Nos últimos anos, pesquisa e desenvolvimento conduziram para diversas ferramentas e *frameworks* para criação, desenvolvimento e disponibilização de serviços

[21] Do inglês, *Design Patterns*

Web. Porém, pouco tem sido feito em relação ao consumo dos serviços web (ZHAO *et al.*, 2006).

Nesse artigo, ZHAO *et al.* (2006) descreve o projeto e a implementação de um *framework*, chamado de DWSIF – *Dynamic Web Service Invocation Framework* – que suporta a invocação sob demanda de serviços Web, onde as aplicações cliente, i.e., os invocadores dos serviços, podem se adaptar as constantes mudanças nos serviços Web.

No DWSIF, um *proxy* para o serviço Web é gerado automaticamente apenas na primeira invocação do serviço, ou quando existe alguma alteração no serviço correspondente a esse *proxy*.

O cliente invoca o serviço através de um invocador dinâmico, que consiste de dois componentes:

- **motor de invocação**: responsável pela localização do *proxy* no *cache*, e invocação do serviço.
- **motor de configuração do proxy**: responsável pela geração do proxy e seu armazenamento em um *cache*, e também pelo monitoramento de alterações no serviço Web.

A abordagem adotada por ZHAO *et al.* (2006) é válida no sentido de que através do armazenamento dos *proxies* dos serviços, a sua invocação se torna mais rápida em relação a uma invocação *stubless*, visto que na última, a cada invocação é preciso ler todo o documento WSDL para a criação dinâmica dos procedimentos de invocação.

Porém, em um ambiente altamente volátil e em alguns casos complexo, como uma OV, a criação, armazenamento e gerenciamento de *proxies* torna-se extremamente difícil, custoso, moroso e muitas vezes desnecessário, visto que as organizações entram e saem de um processo de negócios muito dinamicamente.

Challenges and techniques on the road to dynamically compose web services

Nesse artigo, SANTOS *et al.* (2006) focam na interoperabilidade voltada para a composição automática e dinâmica de serviços Web pertencentes a diferentes organizações. Os autores afirmam que soluções efetivas para possibilitar a interoperabilidade entre sistemas heterogêneos e inter-organizacionais precisam ser

propostas, de forma a permitir um desenvolvimento pleno de aplicações voltadas para a Web. O foco é dado em Arquiteturas Orientadas a Serviços, mais especificamente na camada de Composição de serviços, onde afirmam ainda ser uma grande área de pesquisa e que existem muitos conceitos mal compreendidos.

As principais contribuições do artigo são:

- Propor uma classificação clara das diferentes estratégias de composição, em relação aos níveis de dinamismo e automatização.
- Introdução de uma abordagem genérica, baseada em modelo (*Model-Driven*) para a composição de serviços, de maneira dinâmica e automática, levando em conta aspectos como seleção dinâmica de serviços, e do uso de técnicas de Inteligência Artificial para seleção dos serviços e para geração automática de planos de execução da composição.

Esse artigo apresenta diversas idéias que foram adotadas para o desenvolvimento do protótipo desse trabalho, como a necessidade de uma localização automática e invocação dinâmica de serviços Web entre diferentes organizações, aplicados em Redes Colaborativas de Organizações. É importante ressaltar que SANTOS *et al.* (2006) não restringem e nem aplicam as estratégias propostas a um domínio, tecnologia ou cenário específico.

Considerações sobre os trabalhos acadêmicos

Os trabalhos acadêmicos apresentados acima contribuíram de diferentes maneiras para o desenvolvimento do presente trabalho.

Na avaliação de interoperabilidade entre os diferentes serviços Web disponibilizados nas plataformas descritas no capítulo 5, os trabalhos de SIDDHARTHA *et al.* (2003) e KUMAR *et al.* (2004) contribuíram no sentido de indicar os principais problemas de interoperabilidade entre os padrões para serviços Web, e dessa forma, direcionando os testes dos serviços feitos neste trabalho. Porém, por serem trabalhos apenas investigativos sobre problemas de interoperabilidade, não apresentam soluções reais, apenas indicando alterações manuais no desenvolvimento dos serviços para que se tornem interoperáveis.

Apesar do artigo de ELVESÆTER *et al.* (2005) não lidar com problemas específicos para serviços Web, o artigo apresenta, de forma conceitual, diversas metodologias e definições que serviram como base para o enquadramento e desenvolvimento da abordagem proposta, como por exemplo, a separação dos níveis de interoperabilidade e também os modelos de referência que são apresentados em diferentes níveis.

O trabalho de BUHLER *et al.* (2004) é o que mais se aproxima do atual trabalho, pois também propõe o uso de uma invocação *stubless* de serviços Web. Porém, a abordagem utilizada é através do uso de padrões de modelagem (*design-patterns*) para ocultar do cliente as peculiaridades das plataformas de serviços Web, sendo restrito a serviços Web baseados no protocolo SOAP. O presente trabalho utiliza uma abordagem que permite o uso de outros protocolos e tecnologias além do SOAP, como EJB, JMS e JCA. Além disso, o protótipo desenvolvido em BUHLER *et al.* (2004) é limitado a serviços Web *rpc/literal*, não sendo capaz de invocar serviços baseados no tipo *document/literal* ou *rpc/encoded*, tipos estes que são suportados pelo atual trabalho, e dessa forma se torna mais flexível.

Em ZHAO *et al.* (2006), é descrito um *framework* de invocação de serviços que tem a finalidade de invocar serviços Web sob demanda. Porém, sua abordagem é baseada na utilização da geração de um *proxy* que é armazenado em um *cache* e sempre utilizado quando o serviço precisar ser invocado. Essa abordagem é vantajosa no sentido de que dessa forma a invocação se torna mais rápida, pois não há a necessidade da leitura e criação dos procedimentos de invocação sempre que o serviço precisa ser invocado. Porém, apesar da abordagem que é proposta pelo presente trabalho não utilizar *proxies* ou *caches*, tem a vantagem de se adequar melhor em ambientes voláteis e dinâmicos, onde a manutenção dos *proxies* se torna complexa.

O artigo de SANTOS *et al.* (2006) é voltado principalmente para a composição de serviços Web, onde não são tratados assuntos voltados para a interoperabilidade especificamente, mas sim descrevendo os principais desafios para se atingir uma composição dinâmica de serviços Web. Muitas das idéias propostas neste artigo motivaram o desenvolvimento da abordagem proposta pelo presente trabalho, como por exemplo, a necessidade da criação de métodos que localizem e invoquem serviços Web de maneira mais dinâmica e automática, sem necessidades e interfaceamentos manuais.

2.5 Sumário do capítulo

Este capítulo apresentou o atual estado da arte e das tecnologias relacionadas com o tema de pesquisa desta dissertação bem como um levantamento dos principais trabalhos e projetos de pesquisa relacionados.

Na seção 2.1 são introduzidos os conceitos de serviços e da Arquitetura Orientada a Serviços. O paradigma de orientação a serviços tem o potencial para prover uma maior integração, reusabilidade e principalmente uma maior interoperabilidade entre diferentes aplicações heterogêneas. A forma que tem sido largamente utilizada para a implementação da filosofia SOA é através das especificações e padrões dos serviços Web.

A seção 2.2 apresenta detalhes das tecnologias e padrões mais utilizados pelos serviços Web, e que foram largamente utilizadas para o desenvolvimento da abordagem proposta por este trabalho. Foram apresentadas também as metodologias de desenvolvimento de serviços Web, e das diferentes formas de invocação desses serviços, e ao mesmo tempo dissertando sobre algumas considerações sobre como atingir uma melhor interoperabilidade entre diferentes serviços Web.

Conforme visto na seção 2.3, os trabalhos e projetos relacionados com interoperabilidade tratam de diferentes aspectos, que englobam diferentes níveis dentro de uma organização, como a interoperabilidade entre sistemas de TIC, bem como uma interoperabilidade de mais alto nível, como questões de semântica e de processos de negócios. Os projetos de pesquisa ATHENA e INTEROP-NOE são os principais que tratam de interoperabilidade, onde o ATHENA é um projeto mais genérico que trata de diferentes tipos de interoperabilidade, e o INTEROP-NOE é mais voltado para os sistemas de TIC e a forma como estes podem ser projetados para atingir uma melhor interoperabilidade.

CAPÍTULO 3
Interoperabilidade

Atualmente, a forma como os negócios eram conduzidos a uma década atrás já não é mais aceitável se a organização deseja se manter competitiva. As organizações precisam ser cada vez mais flexíveis em resposta às rápidas mudanças no mercado e à grande competitividade entre organizações (GOLD-BERNSTEIN *et al.*, 2006).

De acordo com TSAGKANI (2005), as organizações precisam **colaborar** entre si, criando uma rede colaborativa de organizações (RCO), que possui a habilidade de reagir mais dinamicamente e de forma mais ágil, de acordo com as necessidades de fornecedores, parceiros de negócios ou clientes.

Porém, isso implica em uma boa comunicação entre os sistemas de informação e também em uma boa compatibilidade entre seus sistemas, modelos e processos de negócio. Para isso, é preciso que sejam criadas formas de se padronizar e harmonizar essa comunicação e compatibilidade, de forma a atingir a **interoperabilidade**.

Os serviços Web, que seguem o paradigma SOA, provêem abstrações e padrões baseados em XML que simplificam a interoperabilidade nos níveis mais baixos, isto é, protocolos e formato de dados. Porém, a integração deve ser atingida também em um nível mais alto, chamado de camada de colaboração (TSAGKANI, 2005), que lida com a localização de novos parceiros e formas de invocação, de forma a atingir a colaboração plena.

Esse capítulo tem o intuito de apresentar as definições e conceitos de interoperabilidade, e apresentar alguns trabalhos acadêmicos voltados especificamente para a área de interoperabilidade. O escopo é principalmente técnico e de serviços, que é o foco dessa dissertação.

3.1 Definições

A interoperabilidade entre sistemas não é um problema relativamente novo na tecnologia da informação, possuindo diversas definições na literatura. Algumas definições mais relevantes e adequadas ao tema desse trabalho são:

- A habilidade de dois ou mais sistemas ou componentes compartilharem informações e usar essa informação que está sendo compartilhada (IEEE, 1991).
- A habilidade dos sistemas de Tecnologia da Informação e Comunicação e de os seus processos de negócio suportados trocarem dados e possibilitar o compartilhamento de informação e conhecimento (IDABC1, 2004).
- A habilidade de transferir e usar informações de múltiplas organizações e diferentes sistemas de tecnologia de informação, de uma maneira uniforme e eficiente (ABLONG *et al.*, 2005).
- Habilidade de ocultar a heterogeneidade dos desenvolvedores de aplicações que utilizam e oferecem serviços Web (ALMEIDA *et al.*, 2002).

Baseado das definições apresentadas acima, é possível definir **interoperabilidade** no âmbito desse trabalho, como a habilidade de dois ou mais **serviços**, desenvolvidos em diferentes organizações com diferentes plataformas, invocarem operações e trocarem informações, ou seja, colaborarem, de maneira **transparente** e **sem costuras**, independente de qual plataforma ou linguagem de programação esse serviço tenha sido desenvolvido. O termo “transparente” nesse contexto se refere ao lado do consumidor do serviço, que não precisa ter nenhum prévio conhecimento do sistema ou da estrutura que o suporta. Já o termo “sem costuras” se refere à integração de diferentes infra-estruturas e serviços pertencentes a diferentes organizações sem qualquer tipo de suporte adicional ou interfaceamentos manuais (ALONSO *et al.*, 2004).

3.1.1 Integração vs. Interoperabilidade

É importante ressaltar a diferença entre integração e interoperabilidade, no âmbito de aplicações e sistemas. A integração é o processo de composição ou combinação de subsistemas para formar um único sistema (FISHER, 2006). A integração é um processo hierárquico, que utiliza um controle centralizado e uma visão global para fazer com que

sistemas que foram desenvolvidos independentemente trabalhem em conjunto, utilizando uma estrutura previamente definida, com funções declaradas e fixas para cada componente do sistema.

Na integração, assume-se que os sistemas têm fronteiras definidas, toda a informação relevante e necessária para a integração está disponível para acesso e os requisitos são conhecidos e fixos (FISHER, 2006). Porém, os sistemas computacionais têm se tornado maiores e mais complexos, onde as formas de se fazer negócios são mais dinâmicas e além das fronteiras internas da organização, e com isso essas suposições já não são mais tão realistas.

Diferentemente da integração, a interoperabilidade diz respeito à capacidade de interação entre diferentes sistemas autônomos, colaborando entre si, de forma a atingir objetivos específicos. A interoperabilidade é desejada em um processo colaborativo, distribuído entre diferentes organizações, onde não existe um controle centralizado e sem configurações prévias dos sistemas envolvidos (FISHER, 2006). A integração diz respeito a uma ação a ser tomada, enquanto que a interoperabilidade é uma propriedade de um sistema (SOBERNIG *et al.*, 2005).

3.2 Níveis de interoperabilidade

A colaboração pode ser considerada um acordo entre participantes de um negócio[22] que visa atingir um objetivo ou resultado em um dado processo em comum (TSAGKANI, 2005). Na colaboração, as entidades participantes compartilham informações, recursos e responsabilidades para, em conjunto, planejar, implementar e avaliar as atividades para atingir um objetivo em comum (CAMARINHA-MATOS *et al.*, 2006).

A partir do momento que dois ou mais participantes passam a colaborar, é preciso que exista a capacidade de todos os envolvidos de interoperarem entre si, ou seja, se comunicarem, trocarem informações e invocarem operações uns dos outros, independentemente das arquiteturas, aplicações e semânticas utilizadas.

Há inúmeras definições e classificações para a interoperabilidade encontradas na literatura, dependendo da ênfase e propósito a serem dados. Neste trabalho, considerando-

[22] Organizações, empresas, serviços, pessoas

se um foco mais direcionado a serviços e ao domínio de aplicação de interoperabilidade entre empresas, optou-se pela classificação de Berre (BERRE *et al.*, 2004), que categoriza a interoperabilidade em três tipos, ou níveis: *técnica*, *semântica* e *pragmática*.

3.2.1 Interoperabilidade técnica

Nesse nível, sistemas e serviços são capazes de trocar mensagens e invocar operações, sem necessidade de qualquer tipo de intervenção ou auxílio. Esse tipo de interoperabilidade é desejado ao se interagir com sistemas e aplicações heterogêneas. Nos serviços Web, por exemplo, deseja-se que diferentes serviços, desenvolvidos em diferentes plataformas possam interoperar. Isso se torna possível através da conformidade com os perfis da WS-I (BALLINGER *et al.*, 2004), e também com ferramentas que possibilitem a invocação de operações/serviços de maneira dinâmica e transparente.

3.2.2 Interoperabilidade semântica

Nesse tipo de interoperabilidade, o conteúdo da mensagem deve ser compreendido da mesma maneira por todas as partes envolvidas na comunicação. Ou seja, deve ser possível que sistemas e aplicações compartilhem e troquem informações de sistemas heterogêneos, dentro e entre organizações, sem a necessidade de adequações manuais adicionais para tornar isso possível (LILLENG, 2005). No âmbito dos serviços Web, isso tem se tornado possível através do mapeamento de ontologias (BATTLE *et al.*, 2005, BURSTEIN *et al.*, 2005) e do uso de linguagens de marcação semânticas como a OWL-S (MARTIN *et al.*, 2004) e a WSML (BRUIJN *et al.*, 2005).

3.2.3 Interoperabilidade pragmática

Captura a disponibilidade dos parceiros para as ações necessárias para a colaboração. A disponibilidade de participar envolve a capacidade dos participantes de se executar certa operação. Nos serviços Web, isso significa que quando houver a necessidade de um consumidor invocar um serviço, este seja capaz de localizar o serviço, e ao localizá-lo, possa então executar a invocação sem esforços adicionais. O provedor do serviço também deve ser disponível no sentido de atender as requisições dos clientes, sem a necessidade de adequações adicionais para tal.

IDABC (2004) ainda propõe um outro tipo de interoperabilidade, chamado de **interoperabilidade organizacional**, que é relacionado com a organização dos processos de negócios – por exemplo, padronizar a operação de compra de produtos – e estruturas internas da organização de forma a atingir uma troca de informações e dados mais ágil e significativa. Esse tipo de interoperabilidade pode se enquadrar no nível da semântica, no sentido de um melhor entendimento dos processos de negócios entre organizações.

3.2.4 Considerações sobre os tipos de interoperabilidade

BERRE *et al.* (2004) diz que empresas podem colaborar de diferentes formas, visando um aumento da produtividade e dinamicidade dos processos. Em um cenário dinâmico como uma RCO, diferentes parceiros de negócio são desacoplados física e geograficamente, e a colaboração é mais complexa, exigindo uma interoperabilidade plena, i.e., uma interoperabilidade que envolve os três tipos citados nas seções anteriores.

Para atingir tal interoperabilidade e usufruir os potenciais benefícios das redes colaborativas, é necessário que sejam criadas e implementadas infra-estruturas mais genéricas e flexíveis, fazendo com que as organizações participantes possam definir e configurar mais agilmente suas relações com outras organizações (RABELO *et al.*, 2007).

3.3 Trabalhos relacionados com interoperabilidade

A interoperabilidade é um tema de pesquisa em diferentes contextos como integração de dados e componentes, EAI, B2B e serviços Web. Algumas soluções foram propostas para lidar com o problema da interoperabilidade, e serão apresentadas a seguir.

Uma forma de lidar com problemas de interoperabilidade são os *wrappers* (GANNOD *et al.*, 2003, WASSON *et al.*, 2004, ZHAO *et al.*, 2004). Um *wrapper* funciona como uma camada intermediária, que encapsula o serviço ou componente, criando uma nova interface mais simples para esse serviço, possibilitando a invocação de diferentes clientes heterogêneos a esse serviço. O *wrapper* irá tratar os tipos de dados e funções de entrada da mensagem, antes de entregá-la ao serviço, tornando dessa forma os serviços mais interoperáveis, independentemente das funções e tipos de dados utilizados por cada um deles.

Por exemplo, deseja-se conectar um componente C++ em uma plataforma que só suporta Java. Para tal, um *wrapper* Java é criado, que irá se conectar com o componente em C++ e tratar os dados e funções do mesmo. Os *wrappers* também são utilizados em GLATARD *et al.* (2006), para possibilitar a invocação de serviços Web em uma infra-estrutura de Grade. Esse *wrapper* facilita a migração de códigos existentes de aplicações para o formato de serviços Web, aumentando a interoperabilidade dentro da infra-estrutura de Grade.

O problema dos *wrappers* está em seu desenvolvimento que pode ser demorado e trabalhoso, e principalmente na sua manutenção, onde cada alteração no serviço ou em algum tipo de dado o qual é tratado, resulta em um novo desenvolvimento ou adequação do *wrapper* (KUSHMERICK, 2000).

Uma outra alternativa empregada para se aumentar a interoperabilidade, mais voltada para a **interoperabilidade semântica**, é através das chamadas pontes (*bridges*), ou dos tradutores (POHL, 2006), que têm a finalidade de tratar e traduzir as diferentes representações entre sistemas e usuários dos serviços, tanto no lado do consumidor, quanto do provedor dos serviços. Esse contexto de interoperabilidade semântica é de grande relevância e vem sendo tratado com muita ênfase por diversos trabalhos acadêmicos e projetos de pesquisa, conforme apresentados na seção 2.3.

É também proposto por BENATALLAH *et al.* (2005), DAVIDSSON *et al.*(2006) e RAY *et al.* (2006) o uso de adaptadores como forma de tratar problemas de interoperabilidade em sistemas e serviços Web. Um adaptador atua como um mediador, e tem a finalidade de gerenciar as interações entre um consumidor e um provedor de serviços, e vice-versa, identificando diferenças entre protocolos, formatos de dados e interfaces e tratando essas diferenças, mapeando para um formato e modelos comuns, antes do encaminhamento da mensagem ao seu destino.

A plataforma IRIS (RADETZKI *et al.*, 2004) trata o problema de interoperabilidade entre serviços Web através de um processo de mediação semi-automática, onde “unidades de transformação” são identificadas e inseridas entre os serviços Web. Essas unidades são chamadas de mediadores, que são componentes de software que implementam as facilidades de adaptação necessárias e utilização de ontologias para que os serviços interoperem.

O uso de adaptadores e mediadores é útil quando um serviço possui diversos clientes, suportando diferentes protocolos e interfaces, fazendo com que o provedor ofereça diferentes interfaces para o mesmo serviço. O problema dos adaptadores, similarmente aos *wrappers*, está em sua manutenção, pois para cada serviço que será exposto, é preciso criar um adaptador correspondente, e a cada alteração do serviço, o adaptador possivelmente precisará ser replanejado e desenvolvido. Os trabalhos de BENATALLAH *et al.* (2005) e RADETZKI *et al.* (2004) propõem a criação semi-automática de adaptadores e mediadores, baseando-se na análise das interações e nas diferenças entre os protocolos utilizando ontologias.

3.3.1 Considerações sobre trabalhos relacionados

Apesar de existirem diferentes soluções para o problema de interoperabilidade, onde diferentes técnicas e modelos são utilizados, poucos destes se encaixam no cenário de RCOs, mais precisamente dentro de uma OV.

Um cenário de RCO está em constante evolução, isto é, as organizações (membros) entram e saem de um negócio de acordo com as necessidades de colaboração, de uma forma sob demanda, ou seja, os membros não se conhecem à priori e não têm uma integração entre si, previamente definida e estabelecida. Estas características inviabilizam as soluções apresentadas acima, onde são apresentadas soluções isoladas e caso-a-caso, de acordo com a necessidade de cada organização membro de uma RCO.

Portanto, neste trabalho optou-se por desenvolver uma ferramenta que é mais aplicável ao cenário de RCOs, ou seja, uma ferramenta que auxilie na localização e invocação dinâmica e interoperável de serviços providos por diferentes organizações que utilizam diferentes plataformas para expor seus serviços Web.

3.4 Interoperabilidade nos serviços Web

Os serviços Web vêm ganhando grande atenção das comunidades de negócios e de TIC, onde fabricantes de plataformas e servidores de aplicações têm focado no desenvolvimento e no suporte aos padrões e especificações dos serviços Web, fazendo com que se aumente o número de aplicações desenvolvidas e publicadas em forma de serviços.

O principal fator que atrai a atenção e promove a expansão dos serviços Web é o fato da utilização de padrões abertos e a interoperabilidade que pode ser provida. Os serviços Web permitem que um cliente acesse ou invoque um serviço através da utilização de um conjunto definido de padrões, independente da plataforma e linguagem utilizada.

Porém a interoperabilidade não é garantida apenas adotando o mesmo conjunto de padrões. Muitos problemas de interoperabilidade podem surgir ao se invocar os serviços Web entre diferentes aplicações rodando em diferentes plataformas, devido a implementações personalizadas dos padrões por parte dos fabricantes de servidores de aplicação e plataformas para serviços Web (COHEN, 2002). Isso ocorre, pois as especificações dos padrões são extensas e não são muito explícitas quando se diz respeito a como os padrões devem ser implementados, fazendo com que cada fabricante fique livre para implementar os padrões da maneira que desejar.

3.4.1 Padrões abertos como forma de alcançar a interoperabilidade

A forma de se obter uma maior interoperabilidade, mantendo a liberdade de escolha da tecnologia, se faz através da utilização de padrões abertos na definição dos protocolos e interfaces dos componentes envolvidos (IDABC2, 2003).

Os padrões abertos são especificações documentadas, contendo especificações técnicas ou outros critérios precisos para serem usados, regras, recomendações ou definições de características para garantir que serviços, processos e produtos atinjam ao seu propósito (BRYDEN, 2003). Estas especificações são declaradas padrão por um órgão independente da indústria[23] ou são um padrão *de facto*, e são disponibilizadas de forma gratuita, para utilização e implementação. Os padrões abertos promovem a liberdade de escolha entre produtos e soluções de diferentes fabricantes. Com a utilização de padrões abertos, existe uma maior interoperabilidade entre produtos de diferentes fabricantes, que implementem os mesmos padrões. Os serviços Web podem ser citados como um exemplo de tecnologia que é composta de padrões abertos.

[23] Esses órgãos são conhecidos como Organização de Desenvolvimento de Padrões e podem ser definidos como uma entidade sem fins lucrativos, cujas atividades incluem desenvolver, coordenar, revisar, promulgar e manter padrões de interesse público. Exemplos: W3C, OASIS.

O fato de diferentes fabricantes, como IBM, BEA e Microsoft, utilizarem padrões abertos na implementação de suas plataformas e servidores de aplicação para serviços Web, não significa que os mesmos irão necessariamente interoperar.

De acordo com NO-REST (2004) e EGYEDI (2006), isso pode ser causado pela forma como os padrões são criados. O processo pelo qual uma tecnologia é submetida para se tornar um padrão normalmente é o mesmo: Um comitê inicia uma idéia, adota essa idéia como um item de trabalho e então essa idéia é submetida a diferentes e sucessivos processos de padronização, gerando então a especificação final do padrão, que será implementado em um produto ou serviço. Porém, alguns problemas podem surgir, como por exemplo:

- Uso de diferentes terminologias em uma especificação do padrão, podendo causar problemas de interpretação, implementação e consequentemente de interoperabilidade.
- Falta de acompanhamento, análise e revisão da evolução do padrão.
- Processo de implementação de baixa qualidade, levando a uma conformidade parcial com o padrão, causando problemas de interoperabilidade.

Devido a esses problemas e a outras necessidades e funcionalidades que precisam ser incluídas nos padrões, diversas “versões” de padrões são criadas, com especificações mais complexas, o que faz com que sistemas, plataformas e *frameworks* para serviços Web não interoperem entre si, apesar de implementarem nominalmente os mesmos padrões.

Visando solucionar esse problema, foi criada a WS-I, que é a organização responsável por promover a interoperabilidade entre diferentes implementações dos serviços Web por diferentes fabricantes e desenvolvedores.

3.4.2 Organização WS-I

A WS-I (*Web Services Interoperability Organization*) é uma organização aberta da indústria, suportada por diversos fabricantes como a IBM, Microsoft, Sun, Oracle, HP, entre outros, e é aberta a qualquer organização comprometida com a interoperabilidade e que se baseie em definições já aceitas pela indústria e com suporte a padrões abertos. Essa iniciativa foi criada para acelerar o desenvolvimento e promover a interoperabilidade dos serviços Web entre diferentes plataformas, sistemas operacionais, aplicações e linguagens

de programação (WS-I, 2007). A organização promove uma interoperabilidade consistente e confiável entre serviços Web de diferentes plataformas, processos de negócio e linguagens de programação.

Os grupos de trabalho da WS-I são encarregados de produzir *deliverables*, que fornecem direcionamentos, recomendações, recursos e ferramentas de suporte. Esses *deliverables* são criados principalmente para auxiliar os desenvolvedores a criar serviços Web interoperáveis, e validar os serviços, verificando se estão em conformidade tanto com os padrões da indústria como com as recomendações da WS-I. Os *deliverables* principais da WS-I incluem:

- os *profiles*, que são especificações de serviços Web já existentes, e convenções sobre como estas interoperam;
- implementações exemplo, que apresentam questões de interoperabilidade;
- direcionamentos para a implementação, com cenários de implementação, soluções de exemplo, e casos de teste ilustrando a verificação de conformidade;
- uma ferramenta de monitoramento, que intercepta e arquiva interações entre serviços Web;
- uma ferramenta de análise, que varre os arquivos salvos pelo monitor para verificar se a implementação do serviço Web está em conformidade.

Profiles

As especificações mais recentes para se descrever, publicar, descobrir e invocar serviços Web são o SOAP 1.2, WSDL 1.1 e UDDI 3.0 (AUSTIN *et al.*, 2004, OASIS, 2007). Cada uma das especificações é baseada na linguagem XML e, em particular, no XML *Schema*. Porém, novas especificações vêm sendo criadas, como anexos binários (SOAP *with Attachments*), roteamento (WS-*Routing*), transações (WS-*Transaction*), fluxo de processos (WS-BPEL), segurança (WS-*Security*), etc.

Devido ao grande número de especificações, que podem ser necessárias durante o desenvolvimento de um serviço Web, se torna uma tarefa muito difícil para um desenvolvedor ou usuário determinar as especificações que são suportadas por um determinado produto ou serviço Web, e até que nível essas especificações são suportadas.

A WS-I então criou o conceito de *profiles* para solucionar ou auxiliar nesse problema. Um *profile* é um grupo de especificações de serviços Web em um nível de versão específico, juntamente com convenções sobre como eles podem trabalhar em conjunto (WS-I, 2002).

A WS-I, após um período de análise em identificar os diversos problemas de interoperabilidade presentes nos protocolos padrão utilizados para descrever, publicar e invocar serviços Web, publicou seu primeiro *profile*, chamado de *Basic Profile 1.0*. Esse *profile* contém uma lista dos padrões básicos dos serviços Web, que inclui o SOAP 1.1, WSDL 1.1, UDDI 2.0, XML 1.0, e XML *Schema* partes 1 e 2, juntamente com direcionamentos, recomendações e esclarecimentos sobre esses padrões de forma a promover uma maior interoperabilidade (BALLINGER *et al.*, 2004). Essas recomendações fazem uso de palavras chave, como por exemplo, “MUST”, “MUST NOT”, “SHOULD”, entre outras, definindo assim os requisitos a serem implementados pela especificação de forma que se obtenha um grande nível de interoperabilidade entre os serviços.

Os *profiles* fazem parte dos chamados *deliverables* da WS-I, que também incluem cenários de exemplo, aplicações de exemplo, e ferramentas de teste que verificam se um serviço Web, ou seja, suas mensagens, WSDL e componentes UDDI, estão em conformidade com os direcionamentos do *profile*.

Cenários de uso

Os cenários de uso do *Basic Profile* definem como os serviços Web são usados em diferentes tipos de interações entre o consumidor e o provedor do serviço, identificando também os requisitos básicos de interoperabilidade para essas interações e as mapeia para os requisitos descritos pelo *Basic Profile* (WERDEN *et al.*, 2002). Os cenários refletem requisitos fundamentais e reais dos serviços Web, porém são independentes de qualquer domínio de aplicação.

A WS-I define três cenários de uso para complementar o *Basic Profile*, chamados de *one-way*, requisição/resposta síncrono e *basic callback*. Esses cenários são apresentados em detalhes a seguir (WERDEN *et al.*, 2002).

Cenário de uso *One-Way*

Nesse cenário, o consumidor envia uma mensagem ao provedor. O provedor recebe a mensagem e a processa, mas não gera a resposta de retorno. A Figura 15 ilustra esse caso:

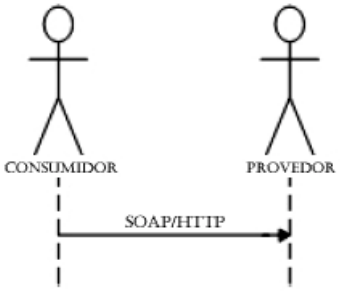


Figura 15: Cenário de uso *One-Way*

Cenário de uso Requisição/Resposta Síncrono

Nesse cenário, o consumidor primeiramente envia uma mensagem SOAP ao provedor. O provedor então processa a mensagem e envia a resposta de retorno. A Figura 16 ilustra esse caso:

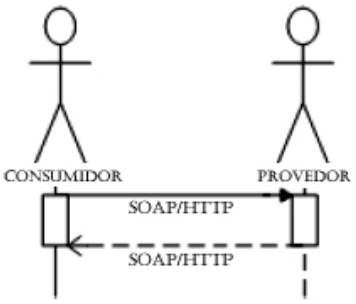


Figura 16: Cenário de uso Requisição/Resposta Síncrono

Cenário de uso *Basic Callback*

Esse cenário utiliza um tipo de troca de mensagens assíncrona para os serviços Web utilizando dois pares de mensagens síncronas de requisição/resposta. Na execução, o consumidor envia a requisição SOAP inicial ao provedor, que retorna um *ACK* de recebimento do pedido. Após isso, em algum momento no tempo, o provedor irá enviar a resposta de retorno para o pedido inicial feito pelo consumidor. Isso é ilustrado na Figura 17:

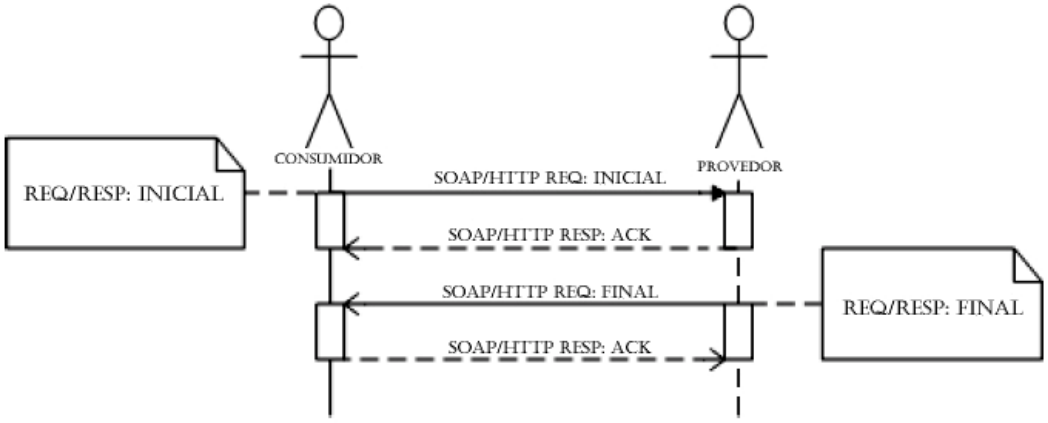


Figura 17: Cenário de uso *Basic Callback*

Considerações sobre os Cenários

Os cenários apresentados acima são apenas seqüências de eventos em tempo de execução, e não envolvem nenhuma atividade de desenvolvimento e publicação. O modelo de dados e os protocolos de transporte são todos definidos e implementados a priori e as mensagens de requisição/resposta são sincronizadas através do protocolo HTTP. É importante ressaltar que é possível criar composições desses cenários, ou seja, utilizá-los como base para a construção de cenários mais complexos. Todas as partes desses cenários são definidas em conformidade com os direcionamentos e recomendações do *Basic Profile*.

Aplicações de exemplo

A WS-I também provê uma aplicação de exemplo, que demonstra todos os cenários do *Basic Profile*, além de demonstrar como este permite que serviços Web de diferentes fabricantes interoperem.

A aplicação de exemplo é um gerenciamento de cadeia de suprimentos (SCM) que modela uma venda de eletrônicos por um revendedor que possui três armazéns, e que negocia com três fabricantes. É possível escolher entre implementações do serviço de diferentes fabricantes para cada um dos armazéns e fabricantes para observar como o *Basic Profile* permite que os serviços Web interoperem (CHAPMAN *et al.*, 2003).

Ferramentas de teste

A WS-I provê duas ferramentas de teste de interoperabilidade, um monitor e um analisador, que são utilizados em conjunto para avaliar se um serviço Web segue ou não os direcionamentos do *Basic Profile*. Ambas são disponibilizadas em versões em Java e C#.

O monitor funciona como um intermediário entre o consumidor e o provedor de serviços Web, capturando e gravando as mensagens SOAP que são trocadas entre si. O analisador examina essas mensagens gravadas pelo monitor, e reporta se o serviço está ou não em conformidade com os direcionamentos e recomendações do *Basic Profile* (CLUNE *et al.*, 2005, SEELY *et al.*, 2005).

A Figura 18 ilustra em detalhes a relação entre o consumidor, o provedor, o monitor e o analisador:

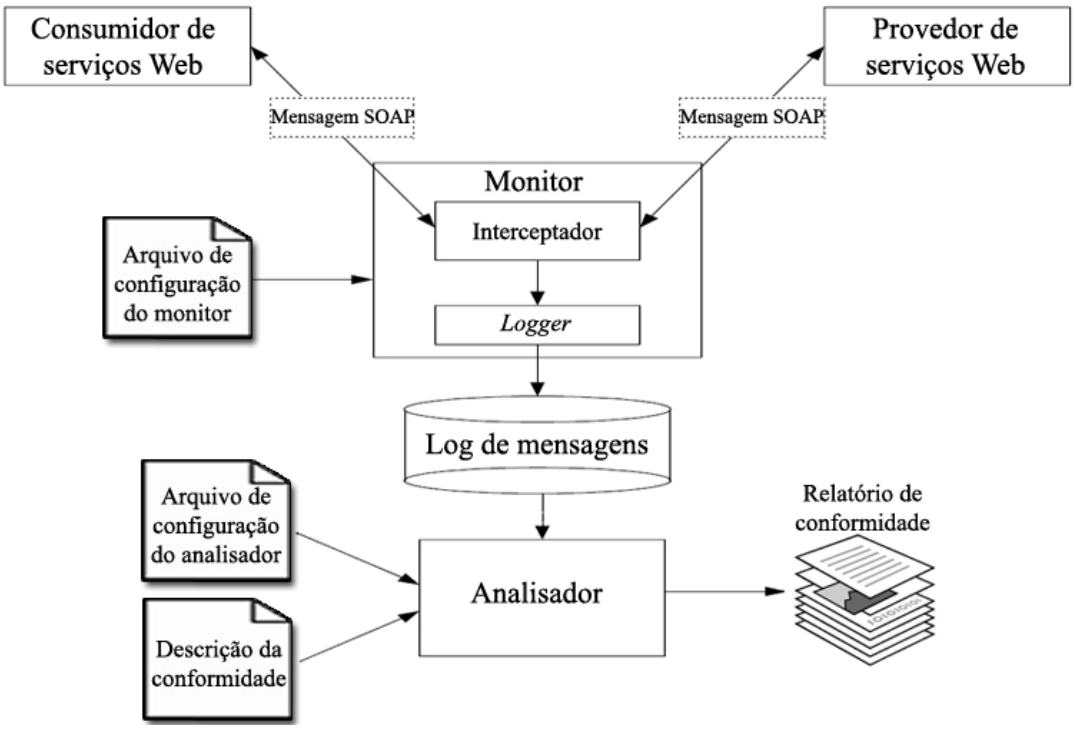


Figura 18: Ferramentas de teste do WS-I.

É importante ressaltar que existe um relacionamento entre os *profiles* e as ferramentas de teste. Os *profiles* servem como um grupo de regras (descrições da conformidade) que as ferramentas de teste irão utilizar enquanto analisam as mensagens entre serviços Web, e geram os relatórios de conformidade. Abaixo, as ferramentas de teste são descritas com mais detalhes.

Monitor

O monitor é uma ferramenta de teste da WS-I que captura mensagens e as grava em um arquivo, chamado de *Log*. Atualmente, o monitor suporta apenas mensagens SOAP sobre HTTP, porém futuras versões irão suportar outros tipos de protocolos e transportes (SEELY *et al.*, 2005).

O monitor é projetado como um interceptador que fica situado entre o consumidor e o provedor de serviços – chamado de *man in the middle* – e é responsável por capturar as mensagens e encaminhá-las ao destino, de acordo com as configurações previamente definidas. A Figura 19 ilustra o arquivo de configuração do monitor:

```
<?xml version="1.0" encoding="utf-8" ?>
<wsi-monConfig:configuration
                    xmlns:wsi-monConfig="http://www.ws-i.org/testing/2003/03/monitorConfig/">
 <wsi-monConfig:logFile replace="true" location="arquivo_de_log.xml"/>
 <wsi-monConfig:manInTheMiddle>
  <wsi-monConfig:redirect>
   <wsi-monConfig:listenPort>9090</wsi-monConfig:listenPort>
   <wsi-monConfig:schemeAndHostPort>http://localhost:8080</wsi-monConfig:schemeAndHostPort>
  </wsi-monConfig:redirect>
 </wsi-monConfig:manInTheMiddle>
</wsi-monConfig:configuration>
```

Figura 19: Arquivo de configuração do monitor

Na figura acima, o monitor está configurado para escutar por conexões na porta 9090. Ao receber uma requisição nesta porta, o monitor irá gravar essa requisição no arquivo *arquivo_de_log.xml* e em seguida irá encaminhar a requisição para o serviço propriamente dito, localizado no *localhost* na porta 8080.

Analisador

O propósito principal do analisador é o de validar as mensagens que foram previamente gravadas pelo monitor, dizendo se está ou não em conformidade com as recomendações do *Basic Profile*. Além das mensagens gravadas pelo monitor, o analisador também valida a descrição WSDL de um serviço. A validação é feita baseando-se no arquivo de configuração do analisador, e também no arquivo de descreve a conformidade de um serviço Web, isto é, as regras que definem se um serviço está ou não em conformidade com o *Basic Profile*. A Figura 20 ilustra o arquivo de configuração do analisador.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsi-analyzerConfig:configuration name="Arquivo de Configuracao do Analisador"
    xmlns:wsi-analyzerConfig="http://localhost/temp/">
  <wsi-analyzerConfig:assertionResults type="all" messageEntry="true" failureMessage="true"/>
  <wsi-analyzerConfig:reportFile replace="true" location="relatorio_de_conformidade.xml"/>
  <wsi-analyzerConfig:testAssertionsFile>BasicProfileTestAssertions.xml
  </wsi-analyzerConfig:testAssertionsFile>
  <wsi-analyzerConfig:logFile correlationType="endpoint">arquivo_de_log.xml
  </wsi-analyzerConfig:logFile>
</wsi-analyzerConfig:configuration>
```

Figura 20: Arquivo de configuração do analisador

Na figura acima o analisador está configurado para validar o arquivo *arquivo_de_log.xml*, gerado previamente pelo monitor, utilizando o arquivo de descrição de conformidade *BasicProfileTestAssertions.xml*. Após a análise, será gerado um relatório final de conformidade com o nome *relatório_de_conformidade.xml*.

3.4.3 Recomendações do *Basic Profile* da WS-I

O *Basic Profile* da WS-I provê recomendações no sentido de se desenvolver serviços Web mais interoperáveis. A grande maioria dessas recomendações é voltada para os fabricantes de ambientes e plataformas para serviços Web, para que possam fornecer ferramentas – como os motores SOAP, geradores e analisadores de WSDL, entre outros – mais interoperáveis e em conformidade com o *Basic Profile*.

Porém, a simples adoção do *Basic Profile* por diferentes fabricantes não significa que os serviços desenvolvidos em suas plataformas serão interoperáveis com outras plataformas e tecnologias. Muitos fabricantes oferecem suporte a funcionalidades de serviços Web que não estão em conformidade com o *Basic Profile*, incluindo detalhes proprietários que podem causar problemas de interoperabilidade com outras plataformas (ANDRADE ALMEIDA *et al.*, 2003).

Portanto, algumas seções do *Basic Profile* são relevantes também para um desenvolvedor de serviços Web, especificamente as que tratam da implementação de serviços Web interoperáveis. A seguir são apresentadas algumas seções mais importantes relacionadas ao desenvolvimento de serviços Web interoperáveis:

- **Seção 4:** Relacionada com o SOAP e seu uso em conjunto com o HTTP. Recomendações como a do tipo de codificação SOAP utilizada e do cabeçalho SOAP, devem ser levadas em conta pelo desenvolvedor.
- **Seção 5:** Diz respeito à forma como o documento WSDL deve ser criado, incluindo detalhes de cada um dos elementos do WSDL. Esta seção é altamente recomendada, principalmente para desenvolvedores que utilizam a abordagem *top-down*, descrita na seção 2.2.7.
- **Seção 6:** Descreve a forma como publicar e descobrir serviços Web em um registro UDDI de maneira interoperável.
- **Seção 7:** Relacionado com questões de segurança nos serviços Web. Atualmente a WS-I tem um *Profile* específico para questões de interoperabilidade em serviços Web seguros.

Muitas das recomendações do *Basic Profile* são acompanhadas de exemplos de mensagens SOAP ou descrições WSDL que demonstram tanto a forma correta (em conformidade) quanto a forma incorreta de implementação.

O *Basic Profile* é, no entanto, apenas um dos *deliverables* que são disponibilizados pela WS-I. Atualmente existem em andamento diferentes *profiles* sendo desenvolvidos, como o *AttachmentsProfile*, que inclui especificações sobre como enviar anexos em uma mensagem SOAP de maneira interoperável, e o *Basic Security Profile*, que especifica a forma de envio de mensagens SOAP seguras e interoperáveis. A organização WS-I tende a evoluir juntamente com as especificações dos serviços Web, criando novos *Profiles* que tratam de novas funcionalidades, como, por exemplo, as extensões dos serviços Web, apresentadas na seção 2.2.6.

3.5 Sumário do capítulo

Este capítulo apresentou de forma mais detalhada a interoperabilidade, suas principais definições, diferentes níveis, soluções que têm sido propostas para atingir uma maior interoperabilidade e detalhes sobre a principal organização responsável por recomendações acerca da interoperabilidade entre serviços Web.

A seção 3.3 apresenta algumas soluções que já foram propostas para tratar do problema da interoperabilidade, com ênfase nos serviços Web. Estas soluções tratam de

todos os níveis da interoperabilidade, i.e., técnica, semântica e pragmática, que incluem mediações, *wrappers,* notações e traduções semânticas entre serviços. Estas soluções, principalmente soluções voltadas para a semântica, podem ser aplicadas em conjunto com a abordagem proposta por esse trabalho, englobando diferentes níveis de interoperabilidade.

A seção 3.4 apresenta detalhes sobre a organização WS-I, que é a principal organização voltada para a interoperabilidade entre serviços Web. A WS-I provê recomendações, cenários de uso, ferramentas de teste e aplicações de exemplo, voltadas para uma implementação mais interoperável das especificações dos serviços Web.

As ferramentas de teste da WS-I – monitor e analisador – foram utilizadas no capítulo 5, para auxiliar na validação dos diferentes serviços Web que foram desenvolvidos, garantindo sua interoperabilidade, independentemente da plataforma e da linguagem na qual este tenha sido desenvolvido.

CAPÍTULO 4
Modelo Conceitual

4.1 Introdução

Um modelo conceitual tem o objetivo de expor os relacionamentos significantes entre as entidades de algum ambiente, habilitando o desenvolvimento de arquiteturas concretas, utilizando padrões ou especificações que suportam esse ambiente. Um modelo conceitual consiste de um conjunto de relações e conceitos dentro de um domínio de problemas em particular, e é independente de padrões, tecnologias, implementações ou outros detalhes mais concretos (MACKENZIE *et al.*, 2006).

Este trabalho propõe uma análise da interoperabilidade entre serviços Web disponibilizados em diferentes plataformas, e uma ferramenta de suporte à localização e invocação dinâmica e interoperável destes serviços. Esta ferramenta, mais especificamente, permite que em um ambiente heterogêneo, como as RCOs, os serviços Web disponibilizados por diferentes plataformas – como Websphere, Weblogic, .NET, Axis, etc. – possam ser localizados em tempo de execução e invocados de maneira flexível e interoperável, utilizando apenas a descrição WSDL do serviço.

Esta ferramenta é independente de tecnologias e da aplicação em RCOs, podendo ser aplicado em diferentes domínios, como o de serviços em *Grade*. Esse capítulo tem o objetivo de detalhar o modelo conceitual, onde a arquitetura é descrita não se atendo a nenhuma tecnologia específica. Na seção 4.6 essa arquitetura é aplicada dentro do cenário de serviços Web e RCOs.

O comportamento geral do cenário é apresentado no diagrama UML de caso de uso na Figura 21. O diagrama ilustra dois tipos básicos de atores interagindo com as funcionalidades providas, que são:

- Provedor de serviços
- Consumidor de serviços

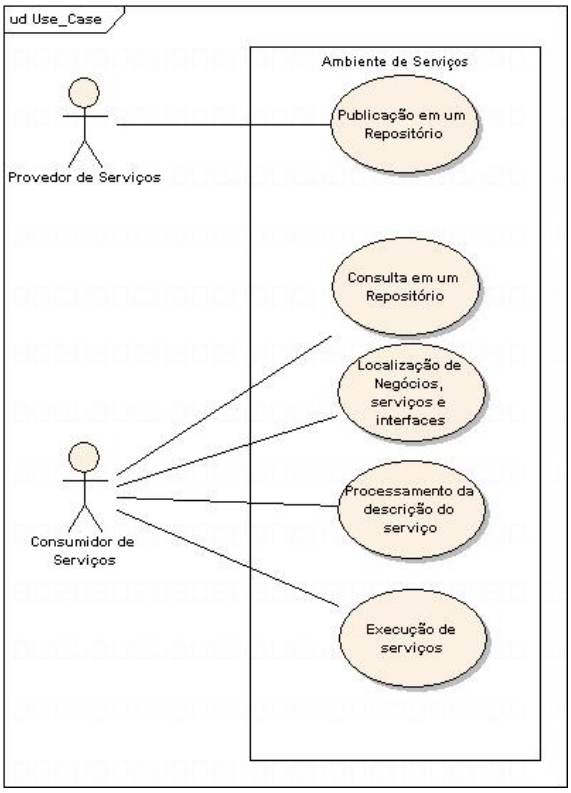


Figura 21: Caso de uso do cenário geral

4.2 Considerações iniciais

Antes de detalhar o modelo conceitual da abordagem proposta, é importante ressaltar alguns passos metodológicos que levaram à concepção deste modelo.

O objetivo inicial deste trabalho era o de avaliar a efetiva interoperabilidade entre as diferentes plataformas de serviços Web disponíveis no mercado, que fazem uso de diferentes linguagens e metodologias para a criação e disponibilização de serviços Web. Isso é feito através do monitoramento e análise do comportamento dos serviços Web e dos consumidores destes serviços em cada uma das plataformas, que são descritas no capítulo 5. Após esta análise, foi possível concluir que os grandes fabricantes de plataformas de serviços Web estão engajados em prover uma maior interoperabilidade entre as diferentes plataformas existentes no mercado, através da adoção do *Basic Profile* da WS-I.

Porém, a forma inicial de avaliação dos serviços e de sua invocação foi feita de maneira estática, onde a localização do serviço é conhecida a priori, e os *stubs* e *proxies* necessários para a efetiva invocação do serviço são criados manualmente pelo consumidor/aplicação cliente de serviços. Portanto, torna-se necessária a criação de um

modelo que oculte a complexidade tecnológica do cliente, i.e. invoque serviços Web de forma dinâmica e interoperável, sem a necessidade de interfaceamentos manuais. Isto é essencial principalmente no âmbito de redes colaborativas, que por definição são altamente dinâmicas e voláteis. Esse capítulo irá detalhar cada uma das entidades envolvidas, incluindo uma descrição dos módulos que compõem a proposta.

A Figura 22 apresenta a forma de invocação *stubless* e independente de tecnologia/ protocolos que é proposta pelo modelo da abordagem, e para fins de comparação, ilustra abaixo a forma tradicional de invocação de serviços Web:

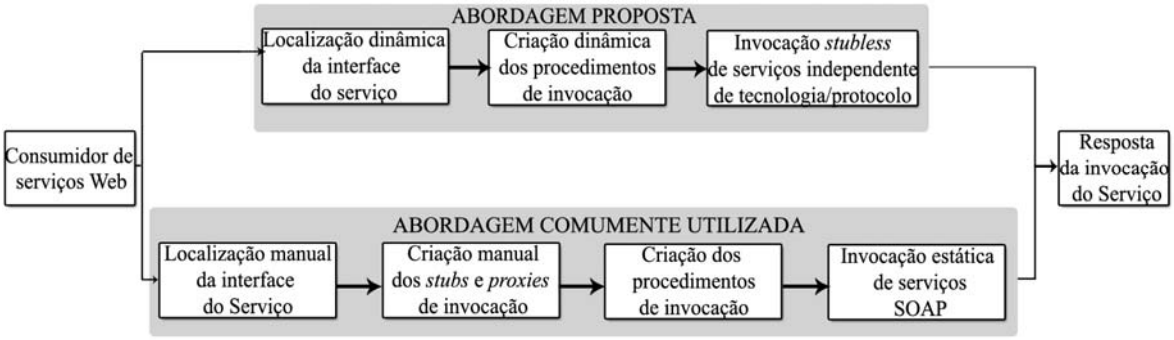


Figura 22: Comparação entre o modelo proposto e a forma comumente utilizada

4.3 Visão geral

Esta seção apresenta a arquitetura genérica e suas entidades, onde o modelo proposto será aplicado. Nas seções seguintes, serão apresentados detalhes de cada uma das entidades apresentados, bem como os componentes que fazem parte do modelo proposto.

A arquitetura é baseada no conceito de orientação a serviços, onde seu propósito é o de manter todas as entidades envolvidas, i.e., consumidores, provedores e registros, desacopladas e independentes umas das outras. É importante ressaltar que dentro dessa arquitetura, uma entidade pode ser tanto um provedor, quanto um consumidor de serviços, dependendo da perspectiva que é assumida. Do ponto de vista do cliente, o modelo da abordagem proposta pode ser vista como um *middleware* de suporte a invocação de serviços, onde consumidores/aplicações o utilizam para executar invocações de maneira flexível e dinâmica. A Figura 23 apresenta esse cenário geral.

Nas seções a seguir, cada uma dessas entidades macro serão descritas, e expandidas, apresentando cada um de seus componentes e suas funções dentro da abordagem proposta.

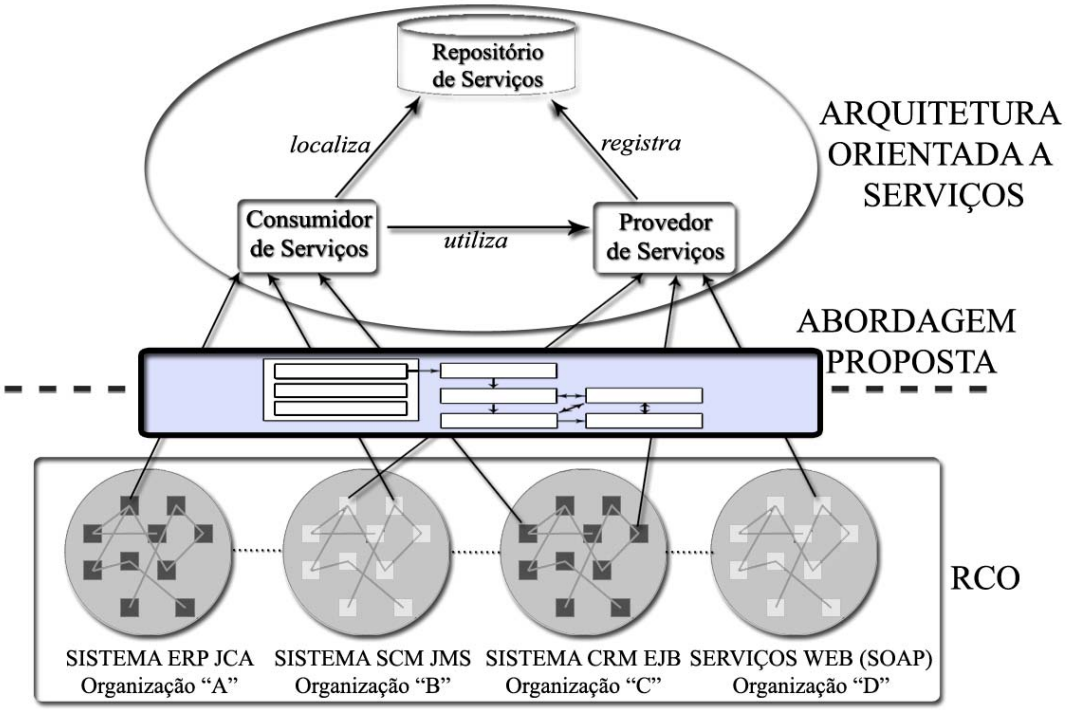


Figura 23: Visão geral do Modelo Conceitual

A abordagem proposta é apresentada na Figura 23 acima como uma camada intermediária, onde os atores de uma organização virtual podem fazer uso dos módulos ali dispostos, para publicar, localizar e invocar serviços de maneira interoperável, independente da tecnologia e protocolos envolvidos. Dos módulos que foram implementados, apenas o publicador de serviços é usado pela entidade *Provedor de serviços*. Todos os outros módulos, como a localização de serviços e a invocação *stubless* de serviços (que inclui o processamento da interface, extração de dados, seleção do tipo de serviço, criação dos procedimentos de invocação) são utilizados pela entidade *Consumidor de serviços*.

4.4 Entidades principais

4.4.1 Repositório de serviços

O Repositório é responsável pelo armazenamento das informações sobre negócios, e os serviços que são oferecidos por esses negócios, inclusive detalhes da forma como os clientes podem utilizar esses serviços. Pode ser interno a uma organização, ou acessível apenas por determinadas organizações pré-estabelecidas, ou público, onde qualquer organização pode utilizá-lo sem restrições.

Nesse Repositório, as organizações publicam suas operações e serviços, tornando-os disponíveis para que outras organizações possam localizar e executar essas operações, de acordo com suas necessidades e critérios de escolha. A segurança de acesso também é gerenciada por essa entidade, que define funções e níveis de acesso, como por exemplo, usuários com permissão apenas de consulta, inclusão, exclusão, etc.

O Repositório é responsável também pelo gerenciamento, autenticação e autorização de acesso. Gerencia as funções e os usuários, determinando se um usuário tem acesso ao Repositório, e quais são suas funções e permissões, i.e., se sua função é apenas *publicador* ou apenas *localizador*.

4.4.2 Provedor de serviços

O provedor de serviços pode ser uma organização ou empresa que deseja tornar disponível publicamente alguma funcionalidade ou processo de negócios (serviço) de forma que outras organizações possam executar essas operações. Essa entidade utiliza um repositório para disponibilizar e tornar público seus serviços, para que outras organizações o localizem.

Os serviços são desenvolvidos por uma organização, possuindo sua lógica e seqüência interna referente à forma na qual a organização executa seus processos de negócios. O Provedor torna disponível publicamente no Repositório a interface de seu serviço, que contém apenas dados relevantes à forma como outras organizações podem utilizar seu serviço, ocultando assim detalhes internos de seus processos e a forma como foram implementados.

4.4.3 Consumidor de serviços

O Consumidor de serviços consiste em uma organização ou empresa, que, ao contrário do provedor, deseja utilizar os serviços que estão disponíveis no Repositório. Para tal, o consumidor faz a consulta por serviços e interfaces no Repositório, que, ao ser localizado, retorna a interface deste serviço, i.e., instruções detalhadas de como utilizar esse serviço.

Após o recebimento da interface do serviço, cabe ao consumidor processar essa interface e criar sua lógica para acesso às funcionalidades oferecidas pelo provedor deste serviço, de maneira interoperável e dinâmica. A interoperabilidade neste caso implica que o consumidor de serviços deve executar as operações necessárias, independente da forma como e onde o serviço foi desenvolvido e disponibilizado pelo Provedor de serviços. É importante que o Consumidor de serviços se mantenha o mais desacoplado possível do Provedor, tornando-o independente em relação a alterações ou indisponibilizações que possam existir por parte do Provedor de serviços.

4.5 Módulos do modelo proposto

Na Figura 23 foi apresentada uma visão geral do Modelo Conceitual, onde a camada 3 é a abordagem proposta por este trabalho. A Figura 24 apresenta uma visão detalhada dos módulos pertencentes a este modelo.

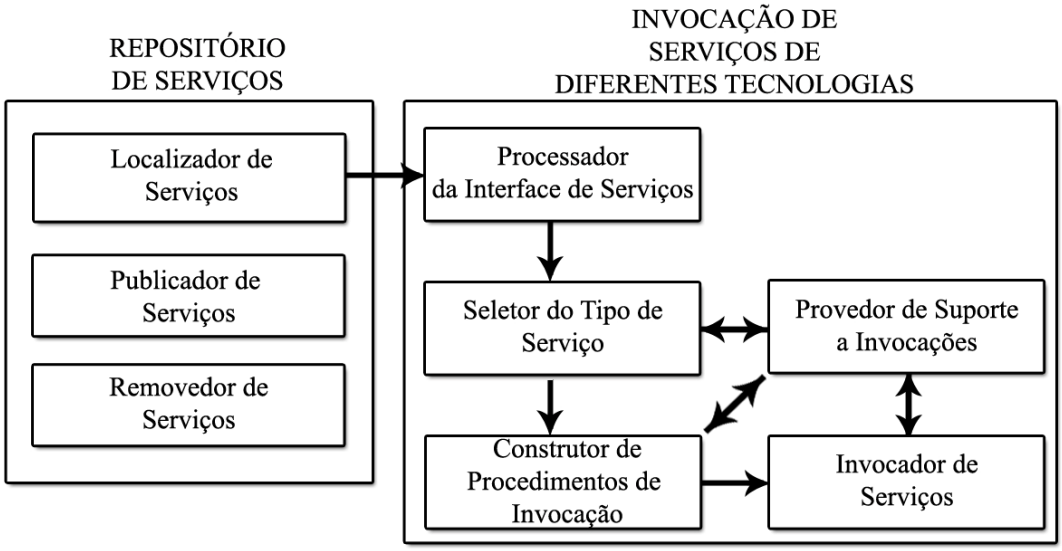


Figura 24: Módulos do modelo proposto

Nas seções seguintes, os módulos são discutidos com mais detalhes, porém sem aplicar uma tecnologia específica ou incluir detalhes de implementação.

4.5.1 Publicador de serviços

É através desse componente que o Provedor de serviços interage com o Repositório, a fim de publicar a descrição do serviço que deseja tornar visível globalmente. São também tratadas questões de segurança – ocultadas do modelo – como o envio de pedido de autenticação ao repositório.

4.5.2 Removedor de serviços

Este componente é responsável pela exclusão de serviços do Repositório. Isso é normalmente utilizado pelo Provedor de serviços quando este não deseja mais manter seu serviço disponível publicamente.

4.5.3 Localizador de serviços

O Consumidor de serviços utiliza esse componente para a localização de negócios, de serviços e interfaces em um Repositório, que retorna um ou mais itens, de acordo com os critérios de busca. Esse componente também trata questões de segurança, como pedido de autorização ao repositório para que possa executar suas buscas.

4.5.4 Processador da interface de serviços

Esse componente tem a função de acessar a descrição/interface do serviço, cuja localização foi previamente obtida pelo componente Localizador de serviços, e baseando-se na leitura dessa descrição, extrair os dados relevantes para a invocação do serviço, como, por exemplo, as operações, o tipo de serviço, os parâmetros e tipos dos dados utilizados.

4.5.5 Provedor de Suporte à invocações

O Provedor de Suporte a Invocação de serviços é o responsável por permitir que o invocador tenha a flexibilidade de executar operações de diferentes serviços, oferecidos em

diferentes tecnologias e protocolos. Possui um conjunto definido de sub-módulos de suporte que são encarregados de lidar com os procedimentos de invocação de um determinado tipo serviço.

4.5.6 Seletor do tipo de serviço

Esse componente é o responsável pela seleção do tipo de serviço que será utilizado, em um cenário com diferentes tipos de serviços e tecnologias. Através do Seletor, é possível definir qual o Provedor de Suporte será utilizado, para um determinado tipo de serviço que foi previamente extraído pelo Processador da interface.

4.5.7 Construtor de procedimentos de invocação

Após a extração do tipo de serviço pelo Processador da interface, e pela seleção do tipo de serviço e consequentemente do Suporte a serviço que será utilizado, o Construtor de procedimentos de invocação irá executar os passos necessários para que o Invocador de serviços seja capaz de executar as operações necessárias.

4.5.8 Invocador de serviços

Através desse componente são processadas e gerenciadas as invocações dos serviços que foram previamente identificados pelos componentes acima. Esse componente é responsável pela invocação propriamente dita do serviço e do tratamento do retorno da invocação.

4.6 Aplicação no cenário de serviços Web

Essa seção irá apresentar as entidades e os módulos genéricos apresentados nas seções anteriores, aplicando-os aos conceitos e nomenclaturas de serviços Web, conforme ilustrado na Figura 25. O objetivo desta seção é o de contextualizar o modelo conceitual dentro da abordagem proposta, incluindo alguns diagramas de seqüência que ilustram como as principais entidades descritas na seção 4.4 fazem uso dos módulos do modelo da abordagem. Os detalhes de implementação de cada um dos módulos descritos abaixo são apresentados no Capítulo 5.

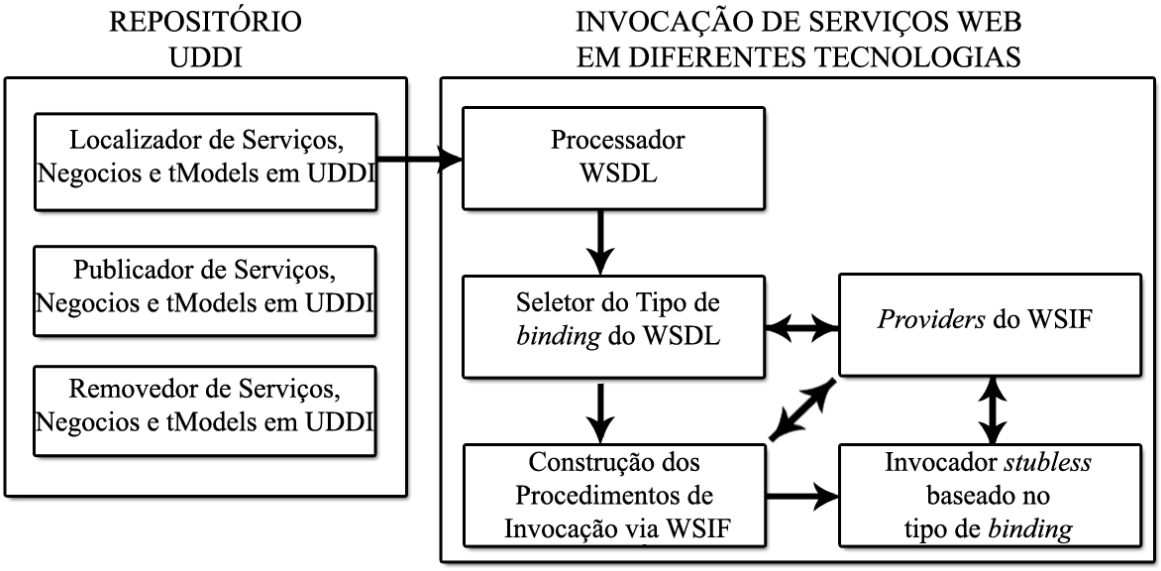


Figura 25: Módulos do modelo, aplicado ao cenário de serviços Web

4.6.1 Interações do Provedor de serviços Web

O provedor de serviços Web é o responsável pelo provimento de uma funcionalidade, e do desenvolvimento de toda a lógica que irá expor essa funcionalidade como um serviço Web. Isso inclui a criação da aplicação, ou da utilização de uma aplicação existente, e a sua exposição como um serviço Web, ou seja, a criação de uma interface WSDL, que irá descrever o serviço, e a forma como esse serviço pode ser invocado.

Cabe também ao provedor do serviço – uma organização – a publicação, em um repositório UDDI, de detalhes sobre o seu negócio, sobre os serviços que são providos por seu negócio, e também publicação da interface WSDL que será utilizada pelos consumidores de serviços para criar seus procedimentos de invocação.

O diagrama de seqüência UML da Figura 26 detalha as possíveis formas de interação do Provedor de serviços.

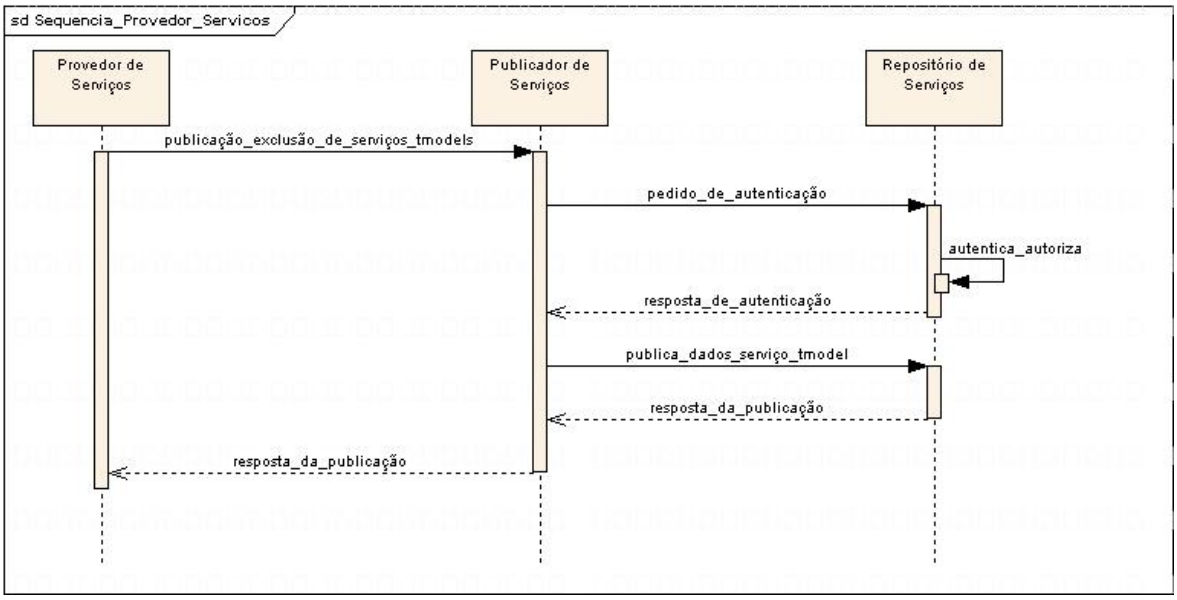


Figura 26: Diagrama de seqüência do Provedor de serviços

4.6.2 Interações do Consumidor de serviços Web

O consumidor de serviços Web pode ser qualquer organização – um usuário, uma aplicação ou outro serviço Web – que deseja invocar um serviço oferecido por outra organização. O consumidor é responsável por localizar as descrições dos serviços, i.e., sua interface WSDL, processar esta interface e extrair os dados relevantes, para então criar os procedimentos de invocação do serviço Web. Os procedimentos de invocação podem ser criados tanto estaticamente, onde são gerados os *stubs* para o serviço em tempo de desenvolvimento, quanto dinamicamente, em tempo de execução.

Em um cenário de RCO, é importante que se tenha o maior nível possível de desacoplamento entre o consumidor e o provedor de serviços, e que o consumidor seja capaz de invocar em tempo de execução, qualquer serviço Web, de acordo com sua necessidade, independentemente da plataforma na qual o serviço foi desenvolvido e disponibilizado.

O diagrama de seqüência UML da Figura 27 detalha as possíveis formas de interação do consumidor de serviços.

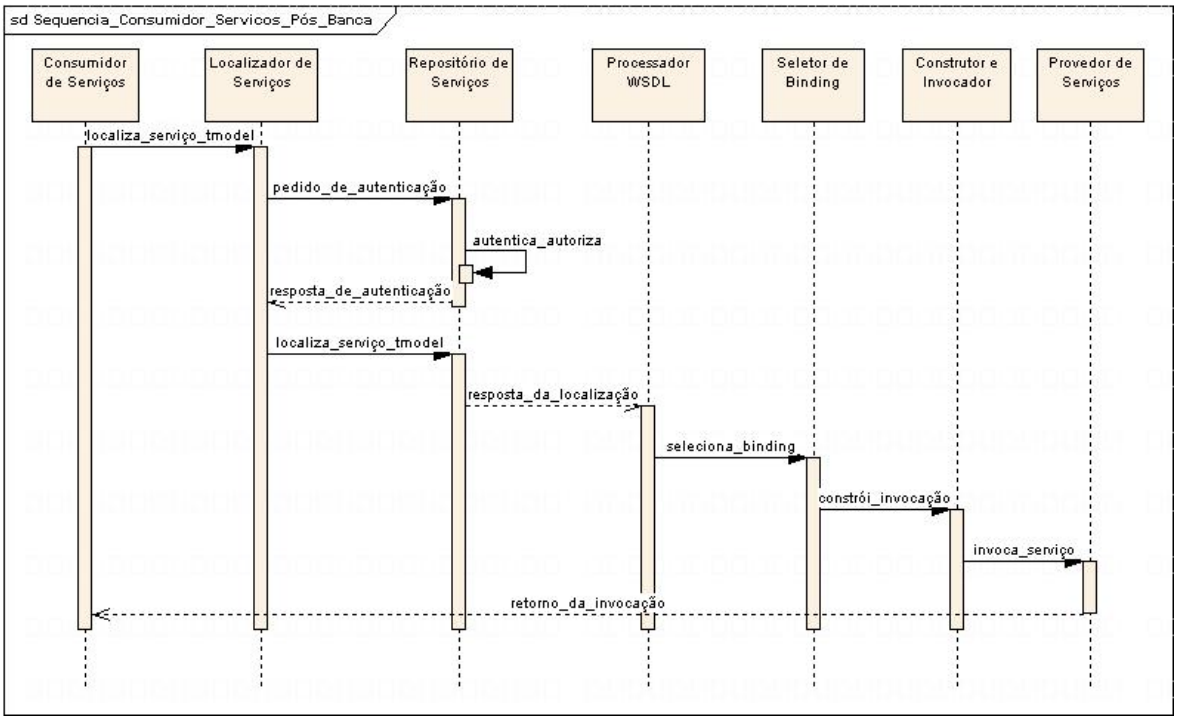


Figura 27: Diagrama de seqüência do Consumidor de serviços

4.7 Comparação entre trabalhos acadêmicos e proposta

Os trabalhos apresentados acima propõem diferentes formas de tratar o problema da interoperabilidade no desenvolvimento de software, focando principalmente nos serviços Web. A Tabela 2 apresenta de forma sumarizada as contribuições de cada um destes trabalhos, e também o enquadramento da proposta desta dissertação.

CARACTERÍSTICAS	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Foca interoperabilidade	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Foca serviços Web	Sim	Sim	Não	Sim	Sim	Sim	Sim
Solução via Protótipo de Software	Não	Não	Não	Sim	Sim	Não	Sim
Invocação sem geração prévia de stubs	-	-	-	Sim	Não	-	Sim
Localização e Invocação Dinâmica	-	-	-	Não	Sim	-	Sim
Suporte a invocação document/literal	-	-	-	Não	Sim	-	Sim
Foco na integração inter-organizacional	Não	Não	Sim	Não	Não	Sim	Sim
Foco na composição de serviços	Não	Não	Sim	Não	Não	Sim	Não

Legenda:
1 – SIDDHARTHA *et al.* (2003)
2 – KUMAR *et al.* (2004)
3 – ELVESÆTER *et al.* (2005)
4 – BUHLER *et al.* (2004)
5 – ZHAO *et al.* (2006)
6 – SANTOS *et al.* (2006)
7 – Proposta do Trabalho

Tabela 2: Comparação entre trabalhos relacionados

4.8 Considerações sobre o modelo conceitual

O modelo da abordagem proposta tem a principal finalidade de ocultar do consumidor de serviços a complexidade tecnológica envolvida na invocação de serviços Web distribuídos em diferentes plataformas e desenvolvidos em diferentes linguagens. Conforme dito na seção 3.1, a interoperabilidade, neste trabalho, pode ser definida como a habilidade de duas organizações colaborarem de maneira transparente e sem costuras entre si, isto é, de localizar e invocar serviços independentemente de qual plataforma tenha sido desenvolvido. Portanto, é possível concluir que ao se ocultar a complexidade tecnológica e agilizar o processo de localização e invocação de serviços, conseqüentemente aumenta-se a interoperabilidade.

Foi possível atingir uma maior transparência e independência de tecnologia através do desenvolvimento de um protótipo de software que utiliza APIs específicas (como, por exemplo, o Apache WSIF descrito no próximo capítulo), voltadas para a invocação de serviços Web de maneira dinâmica e mais flexível, pois permite que se localize e invoque serviços desenvolvidos em diferentes tecnologias e protocolos, com base apenas na descrição WSDL deste serviço. Desta forma, os problemas de interoperabilidade na invocação de serviços sob-demanda são abstraídos, permitindo uma maior colaboração entre diferentes organizações na utilização de serviços.

O invocador *stubless* atua principalmente na entidade do Consumidor de serviços, onde servirá de suporte para a aplicação cliente utilizar diferentes serviços providos por plataformas heterogêneas, de maneira dinâmica e interoperável. Através deste, é possível a localização de serviços, o processamento da interface, a extração dos dados relevantes a invocação, e por fim a criação dos procedimentos de invocação do serviço de acordo com o tipo de serviço descrito no WSDL. Maiores detalhes da implementação deste protótipo são apresentados no Capítulo 5.

4.9 Sumário do capítulo

Este capítulo apresentou o modelo conceitual proposto pela abordagem. A seção 4.2 apresenta algumas considerações sobre a proposta conceitual inicial do trabalho, enquadrando e justificando o caminho de pesquisa que foi escolhido. Na seção 4.3 é apresentada uma visão geral do cenário em que a abordagem trabalha, e nas seções

seguintes são apresentados detalhes das entidades envolvidas no cenário, além de detalhes sobre os módulos que compõem a abordagem proposta. Na seção 4.6 as entidades e os módulos genéricos do modelo são apresentados, aplicando os conceitos e nomenclaturas dos serviços Web, porém ocultando detalhes de implementação.

Então, a principal contribuição deste capítulo foi a apresentação da visão geral do cenário trabalhado e de detalhes de todas as entidades envolvidas, além de expor detalhes dos módulos da abordagem proposta e de como estes podem ser utilizados, possibilitando um melhor enquadramento do trabalho.

CAPÍTULO 5
Implementação e Avaliação

Esse capítulo apresenta detalhes de toda implementação envolvida no desenvolvimento da abordagem proposta.

Conforme dito anteriormente, a concepção desta abordagem envolveu alguns passos preliminares como a avaliação da interoperabilidade entre serviços Web implementados e disponibilizados em diferentes plataformas. Essas plataformas e seus serviços são descritas na seção 5.1. Após a modelagem e implementação de serviços em cada uma das plataformas, uma discussão sobre a interoperabilidade das plataformas é apresentada na seção 5.2, seguido da motivação e necessidade da concepção da abordagem proposta pelo trabalho.

Tendo como base as plataformas e serviços implementados e validados em relação a sua interoperabilidade nas seções 5.1 e 5.2, a abordagem de interoperabilidade proposta pelo trabalho e os detalhes da implementação de todos os módulos do protótipo são detalhados nas seções seguintes deste capítulo.

5.1 Plataformas utilizadas

Esta seção apresenta uma visão geral das plataformas de serviços Web utilizadas, bem como alguns detalhes da instalação e configuração destas. Serão apresentados e detalhados os serviços que foram implementados para os testes de interoperabilidade, a metodologia utilizada e as conclusões e resultados que foram atingidos.

A escolha das plataformas para estudo e desenvolvimento foi feita com base nos seguintes itens:

- Suporte a serviços Web;
- Utilização comercial, através do levantamento feito pela empresa especializada *Port 80* (PORT80SOFTWARE, 2006), onde foram avaliados quais servidores de aplicação para serviços Web são mais utilizados atualmente no mercado;

- Disponibilidade de *download* para período de avaliação;
- Utilização de diferentes implementações e motores SOAP para a disponibilização de serviços Web.

A forma como as plataformas foram distribuídas e disponibilizadas é apresentada na Figura 28. Neste trabalho, toda a comunicação entre as entidades é feita via SOAP sobre HTTP. Para cada uma das plataformas foram implementados 3 serviços diferentes, que são detalhados na seção 5.1.7. Para cada um destes serviços, foram também implementados os clientes que acessam/invocam operações destes serviços, porém de maneira estática e em tempo de projeto. As seções 5.3 e 5.4 detalham a forma como invocar estes serviços de maneira dinâmica, sem a necessidade da criação prévia dos *stubs*.

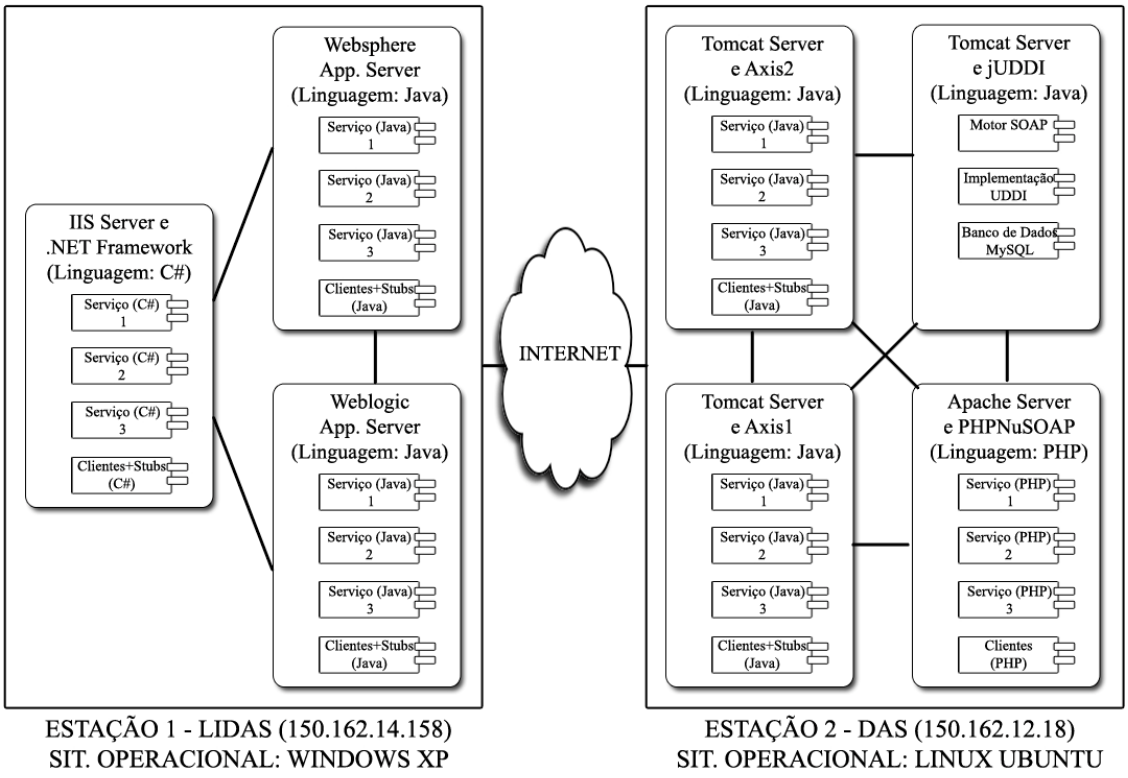


Figura 28: Cenário geral de disponibilização das plataformas avaliadas

As entidades e componentes descritos no Capítulo 4 estão todas instanciadas e apresentadas na figura acima, onde para cada plataforma existem diferentes serviços Web (Provedores de Serviços), os clientes para estes serviços (Consumidores de Serviços) e um registro UDDI onde de cada um dos serviços está publicado (Repositório de Serviços).

Ainda na Figura 28 é possível verificar que foram utilizadas duas estações, em redes separadas, e sistemas operacionais distintos, para a disponibilização das plataformas para a avaliação. No entanto, é importante considerar que, apesar das diferentes plataformas estarem disponibilizadas em apenas duas estações e sistemas operacionais distintos, isso não impede ou limita a proposta da abordagem e também sua avaliação. Isto significa que esta pode ser expandida para quantas estações forem necessárias, e ainda assim contemplar a proposta de interoperabilidade entre diferentes plataformas de serviços Web distribuídos.

5.1.1 Apache Axis

O Axis é um motor SOAP de código aberto provido pela Fundação Apache. Pode ser definido como um *framework* para construção de processadores SOAP, sejam eles clientes ou servidores (APACHEWEBSERVICES, 2007).

O Axis é um *servlet*[24] específico para SOAP, que pode ser executado dentro de qualquer container que suporte *servlets*. Um container que é amplamente utilizado, por ser gratuito e de código aberto, é o *Apache Tomcat* (APACHEFOUNDATION, 2007). Para esse trabalho, o *Apache Tomcat* versão 5.5.12 para sistema operacional Linux foi escolhido para a publicação do *servlet* do Axis.

O Axis possui versões desenvolvidas em duas linguagens, uma escrita em Java, e a outra escrita em C++, esta última ainda em versão beta. A versão Java foi a escolhida para o desenvolvimento dos serviços Web, por ser considerada mais estável. O Axis Java possui também duas versões, o Axis1 e o Axis2:

- **Axis1:** O Axis1 é a uma evolução do Apache SOAP, provendo algumas atualizações, como por exemplo o suporte a serviços *document/literal* para atender as recomendações do *Basic Profile* da WS-I.
- **Axis2:** O Axis2 é uma re-implementação completa do Axis1, provendo uma nova arquitetura, mais modular e eficiente, de suporte a serviços Web. Alguns conceitos do Axis1 foram mantidos, como os *handlers*, porém novas funcionalidades foram incluídas, como, por exemplo, o suporte ao estilo

[24] Um programa que roda em um servidor, normalmente um servidor Web.

arquitetural REST, e também às novas especificações dos serviços Web, como o *WS-Security* e *WS-Addressing*.

5.1.2 IBM Websphere Application Server

O IBM Websphere *Application Server* (WAS) é um servidor de aplicação da IBM, que implementa as funcionalidades do Java *Enterprise Edition*. O WAS é voltado para negócios eletrônicos dinâmicos, provendo integração de dados e de transações, de maneira segura e escalável (SADTLER *et al.*, 2004).

Neste trabalho, a versão do WAS instalada e utilizada para implementação e testes dos serviços Web foi a versão 5.1, que é disponibilizada gratuitamente no site da IBM para avaliação. O WAS possui suporte a serviços Web, onde o motor SOAP pode ser definido na sua configuração. O motor SOAP utilizado foi o Apache SOAP, porém é possível substituí-lo por outro motor SOAP, como por exemplo, o Axis.

Porém, o motivo da utilização do Apache SOAP é a de se criar um cenário extremamente heterogêneo, onde diferentes implementações de serviços Web – e seus motores SOAP – interoperem de maneira transparente e sem costuras.

A IBM também possui um produto chamado Websphere *Application Developer*, que é um ambiente integrado que auxilia os desenvolvedores no projeto, desenvolvimento, teste e disponibilização de aplicações voltadas tanto para o lado do servidor, como serviços Web, quanto para o lado do cliente, como a disponibilização de páginas Web.

5.1.3 BEA Weblogic Server

O BEA Weblogic *Server* é o servidor de aplicação baseado no Java *Enterprise Edition*, para a disponibilização de aplicações baseadas em Java, como os serviços Web, EJBs, etc, provendo o suporte adicional para segurança, disponibilidade e escalabilidade (BEA, 2006).

Neste trabalho, a versão 8.1 foi instalada e utilizada para a implementação dos serviços Web, e testes de interoperabilidade nos serviços Web. O BEA Weblogic *Server* utiliza uma implementação proprietária do motor SOAP, chamada e BEA SOAP, porém permite a utilização de outro motor SOAP, como o Axis.

A BEA também provê um ambiente para desenvolvimento, chamado de BEA *Workshop*, que auxilia o desenvolvedor na implementação de diferentes aplicações através de interfaces de suporte – chamados de *wizards*.

5.1.4 Microsoft .NET Framework

O .NET *Framework* é uma coleção de tecnologias, ferramentas e linguagens que trabalham em conjunto para prover soluções para o desenvolvimento e disponibilização de aplicações empresariais robustas (EVJEN, 2002). Os serviços Web, bem como o motor SOAP, são providos por uma camada do .NET *Framework* chamada de ASP.NET, que são disponibilizadas dentro do servidor Web da Microsoft, o *Internet Information Services* (IIS).

Neste trabalho a versão 2.0 do .NET Framework foi instalada, e disponibilizada dentro do Microsoft IIS versão 5.0. Os serviços Web no ASP.NET foram implementados na linguagem C#, utilizando como ambiente de desenvolvimento o *Microsoft Visual Studio Professional* 2005, que é disponibilizado gratuitamente pela Microsoft para avaliação.

5.1.5 PHP NuSOAP

O PHP é uma linguagem de *script* de código aberto voltada para o desenvolvimento de aplicações Web dinâmicas. É executado dentro de um servidor Web, que esteja previamente configurado para interpretar comandos escritos na linguagem PHP.

O PHP NuSOAP é um conjunto de ferramentas voltadas para serviços Web, composto de uma coleção de classes totalmente escritas em PHP. Essas classes permitem o envio e recebimento mensagens SOAP sobre HTTP, geração de WSDL automaticamente, entre outras funcionalidades (SARANG *et al.*, 2002).

Neste trabalho, o NuSOAP 0.7.2 foi utilizado para a implementação dos serviços Web e testes de interoperabilidade. O servidor Web utilizado para a disponibilização do NuSOAP e dos serviços Web foi o Apache HTTP *Server* (APACHEFOUNDATION, 2007).

5.1.6 jUDDI

O jUDDI (APACHEWEBSERVICES, 2007) é uma implementação de código aberto da especificação 2.0 do UDDI, e foi desenvolvido na linguagem Java pela Fundação Apache. O jUDDI pode ser instalado em qualquer servidor de aplicação que suporte a versão 2.1 da API de Servlets – Websphere, Weblogic, Tomcat – e pode ser utilizado com qualquer banco de dados relacional que suporte o padrão SQL – MySQL, DB2, Sybase, etc.

Para a publicação e localização dos serviços Web que foram desenvolvidos nesse trabalho, foi utilizado o *jUDDI* versão 0.9rc4, que foi disponibilizado dentro do container *Apache Tomcat* versão 5.5.12, utilizando o motor SOAP Axis, e banco de dados *MySQL* versão 4.0.24. A Figura 29 ilustra esse cenário:

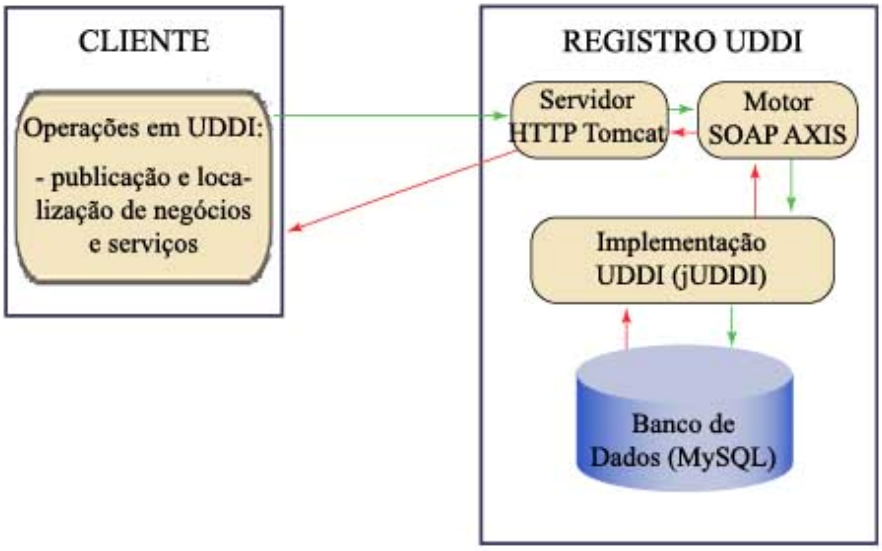


Figura 29: Implementação completa de um Registro UDDI

5.1.7 Serviços nas plataformas

A idéia principal na criação dos serviços é o de disponibilizar diferentes serviços com diferentes tipos de dados de entrada, de saída, quantidade de parâmetros, e tipos dos parâmetros (simples ou complexos). Os serviços foram implementados em diferentes linguagens, de acordo com a plataforma na qual este é disponibilizado.

O primeiro serviço que foi desenvolvido e disponibilizado é uma simples consulta de preços de produtos, onde o usuário informa o nome do produto e o serviço retorna o

valor do produto. Esse serviço foi alterado, incluindo mais um parâmetro, que é a operação que o usuário deseja efetuar no item – consulta ou exclusão. A assinatura do método é apresentada na Figura 30.

```
public String getPrice (String operacao, String item)
```

Figura 30: Assinatura do método do serviço 1

O segundo serviço desenvolvido é uma calculadora, que executa operações aritméticas básicas – soma, subtração, multiplicação e divisão – onde o usuário pode especificar até 3 parâmetros diferentes para a operação escolhida. A assinatura do método é apresentada na Figura 31.

```
public double Calcula (String operacao, double valor1, double valor2, double valor3)
```

Figura 31: Assinatura do método do serviço 2

O último serviço que foi desenvolvido é uma consulta a funcionários, baseando-se em seu número de CPF, que retorna um tipo complexo *Funcionario*, que contém o nome, função e departamento. A assinatura do método é apresentada na Figura 32.

```
public Funcionario detalhaFunc (String CPF)
```

Figura 32: Assinatura do método do serviço 3

Todos os serviços foram disponibilizados e executados com sucesso em suas plataformas. Detalhes sobre a avaliação de interoperabilidade entre cada um desses serviços nas diferentes plataformas são apresentados na seção seguinte.

Considerações sobre serviços nas plataformas

Através da modelagem e implementação prática de diferentes serviços nas diferentes plataformas, e a invocação destes serviços entre si, contribuiu para a avaliação de questões ligadas com a interoperabilidade entre diferentes plataformas e linguagens para o desenvolvimento de serviços Web.

Dentre os resultados e conclusões obtidas através da análise da literatura sobre o assunto, e também por testes feitos entre serviços, é possível enumerar alguns problemas

relacionados com os tipos de dados que são restritos a uma linguagem, e que ao serem mapeados para tipos de dados em XML, podem se tornar interoperáveis com outras linguagens (SWITHINBANK *et al.*, 2005):

- No Java: *Hashtable, Vectors, Hashmap, Set, ArrayList*
- No .NET (C#): *Hashtable*, *SortedList*, *Queue*, *Stack*, *ArrayList*

Apesar de alguns tipos acima serem similares como, por exemplo, o *ArrayList*, isso não os torna interoperáveis, pois os elementos contidos são referências genéricas, fazendo com que a deserialização de uma representação XML de um *Collection* se torne suscetível ao conhecimento prévio, por parte do consumidor do serviço, do tipo de dado contido dentro de um tipo *Collection.*

Portanto, para se manter os serviços Web o mais interoperáveis possível, é preciso adotar, sempre que possível, os tipos simples na exposição de métodos como serviços.

5.2 Interoperabilidade dos serviços Web nas plataformas

5.2.1 Introdução

Devido ao fato do protótipo ser voltado para a arquitetura de serviços Web, a parte de desenvolvimento e implementação dos serviços em cada uma das plataformas descritas acima se torna uma parte importante de todo o trabalho. A avaliação do protótipo, na seção 5.5, é baseada na invocação de todos os serviços que foram desenvolvidos em cada uma destas plataformas, incluindo a avaliação da interoperabilidade em cada um destes serviços.

A metodologia adotada foi a de desenvolver o mesmo serviço em diferentes plataformas, e consequentemente em diferentes linguagens, onde para cada serviço é criado um cliente para invocação – incluindo *stubs* e *proxies* – em cada uma das outras plataformas, que deverá ser capaz de invocar de maneira interoperável este serviço. O cenário é apresentado na Figura 33:

CLIENTES			
Linguagem	Ambiente	Motor SOAP	Sist. Operac.
Java	Eclipse	Axis1	Ubuntu Linux
Java	Eclipse	Axis2	Ubuntu Linux
Java	Websphere	Apache SOAP	Windows XP
Java	Weblogic	BEA SOAP	Windows XP
C#	MS .NET	ASP.NET	Windows XP
PHP	Apache Webserver	PHPNuSOAP	Ubuntu Linux

SERVIÇOS			
Linguagem	Ambiente	Motor SOAP	Sist. Operac.
Java	Apache Tomcat	Axis1	Ubuntu Linux
Java	Apache Tomcat	Axis2	Ubuntu Linux
Java	Websphere	Apache SOAP	Windows XP
Java	Weblogic	BEA SOAP	Windows XP
C#	MS .NET + IIS Server	ASP.NET	Windows XP
PHP	Apache Webserver	PHPNuSOAP	Ubuntu Linux

Figura 33: Cenário de avaliação de interoperabilidade das plataformas

Todas as interações são descritas no diagrama UML de seqüência da Figura 34. Neste caso, o cliente interage com o serviço via os *proxies* e *stubs* que foram gerados em tempo de projeto da aplicação.

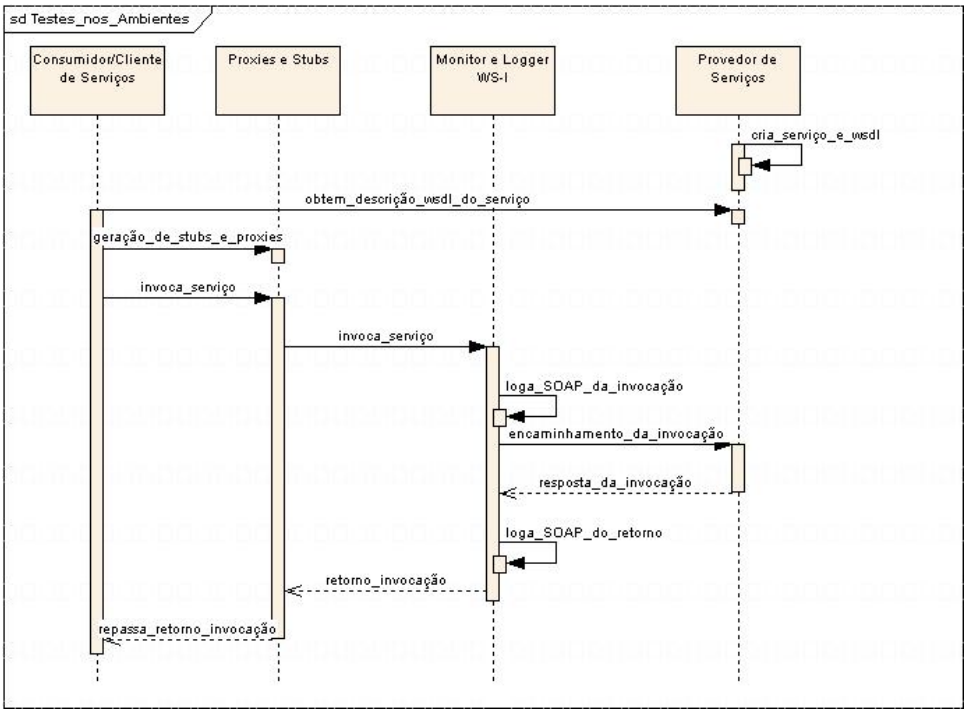


Figura 34: Diagrama de seqüência do cenário de avaliação

Uma das motivações dessa etapa é a de avaliar, utilizando as ferramentas providas pela WS-I descritas na seção 3.4.2, a real interoperabilidade entre diferentes plataformas para serviços Web, devido ao fato de que muitos dos problemas de interoperabilidade entre serviços são causados por divergências na interpretação das especificações do SOAP entre os diferentes fabricantes.

Esse problema não se torna tão crítico quando se está operando dentro das fronteiras da organização, pois as plataformas de serviços Web tendem a ser do mesmo fabricante. Porém, devido à natureza heterogênea do cenário de RCOs em que esse protótipo é aplicado, diferentes plataformas de suporte a serviços Web, providos por fabricantes e localizados fisicamente em diferentes pontos, a questão da interoperabilidade se torna cada vez mais importante.

5.2.2 Metodologia de desenvolvimento nas plataformas

Para cada plataforma, a forma de se desenvolver os serviços é diferente, cada um seguindo uma das metodologias de desenvolvimento previamente descritas na seção 2.2.7.

Para cada uma das plataformas, foi escolhida uma metodologia de desenvolvimento, com o intuito de simular um ambiente realmente heterogêneo, onde diferentes desenvolvedores possuem sua lógica específica de desenvolvimento de serviços Web. A maioria dos serviços foi desenvolvida utilizando a abordagem de *green-field* e *bottom-up* para a criação inicial das aplicações, seguida da sua exposição como serviços Web, com exceção das plataformas IBM Websphere e BEA Weblogic, onde a abordagem escolhida foi a *top-down*.

Para cada serviço criado e disponibilizado em uma plataforma, foram criados também os clientes para consumir esse serviço. Os clientes foram criados em todas as plataformas, inclusive no qual o serviço foi desenvolvido. Com isso é possível tirar conclusões a respeito da interoperabilidade de cada uma das plataformas, tanto no lado do provedor quanto do lado do consumidor dos serviços.

A criação dos clientes nas plataformas e a invocação dos serviços são feitas de maneira estática, onde a localização da URI do documento WSDL é feita manualmente, em alguns casos conhecida a priori, e a geração de todos os *stubs* para cada um dos clientes

também é necessária. Apenas após esses passos, a invocação do serviço Web pode ser efetuada.

5.2.3 Considerações sobre o desenvolvimento e interoperabilidade

Os resultados dos testes mostraram que algumas questões de interoperabilidade em serviços Web apresentadas na literatura (SIDDHARTHA *et al.*, 2003), foram resolvidas com a criação do *Basic Profile* (BALLINGER *et al.*, 2004) da WS-I. O desenvolvimento dos serviços Web foi todo feito em conformidade com os requisitos do *Basic Profile*, e também seguindo algumas recomendações, como as apresentadas em SWITHINBANK *et al.*, (2005) e YE, (2004), dessa forma tornando os serviços mais interoperáveis.

Os serviços Web e seus principais padrões foram concebidos em meados de 2001 (OASIS-UDDI, 2001) e passaram por diversas evoluções nestes padrões, porém somente após o desenvolvimento do *Basic Profile*, em 2004, quando os diferentes fabricantes de soluções passaram a trabalhar em conjunto na definição de diretivas para as especificações e desenvolvimento de serviços Web, é que a interoperabilidade entre diferentes plataformas de serviços Web passou a ser algo mais consolidado (SWITHINBANK *et al.*, 2005).

Apesar dos serviços desenvolvidos nas plataformas serem interoperáveis entre si, na forma mais comum de invocação de serviços Web, na qual a localização do serviço deve ser conhecida *a priori* e os *stubs* e *proxies* devem ser criados previamente à invocação, impede uma efetiva descoberta e invocação dinâmica e automática de serviços em tempo de execução. A geração e manutenção de *stubs* em um cenário heterogêneo, como uma RCO, contendo diversas organizações que utilizam diferentes plataformas para a disponibilização de seus serviços Web, tende a se tornar muito complexa e trabalhosa.

Portanto, é necessário que se crie formas de suportar a localização e invocação de serviços sob-demanda e de maneira *stubless*, i.e., sem a necessidade prévia de conhecimento da localização da interface do serviço e da geração de *stubs* e *proxies*, provendo uma maior dinamicidade e automação, e conseqüentemente facilitando a colaboração entre diferentes parceiros de negócios. A seção seguinte apresenta detalhes da abordagem proposta por este trabalho, onde a localização e invocação de serviços são feitas de maneira dinâmica e, principalmente, **interoperável**, independentemente da plataforma na qual o serviço está disponibilizado.

5.3 Desenvolvimento do Invocador *Stubless*

5.3.1 Introdução

Existem diferentes formas de invocação de serviços Web utilizando o Java. O JAX-RPC, provido pela Sun, pode ser utilizado por um cliente para a invocação de serviços (SUN, 2006). Existem três formas de invocação:

- **Cliente com *stub* estático**: O cliente faz todas as invocações ao serviço via um *stub* previamente gerado, baseando-se na WSDL do serviço. Esse *stub* é gerado em tempo de projeto da aplicação, funcionando como um *proxy*, sendo usado para fazer a comunicação efetiva com o serviço remoto. Qualquer mudança no provedor do serviço (parâmetros, tipos, etc) cria a necessidade de uma nova geração dos *stubs* para o serviço.
- **Cliente com *proxy* dinâmico**: O cliente não precisa ter os *stubs* para o cliente gerados previamente. O *proxy* é gerado em tempo de execução, através da análise do diretório do provedor de serviços, para localizar qual é a classe correspondente. Porém, o cliente precisa saber qual o nome da classe no lado do provedor, fazendo com que haja a necessidade de um acordo prévio entre o consumidor e o provedor de serviços, ficando restrito à linguagem Java, limitando a flexibilidade do cliente.
- **Cliente com DII (*Dynamic Invocation Interface*):** Com a DII, um cliente não precisa saber as classes do lado do provedor de serviço, como no *proxy* dinâmico. Os procedimentos de invocação são criados em tempo de execução, através da análise do documento WSDL do serviço, independentemente da linguagem na qual o serviço foi implementado. Porém, a DII invoca apenas serviços baseados no estilo *rpc/encoded*, o que também limita a flexibilidade do cliente, pois os serviços atuais tendem a ser desenvolvidos baseando-se no estilo *document/literal*, que é um estilo recomendado pela WS-I, por ser mais interoperável.
- **Cliente utilizando SAAJ:** Através da biblioteca SAAJ é possível que o cliente construa uma mensagem SOAP manualmente, baseando-se na leitura de uma descrição WSDL. Porém, esse método se torna muito complexo e exige um

conhecimento avançado das estruturas de mensagens SOAP e do documento WSDL por parte do cliente, dificultando o desenvolvimento e a manutenção do código.

Portanto, após análises e testes com os clientes acima, constatou-se que para o cenário de RCOs, nenhuma destas formas de invocação se torna viável, devido às características já apresentadas. Logo, é preciso que seja provida uma forma de se invocar serviços Web de maneira simples, independente do tipo de serviço e plataforma em que este foi desenvolvido.

5.3.2 Abordagem de desenvolvimento

Baseando-se nos requisitos de suporte ao cenário almejado, onde diferentes sistemas interoperam efetivamente de maneira independente e aberta, este trabalho faz uso de APIs que possibilitam atingir uma invocação transparente, dinâmica e *stubless* de serviços Web, independentemente de detalhes técnicos, tecnologias envolvidas (SOAP, EJB, JMS, etc) e plataforma na qual o serviço foi disponibilizado.

Após pesquisas e avaliações, a API do Apache WSIF (*Web Services Invocation Framework*) (APACHEWEBSERVICES, 2007) foi escolhida como suporte ao desenvolvimento do invocador dinâmico e *stubless*, que irá prover essa transparência à aplicação cliente na invocação de serviços descritos em WSDL, independentemente dos mecanismos de acesso envolvidos. Na seção a seguir, são apresentados maiores detalhes a respeito do WSIF.

5.3.3 WSIF

O WSIF é uma API escrita em Java, para a invocação de serviços Web, independente de como ou onde o serviço é disponibilizado, possibilitando que a aplicação cliente interaja apenas com representações abstratas dos serviços através de sua interface WSDL. Isso significa que o WSIF atua como um tipo de mediador entre provedores e consumidores de serviços, evitando que clientes necessitem saber detalhes da comunicação e protocolos utilizados, se atendo apenas à descrição do serviço, i.e., sua interface. Em resumo, permite a invocação de serviços de maneira transparente e *stubless*.

A principal motivação para a utilização do WSIF é que, através deste, é possível visualizar a Arquitetura Orientada a Serviços como algo mais extensível que apenas serviços e ligações SOAP. Existe atualmente uma grande quantidade de tecnologias para sistemas distribuídos e de diferentes protocolos que são encontrados nas mais diversas organizações, o que torna essa flexibilidade de uso de diferentes protocolos uma característica obrigatória, quando a questão é a interoperabilidade sem costuras de sistemas heterogêneos.

Então, é possível enumerar algumas das principais vantagens oferecidas pelo invocador *stubless*, baseado nas APIs do WSIF:

- Invocação de serviços Web sem a necessidade prévia de criação de *stubs*.
- Possibilidade de se ter um mecanismo independente de ligações (*bindings*) na invocação de serviços Web.
- Permite a escolha de diferentes ligações para um serviço Web.
- Possibilita uma maior interoperabilidade, independentemente do tipo de serviço oferecido por uma organização.

Arquitetura

A API do WSIF permite que clientes invoquem serviços focando na descrição abstrata do serviço, i.e., na parte do WSDL que cobre os *port types*, operações e troca de mensagens, sem se referir aos protocolos reais. Essa invocação abstrata é possível graças a módulos que são providos pelo WSIF, que são específicos para cada protocolo, chamados de **Provedores (***Providers*). Um Provedor é o responsável por lidar com as mensagens que serão trocadas, de acordo com as especificações de cada protocolo em particular. Por exemplo, o Provedor SOAP que acompanha o WSIF utiliza um motor SOAP específico para executar efetivamente as funções de invocação.

De acordo com APACHEWEBSERVICES (2007) é possível ainda estender o WSIF com Provedores adicionais, que podem lidar com as funções e invocações de

serviços implementados em outras tecnologias e protocolos além dos já suportados pela distribuição da Apache[25].

Funcionamento do WSIF

A idéia por trás do uso do WSIF é a sua simplicidade. Além da possibilidade de se utilizar múltiplas ligações (*bindings*), o fato de depender apenas da descrição WSDL do serviço para a criação dos procedimentos de invocação se torna muito atrativo em um cenário de RCOs. Isso se deve ao fato de não existir a necessidade de se criar *stubs* que são tradicionalmente utilizados para invocar um serviço, pois a API lê todo o documento WSDL de um serviço, extraindo informações relevantes, para a então criação automática dos procedimentos e por fim invocar a operação solicitada de maneira *stubless*. Com isso, é possível que um mesmo código de aplicação cliente seja utilizado para invocar qualquer ligação disponível. A Figura 35 ilustra a utilização do WSIF no cenário de serviços Web:

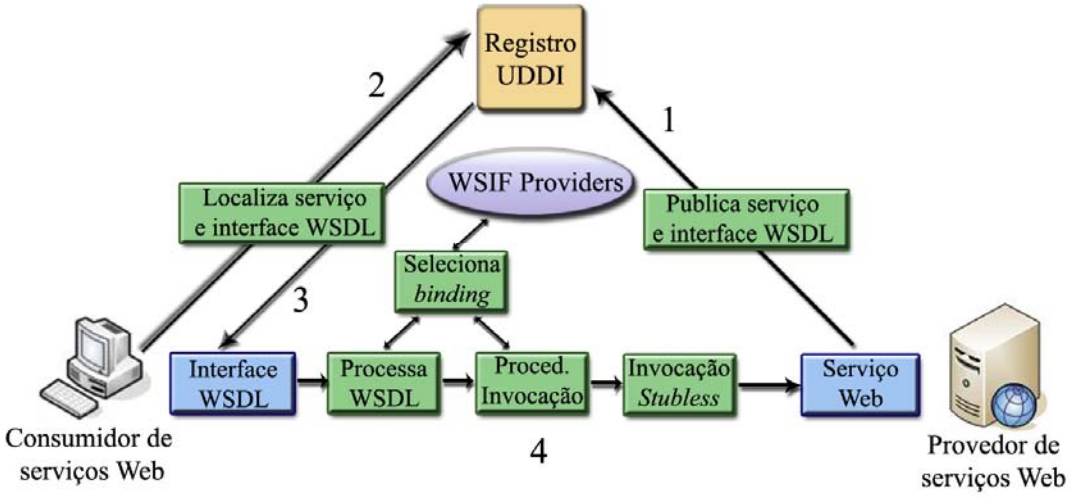


Figura 35: Diagrama de utilização do WSIF

Nesta figura, um provedor de serviços torna seus serviços disponíveis através de sua publicação em um repositório, por exemplo, um registro UDDI (1). Quando uma aplicação cliente precisa consumir um serviço, é necessário que se localize previamente o serviço (2), baseando-se em critérios de escolha, para então obter sua descrição WSDL do serviço (3), que foi previamente publicada no repositório pelo provedor.

[25] Os Provedores atualmente incluídos por padrão na distribuição do WSIF da Apache são: SOAP, EJB, Java, JMS e JCA.

Após isso, o documento WSDL é lido, onde as informações necessárias são extraídas, os procedimentos de invocação – criação do serviço, da operação e das mensagens de entrada e saída – são criados dinamicamente e o serviço é invocado (4), sem a necessidade de qualquer intervenção manual, i.e., nenhuma geração de *stubs* é necessária. Essa invocação é baseada em um tipo particular de protocolo de ligação (como o SOAP, EJB, JMS, JCA, etc.) que é descrita no documento WSDL, e é tratada pelo Provedor do WSIF, que será descrito na seção seguinte.

Classes do WSIF

O WSIF é composto por diferentes classes, que são utilizadas pelo invocador *stubless* para a invocação dinâmica de serviços Web. Estas classes, bem como os métodos que foram utilizados pelo invocador *stubless* são ilustradas na Figura 36 e descritas abaixo:

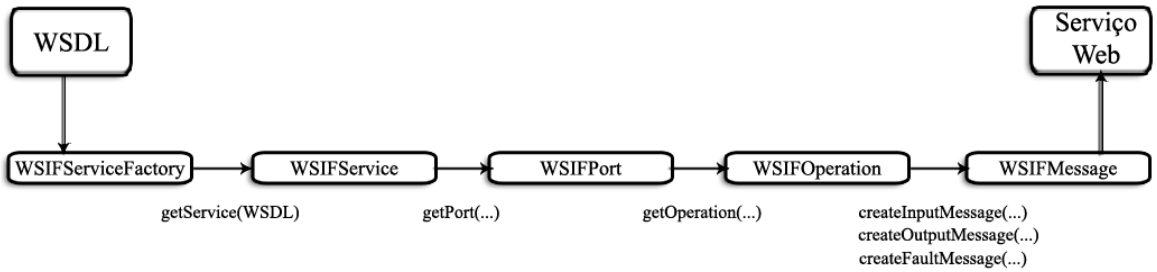


Figura 36: Classes do WSIF e métodos utilizados

- *WSIFServiceFactory*: É uma classe fábrica abstrata que é utilizada para a criação de instâncias de serviços.
- *WSIFService*: É uma classe que representa um serviço em tempo de execução, utilizado para adquirir o *Port* WSDL.
- *WSIFPort*: É uma classe que representa um *Port* WSDL em tempo de execução. A função deste *Port* é o de executar a invocação propriamente dita, baseando-se na ligação descrita no WSDL.
- *WSIFOperation:* É uma classe que representa, em tempo de execução, uma operação WSDL que foi previamente definida pelo cliente. Baseando-se em uma ligação específica, irá invocar o serviço correspondente. É usado para a criação das mensagens de entrada, saída e de possíveis faltas.

- *WSIFMessage:* É a interface que representa uma mensagem WSDL. É a representação em tempo de execução das mensagens de entrada, saída e possíveis faltas de uma operação.

Todas essas classes descritas acima fazem parte da biblioteca do Apache WSIF. Porém, após análises de interoperabilidade através de invocações em serviços Web SOAP, disponibilizados em diferentes plataformas, foi possível verificar a existência de alguns problemas de interoperabilidade na invocação de serviços na plataforma Microsoft .NET e também PHP NUSOAP.

Porém, após avaliação e pesquisas em alguns fóruns sobre o assunto, foi proposto o uso de uma outra biblioteca, que é desenvolvida e mantida por um dos colaboradores do projeto Apache WSIF, chamada XSUL *Library* (INDIANA-XSUL, 2007). Esta biblioteca implementa as funcionalidades do WSIF, porém, por ser constantemente atualizada, possui melhor suporte para invocação de serviços SOAP baseados no estilo *document/literal*, estilo este que tem se tornado a tendência atualmente, além de ser recomendado pela WS-I (BUTEK, 2005).

Cenário geral de uso do WSIF

O WSIF foi utilizado pelo invocador *stubless* para suportar o cenário de RCOs já mencionado. A Figura 37 apresenta uma visão geral desse cenário, ilustrando a existência de diferentes aplicações e serviços heterogêneos, que são providos por diferentes organizações, utilizando diferentes tecnologias de software de diferentes fabricantes.

Então, no invocador *stubless*, além da aplicação cliente não precisar se preocupar com questões relacionadas com os procedimentos de invocação e geração dos *stubs*, com o uso do WSIF é possível que se invoque diferentes serviços Web, desenvolvidos em diferentes tecnologias e protocolos além do SOAP. O único elemento que é necessário se conhecer é a descrição WSDL do serviço – que é previamente localizada no registro UDDI – e a operação a ser invocada.

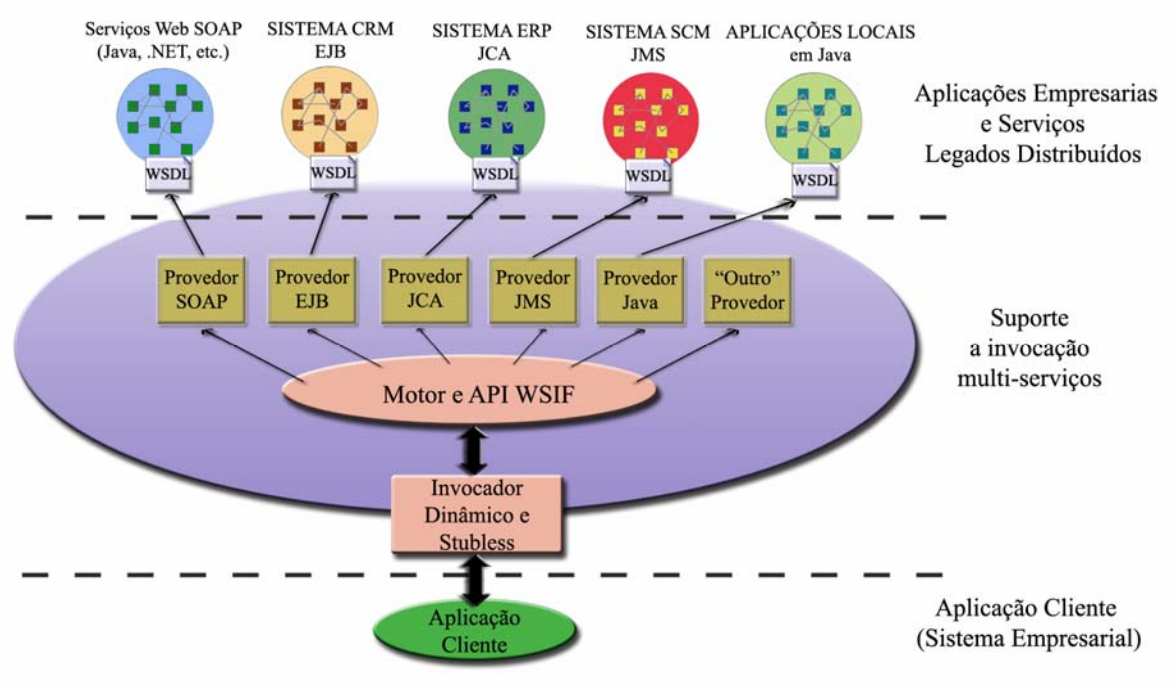


Figura 37: Visão geral da utilização do WSIF e do suporte a invocação multi-serviços

5.3.4 Detalhes da implementação do módulo de invocação

A linguagem padrão utilizada para o desenvolvimento do protótipo – tanto as ferramentas de invocação quanto as ferramentas de UDDI descritas na seção a seguir – é a linguagem Java.

As classes principais que compõem o invocador *stubless* são apresentadas na Figura 38. Essas classes são apresentadas de maneira simplificada, apresentando apenas as classes que estão envolvidas diretamente com a abordagem proposta, ocultando detalhes de classes pertencentes a outras bibliotecas.

A classe principal do invocador *stubless* é chamada de *StublessDynamicInvocation* que é instanciada sempre que existe a necessidade de uma invocação de serviços. Essa classe faz uso das classes da API do WSIF, como forma de suportar diferentes protocolos para a invocação de serviços Web. Neste protótipo, foram tratados apenas serviços SOAP.

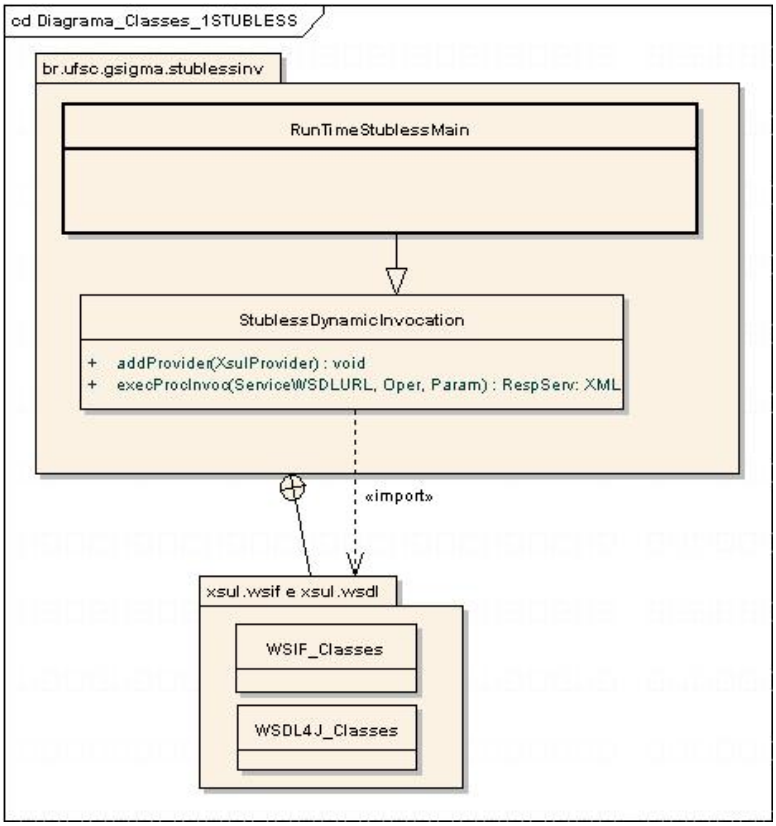


Figura 38: Diagrama de classes do invocador dinâmico *stubless*

As classes do WSIF e do WSDL4J que são utilizadas também pelo invocador foram omitidas nesse diagrama, porém podem ser consultadas em APACHEWEBSERVICES (2007) e (WSDL4J, 2007). Algumas das principais classes e operações do WSIF, e que foram utilizadas na implementação, também são detalhadas na seção 5.3.3 deste trabalho.

Algoritmo de invocação

Os procedimentos criados para a invocação *stubless* são baseados apenas na descrição WSDL da interface do serviço, que pode ser adquirida em tempo de execução. Após obter essa descrição WSDL, o documento é analisado, e, baseando-se no tipo de ligação do serviço, os procedimentos de invocação são criados, o serviço é consumido e a resposta é trazida. O algoritmo que descreve essas operações é apresentado na Figura 39 em pseudo-código.

```
Entrada: URL da interface WSDL do serviço; Operação a ser invocada; Parâmetros
Inicio
   Download da interface WSDL
   Análise da interface WSDL e Extração de dados
   Localização do PortType correspondente
   Extrair Operação
   para cada Operação do PortType faça
      Extrair Mensagem
      para cada Mensagem da Operação faça
      enquanto (não é fim da Mensagem) faça
            se Mensagem é de entrada então
                  extrair dados sobre parâmetros de entrada
            fim se
            se Mensagem é de saída então
                  extrair dados sobre parâmetros de saída
            fim se
      fim enquanto
   fim para
   fim para
   Criação da fábrica do serviço, operações e mensagens a serem enviadas
   Envio da invocação, tratamento da resposta recebida
Fim.
```

Figura 39: Algoritmo de invocação dinâmica de serviços

A Figura 40 apresenta um exemplo de um documento WSDL descrevendo um serviço que possui mais de uma ligação disponível, chamado de serviço com *múltiplas ligações*. Neste exemplo, o documento WSDL do serviço apresenta uma ligação para SOAP e outra para o REST[26] (FIELDING, 2000). A escolha do tipo de ligação para a invocação do serviço deve ser programada pelo desenvolvedor, de acordo com as necessidades e critérios definidos previamente.

```
<wsdl:service name="GetPrice">
    <wsdl:port name="GetPriceSOAP11port_http" binding="axis2:GetPriceSOAP11Binding">
        <soap:address location="http://127.0.0.1:8080/webservices/axis2/services/GetPrice"/>
    </wsdl:port>
    <wsdl:port name="GetPriceHttpport1" binding="axis2:GetPriceHttpBinding">
        <http:address location=" http://127.0.0.1:8080/webservices/axis2/rest/GetPrice"/>
    </wsdl:port>
</wsdl:service>
```

Figura 40: Exemplo de múltiplas ligações de um serviço.

5.3.5 Limitações do invocador

A principal vantagem do invocador *stubless* está na invocação dinâmica e transparente de serviços Web, sem a necessidade de geração prévia de *stubs*, independente

[26] REST (*REpresentational State Transfer*) é uma opção ao SOAP no desenvolvimento de serviços Web.

dos protocolos e tecnologias envolvidas. Porém, isso gera um problema que é o *overhead* na invocação dos serviços, pois a cada necessidade de invocação, o documento WSDL é varrido, e os procedimentos de invocação são criados novamente, independentemente do serviço já ter sido ou não invocado anteriormente.

Esse *overhead* está diretamente ligado com a capacidade de processamento e velocidade de conexão do cliente. No entanto, essa limitação pode ser considerada secundária em um cenário de redes colaborativas, onde não existe a restrição temporal da invocação ser feita em tempo-real.

5.4 Desenvolvimento dos utilitários UDDI

A UDDI (*Universal Description, Discovery and Integration*) é uma especificação da OASIS que define uma forma padrão de publicar e localizar negócios e interfaces de seus serviços Web em um registro, ou diretório (BRYAN *et al.*, 2002). Conforme descrito na seção 2.2.5, os registros UDDI têm sido aplicados efetivamente dentro de um cenário mais restrito e privado, mais voltado para a integração entre organizações, possibilitando que os provedores e consumidores de serviços tenham um maior controle nas operações no registro, i.e., na publicação, localização alteração e exclusão de serviços.

Atualmente, o principal modelo de utilização do UDDI é na descoberta de serviços em tempo de desenvolvimento e não dinamicamente em tempo de execução (SÁNCHEZ-NIELSEN *et al.*, 2006). Isto significa que o usuário localiza os serviços de interesse em um registro UDDI, lê a descrição dos serviços, e então desenvolve um cliente que irá interagir com esse serviço descoberto. Isso inviabiliza o uso do UDDI em um cenário altamente dinâmico – como uma RCO – onde a necessidade de localização e invocação de serviços se dá em tempo de execução. Portanto, é necessário o provimento de mecanismos que possibilitem que essas operações em um registro UDDI sejam executadas de maneira transparente e dinâmica, de forma que clientes possam localizar serviços mais agilmente e em tempo de execução.

Esse é o propósito desse pacote, onde são providas ferramentas especificas para cada operação em um registro UDDI, tanto do lado do provedor de serviços – publicação, alteração e exclusão de negócios, serviços, etc – quanto do lado do consumidor de serviços – localização de negócios, serviços, etc. Essas ferramentas podem ser utilizadas em tempo

de execução, onde um serviço é localizado dinamicamente em um registro UDDI, e em seguida invocado, de maneira dinâmica e *stubless*.

5.4.1 UDDI4J

A especificação da UDDI define uma API baseada em XML para a interação entre cliente – provedor ou consumidor – e o registro UDDI. Conforme descrito anteriormente, a linguagem padrão utilizada para o desenvolvimento dos módulos UDDI foi a linguagem Java. O projeto de código aberto UDDI4J (UDDI4J, 2007) define uma API Java que provê o acesso a qualquer registro UDDI a partir de uma aplicação Java, utilizando o protocolo SOAP para interagir com o registro.

Esta biblioteca será utilizada para o desenvolvimento das ferramentas de interação com o registro UDDI. A escolha dessa biblioteca se deve ao fato de ser de código aberto e gratuita, além de ser compatível com a versão 2.0 da especificação UDDI, que é a versão utilizada pelo registro jUDDI utilizado para os testes.

Interações com um Registro UDDI

A forma de interação com um registro UDDI pode ser feita utilizando duas APIs: a de busca – *inquiry* – e a de publicação – *publish*.

As mensagens de busca normalmente são liberadas para uso geral, onde os clientes fazem a busca sem necessidade de autenticação. Os tipos de consulta a UDDI disponíveis são (BELLWOOD *et al.*, 2002):

Método	Descrição
find_binding	Utilizado para localizar *bindings* (ligações) entre um ou mais *businessServices* registrados.
find_business	Utilizado para localizar informações sobre um ou mais negócios.
find_relatedBusinesses	Utilizado para localizar informações sobre algum *businessEntity* que estão relacionados com uma entidade de negócios específica, a qual tem sua chave passada como parâmetro no momento da busca.
find_service	Utilizado para localizar serviços específicos dentre as entidades de negócio registradas.
find_tModel	Utilizado para localizar uma ou mais estruturas *tModels*
get_bindingDetail	Utilizado para acessar a informação de *bindingTemplate* necessária para

	fazer as requisições de serviços.
get_businessDetail	Utilizado para acessar a informação de *businessEntity* para um ou mais negócios/organizações.
get_businessDetailExt	Utilizado para acessar informações estendidas do *businessEntity*.
get_serviceDetail	Utilizado para acessar detalhes completos de um determinado *businessService*.
get_tModelDetail	Utilizado para acessar detalhes completos de um determinado *tModel*.

Tabela 3: Comandos da API de busca (*inquiry*) em um registro UDDI

As mensagens de publicação são feitas por usuários autenticados e liberados pelo administrador do registro UDDI. Abaixo são apresentadas as mensagens de publicação em um registro UDDI (BELLWOOD *et al.*, 2002):

Método	**Descrição**
add_publisherAssertions	Cria um ou mais relacionamentos ao conjunto de relacionamentos do divulgador (*publisher*).
delete_binding	Exclui uma ou mais instancias de *bindingTemplate* do registro.
delete_business	Exclui um ou mais registros de negócio do registro.
delete_publisherAssertions	Exclui um relacionamento específico do conjunto de relacionamentos do divulgador (*publisher*).
delete_service	Exclui um ou mais *businessServices* do registro
delete_tModel	Exclui uma ou mais estruturas *tModel*.
discard_authToken	Utilizado para informar a um nó que a informação de autenticação transmitida não é mais válida e deve ser descartada.
get_assertionStatusReport	Relata o *status* dos relacionamentos atuais de *publisherAssertions* que envolvem qualquer um dos registros de negócio que são gerenciados pelo divulgador que fez a requisição.
get_authToken	Utilizado para obter uma informação de autenticação.
get_publisherAssertions	Utilizado para obter todos os grupos de *publisherAssertion* associados a um divulgador.
get_registeredInfo	Utilizado para obter uma lista de todos os *businessEntity* e *tModel* de um divulgador.
save_binding	Utilizado para salvar ou atualizar um elemento *bindingTemplate*.

save_business	Utilizado para salvar ou atualizar informações sobre um *businessEntity*.
save_service	Utilizado para incluir ou atualizar um ou mais elementos *businessService*.
save_tModel	Utilizado para incluir ou atualizar um ou mais elementos *tModel*.
set_publisherAssertions	Utilizado para salvar o todo o conjunto de relacionamentos associados a um divulgador.

Tabela 4: Comandos da API de publicação (*publish*) em um registro UDDI

5.4.2 Detalhes da implementação do módulo UDDI

Os pacotes *uddi.util* e *uddi.impl* contêm o código fonte das ferramentas de interação com o registro UDDI.

Foi escolhido pela separação dessas funções em pacotes separados, pois no pacote *uddi.impl* estão localizadas as classes essenciais e mais utilizadas para a publicação e localização de serviços Web em UDDI, e que podem ser executadas em tempo de execução pelo invocador. A Figura 41 ilustra o diagrama de classes desse pacote.

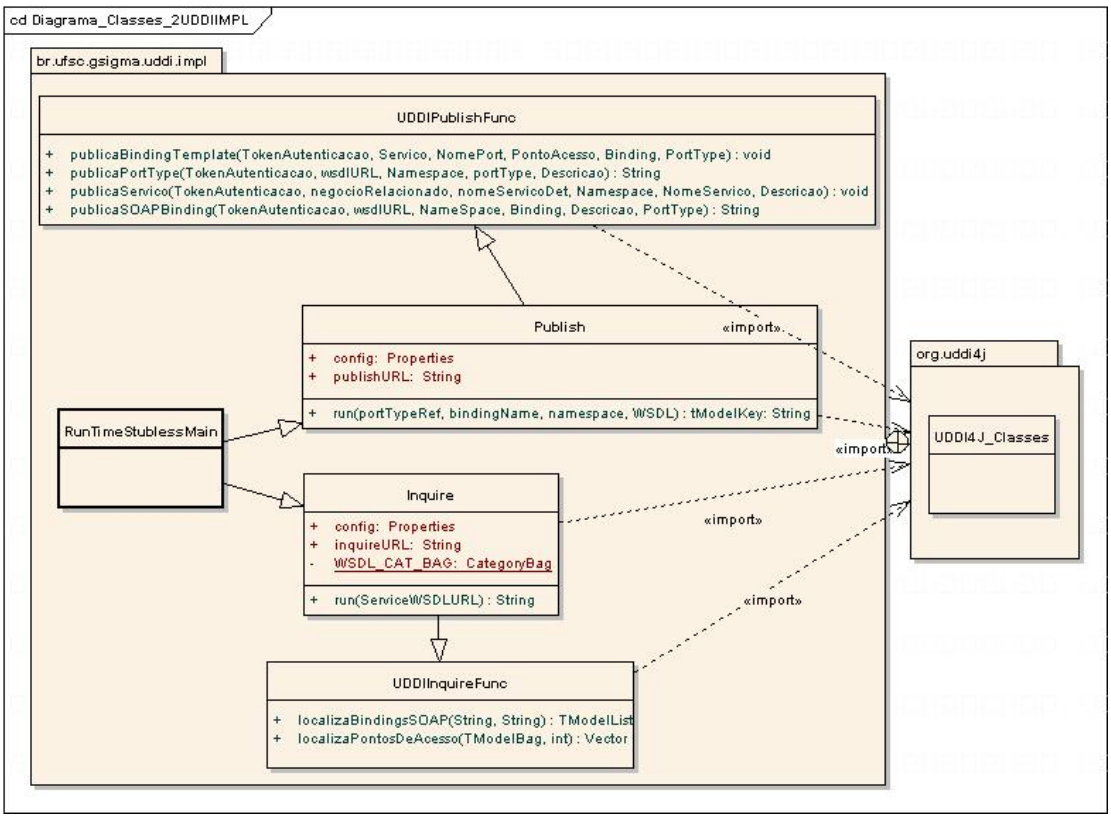


Figura 41: Diagrama de classes do pacote *uddi.impl*

Já no pacote *uddi.util* estão as classes que são mais voltadas para a manutenção e administração dos serviços e negócios – atualização e exclusão – e também contém algumas funções individuais que são menos utilizadas. Estas ferramentas normalmente não são utilizadas em tempo de execução. A Figura 42 ilustra o diagrama de classes desse pacote.

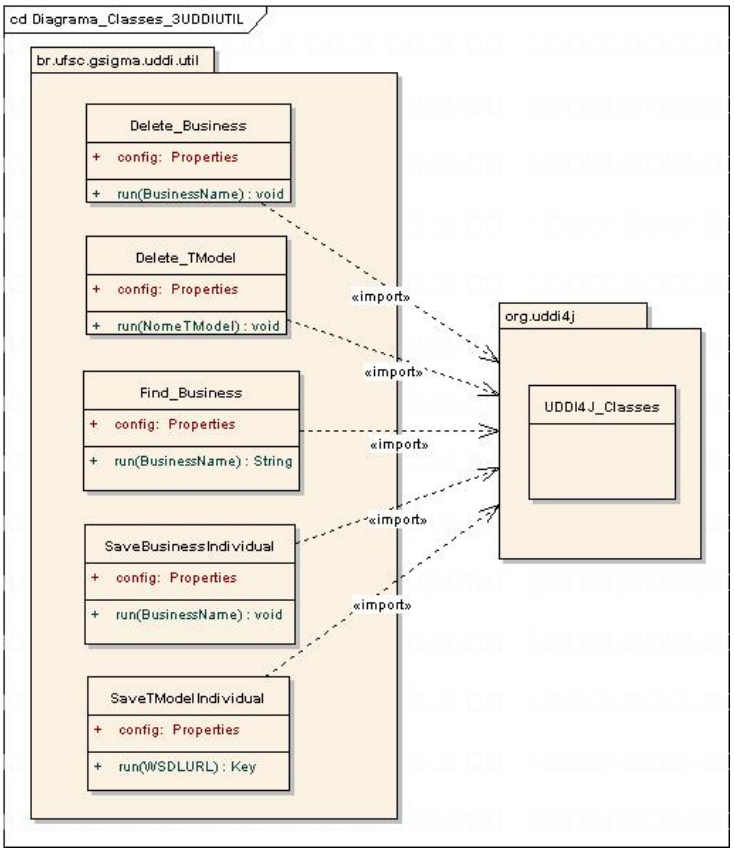


Figura 42: Diagrama de classes do pacote *uddi.util*

5.5 Disponibilização do protótipo

O protótipo de invocação interoperável e *stubless* desenvolvido tem a finalidade de dar suporte à invocação de serviços Web por parte do consumidor de serviços. Essa ferramenta foi escolhida por ser disponibilizada em formato de uma biblioteca (um arquivo *.jar*), onde desenvolvedores, aplicações cliente e componentes de portal podem incluir e utilizar esta biblioteca sempre que houver a necessidade de localizar e invocar um serviço em tempo de execução.

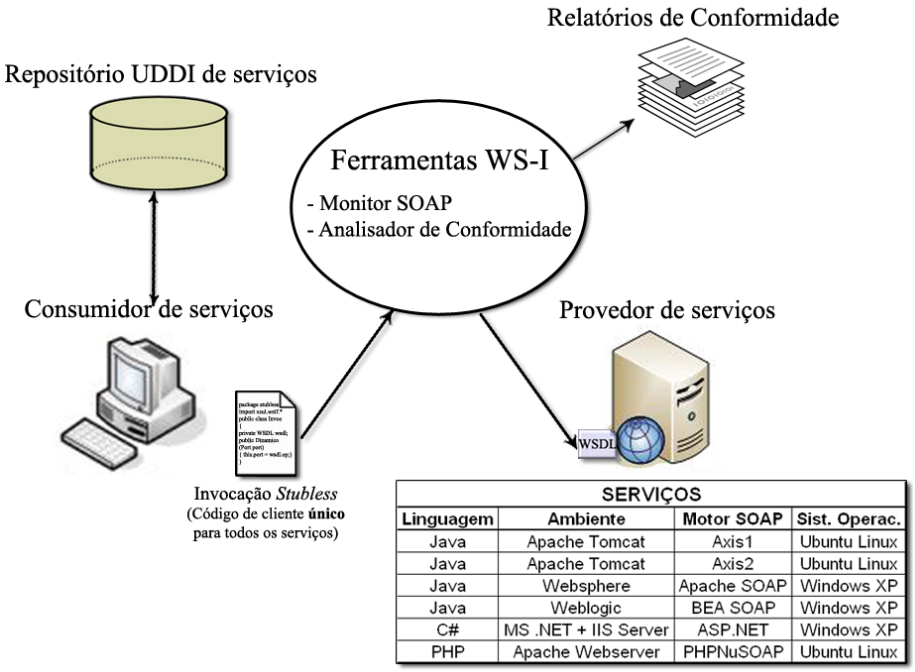
Para invocar um serviço, basta instanciar um objeto da classe *StublessDynamicInvocation* , que é a classe responsável pelo gerenciamento de todas as

operações relacionadas com o processamento de serviços Web, incluindo a sua interface WSDL, extração de operações e a invocação dinâmica propriamente dita. Os parâmetros de entrada que são utilizados pelos métodos desta classe são a URL da descrição WSDL do serviço, a operação a ser executada, e os parâmetros de entrada. As operações de publicação e consulta em UDDI são gerenciadas pelas classes *Publish* e *Inquiry* respectivamente, onde os parâmetros são o nome do serviço que se deseja publicar ou localizar, neste último caso retornando uma URL de uma descrição WSDL, que é utilizada pela classe *StublessDynamicInvocation*.

5.6 Avaliação e testes do modelo proposto

A avaliação do protótipo foi feita através de testes de invocação em diferentes serviços Web. A princípio, os serviços já desenvolvidos anteriormente nas plataformas descritas na seção 5.1 foram utilizados para se testar a invocação dinâmica e *stubless*, inclusive avaliando a real interoperabilidade, utilizando as ferramentas de teste da WS-I, deste invocador para cada um dos tipos de serviços existentes. A Figura 43 ilustra esse cenário.

Todas as interações envolvidas na avaliação são descritas no diagrama UML de seqüência da Figura 44. Neste caso, o cliente interage com o serviço em tempo de execução, sem a necessidade da geração prévia dos *proxies* e *stubs*.



SERVIÇOS			
Linguagem	**Ambiente**	**Motor SOAP**	**Sist. Operac.**
Java	Apache Tomcat	Axis1	Ubuntu Linux
Java	Apache Tomcat	Axis2	Ubuntu Linux
Java	Websphere	Apache SOAP	Windows XP
Java	Weblogic	BEA SOAP	Windows XP
C#	MS .NET + IIS Server	ASP.NET	Windows XP
PHP	Apache Webserver	PHPNuSOAP	Ubuntu Linux

Figura 43: Cenário de teste de interoperabilidade do invocador *stubless*

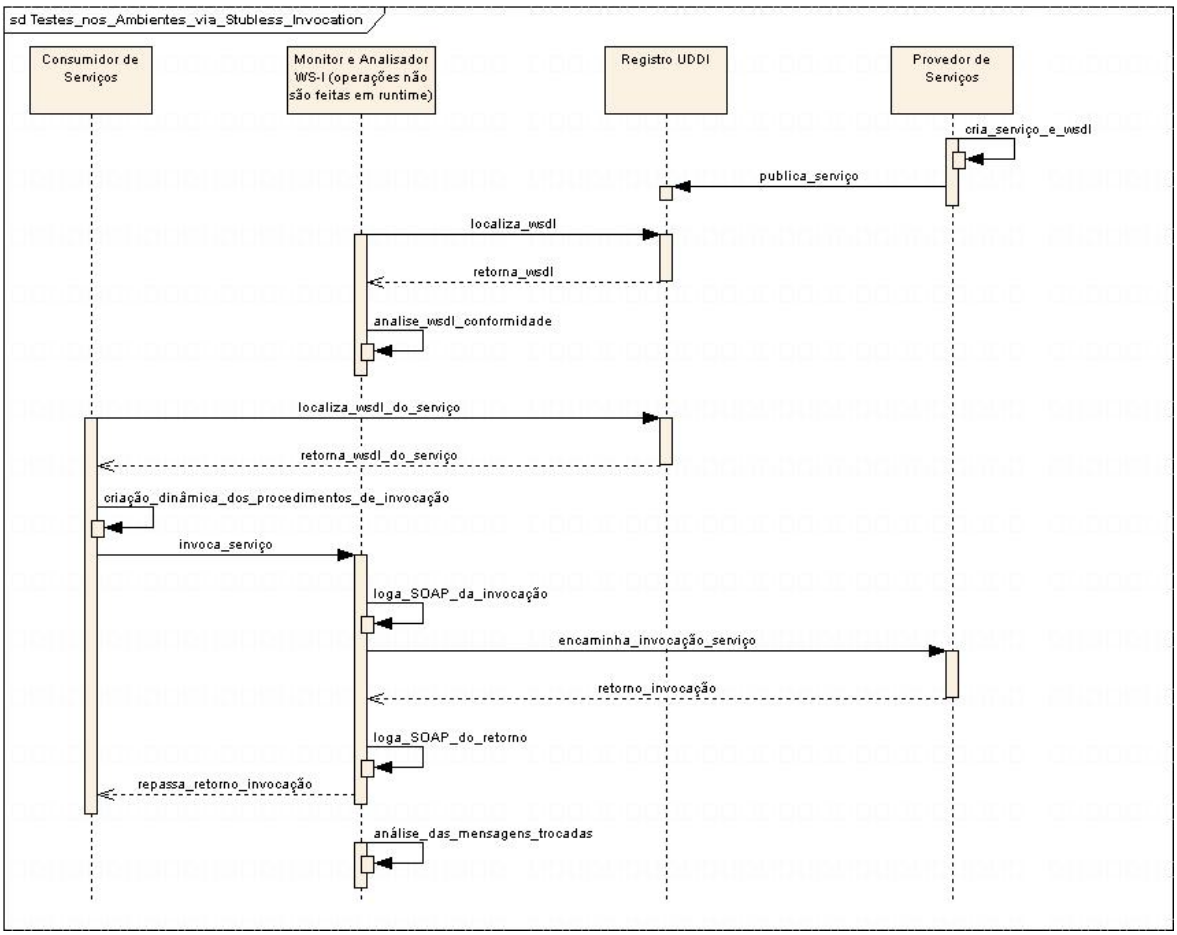


Figura 44: Diagrama de seqüência do cenário de testes e avaliação da invocação *stubless*

5.6.1 Avaliação do documento WSDL

Para se atingir uma maior interoperabilidade entre serviços, um fator inicial que deve ser considerado é que a descrição WSDL deste serviço seja interoperável. A avaliação de um serviço Web é feita através da análise de conformidade de seu documento WSDL. Estas ferramentas ainda não são visuais e automáticas, sendo dependentes de configurações manuais de arquivos XML. Para avaliar um documento WSDL, as seguintes informações são necessárias:

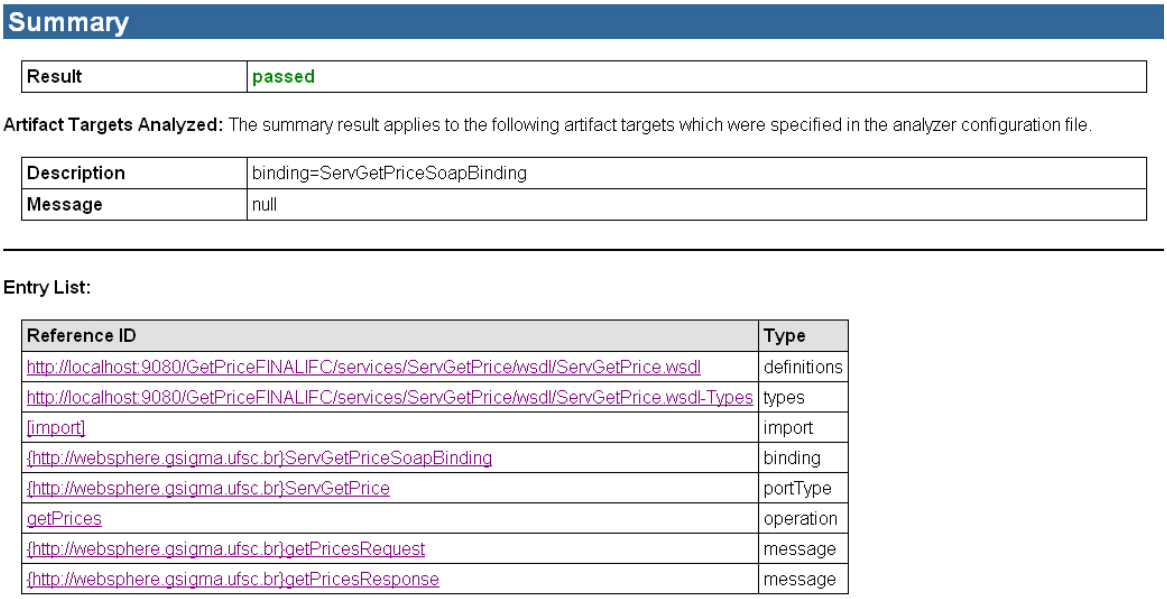
- A URI do espaço de nomes utilizado pelo WSDL.
- O nome da ligação SOAP que é definida no WSDL.
- Localização do arquivo WSDL.
- A URI do ponto de acesso do serviço em si.

Abaixo, a Figura 45 ilustra um exemplo de parte do arquivo de configuração do analisador, incluindo todas as informações descritas acima:

```
<wsi-analyzerConfig:wsdlReference>
   <wsi-analyzerConfig:wsdlElement type="binding" namespace="http://websphere.gsigma.ufsc.br">
      ServGetPriceSoapBinding
   </wsi-analyzerConfig:wsdlElement>
   <wsi-analyzerConfig:wsdlURI>
      http://localhost:9080/GetPriceFINALIFC/services/ServGetPrice/wsdl/ServGetPrice.wsdl
   </wsi-analyzerConfig:wsdlURI>
   <wsi-analyzerConfig:serviceLocation>
      http://localhost:9080/GetPriceFINALIFC/services/ServGetPrice
   </wsi-analyzerConfig:serviceLocation>
</wsi-analyzerConfig:wsdlReference>
```

Figura 45: Exemplo do arquivo de configuração do analisador WS-I

O relatório de conformidade resultante da avaliação acima é gerado, também em XML, onde são descritos detalhes da interoperabilidade do documento WSDL. Aplicou-se no relatório em XML um *template* XSLT, que é utilizado para gerar um relatório mais visível para o usuário final. Um exemplo de parte desse relatório é ilustrado na figura abaixo:



Summary

Result	passed

Artifact Targets Analyzed: The summary result applies to the following artifact targets which were specified in the analyzer configuration file.

Description	binding=ServGetPriceSoapBinding
Message	null

Entry List:

Reference ID	Type
http://localhost:9080/GetPriceFINALIFC/services/ServGetPrice/wsdl/ServGetPrice.wsdl	definitions
http://localhost:9080/GetPriceFINALIFC/services/ServGetPrice/wsdl/ServGetPrice.wsdl-Types	types
[import]	import
{http://websphere.gsigma.ufsc.br}ServGetPriceSoapBinding	binding
{http://websphere.gsigma.ufsc.br}ServGetPrice	portType
getPrices	operation
{http://websphere.gsigma.ufsc.br}getPricesRequest	message
{http://websphere.gsigma.ufsc.br}getPricesResponse	message

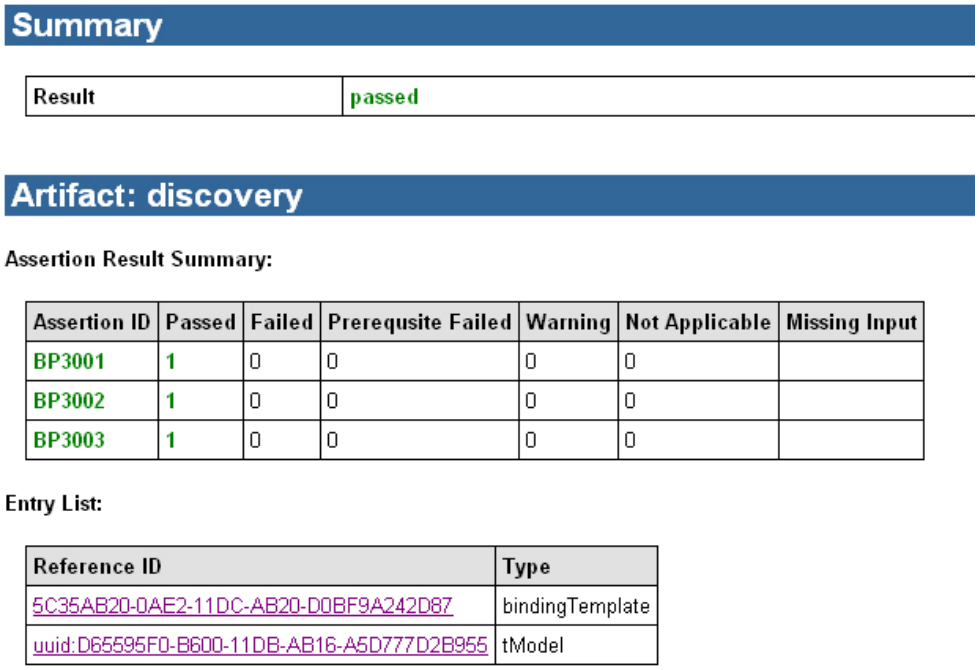
Figura 46: Exemplo de relatório de conformidade de WSDL com o *Basic Profile*

5.6.2 Avaliação da publicação em UDDI

A forma como um serviço e sua interface são publicados em um repositório UDDI deve seguir um padrão, de forma que qualquer cliente/consumidor de serviços, desenvolvido em qualquer linguagem e plataforma, seja capaz de localizar e invocar o serviço de maneira interoperável.

Para avaliar a interoperabilidade do publicador e localizador de serviços em UDDI, o analisador da WS-I foi utilizado. Este analisador, avalia os dois principais artefatos UDDI responsáveis pela publicação de serviços e interfaces, que são o *bindingTemplate* e o *tModel*.

O relatório de conformidade resultante da avaliação da publicação em UDDI foi efetivado com sucesso, significando que as ferramentas de UDDI desenvolvidas estão em conformidade com a WS-I. Um exemplo de parte desse relatório é ilustrado na figura abaixo:



Summary

Result	passed

Artifact: discovery

Assertion Result Summary:

Assertion ID	Passed	Failed	Prerequsite Failed	Warning	Not Applicable	Missing Input
BP3001	1	0	0	0	0	
BP3002	1	0	0	0	0	
BP3003	1	0	0	0	0	

Entry List:

Reference ID	Type
5C35AB20-0AE2-11DC-AB20-D0BF9A242D87	bindingTemplate
uuid:D65595F0-B600-11DB-AB16-A5D777D2B955	tModel

Figura 47: Exemplo de relatório de conformidade de UDDI com o *Basic Profile*

5.6.3 Avaliação das mensagens SOAP trocadas

As mensagens SOAP trocadas entre o provedor e consumidor do serviço também podem ser avaliadas, e validadas em relação a sua conformidade com o *Basic Profile* da WS-I. Na avaliação do invocador *stubless*, as mensagens trocadas foram capturadas e arquivadas pela ferramenta de monitoramento da WS-I, para a então execução do analisador que se baseia nesses registros (*logs*) para determinar se as mensagens (tanto vindas do provedor, quanto do consumidor de serviços) estão ou não em conformidade com o *Basic Profile*.

A configuração do monitor também é feita manualmente, através da edição de um arquivo XML. A Figura 48 ilustra um exemplo de parte do arquivo de configuração do monitor, que utiliza a abordagem de monitoramento *man-in-the-middle*, que funciona como um receptor, e encaminhador de mensagens:

```
<wsi-monConfig:manInTheMiddle>
  <wsi-monConfig:redirect>
     <wsi-monConfig:comment>Porta 4001 -> 9080 (IBM WEBSPHERE)</wsi-monConfig:comment>
     <wsi-monConfig:listenPort>4001</wsi-monConfig:listenPort>
     <wsi-monConfig:schemeAndHostPort>
          http://localhost:9080
     </wsi-monConfig:schemeAndHostPort>
     <wsi-monConfig:maxConnections>1000</wsi-monConfig:maxConnections>
     <wsi-monConfig:readTimeoutSeconds>15</wsi-monConfig:readTimeoutSeconds>
  </wsi-monConfig:redirect>
 <wsi-monConfig:redirect>
</wsi-monConfig:manInTheMiddle>
```

Figura 48: Exemplo de arquivo de configuração do monitor

No exemplo acima, o monitor está escutando por requisições na porta 4001. Portanto, para ser possível o monitoramento das mensagens, o cliente é configurado para enviar suas requisições de invocação para a porta 4001, que é a porta do monitor, que se encarrega de encaminhar a requisição para a porta 9080, que é a porta do IBM Websphere.

Além do IBM Websphere, essa avaliação foi efetuada também com as outras plataformas utilizadas, conforme apresentado na Figura 43. Após a geração do *log* dos serviços e mensagens para todas as plataformas, foi então executado o analisador, que então gerou o relatório de conformidade, que é apresentado em maiores detalhes no APÊNDICE A .

Para todos os serviços das plataformas, o relatório de conformidade foi favorável, com exceção do Apache Axis2. Isso se deve ao fato do Axis2 ser um motor SOAP novo, e inclui novos tipos de cabeçalhos nas mensagens, que o analisador identifica como “não interoperáveis”. Porém, a não-conformidade de uma mensagem gerada por um serviço Web não significa necessariamente que este serviço não será acessível, ou impossibilitado de ser invocado. O relatório de conformidade avalia com base em **todas** as recomendações do *Basic Profile*, que inclui detalhes referentes a encaminhamento de mensagens e uso de cabeçalhos específicos.

No entanto, pelos testes executados, foi verificado também que as atuais plataformas de disponibilização de serviços Web já são capazes de lidar com alguns cabeçalhos SOAP específicos, que ainda não são recomendados pela WS-I, porém são suportados por essas plataformas.

Novos *profiles* estão em desenvolvimento pela WS-I, incluindo novas recomendações e restrições, que incluem, entre outros, detalhes sobre extensões para serviços Web, que foram descritos na seção 2.2.6.

5.6.4 Considerações sobre a avaliação dos serviços e interoperabilidade

A invocação em todos os serviços foi efetuada com sucesso, não havendo a necessidade de nenhuma adequação no serviço para que fosse possível sua invocação. É importante lembrar que os serviços desenvolvidos nas plataformas estão em conformidade com o *Basic Profile* 1.0 da WS-I.

Porém, para fins de avaliação do invocador *stubless*, um serviço foi desenvolvido em não-conformidade com o *Basic Profile,* utilizando o estilo *rpc/encoded.* Apesar do serviço não estar em conformidade, a invocação de serviços utilizando o estilo *rpc/encoded* também foi efetuada com sucesso. O suporte ao estilo *rpc/encoded* é um fator muito importante, pois, por ser o estilo padrão do Axis, ainda é muito utilizado por diversos serviços Web legados.

Através da avaliação efetuada foi possível verificar um aumento da agilidade na invocação de serviços de diferentes tecnologias. É possível verificar também a diminuição do tempo de geração de código, pois para invocar diversos serviços é utilizado apenas um código cliente, diminuindo a complexidade de *n* clientes para 1 cliente.

Não foi possível comparar e avaliar o presente trabalho com os outros trabalhos relacionados e semelhantes, pois os protótipos não são disponibilizados publicamente. Desta forma, a métrica de avaliação passou a ser simplesmente a garantia de invocação (dinâmica e *stubless*) e interoperabilidade (seguir as recomendações da WS-I), provendo assim uma solução que fosse capaz de satisfazer os problemas e objetivos que foram descritos nas seções 1.2 e 1.3.

5.7 Principais dificuldades

O invocador *stubless* é voltado para o reuso de diferentes serviços individuais que são desacoplados e disponibilizados em diversas plataformas, suportando diferentes processos de negócios nas organizações.

Portanto, com a intenção de criar um cenário realmente heterogêneo, que possa validar a abordagem proposta, foi necessário o desenvolvimento de diversos serviços Web em cada uma das plataformas. O problema está na metodologia de desenvolvimento e disponibilização dos serviços Web nas plataformas, que varia muito entre os diferentes fabricantes. Mesmo em plataformas que utilizam a mesma linguagem, como o Websphere, Weblogic, Axis1 e Axis2, que usam Java, cada um tem sua particularidade na implementação e exposição dos serviços.

No entanto, cada uma das implementações nas plataformas contribuiu de alguma maneira para a avaliação da interoperabilidade entre diferentes serviços Web, pois através destas implementações foi possível examinar e avaliar problemas comuns de interoperabilidade nos protocolos de serviços Web, que são encontrados na literatura.

Um outro problema está na validação dos dados de entrada do invocador, quando um usuário deseja invocar um serviço Web que expõe um tipo complexo. Neste caso específico, o usuário/aplicação cliente deve passar como parâmetro um documento XML que contenha a descrição do tipo complexo, descrito no esquema do serviço Web a ser invocado.

5.8 Sumário do capítulo

A seção 5.1 apresentou detalhes de todas as plataformas de serviços Web que foram instaladas e configuradas para a disponibilização e execução dos serviços avaliados. Nesta etapa, foi concebido também um cenário onde todos os serviços Web que foram implementados se invocavam entre si, utilizando a abordagem tradicional de invocação, que consiste na utilização da descrição WSDL do serviço - que é conhecida previamente - para a então geração dos *stubs* que irão suportar a invocação deste serviço.

A seção 5.2 apresenta detalhes de como os serviços foram desenvolvidos nestas plataformas, também apresentando detalhes relacionados com a interoperabilidade destas plataformas, incluindo uma lista de alguns tipos de dados que devem ser evitados ao se expor uma aplicação como serviço Web. Após o desenvolvimento do *Basic Profile,* que prove direcionamentos, recomendações para os fabricantes de plataformas e também melhores práticas para o desenvolvimento de serviços Web, muitos dos problemas de interoperabilidade foram solucionados.

Porém, apesar desses problemas terem sido solucionados, muitas das invocações de serviços são feitas em tempo de desenvolvimento, o que resultou na concepção de uma abordagem que propõe a localização e invocação dinâmica de serviços Web. A abordagem propõe o uso das APIs do Apache WSIF, para dessa forma invocar diferentes serviços Web que sejam desenvolvidos em outras tecnologias além do SOAP, como, por exemplo, EJBs, REST e JCA.

A seção 5.3 apresentou detalhes do desenvolvimento do invocador *stubless*, além de detalhes do funcionamento da API do WSIF que foi utilizada, além do cenário o qual esta pode ser aplicada, enquadrando principalmente dentro da área de redes colaborativas. A seção 5.4 descreve em detalhes o módulo que é responsável pelas interações com o registro UDDI, fazendo a publicação e localização de serviços Web em tempo de execução.

Após todas as implementações, uma fase importante foi a avaliação da abordagem, que consiste nos testes de interoperabilidade do invocador *stubless*, ou seja, a real capacidade de se invocar os diversos serviços que foram desenvolvidos e disponibilizados nas diferentes plataformas que foram instaladas. A seção 5.5 apresentou detalhes da forma como essa avaliação foi feita, utilizando as ferramentas providas pela organização WS-I. O Apêndice A apresenta parte dos relatórios gerados por essas ferramentas, indicando a interoperabilidade e conformidade dos serviços com os requisitos da WS-I.

CAPÍTULO 6
Conclusões

Este capítulo apresenta a conclusão da dissertação através da revisão das motivações e objetivos já apresentados no capítulo 1, seguido da visão geral do trabalho e das principais contribuições desta dissertação. Na última seção são apresentadas algumas possibilidades de trabalhos futuros a serem pesquisados e desenvolvidos.

6.1 Revisão das motivações e objetivos

Este trabalho apresentou um estudo detalhado sobre interoperabilidade em serviços Web e de suas diferentes plataformas de disponibilização e suporte, propondo também uma abordagem para a interoperabilidade sem costuras entre estas plataformas heterogêneas de serviços Web, aplicados no cenário de Redes Colaborativas de Organizações.

A adoção de Arquiteturas Orientadas a Serviços e das tecnologias de serviços Web tem se tornado cada vez mais comum dentro das organizações. De acordo com um estudo do instituto americano de análise de mercado e consultoria *Gartner Group*, até 2010 acredita-se que 70% de todas as aplicações em uma empresa serão desenvolvidas na filosofia SOA/serviços (CEARLEY *et al.*, 2005).

No entanto, apesar dos serviços Web serem amparados por padrões abertos, diversos problemas de interoperabilidade surgem na prática. Por exemplo, quando organizações desejam fazer seus serviços, que normalmente são desenvolvidos em diferentes plataformas e linguagens se comunicarem plenamente e sem costuras entre si.

Portanto, a possibilidade de uma maior interoperabilidade entre serviços Web pertencentes a diferentes organizações e seus sistemas de TIC, contribui efetivamente para o aumento da colaboração e conseqüentemente da produtividade, qualidade e dinamicidade dos processos de negócio das organizações. Ao tornar acessíveis publicamente serviços que antes eram isolados, são abertas novas possibilidades de negócio, além de uma maior agilidade e facilidade no acesso a informações por parte de membros de uma rede colaborativa.

6.2 Visão geral do trabalho

Neste trabalho, foram pesquisadas diferentes plataformas de suporte a serviços Web, avaliando a interoperabilidade de cada uma destas, tendo com base o cenário heterogêneo descrito na seção 5.2.1. Essa avaliação foi feita com base na invocação tradicional de serviços Web, onde os *stubs* e *proxies* são gerados previamente com base na descrição do serviço, para a sua posterior invocação. Este trabalho ainda propõe uma abordagem para aumentar a interoperabilidade entre diferentes serviços Web, onde a invocação dos serviços é feita de maneira dinâmica, transparente e *stubless*, dessa forma ocultando a complexidade e detalhes de implementação destes serviços.

Os tipos de interoperabilidade que foram cobertos pela abordagem proposta são a técnica e pragmática, que tratam da habilidade de dois sistemas heterogêneos trocarem informações e também da capacidade de uma aplicação invocar um serviço. Isso significa que a abordagem proposta garante que um cliente será capaz efetivamente de invocar de maneira interoperável um serviço Web, de acordo com suas necessidades de negócio, independentemente da plataforma computacional, do sistema operacional, das tecnologias e linguagens de implementação, e da localização geográfica do serviço. Para a concepção desta abordagem, a API do WSIF foi utilizada, onde, na seção 5.3.3, o seu funcionamento e arquitetura foram descritos em maiores detalhes, como forma de prover uma documentação e suporte para trabalhos futuros, onde a interoperabilidade pode ser avaliada em todos os Provedores do WSIF.

Conforme dito anteriormente, a interoperabilidade é uma área de intensa pesquisa, em diferentes âmbitos e níveis (ATHENA-IP, 2004). Consequentemente, o objetivo desse trabalho não é o de prover um estudo exaustivo sobre a interoperabilidade de forma definitiva e holística, e sim, prover um estudo sobre questões da interoperabilidade dentro do escopo de serviços Web, mais especificamente na forma de invocação desses serviços em uma rede colaborativa de organizações, no intuito de prover uma maior dinamicidade e reusabilidade destes serviços.

6.3 Contribuições e conclusões

Dentro dos objetivos específicos propostos na seção 1.3.2, e das atividades executadas para atingi-los, essa dissertação contribui nos seguintes aspectos:

- A principal contribuição deste trabalho foi a concepção de um modelo de suporte a invocação de serviços Web sob-demanda, isto é, a partir do momento que uma aplicação cliente precisa invocar um serviço, este serviço será localizado e invocado, independentemente da plataforma, linguagem e tecnologia utilizada. Do ponto de vista do cliente, o invocador *stubless* pode ser visto como um *middleware* de suporte a invocação, onde clientes o utilizam para executar invocações de maneira flexível e dinâmica.
- Desenvolvimento de um protótipo de *software* que implementa a abordagem de interoperabilidade proposta, que é disponibilizada em forma de uma biblioteca que pode ser incluída e utilizada por diferentes consumidores de serviços pertencentes a uma rede colaborativa.
- Sugestão de uma maior adoção e aplicação dos mecanismos “multi-Provedores” proposto pelas APIs do WSIF, principalmente no âmbito de redes colaborativas, devido a sua capacidade de lidar com diferentes protocolos, bastando que estes sejam definidos como serviços Web, isto é, tenham sua descrição definida utilizando a WSDL.

Pelos experimentos executados, foi possível verificar que através da abordagem proposta é possível aumentar a interoperabilidade e flexibilidade na invocação de serviços, e também aumentar a reusabilidade de serviços, principalmente em um cenário dinâmico e volátil, como as redes colaborativas, onde diferentes serviços “entram” e “saem” do cenário com muita freqüência.

Além disso, isso cria um nível mais elevado de potencial de colaboração entre as empresas, uma vez que elas podem se utilizar de serviços umas das outras, independentemente de quais plataformas e tecnologias foram utilizadas nas suas implementações. Em uma escala maior, este trabalho cria uma importante base para uma integração e interoperabilidade plena entre sistemas e empresas, dentro de um cenário onde se prevê a criação e existência de federações de serviços (RABELO *et al.*, 2007). Nesta, serviços de vários provedores/empresas serão logicamente agrupados de forma a que parceiros possam (re)utilizar serviços uns dos outros sem problemas maiores de interoperabilidade, dando aos consumidores/aplicações clientes melhores condições de agilidade, flexibilidade para suportar mudanças nos processos e, por fim, economia.

Porém, a interoperabilidade plena entre os sistemas de TIC das organizações está ainda muito nova. Outras abordagens e estudos mais aprofundados dos aspectos mencionados neste trabalho devem ser desenvolvidos. Alguns destes aspectos são apresentados na seção 6.4.

6.3.1 Limitações

Conforme descrito anteriormente, o protótipo de invocação *stubless*, uma das bibliotecas utilizadas é o Apache WSIF, que suporta a invocação de serviços Web disponibilizados em diferentes tecnologias e protocolos, como o EJB, JCA, etc. Essa seção apresenta algumas limitações relacionadas com esse protótipo:

- A implementação e avaliação em relação a interoperabilidade do invocador foi feita apenas para serviços Web SOAP.
- O protótipo de software foi implementado na linguagem Java, pois a biblioteca WSIF de suporte a multi-protocolos é desenvolvida e suportada apenas em Java. Consequentemente, o código do cliente também se torna dependente do Java.
- Conforme dito anteriormente, no caso de múltiplas ligações em um documento WSDL, a escolha da ligação a ser utilizada fica a cargo do programador.
- A localização de serviços foi efetuada com sucesso, porém serviços publicados em UDDI utilizando outras formas e abordagens de terceiros não foram testados.
- Redução da complexidade para o consumidor de serviços, porém aumento de processamento, devido a constante varredura do documento WSDL.

6.3.2 Produção bibliográfica

Como forma de validar a abordagem proposta no presente trabalho junto a comunidade científica, foi escrito um artigo, com o título “Uma abordagem para a integraçao plena de sistemas empresariais através de serviços Web”, que foi submetido ao congresso nacional SBAI (Simpósio Brasileiro de Automação Inteligente). No momento da

escrita deste documento, o artigo não havia sido publicado, com previsão de notificação de aceitação por parte da comissão para Julho/2007.

A abordagem proposta por este trabalho também será utilizada pelo projeto ECOLEAD (ECOLEAD, 2007), onde dois relatórios técnicos já foram produzidos (RATTI *et al.*, 2007, RODRIGO *et al.*, 2007), e a previsão de inclusão e utilização da abordagem proposta por este trabalho, que será disponibilizada em forma de uma biblioteca, é prevista para Outubro/2007.

6.4 Trabalhos futuros

Este trabalho pode ser complementado, com o estudo e pesquisa em diferentes campos. Algumas sugestões de trabalhos futuros e tópicos em aberto são apresentados a seguir.

É importante ressaltar que os tópicos descritos abaixo apenas destacam algumas alternativas que podem futuramente ser pesquisadas e desenvolvidas, porém, derivações do atual trabalho podem surgir como conseqüência de interesses específicos.

6.4.1 Avaliação de interoperabilidade com outros Provedores

Apesar de o WSIF, através de seus diversos Provedores, permitir a invocação de serviços Web disponibilizados por diferentes tecnologias e protocolos, como o EJB e o JCA, neste trabalho a avaliação de interoperabilidade nas diferentes plataformas foi feita somente para serviços Web SOAP.

Portanto, é necessário que se avalie a real interoperabilidade das outras tecnologias e protocolos suportados, para, dessa forma, atingir uma interoperabilidade plena entre os vários sistemas heterogêneos providos por diferentes organizações em uma rede colaborativa de organizações.

6.4.2 Buscas de serviços baseados em semânticas e QoS

O desenvolvimento do módulo de buscas de serviços em UDDI em tempo de execução tem como finalidade tornar a invocação de serviços Web algo realmente transparente e dinâmico para os usuários e aplicações cliente.

Porém, a busca é feita se baseando apenas no quesito **sintático**, i.e., a localização do serviço é feita apenas baseando-se em palavras chave, escolhidas previamente pelo usuário. Portanto, é preciso que novas formas de se localizar serviços sejam providas, baseando-se em **semânticas**, tanto na descrição do serviço, como na forma como este serviço é publicado, de forma que a busca por serviços seja mais inteligente, eficiente e precisa.

A busca em UDDI também pode se basear em quesitos de qualidade de serviço (QoS), como, por exemplo, a escolha por serviços baseando-se na sua disponibilidade, velocidade e preço – em um cenário futurista, onde serviços Web serão disponibilizados por um determinado preço de utilização.

Um trabalho futuro em relação a UDDI pode ser também a extensão do modelo de publicação e localização proposto, para outros protocolos, além do SOAP.

6.4.3 Interoperabilidade entre serviços Web e redes P2P

Apesar da tecnologia P2P ter tido sucesso até agora, com suas redes de compartilhamento e troca de arquivos, ainda falta um consenso em como as aplicações de serviços Web devem ser construídas e disponibilizadas, e qual a semântica que deve ser suportada, de forma que os serviços Web nessas redes possam ser totalmente interoperáveis.

6.4.4 Avaliação do REST como forma de aumentar a interoperabilidade em serviços Web

O REST (*REpresentational State Transfer*) é um estilo de arquitetura de software para sistemas distribuídos que utiliza o conceito inicialmente proposto para a Internet: uma rede de páginas Web, onde o usuário avança na aplicação através da seleção de ligações, resultando na próxima página (FIELDING, 2000).

Esse conceito tem sido também aplicado dentro do âmbito de serviços Web, como uma alternativa ao uso do SOAP. Por ser baseado apenas em XML, e nas operações do HTTP (POST, GET, etc.) esse estilo é considerado mais interoperável quando utilizado com serviços Web (VOGELS, 2003).

6.4.5 Interoperabilidade entre serviços Web e serviços em Grade

A Arquitetura Aberta de Serviços em Grade (OSGA – *Open Grid Services Architecture*) é uma arquitetura baseada na integração dos conceitos e tecnologias de serviços Web e computação em Grade. Através do uso da WSDL, a arquitetura OSGA define extensões para serviços Web que especificam as propriedades de aplicações em Grade.

Estas extensões, e suas definições na especificação da OSGA, devem ser efetivamente avaliadas, de forma que se garanta uma maior interoperabilidade e dinamicidade entre serviços Web e serviços em Grade sendo executados. Com uma maior interoperabilidade entre estes serviços, é possível garantir a composição híbrida e mais flexível de serviços.

APÊNDICE A
Relatório de conformidade

Abaixo, é apresentado um dos relatórios de conformidade com o *Basic Profile* que foram gerados com base em um dos serviços Web disponibilizados no IBM Websphere. É importante ressaltar que este relatório foi gerado para todos os serviços de todas as plataformas, no entanto, por serem muito extensos, não são incluídos neste documento.

A sessão A.1 apresenta o relatório em formato XML, e a sessão A.2 apresenta o mesmo relatório aplicado a uma folha de estilo XSL. As opções de conformidade podem ser:

- *Passed*: O item/artefato analisado está em conformidade com o *Basic Profile*
- *Not Applicable*: O item/artefato analisado não é aplicável a nenhuma restrição/regra do *Basic Profile*.
- *Warning*: O item/artefato analisado está em conformidade, porém uma alteração é recomendada (por exemplo, utilizar HTTP versão 1.1 ao invés de 1.0).
- *Failed*: O item/artefato não está em conformidade com o *Basic Profile*, podendo causar problemas de interoperabilidade.

A.1 Relatório XML de conformidade referente ao serviço na plataforma Websphere

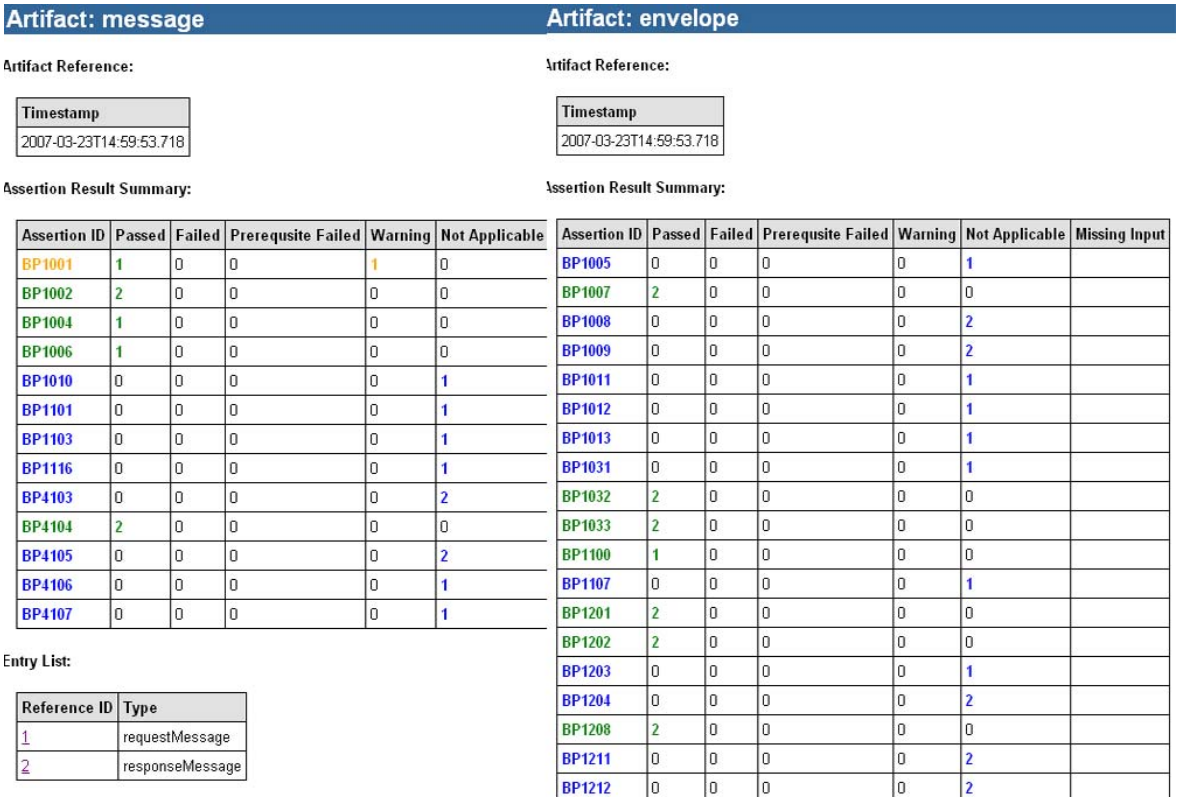
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="../common/xsl/report.xsl" type="text/xsl" ?>
<report name="WS-I Basic Profile Conformance Report."  timestamp="2007-03-23T15:02:47.296"
   xmlns="http://www.ws-i.org/testing/2004/07/report/"  xmlns:wsi-report="http://www.ws-
i.org/testing/2004/07/report/"  xmlns:wsi-log="http://www.ws-i.org/testing/2003/03/log/"
   xmlns:wsi-analyzerConfig="http://www.ws-i.org/testing/2004/07/analyzerConfig/"
   xmlns:wsi-monConfig="http://www.ws-i.org/testing/2003/03/monitorConfig/"
   xmlns:wsi-assertions="http://www.ws-i.org/testing/2004/07/assertions/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <analyzer version="1.0.0" releaseDate="2005-07-04">
   <implementer name="WS-I Organization" location="http://www.ws-i.org"/>
   <environment> <runtime name="Java(TM) SE Runtime Environment" version="1.6.0_01-b06"/>
```

```
      <operatingSystem name="Windows XP" version="5.1"/>
      <xmlParser name="Apache Xerces" version="Xerces-J 2.6.2"/>
    </environment>
    <wsi-analyzerConfig:configuration>
      <wsi-analyzerConfig:verbose>false</wsi-analyzerConfig:verbose>
      <wsi-analyzerConfig:assertionResults type="all" messageEntry="true"
assertionDescription="false" failureMessage="true" failureDetail="true"/>
        <wsi-analyzerConfig:reportFile replace="true" location="report.xml">
      <wsi-analyzerConfig:addStyleSheet href="../common/xsl/report.xsl" type="text/xsl" />
        </wsi-analyzerConfig:reportFile>
        <wsi-analyzerConfig:testAssertionsFile>../../common/profiles/BasicProfile_1.1_TAD.xml</wsi-
analyzerConfig:testAssertionsFile>
        <wsi-analyzerConfig:logFile correlationType="endpoint">../bin/log_monitor.xml</wsi-
analyzerConfig:logFile>
    </wsi-analyzerConfig:configuration>
  </analyzer>
  <artifact type="message">
    <artifactReference timestamp="2007-03-23T14:59:53.718">
    </artifactReference>
    <entry type="requestMessage" referenceID="1">
<wsi-log:messageEntry xsi:type="httpMessageEntry" ID="1" conversationID="1" type="request"
timestamp="2007-03-23T14:59:57.000">
     <assertionResult id="BP1004" result="passed">
      </assertionResult>
      <assertionResult id="BP1006" result="passed">
      </assertionResult>
      <assertionResult id="BP1116" result="notApplicable">
      </assertionResult>
      <assertionResult id="BP1002" result="passed">
      </assertionResult>
      <assertionResult id="BP1001" result="warning">
        <failureMessage xml:lang="en">The message is not sent using HTTP/1.1.</failureMessage>
      <failureDetail xml:lang="en" >POST /GetPriceFINALIFC/services/ServGetPrice HTTP/1.0
Host: localhost:4002
User-Agent: XSUL/2.10.2
Content-Type: text/xml; charset=utf-8
Content-Length: 612
Keep-Alive: 300
SOAPAction: &amp;quot;&amp;quot;
Connection: keep-alive
Element Location:
  lineNumber=42
      </failureDetail>
     </assertionResult>
    </entry>
<entry type="responseMessage" referenceID="2">
<wsi-log:messageEntry xsi:type="httpMessageEntry" ID="2" conversationID="1" type="response"
timestamp="2007-03-23T14:59:57.062">
      <assertionResult id="BP1002" result="passed">
      </assertionResult>
      <assertionResult id="BP1001" result="passed">
      </assertionResult>
      <assertionResult id="BP1010" result="notApplicable">
      </assertionResult>
      <assertionResult id="BP1101" result="notApplicable">
      </assertionResult>
      <assertionResult id="BP1103" result="notApplicable">
      </assertionResult>
      <assertionResult id="BP4103" result="notApplicable">
```

```
      </assertionResult>
   </entry>
  </artifact>

  <artifact type="envelope">
    <artifactReference timestamp="2007-03-23T14:59:53.718">
     <assertionResult id="BP1601" result="passed">
      </assertionResult>
      <assertionResult id="BP1201" result="passed">
      </assertionResult>
      <assertionResult id="BP1701" result="passed">
      </assertionResult>
      <assertionResult id="BP1308" result="passed">
      </assertionResult>
      <assertionResult id="BP1011" result="notApplicable">
      </assertionResult>
      <assertionResult id="BP1204" result="notApplicable">
      </assertionResult>
....
..... <!—continua com os outros testes --> ...
....
  </entry>
  </artifact>
  <summary result="passed">
  </summary>
</report>
```

A.2 Relatório HTML de conformidade referente ao serviço na plataforma Websphere



Artifact: message

Artifact Reference:

Timestamp
2007-03-23T14:59:53.718

Assertion Result Summary:

Assertion ID	Passed	Failed	Prerequsite Failed	Warning	Not Applicable
BP1001	1	0	0	1	0
BP1002	2	0	0	0	0
BP1004	1	0	0	0	0
BP1006	1	0	0	0	0
BP1010	0	0	0	0	1
BP1101	0	0	0	0	1
BP1103	0	0	0	0	1
BP1116	0	0	0	0	1
BP4103	0	0	0	0	2
BP4104	2	0	0	0	0
BP4105	0	0	0	0	2
BP4106	0	0	0	0	1
BP4107	0	0	0	0	1

Entry List:

Reference ID	Type
1	requestMessage
2	responseMessage

Artifact: envelope

Artifact Reference:

Timestamp
2007-03-23T14:59:53.718

Assertion Result Summary:

Assertion ID	Passed	Failed	Prerequsite Failed	Warning	Not Applicable	Missing Input
BP1005	0	0	0	0	1	
BP1007	2	0	0	0	0	
BP1008	0	0	0	0	2	
BP1009	0	0	0	0	2	
BP1011	0	0	0	0	1	
BP1012	0	0	0	0	1	
BP1013	0	0	0	0	1	
BP1031	0	0	0	0	1	
BP1032	2	0	0	0	0	
BP1033	2	0	0	0	0	
BP1100	1	0	0	0	0	
BP1107	0	0	0	0	1	
BP1201	2	0	0	0	0	
BP1202	2	0	0	0	0	
BP1203	0	0	0	0	1	
BP1204	0	0	0	0	2	
BP1208	2	0	0	0	0	
BP1211	0	0	0	0	2	
BP1212	0	0	0	0	2	

Figura 49: Exemplo de relatório de conformidade das mensagens SOAP

Glossário

O domínio da Tecnologia da Informação e Comunicação é fortemente marcado pela utilização de siglas já relativamente estabelecidas no mercado e no meio acadêmico. Com o intuito de facilitar a compreensão do leitor, foi criado um glossário das abreviaturas e siglas utilizadas nesta dissertação.

A

API	–	*Application Programming Interface*

B

B2B	–	*Business-to-Business*
BPM	–	*Business Process Management*

C

CRM	–	*Customer Relationship Management*

D

DII	–	*Dynamic Invocation Interface*
DTD	–	*Document Type Definitions*

E

EAI	–	*Enterprise Application Integration*
EbXML	–	*Electronic Business using Extensible Markup Language*
EDI	–	*Electronic Data Interchange*
EJB	–	*Enterprise Java Beans*
ERP	–	*Enterprise Resource Planning*

H

HTML	–	*HyperText Markup Language*
HTTP	–	*Hypertext Transfer Protocol*

I

IDL	–	*Interface description language*

J

JCA	–	*Java Conector Architecture*
JMS	–	*Java Message Service*

O

OASIS	–	*Organization for the Advancement of Structured Information Standards*
OV	–	Organização Virtual

P

P2P	–	*Peer-to-Peer*
PMEs	–	Pequenas e Médias Empresas

R

RCO	–	Redes Colaborativas de Organizações
REST	–	*Representational State Transfer*
RPC	–	*Remote Procedure Call*

S

SCM	–	*Supply-Chain Management*
SGML	–	*Standard Generalized Markup Language*
SOA	–	*Service Oriented Architecture*
SOAP	–	*Simple Object Access Protocol*

T

TIC	–	Tecnologia da Informação e Comunicação

U

UDDI	–	Universal Description, Discovery and Integration
UML	–	Unified Modeling Language
URI	–	Uniform Resource Identifier

W

W3C	–	World Wide Web Consortium
WAS	–	Websphere Application Server
WS-BPEL	–	Business Process Execution Language for Web Services
WSDL	–	Web Services Description Language
WSIF	–	Web Services Invocation Framework
WSIT	–	Web Services Interoperability Technology

X

XML	–	Extensible Markup Language
XSLT	–	Extensible Stylesheet Language Transformations

Referências Bibliográficas

ABILITIES-PROJECT. (2007). "ABILITIES - Application Bus for InteroperabiLITy In enlarged Europe SMEs." Acessado: 15 de Março, 2007, em http://services.txt.it/abilities/.

ABLONG, T.; BARTLETT, J.; BARTLEY, D.; BUSBY, J.; et al.; 2005. Interoperability Technical Framework for the Australian Government. Technical Report - National Office for the Information Economy. Canberra, Australia.

AKRAM, A.; MEREDITH, D.; ALLAN, R.; 2006. Best Practices in Web Service Style, Data Binding and Validation for use in Data-Centric Scientific Applications. In: UK e-Science All Hands Conference 2006 (AHM 2006) (Nottingham, Reino Unido). *Proceedings*.

ALMEIDA, J. P. A.; PIRES, L. F.; POKRAEV, S.; PRAS, A.; et al.; 2002. Web Services Technologies - State-of-the-art Survey- Telematica Instituut, Ericsson, CTIT.

ALONSO, G.; CASATI, F.; KUNO, H.; MACHIRAJU, V.; 2004. *Web Services - Concepts, Architectures and Applications*. Spinger-Verlag Berlin Heidelberg.

ANDRADE ALMEIDA, J. P.; FERREIRA PIRES, L.; VAN SINDEREN, M. J.; 2003. Web services and seamless interoperability. In: First European Workshop on Object-Orientation and Web Services (ECOOP 2003) (Darmstadt, Germany).

APACHEFOUNDATION. (2007). "The Apache Software Foundation." Acessado: 28 de Março, 2007, em http://www.apache.org/.

APACHEWEBSERVICES. (2007). "Apache Web Services Project." Acessado: 12 de Março, 2007, em http://ws.apache.org/.

ATHENA-IP. (2004). "ATHENA General description." Acessado: 15 de Março, 2007, em http://www.athena-ip.org/index.php?option=com_docman.

ATHENA-IP. (2005). "What is Architectures and Platforms." Acessado: 15 de Março, 2007, em http://www.athena-ip.org/index.php?option=com_docman.

AUSTIN, D.; BARBIR, A.; FERRIS, C.; GARG, S. (2004). "Web Services Architecture Requirements." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/wsa-reqs.

BALDO, F.; 2006. *Arcabouço Computacional para seleção de organizações em redes colaborativas integrando indicadores de desempenho e análise de riscos* (Qualificação de Doutorado em Engenharia Elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

BALLINGER, K.; EHNEBUSKE, D.; GUDGIN, M.; NOTTINGHAM, M.; et al. (2004). "WS-I Basic Profile Version 1.0." Acessado: 12 de Março, 2007, em http://www.ws-i.org/Profiles/BasicProfile-1.0.html.

BATTLE, S.; BERNSTEIN, A.; BOLEY, H.; GROSOF, B. (2005). "W3C Semantic Web Services Framework (SWSF)." Acessado, em http://www.w3.org/Submission/SWSF/.

BEA. (2006). "BEA WebLogic Server V8.1 Release Notes." Acessado: 28 de Março, 2007, em http://e-docs.bea.com/wls/docs81/pdf/notes.pdf.

BELLWOOD, T.; EHNEBUSKE, D.; HUSBAND, Y. L.; KARP, A.; et al. (2002). "UDDI Version 2.03 Data Structure Reference." Acessado: 12 de Março, 2007, em http://uddi.org/pubs/DataStructure_v2.htm.

BENATALLAH, B.; CASATI, F.; GRIGORI, D.; NEZHAD, H. R. M.; et al.; 2005. Developing Adapters for Web Services Integration. In: *Advanced Information Systems Engineering*. Springer Berlin / Heidelberg.

BERNERS-LEE, T.; FIELDING, R.; IRVINE, U. C.; MASINTER, L. (1998). "Uniform Resource Identifiers (URI): Generic Syntax." RFC2396 Acessado: 10 de Março, 2007, em http://www.ietf.org/rfc/rfc2396.txt.

BERRE, A.-J.; HAHN, A.; AKEHURST, D.; BEZIVIN, J.; et al.; 2004. State-of-the art for Interoperability architecture approaches.- INTEROP-NOE.

BISBAL, J.; LAWLESS, D.; WU, B.; GRIMSON, J.; 1999. Legacy Information System Migration: A Brief Review of Problems, Solutions and Research Issues. *IEEE Software*, v. 16, n. 5.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; et al. (2004). "Web Services Architecture." Acessado, em http://www.w3.org/TR/ws-arch/.

BOSE, R.; SUGUMARAN, V.; 2006. Challenges for Deploying Web Services-Based E-Business Systems in SMEs. *International Journal of E-Business Research*, v. 2, n. 1-18.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, M.; MALER, E.; et al. (2006). "Extensible Markup Language (XML) 1.0 (Fourth Edition)." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/xml/.

BRITTENHAM, P. (2001). "Web Services Development Concepts." IBM Software Group Acessado: 09 de Março, 2007, em http://awwebx04.alphaworks.ibm.com/ettk/demos/wstkdoc/doc/WebServicesDevelopmentConcepts.pdf.

BROWN, A. W.; DELBAERE, M.; EELES, P.; JOHNSTON, S.; et al.; 2005. Realizing service-oriented solutions with the IBM Rational Software Development Platform. *IBM Systems Journal*, v. 44, n. 4.

BRUIJN, J. D.; LAUSEN, H.; KRUMMENACHER, R.; POLLERES, A.; et al. (2005). "Web Service Modeling Language." Acessado: 12 de Março, 2007, em http://www.wsmo.org/TR/d16/d16.1/v0.21/.

BRYAN, D.; DRALUK, V.; EHNEBUSKE, D.; GLOVER, T.; et al. (2002). "UDDI Version 2.04 API Specification." Acessado: 12 de Março, 2007, em http://uddi.org/pubs/ProgrammersAPI_v2.htm.

BRYDEN, A.; 2003. Open And Global Standards For Achieving An Inclusive Information Society. In: SIST Conference (Ljubljana, Slovenia). *Proceedings*.

BUHLER, P. A.; STARR, C. W.; SCHRODER, W. H.; VIDAL, J. M.; 2004. Preparing for Service-Oriented Computing: A Composite Design Pattern for Stubless Web Service Invocation. In: 4th International Conference on Web Engineering (ICWE 2004) (July 26-30, 2004: Munich, Germany). *Proceedings*. Springer.

BURSTEIN, M. H.; MCDERMOTT, D. V.; 2005. Ontology translation for interoperability among semantic Web services. *AI Magazine*, v. 26, n. 1 - Special issue on semantic integration.

BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; et al.; 1996. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. New York: John Wiley & Sons.

BUTEK, R. (2005). "Which style of WSDL should I use?" IBM Whitepaper Acessado: 14 de Março, 2007, em http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl/.

CAMARINHA-MATOS, L. M.; 2003. Infrastructures for virtual organizations - where we are. In: Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference (Lisboa, Portugal). *Proceedings*.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; 1999. The Virtual Enterprise Concept. In: Working Conference On Infrastructure For Virtual Enterprises (PROVE'99) (Porto, Portugal). *Proceedings*.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; 2002. Dynamic virtual organizations, or not so dynamic? In: International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services. *Proceedings*. Kluwer Academic Publishers. p. 111-124.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; 2004. The Emerging Discipline of Collaborative Networks. In: Fifth Working Conference on Virtual Enterprises (PRO-VE'04) (Toulouse, France). *Proceedings*.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; 2005. Collaborative networks: a new scientific discipline. In: *Virtual Organizations - Systems and Practices*. Springer.

CAMARINHA-MATOS, L. M.; AFSARMANESH, H.; 2006. Collaborative networks: Value creation in a knowledge society. In: PROLAMAT 2006, IFIP International Conference On Knowledge Enterprise (Shanghai, China). *Proceedings*. Springer.

CAPE-CLEAR; 2005. Leveraging the Enterprise Service Bus to Lower the Total Cost of Integration. Whitepaper - (May, 2005). Cape Clear Software.

CARR, H. (2006). "Sun's Project Tango." Acessado: 19 de Março, 2007, em http://java.sun.com/developer/technicalArticles/glassfish/ProjectTango/.

CEARLEY, D. W.; FENN, J.; PLUMMER, D. C.; 2005. Gartner's Positions on the Five Hottest IT Topics and Trends in 2005- Gartner Group. Stamford, CT, USA.

CHAPMAN, M.; GOODNER, M.; LUND, B.; MCKEE, B.; et al. (2003). "Supply Chain Management Sample Application Architecture." Acessado: 12 de Março, 2007, em.

CHRISTENSEN, E.; CURBERA, F.; MEREDITH, G.; WEERAWARANA, S. (2001). "Web Services Description Language (WSDL) 1.1." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/wsdl.

CLUNE, J.; DURAND, J.; KLEIJKERS, L.; SANKAR, K.; et al. (2005). "WS-I Analyzer Tool Functional Specification." Acessado: 12 de Março, 2007, em http://www.ws-i.org/Testing/Specs/AnalyzerFunctionalSpecification_Final_1.1.pdf.

COHEN, F. (2002). "Understanding Web service interoperability." IBM developerWorks Acessado: 10 de Março, 2007, em http://www-128.ibm.com/developerworks/webservices/library/ws-inter.html.

COLGRAVE, J.; JANUSZEWSKI, K. (2004). "Using WSDL in a UDDI Registry, Version 2.0.2." Technical Note Acessado: 21 de Março, 2007, em http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; 2005. *Distributed Systems - Concepts and Design. Fourth Edition*. Addison Wesley.

DAVIDSSON, P.; RAMSTEDT, L.; TÖRNQUIST, J.; 2006. Inter-Organization Interoperability in Transport Chains Using Adapters Based on Open Source Freeware. In: *Interoperability of Enterprise Software and Applications* Springer London.

DUDLEY, C.; RIEU, L.; SMITHSON, M.; VERMA, T.; et al.; 2007. *WebSphere Service Registry and Repository Handbook*. First Edition ed. IBM RedBooks: IBM.

ECOLEAD. (2007). "European Collaborative Networked Organizations Leadership Initiative." Acessado: 10 de Março, 2007, em http://www.ecolead.org.

EGYEDI, T. M.; 2006. Standard-Compliant, but Incompatible?! In: I-ESA'06 - Interoperability for Enterprise Software and Applications Conference (Bordeaux, França). *Proceedings*.

ELVESÆTER, B.; HAHN, A.; BERRE, A.; NEPLE, T.; 2005. Towards an Interoperability Framework for Model-Driven Development of Software Systems. In: 1st Int. Conf. on Interoperability of Enterprise Software and Applications (Switzerland). *Proceedings*.

EVJEN, B.; 2002. *XML Web Services with ASP.NET*. Wiley.

FEUERLICHT, G.; 2006. Enterprise SOA: What are the benefits and challenges? In: International Conference Systems Integration 2006 (Prague, Czech Republic). *Proceedings*.

FIELDING, R. T.; 2000. *Architectural Styles and the Design of Network-based Software Architectures* (PhD) - Information and Computer Science, UNIVERSITY OF CALIFORNIA, Irvine, California.

FISCHER, K.; LARRUCEA, X.; BERRE, A.-J.; ELVESÆTER, B.; et al.; 2005a. Model-Driven and Adaptive Interoperability Architectures Deliverable - ATHENA European Integrated Project.

FISCHER, K.; LARRUCEA, X.; FRIESE, T.; 2005b. Specification of a Basic Architecture Reference Model. Deliverable - ATHENA European Integrated Project.

FISHER, D. A.; 2006. An Emergent Perspective on Interoperation in Systems of Systems. Technical Report CMU/SEI-2006-TR-003 - Software Engineering Institute - Carnegie Mellon University.

GANNOD, G. C.; ZHU, H.; MUDIAM, S. V.; 2003. On-the-fly wrapping of web services to support dynamic integration. In: 10th Working Conference on Reverse Engineering, 2003. WCRE 2003 (Victoria, Canada). *Proceedings*. IEEE Computer Society.

GENESIS-PROJECT. (2007). "GENESIS Project." Acessado: 16 de Março, 2007, em http://www.genesis-ist.eu/.

GESSER, C. E.; 2006. *Uma abordagem para a integração dinâmica de serviços Web em portais* (Dissertação de Mestrado em Engenharia Elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

GLATARD, T.; EMSELLEM, D.; MONTAGNAT, J.; 2006. Generic web service wrapper for efficient embedding of legacy codes in service-based workflows. In: Grid-Enabling Legacy Applications and Supporting End Users Workshop (GELA'06) (Paris, França). *Proceedings*.

GOLD-BERNSTEIN, B.; SO, G. (2006). Integration and SOA - Concepts, Technologies, and Best Practices, webMethods, Inc.

GUDGIN, M.; HADLEY, M.; MENDELSOHN, N.; MOREAU, J.-J.; et al. (2003a). "SOAP Version 1.2 Part 1: Messaging Framework." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/soap12/.

GUDGIN, M.; HADLEY, M.; MENDELSOHN, N.; MOREAU, J.-J.; et al. (2003b). "SOAP Version 1.2 Part 2: Adjuncts." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/soap12-part2/.

HOHPE, G.; WOOLF, B.; 2003. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston: Addison-Wesley Professional.

IBM-SOFTWARE-GROUP. (2005). "Standards for success:IBM commitment to SOA and standards for the insurance industry." IBM Whitepaper Acessado: 13 de Março,

2007, em http://www-03.ibm.com/industries/financialservices/doc/content/bin/fss_insurance_standards.pdf.

IDABC1, C. S.; 2004. European Interoperability Framework for pan-European eGovernment Services- Interoperable Delivery of European eGovernment Services.

IDABC2, C. S.; 2003. Linking up Europe: the Importance of Interoperability for eGovernment Services- Interoperable Delivery of European eGovernment Services.

IEEE; 1991. Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries- Institute of Electrical and Electronics Engineers (IEEE).

INDIANA-XSUL. (2007). "WS/XSUL2: Web and XML Services Utility Library (Version 2)." Acessado: 13 de Abril, 2007, em http://www.extreme.indiana.edu/xgws/xsul.

INTEROP-NOE. (2007). "INTEROP-NOE Website." Acessado: 15 de Março, 2007, em http://interop-noe.org/.

KUMAR, K. M. S.; DAS, A. S.; PADMANABHUNI, S.; 2004. WS-I Basic Profile: A Practitioner's View. In: IEEE International Conference on Web Services (ICWS'04) (Los Alamitos, CA, USA). *Proceedings*.

KUSHMERICK, N.; 2000. Wrapper verification. *World Wide Web Journal*, v. 3, n. 2 - Special issue on Web Data Management.

LAKATOS, E. M.; MARCONI, M. D. A.; 1985. *Fundamentos de metodologia científica*. 3a Ed. São Paulo: Editora Atlas.

LEHMANN, M. (2005). "Deploying Large-Scale Interoperable Web Services Infrastructures." Web Services Journal Acessado: 13 de Março, 2007, em http://soa.sys-con.com/read/47665.htm.

LI, M.-S.; CABRAL, R.; DOUMEINGTS, G.; POPPLEWELL, K.; 2006. Enterprise Interoperability - Research Roadmap (Final Version - v4.0) - Information Society Technologies - European Commission.

LILLENG, J.; 2005. Towards semantic interoperability. In: IFIP/ACM SIGAPP INTEROP-ESA Conference (Geneva, Switzerland). *Proceedings*. Springer.

LINDEMANN, J.; DAHLBLOM, O.; SANDBERG, G.; 2006. Using CORBA middleware in finite element software. *Future Generation Computer Systems*, v. 22, n. 1 (Jan/2006).

MACKENZIE, C. M.; LASKEY, K.; MCCABE, F.; BROWN, P. F.; et al.; 2006. Reference Model for Service Oriented Architecture 1.0. Committee Specification 1 - OASIS Group.

MALER, E. (1998). "Guide to the W3C XML Specification DTD, Version 2.1." Acessado: 12 de Março, 2007, em http://www.w3.org/XML/1998/06/xmlspec-report.htm.

MARCZYK, G. R.; DEMATTEO, D.; FESTINGER, D.; 2005. *Essentials of Research Design and Methodology*. Hoboken, New Jersey: John Wiley & Sons, Inc.

MARTIN, D.; BURSTEIN, M.; HOBBS, J.; LASSILA, O.; et al. (2004). "OWL-S: Semantic Markup for Web Services." Acessado: 12 de Março, 2007, em http://www.w3.org/Submission/OWL-S.

MATOS, K. S. L.; VIEIRA, S. L.; 2002. *Pesquisa educacional: o prazer de conhecer*. 2.ed. Fortaleza: Edições Demócrito Rocha.

MCCARTHY, J. (2002). "Reap the benefits of document style Web services." IBM Whitepaper Acessado: 2007, 14 de Março, em http://www-128.ibm.com/developerworks/webservices/library/ws-docstyle.html.

MEL-NIST. (2007). "Manufacturing Engineering Laboratory." NIST (National Institute of Standards and Technology) Acessado: 19 de Março, 2007, em http://www.mel.nist.gov.

MELLOR, S. J.; CLARK, A. N.; FUTAGAMI, T.; 2003. Model-driven development - Guest editor's introduction. *IEEE Software*, v. 20, n. 5 (September/October 2003).

MICROSOFT-UDDI. (2007). "UDDI Shutdown FAQ." Acessado: 21 de Março, 2007, em http://uddi.microsoft.com/about/FAQshutdown.htm.

MONSON-HAEFEL, R.; 2003. *J2EE Web Services*. Addison Wesley.

MORELAND, B.; 2004. UDDI Interoperatility. In: Gartner Application Integration and Web Services Summit (November, 2004: Orlando, Flórida, EUA). *Proceedings*. Charles Schwab and The Hartford.

NO-REST; 2004. Deliverable on Standards Impact Assessment- Networked Organisations - REsearch into STandards & standardisation.

O'BRIEN, L.; BASS, L.; MERSON, P.; 2005. Quality Attributes and Service-Oriented Architectures. Technical Note - Software Engineering Institute, Carnegie Mellon University.

OASIS-UDDI. (2001). "UDDI Executive White Paper." Acessado: 12 de Março, 2007, em http://uddi.org/pubs/UDDI_Executive_White_Paper.pdf.

OASIS. (2007). "Organization for the Advancement of Structured Information Standards." Acessado: 17 de Março, 2007, em http://www.oasis-open.org.

ORT, E.; MANDAVA, R. (2002). "Web Services Developer Pack Part 1: Registration and the JAXR API." The Java Web Services Developer Pack Acessado: 02 de Abril, 2007, em http://java.sun.com/developer/technicalArticles/WebServices/WSPack/.

PAGANELLI, P.; PETERSEN, S. A.; SCHALLOCK, B.; 2005. Business Interoperability Analysis of Networked Organisations. In: e-Chalenges'05 (Ljubliana, Slovenia). *Proceedings*. IOS Press.

PEREPLETCHIKOV, M.; RYAN, C.; TARI, Z.; 2005. The impact of software development strategies on project and structural software attributes in SOA. In: INTEROP Network of Excellence Dissemination Workshop (INTEROP'05), in conjunction with the OTM 2005 (Ayia Napa, Cyprus). *Proceedings*.

PETERSON, D.; BIRON, P. V.; MALHOTRA, A.; SPERBERG-MCQUEEN, M. (2006). "XML Schema 1.1 Part 2: Datatypes." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/xmlschema11-2/.

POHL, K.; 2006. A Translational Solution To Semantic-Interoperability Among Expressive Systems. In: 10th World Multi-Conference on Systemics, Cybernetics and Informatics, WMSCI'06 (Orlando, Florida, EUA). *Proceedings*.

PORT80SOFTWARE. (2006). "Application Servers Survey." Acessado: 28 de Março, 2007, em http://www.port80software.com/surveys/top1000appservers/.

PUSCHMANN, T.; ALT, R.; 2004. Enterprise application integration systems and architecture – the case of the Robert Bosch Group. *The Journal of Enterprise Information Management*, v. 17, n. 2.

RABELO, R. J.; GUSMEROLI, S.; 2007. A Service-Oriented Platform for Collaborative Networked Organizations. In: 8th IFAC Symposium on Cost Oriented Automation Affordable Automation Systems (Cuba). *Proceedings*.

RABELO, R. J.; PEREIRA-KLEN, A. A.; KLEN, E. R.; 2004. Effective management of dynamic and multiple supply chains. *International Journal of Networking and Virtual Organisations*, v. 2, n. 3.

RADETZKI, U.; LESER, U.; CREMERS, A. B.; 2004. IRIS: A Mediator-Based Approach Achieving Interoperability of Web Services in Life Science Applications (invited talk). In: 3rd E-BioSci / ORIEL Workshop (Hinxton, Inglaterra). *Proceedings*.

RATTI, R.; MORES, S.; RODRIGO, M. M.; ARANA, C.; et al.; 2007. ICT-I Reference Framework (version 3). Deliverable D61.1c - ECOLEAD.

RAY, S.; JONES, A.; 2006. Manufacturing interoperability. *Journal of Intelligent Manufacturing*, v. 17, n. 6 (December, 2006).

RODRIGO, M. M.; ARANA, C.; RATTI, R.; RABELO, R.; et al.; 2007. ICT-I Integrated prototype (version 3). Deliverable D64.1c - ECOLEAD.

SADTLER, C.; CLIFFORD, L.; HEYWARD, J.; IWAMOTO, A.; et al.; 2004. *IBM WebSphere Application Server V5.1 System Management and Configuration WebSphere Handbook Series*. IBM RedBooks: IBM.

SÁNCHEZ-NIELSEN, E.; MARTÍN-RUIZ, S.; RODRÍGUEZ-PEDRIANES, J.; 2006. Mobile and Dynamic Web Services. In: Workshop on Emerging Web Services Technology - WEWST'06 (Zurich, Switzerland). *Proceedings*.

SANTOS, I. J. G.; FLUEGGE, M.; TIZZO, N. P.; MADEIRA, E. R. M.; 2006. Challenges and techniques on the road to dynamically compose web services. In: 6th international conference on Web engineering (ICWE'06) (July 11-14, 2006: Palo Alto, California, USA). *Proceedings*.

SANTOS, R. S. D.; 2004. Interoperabilidade do governo brasileiro: A necessidade de construção de serviços latino-americanos. In: IX Congreso Internacional del CLAD sobre la Reforma del Estado y de la Administración Pública (Madrid, Espanha). *Proceedings*.

SARANG, P. G.; BROWNE, C.; AYALA, D.; CHOPRA, V.; 2002. *Open Source Web Services*. Professional Series: Wrox Press, Inc.

SATINE-PROJECT. (2007). "SATINE - Semantic-based interoperability infrastructure for integrating Web service platforms to peer-to-peer networks." Acessado: 16 de Março, 2007, em http://www.srdc.metu.edu.tr/webpage/projects/satine/index.html.

SCHWARTZMAN, S. (1979). "Pesquisa acadêmica, pesquisa básica e pesquisa aplicada em duas comunidades cientificas." não publicado Acessado: 12 de Março, 2007, em http://www.schwartzman.org.br/simon/acad_ap.htm.

SEELY, S.; LAUZON, D.; BRITTENHAM, P.; DURAND, J.; et al. (2005). "WS-I Monitor Tool Functional Specification." Acessado: 12 de Março, 2007, em http://www.ws-i.org/Testing/Specs/MonitorFunctionalSpecification_Final_1.1.pdf.

SEMANTICGOV-PROJECT. (2007). "SemanticGov - Services for Public Administration." Acessado: 16 de Março, 2007, em http://www.semantic-gov.org.

SICOP-RESEARCH. (2007). "Semantic Interoperability Community of Practice." Acessado: 19 de Março, 2007, em http://web-services.gov/.

SIDDHARTHA, P.; SENGUPTA, S.; 2003. Web Services Interoperability: A Practitioner's Experience. In: On the Move to Meaningful Internet Systems (CoopIS/DOA/ODBASE) (Irvine, California, USA). *Proceedings*.

SILVA, E. L. D.; MENEZES, E. M.; 2005. *Metodologia da pesquisa e elaboração de dissertação*. 4a Ed. rev. Florianópolis: UFSC/PPGEP/LED.

SOBERNIG, S.; WILD, F.; 2005. Tutorial on Interoperability in Technology Enhanced Learning. In: ProLearn Summer School (Istanbul, Turquia). *Proceedings*.

SUN. (2006). "Java JAX-RPC Specification." Acessado: 17 de Novembro, 2006, em http://java.sun.com/xml/downloads/jaxrpc.html.

SWITHINBANK, P.; GIGANTE, F.; PROCTOR, H.; RATHORE, M.; et al.; 2005. *WebSphere and .Net Interoperability Using Web Services*. IBM RedBooks: IBM.

TAO, A. T.; YANG, J.; 2006. Develop Service Oriented Finance Business Processes: A Case Study in Capital Market. In: IEEE International Conference on Services Computing (SCC'06) (Rosemont, Illinois, EUA). *Proceedings*.

TERREGOV-PROJECT. (2007). "TERREGOV - Impact of eGovernment on Territorial Government Services." Acessado: 17 de Março, 2007, em http://www.terregov.eupm.net.

THOMPSON, H. S.; SPERBERG-MCQUEEN, M.; GAO, S. S.; MENDELSOHN, N.; et al. (2006). "XML Schema 1.1 Part 1: Structures." Acessado: 12 de Março, 2007, em http://www.w3.org/TR/xmlschema11-1/.

TRAMONTIN JR., R. J.; 2004. *Configuração e Integração de Dados em Plataformas para Empresas Virtuais* (Dissertação de Mestrado em Engenharia Elétrica), Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

TSAGKANI, C.; 2005. Inter-Organizational Collaboration on the Process Layer. In: IFIP/ACM SIGAPP INTEROP-ESA Conference (Geneva, Switzerland). *Proceedings*. Springer Science.

TSALGATIDOU, A.; PILIOURA, T.; 2002. An Overview of Standards and Related Technology in Web Services. *Distributed and Parallel Databases*, v. 12, n. 2-3.

UDDI4J. (2007). "UDDI4J - Java UDDI API." Acessado: 16 de Abril, 2007, em http://uddi4j.sourceforge.net/.

UDDI-BROWSER. (2007). "UDDI Browser GUI." Acessado: 22 de Março, 2007, em http://uddibrowser.org/.

VALLESPIR, B.; BOURRIERES, J.-P.; DUCQ, Y.; ZANNIER, G.; et al.; 2004. INTEROP NoE Project Presentation. Deliverable D15 - INTEROP-NoE.

VOGELS, W.; 2003. Web Services Are Not Distributed Objects. *IEEE Internet Computing*, v. 07, n. 6 (Dez/2003).

W3C. (2007). "World Wide Web Consortium." Acessado: 12 de Março, 2007, em http://www.w3.org/.

WASSON, G.; BEEKWILDER, N.; MORGAN, M.; HUMPHREY, M.; 2004. WS-ResourceFramework on .NET. In: 13th IEEE International Symposium on High performance Distributed Computing (Hawaii, EUA). *Proceedings*.

WEERAWARANA, S.; CURBERA, F.; LEYMANN, F.; STOREY, T.; et al.; 2005. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*. Crawfordsville, Indiana, USA: Prentice Hall PTR.

WERDEN, S.; EVANS, C.; GOODNER, M. (2002). "WS-I Usage Scenarios." Acessado: 12 de Março, 2007, em http://ws-i.org/SampleApplications/SupplyChainManagement/2002-11/UsageScenarios-1.00-CRD-02a.pdf.

WILKES, L. (2005). "The Web Services Protocol Stack." Acessado: 09 de Abril, 2007, em http://roadmap.cbdiforum.com/reports/protocols/index.php.

WS-I. (2002). "Web Service Profiles – An Introduction." Acessado: 12 de Março, 2007, em http://www.ws-i.org/docs/WS-I_Profiles.pdf.

WS-I. (2007). "Web Services Interoperability Organization." Acessado: 12 de Março, 2007, em http://www.ws-i.org.

WSDL4J. (2007). "Web Services Description Language for Java Toolkit." Acessado: 25 de Abril, 2007, em http://sourceforge.net/projects/wsdl4j.

YE, W. (2004). "Web services programming tips and tricks: Improve interoperability between J2EE technology and .NET, Part 1." IBM Whitepaper Acessado: 14 de Março, 2007, em http://www-128.ibm.com/developerworks/webservices/library/ws-tip-j2eenet1/.

YU, J.; ZHOU, G.; 2004. Dynamic Web Service Invocation Based on UDDI. In: IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04) (Beijing, China). *Proceedings*.

ZACHARIAS, A. P. D. L.; OLIVEIRA, M. F. D.; CÔRTES, M. R.; 2007. Consórcio de exportação de software: Pequenas e médias empresas sob a perspectiva de redes sociais. *Revista Gestão Industrial*, v. 3, n. 3.

ZHAO, W.; BRYANT, B. R.; BURT, C. C.; RAJE, R. R.; et al.; 2004. Automated glue/wrapper code generation in integration of distributed and heterogeneous software components. In: Eighth IEEE International Conference on Enterprise Distributed Object Computing (EDOC'04) (Monterey, CA - EUA). *Proceedings*.

ZHAO, Z.; SHENG, Q. Z.; NGU, A. H. H.; 2006. On-Demand Invocation of Web Services. In: Web Technologies, Applications, and Services (WTAS 2006) (July 17-19, 2006: Calgary, Alberta, Canada). *Proceedings*.