

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

**DESENVOLVIMENTO DE UM MECANISMO PARA SELEÇÃO
DINÂMICA DE WEB SERVICES PARA APLICAÇÕES
COMPOSTAS UTILIZANDO MÉTRICAS DE QOS**

Rafael Bosse Brinhosa

Trabalho de conclusão de curso apresentado
como parte dos requisitos para
obtenção do grau de Bacharel
em Sistemas de Informação.

Florianópolis – SC
2006 / 2
Rafael Bosse Brinhosa

DESENVOLVIMENTO DE UM MECANISMO PARA SELEÇÃO DINÂMICA DE WEB SERVICES PARA APLICAÇÕES COMPOSTAS UTILIZANDO MÉTRICAS DE QOS

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de
Bacharel em Sistemas de Informação

Prof. Carlos Becker Westphall
Dr. Orientador

Banca Examinadora

Prof. Carlos Becker Westphall, Dr.

Mário Antônio Ribeiro Dantas, Dr.

Rafael Kruger Tavares, M.Sc.

”Na vida não existe sorte, existe providência divina”.

(Anônimo)

Agradecimentos

Gostaria de agradecer primeiramente a Deus e a todas as pessoas que colaboraram de forma direta e indireta na concepção e desenvolvimento deste trabalho. Especialmente ao meu orientador Carlos Becker Westphall, aos meus colegas do Laboratório de Gerência de Redes da UFSC, aos meus familiares, a minha namorada, aos colegas de turma e de juventude.

Sumário

Agradecimentos	4
Sumário.....	5
Lista de Figuras.....	6
Lista de Reduções	7
Resumo	8
Abstract.....	9
1. Introdução.....	10
1.3 Objetivos.....	10
1.5 Problema.....	11
2. Fundamentação Teórica.....	12
2.1 XML	12
2.2 Web Services	12
2.2.1 SOAP.....	13
2.2.2 UDDI	14
2.2.3 WSDL	14
2.3 QoS	15
2.4 Arquitetura.....	15
2.4.1 Servidor.....	16
2.4.2 Cliente.....	16
2.4.3 Broker	17
3.1 Arquitetura Estendida.....	20
3.2 Mecanismo proposto.....	20
3.2.1 QoS	21
3.2.2 Vantagens	22
4. Implementação.....	22
4.1 Ferramentas utilizadas para a implementação	22
4.1.1 Java EE	22
4.1.2 Apache Axis.....	23
4.1.3 Java Web Services Developer Pack.....	24
4.1.4 jUDDI	24
4.1.5 UDDI4j	24
5. Exemplo prático.....	25
6. Conclusão	28
6.1 Sugestões de trabalhos futuros.....	28
Referências Bibliográficas.....	29
Anexo.....	31
Artigo.....	31
Fontes.....	36

Lista de Figuras

Figura 1 Esquema da arquitetura de Web Services	13
Figura 2 Estrutura de uma mensagem SOAP.	13
Figura 3 Relacionamento da UDDI entre seus elementos businessEntity, businessService, bindingTemplate, e tModel (Komatineni, 2002).	14
Figura 4 Uma visão geral da Arquitetura (Tavares, 2005) ...	Error! Bookmark not defined.
Figura 5 Uma visão geral da Arquitetura com o novo mecanismo incorporado	20
Figura 6 Mecanismo esquematizado	21
Figura 7 As camadas de um Web Service com QoS (Sumra, 2006)	Error! Bookmark not defined.
Figura 9 Esquema de troca de Mensagens SOAP no Servidor utilizando Apache Axis	23
Figura 10 Esquema de troca de Mensagens SOAP no Cliente utilizando Apache Axis	24
Figura 8 Diagrama da arquitetura e Web Services utilizando UDDI privada	Error! Bookmark not defined.
Figura 11 Tela principal da Aplicação NoSPAM.....	25
Figura 12 Tela de Configurações.....	26
Figura 13 Tela de Edição de contas de email	26

Lista de Reduções

UFSC	Universidade Federal de Santa Catarina
UDDI	Universal Description, Discovery and Integration
XML	Extensible Markup language
URI	Uniform Resource Identifier
JDBC	Java Database Connectivity
API	Application Program Interface
JSP	Java Server Pages
HTML	Hyper-text Markup language
UML	Unified Modeling Language
JAVA EE	Java Enterprise Edition
IDE	Integrated Development Environment
QOS	Quality of Service

Resumo

O presente trabalho tem por objetivo o desenvolvimento de um mecanismo para seleção dinâmica de serviços para Aplicações que utilizem Web Services através de métricas de QoS. Para tanto, foi utilizada a arquitetura proposta por Rafael Tavares em 2005 para o fornecimento de QoS para Web Services, já que esta se enquadrava no perfil desejado para a implementação da proposta e já era conhecida.

Esta arquitetura possibilita a mensuração de QoS para Web Services através de sua utilização.

Apresentaremos um mecanismo desenvolvido com o objetivo de aprimorar a Qualidade de Serviço de aplicações que façam o uso de um ou mais Web Services através da seleção dinâmica dos serviços.

Palavras-chave: Web Services, QoS, Mecanismos, Otimização.

Abstract

The present work has for objective the development of a mechanism for dynamic election of services for Applications that use Web Services through QoS metrics. For this propose an architecture was selected to supply QoS for Web Services.

It will be presented a mechanism developed with the objective to improve the Quality of Service of applications that make use of one or more Web Services through the dynamic selection of the services.

Keywords: Web Services, QoS, Composition.

1. Introdução

O tema deste trabalho é o desenvolvimento de um mecanismo para seleção dinâmica de serviços para aplicações compostas por Web Services utilizando métricas de QoS.

Restringiu-se a utilização da arquitetura proposta por Rafael Tavares em 2005 para o fornecimento de QoS para web services, focando-se então na implementação do mecanismo de seleção dinâmica para a execução dos melhores serviços necessários para o aplicativo. Foram alterados apenas certos elementos da arquitetura para uma melhor adequação ao presente trabalho. Também utilizamos métricas pré-definidas de QoS como: disponibilidade, tempo de resposta, preço e reputação, sendo estas escolhidas por serem amplamente aceitas pela comunidade científica.

1.3 Objetivos

Com este projeto pretende-se melhorar aplicações desenvolvidas utilizando-se Web Services, garantindo-se a qualidade de serviço desejada ou acordada em SLA's (Service Level Agreement ou Acordo de nível de serviço).

É fornecido um meio para a escolha e execução dinâmica de Web Services de acordo com sua QoS.

A tecnologia de Web Services com seu crescente uso e sua rápida consolidação como uma tecnologia base para a evolução do desenvolvimento de softwares proporciona um ambiente cujas necessidades emergentes precisam ser solucionadas por padrões que a aprimorem e a aperfeiçoem. A motivação deste trabalho é aprimorar a qualidade de softwares que fazem uso da tecnologia de Web Services, facilitando a escolha dos melhores serviços que devem compor a aplicação de acordo com as métricas desejadas.

1.5 Problema

O desenvolvimento de aplicações baseadas em Web Services ainda é visto com desconfiança pelos fabricantes de software, principalmente por utilizar a internet e ser difícil de mensurar a sua qualidade ou disponibilidade, visamos aumentar a confiabilidade das aplicações desenvolvidas utilizando-se Web Services através do mecanismo proposto.

Ao realizar a escolha de serviços que irão compor uma aplicação, a fábrica de softwares tenta optar pela composição que considera ser a mais apropriada de acordo com seus próprios parâmetros nem sempre confiáveis. Para solucionar este problema este trabalho realiza a proposta de um framework que fornece a possibilidade de escolha dinâmica de serviços através de métricas de QoS para serem executados. Este mecanismo é capaz de, em tempo de execução, optar pelos serviços que irão compor a aplicação objetivando a melhora da qualidade da aplicação final e diminuindo a possibilidade de falhas causadas por indisponibilidade do serviço, lentidão ou custo.

2. Fundamentação Teórica

Existem tecnologias chaves que foram utilizadas para o desenvolvimento deste trabalho, abaixo faremos uma breve explicação sobre cada uma delas.

2.1 XML

XML pode ser considerado como a principal tecnologia que fundamenta a construção de Web Services. Ele provê um formato padrão que pode ser utilizado para trocar informações com praticamente qualquer plataforma. No entanto, esta facilidade tem um custo, o XML não é o mecanismo mais eficiente para se trabalhar ou se transferir dados.

2.2 Web Services

O W3C (Web Services Architecture Working Group) define Web Service como:

Um Web Service é um software identificado por uma URI, no qual as conexões e ligações são capazes de ser definidas, descritas e reveladas como artefatos XML. Um Web Service suporta interações diretas com outros agentes de software através de mensagens baseadas em XML trocadas através de protocolos baseados em Internet.

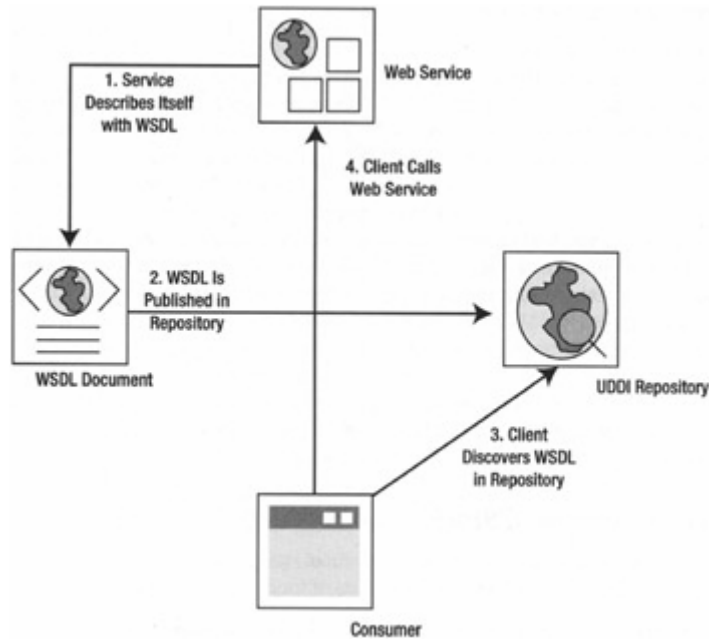


Figura 1 Esquema da arquitetura de Web Services

2.2.1 SOAP

SOAP é um mecanismo para comunicação entre sistemas e aplicações escritos em linguagens iguais ou diferentes através da Internet. SOAP geralmente troca mensagens utilizando o protocolo HTTP: o cliente envia uma requisição SOAP, e recebe tanto um código de sucesso HTTP quanto uma resposta SOAP ou um erro de código HTTP.

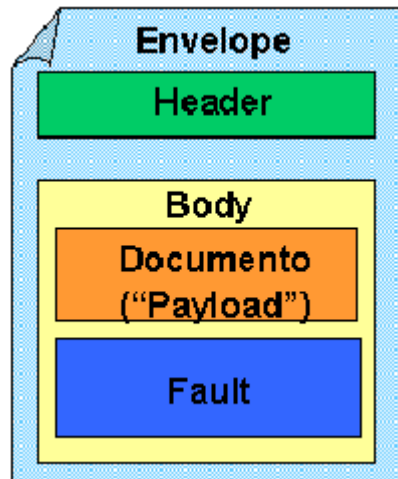


Figura 2 Estrutura de uma mensagem SOAP.

2.2.2 UDDI

A UDDI é um serviço de diretórios para Web Services, que possibilita basicamente três funções: publicação, procura e obrigações.

- ❑ A função de **publicação** refere-se em como o provedor do Web Service se registra e registra seus serviços.
- ❑ A função de **procura** refere-se como a aplicação cliente acha a desce um Web Service ou de um provedor de serviço.
- ❑ Já as **obrigações** referem-se em como a aplicação cliente se conecta e interaje com o Web Service depois de encontrá-lo.

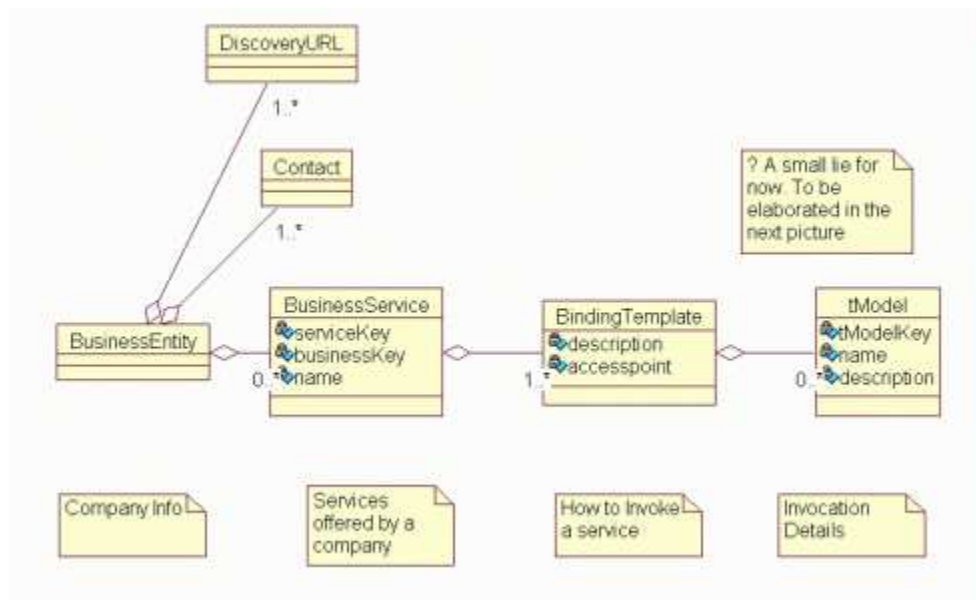


Figura 3 Relacionamento da UDDI entre seus elementos businessEntity, businessService, bindingTemplate, e tModel (Komatineni, 2002).

2.2.3 WSDL

WSDL é um arquivo XML para a descrição de Web Services, definindo as formas de acessá-lo e formatos de mensagens necessários para a comunicação. Nele encontramos tanto os serviços de

rede como um conjunto de pontos de operação com mensagens contendo tanto informações de documentação ou de procedimentos. As operações e mensagens são descritas abstratamente mas apontam para um protocolo de rede concreto e formato de mensagem que definem um ponto de operação específico como funções e métodos oferecidos pelo serviço.

2.3 QoS

É o conceito de aplicar, garantir e quantificar níveis de performance desejados para determinados serviços de acordo com métricas definidas. Refere-se á capacidade de uma rede de prover o melhor serviço.

2.4 Arquitetura

A arquitetura utilizada visa complementar a UDDI, já que esta tem algumas limitações como a falta de qualquer informação sobre a QoS dos Web Services e o fato que o cliente não pode expressar seus desejos de QoS para escolher um determinado serviço. Nela essas limitações são resolvidas ao serem acrescentadas informações sobre a QoS que o provedor do Web Service disponibiliza.

Em nosso trabalho utilizamos essa arquitetura dinamicamente e de forma transparente ao usuário através do mecanismo proposto. A arquitetura adiciona um broker, que serve como módulo intermediário entre o cliente, a UDDI e o provedor do Web Service. O broker é a entidade responsável por armazenar as informações de QoS dos Web Services e fazer a seleção de serviço entre os Web Services encontrados.

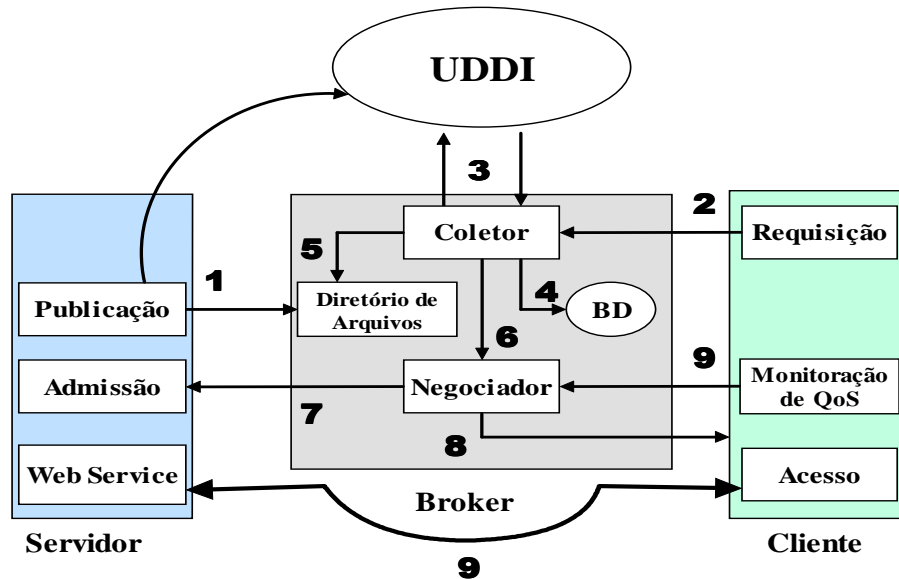


Figura 4 Uma visão geral da Arquitetura (Tavares, 2005)

Abaixo veremos uma breve descrição de cada componente que compõe a arquitetura acima citada.

2.4.1 Servidor

O Servidor é a entidade que possui os serviços disponibilizados para a Web, geralmente chamada de provedor do serviço. Cada Web Service disponibilizado pelo Servidor deve possuir um documento XML que descreve os parâmetros de QoS que ele fornece ao cliente.

2.4.2 Cliente

O cliente foi projetado apenas como sendo uma entidade que deseja acessar algum Web Service (podendo ser uma pessoa através de uma GUI ou um outro Web Service) e que através de uma mensagem XML informe o *broker* sobre o tipo de serviço e os parâmetros de QoS que ele deseja.

2.4.3 Broker

O *broker* foi construído como um Web Service, assim, ele se comunica com o cliente, UDDI e servidor através de mensagens SOAP. O *broker* é responsável por receber e armazenar os arquivos XML que definem a QoS dos Web Services e pela seleção do melhor serviço para o cliente com base nos parâmetros de QoS que ele definiu. Ele possui três módulos: o coletor, o negociador, e o banco de dados. Além desses módulos o *broker* possui um Diretório de Arquivos onde são armazenados os arquivos XML publicados pelo provedor dos Web Services.

2.4.3.1 Coletor

O Coletor é a entidade responsável por armazenar e receber os dados de entrada e saída do *broker*. Ele que recebe os arquivos XML publicados pelo provedor do Web Service e armazena-os no Diretório de Arquivos, recebe as requisições do cliente com os parâmetros do cliente, acessa e recebe a resposta da UDDI e do banco de dados. Pode-se considerar que o Coletor serve como interface entre as outras entidades da arquitetura (cliente, Servidor e UDDI) e o *broker*.

2.4.3.2 Negociador

É a parte principal do broker, responsável pela seleção do melhor serviço e pela verificação dos parâmetros de QoS com o Servidor. Ele irá receber do Coletor as informações dos parâmetros de QoS que o provedor do Web Service oferece, e as informações sobre a requisição do cliente.

2.4.3.3 Banco de dados

O banco de dados armazena as informações de todos os serviços que foram publicados pelo provedor do Web Service.

3. Desenvolvimento

O processo de desenvolvimento de software tem progredido de forma significativa nos últimos anos, graças ao surgimento de tecnologias e conceitos como, por exemplo, os Web Services. Graças a esta tecnologia está sendo cada vez mais fácil desenvolver aplicações complexas em menor tempo e com melhor qualidade.

A utilização de Web Services no desenvolvimento de aplicações trouxe a grande vantagem da interoperabilidade para um mercado cheio de ferramentas e especificações distintas que não eram compatíveis entre si.

Juntamente com a popularização do uso de Web Services surgiram novas necessidades como, por exemplo, a Composição de Web Services e o provimento e mensuração de QoS para Web Services.

Diante de uma quantidade cada vez maior de provedores de Web Services disponíveis no mercado, a escolha do que melhor proverá os serviços necessários para o funcionamento de determinada aplicação tornou-se difícil.

Para isso, esse trabalho estabelece um mecanismo automático para seleção de Web Services com as características solicitadas pela aplicação de forma dinâmica e em tempo de execução. Isto possibilitará que aplicações compostas por vários serviços sejam otimizadas, sempre buscando os melhores serviços para execução.

Vamos supor que determinada Fábrica de Softwares precise desenvolver um portal para um hotel, que tenha como componentes um Web Service que retorne as informações do clima, um outro que retorne o horário oficial do Brasil e um terceiro que busque informações sobre o câmbio de moedas estrangeiras. E que para estes três Web Services existam diferentes provedores que possam retornar as informações desejadas. Sendo assim, é obvio que a Fábrica desejará escolher os melhores

serviços para que sua aplicação funcione com maior disponibilidade, menor custo e fique menos sujeita a falhas.

Atualmente, neste caso, a Fábrica precisaria escolher os serviços de forma empírica e caso este deixasse de funcionar ocorreria uma falha na aplicação.

A proposta deste trabalho é fornecer um meio no qual seja possível escolher os melhores serviços para a aplicação automaticamente. Sendo assim, neste caso, é preciso apenas especificar qual é o serviço desejado e em tempo de execução a aplicação solicitaria o serviço para um broker que retornaria a melhor opção para a aplicação.

Ao longo dos próximos capítulos serão abordados o funcionamento do mecanismo, como foi implementado, suas características; funções e um exemplo prático validando sua utilização.

Este mecanismo tem como foco as seguintes características:

- Definição de métricas de QoS à serem utilizadas;
- Seleção de serviços por nome do provedor, nome do serviço e métodos necessários;
- Busca do melhor serviço de acordo com as características (QoS) desejadas;
- Execução do serviço pela aplicação.

Para dar suporte a este mecanismo foram pesquisadas arquiteturas para fornecimento de QoS para Web Services disponíveis pela comunidade científica e após pesar os prós e contras optamos pela arquitetura definida por Rafael Tavares (2005) em sua tese de mestrado, abaixo serão apresentados os componentes da referida arquitetura.

3.1 Arquitetura Estendida

Para que fosse possível a utilização do mecanismo desenvolvido neste trabalho a arquitetura original foi estendida com a entidade Mecanismo substituindo a entidade Cliente e com os diversos tipos de serviços sendo solicitados pela aplicação ao Mecanismo que possui funções adicionais ao Cliente e que serve como um automatizador das funções do cliente. Na figura 5 é possível observar com maior clareza a Arquitetura Estendida.

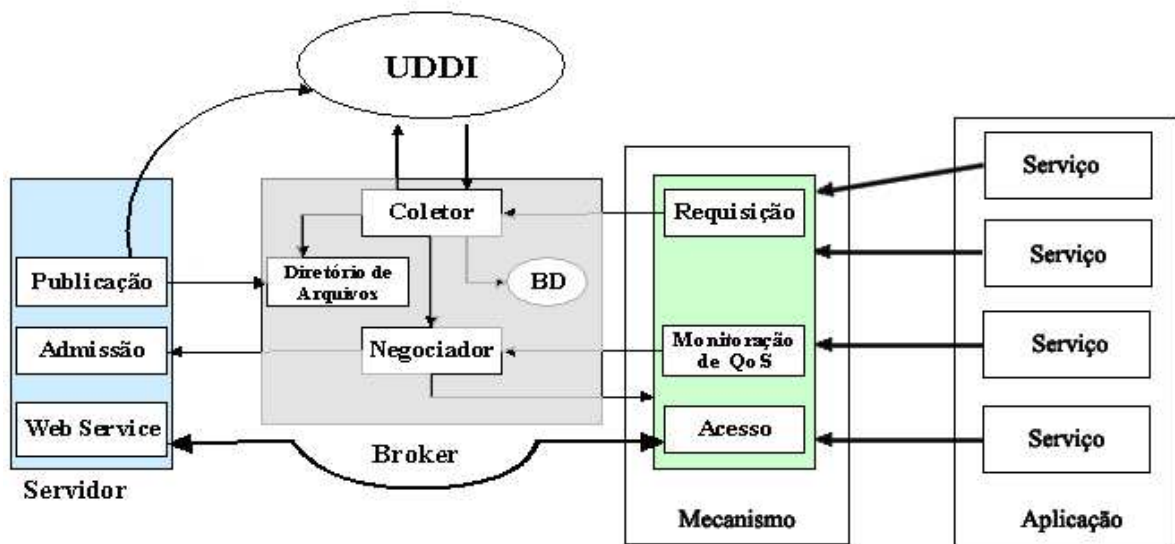


Figura 5 Uma visão geral da Arquitetura com o novo mecanismo incorporado

3.2 Mecanismo proposto

Para o desenvolvimento deste trabalho, pesquisou-se o que há de mais recente no meio acadêmico no que se refere à Composição de Web Services.

Para melhor entendimento o desenvolvimento do mecanismo foi dividido nos seguintes pacotes:

- Application: aplicação cliente que utiliza os serviços;
- Services: serviços disponíveis localmente ou remotamente;

- Wsqos: framework contendo o broker, que faz as consultas ao banco de dados e à UDDI e retorna o serviço escolhido;
- WsqosClient: biblioteca de funções utilizada pelo cliente para solicitar o serviço ao WSQoS.

O mecanismo é um Web Service que quando acessado pela aplicação

através da Biblioteca Java WSQoSClient ou diretamente através de uma chamada ao Web Service retorna o serviço para ser utilizado. Sendo que a WsqosClient pode ou não ser utilizada.

A figura abaixo ilustra esse esquema:

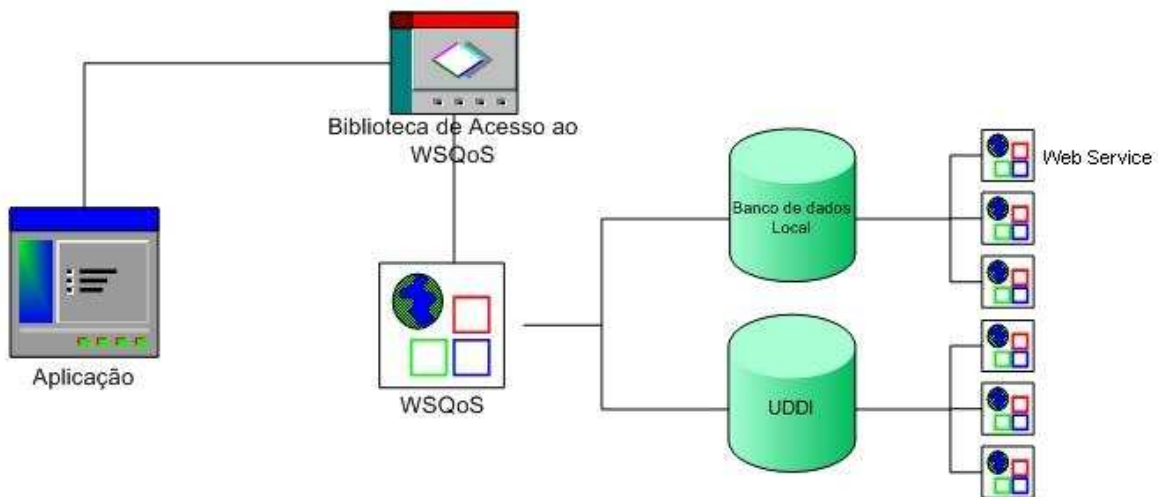


Figura 6 Mecanismo esquematizado

3.2.1 QoS

No mecanismo foram pré-definidas métricas de QoS desejadas para a Aplicação, para isto optamos pelas 4 métricas mais comuns que são:

- Disponibilidade: é o aspecto da qualidade do web service estar disponível ou pronto para uso imediato, representado como uma porcentagem do tempo ligado de um serviço em um período observado.

- ❑ Reputação: é a medida de confiança de um serviço e depende das experiências do usuário final utilizando-o.
- ❑ Tempo de resposta: a quantidade de tempo entre enviar uma requisição e receber a resposta.
- ❑ Preço: o valor monetário do serviço conforme determinado pelo provedor.

3.2.2 Vantagens

A utilização do mecanismo abordado possui algumas vantagens como:

- Capacidade de otimização do desempenho de aplicações ao selecionar-se os melhores serviços para execução.
- Menor tendência à falhas devido a escolha dinâmica dos serviços, caso um serviço desejado não esteja disponível, outra opção é automaticamente escolhida, evitando problemas por falta de disponibilidade.
- Maior garantia do cumprimento de SLA's através da escolha dos serviços que fornecem maior QoS (Qualidade de Serviço).

4. Implementação

4.1 Ferramentas utilizadas para a implementação

Para o desenvolvimento deste trabalho, foram utilizadas as seguintes ferramentas e tecnologias:

4.1.1 Java EE

Segundo a Wikipédia, o Java EE é uma plataforma de programação de computadores que faz parte da plataforma Java. Ela é voltada para aplicações multi-camadas, baseadas em componentes que são executados em um servidor de aplicações. A plataforma Java EE é considerada um padrão de

desenvolvimento já que o fornecedor de software nesta plataforma deve seguir determinadas regras se quiser declarar os seus produtos como compatíveis com Java EE. Ela contém bibliotecas desenvolvidas para o acesso a base de dados, RPC, CORBA, etc.

4.1.2 Apache Axis

Apache Axis é uma implementação Open Source cliente/servidor do SOAP ("Simple Object Access Protocol") padrão do W3C. Foi utilizada para facilitar o desenvolvimento

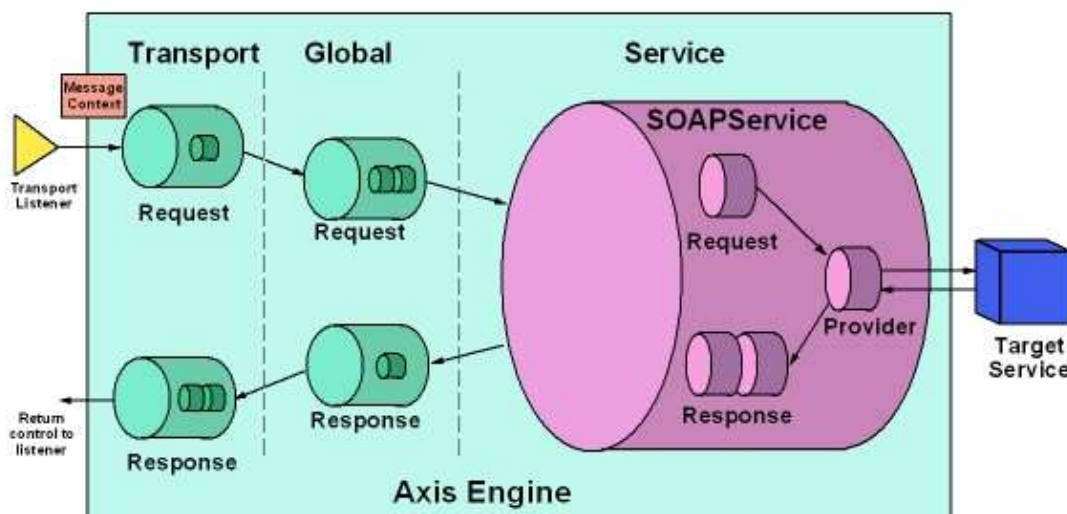


Figura 7 Esquema de troca de Mensagens SOAP no Servidor utilizando Apache Axis

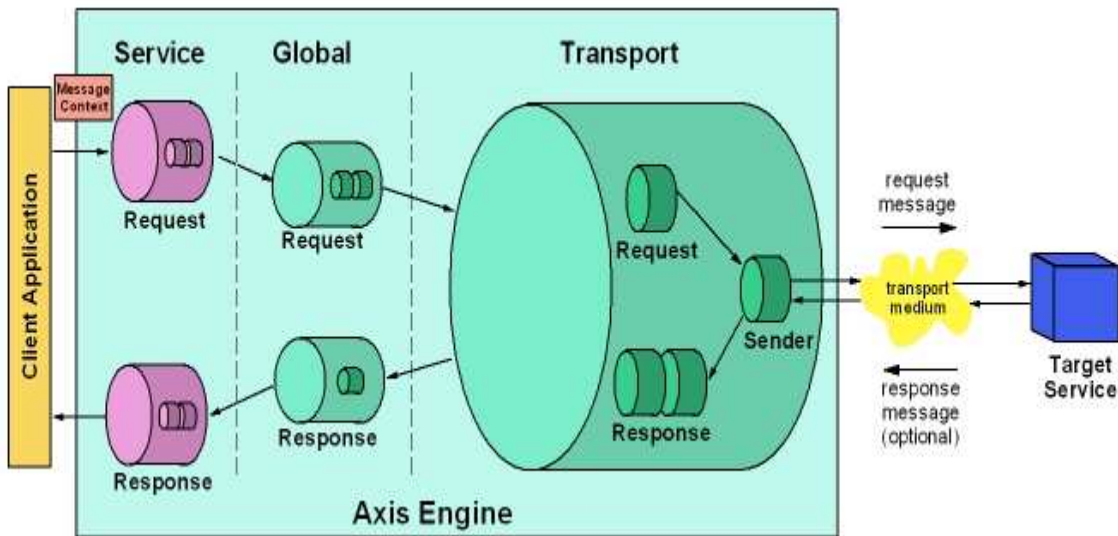


Figura 8 Esquema de troca de Mensagens SOAP no Cliente utilizando Apache Axis

4.1.3 Java Web Services Developer Pack

O Java Web Services Developer Pack (Java WSDP) é um conjunto gratuito de ferramentas que pode ser usado para construir, testar e desenvolver aplicações XML, Web services, e aplicações Web com as últimas tecnologias de Web Service e padrões de implementação.

4.1.4 jUDDI

O jUDDI é uma implementação Java open source da especificação UDDI (Universal Description, Discovery, and Integration) para Web Services. Sendo o jUDDI uma aplicação web Java pode ser utilizado com qualquer servidor de aplicação ou mecanismo servlet que suporte a versão 2.1 ou superior do servlet API.

4.1.5 UDDI4j

UDDI4J é uma biblioteca de classes Java que fornece uma API para a interação com um registro UDDI (Universal Description, Discovery and Integration). O projeto UDDI é um extenso projeto, que foi desenvolvido como uma iniciativa de grandes indústrias para possibilitar que empresas se descubram e define como elas interagem através da internet e compartilham informação através de uma arquitetura de registros global. UDDI é um bloco de construção que possibilitará empresas a rapidamente, facilmente e dinamicamente achar e efetuar transações com outras através de suas aplicações preferidas.

5. Exemplo prático

Para exemplificar a utilização do mecanismo foi desenvolvida uma aplicação baseada em Web Services que têm a função de eliminar SPAMs de uma determinada conta de email, como se pode ver nas telas abaixo.

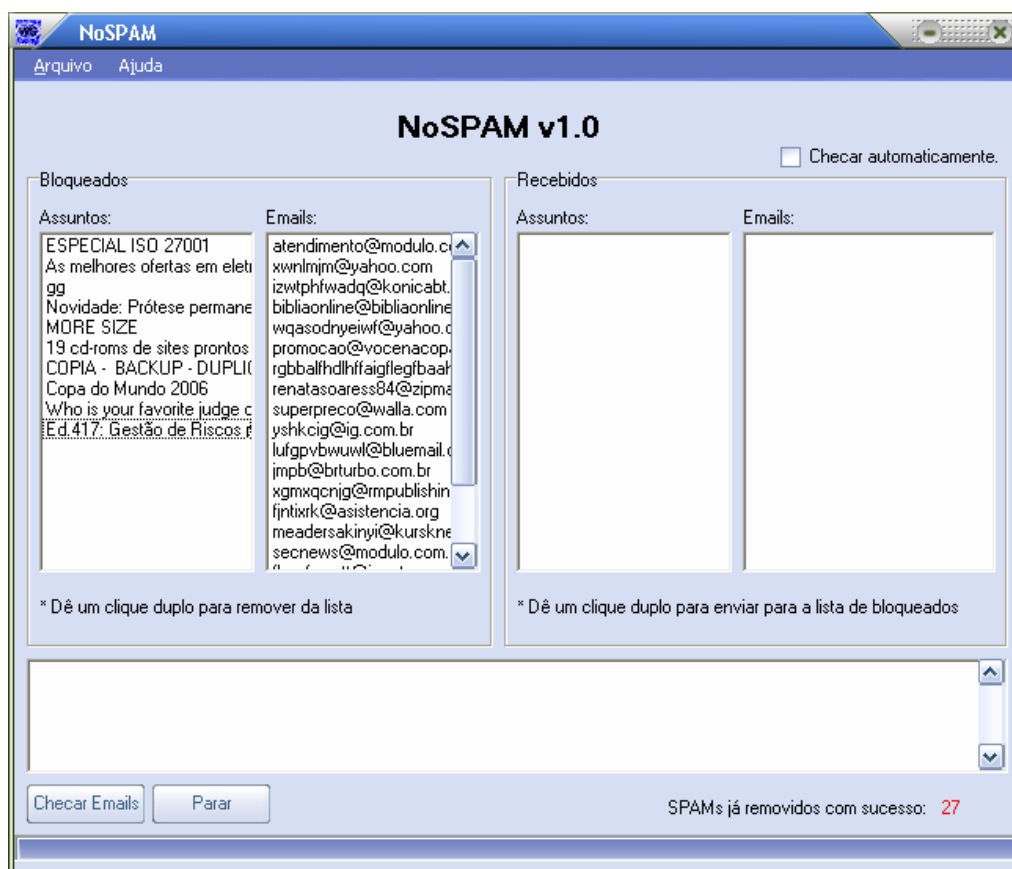


Figura 9 Tela principal da Aplicação NoSPAM

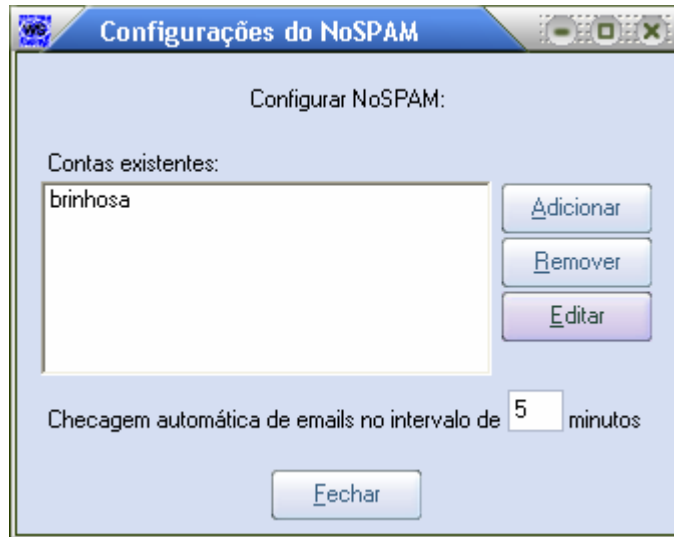


Figura 10 Tela de Configurações

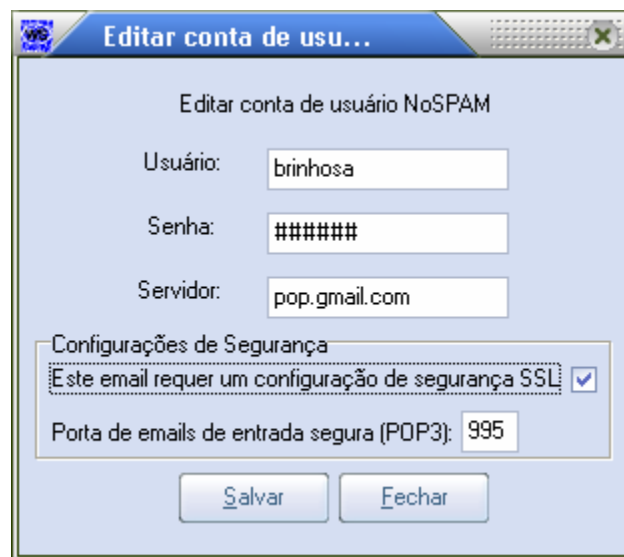


Figura 11 Tela de Edição de contas de email

Este sistema funciona da seguinte forma, ao abrir a aplicação o sistema entra em contato com o broker do mecanismo, solicitando o endereço do melhor de Web Service de detecção de SPAMs disponível de acordo com o QoS.

Então, o usuário adiciona sua conta de email, após isto, manda checar emails, ao checar os emails o sistema recebe as mensagens do servidor POP configurado e entra em contato com o Web Service anteriormente selecionado recebendo a resposta se o email é considerado ou não SPAM, caso seja, apaga o email do servidor e avisa o usuário.

Neste exemplo, pode-se observar a possibilidade de escolha do melhor serviço pelo mecanismo através do Broker de forma transparente ao usuário.

Para o exemplo foram criados dois Web Services de verificação de SPAM, um desenvolvido em JAVA utilizando o framework Apache Axis e outro desenvolvido em PHP utilizando o framework nuSOAP.

6. Conclusão

Com este estudo foi possível observar as qualidades desta tecnologia emergente que são os Web Services, focando-se na solução de um problema que se tornará cada vez mais comum que é a utilização de múltiplos Web Services no desenvolvimento de aplicações.

Pôde-se constatar que a evolução tecnológica no desenvolvimento de aplicações é constante e traz desafios cada vez mais complexos que a comunidade acadêmica visa solucionar através de novas propostas de soluções.

6.1 Sugestões de trabalhos futuros

Como sugestão de trabalhos futuros pode-se mencionar a pesquisa e integração ao mecanismo de algoritmos ótimos para a composição de Web Services como os citados em (Liu, 2006), a implementação da parte de seleção de serviços pela UDDI, desenvolver uma ontologia para lidar com os diferentes nomes de métodos dos serviços e desenvolver um mecanismo que execute o método nos Web Services e selecione a melhor resposta dentro das obtidas.

Referências Bibliográficas

ALMAER, D. Creating Web Services with Apache Axis. Acessado em 11 de agosto de 2006. Disponível em <http://www-scf.usc.edu/~csci571/Special/1578.html>

BRINHOSA, R. Web Services Utilizando JWSDP, Tomcat e Axis. Acessado em 11 de agosto de 2006. Disponível em <http://www.inf.ufsc.br/~brinhosa/wst.ppt>

CHRISTENSEN, Erik; et al. Web Service Description Language (WSDL). [S.I:s.n.], mar. 2001. Disponível em: <<http://www.w3.org/TR/wsdl>>. Acesso em: 5 outubro.2005.

CUNHA D. Web Services, SOAP e Aplicações Web. Acessado em 20 de janeiro de 2006. Disponível em http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html.

DEITEL, H. M.; et al. Web Services: A Technical Introduction. New Jersey, USA:Prentice Hall, 2003.

DONG, W. YU, H.; Optimizing Web Service Composition Based on QoS Negotiation. 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06).

Installing Apache Axis. Acessado em 11 de agosto de 2006. Disponível em: <http://www-scf.usc.edu/~csci571/2006Spring/axisinstall.html>

jUDDI. “jUDDI User’s Guide”. Acessado em 30 de maio de 2006. Disponível em <http://ws.apache.org/juddi/usersguide.html>.

JWSDP. “Java Web Services Development Pack”. Acessado em 30 de maio de 2006. Disponível em <http://java.sun.com/webservices/jwsdp/index.jsp>

KOMATINENI, S. “Understanding UDDI and JAXR”. Acessado em 22 de dezembro de 2006. Disponível em <http://www.onjava.com/pub/a/onjava/2002/02/27/uddi.html>.

LIU, G. GU, N. ZONG, Y. DING, Z. ZHANG, S. “Web Services Automatic Composition Based on QoS”. Proceedings of the 2005 IEEE International Conference on e-Business Engineering (ICEBE'05).

MANSHAN, L. GUO, H.; Goal Description Language for Semantic Web Service Automatic Composition. Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05).

MILANOVIC, N. MALEK M.; Malek Current Solutions for Web Service Composition. IEEE Internet Computing, 2004.

OELLERMANN, L. WILLIAM. M.; et al. XML Technical Primer Architecting Web Services. USA: Apress, 2001.

SUMRA, R. D, A. “Quality of Service for Web Services—Demystification, Limitations, and Best Practices”. Acessado em 20 de dezembro de 2006. Disponível em www.developer.com/services/article.php/2027911.

SUN. The Java Web Services Tutorial. Acessado em 11 de agosto de 2006. Disponível em <http://java.sun.com/Web Services/docs/1.6/tutorial/doc/>

SYSTINET Corp. Developers Corner Tutorial: Your First Web Service In 10 Minutes. Acessado em 10 de agosto de 2006. Disponível em <http://www.systinet.com/download/TutorialOne.pdf>

TAVARES, K. RAFAEL; Arquitetura Para Fornecer Qos em Web Services. UFSC, 2005.

UDDI4j “UDDI for Java Project”. Acessado em 25 de junho de 2006. Disponível em <http://uddi4j.sourceforge.net/index.html>.

VIANNA, L. R. ALMEIDA, M. J. B. TAROUCO, L. M. R. GRANVILLE L. Z.; Investigating Web Services Composition Applied to Network Management. IEEE International Conference on Web Services (ICWS'06).

W3SCHOOLS. SOAP Tutorial. Acessado em 11 de agosto de 2006. Disponível em <http://www.w3schools.com/soap/default.asp>

ZENG L., BENATALLAH B., NGU A. H. H., et al. (2004) “QoS-Aware Middleware for Web Services Composition”. IEEE Transactions on Software Engineering, Vol. 30, No. 5, May 2004.

Anexo

Artigo

DESENVOLVIMENTO DE UM MECANISMO PARA SELEÇÃO DINÂMICA DE WEB SERVICES PARA APLICAÇÕES COMPOSTAS UTILIZANDO MÉTRICAS DE QOS

Rafael Bosse Brinhosa

Sistemas de Informação – Laboratório de Redes e Gerência (LRG) – Universidade Federal de Santa Catarina
Caixa Postal 476 - 88040-900, Florianópolis, SC
{ brinhosa }

Abstract

The present work has for objective the development of a mechanism for dynamic election of services for Applications that use Web Services through QoS metrics. For this propose an architecture was selected to supply QoS for Web Services.

It will be presented a mechanism developed with the objective to improve the Quality of Service of applications that make use of one or more Web Services through the dynamic selection of the services..

1. Introdução

O tema deste trabalho é o desenvolvimento de um mecanismo para seleção dinâmica de serviços para aplicações compostas por Web Services utilizando métricas de QoS.

Restringiu-se a utilização da arquitetura proposta por Rafael Tavares em 2005 para o fornecimento de QoS para web services, focando-se então na implementação do mecanismo de seleção dinâmica para a execução dos melhores serviços necessários para o aplicativo. Foram alterados apenas certos elementos da arquitetura para uma melhor adequação ao presente trabalho. Também utilizamos métricas pré-definidas de QoS como: disponibilidade, tempo de resposta, preço e reputação, sendo estas escolhidas por serem amplamente aceitas pela comunidade científica.

2. Fundamentação Teórica

Existem tecnologias chaves que foram utilizadas para o desenvolvimento deste trabalho, abaixo faremos uma breve explanação sobre cada uma delas.

2.1 XML

XML pode ser considerado como a principal tecnologia que fundamenta a construção de Web Services. Ele provê um formato padrão que pode ser utilizado para trocar informações com praticamente qualquer plataforma. No entanto, esta facilidade tem um custo, o XML não é o mecanismo mais eficiente para se trabalhar ou se transferir dados.

2.2 Web Services

O W3C (Web Services Architecture Working Group) define Web Service como:

Um Web Service é um software identificado por uma URI, no qual as conexões e ligações são capazes de ser definidas, descritas e reveladas como artefatos XML. Um Web Service suporta interações diretas com outros agentes de software através de mensagens baseadas em XML trocadas através de protocolos baseados em Internet.

SOAP é um mecanismo para comunicação entre sistemas e aplicações escritos em linguagens iguais ou diferentes através da Internet. SOAP geralmente troca mensagens utilizando o protocolo HTTP: o cliente envia uma requisição SOAP, e recebe tanto um código de sucesso HTTP quanto uma resposta SOAP ou um erro de código HTTP.

2.2.2 UDDI

A UDDI é um serviço de diretórios para Web Services, que possibilita basicamente três funções: publicação, procura e obrigações.

- A função de publicação refere-se em como o provedor do Web Service se registra e registra seus serviços.
- A função de procura refere-se como a aplicação cliente acha a desce um Web Service ou de um provedor de serviço.
- Já as obrigações referem-se em como a aplicação cliente se conecta e interaje com o Web Service depois de encontrá-lo.

2.2.3 WSDL

WSDL é um arquivo XML para a descrição de Web Services, definindo as formas de acessá-lo e formatos de mensagens necessários para a comunicação. Nele encontramos tanto os serviços de rede como um conjunto de pontos de operação com mensagens contendo tanto informações de documentação ou de procedimentos. As operações e mensagens são descritas abstratamente mas apontam para um protocolo de rede concreto e formato de mensagem que definem um ponto de operação específico como funções e métodos oferecidos pelo serviço.

2.3 QoS

É o conceito de aplicar, garantir e quantificar níveis de performance desejados para determinados serviços de acordo com métricas definidas. Refere-se á capacidade de uma rede de prover o melhor serviço.

2.4 Arquitetura

A arquitetura utilizada visa complementar a UDDI, já que esta tem algumas limitações como a falta de qualquer informação sobre a QoS dos Web Services e o fato que o cliente não pode expressar seus desejos de QoS para escolher um determinado serviço. Nela essas limitações são resolvidas ao serem acrescentadas informações sobre a QoS que o provedor do Web Service disponibiliza.

Em nosso trabalho utilizamos essa arquitetura dinamicamente e de forma transparente ao usuário através do mecanismo proposto. A arquitetura adiciona um broker, que serve como módulo intermediário entre o cliente, a UDDI e o provedor do Web Service. O broker é a entidade responsável por armazenar as informações de QoS dos Web Services e fazer a seleção de serviço entre os Web Services encontrados.

2.4.1 Servidor

O Servidor é a entidade que possui os serviços disponibilizados para a Web, geralmente chamada de provedor do serviço. Cada Web Service disponibilizado pelo Servidor deve possuir um documento XML que descreve os parâmetros de QoS que ele fornece ao cliente.

2.4.2 Cliente

O cliente foi projetado apenas como sendo uma entidade que deseja acessar algum Web Service (podendo ser uma pessoa através de uma GUI ou um outro Web Service) e que através de uma mensagem XML informe o broker sobre o tipo de serviço e os parâmetros de QoS que ele deseja.

2.4.3 Broker

O broker foi construído como um Web Service, assim, ele se comunica com o cliente, UDDI e servidor através de mensagens SOAP. O broker é responsável por receber e armazenar os arquivos XML que definem a QoS dos Web Services e pela seleção do melhor serviço para o cliente com base nos parâmetros de QoS que ele definiu. Ele possui três módulos: o coletor, o negociador, e o banco de dados. Além desses módulos o broker possui um Diretório de Arquivos onde são armazenados os arquivos XML publicados pelo provedor dos Web Services.

2.4.3.1 Coletor

O Coletor é a entidade responsável por armazenar e receber os dados de entrada e saída do broker. Ele que recebe os arquivos XML publicados pelo provedor do Web Service e armazena-os no Diretório de Arquivos,

recebe as requisições do cliente com os parâmetros do cliente, acessa e recebe a resposta da UDDI e do banco de dados. Pode-se considerar que o Coletor serve como interface entre as outras entidades da arquitetura (cliente, Servidor e UDDI) e o broker.

2.4.3.2 Negociador

É a parte principal do broker, responsável pela seleção do melhor serviço e pela verificação dos parâmetros de QoS com o Servidor. Ele irá receber do Coletor as informações dos parâmetros de QoS que o provedor do Web Service oferece, e as informações sobre a requisição do cliente.

2.4.3.3 Banco de dados

O banco de dados armazena as informações de todos os serviços que foram publicados pelo provedor do Web Service.

3. Desenvolvimento

O processo de desenvolvimento de software tem progredido de forma significativa nos últimos anos, graças ao surgimento de tecnologias e conceitos como, por exemplo, os Web Services. Graças a esta tecnologia está sendo cada vez mais fácil desenvolver aplicações complexas em menor tempo e com melhor qualidade.

A utilização de Web Services no desenvolvimento de aplicações trouxe a grande vantagem da interoperabilidade para um mercado cheio de ferramentas e especificações distintas que não eram compatíveis entre si.

Juntamente com a popularização do uso de Web Services surgiram novas necessidades como, por exemplo, a Composição de Web Services e o provimento e mensuração de QoS para Web Services.

Diante de uma quantidade cada vez maior de provedores de Web Services disponíveis no mercado, a escolha do que melhor proverá os serviços necessários para o funcionamento de determinada aplicação tornou-se difícil.

Para isso, esse trabalho estabelece um mecanismo automático para seleção de Web Services com as características solicitadas pela aplicação de forma dinâmica e em tempo de execução. Isto possibilitará que aplicações compostas por vários serviços sejam otimizadas, sempre buscando os melhores serviços para execução.

Vamos supor que determinada Fábrica de Softwares precise desenvolver um portal para um hotel, que tenha como componentes um Web Service que retorne as informações do clima, um outro que retorne o horário oficial do Brasil e um terceiro que busque informações sobre o câmbio de moedas estrangeiras. E que para estes três Web Services existam diferentes provedores que possam retornar as informações desejadas. Sendo assim, é óbvio que a Fábrica desejará escolher os melhores serviços para que sua aplicação funcione com maior disponibilidade, menor custo e fique menos sujeita a falhas.

Atualmente, neste caso, a Fábrica precisaria escolher os serviços de forma empírica e caso este deixasse de funcionar ocorreria uma falha na aplicação.

A proposta deste trabalho é fornecer um meio no qual seja possível escolher os melhores serviços para a aplicação automaticamente. Sendo assim, neste caso, é preciso apenas especificar qual é o serviço desejado e em tempo de execução a aplicação solicitará o serviço para um broker que retornaria a melhor opção para a aplicação.

Ao longo dos próximos capítulos serão abordados o funcionamento do mecanismo, como foi implementado, suas características; funções e um exemplo prático validando sua utilização.

Este mecanismo tem como foco as seguintes características:

- Definição de métricas de QoS à serem utilizadas;
- Seleção de serviços por nome do provedor, nome do serviço e métodos necessários;
- Busca do melhor serviço de acordo com as características (QoS) desejadas;
- Execução do serviço pela aplicação.

Para dar suporte a este mecanismo foram pesquisadas arquiteturas para fornecimento de QoS para Web Services disponíveis pela comunidade científica e após pesar os prós e contras optamos pela arquitetura definida por Rafael Tavares (2005) em sua tese de mestrado, abaixo serão apresentados os componentes da referida arquitetura.

4. Arquitetura estendida

Para que fosse possível a utilização do mecanismo desenvolvido neste trabalho a arquitetura original foi estendida com a entidade Mecanismo substituindo a entidade Cliente e com os diversos tipos de serviços

sendo solicitados pela aplicação ao Mecanismo que possui funções adicionais ao Cliente e que serve como um automatizador das funções do cliente.

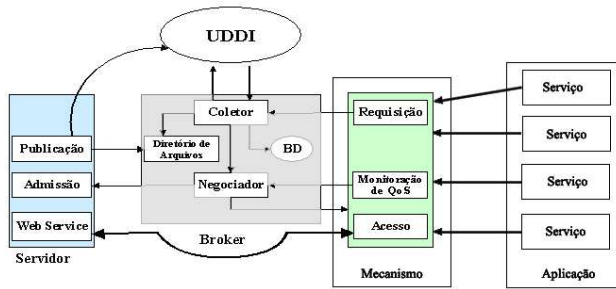


Figura 1 Uma visão geral da Arquitetura com o novo mecanismo incorporado

5. Mecanismo proposto

Para o desenvolvimento deste trabalho, pesquisou-se o que há de mais recente no meio acadêmico no que se refere à Composição de Web Services.

Para melhor entendimento o desenvolvimento do mecanismo foi dividido nos seguintes pacotes:

- Application: aplicação cliente que utiliza os serviços;
- Services: serviços disponíveis localmente ou remotamente;
- Wsqos: framework contendo o broker, que faz as consultas ao banco de dados e à UDDI e retorna o serviço escolhido;
- WsqosClient: biblioteca de funções utilizada pelo cliente para solicitar o serviço ao WSQoS.

O mecanismo é um Web Service que quando acessado pela aplicação através da Biblioteca Java WSQoSClient ou diretamente através de uma chamada ao Web Service retorna o serviço para ser utilizado. Sendo que a WsqosClient pode ou não ser utilizada..

6. Vantagens

A utilização do mecanismo abordado possui algumas vantagens como:

- Capacidade de otimização do desempenho de aplicações ao selecionar-se os melhores serviços para execução.
- Menor tendência à falhas devido a escolha dinâmica dos serviços, caso um serviço desejado não esteja disponível, outra opção é automaticamente escolhida, evitando problemas por falta de disponibilidade.
- Maior garantia do cumprimento de SLA's através da escolha dos serviços que fornecem maior QoS (Qualidade de Serviço).

7. Exemplo prático

Para exemplificar a utilização do mecanismo foi desenvolvida uma aplicação baseada em Web Services que têm a função de eliminar SPAMs de uma determinada conta de email, como se pode ver nas telas abaixo.

Este sistema funciona da seguinte forma, ao abrir a aplicação o sistema entra em contato com o broker do mecanismo, solicitando o endereço do melhor de Web Service de detecção de SPAMs disponível de acordo com o QoS.

Então, o usuário adiciona sua conta de email, após isto, manda checar emails, ao checar os emails o sistema recebe as mensagens do servidor POP configurado e entra em contato com o Web Service anteriormente selecionado recebendo a resposta se o email é considerado ou não SPAM, caso seja, apaga o email do servidor e avisa o usuário.

Neste exemplo, pode-se observar a possibilidade de escolha do melhor serviço pelo mecanismo através do Broker de forma transparente ao usuário.

Para o exemplo foram criados dois Web Services de verificação de SPAM, um desenvolvido em JAVA utilizando o framework Apache Axis e outro desenvolvido em PHP utilizando o framework nuSOAP.

8. Conclusão

Com este estudo foi possível observar as qualidades desta tecnologia emergente que são os Web Services, focando-se na solução de um problema que se tornará cada vez mais comum que é a utilização de múltiplos Web Services no desenvolvimento de aplicações.

Pôde-se constatar que a evolução tecnológica no desenvolvimento de aplicações é constante e traz desafios cada vez mais complexos que a comunidade acadêmica visa solucionar através de novas propostas de soluções.

9. Sugestões e trabalhos futuros

Como sugestão de trabalhos futuros pode-se mencionar a pesquisa e integração ao mecanismo de algoritmos ótimos para a composição de Web Services como os citados em (Liu, 2006), a implementação da parte de seleção de serviços pela UDDI, desenvolver uma ontologia para lidar com os diferentes nomes de métodos dos serviços e desenvolver um mecanismo que execute o método nos Web Services e selecione a melhor resposta dentro das obtidas.

12. Referências

- ALMAER, D. Creating Web Services with Apache Axis. Acessado em 11 de agosto de 2006. Disponível em <http://www-scf.usc.edu/~csci571/Special/1578.html>
- BRINHOSA, R. Web Services Utilizando JWSDP, Tomcat e Axis. Acessado em 11 de agosto de 2006. Disponível em <http://www.inf.ufsc.br/~brinhosa/wst.ppt>
- CHRISTENSEN, Erik; et al. Web Service Description Language (WSDL). [S.I:s.n.], mar. 2001. Disponível em: <<http://www.w3.org/TR/wsdl>>. Acesso em: 5 outubro.2005.
- CUNHA D. Web Services, SOAP e Aplicações Web. Acessado em 20 de janeiro de 2006. Disponível em http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html.
- DEITEL, H. M.; et al. Web Services: A Technical Introduction. New Jersey, USA:Prentice Hall, 2003.
- DONG, W. YU, H.; Optimizing Web Service Composition Based on QoS Negotiation. 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06).
- Installing Apache Axis. Acessado em 11 de agosto de 2006. Disponível em: <http://www-scf.usc.edu/~csci571/2006Spring/axisinstall.html>
- jUDDI. "jUDDI User's Guide". Acessado em 30 de maio de 2006. Disponível em <http://ws.apache.org/juddi/usersguide.html>.
- JWSDP. "Java Web Services Development Pack". Acessado em 30 de maio de 2006. Disponível em <http://java.sun.com/webservices/jwsdp/index.jsp>
- KOMATINENI, S. "Understanding UDDI and JAXR". Acessado em 22 de dezembro de 2006. Disponível em <http://www.onjava.com/pub/a/onjava/2002/02/27/uddi.html>.
- LIU, G. GU, N. ZONG, Y. DING, Z. ZHANG, S. "Web Services Automatic Composition Based on QoS". Proceedings of the 2005 IEEE International Conference on e-Business Engineering (ICEBE'05).

Fontes

```
/*  
  
* Broker.java  
  
*  
  
* Created on 19 de Dezembro de 2006, 19:29  
  
*  
  
* To change this template, choose Tools | Template Manager  
  
* and open the template in the editor.  
  
*/
```

```
package wsqos;  
  
import java.util.*;  
  
/**  
  
*  
  
* @author Rafael  
  
*/  
  
public class Broker {  
  
    Db db;  
  
    Uddi uddi;  
  
    /** Creates a new instance of Broker */  
  
    public Broker() {  
  
        db = new Db();  
  
        uddi = new Uddi();  
  
    }  
  
}
```

```
}
```

```
public void publish (String xml, String nome, String description, String url){
```

```
//Serviço ou fornecedor se publica no DB
```

```
    db.publish(xml,nome,description,url);
```

```
}
```

```
public String consult(String description){
```

```
//Consulta no DB e depois na UDDI, recebe e compara
```

```
    List resultList = new Vector();
```

```
    String result = "";
```

```
    if (db.exists(description)){
```

```
        resultList = db.consultList(description);
```

```
        result = compare(resultList);
```

```
    }
```

```
    else{
```

```
        result = uddi.Consult(description);
```

```
    }
```

```
    return result;
```

```
}
```

```
public String compare(List wsList){
```

```
    System.out.println(wsList.toString());
```

```
    List v = new Vector();
```

```
v=wsList;

String[] res;

String[] resi;

    for (int i = 1; i < v.size(); i++) {

        res = v.get(i).toString().split(",");

        int value = Integer.parseInt(res[0]), j = i - 1;

        while ( j >= 0 && value < v [j] ) {

            resi = v.get(j).toString().split(",");

            v.set(j + 1,v.get(j));

            j--;

        }

        v[j + 1] = value;

    }

    return "http://127.0.0.1:8080/axis/services/WSSPAMServer";

}

}

/*

* db.java

*

* Created on 19 de Dezembro de 2006, 19:44
```

*

* To change this template, choose Tools | Template Manager

* and open the template in the editor.

*/

```
package wsqos;
```

```
import java.util.*;
```

```
/**
```

```
*
```

```
* @author Rafael
```

```
*/
```

```
public class Db {
```

```
    List services = new Vector();
```

```
    /** Creates a new instance of db */
```

```
    public Db() {
```

```
        services.add("1"+"","+"WS SPAM Server 1"+"","+"WS SPAM Server  
description"+"","+"http://127.0.0.1:8080/axis/services/WSSPAMServer");
```

```
        services.add("2"+"","+"WS SPAM Server 2"+"","+"WS SPAM Server  
description"+"","+"http://127.0.0.1:8080/axis/services/WSSPAMServerb");
```

```
    }
```

```
    public boolean exists(String description){
```

```
        boolean result = false;
```

```
        String[] res;
```

```
String desc;

for (int i = 0; i < services.size(); i++) {

    System.out.println(services.get(i).toString());

    res = services.get(i).toString().split(",");

    desc=res[2];

    System.out.println("desc: "+desc);

    if (description.equals(desc)){

        result=true;

    }

}

return result;

}
```

```
public String consult(String description){

    String result = "";

    String[] res;

    if (exists(description)){

        for (int i = 0; i < services.size(); i++) {

            System.out.println(services.get(i).toString());

            res = services.get(i).toString().split(",");

            System.out.println("res3: "+res[3].toString());

            if (description.equals(res[3])){

                result=res[3];

            }

        }

    }

}
```



```
    }  
    }  
}  
return result;  
}  
  
public List consultList(String description){  
    List result = new Vector();  
    String[] res;  
    if (exists(description)){  
        for (int i = 0; i < services.size(); i++) {  
            System.out.println(services.get(i).toString());  
            res = services.get(i).toString().split(",");  
            if (description.equals(res[3])){  
                result.add(services.get(i));  
            }  
        }  
    }  
    return result;  
}  
  
public void publish (String xml, String nome, String description, String url){  
    services.add(xml+","+nome+","+description+","+url);
```

```
}
```

```
}
```

```
/*
```

```
* Uddi.java
```

```
*
```

```
* Created on 19 de Dezembro de 2006, 19:45
```

```
*
```

```
* To change this template, choose Tools | Template Manager
```

```
* and open the template in the editor.
```

```
*/
```

```
package wsqos;
```

```
/**
```

```
*
```

```
* @author Rafael
```

```
*/
```

```
public class Uddi {
```

```
    /** Creates a new instance of Uddi */
```

```
    public Uddi() {
```

```
    }
```

```
public String Consult(String description){  
    return "URL";  
}  
}
```

```
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class WSSPAMServer {  
  
    public boolean ISSPAM(String email) {  
        Scanner scanner;  
        File file;  
        String t = new String();  
        boolean b = false;  
        try {  
            file = new File("c:\\TEXT.txt");  
            scanner = new Scanner(file);  
            t = scanner.findInLine(email);  
            if (t != null){  
                b = (t.equals(email));  
            }  
        }  
    }  
}
```

```
    }  
  
    System.out.println(b);  
  
    scanner.close();  
  
} catch (FileNotFoundException e) {  
    e.printStackTrace();  
  
}  
  
return b;  
  
}  
  
public void ADD(String email){  
    if (ISSPAM(email) == false) {  
        try {  
            FileOutputStream fis = new FileOutputStream("c:\\TEXT.txt", true);  
            fis.write(("\\n"+" "+email).getBytes());  
            fis.close();  
        } catch (IOException e) {  
            System.err.println(e);  
        }  
    }  
  
}  
  
}  
  
/*  
  
* SampleApplication.java
```

*

* Created on 11 de Dezembro de 2006, 21:11

*/

/**

* Exemplo de Aplicação que chama o mecanismo de busca de Serviços para receber o Serviço desejado.

* @author Rafael

*/

package application;

import org.apache.axis.client.Service;

import org.apache.axis.client.Call;

import javax.swing.*;

public class SampleApplication extends javax.swing.JFrame {

/** Creates new form SampleApplication */

public SampleApplication() {

 initComponents();

}

/** This method is called from within the constructor to

* initialize the form.

* WARNING: Do NOT modify this code. The content of this method is

* always regenerated by the Form Editor.

*/

// <editor-fold defaultstate="collapsed" desc=" Generated Code ">

```
private void initComponents() {  
  
    jTextField1 = new javax.swing.JTextField();  
  
    jLabel1 = new javax.swing.JLabel();  
  
    jButton1 = new javax.swing.JButton();  
  
    menuBar = new javax.swing.JMenuBar();  
  
    fileMenu = new javax.swing.JMenu();  
  
    openMenuItem = new javax.swing.JMenuItem();  
  
    saveMenuItem = new javax.swing.JMenuItem();  
  
    saveAsMenuItem = new javax.swing.JMenuItem();  
  
    exitMenuItem = new javax.swing.JMenuItem();  
  
    editMenu = new javax.swing.JMenu();  
  
    cutMenuItem = new javax.swing.JMenuItem();  
  
    copyMenuItem = new javax.swing.JMenuItem();  
  
    pasteMenuItem = new javax.swing.JMenuItem();  
  
    deleteMenuItem = new javax.swing.JMenuItem();  
  
    helpMenu = new javax.swing.JMenu();  
  
    contentsMenuItem = new javax.swing.JMenuItem();  
  
    aboutMenuItem = new javax.swing.JMenuItem();  
  
  
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
  
    jTextField1.setText("brinhosa@rafael.com.br");
```

```
jLabel1.setText("Email:");
```

```
jButton1.setText("Checar");
```

```
jButton1.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        jButton1ActionPerformed(evt);  
    }  
});
```

```
fileMenu.setText("File");
```

```
openMenuItem.setText("Open");
```

```
fileMenu.add(openMenuItem);
```

```
saveMenuItem.setText("Save");
```

```
fileMenu.add(saveMenuItem);
```

```
saveAsMenuItem.setText("Save As ...");
```

```
fileMenu.add(saveAsMenuItem);
```

```
exitMenuItem.setText("Exit");
```

```
exitMenuItem.addActionListener(new java.awt.event.ActionListener() {  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
        exitMenuItemActionPerformed(evt);  
    }  
});
```

```
    }  
});  
  
fileMenu.add(exitMenuItem);  
  
menuBar.add(fileMenu);  
  
editMenu.setText("Edit");  
cutMenuItem.setText("Cut");  
editMenu.add(cutMenuItem);  
  
copyMenuItem.setText("Copy");  
editMenu.add(copyMenuItem);  
  
pasteMenuItem.setText("Paste");  
editMenu.add(pasteMenuItem);  
  
deleteMenuItem.setText("Delete");  
editMenu.add(deleteMenuItem);  
  
menuBar.add(editMenu);  
  
helpMenu.setText("Help");  
contentsMenuItem.setText("Contents");
```



```
helpMenu.add(contentsMenuItem);

aboutMenuItem.setText("About");
helpMenu.add(aboutMenuItem);

menuBar.add(helpMenu);

setJMenuBar(menuBar);

org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(layout.createSequentialGroup()
            .add(31, 31, 31)
            .add(jLabel1)
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
            .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(jButton1)
                .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 89,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
            .addContainerGap(47, Short.MAX_VALUE))
        );
```

```
layout.setVerticalGroup(  
    layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)  
    .add(layout.createSequentialGroup()  
        .add(32, 32, 32)  
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)  
            .add(jTextField1, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,  
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,  
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)  
            .add(jLabel1))  
        .add(19, 19, 19)  
        .add(jButton1)  
        .addContainerGap(26, Short.MAX_VALUE))  
    );  
pack();  
} // </editor-fold>
```

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
// TODO add your handling code here:  
    String description = "WS SPAM Server description";  
    WSRequester wsr = new WSRequester();  
    String ret= wsr.Request(description);  
    //String ret= "http://127.0.0.1:8080/axis/services/WSSPAMServer";  
    String par= jTextField1.getText();  
    if (((wsExecuteAction(ret,par))).equals("true")){
```

```
JOptionPane.showMessageDialog(this,"Email SPAMMER!","Aviso",
JOptionPane.ERROR_MESSAGE);
}
else{
    JOptionPane.showMessageDialog(this,"Email não considerado
SPAMMER!","Aviso", JOptionPane.INFORMATION_MESSAGE);
}
}
```

```
private String wsExecuteAction(String url, String par){
    // Endereço, local onde encontra-se o Web Service
    //String local = "http://localhost:8080/axis/Servico.jws";

    String local = url;
    String result = "none";

    // Criando e configurando o serviço
    try {
        System.out.println("Executando Try");
        Call call = (Call) new Service().createCall();

        // Configurando o endereço.
        call.setTargetEndpointAddress(local);

        // Marcando o método a ser chamado.
        call.setOperationName("ISSPAM");
```

```
// Parâmetros da função soma.

Object[] param = new Object[]{new String(par)};

// Retorno da Função

boolean ret = (Boolean)call.invoke(param);

// Imprime o resultado: ret = 2 + 4.

System.out.println("Results: " + ret);

result=Boolean.toString(ret);

}

catch (Exception e){

    e.printStackTrace();

}

return result;

}

private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {

    System.exit(0);

}

/**

 * @param args the command line arguments

 */

public static void main(String args[]) {
```

```
java.awt.EventQueue.invokeLater(new Runnable() {  
    public void run() {  
        new SampleApplication().setVisible(true);  
    }  
});  
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JMenuItem aboutMenuItem;  
private javax.swing.JMenuItem contentsMenuItem;  
private javax.swing.JMenuItem copyMenuItem;  
private javax.swing.JMenuItem cutMenuItem;  
private javax.swing.JMenuItem deleteMenuItem;  
private javax.swing.JMenu editMenu;  
private javax.swing.JMenuItem exitMenuItem;  
private javax.swing.JMenu fileMenu;  
private javax.swing.JMenu helpMenu;  
private javax.swing.JButton jButton1;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JTextField jTextField1;  
private javax.swing.JMenuBar menuBar;  
private javax.swing.JMenuItem openMenuItem;  
private javax.swing.JMenuItem pasteMenuItem;  
private javax.swing.JMenuItem saveAsMenuItem;
```

```
private javax.swing.JMenuItem saveMenuItem;

// End of variables declaration

}

/*
 * WSRequester.java
 *
 * Created on 19 de Dezembro de 2006, 20:54
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package application;

import org.apache.axis.client.Service;
import org.apache.axis.client.Call;

/**
 * Classe que consulta o broker, retornando o endereço do serviço para o cliente.
 * @author Rafael
 */
public class WSRequester {

    wsqos.Broker broker;
```

```
/** Creates a new instance of WSRequester */  
  
public WSRequester() {  
  
}  
  
public String Request(String description){  
  
    // Endereço, local onde encontra-se o Web Service  
  
    //String local = "http://localhost:8080/axis/Servico.jws";  
  
    String local = "http://localhost:8080/axis/Broker";  
  
    String result = "none";  
  
    // Criando e configurando o serviço  
  
    broker = new wsqos.Broker();  
  
    result = broker.consult(description);  
  
    /* try {  
  
        Call call = (Call) new Service().createCall();  
  
        // Configurando o endereço.  
  
        call.setTargetEndpointAddress(local);  
  
        // Marcando o método a ser chamado.  
  
        call.setOperationName("consult");  
  
        // Parâmetros da função soma.  
  
        Object[] param = new Object[]{new String()};  
  
        // Retorno da Função  
  
        String ret = (String)call.invoke(param);  
  
        // Imprime o resultado: ret = 2 + 4.
```

```
System.out.println("Results: " + ret);

result=ret;

}

catch (Exception e){

}*/

return result;

}

}
```

ANEXO A – CONFIGURANDO O AMBIENTE

Primeiro, instale o JDK 1.5.01 ou superior.

Descompacte o Tomcat para Java WSDP2 no lugar de sua preferência.

Depois, instale o Java WSDP 2.03 ou superior. Na tela “Select a Web Container”, clique no botão “Browse” e selecione a pasta tomcat50-jwsdp que foi

descompactada, assim você estará utilizando como container para o JWSDP o

Tomcat 5.0, na tela “Create a Tomcat user” digite o usuário e senha que desejar.

Descompacte o Apache Axis4 também no local de sua preferência. Copie a pasta axis-1_3\webapps\axis descompactada para dentro do diretório \webapps do

Tomcat.

Para verificar se a instalação ocorreu corretamente, inicialize o servidor Tomcat, clicando no Menu Iniciar -> Programas -> Sun Microsystems -> Java(TM) Web

Services Developer Pack 2.0 -> Start Tomcat e entre no seguinte endereço através

de um browser: <http://127.0.0.1:8080/axis/>

Se a instalação foi bem sucedida aparecerá uma página com o título Apache-AXIS e com a frase “Hello! Welcome to Apache-Axis”.

1 <http://java.sun.com/downloads/index.html>

2 http://java.sun.com/Web Services/containers/tomcat_for_JWSDP_1_5.html

3 <http://java.sun.com/Web Services/downloads/Web Servicespack.html>

4 http://www.apache.org/dyn/closer.cgi/ws/axis/1_3

18

ANEXO B – CÓDIGO FONTE SERVIDOR

```
import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Scanner;

public class WSSPAMServer {

public boolean ISSPAM(String email) {

Scanner scanner;

File file;

String t = new String();

boolean b = false;

try {

file = new File("c:\\TEXT.txt");

scanner = new Scanner(file);
```

```
t = scanner.findInLine(email);

if (t != null){

b = (t.equals(email));

}

System.out.println(b);

scanner.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return b;

}

public static void main(String args[]){

WSSPAMServer s = new WSSPAMServer();

s.ADD("s@s.s");

s.ADD("s2@s.s");

s.ISSPAM("s@s.s");

}

public void ADD(String email){

if (ISSPAM(email) == false) {

try {

FileOutputStream fis = new FileOutputStream("c:\\TEXT.txt", true);

fis.write(("\\n"+ " "+email).getBytes());

fis.close();

} catch (IOException e) {
```

```
System.err.println(e);
```

```
}
```

```
}
```

```
}
```

```
}
```

19

ANEXO C – WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<wsdl:definitions targetNamespace="urn:ws.WSSPAMServer"
```

```
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

```
xmlns:impl="urn:ws.WSSPAMServer"
```

```
xmlns:intf="urn:ws.WSSPAMServer"
```

```
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
```

```
xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<!--WSDL created by Apache Axis version: 1.3
```

```
Built on Oct 05, 2005 (05:23:37 EDT)-->
```

```
<wsdl:message name="ADDResponse">
```

```
</wsdl:message>
```

```
<wsdl:message name="ADDRequest">
```

```
<wsdl:part name="in0" type="soapenc:string"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="ISSPAMResponse">
```

```
<wsdl:part name="ISSPAMReturn" type="xsd:boolean"/>
</wsdl:message>
<wsdl:message name="ISSPAMRequest">
  <wsdl:part name="in0" type="soapenc:string"/>
</wsdl:message>
<wsdl:portType name="WSSPAMServer">
  <wsdl:operation name="ISSPAM" parameterOrder="in0">
    <wsdl:input message="impl:ISSPAMRequest" name="ISSPAMRequest"/>
    <wsdl:output message="impl:ISSPAMResponse" name="ISSPAMResponse"/>
  </wsdl:operation>
  <wsdl:operation name="ADD" parameterOrder="in0">
    <wsdl:input message="impl:ADDRequest" name="ADDRequest"/>
    <wsdl:output message="impl:ADDResponse" name="ADDResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="WSSPAMServerSoapBinding" type="impl:WSSPAMServer">
  20
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="ISSPAM">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="ISSPAMRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:ws.WSSPAMServer" use="encoded"/>
    </wsdl:input>
```

```
<wsdl:output name="ISSPAMResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ws.WSSPAMServer" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="ADD">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="ADDRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ws.WSSPAMServer" use="encoded"/>
</wsdl:input>
<wsdl:output name="ADDResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ws.WSSPAMServer" use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="WSSPAMServerService">
<wsdl:port binding="impl:WSSPAMServerSoapBinding" name="WSSPAMServer">
<wsdlsoap:address location="http://127.0.0.1:8080/axis/WSSPAMServer"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

ANEXO D – AUTOMATIZADOR

echo off

echo -----

echo Processando Web Service

echo -----

c:

cd C:\Tomcat\tomcat50-jwsdp\webapps\axis

set AXIS1=C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\axis-

ant.jar;C:\Tomcat\tomcat50-

jwsdp\webapps\axis\WEB-INF\lib\axis-schema.jar;C:\Tomcat\tomcat50-

jwsdp\webapps\axis\WEBINF\

lib\axis.jar;C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\commons-discovery-

0.2.jar;C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\commons-logging-

1.0.4.jar;C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-

INF\lib\jaxrpc.jar;C:\Tomcat\tomcat50-

jwsdp\webapps\axis\WEB-INF\lib\log4j-1.2.8.jar;C:\Tomcat\tomcat50-

jwsdp\webapps\axis\WEBINF\

lib\saaj.jar;C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\wsdl4j-1.5.1.jar

javac ws\WSSPAMServer.java

java -cp %axis1%;C:\Tomcat\tomcat50-jwsdp\webapps\axis

org.apache.axis.wsdl.Java2WSDL -o

ws.wsdl -l "http://127.0.0.1:8080/axis/WSSPAMServer" -n "urn:ws.WSSPAMServer"

ws.WSSPAMServer

```
java -cp %AXIS1% org.apache.axis.wsdl.WSDL2Java -o C:\Tomcat\tomcat50-  
jwsdp\webapps\axis\ -d  
Session -s -p ws ws.wsdl  
  
echo -----  
  
echo Insira os metodos necessarios  
  
echo -----  
  
pause  
  
javac -classpath %axis1% ws\*.java  
  
jar cvf wsspam.jar ws\*.class  
  
copy C:\Tomcat\tomcat50-jwsdp\webapps\axis\wsspam.jar C:\Tomcat\tomcat50-  
jwsdp\webapps\axis\WEB-INF\lib  
  
java -cp %axis1%;C:\Tomcat\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\  
org.apache.axis.client.AdminClient -p 8080 ./ws/deploy.wsdd  
  
echo -----  
  
echo Processo realizado com sucesso  
  
echo -----  
  
echo on  
  
22
```

ANEXO E – CÓDIGO FONTE GERADO AUTOMATICAMENTE

WSSPAMServer_PortType.java

```
package ws;  
  
public interface WSSPAMServer_PortType extends java.rmi.Remote {  
  
public boolean ISSPAM(java.lang.String in0) throws java.rmi.RemoteException;  
  
public void ADD(java.lang.String in0) throws java.rmi.RemoteException;
```

```
}
```

```
WSSPAMServerService.java
```

```
package ws;
```

```
public interface WSSPAMServerService extends javax.xml.rpc.Service {
```

```
public java.lang.String getWSSPAMServerAddress();
```

```
public ws.WSSPAMServer_PortType getWSSPAMServer() throws
```

```
javax.xml.rpc.ServiceException;
```

```
public ws.WSSPAMServer_PortType getWSSPAMServer(java.net.URL portAddress)
```

```
throws
```

```
javax.xml.rpc.ServiceException;
```

```
}
```

```
WSSPAMServerServiceLocator.java
```

```
package ws;
```

```
public class WSSPAMServerServiceLocator extends org.apache.axis.client.Service
```

```
implements
```

```
ws.WSSPAMServerService {
```

```
public WSSPAMServerServiceLocator() {
```

```
}
```

```
public WSSPAMServerServiceLocator(org.apache.axis.EngineConfiguration config) {
```

```
super(config);
```

```
}
```

```
public WSSPAMServerServiceLocator(java.lang.String wsdlLoc,
```

```
javax.xml.namespace.QName
```

```
sName) throws javax.xml.rpc.ServiceException {
```



```
super(wsdlLoc, sName);

}

// Use to get a proxy class for WSSPAMServer

private java.lang.String WSSPAMServer_address =

"http://127.0.0.1:8080/axis/WSSPAMServer";

public java.lang.String getWSSPAMServerAddress() {

return WSSPAMServer_address;

}

// The WSDD service name defaults to the port name.

23

private java.lang.String WSSPAMServerWSDDServiceName = "WSSPAMServer";

public java.lang.String getWSSPAMServerWSDDServiceName() {

return WSSPAMServerWSDDServiceName;

}

public void setWSSPAMServerWSDDServiceName(java.lang.String name) {

WSSPAMServerWSDDServiceName = name;

}

public ws.WSSPAMServer_PortType getWSSPAMServer() throws

javax.xml.rpc.ServiceException {

java.net.URL endpoint;

try {

endpoint = new java.net.URL(WSSPAMServer_address);

}

catch (java.net.MalformedURLException e) {
```

```
throw new javax.xml.rpc.ServiceException(e);
}
return getWSSPAMServer(endpoint);
}
public ws.WSSPAMServer_PortType getWSSPAMServer(java.net.URL portAddress)
throws
javax.xml.rpc.ServiceException {
try {
ws.WSSPAMServerSoapBindingStub _stub = new
ws.WSSPAMServerSoapBindingStub(portAddress, this);
_stub.setPortName(getWSSPAMServerWSDDServiceName());
return _stub;
}
catch (org.apache.axis.AxisFault e) {
return null;
}
}
public void setWSSPAMServerEndpointAddress(java.lang.String address) {
WSSPAMServer_address = address;
}
/**
* For the given interface, get the stub implementation.
* If this service has no port for the given interface,
* then ServiceException is thrown.
```

```
*/  
  
public java.rmi.Remote getPort(Class serviceEndpointInterface) throws  
javax.xml.rpc.ServiceException {  
    try {  
        if (ws.WSSPAMServer_PortType.class.isAssignableFrom(serviceEndpointInterface)) {  
            ws.WSSPAMServerSoapBindingStub _stub = new  
            ws.WSSPAMServerSoapBindingStub(new java.net.URL(WSSPAMServer_address), this);  
            _stub.setPortName(getWSSPAMServerWSDDServiceName());  
            return _stub;  
        }  
    }  
    catch (java.lang.Throwable t) {  
        throw new javax.xml.rpc.ServiceException(t);  
    }  
    throw new javax.xml.rpc.ServiceException("There is no stub implementation for the  
interface: " +  
(serviceEndpointInterface == null ? "null" : serviceEndpointInterface.getName()));  
}  
24  
/**  
* For the given interface, get the stub implementation.  
* If this service has no port for the given interface,  
* then ServiceException is thrown.  
*/
```

```
public java.rmi.Remote getPort(javax.xml.namespace.QName portName, Class
serviceEndpointInterface) throws javax.xml.rpc.ServiceException {
if (portName == null) {
return getPort(serviceEndpointInterface);
}
java.lang.String inputPortName = portName.getLocalPart();
if ("WSSPAMServer".equals(inputPortName)) {
return getWSSPAMServer();
}
else {
java.rmi.Remote _stub = getPort(serviceEndpointInterface);
((org.apache.axis.client.Stub) _stub).setPortName(portName);
return _stub;
}
}
public javax.xml.namespace.QName getServiceName() {
return new javax.xml.namespace.QName("urn:ws.WSSPAMServer",
"WSSPAMServerService");
}
private java.util.HashSet ports = null;
public java.util.Iterator getPorts() {
if (ports == null) {
ports = new java.util.HashSet();
```

```
ports.add(new javax.xml.namespace.QName("urn:ws.WSSPAMServer",
"WSSPAMServer"));
}
return ports.iterator();
}
/**
 * Set the endpoint address for the specified port name.
 */
public void setEndpointAddress(java.lang.String portName, java.lang.String address)
throws
javax.xml.rpc.ServiceException {
if ("WSSPAMServer".equals(portName)) {
setWSSPAMServerEndpointAddress(address);
}
else
{ // Unknown Port Name
throw new javax.xml.rpc.ServiceException(" Cannot set Endpoint Address for Unknown
Port" +
portName);
}
}
/**
 * Set the endpoint address for the specified port name.
 */
```

```
public void setEndpointAddress(javax.xml.namespace.QName portName, java.lang.String  
address)
```

```
throws javax.xml.rpc.ServiceException {
```

```
setEndpointAddress(portName.getLocalPart(), address);
```

```
}
```

```
}
```

25

WSSPAMServerSoapBindingImpl.java

```
package ws;
```

```
import ws.WSSPAMServer;
```

```
public class WSSPAMServerSoapBindingImpl implements
```

```
ws.WSSPAMServer_PortType{
```

```
WSSPAMServer wse = new WSSPAMServer();
```

```
public boolean ISSPAM(java.lang.String in0) throws java.rmi.RemoteException {
```

```
return wse.ISSPAM(in0);
```

```
}
```

```
public void ADD(java.lang.String in0) throws java.rmi.RemoteException {
```

```
wse.ADD(in0);
```

```
}
```

```
}
```

WSSPAMServerSoapBindingStub.java

```
package ws;
```

```
public class WSSPAMServerSoapBindingStub extends org.apache.axis.client.Stub
```

```
implements
```

```
ws.WSSPAMServer_PortType {  
  
private java.util.Vector cachedSerClasses = new java.util.Vector();  
private java.util.Vector cachedSerQNames = new java.util.Vector();  
private java.util.Vector cachedSerFactories = new java.util.Vector();  
private java.util.Vector cachedDeserFactories = new java.util.Vector();  
  
static org.apache.axis.description.OperationDesc [] _operations;  
  
static {  
  
_operations = new org.apache.axis.description.OperationDesc[2];  
_initOperationDesc1();  
  
}  
  
private static void _initOperationDesc1(){  
  
org.apache.axis.description.OperationDesc oper;  
  
org.apache.axis.description.ParameterDesc param;  
  
oper = new org.apache.axis.description.OperationDesc();  
  
oper.setName("ISSPAM");  
  
param = new org.apache.axis.description.ParameterDesc(new  
javax.xml.namespace.QName("",  
"in0"), org.apache.axis.description.ParameterDesc.IN, new  
javax.xml.namespace.QName("http://schemas.xmlsoap.org/soap/encoding/", "string"),  
java.lang.String.class, false, false);  
  
oper.addParameter(param);  
  
oper.setReturnType(new  
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema",  
"boolean"));  
  
}
```

```
oper.setReturnClass(boolean.class);

oper.setReturnQName(new javax.xml.namespace.QName("", "ISSPAMReturn"));

oper.setStyle(org.apache.axis.constants.Style.RPC);

oper.setUse(org.apache.axis.constants.Use.ENCODED);

26

_operations[0] = oper;

oper = new org.apache.axis.description.OperationDesc();

oper.setName("ADD");

param = new org.apache.axis.description.ParameterDesc(new

javax.xml.namespace.QName("",

"in0"), org.apache.axis.description.ParameterDesc.IN, new

javax.xml.namespace.QName("http://schemas.xmlsoap.org/soap/encoding/", "string"),

java.lang.String.class, false, false);

oper.addParameter(param);

oper.setReturnType(org.apache.axis.encoding.XMLType.AXIS_VOID);

oper.setStyle(org.apache.axis.constants.Style.RPC);

oper.setUse(org.apache.axis.constants.Use.ENCODED);

_operations[1] = oper;

}

public WSSPAMServerSoapBindingStub() throws org.apache.axis.AxisFault {

this(null);

}

public WSSPAMServerSoapBindingStub(java.net.URL endpointURL,

javax.xml.rpc.Service service)
```



```
throws org.apache.axis.AxisFault {

this(service);

super.cachedEndpoint = endpointURL;

}

public WSSPAMServerSoapBindingStub(javax.xml.rpc.Service service) throws

org.apache.axis.AxisFault {

if (service == null) {

super.service = new org.apache.axis.client.Service();

} else {

super.service = service;

}

((org.apache.axis.client.Service)super.service).setTypeMappingVersion("1.2");

}

protected org.apache.axis.client.Call createCall() throws java.rmi.RemoteException {

try {

org.apache.axis.client.Call _call = super._createCall();

if (super.maintainSessionSet) {

_call.setMaintainSession(super.maintainSession);

}

if (super.cachedUsername != null) {

_call.setUsername(super.cachedUsername);

}

if (super.cachedPassword != null) {

_call.setPassword(super.cachedPassword);

}
```

```
}  
  
if (super.cachedEndpoint != null) {  
    _call.setTargetEndpointAddress(super.cachedEndpoint);  
}  
  
if (super.cachedTimeout != null) {  
    _call.setTimeout(super.cachedTimeout);  
}  
  
if (super.cachedPortName != null) {  
    _call.setPortName(super.cachedPortName);  
}  
  
java.util.Enumeration keys = super.cachedProperties.keys();  
while (keys.hasMoreElements()) {  
    java.lang.String key = (java.lang.String) keys.nextElement();  
    27  
    _call.setProperty(key, super.cachedProperties.get(key));  
}  
  
return _call;  
}  
  
catch (java.lang.Throwable _t) {  
    throw new org.apache.axis.AxisFault("Failure trying to get the Call object", _t);  
}  
}  
  
public boolean ISSPAM(java.lang.String in0) throws java.rmi.RemoteException {  
    if (super.cachedEndpoint == null) {
```

```
throw new org.apache.axis.NoEndPointException();

}

org.apache.axis.client.Call _call = createCall();

_call.setOperation(_operations[0]);

_call.setUseSOAPAction(true);

_call.setSOAPActionURI("");

_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);

_call.setOperationName(new javax.xml.namespace.QName("urn:ws.WSSPAMServer",
"ISSPAM"));

setRequestHeaders(_call);

setAttachments(_call);

try { java.lang.Object _resp = _call.invoke(new java.lang.Object[] {in0});

if (_resp instanceof java.rmi.RemoteException) {

throw (java.rmi.RemoteException)_resp;

}

else {

extractAttachments(_call);

try {

return ((java.lang.Boolean) _resp).booleanValue();

} catch (java.lang.Exception _exception) {

return ((java.lang.Boolean) org.apache.axis.utils.JavaUtils.convert(_resp,
boolean.class)).booleanValue();

}

}

}
```

```
} catch (org.apache.axis.AxisFault axisFaultException) {  
  
throw axisFaultException;  
  
}  
  
}  
  
public void ADD(java.lang.String in0) throws java.rmi.RemoteException {  
  
if (super.cachedEndpoint == null) {  
  
throw new org.apache.axis.NoEndPointException();  
  
}  
  
org.apache.axis.client.Call _call = createCall();  
  
_call.setOperation(_operations[1]);  
  
_call.setUseSOAPAction(true);  
  
_call.setSOAPActionURI("");  
  
_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_CONSTANTS);  
  
_call.setOperationName(new javax.xml.namespace.QName("urn:ws.WSSPAMServer",  
"ADD"));  
  
setRequestHeaders(_call);  
  
setAttachments(_call);  
  
try { java.lang.Object _resp = _call.invoke(new java.lang.Object[] {in0});  
  
if (_resp instanceof java.rmi.RemoteException) {  
  
throw (java.rmi.RemoteException)_resp;  
  
}  
  
28  
  
extractAttachments(_call);  
  
} catch (org.apache.axis.AxisFault axisFaultException) {
```

```
throw axisFaultException;
```

```
}
```

```
}
```

```
}
```

29

ANEXO F – MENSAGENS SOAP

ISSPAM Request

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="
```

```
http://schemas.xmlsoap.org/soap/envelope/">
```

```
<soap:Body>
```

```
<ISSPAM xmlns="urn:ws.WSSPAMServer">
```

```
<op1>email@email.com.br</op1>
```

```
</ISSPAM>
```

```
</soap:Body>
```

```
</soap:Envelope>
```

ISSPAM Response

```
<?xml version="1.0"?>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="
```

```
http://www.w3.org/2001/XMLSchema-instance"><soapenv:Body><ns1:ISSPAMResponse
```

```
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:ns1="urn:ws.WSSPAMServer"><ISSPAMReturn
```

```
xsi:type="xsd:boolean">>false</ISSPAMReturn></ns1:ISSPAMResponse></soapenv:Body
```

```
></soapenv:
```

```
Envelope>
```

ADD Request

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="
```

```
http://schemas.xmlsoap.org/soap/envelope/">
```

```
<soap:Body>
```

```
<aDD xmlns="urn:ws.WSSPAMServer">
```

```
<op1>email@email.com.br</op1>
```

```
</ADD>
```

```
</soap:Body>
```

```
</soap:Envelope>
```

ADD Response

```
<?xml version="1.0"?>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="
```

```
http://www.w3.org/2001/XMLSchema-instance"><soapenv:Body><ns1:ADDResponse
```

```
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:ns1="urn:ws.WSSPAMServer"/></soapenv:Body></soapenv:Envelope>
```

Broker WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:wsqos.Broker"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:wsqos.Broker" xmlns:intf="urn:wsqos.Broker"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->

  <wsdl:message name="publishRequest">

    <wsdl:part name="in0" type="soapenc:string"/>

    <wsdl:part name="in1" type="soapenc:string"/>

    <wsdl:part name="in2" type="soapenc:string"/>

    <wsdl:part name="in3" type="soapenc:string"/>

  </wsdl:message>

  <wsdl:message name="consultRequest">

    <wsdl:part name="in0" type="soapenc:string"/>

  </wsdl:message>

  <wsdl:message name="compareResponse">

    <wsdl:part name="compareReturn" type="soapenc:string"/>

  </wsdl:message>

  <wsdl:message name="consultResponse">

    <wsdl:part name="consultReturn" type="soapenc:string"/>

  </wsdl:message>

  <wsdl:message name="compareRequest">

    <wsdl:part name="in0" type="soapenc:string"/>

  </wsdl:message>

  <wsdl:message name="publishResponse">

  </wsdl:message>

  <wsdl:portType name="Broker">

    <wsdl:operation name="consult" parameterOrder="in0">
```

```
<wsdl:input message="impl:consultRequest"
name="consultRequest"/>

<wsdl:output message="impl:consultResponse"
name="consultResponse"/>

</wsdl:operation>

<wsdl:operation name="compare" parameterOrder="in0">

<wsdl:input message="impl:compareRequest"
name="compareRequest"/>

<wsdl:output message="impl:compareResponse"
name="compareResponse"/>

</wsdl:operation>

<wsdl:operation name="publish" parameterOrder="in0 in1 in2 in3">

<wsdl:input message="impl:publishRequest"
name="publishRequest"/>

<wsdl:output message="impl:publishResponse"
name="publishResponse"/>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="BrokerSoapBinding" type="impl:Broker">

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="consult">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="consultRequest">

<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded"/>

</wsdl:input>

<wsdl:output name="consultResponse">

<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded"/>

</wsdl:output>

</wsdl:operation>
```



```
<wsdl:operation name="compare">
  <wsdlsoap:operation soapAction="" />
  <wsdl:input name="compareRequest">
    <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded" />
  </wsdl:input>
  <wsdl:output name="compareResponse">
    <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded" />
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="publish">
  <wsdlsoap:operation soapAction="" />
  <wsdl:input name="publishRequest">
    <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded" />
  </wsdl:input>
  <wsdl:output name="publishResponse">
    <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded" />
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="BrokerService">
  <wsdl:port binding="impl:BrokerSoapBinding" name="Broker">
    <wsdlsoap:address
location="http://localhost:8080/axis/services/Broker" />
  </wsdl:port>
</wsdl:service>
```

</wsdl:definitions>

Automatizador de Deploy - WSQoS_passo-a-passo.bat

echo on

echo -----

echo Processando Web Service

echo -----

D:

cd D:\tomcat50-jwsdp\webapps\axis

set AXIS1=D:\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\axis-ant.jar;D:\tomcat50-

jwsdp\webapps\axis\WEB-INF\lib\axis-schema.jar;D:\tomcat50-

jwsdp\webapps\axis\WEB-INF\lib\axis.jar;D:\tomcat50-jwsdp\webapps\axis\WEB-

INF\lib\commons-discovery-0.2.jar;D:\tomcat50-jwsdp\webapps\axis\WEB-

INF\lib\commons-logging-1.0.4.jar;D:\tomcat50-jwsdp\webapps\axis\WEB-

INF\lib\jaxrpc.jar;D:\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\log4j-

1.2.8.jar;D:\tomcat50-jwsdp\webapps\axis\WEB-INF\lib\saaj.jar;D:\tomcat50-

jwsdp\webapps\axis\WEB-INF\lib\wsdl4j-1.5.1.jar

javac wsqos*.java

"C:\Program Files\Java\jre1.5.0_07\bin\java" -cp %axis1%;D:\tomcat50-

jwsdp\webapps\axis org.apache.axis.wsdl.Java2WSDL -o wsqos.wsdl -l

"http://127.0.0.1:8080/axis/Broker" -n "urn:wsqos.Broker" wsqos.Broker

```
"C:\Program Files\Java\jre1.5.0_07\bin\java" -cp %AXIS1%  
  
org.apache.axis.wsdl.WSDL2Java -o D:\tomcat50-jwsdp\webapps\axis\ -d Session -s -p  
  
wsqos wsqos.wsdl  
  
echo -----  
echo    Insira os metodos necessarios  
echo -----  
  
pause  
  
javac -classpath %axis1% wsqos\*.java  
  
jar cvf wsqos.jar wsqos\*.class  
  
copy D:\tomcat50-jwsdp\webapps\axis\wsqos.jar D:\tomcat50-jwsdp\webapps\axis\WEB-  
INF\lib  
  
"C:\Program Files\Java\jre1.5.0_07\bin\java" -cp %axis1%;D:\tomcat50-  
jwsdp\webapps\axis\WEB-INF\lib\ org.apache.axis.client.AdminClient -p 8080  
  
./wsqos/deploy.wsdd  
  
echo -----  
echo    Processo realizado com sucesso  
echo -----  
  
echo on
```

<http://127.0.0.1:8080/axis/services/Broker?wsdl>

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:wsqos.Broker"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn:wsqos.Broker"
xmlns:intf="urn:wsqos.Broker"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->

<wsdl:message name="publishRequest">

  <wsdl:part name="in0" type="soapenc:string"/>

  <wsdl:part name="in1" type="soapenc:string"/>

  <wsdl:part name="in2" type="soapenc:string"/>

  <wsdl:part name="in3" type="soapenc:string"/>

</wsdl:message>
```

```
<wsdl:message name="consultRequest">
```

```
  <wsdl:part name="in0" type="soapenc:string"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="compareResponse">
```

```
  <wsdl:part name="compareReturn" type="soapenc:string"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="consultResponse">
```

```
  <wsdl:part name="consultReturn" type="soapenc:string"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="compareRequest">
```

```
  <wsdl:part name="in0" type="soapenc:string"/>
```

```
</wsdl:message>
```

```
<wsdl:message name="publishResponse">
```

```
</wsdl:message>
```

```
<wsdl:portType name="Broker">
```

```
<wsdl:operation name="consult" parameterOrder="in0">
```

```
<wsdl:input message="impl:consultRequest" name="consultRequest"/>
```

```
<wsdl:output message="impl:consultResponse" name="consultResponse"/>
```

```
</wsdl:operation>
```

```
<wsdl:operation name="compare" parameterOrder="in0">
```

```
<wsdl:input message="impl:compareRequest" name="compareRequest"/>
```

```
<wsdl:output message="impl:compareResponse" name="compareResponse"/>
```

```
</wsdl:operation>
```

```
<wsdl:operation name="publish" parameterOrder="in0 in1 in2 in3">
```

```
<wsdl:input message="impl:publishRequest" name="publishRequest"/>
```

```
<wsdl:output message="impl:publishResponse" name="publishResponse"/>
```

```
</wsdl:operation>
```

```
</wsdl:portType>
```

```
<wsdl:binding name="BrokerSoapBinding" type="impl:Broker">
```

```
<wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
<wsdl:operation name="consult">
```

```
<wsdlsoap:operation soapAction=""/>
```

```
<wsdl:input name="consultRequest">
```

```
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
namespace="urn:wsqos.Broker" use="encoded"/>
```

```
</wsdl:input>
```

```
<wsdl:output name="consultResponse">
```



```
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
namespace="urn:wsqos.Broker" use="encoded"/>
```

```
</wsdl:output>
```

```
</wsdl:operation>
```

```
<wsdl:operation name="compare">
```

```
<wsdlsoap:operation soapAction=""/>
```

```
<wsdl:input name="compareRequest">
```

```
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
namespace="urn:wsqos.Broker" use="encoded"/>
```

```
</wsdl:input>
```

```
<wsdl:output name="compareResponse">
```

```
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"  
namespace="urn:wsqos.Broker" use="encoded"/>
```

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="publish">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="publishRequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded"/>

</wsdl:input>

<wsdl:output name="publishResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:wsqos.Broker" use="encoded"/>

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="BrokerService">

<wsdl:port binding="impl:BrokerSoapBinding" name="Broker">

<wsdlsoap:address location="http://127.0.0.1:8080/axis/services/Broker"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

testebroker.htm

```
<html>

  <head>

    <title>Test Broker </title>

    <script type="text/javascript" src="zxml.js"></script>

    <script type="text/javascript"

src="UniversalWebServiceExample.js"></script>

    <script type="text/javascript">

var SERVICE_URL = "http://127.0.0.1:8080/axis/services/Broker";

var SOAP_ACTION_BASE = "urn:wsqos.Broker";

function setUIEnabled(bEnabled) {

  var oButton = document.getElementById("cmdRequest");

  oButton.disabled = !bEnabled;

  var oList = document.getElementById("lstMethods");

  oList.disabled = !bEnabled

}

function performOperation() {

  var oList = document.getElementById("lstMethods");

  var sMethod = oList.options[oList.selectedIndex].value;

  var sOp1 = document.getElementById("txtOp1").value;
```

```
//Clear the message panes

document.getElementById("txtRequest").value = "";

document.getElementById("txtResponse").value = "";

document.getElementById("txtResult").value = "";

performSpecificOperation(sMethod, sOp1);

}

</script>

</head>

<body onload="setUIEnabled(true)">

    Operation:

    <select id="lstMethods" style="WIDTH: 200px" disabled>

        <option value="consult" selected>consult</option>

        <option value="publish">publish</option>

    </select>

    <br>

    <br>

    Operand 1: <input type="text" id="txtOp1" size="10">WS SPAM Server
description<br>

    <br>

    <input type="button" id="cmdRequest" value="Perform Operation"
onclick="performOperation();"

        disabled>

    <br>
```

```

        <br>
        <input type="text" size="20" id="txtResult">
        <br>
        <textarea rows="30" cols="60" id="txtRequest"></textarea> <textarea
rows="30" cols="60" id="txtResponse"></textarea>
    </body>

```

```
</html>
```

```
zxml.js
```

```

var zXml={useActiveX:(typeof
ActiveXObject!="undefined"),useDom:document.implementation&&document.implement
ation.createDocument,useXmlHttp:(typeof
XMLHttpRequest!="undefined");zXml.ARR_XMLHTTP_VERS=["MSXML2.XmlHttp.
5.0","MSXML2.XmlHttp.4.0","MSXML2.XmlHttp.3.0","MSXML2.XmlHttp","Microsoft.
XmlHttp"];zXml.ARR_DOM_VERS=["MSXML2.DOMDocument.5.0","MSXML2.DOM
Document.4.0","MSXML2.DOMDocument.3.0","MSXML2.DOMDocument","Microsoft.
XmlDom"];function
zXmlHttp(){zXmlHttp.createRequest=function(){if(zXml.useXmlHttp){return new
XMLHttpRequest();}else if(zXml.useActiveX){if(!zXml.XMLHTTP_VER){for(var
i=0;i<zXml.ARR_XMLHTTP_VERS.length;i++){try{new
ActiveXObject(zXml.ARR_XMLHTTP_VERS[i]);zXml.XMLHTTP_VER=zXml.ARR_X
MLHTTP_VERS[i];break;}catch(oError){}}}if(zXml.XMLHTTP_VER){return new
ActiveXObject(zXml.XMLHTTP_VER);}else{throw new Error("Could not create XML
HTTP Request.");}}else{throw new Error("Your browser doesn't support an XML HTTP
Request.");}};zXmlHttp.isSupported=function(){return
zXml.useXmlHttp||zXml.useActiveX;};function
zXmlDom(){zXmlDom.createDocument=function(){if(zXml.useDom){var
oXmlDom=document.implementation.createDocument("", "", null);oXmlDom.parseError={
valueOf:function(){return this.errorCode;},toString:function(){return
this.errorCode.toString();}};oXmlDom.__initError__();oXmlDom.addEventListener("load",
function(){this.__checkForErrors__();this.__changeReadyState__(4);},false);return
oXmlDom;}}else if(zXml.useActiveX){if(!zXml.DOM_VER){for(var
i=0;i<zXml.ARR_DOM_VERS.length;i++){try{new
ActiveXObject(zXml.ARR_DOM_VERS[i]);zXml.DOM_VER=zXml.ARR_DOM_VERS
[i];break;}catch(oError){}}}if(zXml.DOM_VER){return new
ActiveXObject(zXml.DOM_VER);}else{throw new Error("Could not create XML DOM
document.");}}else{throw new Error("Your browser doesn't support an XML DOM
document.");}};zXmlDom.isSupported=function(){return
zXml.useDom||zXml.useActiveX;};var oMozDocument=null;if(typeof
XMLDocument!="undefined"){oMozDocument=XMLDocument;}else if(typeof
Document!="undefined"){oMozDocument=Document;}if(oMozDocument&&!window.op

```

```

era){oMozDocument.prototype.readyState=0;oMozDocument.prototype.onreadystatechange
e=null;oMozDocument.prototype.__changeReadyState__=function(iReadyState){this.read
yState=iReadyState;if(typeof
this.onreadystatechange=="function"){this.onreadystatechange();}};oMozDocument.protot
ype.__initError__=function(){this.parseError.errorCode=0;this.parseError.filepos=-
1;this.parseError.line=-1;this.parseError.linepos=-
1;this.parseError.reason=null;this.parseError.srcText=null;this.parseError.url=null;};oMoz
Document.prototype.__checkForErrors__=function(){if(this.documentElement.tagName=="
parsererror"){var reError=/>([\s\S]*?)Location:([\s\S]*?)Line
Number(\d+),Column(\d+):<sourcetext>([\s\S]*?)(?:\s
*\^)/;reError.test(this.xml);this.parseError.errorCode=-
999999;this.parseError.reason=RegExp.$1;this.parseError.url=RegExp.$2;this.parseError.li
ne=parseInt(RegExp.$3);this.parseError.linepos=parseInt(RegExp.$4);this.parseError.srcTe
xt=RegExp.$5;};};oMozDocument.prototype.loadXML=function(sXml){this.__initError__(
);this.__changeReadyState__(1);var oParser=new DOMParser();var
oXmlDom=oParser.parseFromString(sXml,"text/xml");while(this.firstChild){this.removeC
hild(this.firstChild);}for(var i=0;i<oXmlDom.childNodes.length;i++){var
oNewNode=this.importNode(oXmlDom.childNodes[i],true);this.appendChild(oNewNode);
}this.__checkForErrors__();this.__changeReadyState__(4);};oMozDocument.prototype.__l
oad__=oMozDocument.prototype.load;oMozDocument.prototype.load=function(sURL){th
is.__initError__();this.__changeReadyState__(1);this.__load__(sURL);};Node.prototype._
__defineGetter__("xml",function(){var oSerializer=new XMLSerializer();return
oSerializer.serializeToString(this,"text/xml");});Node.prototype.__defineGetter__("text",fu
nction(){var sText="";for(var
i=0;i<this.childNodes.length;i++){if(this.childNodes[i].hasChildNodes()){sText+=this.chil
dNodes[i].text;}else{sText+=this.childNodes[i].nodeValue;}}return sText;});function
zXslt(){zXslt.transformToText=function(oXml,oXslt){if(typeof
XSLTProcessor!="undefined"){var oProcessor=new
XSLTProcessor();oProcessor.importStylesheet(oXslt);var
oResultDom=oProcessor.transformToDocument(oXml);var
sResult=oResultDom.xml;if(sResult.indexOf("<transformiix:result">-
1){sResult=sResult.substring(sResult.indexOf(">")+1,sResult.lastIndexOf("<"));}return
sResult;};else if(zXml.useActiveX){return oXml.transformNode(oXslt);};else{throw new
Error("No XSLT engine found.");};};function
zXPath(){zXPath.selectNodes=function(oRefNode,sXPath,sXmlNs){if(typeof
XPathEvaluator!="undefined"){oXmlNs=oXmlNs||{};var
nsResolver=function(sPrefix){return oXmlNs[sPrefix];};var oEvaluator=new
XPathEvaluator();var
oResult=oEvaluator.evaluate(sXPath,oRefNode,nsResolver,XPathResult.ORDERED_NOD
E_ITERATOR_TYPE,null);var aNodes=new Array;if(oResult!=null){var
oElement=oResult.iterateNext();while(oElement){aNodes.push(oElement);oElement=oRes
ult.iterateNext();}}return aNodes;};else if(zXml.useActiveX){if(oXmlNs){var
sXmlNs="";for(var sProp in oXmlNs){sXmlNs+="xmlns:"+sProp+"="+oXmlNs[sProp]+
";"}oRefNode.ownerDocument.setProperty("SelectionNamespaces",sXmlNs);};return
oRefNode.selectNodes(sXPath);};else{throw new Error("No XPath engine
found.");};};zXPath.selectSingleNode=function(oRefNode,sXPath,oXmlNs){if(typeof
XPathEvaluator!="undefined"){oXmlNs=oXmlNs||{};var
nsResolver=function(sPrefix){return oXmlNs[sPrefix];};var oEvaluator=new

```

```

XPathEvaluator();var
oResult=oEvaluator.evaluate(sXPath,oRefNode,nsResolver,XPathResult.FIRST_ORDERED_NODE_TYPE,null);if(oResult!=null){return oResult.singleNodeValue;}else{return null;};}else if(zXML.useActiveX){if(oXmlNs){var sXmlNs="";for(var sProp in oXmlNs){sXmlNs+="xmlns:"+sProp+"="+oXmlNs[sProp]+"";}oRefNode.ownerDocument.setProperty("SelectionNamespaces",sXmlNs);}return oRefNode.selectSingleNode(sXPath);}else{throw new Error("No XPath engine found.");};};function
zXMLSerializer(){zXMLSerializer.prototype.serializeToString=function(oNode){var sXml="";switch(oNode.nodeType){case 1:sXml="<"+oNode.tagName;for(var i=0;i<oNode.attributes.length;i++){sXml+=" "+oNode.attributes[i].name+"=\""+oNode.attributes[i].value+"\"";}sXml+=">";for(var i=0;i<oNode.childNodes.length;i++){sXml+=this.serializeToString(oNode.childNodes[i]);}sXml+="</"+oNode.tagName+">";break;case 3:sXml=oNode.nodeValue;break;case 4:sXml="<![CDATA["+oNode.nodeValue+"]>";break;case 7:sXml="<?"+oNode.nodeValue+"?>";break;case 8:sXml="<!--"+oNode.nodeValue+"-->";break;case 9:for(var i=0;i<oNode.childNodes.length;i++){sXml+=this.serializeToString(oNode.childNodes[i]);}break;};return sXml;};
    
```

UniversalWebServiceExample.js

//Universal Test Harness Example

//handles the request and response

var oXmlHttp = null;

//This function constructs the SOAP request as a string.

function getRequest(sMethod, sOp1, sOp2)

{

```

var sRequest = "<soap:Envelope xmlns:xsi=\""
    + "http://www.w3.org/2001/XMLSchema-instance\" "
    + "xmlns:xsd=\""http://www.w3.org/2001/XMLSchema\" "
    + "xmlns:soap=\""http://schemas.xmlsoap.org/soap/envelope/\">\n"
    + "<soap:Body>\n"
    
```



```
+ "<" + sMethod + " xmlns=\"" + SOAP_ACTION_BASE + "\">\n"  
+ "<op1>" + sOp1 + "</op1>\n"  
+ "</" + sMethod + ">\n"  
+ "</soap:Body>\n"  
+ "</soap:Envelope>\n";  
  
return sRequest;  
}
```

```
function performSpecificOperation(sMethod, sOp1, sOp2)  
{  
    oXmlHttp = zXmlHttp.createRequest();  
    setUIEnabled(false);  
    var sRequest = getRequest(sMethod, sOp1, sOp2);  
    var sSoapActionHeader = SOAP_ACTION_BASE + "/" + sMethod;  
    oXmlHttp.open("POST", SERVICE_URL, true);  
    oXmlHttp.onreadystatechange = handleResponse;  
    oXmlHttp.setRequestHeader("SOAPAction", sSoapActionHeader);  
    oXmlHttp.setRequestHeader("Content-Type", "text/xml");  
    oXmlHttp.send(sRequest);  
    document.getElementById("txtRequest").value = sRequest;  
}
```

//This handles the response

```
function handleResponse()
```

```
{  
  if (oXmlHttp.readyState == 4)  
  {  
    setUIEnabled(true);  
    var oResponseOutput = document.getElementById("txtResponse");  
    var oResultOutput = document.getElementById("txtResult");  
    var oXmlResponse = oXmlHttp.responseXML;  
    var sHeaders = oXmlHttp.getAllResponseHeaders();  
    if (oXmlHttp.status != 200 || !oXmlResponse.xml)  
    {  
      alert("Error accessing Web service.\n"  
        + oXmlHttp.statusText  
        + "\nSee response pane for further details.");  
      var sResponse = (oXmlResponse.xml ? oXmlResponse.xml : oXmlResponseText);  
      oResponseOutput.value = sHeaders + sResponse;  
      return;  
    }  
    oResponseOutput.value = sHeaders + oXmlResponse.xml;  
    var sResult =  
oXmlResponse.documentElement.firstChild.firstChild.firstChild.firstChild.data;  
    oResultOutput.value = sResult;  
  }  
}
```