UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CLAUDIMIR ZAVALIK

Integração de Sistemas de Informação através de *Web services*

Trabalho de Conclusão de Mestrado apresentado como requisito parcial para a obtenção do grau de Mestre em Informática.

Prof. Dr. José Palazzo M.de Oliveira
Orientador

Porto Alegre, novembro de 2004.

CIP - CATALOGAÇÃO DA PUBLICAÇÃO

Zavalik, Claudimir

Integração de Sistemas de Informação através de Web Services / Claudimir Zavalik. - Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

71 f.: il.

Trabalho de Conclusão (mestrado) - Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR-RS, 2004. Orientador: José Palazzo M.de Oliveira.

1. *Web Services.* 2. Integração de Sistemas de Informação. 3.Internet. I. Oliveira, José Palazzo M.de. II.Título

AGRADECIMENTOS

À família, que sempre motivou a busca constante de novos horizontes e desafios.

Aos professores do Instituto de Informática, que se empreenderam de forma exemplar na tarefa de transmitir o conhecimento.

Aos amigos, e em especial ao amigo Guilherme Lacerda, sem o qual jamais seria possível a concretização deste trabalho, pela sua dedicação, persistência e companheirismo no ambiente de trabalho e vida acadêmica.

À comunidade do software livre dispersa no mundo, que trabalha incansavelmente para produzir software de qualidade, beneficiando toda a sociedade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
BPEL4WS	*Business Process Extensible Language for Web Service*
BPMI	*Business Process Management Initiative*
BPML	*Business Process Modeling Language*
CIMOSA	*CIM Open System Architecture*
DTD	*Document Type Definition*
ebXML	*Eletronic Business XML*
FTP	*File Transfer Protocol*
GERAM	*Generalised Enterprise Reference Architecture and Methodology*
GNU/GPL	*Gnu is Not Unix/General Public License*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
JWSDP	*Java Web Service Developer Pack*
OASIS	*Organization for the Advancement of Structured Information Standards*
PHP	*Personal Home Page*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Access Protocol*
TI	*Tecnologia da Informação*
UDDI	*Universal Description, Discovery, and Integration*
URL	*Uniform Resource Locator*
WSDL	*Web Service Description Language*
XML	*eXtensible Markup Language*
XPDL	*XML Process Definition Language*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Sistemas de informação incorporam processos de negócios particulares de cada organização. A medida em que se observa uma crescente pressão de mercado para que empresas troquem informações de forma automatizada e segura para obtenção de melhores resultados, faz-se necessário repensar a forma como são concebidos os sistemas de informação, desde a modelagem da empresa propriamente dita até a modelagem dos processos de negócio e sua interação com os demais colaboradores.

Modelar os processos de negócio de uma empresa em um contexto global significa não apenas estabelecer regras de comportamento, mas também expressar a forma como os processos poderão ser acionados e interagir com sistemas de informação diferentes.

Existem várias tecnologias empregadas para a integração de sistemas de informação. Entre tantas tecnologias, uma delas vêm recebendo especial atenção: a tecnologia *Web services.*

A suposta interoperabilidade dos *Web services* permite a comunicação de aplicações desenvolvidas em diferentes plataformas de *hardware* e diferentes linguagens de programação através da Internet ou de uma rede local.

No entanto, algumas particularidades devem ser observadas para que a implementação de *Web services* seja eficiente. Disponibilizar processos de negócio de uma empresa através da Internet pode ser uma ótima opção para o incremento de suas atividades, mas requer cuidados especiais.

Este trabalho apresenta uma revisão bibliográfica sobre a modelagem de empresas, modelagem de processos de negócio e a integração de sistemas de informação através do uso de *Web services.*

Através de um estudo de caso, são apresentados os principais conceitos e as etapas necessárias para a implementação de *Web services* em um sistema *Web*.

Como contribuição deste trabalho, é proposta uma alternativa de modelagem de sistemas que permite um melhor controle sobre o tratamento de exceções em *Web services.* O trabalho desenvolvido compreendeu a especificação, desenvolvimento e aplicação de um ambiente para suportar esta classe de aplicação. No texto é descrito o funcionamento da biblioteca NuSOAP, apresentando o código-fonte completo da aplicação desenvolvida, acessando *Web services* através de chamadas em alto nível (WSDL). Com o presente trabalho, tem-se uma proposta, já avaliada e validada, para funcionar como referencial conceitual e prático para o desenvolvimento de aplicações usando a tecnologia de *Web services*.

Palavras-chave: integração de sistemas de informação, *Web*, *Web services*, computação distribuída.

Integration of information systems through Web services

ABSTRACT

Information systems are able to comprise the different and singular processes that are unique to each organization business. Today, we see an increasing market pressure for companies to exchange information automatically and safely, in order to obtain better results, it is necessary to rethink the way how information systems are conceived. The entire process should be given attention, from the company modeling to the modeling of negotiation processes and the interaction with collaborators.

To model the business processes of a company in a global context does not mean only to set behavior rules, but also to express the way how processes could be triggered and how they could interact with different information systems.

Many different technologies have been used to integrate information systems. Of these, one of them has been given special attention: the Web services technology.

The interoperability assumed for Web services enables the communication of applications developed in different hardware platforms and different programming languages over the Internet or local network.

Nevertheless, some particular issues should be given attention so that the implementation of web services works efficiently. To make the negotiation process of a company available on the Internet, for example, can be a good choice in the incrementation of the company's activities, however, it requires especial concern.

This study makes a bibliographic review on companies modeling, modeling of negotiation processes and the interaction of information systems by using Web services. The study case presented here shows the main concepts and necessary steps for the implementation of Web services in a Web system.

The main contribution intended is the proposal of an alternative for systems modeling that can be able to improve the management of exceptions in Web services. The study we developed comprised specification, development and application of an environment that could support this type of application. The NuSOAP library, whose Web services are accessed through high level calls (WSDL), is described and its complete source code is opened. Our proposal was already tested and assessed, and it has proved to work as a conceptual and practical reference for the development of applications that use the Web services technology.

Keywords: integration of information systems, Web, Web services, distributed computing.

1 INTRODUÇÃO

Historicamente, a Revolução Industrial colocou no centro das atenções a produção como elemento chave para o enriquecimento das nações. Nesta fase, produzir com rapidez e em larga escala eram princípios norteadores da atividade de qualquer empresa.

Vários pesquisadores teceram estudos de grande valia para aumento da produção industrial, com otimização de tempos e movimentos, produção linear e até mesmo relações humanas de trabalho, algo desconsiderado até certo momento.

Com a popularização do uso da informática, em meados de 1980, as empresas perceberam a necessidade de controlar seus processos de produção com o uso de computadores.

Desde a introdução da informática no mercado corporativo, muitas foram as evoluções tecnológicas e mudanças culturais que influenciaram a dinâmica de trabalho das empresas. As relações comerciais entre as empresas mudaram, surgindo empresas que simplesmente integravam produtos de outros fabricantes, gerando um novo produto.

Em conseqüência, surgiram várias iniciativas no sentido de buscar formalizar o processo de produção das empresas, culminando com o surgimento de novas metodologias para *Enterprise Modeling* (Modelagem de Empresas), como CIMOSA e GERAM.

Algumas propostas para Modelagem de Empresas foram criadas com o intuito de uniformizar as relações entre as empresas, criando procedimentos padronizados, onde diferentes fornecedores poderiam interagir seus núcleos de produção ou administração de recursos.

Nota-se que a visão deste tipo de padronização de processos é difícil de ser implementada, visto que implicaria na mudança do modo de operação das empresas.

Mas, ao mesmo tempo que vê-se uma pressão pela não-uniformização dos processos nas empresa, percebe-se a importância em analisar o fluxo de trabalho interno, e a forma como estes processos podem interagir entre si, mesmo entre empresas diferentes.

A grande quantidade de aplicações voltadas para a *Web*, a adoção de soluções corporativas de gestão, a implantação de linhas de produção *just-in-time* são alguns exemplos que motivam a modelagem de processos de negócio.

Formalizar estes processos de negócio torna-se, então, uma necessidade para um melhor entendimento do fluxo de trabalho de uma empresa, bem como para permitir, se necessário, a terceirização ou a reestruturação de núcleos de produção em específico.

Com um número crescente de processos de negócio disponibilizados publicamente, surgiu a necessidade de integração de sistemas de informação através da Internet, por ser um meio de comunicação abrangente e de baixo custo.

Várias foram as tecnologias desenvolvidas para a integração de sistemas de informação, como comunicação por *sockets*, RMI e RPC. Entre estas novas tecnologias, observa-se a crescente adoção de *Web services* como meio de disponibilização e uso de serviços pela Internet.

Este trabalho visa apresentar um estudo sobre a integração de sistemas de informação. São apresentados alguns padrões XML para modelagem conceitual de processos de negócio, conceitos e características de *Web services*, bem como um estudo de caso da implementação de *Web services* em um sistema já existente.

Concluindo o trabalho, são apresentadas algumas considerações finais com ênfase na proposta de adoção de uma metodologia para modelagem de tratamento de exceções em *Web services*.

2 MODELAGEM DE EMPRESAS

Desde a introdução da informática no mercado corporativo, muitas foram as evoluções tecnológicas e mudanças culturais que influenciaram a dinâmica de trabalho das empresas. Algumas vezes, um processo de informatização mal conduzido vinha a comprometer todo o funcionamento da organização.

Modelar empresas, então, passa a ser uma atividade antecessora de qualquer planejamento de informatização. A transcrição do modo de operação de qualquer organização para um modelo formal deve ser suficiente completo e tecnicamente conciso a ponto de eliminar ambigüidades e conseguir apresentar o seu funcionamento em diferentes graus de granularidade.

2.1 Conceitos

A modelagem de empresas é uma representação computacional de estrutura, atividades, processos, informações, recursos, pessoas, comportamento, objetivos e restrições de um negócio, governo ou empresa [EIL 2002].

Muitos destes processos ocorrem de forma concorrente por um conjunto de agentes, visando atingir os objetivos do negócio.

A modelagem de empresas é uma maneira de gerenciar a complexidade de uma empresa tornando-se um pré-requisito para a integração de empresas [VER 1999].

Ao modelar-se uma empresa, são descritos vários aspectos referentes à estrutura, organização e comportamento da empresa, permitindo uma melhor análise e compreensão de sua complexidade. A Figura 2.1 apresenta o processo de modelagem de empresas.

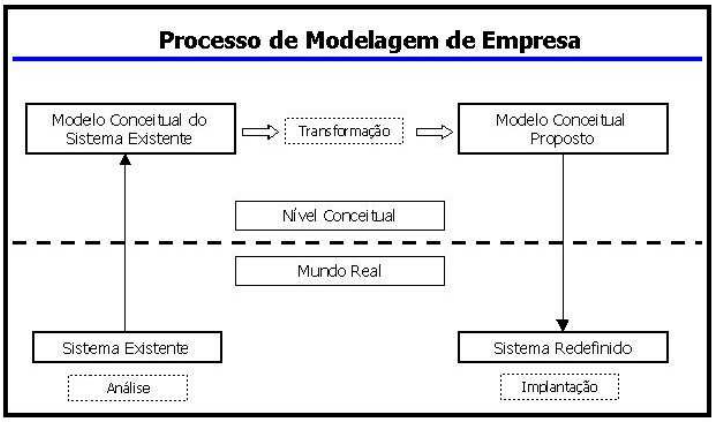

Figura 2.1: Processo de Modelagem de Empresa

A partir de um modelo existente, e após a realização de uma análise da estrutura e comportamento da empresa, é construído um modelo conceitual do sistema existente.

Em uma segunda fase, o modelo conceitual elaborado é então revisado e são feitas alterações no modelo, gerando um modelo conceitual proposto. Uma vez validado este modelo, ele é então implantado no mundo real.

Como benefícios da modelagem de empresas, pode-se citar:

- formalização do *know-how* da organização, suas práticas e conhecimentos;
- documentação dos processos, permitindo um melhor suporte às decisões que visem melhoria e controle das operações da organização, bem como sua informatização;
- criação de uma linguagem compartilhada para descrição das operações da empresa.

A utilização de ferramentas de software para esta finalidade vêm sendo alvo de investimentos de grandes corporações que buscam na formalização do modelo da empresa identificar suas potencialidades e gargalos no processo produtivo.

Como conseqüência, consegue-se encontrar no mercado vários softwares proprietários com esta finalidade, como o Silverrun-BPM, BPWin e outros.

Estas ferramentas normalmente têm um custo elevado, dificultando o acesso das organizações de pequeno porte ao seu uso.

2.2 Metodologias

Entende-se por metodologia todo o conjunto de técnicas e práticas utilizado para atingir algum objetivo. Em suma, as formas que utilizamos para conseguir, enfim o código, o modelo, o protótipo ou o que quer que seja [MAT 2002].

Ainda em [MAT 2002], muitos autores defendem que as metodologias apenas existem para que possam nos nortear, e as etapas estabelecidas por elas podem ser facilmente transpostas se a equipe julgar que o objetivo daquele ponto já foi atingido.

O uso de metodologias consolidadas para a modelagem de empresas garante a integração dos elementos de modelagem dentro do processo.

Este trabalho busca apresentar uma visão geral sobre algumas metodologias amplamente empregadas para a modelagem de empresas e de processos de negócio.

2.2.1 CIMOSA

O *framework* de modelagem exibido na Figura 2.2 estrutura a arquitetura de referência CIMOSA em uma modelagem genérica e parcial a cada nível, suportando diferentes visões sobre um modelo particular de empresa. O conceito de visões permite trabalhar com um subconjunto do modelo ao invés de um modelo completo, fornecendo especialmente os negócios do usuário com uma complexidade reduzida para sua particular área de interesse. CIMOSA tem definidas quatro diferentes visões de modelagem: funções, informações, recursos e organização. Todavia, este conjunto de visões pode ser estendido, se necessário [CIM 2002].

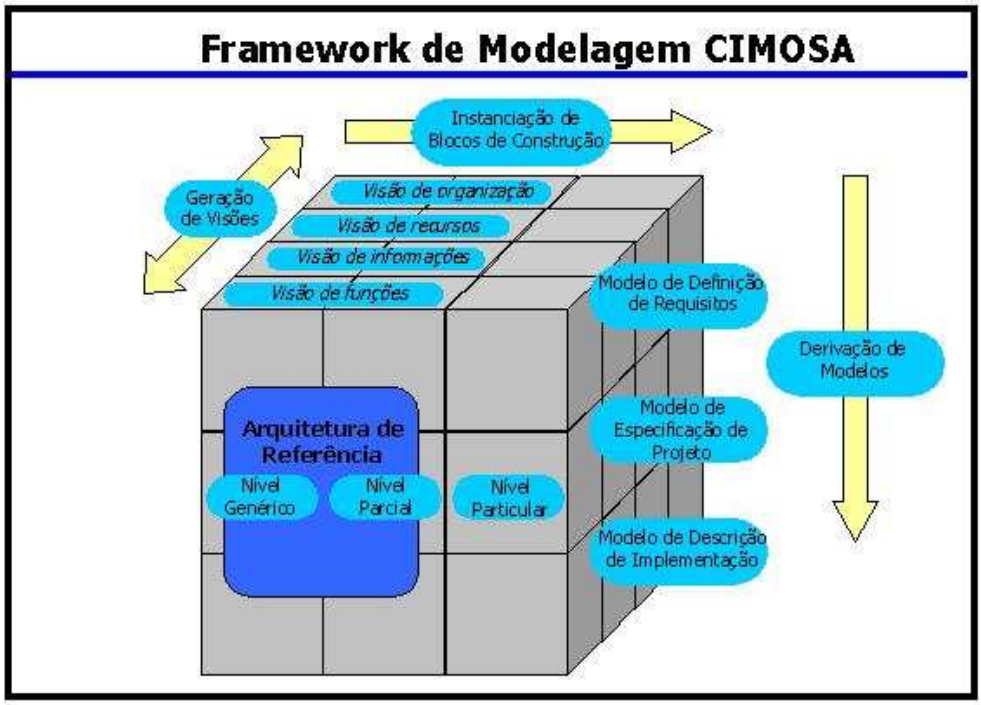

Figura 2.2 : Framework de Modelagem CIMOSA

Na especificação da arquitetura de referência CIMOSA são apresentados três níveis de modelagem para descrever todas as etapas das operações da empresa durante seu ciclo de vida. Estes três níveis (definição de requisitos, especificação de projeto e descrição de implementação) podem ser iniciados a qualquer momento, bem como podem ser interativos.

As operações da empresa devem ser modeladas não como um modelo único e complexo, mas como um conjunto de processos cooperativos. Esta abordagem permite a diferentes pessoas modelar diferentes áreas da empresa, mantendo a integridade do modelo completo.

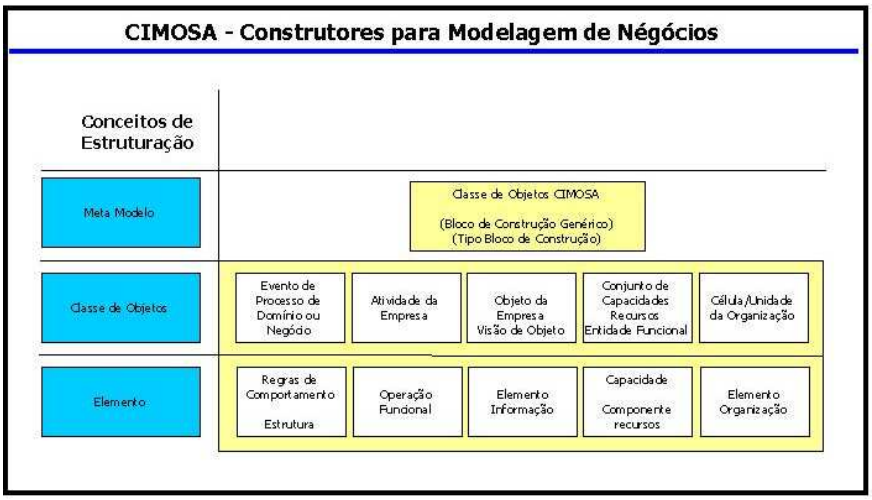

Figura 2.3: Construtores para Modelagem de Negócios CIMOSA

O conjunto básico de blocos construtores para modelagem de negócios é apresentado na Figura 2.3. Processos, eventos e atividades da empresa são as classes de objetos que descrevem funcionalidades e o comportamento (dinâmico) das operações da empresa. Entradas e saídas de Atividades da Empresa definem as informações (Objeto Empresa) e os recursos necessários. Aspectos organizacionais são definidos em termos de

responsabilidades e autorização (Elementos da Organização) para funcionalidades, informações, recursos e organização. Eles estão estruturados em Unidades ou Células Organizacionais. CIMOSA emprega os conceitos da orientação a objetos e herança, estruturando seus construtores em uma hierarquia de classes de objeto [CIM 2002].

A Modelagem de Processos de Negócio com CIMOSA

CIMOSA estrutura os processos sob diferentes granularidades. Inicialmente, são identificados dentro do Domínio da Empresa os Domínios CIMOSA e os Domínios Não-CIMOSA.

Estes, por sua vez, são compostos por Processos de Domínio. Cada Processo de Domínio é formado por um conjunto de Processos de Negócio e Atividades da Empresa, sujeitos às regras de comportamento.

A Figura 2.4 apresenta a decomposição de Processos de Domínio e Processos de Negócio, identificando as atividades da empresa.

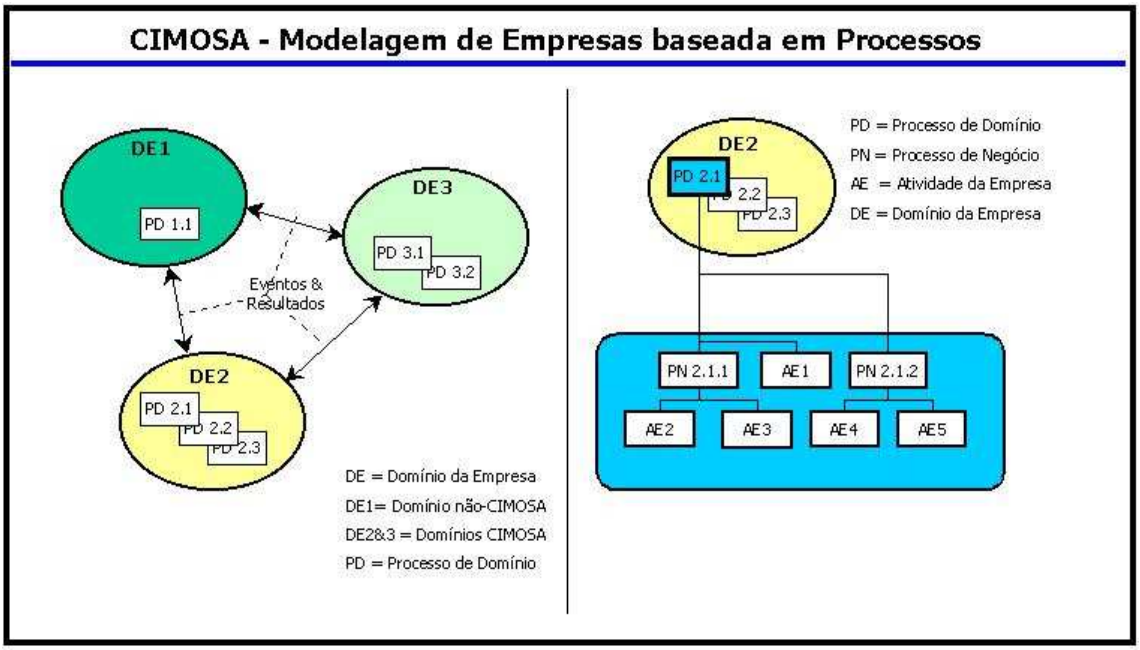

Figura 2.4 : Modelagem de Processos CIMOSA

3 MODELAGEM DE PROCESSOS DE NEGÓCIO

Antes do desenvolvimento de um sistema de informação, é comum os analistas e desenvolvedores fazerem um estudo prévio sobre os requisitos do sistema e suas funcionalidades. A análise e modelagem de processos de negócio permitem a introdução segura de regras, tempos, rotas e papéis funcionais no ambiente organizacional, garantindo o compartilhamento do conhecimento entre os participantes de uma organização.

A modelagem de processos de negócio pode ser realizada de forma mais rápida, fácil e concisa quando apoiada em uma metodologia, linguagem ou ferramenta de *software*.

3.1 Conceitos

O principal objetivo da modelagem de processos de negócio é definir processos de forma que a definição destes possa ser trocada entre diferentes organizações.

Uma "Definição de Processo" é definida como a representação de um negócio em um formato que suporte manipulação automatizada, como uma modelagem para um sistema de gerenciamento de *workflow*. A definição de processo consiste de uma rede de atividades e seus relacionamentos, critério para indicar o início e o término de um processo e informações sobre as atividades individuais, como os participantes, associando dados e aplicações de tecnologia da informação, etc [WMC 2003].

Tem-se, então, como objetivo principal da modelagem de processos de negócio, automatizar os fluxos de trabalho de processos estruturados das organizações. Para aplicarmos uma metodologia de modelagem de processos de negócio, é necessário que os fluxos de trabalho possuam regras, além de uma previsibilidade na seqüência de eventos e na utilização de recursos.

A definição de processos pode ser utilizada para:

- Atuar como um formato modelo para criação e controle de instancias dos processos durante a atividade destes;
- Simulação e previsão;
- Monitoria e análise de processos ativos;
- Documentação, visualização e gerenciamento do conhecimento.

3.2 O uso de XML para modelar processos

Devido a um processo de desenvolvimento tecnológico das empresas, em um dado momento, estas começaram a trocar informações através de meio eletrônico.

Muitas foram as tecnologias e técnicas utilizadas para permitir que os dados de diversas organizações pudessem ser tratadas de forma estruturada e integrada. A troca de informações entre aplicativos diferentes era realizada de diferentes formas, seja por rotinas de transferência de arquivos, envio de mensagens eletrônicas semi-estruturadas, troca de dados em formatos proprietários, entre outras.

Dentro deste cenário surgiu a XML – *eXtensible Markup Language*, uma linguagem flexível, que busca estruturar os dados, desvinculando totalmente a apresentação dos dados de seu conteúdo.

XML é uma linguagem padronizada para a construção de documentos eletrônicos com formato simples, textual, estruturado e flexível a mudanças e portável em diversas plataformas tecnológicas, que tem como objetivo principal descrever informações [FRA 2002].

A adoção de XML como um meio universal de transmissão de informações entre sistemas eletrônicos deve-se também ao fato de ser um padrão aberto, que proporciona ao desenvolvedor acesso as regras e metadados das informações.

XML é uma linguagem padronizada, originalmente direcionada para o processamento de documentos, proposta e controlada pelo mesmo organismo que mantém o padrão HTML (W3C) [DÉC 2000].

Em um nível, XML é um protocolo para conter e gerenciar informações. Em outro nível, ela é uma família de tecnologias que podem fazer várias coisas desde formatar documentos até filtrar dados. E, em um nível mais alto, ela é uma filosofia para tratamento de informações que procura a máxima usabilidade e flexibilidade para dados através do refinamento destes de forma mais estruturada. Uma compreensão completa de XML abrange todos estes níveis [RAY 2001].

3.2.1 Vantagens de XML

XML tem suas raízes na indústria de publicações (documentação, livros, páginas *Web*) e, portanto, sua aplicação nestas áreas é claramente visível. Entretanto, graças à simplicidade sintática, à disponibilidade de ferramentas, e à capacidade de descrever dados de maneira estruturada, XML passou a ser utilizada em múltiplas aplicações [DÉC 2000].

Entre as principais vantagens da XML, pode-se citar:

Padrão aberto

Por se tratar de um padrão aberto, permite customizações e alterações em sua estrutura.

Formato texto

O uso de um formato comum, ao invés de um formato proprietário de armazenamento de dados, garante a portabilidade necessária para a integração de sistemas entre diferentes plataformas e sistemas operacionais.

Separação entre estrutura, conteúdo e apresentação

Com XML, é possível tratar as informações sobre estes diferentes aspectos. Esta separação facilita a geração de dados para visualização dinâmica, bem como evita a repetição de informação resultando em uma simplificação da manutenção dos dados.

Permite análise semântica na *Web*

O linguagem HTML não carrega consigo a semântica dos dados apresentados, apenas informações sobre a apresentação, prejudicando os mecanismos de busca de conteúdo na *Web*. Por outro lado, XML permite apresentar a semântica dos dados, e sua implementação depende apenas do suporte nas aplicações.

Facilidade de transformação

Dados apresentados em formato XML podem ser facilmente transformados em outros formatos, através de *parsers* e APIs já disponíveis para diversas finalidades.

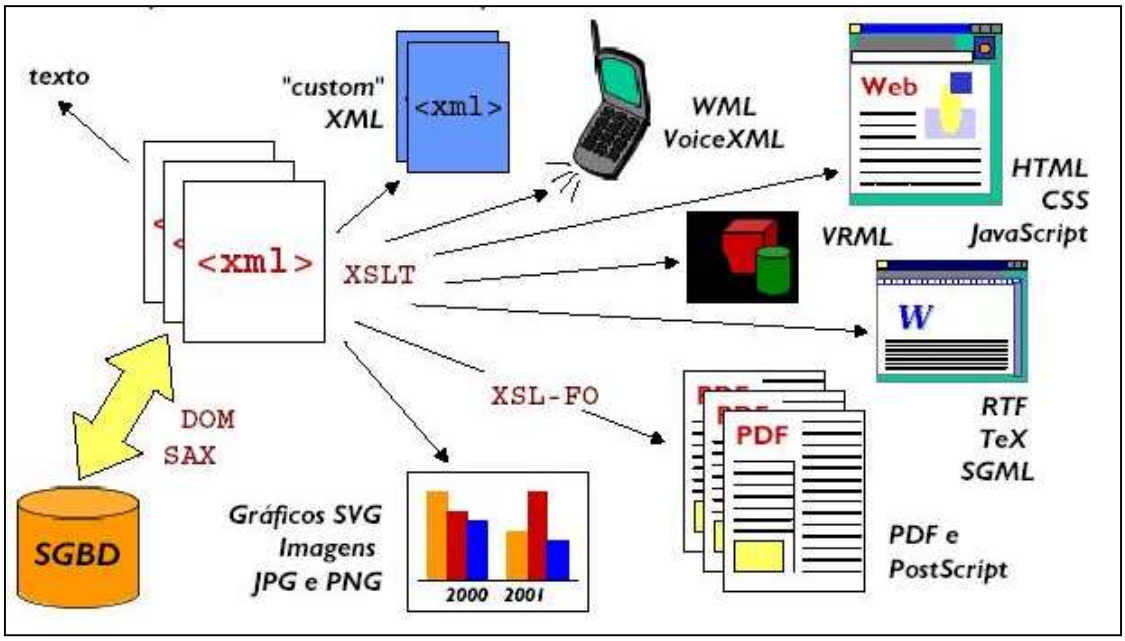

Figura 3.1: Algumas transformações possíveis a partir de XML

3.2.2 Validação de documentos XML através de DTD

Ao criarmos um documento XML, faz-se necessário descrever sua estrutura, para evitar que os dados sejam armazenados de forma incorreta. Isto é feito através da criação de uma DTD (*Document Type Definition*).

A DTD é descrita em uma sintaxe própria, distinta de XML, e visa identificar os elementos possíveis dentro do documento, seus atributos, relacionamentos e restrições de conteúdo.

No entanto, por possuir algumas limitações, não é o padrão adotado para a validação dos documentos XML das linguagens de modelagem de processos de negócio apresentadas.

Usualmente, esta validação dos documentos XML descritos nas linguagens de modelagem analisadas é feita através de um XML Schema.

3.2.3 O Uso de XML Schema

A validação da estrutura de um documento XML através de DTD impõe ao desenvolvedor o conhecimento de uma nova sintaxe, visto que a DTD possui uma estrutura diferente de XML.

Devido a esta dificuldade, bem como a limitações de DTD, surgiu XML Schema.

Ao contrário da sintaxe DTD, a sintaxe de XML Schema é um documento XML bem formado, tornando possível usar editores XML para editá-lo. Ela também fornece muito mais controle sobre tipos de dados e padrões, tornando-a uma linguagem mais atrativa para reforçar (enforcing) requisitos de entrada de dados restritos.

Em XML Schema, há um maior controle de tipos de dados válidos do que em DTD. Alguns tipos de dados pré definidos em XML Schema são: *byte, float, long, time, date, boolean, binary, language*. Existem vários outros tipos, os quais tornam XML Schema muito excitante para certos documentos, especialmente aqueles que tratam com tipos específicos de aplicações de dados, como os bancos de dados e formulários de entrada de pedidos. Ao invés do desenvolvedor ter que escrever um programa que verifica os tipos de dados, o *parser* XML executa o serviço do desenvolvedor [RAY 2001].

Em [BRA 2002], XML Schema é apresentada como uma alternativa nova e poderosa ao recurso de DTD utilizado até então.

Em [RAY 2001], XML Schema fornece uma alternativa interessante a DTDs porque permite aos arquitetos de documentos projetar campos em detalhes muito mais finos.

O uso de XML Schema para a criação de linguagens de modelagem de processos de negócio busca, entre outras finalidades, expressar de forma transparente em um padrão aberto, os conceitos sugeridos e as regras definidas na linguagem.

Esta transparência e flexibilidade fornecem ao desenvolvedor uma facilidade de interpretação e uma possibilidade de adequar uma determinada linguagem a seus conceitos. Talvez por este motivo, tenhamos tantas variedades de linguagens de modelagem de processos de negócios, com enfoques diferentes.

A partir da XML, fabricantes e instituições passaram a desenvolver uma série de modelos, objetivando atender de forma particular a troca de informações sobre fluxos de trabalho, os chamados XML Schemas.

Neste trabalho, descreve-se alguns destes modelos, com enfoque naqueles que são identificados como potenciais padrões a se estabelecerem.

É o caso do BPML – *Business Process Modeling Language,* que está em pleno desenvolvimento e passando por etapas de validação pelos seus associados e tende a se tornar o padrão mais empregado nos próximos tempos, pelas soluções de *Workflow /* BPM [FRA 2002].

3.3 Linguagens para Modelagem de processos de negócio

A análise e modelagem dos Processos de Negócio permitem a introdução segura de regras, tempos, rotas e papéis funcionais no ambiente organizacional. Quer os processos já existam de forma não estruturada, quer seja um novo processo necessário para suportar um novo negócio. A análise e modelagem garantem o compartilhamento do conhecimento de todos os participantes de uma organização e a gerência desse mesmo conhecimento no cotidiano das operações.

Uma variedade de diferentes ferramentas podem ser utilizadas para analisar, modelar, descrever e documentar um processo de negócio. A interface de definição de processo de *workflow* define um formato comum de intercâmbio, o qual suporta a transferência de definições de processos de *workflow* entre produtos separados .

A interface também define uma separação formal entre o desenvolvimento e o ambiente de execução, habilitando a uma definição de processo, gerada por uma ferramenta de modelagem, ser usada como entrada para diferentes produtos de execução de *workflow* [XPD 2002].

Em [XPD 2002], uma definição de processo é definida como a representação de um processo de negócios em um formato que suporta manipulação automatizada, assim como a modelagem ou ativação através de um sistema de gerenciamento de *workflow.* A definição de processo consiste de uma rede de atividades e seus relacionamentos, critérios para indicar

o início e o término dos processos, e informação sobre as atividades individuais,como os participantes, dados e aplicações de TI associadas, etc.

A adoção de padrões XML para gerenciamento de processos de negócio visa viabilizar a troca de dados de fluxos de trabalhos. Porém, simplesmente adotar XML não é suficiente para garantir a padronização do meio empregado para troca de dados. Surgiram iniciativas de fabricantes e instituições representativas do setor que passaram a desenvolver uma série de modelos atendendo de forma particular a troca de informações sobre fluxos de trabalho.

3.3.1 BPEL4WS (*Business Process Execution Language for Web Services)*

BPEL4WS é uma linguagem para explicitamente descrever fluxos dentro de um processo de negócio que podem ser compostos de múltiplos serviços *Web*. Ele assume um ambiente de execução para programas escritos na linguagem. As novas linguagens fazem a comunicação e estrutura de interação de um processo de negócio explícito, o qual é um pré-requisito para a composição flexível de processos [BPE 2002].

BPEL4WS procura analisar de forma separada os aspectos públicos do comportamento dos processos de negócio e os aspectos internos ou privados. Um dos motivos é que, nos negócios realizados, as organizações obviamente não querem expor as tomadas de decisão internas, bem como o gerenciamento de seus dados a seus parceiros de negócios. Outro motivo é a possibilidade de modificar aspectos privados da implementação do processo sem afetar o protocolo de processos de negócio.

BPEL4WS usa a noção de propriedades de mensagem para identificar dados relevantes ao protocolo em mensagens. Propriedades podem ser vistas como dados relevantes “transparentes” para aspectos públicos, o oposto aos dados “opacos” que as funções privadas e internas utilizam [BUS 2003].

A linguagem é uma camada sobre várias especificações XML: WSDL 1.1, XML Schema 1.0 e Xpath 1.0. Dentre estas especificações, a WSDL foi a que teve a maior influência para a linguagem BPEL4WS. Mensagens WSDL e definições de tipos em XML Schema fornecem o modelo de dados usados por processos BPEL4WS. XPath fornece apoio para manipulação de dados. Todos os recursos externos e parceiros são representados como serviços WSDL. BPEL4WS fornece extensibilidade para acomodar futuras versões destes padrões, especificamente o XPath e padrões relacionados utilizados em computação XML.

A descrição de processos de negócio com BPEL4WS pode ser feita de duas formas. Processos de negócio **executáveis** modelam o comportamento atual de um participante em uma interação de negócios. Em processos executáveis, a atenção é fazer com que sejam separados os aspectos externamente visíveis ou públicos dos processos de negócio de suas funções internas. **Protocolos de negócio**, em contraste, usam descrições de processos os quais especificam o comportamento de troca de mensagens mutuamente visível para cada uma das partes envolvidas no processo, sem revelar seu comportamento interno[BUS 2003].

A BEL4WS define a notação para especificação de comportamento de processos de negócio. Os processos em BPEL4WS exportam e importam funcionalidades através do uso unicamente de interfaces *Web services*.

BPEL4WS fornece uma linguagem para especificação formal de processos de negócio e protocolos de interação de negócios [BPE 2002].

Existem duas boas razões para separar os aspectos públicos do comportamento dos processos de negócios dos aspectos internos ou privados. Um é que negociadores obviamente não querem revelar todas as suas decisões internas e gerenciamento de dados para seus parceiros de negócios. A outra razão é que, separar processos públicos de

processos privados fornece a liberdade para trocar aspectos privados da implementação de processos sem afetar o protocolo de negócios.

As mensagens em BPEL4WS são tratadas de duas formas: elas podem ser “transparentes”, para descrever dados relevantes para aspectos públicos; ou “opacas”, para descrever dados para uso de funções internas e privadas.

```
<process name="ncname" targetNamespace="uri"
      queryLanguage="anyURI"?
      expressionLanguage="anyURI"?
      supressJoinFailure="yes|no"?
      xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/~">

   <partners>?
      <!-- Nota: No mínimo uma regra deve ser especificada -->
      <partner name="ncname" serviceLinkType= "qname"
               myRole="ncname"? partnerRole="ncname"?>+
      </partner>
   </partners>

   <containers>
      <!-- Nota: O tipo da mensagem deve ser indicado no atributo
                 MessageType ou com um elemento <wsdl:message>  -->
      <container name="ncname" messageType= "qname"?>
            <wsdl:message name="ncname">?
               ...
            </wsdl:message>
      </container>
   </containers>

   <correlationSets>?
      <correlationSet name="ncname" properties="qname-list"/>+
   </correlationSets>

   <faultHandlers>?
      <!-- Nota: Deve haver no mínimo um tratador de falhas -->
      <catch faultName="qname"? faultContainer="ncname"?>
          atividade
      </catch>
      <catchAll>
          atividade
      </catchAll>
   </faultHandlers>

   <compensationHandler>?
        atividade
   </compensationHandler>

   atividade

</process>
```

Figura 3.2 : Estrutura de um documento BPEL4WS

3.3.2 BPML (*Business Process Modeling Language*)

A BPML é uma iniciativa do BPMI.org (*Business Process Management Initiative*), uma corporação voluntária que reúne várias companhias e indústrias para desenvolver e operar processos de negócio [BPM 2002].

Trata-se de uma meta-linguagem para modelagem de processos de negócio. A especificação da BPML fornece um modelo abstrato e uma sintaxe XML para expressar processos de negócio executáveis e entidades suportadas. A BPML, por si só, não define qualquer semântica da aplicação como processos particulares ou aplicação de processos em um domínio específico; ao invés disto, ela define um modelo e uma gramática para expressar processos genéricos. Isto permite a BPML ser utilizada por uma variedade de propósitos que incluem a definição de processos de negócios da empresa, a definição de serviços *Web* complexos e a definição de colaborações entre diversos partes.

O conjunto de componentes de processos de negócio modelados com BPML inclue: atividades, processos, contextos, propriedades, sinais, agendas, exceções, transações, tipos de atividade e funções.

Alguns recursos que se destacam na BPML são a separação de fluxo de dados e controle de fluxo, persistência transparente, transações distribuídas e tratamento de exceções.

A documentação da BPML é clara e objetiva, induzindo a um fácil entendimento de suas funcionalidades. Além da facilidade de uso, também são reportadas em sua documentação vantagens como a possibilidade de modelagem de processos de negócio complexos, a solução de problemas criados artificialmente pela tecnologia e uma suave curva de aprendizado.

BPML define um modelo formal para expressar processos executáveis e abstratos os quais contemplam todos aspectos de processos de negócios das empresas, incluindo atividades de complexidade variante, transações e suas compensações, gerenciamento de dados, concorrência, tratamento de exceções e semântica operacional. BPML também fornece uma gramática na forma de um XML Schema para habilitar a persistência e o intercâmbio de definições entre sistemas heterogêneos e ferramentas de modelagem.

A versão 1.0 da BPML trata especificamente com processos de negócios executáveis. A especificação BPML depende das seguintes especificações: XML 1.0, XML-Namespaces, XML-Schema 1.0 e XPath 1.0. Completando, o suporte para importação e referência a definições de serviços contida em WSDL 1.1 é uma parte normativa da especificação BPML [BPM 2002].

3.3.3 XPDL (*XML Process Definition Language*)

Em 25 de outubro de 2002, a *Workflow Management Coalition* publicou a versão 1.0 da XPDL, com o intuito de fornecer a especificação aos fornecedores de soluções que procuram implementar a XPDL.

Segundo [XPD 2002], a interface de definição de processos de fluxo de trabalho define uma formato de intercâmbio comum, o qual suporta a transferência de definições de processos entre produtos separados. Além disto, ela define uma separação formal entre o ambiente de desenvolvimento e o ambiente de execução, habilitando um definição de processo, gerada por uma ferramenta de modelagem, a ser usada como entrada em vários produtos executáveis de fluxo de trabalho diferentes. A XPDL foi estabelecida para atender à esta demanda e tem como elementos chave:

- extensibilidade para tratar informações usadas por uma variedade de diferentes ferramentas;
- é baseada em um número limitado de entidades as quais descrevem uma definição de processo de fluxo de trabalho, suportando diferentes visões.

Em [XPD 2002], uma definição de processo é definida como a representação de um processo de negócio de forma que suporte manipulação automatizada, assim como

modelagem ou ativação por sistema de gerenciamento de *workflow*. A definição de processo consiste de uma rede de atividades e suas responsabilidades, critérios para indicar o início e término de processo e informação sobre as atividades individuais, como os participantes, dados e aplicações de TI associadas, etc.

XPDL é uma linguagem para descrição de definições de processos de negócio, implementada através de um XML Schema para intercâmbio destas definições.

Um dos elementos chaves da XPDL é a extensibilidade para tratar informações utilizadas por uma variedade de diferentes ferramentas. A XPDL nunca poderá ser capaz de suportar todas os requisitos de informações adicionais em todas as ferramentas. Baseada em um número limitado de entidades que descrevem uma definição de processo de *workflow*, a XPDL suporta um número de diferentes visões [XPD 2002].

Cada elemento necessário para a modelagem de um processo de negócio é considerado uma entidade dentro da XPDL. Estes elementos possuem hierarquia e relacionamentos entre si.

A Tabela 3.1 apresenta uma visão geral sobre os principais elementos definidos dentro da XPDL, observando a seguinte disposição:

A primeira linha contém atributos e elementos comuns a todos os principais elementos. Percebe-se que os elementos *Id* e *Name* estão presentes em todos os elementos, assim como os elementos *Description* e *Extended Atributes*, que são opcionais;

A segunda linha contém propriedades específicas do respectivo elemento principal;

A terceira linha consiste em elementos que podem conter referências a outros elementos;

A quarta linha contém elementos de apresentação a serem utilizadas pela máquina de execução;

A quinta e última linha tem contém informações relevantes para simulação e otimização de processos.

Tabela 3.1: Elementos da XPDL

Package	WorkFlow Process	Activity	Transition	Application	Data Field (Workflow Relevant Data)	Participant
- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes	- Id - Name - Description - Extended Atributes
- XPDL Version - Source Vendor ID - Creation Date - Version - Author - Codepage - Country Key - Publication Status - Conformance Class - Priority Unit	- Creation Date - Version - Author - Codepage - Country Key - Publication Status - Priority - Limit - Valid from Date - Valid to Date	- Automation Mode - Split - Join - Priority - Limit - Start Mode - Finish Mode - Deadline			- Data Type	- Participant Types
- Responsible - External Package	- Parameters - Responsible	- Performer - Tool - Subflow - Actitivity Set - Actual Parameter	- Condition - From - To	- Parameters	- Initial Value	
- Documentation - Icon	- Documentation - Icon	- Documentation - Icon				
- Cost Unit	- Duration Unit - Duration - Waiting Time - Working Time	- Cost - Duration - Waiting Time - Working Time				

3.4 A integração de empresas, processos de negócio e sistemas

Modelar empresas, processos de negócio e sistemas de informação são atividades essenciais, mas integrar o processo de produção e comercialização das empresas através do uso de sistemas de informação que se comuniquem de forma ágil sempre foi abordado como um problema tecnológico.

Tem-se na atualidade um grande número de sistemas de informação, desenvolvido em várias plataformas de *hardware* diferente e usando tecnologias distintas. Integrá-las de forma eficaz e eficiente requer, e muitos casos, um profundo conhecimento de todos a modelagem da organização, dos processos e dos sistemas.

Como uma forma de possibilitar a integração sem expôr os processos internos, várias foram as tentativas de se criar protocolos que permitissem a comunicação entre aplicativos, ocultando seus detalhes de implementação. As empresas passaram a disponibilizar APIs para acesso a seus sistemas internos. Outras optavam por tecnologias como CORBA, *sockets* , RMI ou simplesmente transferência de documentos em formato XML, na tentativa de estabelecer um padrão de comunicação entre sua cadeia de fornecedores.

Estas alternativas mostravam-se eficiente em alguns casos, mas a necessidade de integrar sistemas em um ambiente totalmente heterogêneo influenciou o surgimento de uma nova forma de integrar sistemas de informação: o uso de *Web services.*

4 WEB SERVICES

A grande maioria dos sistemas de informação foram concebidos para serem executados em computadores isolados e em redes locais. Com advento da Internet e de seus protocolos de comunicação, vimos uma migração de tais aplicativos para a plataforma *Web* a fim de que vários clientes (usuários nos seus computadores pessoais) pudessem compartilhar as mesmas informações armazenadas num só lugar (Servidor *Web*). Assim, essas aplicações se tornam globalmente disponíveis através de um Servidor *Web*, e de sua UI (interface do usuário) através do *browser*.

Porém, as aplicações *Web* são fracas no aproveitamento da lógica da aplicação, quando desejam tornar disponível, pela *Internet*, essa lógica para vários clientes. Cada site tem sua própria ilha de informações e os usuários finais têm que navegar por várias URLs para encontrar as informações desejadas.

A tecnologia de *Web services* surge com uma proposta de incrementar a interoperabilidade dos sistemas, usando padrões abertos. A promessa desta tecnologia é integrar sistemas de informação, não importando a linguagem utilizada, plataforma de *hardware* ou *software*, bem como localização geográfica dos sistemas integrados. Trata-se de uma proposta que vislumbra a integração de sistemas legados com novos sistemas, já operando no ambiente *Web*.

Tais promessas têm gerado grande entusiasmo no mercado, que pode ser constatado não só por sua utilização em grandes companhias de comércio eletrônico, bem como pela preocupação demonstrada por vários comitês na busca de uma definição clara de padrões a serem utilizados.

4.1 Conceitos

Uma possível solução seria integrar as aplicações de um jeito simples e tornar a lógica dessas aplicações disponíveis para outros clientes. Esta integração pode ser programada através de componentes, que seriam responsáveis em obter informações distribuídas, que possam ser acessadas a partir dos protocolos da Internet, com a finalidade de comunicação entre as aplicações. Esses componentes, que são executados pelo Servidor *Web*, são chamados de serviços *Web* (em inglês, *Web services*).

Um *Web service* é um sistema de *software* projetado para suportar interações interoperáveis máquina-a-máquina através de uma rede. Ele possui uma interface descrita em um formato processável pela máquina (especificamente WSDL). Outros sistemas interagem com o *Web service* em uma maneira prescrita por sua descrição usando

mensagens SOAP, tipicamente transportadas usando HTTP com uma serialização XML em conjunção com outros padrões relacionados a *Web* [WSG 2004].

Os usuários de *Web services* não precisam saber nada sobre a plataforma, modelo de objeto, ou a linguagem de programação que foi usada para implementar o serviço; eles só precisam saber como solicitar o serviço para que possam receber a resposta da solicitação adequadamente.

Pode-se considerar os *Web services* como uma aplicação de software que pode ser acionada remotamente e que baseia-se no padrão XML para troca de mensagens.

O futuro dos sistemas baseados na *Web* será avaliado em função da capacidade das aplicações locais e da *Web* se comunicarem, aproveitando a Internet para oferecer um conjunto comum de serviços para vários clientes.

4.2 Arquitetura de Web services

O objetivo dos *Web services* é obter interoperabilidade entre aplicações através do uso de padrões *Web.*

A arquitetura de *Web services* é uma arquitetura de interoperabilidade: ela identifica aqueles elementos globais da rede de serviços *Web o*s quais são necessários para assegurar a interoperabilidade entre *Web services* [WSA 2004].

O funcionamento de um *Web service* presume que devam existir, no mínimo, dois agentes trocando informações (um solicitante do serviço e um provedor do serviço). Existem também agentes de descobrimento de *Web services* disponíveis na *Web*, que funcionam como verdadeiros catálogos de serviços disponíveis. A Figura 4.1 apresenta a arquitetura básica de *Web services*, onde percebe-se a interação que ocorre entre os agentes citados.

Ao solicitar um serviço, o solicitante interage com o provedor do serviço, que possui a aplicação propriamente dita. Eventualmente, o solicitante poderá procurar informações em uma das agências de descobrimento de serviços. O provedor de serviços disponibiliza ao cliente somente a descrição dos serviços e o serviço propriamente dito. Detalhes particulares da aplicação não são revelados. O provedor de serviços pode, ainda, publicar os serviços que disponibiliza em agências de descobrimento de serviços.

Alguns prestadores de serviço na Internet, como Google e Amazon, fornecem a descrição dos serviços disponibilizados de forma pública em seus *sites*, exigindo um cadastro prévio do usuário, que recebe um par identificação/senha.

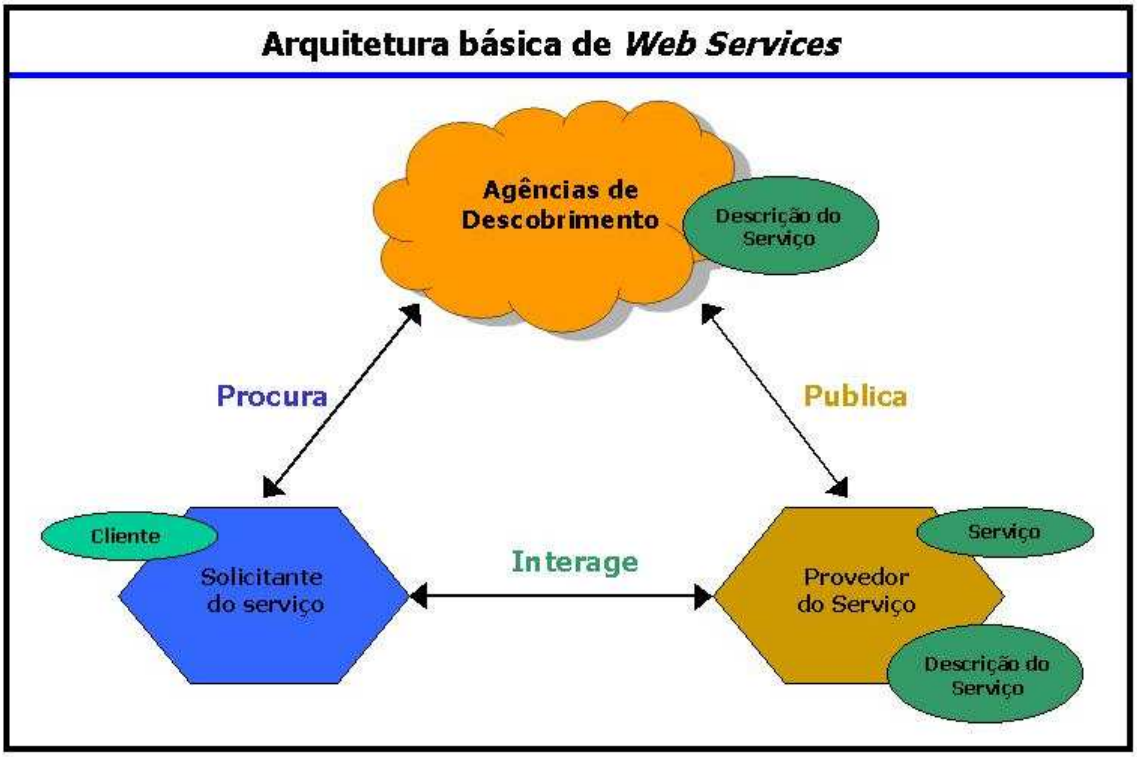

Figura 4.1: Arquitetura Básica de *Web services*

4.3 Protocolos de Comunicação de *Web services*

Todos os *Web services* usam protocolos *Web* (como HTTP e SOAP) e o formato XML para troca de mensagens [WSA 2004].

Os *Web services* são construídos originalmente a partir das seguintes especificações: SOAP, WSDL (*Web Service Description Language*) e UDDI (*Universal Description, Discovery, and Integration*).

SOAP define um protocolo XML para mensagens transportadas, WSDL introduz uma gramática comum para descrição de serviços; e UDDI fornece a infra-estrutura necessária para a publicação e descoberta de serviços de uma forma sistemática [BPE 2002].

Os protocolos de comunicação de *Web services*, conforme apresentados na Figura 4.2 , estão dispostos em 5 camadas, quais sejam:

- Camada de Transporte
- Camada de Mensagens
- Camada de Dados
- Camada de Descrição
- Camada de Descoberta

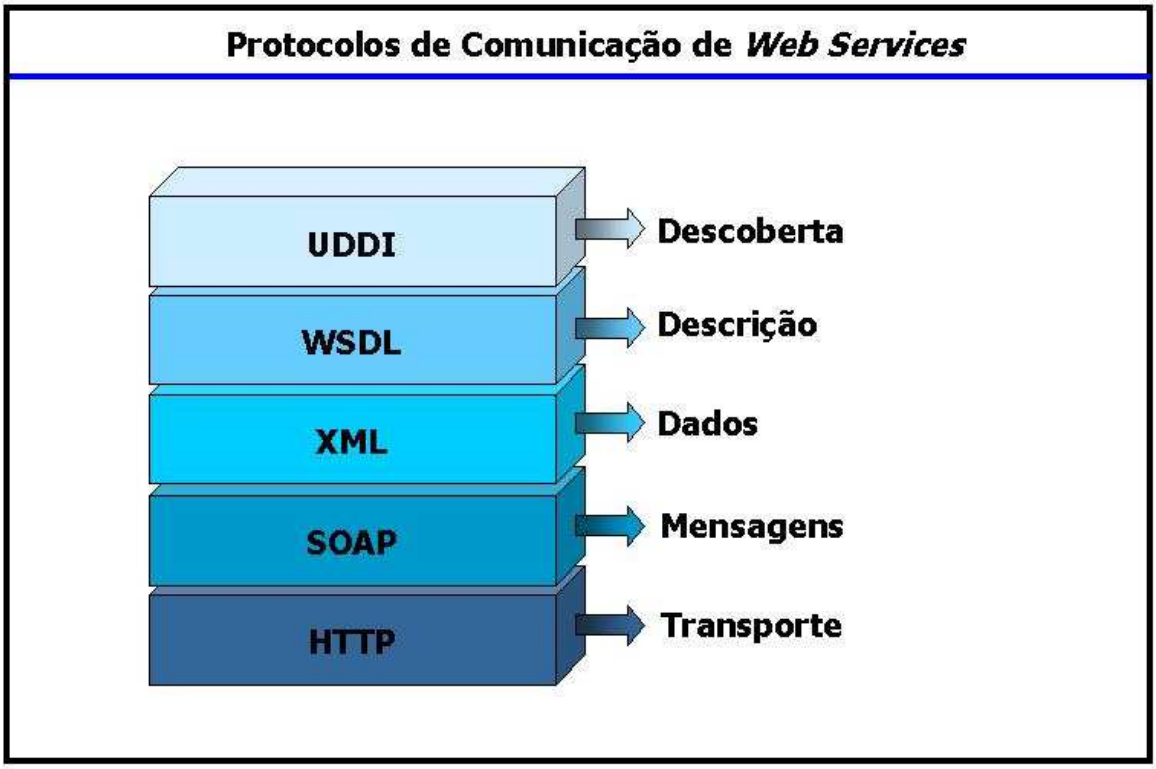

Figura 4.2: Protocolos de Comunicação de *Web services*

4.3.1 A camada de transporte - HTTP

A camada de transporte é a responsável pelo transporte das mensagens, normalmente sobre protocolo HTTP.

O protocolo HTTP (*Hypertext Transfer Protocol*) é um protocolo em nível de aplicação para sistemas de informação hipermídia, colaborativos e distribuídos. Ele é um protocolo genérico, que pode ser utilizado para várias tarefas, além de ser utilizado para hipertexto, como sistemas de gerenciamento de objetos distribuídos e servidores de nomes. Um recurso do protocolo HTTP é a tipificacação e negociação da representação de dados, permitindo ao sistemas serem construídos independentemente dos dados que estão sendo transferidos [HTT 2003].

O protocolo HTTP é o mais utilizado para comunicação entre *Web services*. No entanto, é possível realizar o transporte de mensagens através de outros protocolos, como SMTP, FTP e Jabber. O predomínio do uso de HTTP se dá por questões de segurança: o protocolo HTTP normalmente não é bloqueado por *firewalls* em equipamentos conectados à Internet.

Como os *Web services* normalmente trafegam sob HTTP utilizando POST e GET na porta 80, a base de funcionamento da *Web*, estas características permitem a comunicação entre as mais diferentes plataformas pois é comum a grande maioria delas.

4.3.2 A camada de mensagens - SOAP

A camada de mensagens é a camada responsável pela comunicação entre os objetos.

Os *Web services* estão apoiados em XML e no protocolo SOAP; o XML descreve e define os *Web services*, e, como os *Web services* são usados para disponibilizar serviços interativos na Internet, devem ser acessados por qualquer aplicação usando um protocolo universal; neste momento é que ocorre o uso de SOAP (*Simple Object Access Protocol*).

SOAP fornece um mecanismo leve e simples para troca de informações tipificadas e estruturadas entre pontos de uma ambiente descentralizado e distribuído, usando XML. SOAP por si só não define qualquer semântica de aplicação como um modelo de programação ou semântica específica de implantação; ao invés disto, define um simples mecanismo para expressar semânticas de aplicações fornecendo um modelo de pacotes modular em mecanismos de codificação para codificar dados entre módulos. Isto permite ao protocolo SOAP ser utilizado para uma grande variedade de sistemas, desde sistemas de troca de mensagens até RPC [SOA 2003].

Algumas das características do protocolo SOAP:

- definido pelo consórcio W3C;
- baseado em XML para intercâmbio de informações;
- padrão utilizado para acessar *Web services*;
- utiliza HTTP como protocolo de transporte;
- É constituído pelos seguintes elementos:
    - Envelope – Elemento raiz do documento XML
    - *Header* – Cabeçalho opcional que define um namespace para o elemento.
    - *Body* – Elemento obrigatório com a informação a ser transportada para o destino.

A Figura 4.3 apresenta a estrutura das mensagens SOAP.

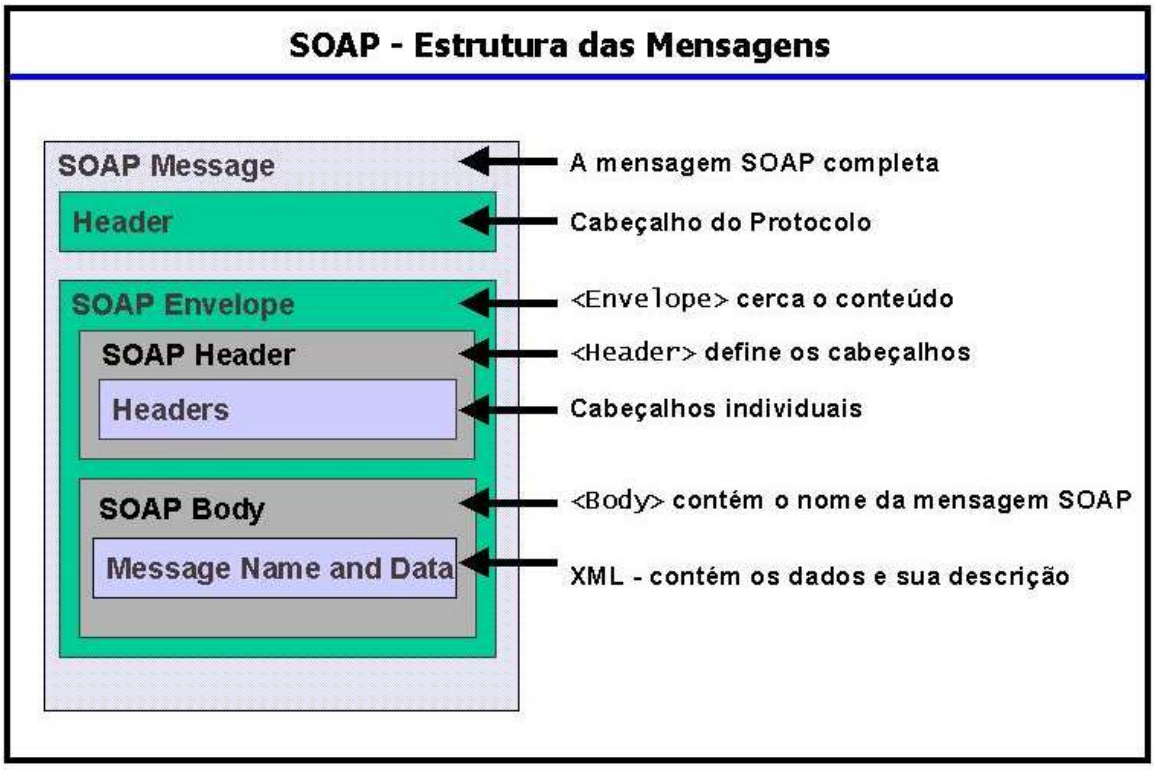

Figura 4.3 : Estrutura de Mensagens SOAP

SOAP permite a troca de informações entre aplicações de diferentes plataformas, por ser baseado em XML e independente de linguagem.

O protocolo SOAP possui elementos opcionais para tratamento de erros ocorridos durante o processamento das mensagens. Os erros são tipificados e o usuário pode criar suas próprias rotinas para descrição dos erros ocorridos.

Uma das opções para a integração de sistemas de informação através da *Web* é a utilização de RPC (*Remote Procedure Calls*). No entanto, isto gera um problema de segurança, visto que um sistema poderia fazer chamadas RPC a um servidor sem autorização, e muitas vezes estas chamadas são bloqueadas através de um *firewall*. Como o protocolo SOAP trafega encapsulado em no protocolo HTTP, as informações podem passar por *firewalls* existentes na Internet, permitindo a integração dos sistemas.

Uma mensagem SOAP é um documento XML que consiste de um envelope SOAP obrigatório, um cabeçalho SOAP opcional e um corpo de mensagem SOAP obrigatório [SOA 2003].

A Figura 4.4 apresenta um exemplo de mensagem SOAP, que envia uma requisição de chamada de um *Web service*.

```
POST /mestrado/despesas/ws/despesasWeb.php
HTTP/1.0 User-Agent: NuSOAP/0.6.3 Host: 192.168.1.1
Content-Type: text/xml;
charset=ISO-8859-1 Content-Length: 608
SOAPAction: "buscaDadosLancamento"
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:si="http://soapinterop.org/xsd"
xmlns:nu="urn:despesasweb_Service">

<SOAP-ENV:Body>
<nu:buscaDadosLancamento>
<login xsi:type="xsd:string">pjose</login>
<senha xsi:type="xsd:string">teste1</senha>
<codigo_lancamento xsi:type="xsd:integer">5</codigo_lancamento>
</nu:buscaDadosLancamento>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Figura 4.4: Exemplo de mensagem SOAP

4.3.3 A camada de dados - XML

A camada de dados ou serviços é responsável pelas informações trocadas entre as demais camadas. Em *Web services*, a camada de dados é descrita em formato XML.

As informações trafegadas em um pacote SOAP podem ser de dois estilos diferentes: RPC (*Remote Procedure Call*) ou *document.*

- RPC: RPC indica que os corpo da mensagem SOAP contém um elemento com o nome do método ou procedimento remoto que está sendo invocado. Este elemento internamente contém um elemento para cada parâmetro deste procedimento.
- *Document*: indica que o corpo da mensagem SOAP contém um ou mais elementos filhos chamados **partes**. Não existem regras de formatação SOAP para o conteúdo do corpo da mensagem SOAP; ela pode conter qualquer coisa que o transmissor e o receptor concordam.

4.3.4 A camada de descrição - WSDL

A camada de descrição de serviços é implementada através de instruções em WSDL e descrevem os serviços disponibilizados, bem como as trocas de mensagens possíveis.

WSDL *(Web Services Description Language)* fornece um modelo e um formato XML para descrição de serviços *Web*. WSDL possibilita uma separação da descrição das funcionalidades abstratas oferecidas por um serviço dos detalhes concretos da descrição de um serviço, como "onde" e "como" as funcionalidades são oferecidas [WSD 2003].

Com o uso de WSDL, o cliente não precisa saber qual a linguagem de programação ou plataforma de execução em que o provedor de serviços está baseado. Há uma camada de descrição de serviços que omite detalhes concretos de implementação tanto por parte do servidor como do cliente da aplicação.

As operações e mensagens de um serviço são descritas de uma forma abstrata e em seguida ligados a protocolos de rede e formatos de mensagens concretos com o objetivo de definir um ponto de serviço.

Existem vários tipos de operações que podem ser declarados em um documento WSDL. As operações de comunicação possíveis para os *Web services* são:

- **Requisição/Resposta**: um cliente faz uma requisição de serviço e o *Web service* responde a esta requisição;
- **Solicitação/Resposta**: um *Web service* envia uma mensagem ao cliente e o cliente responde;
- **Sentido único**: um cliente envia uma mensagem ao *Web service* mas não espera uma resposta;
- **Notificação**: um *Web service* envia uma mensagem ao cliente mas não espera uma resposta.

A época da elaboração deste trabalho, já estava em estudos a criação da versão 2.0 da WSDL.

Escrever interfaces de sistemas a partir de especificações em WSDL requer um bom conhecimento do padrão. Para facilitar esta tarefa, existem no mercado ferramentas que automatizam este processo, como o Java Web Service Developer Pack (disponível em http://java.sun.com) . Cita-se, por exemplo, a ferramenta `wscompile`, que permite a criação de uma classe de interface Java a partir de uma especificação WSDL.

4.3.5 A camada de descoberta - UDDI

A camada de descoberta permite o registro de *Web services*, através de UDDI e ebXML.

O protocolo UDDI (*Universal Description, Discovery, and Integration*) cria um padrão com interoperabilidade entre plataformas que habilita companhias e aplicações a encontrar e usar *Web services* na Internet, de forma rápida, fácil e dinâmica. UDDI também permite que registros operacionais sejam mantidos para diferentes propósitos em diferentes contextos [UDD 2003].

Universal Description, Discorevy, and Integration (UDDI) é uma especificação de uma API baseada em XML para publicação e busca de operações com os quais requisitantes de serviços e provedores de serviços podem acessar *brokers* de serviços [MAR 2002].

O UDDI é um diretório universal de registro de *Web services*, onde são encontradas informações sobre empresas fornecedoras de serviços, descritos em formato XML.

Existem vários repositórios de *Web services* na Internet, visto que qualquer usuário pode disponibilizar um serviço a partir de seu servidor *Web*.

O repositório UDDI para *Web services* padrão é o UDDI.org (http://www.uddi.org), e exige o registro de usuário para disponibilização e acesso aos serviços cadastrados.

Os registros UDDI podem ser de vários tipos, quais sejam:

- **Públicos**: são registros que permitem acesso por qualquer usuário/empresa;
- **Privados**: são registros de uma mesma corporação, que são catalogados como uma forma de catalogar os *softwares* da organização sem expô-los a outras corporações;
- **Restritos**: são registros que podem ser acessados somente por certas organizações que mantém interesses de negócio e que têm permissão para acessá-los.

Uma característica interessante do registro UDDI é a forma como estão distribuídos os registros UDDI. Algumas grandes corporações (como IBM e Microsoft) possuem sites na Internet que permitem o registro UDDI. Todas as informações sobre os registros UDDI são sincronizadas entre os diferentes servidores UDDI. Desta forma, basta o usuário registrar em um único servidor UDDI que todos os demais também estarão disponibilizando o registro.

O registro UDDI pode ser considerado um grande catálogo de *Web services*, No entanto, os repositórios mais conhecidos são o XMethods (http://www.xmethods.net) e o Salcentral (http://www.salcentral.com). Lá, o usuário poderá encontrar uma séria de serviços Web que são disponibilizados através de SOAP e XML-RPC.

Algumas empresas disponibilizam APIs específicas para acessos a seus *Web services* , como o site Google (http://www.google.com) e Amazon (http://www.amazon.com). Para que se possa utilizar os serviços, o usuário deve estar registrado na empresa, que lhe fornece uma chave de acesso, que é incorporada a mensagem SOAP enviada ao servidor.

Se o usuário tem acesso à especificação WSDL de um serviço, ele não precisará acessar o registro UDDI para obter informações sobre como utilizar o *Web service*.

Diferente de muitas outras especificações de *Web services*, a UDDI não é administrada pelo W3C, mas por um grupo chamado *Organization for the Advancement of Structured Information Standards (OASIS)*.

4.4 Benefícios

O uso de *Web services* vem recebendo especial atenção nos últimos anos devido a vários fatores, quais sejam:

Facilidade de Integração

A utilização de *Web services* traz grandes benefícios para os desenvolvedores de sistemas, facilitando a integração de diferentes sistemas através de um protocolo de comunicação de alto nível.

Pode-se, com o uso de *Web services*, desenvolver módulos que disponibilizem através da Internet acesso a sistemas legados desenvolvidos em praticamente qualquer tecnologia. A criação de *interfaces* entre sistemas legados usando os *Web services* pode proporcionar uma integração mais rápida, com menor esforço de desenvolvimento.

Segurança

Ao implementar *Web services*, o desenvolvedor cria uma camada de acesso aos serviços da aplicação, mantendo sigilo sobre a forma como os métodos foram implementados na aplicação. A aplicação cliente não precisa conhecer detalhes do funcionamento interno de um *Web service*, apenas a descrição dos métodos disponibilizados, suas assinaturas e parâmetros. Desta forma, o fornecedor do serviço não expõe o código desenvolvido nem a lógica do domínio do problema.

Portabilidade

Um cliente de *Web service* não precisa necessariamente estar utilizando a mesma tecnologia que o fornecedor de serviço. O fornecedor do serviço disponibiliza a descrição dos métodos em WSDL, e não importa qual o sistema operacional, linguagem de programação ou equipamento de onde está sendo solicitado o serviço. A única exigência é que a comunicação siga os padrões estabelecidos entre as partes.

Extensibilidade

Os *Web services* são extensíveis, à medida que permitem ao desenvolvedor acrescentar novas funcionalidades assim que se tornarem necessárias.

Conectividade

Algumas soluções adotadas para integração de sistemas (como *sockets*, RMI e RPC), utilizam conexões persistentes entre o servidor e o cliente da aplicação. Tais soluções são chamadas de soluções invasivas, pois consomem recursos de ambos os usuários mesmo que não estejam realizando qualquer tarefa. Os *Web services*, por sua vez, são sistemas não invasivos, que são acionados somente quando há necessidade de utilização do serviço. Outro aspecto importante a se destacar é o fato dos *Web services* normalmente trafegam sob HTTP. Então, estas requisições não exigem a liberação de portas de comunicação adicionais em *firewalls* de rede.

Integração com parceiros externos da organização

O mundo globalizado exige das empresas novas competências e posturas. Existe uma grande quantidade de empresas que disponibilizam produtos e serviços na Internet. O uso dos *Web services* permite uma integração maior entre estas empresas, reduzindo o gargalo da comunicação. Por exemplo, cita-se o caso de uma empresa que compra componentes de uma outra empresa. Se ela tem que usar via Internet o sistema da empresa fornecedora, com o uso de *Web services* ela poderia incorporar aos seus sistemas existentes clientes de *Web services* que realizassem de forma automática esta atividade. Há, portanto, uma maior integração da organização com seus fornecedores e clientes.

4.5 Desvantagens

Ao contrário do que se possa pensar em um primeiro momento, o emprego de *Web services* também traz consigo algumas desvantagens. Como toda tecnologia recente,alguns problemas são encontrados, quais sejam:

Persistência

Quando um cliente de *Web service* faz uma requisição de chamada de um método, não há mecanismos que garantam que esta requisição será encaminhada. Poderão ocorrer problemas como falta de energia no servidor, queda da linha de comunicação ou falha em algum nodo de roteamento da rede e esta requisição (ou o seu resultado) nunca chegará ao seu destino.

Segurança

As mensagens que trafegam entre um fornecedor de *Web service* e seu cliente são transportadas em formato XML, usando o protocolo SOAP. Isto significa que estas informações não estão devidamente criptografadas e poderiam ser interceptadas por sistemas não autorizados. Para solucionar este problema, estão sendo propostos padrões (como XML-Signature, XML-Encryption e WS-Security) para segurança das informações transportadas.

Autenticação

Os *Web services* não possuem um mecanismo de autenticação de comunicação. Normalmente, os fornecedores de *Web services* que não são de uso gratuito disponibilizam

uma identificação e senha de acesso para o cliente do serviço. Com isto, a cada comunicação, faz-se necessário o envio destas informações.

Diversos Padrões

A tecnologia de *Web services* está construída sobre diversos padrões. Qualquer modificação em um destes padrões poderá fazer com que um *Web service* deixe de funcionar.

Da mesma forma, a evolução destes padrões tende a criar versões mais completas das especificações, o que provavelmente criaria "versões" de *Web services* diferentes. Para resolver este problema, foi criado a *Web Services Interoperability Organization* (www.ws-i.org), que conta com a associação de empresas com IBM, Microsoft, Oracle, BEA, HP e Intel. Esta organização tem como objetivo acelerar a adoção dos *Web services*, auxiliando na seleção e interpretação das especificações, bem como produzindo perfis, ferramentas e aplicações-modelo para *Web services*.

5 ESTUDO DE CASO - IMPLEMENTANDO WEB SERVICES

Para uma melhor compreensão dos tópicos apresentados anteriormente, apresenta-se um estudo de caso da implementação de *Web services* para um sistema *Web* já existente, o Sistema Despesas *Web*.

O desenvolvimento dos *Web services* para o Sistema Despesas *Web* foi feito com o objetivo de exercitar os conceitos aprendidos ao longo do trabalho, bem como elencar as principais dificuldades e situações peculiares no emprego desta tecnologia. O Sistema Despesas *Web*

O sistema Despesas *Web* é um software livre que opera em ambiente *Web* e encontra-se disponível para *download* em http://www.apoenasoftwarelivre.com.br . Ele foi desenvolvido com a finalidade de disponibilizar para a sociedade um sistema *Web* de estrutura simples, de f' ácil utilização e uso gratuito. Não há qualquer tipo de cobrança ou envio de mensagens publicitárias aos seus usuários.

O sistema Despesas *Web* está disponível para uso por qualquer usuário no site http://www.despesas.com.br .

A finalidade do software é permitir a qualquer usuário controlar suas despesas pessoais, registrando as informações no banco de dados do sistema, tornando a informação acessível através de qualquer *browser*.

O sistema é executado totalmente via Internet, sem a necessidade de instalação de qualquer módulo cliente ou *plugin* de *browser* para sua utilização.

5.1 Funcionalidades

Uma das características do sistema é que a operação é conduzida totalmente pelo usuário. Não há qualquer necessidade de solicitação prévia de cadastro e/ou senha para operar o sistema. O próprio usuário pode acessar o sistema, cadastrar seus dados, senha de acesso, contas de controle e lançamentos.Ao acessar o sistema, o usuário é saudado com a tela inicial do sistema, onde poderá cadastrar-se ou acessar sua conta, caso já esteja cadastrado no sistema, conforme exibido na Figura 5.1.

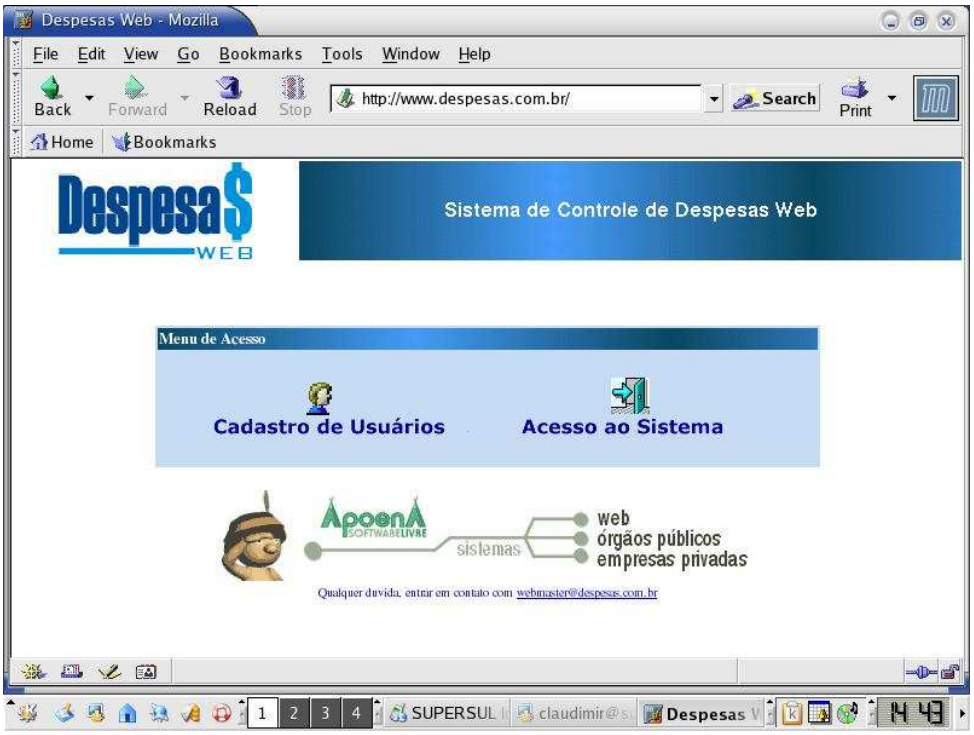

Figura 5.1: Tela inicial do sistema Despesas *Web*

Ao realizar seu cadastro, o sistema solicita que o usuário informe seu nome, *login* de acesso, senha e e-mail para contato. Uma vez cadastrado no sistema, o usuário poderá passar a usá-lo imediatamente, através da opção "Acesso ao sistema", a qual irá lhe proporcionar acesso ao menu de opções apresentado na Figura 5.2.

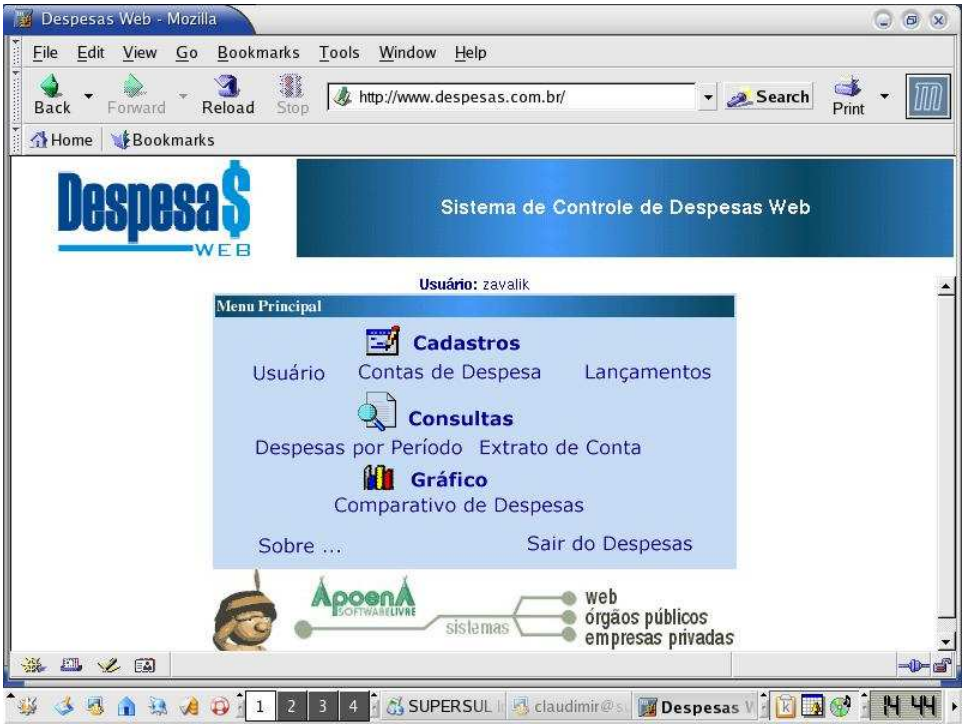

Figura 5.2: Menu Principal do Sistema Despesas *Web*

As contas de despesa referenciadas pelo sistema são aquelas que serão utilizadas para controlar as despesas efetuadas. Cada usuário do sistema tem seu próprio cadastro de contas de despesa.

Uma vez cadastradas as contas de despesa do usuário, este poderá digitar no sistema os lançamentos feitos (despesas pagas), sempre indicando a qual conta de despesa estão vinculados, conforme apresentado na Figura 5.3.

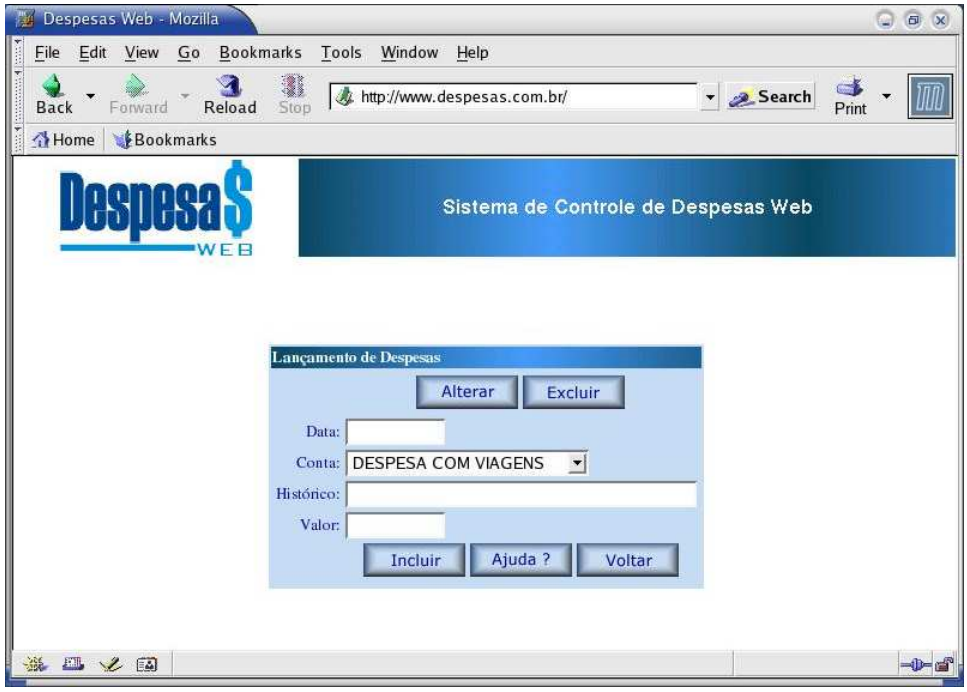

Figura 5.3: Tela de Digitação de Lançamentos

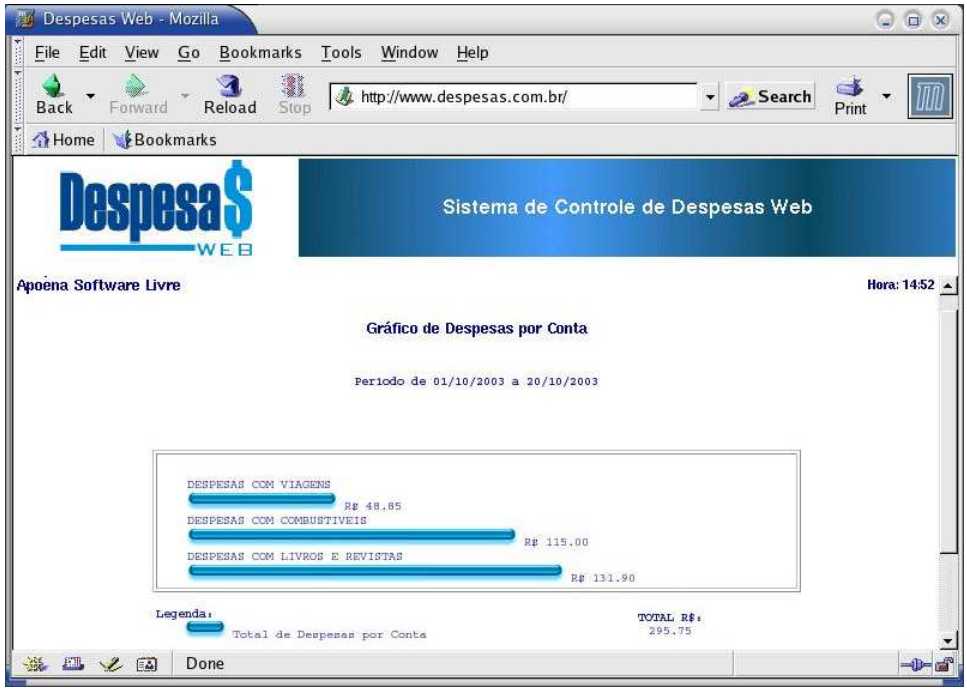

Figura 5.4: Gráfico Comparativo gerado pelo Sistema Despesas *Web*

O usuário poderá digitar suas despesas pessoais e, a partir destas informações, obter consultas, relatórios e gráficos baseados nas informações armazenadas. A Figura 5.4 apresenta o gráfico comparativo de despesas gerado pelo sistema.

5.2 Características técnicas

O sistema Despesas *Web* foi desenvolvido sob plataforma de software livre, usando as seguintes ferramentas:

- Sistema operacional GNU/Linux;
- Servidor Web Apache;
- Linguagem de programação PHP;
- Banco de dados MySQL.

Sua implementação foi feita usando o paradigma da Orientação a Objetos e a UML como linguagem de modelagem.

5.3 *Web services* a serem disponibilizados

Ao disponibilizar *Web services* publicamente, há a necessidade de se avaliar quais as funcionalidades da aplicação as quais se deseja autorizar o acesso.

Com o uso de *Web services*, pode-se disponibilizar apenas a interface para acesso a métodos do sistema, sem contudo expôr as regras de negócio e lógica de programação empregada para sua implementação.

Tendo como objetivo permitir o uso das principais funcionalidades do sistema Despesas *Web*, constatou-se a necessidade de disponibilizar os seguintes serviços:

Tabela 5.1: *Web services* do Sistema Despesas *Web*

Serviço	Parâmetros de Entrada	Parâmetros Retornados
insereUsuario	Nome: string Login: string Senha: string Email: string	Codigo: integer
alteraUsuario	Login: string Senha: string Novonome: string Novologin: string Novasenha: string Novoemail: string	Resposta: boolean
excluiUsuario	Login: string Senha: string	Resposta: boolean
buscaCodigoUsuario	Login: string Senha: string	Codigo: integer
insereContaDespesa	Login: string Senha: string Descricao: string	Codigo integer
alteraContaDespesa	Login: string Senha: string Descricao: string Novadescricao: string	Resposta: boolean
excluiContaDespesa	Login: string Senha: string Descricao: string	Resposta: boolean
enviaCodigoConta	Login: string Senha: string Descricao: string	Codigo:integer
insereLancamento	Codigo_usuario: integer Codigo_conta: integer Data: string Historico: string Valor: float	Codigo: integer
alteraLancamento	Login: string Senha: string Codigo_lancamento: integer Codigo_conta: integer Data: string Historico: string Valor: float	Resposta: boolean
excluiLancamento	Login: string Senha: string Codigo_lancamento: string	Resposta: boolean
buscaDadosLancamento	Login: string Senha: string Codigo_lancamento: string	Codigo_lancamento: integer Codigo_conta: integer Data: string Historico: string Valor: float
buscaLancamentosPeriodo	Login: string Senha: string Data_inicial: string Data_final: string	ArrayCodigos: array[] of integer

Observa-se claramente que, em todos os *Web services* apresentados na Tabela 5.1 são passados como parâmetros o par *login*/senha. Isto se faz necessário para que se possa identificar qual o cliente que está solicitando o serviço.

5.4 Implementação dos *Web services*

Para a implementação dos *Web services* do sistema Despesas *Web*, poderiam ser utilizadas várias linguagens de programação diferentes.

Além de linguagens já tradicionais (como Perl, Java, PHP, C++), existem também ferramentas que auxiliam esta tarefa, como o Apache Axis, JWSDP (*Java Web Services Developer Pack)*, GLUE, PEAR SOAP e outros.

No entanto, devido ao sistema ser totalmente desenvolvido em linguagem PHP, bem como a disponibilidade de biblioteca de classes que auxilia a implementação, optou-se por desenvolver os módulos *Web* também em linguagem PHP, fazendo uso da biblioteca NuSOAP.

NuSOAP é uma biblioteca de classes para desenvolvimento e utilização de *Web services*, desenvolvido em PHP e distribuído sob os termos da licença pública GNU/GPL [NUS 2003].

A seguir, descreve-se as etapas de desenvolvimento dos *Web services*, exemplificando sempre com o processo de implantação da rotina de inserção de usuários.

Primeira Etapa - Criação do método concreto

Embora não seja absolutamente necessário a criação do método concreto em um primeiro momento, optou-se por escrever estes métodos para uma melhor compreensão do funcionamento da arquitetura do sistema.

Para cada *Web service* disponibilizado, foi criado um respectiva função em linguagem PHP. Para facilitar o desenvolvimento, com uma reutilização maior dos métodos, os *Web services* foram desenvolvidos em um único arquivo (despesasWeb.php).

O código fonte completo dos *Web services* apresentados apresenta-se no Anexo B - Código fonte dos *Web services*. Versões atualizadas posteriormente a edição deste trabalho se encontrarão a disposição no site da empresa que mantém o sistema.

A seguir, descreve-se as rotinas desenvolvidas, ilustrando as explicações com fragmentos do código anexado, com o intuito de possibilitar uma melhor compreensão.

No início da aplicação PHP, onde é incorporado a referência à biblioteca NuSOAP. Como pode-se observar no código da aplicação, é a única referência a biblioteca que deve ser incorporada ao código do *Web service*. Embora não esteja explicitamente documentada no site do desenvolvedor, a biblioteca NuSOAP exige a instalação da versão 4.0.4 ou superior da linguagem PHP.

```
<?php
require_once('nusoap.php');
$s = new soap_server;
$s->register('insereUsuario');
```

Logo a seguir, a aplicação cria um novo objeto do tipo **`soap_server`**. A classe de objetos **`soap_server`** define todos os atributos e métodos genéricos para o servidor de mensagens SOAP.

Após a instanciação do servidor SOAP, a aplicação chama o método `register` para registrar os métodos que serão disponibilizados. O usuário pode optar entre desenvolver um módulo para cada *Web service* ou criar um módulo contendo a especificação de todos os *Web services* desejados.

A seguir, apresenta-se o fragmento de código que implementa o *Web service* `insereUsuario`.

```
function insereUsuario($nome, $login, $senha, $email){

        // Validacao dos parametros passados
        //----------------------------------
        if($nome  == '' ){
           return new soap_fault('Client','','Voce deve fornecer o nome do usuario.');
        }
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($email == '' ){
           return new soap_fault('Client','','Voce deve fornecer o email do usuario.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario ja existe, impedindo sua inclusao
        //--------------------------------------------------------
        $instrucao   =   "select   count(*)   from   usuario   where   login   =   '$login'   and
senha='$senha';";
        $resultado = @mysql_db_query("despesas",$instrucao,$banco);
        $quantidade= @mysql_result( $resultado, 0, 0);
        if ($quantidade > 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario JA Cadastrado.');
        }

        // Monta a query a ser executada
        //------------------------------
        $instrucao   =   "insert   into   usuario   (codigo,nome,login,senha,e_mail)   values
(0,'$nome','$login', '$senha', '$email');";
        $resultado = @mysql_db_query( "despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return  new  soap_fault('Server','','Nao  conseguiu  inserir  os  dados  no  banco  de
dados');
        }

        // Como o codigo  auto_increment, tem que buscar o codigo do usuario inserido
        //----------------------------------------------------------------------------
        $instrucao          =  "select  codigo  from  usuario  where  login  =  '$login'  and
senha='$senha';";
        $resultado     = @mysql_db_query("despesas",$instrucao,$banco);
        $codigousuario = @mysql_result( $resultado, 0, 0);

        Fecha_banco( $banco );

        return $codigousuario;
}
```

Percebe-se claramente que a primeira ação tomada pelo método é a validação dos parâmetros passados ao *Web service*. Conforme citado anteriormente, o SOAP permite o tratamento de erros. Neste caso, são relacionados os erros que podem ocorrer por falta de

parâmetros. Todas as validações de parâmetros são feitas pela aplicação, e caso um dos parâmetros estiver incorreto, o sistema retorna uma mensagem SOAP de falha.

Ao encontrar uma falha na aplicação, esta deverá retornar uma falha SOAP, que é qualificada como do tipo *Server* ou *Client* sendo passado também uma descrição do problema ocorrido. O cliente do *Web service*, por sua vez, deverá sempre fazer o tratamento de exceções quando invocar um serviço.

Ao final do método, encontra-se o comando

```
return $codigoUsuario;
```

Este é o retorno do método `insereUsuario`. Ao inserir um usuário no sistema, a aplicação cliente do *Web service* receberá a informação do código que foi atribuído ao usuário.

Segunda Etapa - Criação da especificação WSDL

Uma vez implementados os métodos, há a necessidade de se criar uma interface em alto nível que garanta a portabilidade do uso dos métodos em diferentes plataformas e linguagens de programação. Esta portabilidade é garantida com a definição das mensagens com WSDL. Qualquer aplicação poderá usar os serviços disponibilizados, sabendo somente a localização do arquivo WSDL.

A especificação WSDL dos *Web services* do sistema Despesas *Web* encontra-se no Anexo A - WSDL do Sistema Despesa *Web*.

Ao relacionar as mensagens trocadas, pode-se reutilizar definições de mensagens usadas em diferentes métodos, reduzindo a complexidade da definição WSDL, bem como a redundância de definições.

```
<message name="recebeDadosUsuario">
    <part name="nome"  type="xsd:string"/>
    <part name="login" type="xsd:string"/>
    <part name="senha" type="xsd:string"/>
    <part name="email" type="xsd:string"/>
</message>
<message name="enviaRespostaUsuario">
    <part name="codigo" type="xsd:integer"/>
</message>
```

A especificação acima apresenta a definição das mensagens `recebeDadosUsuario` e `enviaRespostaUsuario`, que são utilizadas pelo método `insereUsuario.` Elas são utilizadas para comunicação entre o *Web service* e a aplicação cliente, descrevendo os parâmetros recebidos e enviados.

Em muitos casos, uma mensagem de resposta pode conter um tipo de dado referenciado em mais de um *Web service* na mesma definição WSDL. Nestes casos, elas podem ser reutilizadas, como acontece com a mensagem `enviaOkUsuario` usada várias vezes na definição WSDL do estudo de caso analisado.

Após definidas as mensagens que serão enviadas e recebidas pelos *Web services*, são então relacionadas quais as relações entre elas, e quais métodos irão utilizar estas mensagens.

Através de uma definição `portType`, são identificados os parâmetros de entrada e os valores retornados por um *Web service.*

```
  <portType name="despesasWeb_PortType">
        <operation name="insereUsuario">
           <input message="tns:recebeDadosUsuario"/>
           <output message="tns:enviaRespostaUsuario"/>
        </operation>

        ...

  </portType>
```

Em seu elemento `binding` , o documento WSDL descreve um protocolo concreto e uma especificação de formato de dados para um `PortType` particular [WSD 2003].

```
<binding name="despesasWeb_Binding" type="tns:despesasWeb_PortType">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="insereUsuario">
         <soap:operation soapAction="insereUsuario"/>
         <input>
            <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
         </input>
         <output>
            <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
         </output>
      </operation>
      ...
</binding>
```

Ao final da definição WSDL, tem-se a referência ao programa que executa os *Web services.* É importante destacar que, usando uma definição WSDL, o cliente do serviço não precisa conhecer detalhes de implementação, como o nome do programa executor.

No caso de uso apresentado, tem-se a seguinte definição do elemento `service,` conforme apresentado a seguir.

```
<service name="despesasWeb_Service">
     <port binding="tns:despesasWeb_Binding" name="despesasWeb_Port">
         <soap:address
            location="http://www.despesas.com.br/ws/despesasWeb.php"/>
      </port>
   </service>
```

Terceira Etapa - Implementação do Cliente SOAP

Uma vez desenvolvidos os *Web services,* procurou-se testar os módulos. Então, foram criados clientes SOAP que permitisse acessar estes *Web services*, com a finalidade de validar seu funcionamento. Estes clientes de serviço poderiam ser escritos em qualquer linguagem de programação e executados em qualquer plataforma de *hardware* e sistema operacional. Com *Web services* é possível a criação de interfaces gráficas para sistemas

Web. A seguir, apresenta-se uma aplicação cliente SOAP escrito para teste do *Web service* `insereUsuario`, desenvolvido também em PHP.

```
<?php

// Modulo de Teste do Web service insereUsuario
// ------------------------------------------

require_once('nusoap.php');
$soapclient = new    soapclient (
                'http://www.despesas.com.br/ws/despesasWeb.wsdl' , true );
$proxy = $soapclient->getProxy();
if ($err = $proxy->getError()) {
   print "ERRO: $err";
}

echo     "<center><h1>Web   services   com   WSDL   do   Sistema   Despesas
Web<br></h1><h2> TESTE <br></h2>";

// Método insereUsuario( $nome, $login, $senha, $email )
$resultado    =    $proxy->insereUsuario('Pedro    Jose    da    Silva',
'pjose','teste1', 'pjose@bol.com.br');

// Para ver as mensagens SOAP enviadas e recebidas, descomente as linhas
abaixo
// echo "<xmp>". $proxy->request."</xmp>";
// echo "<xmp>". $proxy->response."</xmp>";

if ($proxy->fault) {
   echo "<br>Ocorreu uma falha no sistema = <b>" . $proxy->faultstring .
"</b>";
}
else
{
   echo "<br>O resultado obtido do Web service foi <b>" . $resultado .
"</b>";
}

echo "<br><form action=\"javascript:back();\">";
echo "<br><input type=\"submit\" value=\"Voltar\">";
?>
```

Pode-se observar claramente a preocupação que a aplicação cliente deve ter com o tratamento de exceções ocorridas durante a execução do *Web service*.

Quarta Etapa - O registro UDDI

Após implementados e devidamente testados os *Web services*, procurou-se realizar o registro UDDI destes, permitindo então, que outras pessoas pudessem utilizá-los.

Ao publicar um *Web service* em um registro UDDI, o usuário pode optar entre os *sites* operadores UDDI disponíveis. Não há a necessidade de registrar em todos os provedores. As informações sobre serviços catalogados em *sites* operadores são replicadas entre si. Trata-se de um serviço fisicamente distribuído, mas logicamente centralizado.

Os *Web services* desenvolvidos para o sistema Despesa *Web* foram catalogados no *site* www.ibm.uddi.org.

O processo de registro UDDI é bastante simples. Ao acessar o registro UDDI, o usuário preenche uma série de formulários que descrevem o serviço disponibilizado.

O uso de UDDI permite que as aplicações clientes descubram *Web services* em tempo de execução, fazendo a consulta ao catálogo e identificando os *Web services* necessários. Esta possibilidade, além de trazer uma visão futurística para os sistemas de informação, também trás consigo dúvidas sobre como serão os mecanismos adotados para garantir a confiabilidade do serviço escolhido. Em um mundo totalmente conectado, não seria difícil a alguns *hackers* criarem *Web services* com outras finalidades e publicarem no registro UDDI.

6 O TRATAMENTO DE EXCEÇÕES EM *WEB SERVICES*

Como observado no estudo de caso apresentado, para cada *Web service* desenvolvido, há a necessidade de realizar todo o tratamento de exceções.

Os erros durante a execução de um *Web service* ocorrem essencialmente três motivos:

- erro na passagem de parâmetros (*Client*);
- erro de processamento do *Web service* (*Server*);
- falha na comunicação entre o servidor e o cliente (problemas de rede).

Neste último caso, a única alternativa para validar uma transação é criar mecanismos no cliente de *Web service* que estipulem um tempo máximo que o cliente irá aguardar por um retorno do servidor.

A seguir, apresenta-se uma proposta para modelagem de classes em sistemas de informação que utilizem *Web services.* Tal proposta visa permitir um melhor controle sobre as exceções ocorridas na operação do sistema, mantendo um histórico destas ocorrências no fornecedor do serviço.

Para cada classe que implementa *Web services*, seria estendida uma superclasse `WebService`, a qual manteria as propriedades das falhas ocorridas no sistema, conforme apresentado na Figura 6.1.

Figura 6.1: Classe *WebService*

Ao ocorrer uma exceção no sistema, a superclasse `WebService` seria a responsável pelo retorno da falha SOAP ocorrida para o cliente, e, através do método `gravaFalhaSOAP` , iria manter um histórico de todas as falhas ocorridas. Obviamente, este histórico seria armazenado no sistema servidor.

Com isso, ganha-se produtividade ao desenvolver o código da aplicação, reduzindo a redundância de código.

A Figura 6.2 apresenta-se um Diagrama de Classes exemplificando o uso da superclasse `WebService` em conjunto com a classe `Usuario.`

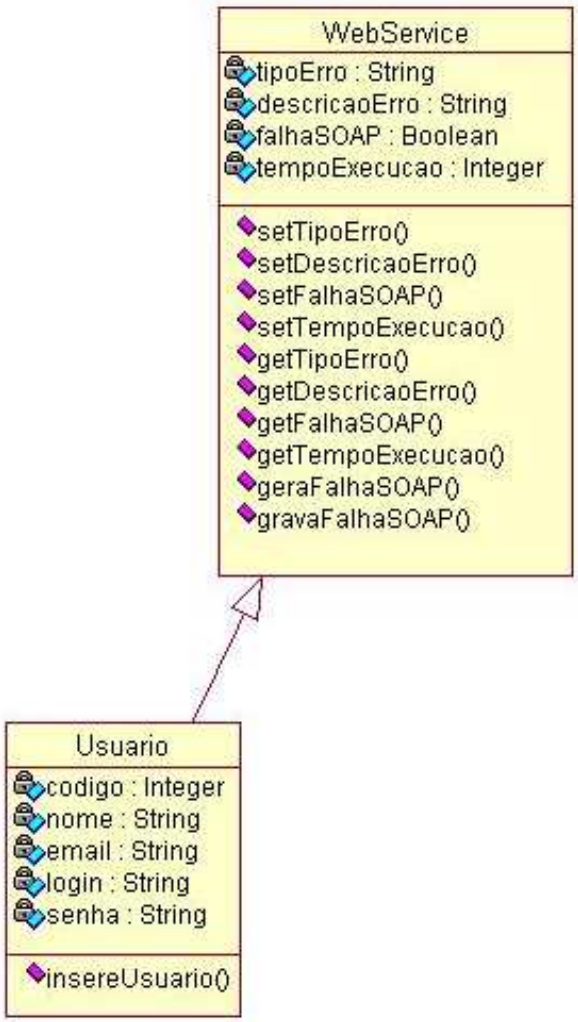

Figura 6.2: Exemplo de uso da superclasse WebService

Como se pode observar, com este modelo se repassa a responsabilidade sobre o tratamento de exceções para a superclasse `WebService`, aumentando o reuso do sistema e centralizando o controle de falhas.

A modelo proposto é extremamente simples e se baseia simplesmente no conceito de herança.

O fato que se procura enfatizar neste estudo é a necessidade de se fazer esta modelagem prévia nos sistemas de informação antes de seu desenvolvimento.

Para tanto, o desenvolvedor deverá identificar quais as classes do sistema que terão *Web services* disponibilizados. Ainda cabe uma observação que, embora se esteja propondo a adoção do paradigma da orientação a objetos na modelagem dos *Web services*, isto não é uma obrigatoriedade e nem todos os sistemas legados foram desenvolvidos sobre este paradigma. Os *Web services*, então, podem ser desenvolvidos como *proxies* para os sistemas internos das organizações, atendendo também a finalidade de registro de histórico de solicitações, respostas e falhas de comunicação.

Mais que um simples modelo de classes, o que se propõe é uma nova forma de pensar os sistemas de informação atuais, modelados de forma a armazenar o histórico das transações ocorridas.

7 CONCLUSÃO

A tecnologia de *Web services*, embora recente, vem se firmando como uma opção para a interoperabilidade de serviços distribuídos e integração de sistemas legados. O principal fator que contribuiu para a grande aceitação desta tecnologia é a adoção de protocolos abertos, que utilizam tecnologias já estabelecidas na Internet, como HTTP e XML. A possibilidade de integração entre sistemas legados desenvolvidos nas mais diferentes plataformas através do uso de *Web services* sem necessidade de reescrita de toda a aplicação reduz significativamente o custo de manutenção de sistemas e o tempo necessário para sua implementação.

Outro fator que merece destaque é a possibilidade de acionar serviços somente sob demanda. Com isso, o tráfego na Internet é reduzido, visto que não há a necessidade de se manterem conexões persistentes entre os provedores de *Web services* e seus clientes.

Existem várias ferramentas para desenvolvimento de *Web services*, e a incorporação deste recurso nas versões mais recentes de conhecidas linguagens de programação tende a incrementar significativamente sua aplicação e uso.

O princípio de funcionamento de um *Web service* é razoavelmente simples. No entanto, sua implementação obriga o desenvolvedor a obter conhecimento sobre um grande número de protocolos e tecnologias, como HTTP, XML, SOAP e WSDL, além, é claro, de um domínio sobre a linguagem de programação utilizada para tal. Esta grande variedade de padrões sob os quais se baseiam os *Web services* pode ser considerado um grande problema. Qualquer mudança severa em um destes padrões pode tornar um *Web service* restrito quanto à sua utilização. Percebe-se claramente uma ação conjunta, coordenada por grandes companhias desenvolvedoras de *software* e consórcios de empresas, que busca consolidar os padrões existentes, ajustando suas eventuais falhas. Se forem criados muitos novos padrões, a interoperabilidade ficará comprometida.

Quanto ao aspecto de segurança, é possível concluir que os padrões apresentados até o momento não são suficientemente claros e consolidados a ponto de se tornarem um item obrigatórios nos *Web services.* Tanto o provedor do *Web service* como seu cliente devem usar entre si o mesmo padrão, método ou algoritmo criptográfico de informações. Ao estabelecer estas normas, o provedor de *Web services* pode estar criando uma barreira para o uso de seus serviços pelos seus clientes, por discordância das normas adotadas ou simplesmente desconhecimento dos procedimentos para sua implementação.

Ao desenvolver um sistema *Web*, faz-se necessário modelar previamente todos os *Web services* que serão disponibilizados. A simples falta de um mecanismo de autenticação de *Web services* obriga o desenvolvedor a enviar informações adicionais nas mensagens de requisição de serviços. Dependendo do domínio do problema, bem como da forma como os *Web services* serão disponibilizados, as classes e métodos da aplicação poderão ser melhor

estruturados, aumentando a coesão, incrementando o reuso e facilitando a manutenção do sistema.

No momento em que se busca uma maior integração de sistemas de informação, e a adoção de padrões abertos para comunicação entre estes, os *Web services* surgiram como uma promessa revolucionário que, se não atende ainda a todas as necessidades, possui condições de se tornar um padrão de fato na indústria de *software*.

Grandes empresas estão apostando alto na tecnologia de *Web services*. Mais que isto, estão dispostas a fazerem concessões, consórcios, elaboração de padrões e integração de sistemas de diferentes fornecedores. No mundo globalizado, percebe-se que não há mais espaço para uma solução única, baseada em um único fornecedor de sistemas. A facilidade de integração dos sistemas de informação tende a ser tratado como um assunto tão importante quanto suas respectivas funcionalidades.

Com isto, urge a necessidade de que, em trabalhos futuros, sejam analisados os *frameworks* de modelagem de sistemas de informação existentes, adequando-os para contemplar de forma adequada o emprego de *Web services.*

Como contribuição deste trabalho, tem-se também um referencial sobre as etapas do desenvolvimento dos *Web services* usando software livre. O desenvolvimento de *Web services* usando ferramentas de *Software* Livre concede ao desenvolvedor uma liberdade ampla, à medida que este dispõe de todo o código-fonte da biblioteca utilizada e pode incluir modificações nestas, customizando seus resultados ou analisando transações solicitadas e realizadas em tempo de execução [ZAV 2004].

REFERÊNCIAS

[BPE 2002] BPEL. Disponível em: <http://www.informatik.uni-freiburg.de/~koehler/bpia/bpia.html> . Acesso em: nov.2002.

[BPM 2002] BUSINESS Process Modeling Language. Disponível em: <http://xml.coverpages.org/BPML-2002.pdf> . Acesso em: dez.2002.

[BRA 2002] BRADLEY, N. **The XML Companion** 3rd ed. Boston: Addison-Wesley, 2002.

[BUS 2003] BUSINESS Process Execution Language for Web Services. Disponível em: <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>. .Acesso em: jan.2003.

[CIM 2002] CIMOSA Primer 5**.** Disponível em: <http://cimosa.cnt.pl/Docs/Primer/primer5.htm> . Acesso em: jun.2002.

[DÉC 2000] DÉCIO, O. C. **XML:** Guia de Consulta Rápida. São Paulo: Novatec, 2000.

[EIL 2002] ENTERPRISE Modelling. Disponível em: <http://www.eil.utoronto.ca/enterprise-modelling/>. Acesso em: jun.2002.

[FRA 2002] FRANCESCONI, M. **Padrões XML para Gerenciamento de Processo de Negócio**. Disponível em: <http://www.imageware.com.br/MBA_MF.pdf>. Acesso em: jun.2003.

[HTT 2003] HTTP Specification and Drafts. Disponível em: <http://www.w3.org/Protocols/Spec.html>. Acesso em: out.2003.

[MAR 2002] MARUYAMA, H. et al. **XML and Java** 2nd ed. Boston: Addison-Wesley, 2002.

[MAT 2002] MATOS, A. V. de. **UML**: Prático e Descomplicado**.** São Paulo:

Érica, 2002.

[NUS 2003] NuSOAP Home Page. Disponível em: <http://dietrich.ganx4.nusoap/index.php> . Acesso em: out.2003.

[RAY 2001] RAY, E. T. ***Learning XML***. Beijing: O'Reilly, 2001.

[SOA 2003] SIMPLE Object Access Protocol (SOAP) 1.1. Disponível em: <http://www.w3.org/TR/SOAP> . Acesso em: ago.2003.

[UDD 2003] UDDI.org. Disponível em: <http://www.uddi.org>. Acesso em: ago.2003.

[VER 1999] VERNADAT, François. Requirements for simulation tools in Enterprise Engineering. In: INTERNATIONAL CONFERENCE ON CAD/CAM, ROBOTICS, AND FACTORIES OF THE FUTURE, CARS & FOF, 15., 1999, Águas de Lindóia. **Proceedings...** [S.l.: s.n.], 1999.

[WMC 2003] WORKFLOW Management Coalition. Disponível em: <http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf>. Acesso em: fev.2003.

[WSA 2004] WEB Services Architecture. Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: mar.2004.

[WSD 2003] WEB Services Description Language (WSDL) Version 1.2. Disponível em: <http://www.w3.org/TR/2003/WD-wsdl12-20030303> . Acesso em: ago.2003.

[WSG 2004] WEB Services Glossary. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>. Acesso em: mar.2004.

[XPD 2002] XML Process Definition Language, Versão 1.0. Disponível em: <http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf>. Acesso em: dez.2002.

[ZAV 2004] ZAVALIK, C.; LACERDA, G.; OLIVEIRA, J.P.M.de. Implementando Web Services com Software Livre. In: FORUM INTERNACIONAL DE SOFTWARE LIVRE, 5., 2004, Porto Alegre. **Anais...** Porto Alegre: SBC, 2004. p.35-38.

Anexo A WSDL do Sistema Despesas *Web*

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="despesasWeb"
   targetNamespace="http://192.168.1.1/mestrado/despesas/ws/"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://192.168.1.1/mestrado/despesas/ws/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <types>
       <complexType name="arraydeInteiros">
          <complexContent>
             <restriction base="SOAP-ENC:Array">
                <attribute ref="SOAP-ENC:arrayType" arrayType="xsd:integer[]"/>
             </restriction>
          </complexContent>
       </complexType>
   </types>

   <message name="recebeDadosUsuario">
      <part name="nome"  type="xsd:string"/>
      <part name="login" type="xsd:string"/>
      <part name="senha" type="xsd:string"/>
      <part name="email" type="xsd:string"/>
   </message>
   <message name="enviaRespostaUsuario">
      <part name="codigo" type="xsd:integer"/>
   </message>

   <message name="recebeDadosAlteracaoUsuario">
      <part name="login" type="xsd:string"/>
      <part name="senha" type="xsd:string"/>
      <part name="novonome"  type="xsd:string"/>
      <part name="novologin" type="xsd:string"/>
      <part name="novasenha" type="xsd:string"/>
      <part name="novoemail" type="xsd:string"/>
   </message>
   <message name="enviaOkUsuario">
      <part name="resposta" type="xsd:boolean"/>
   </message>

   <message name="recebeIDUsuario">
      <part name="login" type="xsd:string"/>
      <part name="senha" type="xsd:string"/>
   </message>

   <message name="enviaCodigoUsuario">
      <part name="codigo" type="xsd:integer"/>
   </message>

   <message name="recebeDadosConta">
      <part name="login" type="xsd:string"/>
      <part name="senha" type="xsd:string"/>
      <part name="descricao" type="xsd:string"/>
   </message>

   <message name="enviaCodigoConta">
      <part name="codigo" type="xsd:integer"/>
   </message>

   <message name="recebeDadosAlteracaoConta">
      <part name="login" type="xsd:string"/>
      <part name="senha" type="xsd:string"/>
      <part name="descricao"  type="xsd:string"/>
      <part name="novadescricao" type="xsd:string"/>
   </message>

   <message name="recebeDadosLancamento">
```

```
   <part name="codigo_usuario" type="xsd:integer"/>
   <part name="codigo_conta"   type="xsd:integer"/>
   <part name="data"           type="xsd:string"/>
   <part name="historico"      type="xsd:string"/>
   <part name="valor"          type="xsd:float"/>
</message>

<message name="enviaCodigoLancamento">
   <part name="codigo" type="xsd:integer"/>
</message>

<message name="enviaDadosLancamento">
   <part name="codigo_lancamento" type="xsd:integer"/>
   <part name="codigo_conta"      type="xsd:integer"/>
   <part name="data"              type="xsd:string"/>
   <part name="historico"         type="xsd:string"/>
   <part name="valor"             type="xsd:float"/>
</message>

<message name="recebeDadosAlteracaoLancamento">
   <part name="login" type="xsd:string"/>
   <part name="senha" type="xsd:string"/>
   <part name="codigo_lancamento" type="xsd:integer"/>
   <part name="codigo_conta"      type="xsd:integer"/>
   <part name="data"              type="xsd:string"/>
   <part name="historico"         type="xsd:string"/>
   <part name="valor"             type="xsd:float"/>
</message>

<message name="recebeDadosExclusaoLancamento">
   <part name="login" type="xsd:string"/>
   <part name="senha" type="xsd:string"/>
   <part name="codigo_lancamento" type="xsd:integer"/>
</message>

<message name="recebeDadosConsultaPeriodo">
   <part name="login" type="xsd:string"/>
   <part name="senha" type="xsd:string"/>
   <part name="data_inicial" type="xsd:string"/>
   <part name="data_final"   type="xsd:string"/>
</message>

<message name="enviaArrayLancamentos">
   <part name="arrayCodigos"    type="tns:arraydeInteiros"/>
</message>

<portType name="despesasWeb_PortType">
   <operation name="insereUsuario">
      <input message="tns:recebeDadosUsuario"/>
      <output message="tns:enviaRespostaUsuario"/>
   </operation>
   <operation name="alteraUsuario">
      <input message="tns:recebeDadosAlteracaoUsuario"/>
      <output message="tns:enviaOkUsuario"/>
   </operation>
   <operation name="excluiUsuario">
      <input message="tns:recebeIDUsuario"/>
      <output message="tns:enviaOkUsuario"/>
   </operation>
   <operation name="buscaCodigoUsuario">
      <input message="tns:recebeIDUsuario"/>
      <output message="tns:enviaCodigoUsuario"/>
   </operation>
   <operation name="insereContaDespesa">
      <input message="tns:recebeDadosConta"/>
      <output message="tns:enviaCodigoConta"/>
   </operation>
   <operation name="alteraContaDespesa">
      <input message="tns:recebeDadosAlteracaoConta"/>
      <output message="tns:enviaOKUsuario"/>
   </operation>
   <operation name="excluiContaDespesa">
      <input message="tns:recebeDadosConta"/>
      <output message="tns:enviaOKUsuario"/>
```

```
   </operation>
   <operation name="insereLancamento">
      <input message="tns:recebeDadosLancamento"/>
      <output message="tns:enviaCodigoLancamento"/>
   </operation>
   <operation name="alteraLancamento">
      <input message="tns:recebeDadosAlteracaoLancamento"/>
      <output message="tns:enviaOkUsuario"/>
   </operation>
   <operation name="excluiLancamento">
      <input message="tns:recebeDadosExclusaoLancamento"/>
      <output message="tns:enviaOkUsuario"/>
   </operation>
   <operation name="buscaLancamentosPeriodo">
      <input message="tns:recebeDadosConsultaPeriodo"/>
      <output message="tns:enviaArrayLancamentos"/>
   </operation>

   <operation name="buscaDadosLancamento">
      <input message="tns:recebeDadosExclusaoLancamento"/>
     <output message="tns:enviaDadosLancamento"/>
   </operation>

</portType>

<binding name="despesasWeb_Binding" type="tns:despesasWeb_PortType">
   <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="insereUsuario">
      <soap:operation soapAction="insereUsuario"/>
      <input>
        <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </input>
      <output>
         <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </output>
   </operation>

   <operation name="alteraUsuario">
      <soap:operation soapAction="alteraUsuario"/>
      <input>
        <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </input>
      <output>
         <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </output>
   </operation>

   <operation name="excluiUsuario">
      <soap:operation soapAction="excluiUsuario"/>
      <input>
        <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </input>
      <output>
         <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </output>
   </operation>

   <operation name="buscaCodigoUsuario">
      <soap:operation soapAction="buscaCodigoUsuario"/>
      <input>
         <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </input>
      <output>
         <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </output>
   </operation>

   <operation name="insereContaDespesa">
      <soap:operation soapAction="insereContaDespesa"/>
      <input>
        <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
      </input>
      <output>
```

```
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="alteraContaDespesa">
   <soap:operation soapAction="alteraContaDespesa"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="excluiContaDespesa">
   <soap:operation soapAction="excluiContaDespesa"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="insereLancamento">
   <soap:operation soapAction="insereLancamento"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="alteraLancamento">
   <soap:operation soapAction="alteraLancamento"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="excluiLancamento">
   <soap:operation soapAction="excluiLancamento"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="buscaLancamentosPeriodo">
   <soap:operation soapAction="buscaLancamentosPeriodo"/>
   <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>

<operation name="buscaDadosLancamento">
   <soap:operation soapAction="buscaDadosLancamento"/>
  <input>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </input>
   <output>
      <soap:body namespace="urn:despesasWeb_Service" use="encoded"/>
   </output>
</operation>
```

```
   </binding>


   <service name="despesasWeb_Service">
     <port binding="tns:despesasWeb_Binding" name="despesasWeb_Port">
         <soap:address
            location="http://192.168.1.1/mestrado/despesas/ws/despesasWeb.php"/>
      </port>
   </service>

</definitions>
```

Anexo B Código Fonte dos *Web services*

```
<?php
require_once('nusoap.php');
$s = new soap_server;
$s->register('insereUsuario');
$s->register('alteraUsuario');
$s->register('excluiUsuario');
$s->register('buscaCodigoUsuario');
$s->register('insereContaDespesa');
$s->register('alteraContaDespesa');
$s->register('excluiContaDespesa');
$s->register('buscaCodigoConta');
$s->register('insereLancamento');
$s->register('alteraLancamento');
$s->register('excluiLancamento');
$s->register('buscaLancamentosPeriodo');
$s->register('buscaDadosLancamento');

function insereUsuario($nome, $login, $senha, $email){

        // Validacao dos parametros passados
        //----------------------------------
        if($nome  == '' ){
           return new soap_fault('Client','','Voce deve fornecer o nome do usuario.');
        }
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($email == '' ){
           return new soap_fault('Client','','Voce deve fornecer o email do usuario.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario ja existe, impedindo sua inclusao
        //--------------------------------------------------------
        $instrucao = "select count(*) from usuario where login = '$login' and
senha='$senha';";
        $resultado = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        $quantidade= @mysql_result( $resultado, 0, 0);
        if ($quantidade > 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario JA Cadastrado.');
        }

        // Monta a query a ser executada
        //------------------------------
        $instrucao = "insert into usuario (codigo,nome,login,senha,e_mail) values
(0,'$nome','$login', '$senha', '$email');";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
```

```
           return new soap_fault('Server','','Nao conseguiu inserir os dados no banco de
dados');
        }

        // Como o codigo  auto_increment, tem que buscar o codigo do usuario inserido
        //--------------------------------------------------------------------------
        $instrucao     = "select codigo from usuario where login = '$login' and
senha='$senha';";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        $codigousuario = @mysql_result( $resultado, 0, 0);

        Fecha_banco( $banco );

        return $codigousuario;
}

function alteraUsuario($login, $senha, $novonome, $novologin, $novasenha, $novoemail){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso atual.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso atual.');
        }
        if($novonome == '' ){
           return new soap_fault('Client','','Voce deve fornecer o novo nome do usuario.');
        }
        if($novologin == '' ){
           return new soap_fault('Client','','Voce deve fornecer o novo login do usuario.');
        }
        if($novasenha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a nova senha do usuario.');
        }
        if($novoemail == '' ){
           return new soap_fault('Client','','Voce deve fornecer o novo e-mail do usuario.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "update usuario ";
        $instrucao .= "set nome = '$novonome', ";
        $instrucao .= "    login = '$novologin', ";
        $instrucao .= "    senha = '$novasenha', ";
        $instrucao .= "    e_mail = '$novoemail' ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu alterar os dados no banco de
dados ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
        }

        Fecha_banco( $banco );
        return true;
}

function excluiUsuario($login, $senha ){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso atual.');
        }
```

```
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso atual.');
        }

        // Abre a conexao com o banco de dados
        //-----------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "delete from usuario ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu excluir os dados no banco de
dados ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
        }

        Fecha_banco( $banco );
        return true;
}

function insereContaDespesa($login, $senha, $descricao){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($descricao == '' ){
           return new soap_fault('Client','','Voce deve fornecer a descricao da conta.');
        }

        // Abre a conexao com o banco de dados
        //-----------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Busca o codigo do usuario para incluir a conta
        //-----------------------------------------------
        $codigo_usuario = buscaCodigo( $banco, $login, $senha);
        if ( $codigo_usuario == 0 )
           return new soap_fault('Server','','Usuario NAO Cadastrado '. $codigo_usuario .
$login . $senha );

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "insert into contas_despesa (codigo,descricao,cod_usuario) ";
        $instrucao .= "values (0,'$descricao',$codigo_usuario);";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu inserir os dados no banco de
dados ');
        }

        // Como o codigo e' auto_increment, tem que buscar o codigo da conta inserida
        //---------------------------------------------------------------------------
        $instrucao     = "select codigo from contas_despesa where descricao = '$descricao'
and cod_usuario='$codigo_usuario';";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        $codigoconta   = @mysql_result( $resultado, 0, 0);

        Fecha_banco( $banco );
```

```
        return $codigoconta;
}

function alteraContaDespesa($login, $senha, $descricao, $novadescricao){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($descricao == '' ){
           return new soap_fault('Client','','Voce deve fornecer a descricao da conta.');
        }
        if($novadescricao == '' ){
           return new soap_fault('Client','','Voce deve fornecer a nova descricao da
conta.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Busca o codigo do usuario para altera a conta
        //-----------------------------------------------
        $codigo_usuario = buscaCodigo( $banco, $login, $senha);
        if ( $codigo_usuario == 0 )
           return new soap_fault('Server','','Usuario NAO Cadastrado '. $codigo_usuario .
$login . $senha );

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "update contas_despesa ";
        $instrucao .= "set descricao = '$novadescricao' ";
        $instrucao .= "where cod_usuario = '$codigo_usuario' and descricao = '$descricao';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_affected_rows($banco) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','A Conta solicitada NAO Existe ');
        }

        Fecha_banco( $banco );

        return true;
}

function excluiContaDespesa($login, $senha, $descricao){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($descricao == '' ){
           return new soap_fault('Client','','Voce deve fornecer a descricao da conta.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Busca o codigo do usuario para altera a conta
        //-----------------------------------------------
        $codigo_usuario = buscaCodigo( $banco, $login, $senha);
        if ( $codigo_usuario == 0 )
           return new soap_fault('Server','','Usuario NAO Cadastrado '. $codigo_usuario .
$login . $senha );
```

```
        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "delete from contas_despesa ";
        $instrucao .= "where cod_usuario = '$codigo_usuario' and descricao = '$descricao';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_affected_rows($banco) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','A Conta solicitada NAO Existe ');
        }

        Fecha_banco( $banco );

        return true;
}

function insereLancamento($codigo_usuario, $codigo_conta, $data, $historico, $valor ){

        // Validacao dos parametros passados
        //----------------------------------
        if($codigo_usuario == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo do usuario.');
        }
        if($codigo_conta == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo da conta.');
        }
        if($data == '' ){
           return new soap_fault('Client','','Voce deve fornecer a data do lancamento.');
        }
        if($historico == '' ){
           return new soap_fault('Client','','Voce deve fornecer o historico do
lancamento.');
        }
        if($valor == '' ){
           return new soap_fault('Client','','Voce deve fornecer o valor do lancamento.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario existe
        //-----------------------------
        $usuarioValido = verificaUsuario($banco,$codigo_usuario);
        if (!$usuarioValido)
           return new soap_fault('Client','','Usuario NAO Cadastrado');

        // Verifica se a conta existe
        //---------------------------
        $contaValida = verificaConta( $banco, $codigo_conta, $codigo_usuario);
        if (!$contaValida)
           return new soap_fault('Client','','Conta NAO Cadastrado');

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "insert into lancamento
(numero,cod_usuario,data_lanc,cod_conta,historico,valor) ";
        $instrucao .= "values (0,$codigo_usuario,'$data', $codigo_conta, '$historico',
$valor);";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu inserir os dados no banco de
dados ');
        }

        // Como o codigo e' auto_increment, tem que buscar o codigo do lancamento inserida
        //--------------------------------------------------------------------------
        $instrucao     = "select numero from lancamento ";
        $instrucao    .= "where cod_usuario = $codigo_usuario and data_lanc = '$data' and
cod_conta=$codigo_conta and historico = '$historico' and valor=$valor;";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        $codigo_lanc   = @mysql_result( $resultado, 0, 0);
```

```
        Fecha_banco( $banco );

        return $codigo_lanc;
}

function alteraLancamento($login, $senha, $codigo_lancamento, $codigo_conta, $data,
$historico, $valor ){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($codigo_lancamento == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo do lancamento.');
        }
        if($codigo_conta == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo da conta.');
        }
        if($data == '' ){
           return new soap_fault('Client','','Voce deve fornecer a data do lancamento.');
        }
        if($historico == '' ){
           return new soap_fault('Client','','Voce deve fornecer o historico do
lancamento.');
        }
        if($valor == '' ){
           return new soap_fault('Client','','Voce deve fornecer o valor do lancamento.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario existe e senha Ok
        //----------------------------------------
        $instrucao  = "select codigo from usuario ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu localizar dados do usuario
informado ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
        }
        $codigo_usuario = @mysql_result( $resultado, 0, 0);

        // Verifica se a conta existe
        //---------------------------
        $contaValida = verificaConta( $banco, $codigo_conta, $codigo_usuario);
        if (!$contaValida)
           return new soap_fault('Client','','Conta NAO Cadastrado');

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "update lancamento ";
        $instrucao .= "set data_lanc = '$data',";
        $instrucao .= "    cod_conta = $codigo_conta,";
        $instrucao .= "    historico = '$historico',";
        $instrucao .= "    valor = $valor ";
        $instrucao .= "where numero = $codigo_lancamento; ";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_affected_rows($banco) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Lancamento Inexistente ou nao alterado ');
        }
```

```
        Fecha_banco( $banco );

        return true;
}

function excluiLancamento( $login, $senha, $codigo_lancamento ){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($codigo_lancamento == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo do lancamento.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario existe e senha Ok
        //----------------------------------------
        $instrucao  = "select codigo from usuario ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu localizar dados do usuario
informado ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
        }
        $codigo_usuario = @mysql_result( $resultado, 0, 0);

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "delete from lancamento ";
        $instrucao .= "where numero = $codigo_lancamento ";
        $instrucao .= "and cod_usuario = $codigo_usuario ;";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_affected_rows($banco) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Lancamento Inexistente - Impossivel excluir ');
        }

        Fecha_banco( $banco );
        return true;
}

function buscaDadosLancamento( $login, $senha, $codigo_lancamento ){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($codigo_lancamento == '' ){
           return new soap_fault('Client','','Voce deve fornecer o codigo do lancamento.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');
```

```
        // Verifica se o usuario existe e senha Ok
        //---------------------------------------
        $instrucao  = "select codigo from usuario ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu localizar dados do usuario
informado ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
        }
        $codigo_usuario = @mysql_result( $resultado, 0, 0);

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "select numero, cod_conta, data_lanc, historico, valor from lancamento
";
        $instrucao .= "where numero = $codigo_lancamento ";
        $instrucao .= "and cod_usuario = $codigo_usuario ;";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_num_rows($resultado) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Lancamento Inexistente - Impossivel excluir ');
        }

        // Coloca no vetor de retorno (TEM QUE SER POR REFERENCIA )
        //--------------------------
        $vet_dados = array();
        for ($ind=0;$ind < 5;$ind++) {
           $vet_dados[$ind] = @mysql_result( $resultado, 0, $ind);
        }
        Fecha_banco( $banco );
        return $vet_dados;
}

function buscaLancamentosPeriodo( $login, $senha, $data_inicial, $data_final ){

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($data_inicial == '' ){
           return new soap_fault('Client','','Voce deve fornecer a data inicial');
        }
        if($data_final == '' ){
           return new soap_fault('Client','','Voce deve fornecer a data final');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! $banco=Conecta_banco() )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Verifica se o usuario existe e senha Ok
        //----------------------------------------
        $instrucao  = "select codigo from usuario ";
        $instrucao .= "where login = '$login' and senha = '$senha';";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( !$resultado ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Nao conseguiu localizar dados do usuario
informado ');
        }
        if ( @mysql_affected_rows( $banco ) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado ');
```

```
        }
        $codigo_usuario = @mysql_result( $resultado, 0, 0);

        // Monta a query a ser executada
        //------------------------------
        $instrucao  = "select numero,data_lanc,cod_conta,historico,valor ";
        $instrucao .= " from lancamento ";
        $instrucao .= " where cod_usuario = $codigo_usuario ";
        $instrucao .= " and data_lanc between '$data_inicial' and '$data_final' ";
        $instrucao .= " order by data_lanc; ";
        $resultado = @mysql_db_query( "despesas_com_br_-_despesas",$instrucao,$banco );
        if ( @mysql_num_rows($resultado) == 0 ) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Lancamentos Inexistentes para o criterio
informado ');
        }

        $vet_numero = array();

        // Coloca tudo nos arrays
        //-----------------------
        for ($ind = 0; $ind < @mysql_num_rows( $resultado ); $ind++) {
            $vet_numero[$ind]  = @mysql_result( $resultado, $ind, 0);
        }
        Fecha_banco( $banco );
        return $vet_numero;
}


function buscaCodigoUsuario($login, $senha) {

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }

        // Abre a conexao com o banco de dados
        //------------------------------------
        if ( ! ($banco=Conecta_banco() ) )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Busca o codigo do usuario no banco de dados
        //--------------------------------------------
        $instrucao     = "select codigo from usuario where login = '$login' and
senha='$senha';";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado.');
        }
        else
        {
            $codigousuario = @mysql_result( $resultado, 0, 0);
            Fecha_banco( $banco );
            return $codigousuario;
        }
}

function buscaCodigoConta($login, $senha, $descricao) {

        // Validacao dos parametros passados
        //----------------------------------
        if($login == '' ){
           return new soap_fault('Client','','Voce deve fornecer o login de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a senha de acesso.');
        }
        if($senha == '' ){
           return new soap_fault('Client','','Voce deve fornecer a descricao da conta.');
```

```
        }

        // Abre a conexao com o banco de dados
        //-----------------------------------
        if ( ! ($banco=Conecta_banco() ) )
           return new soap_fault('Server','','Nao conseguiu conectar ao banco de dados.');

        // Busca o codigo do usuario no banco de dados
        //--------------------------------------------
        $instrucao     = "select codigo from usuario where login = '$login' and
senha='$senha';";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Usuario NAO Cadastrado.');
        }
        $codigousuario = @mysql_result( $resultado, 0, 0);

        // Busca o codigo da conta no banco de dados
        //------------------------------------------
        $instrucao     = "select codigo from contas_despesa where cod_usuario =
'$codigousuario' and descricao like '$descricao';";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0) {
           Fecha_banco($banco);
           return new soap_fault('Server','','Conta NAO Cadastrada.');
        }
        $codigoconta = @mysql_result( $resultado, 0, 0);
        Fecha_banco( $banco );
        return $codigoconta;


}

$s->service($HTTP_RAW_POST_DATA);
exit();



// Funcoes de uso local, pelos Web services
//-----------------------------------------

function Conecta_banco() {
        $hostbanco    = "localhost";
        $usuariobanco = "despesas";
        $senhabanco   = "desp980";
        return @mysql_connect($hostbanco,$usuariobanco,$senhabanco);
}

function Fecha_banco($banco) {
        return @mysql_close($banco);
}


function verificaUsuario($banco, $codigo_usuario) {
        // Pega o codigo do Usuario
        //-------------------------
        $instrucao = "select codigo from usuario where codigo = '$codigo_usuario' ";
        $resultado = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0)
           return false;
        return true;
}

function verificaConta($banco, $codigo_conta, $codigo_usuario) {
        // Pega o codigo do Usuario
        //-------------------------
        $instrucao = "select codigo from contas_despesa where codigo = '$codigo_conta' and
cod_usuario = '$codigo_usuario';";
        $resultado = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0)
           return false;
        return true;
```

```
}

function buscaCodigoLancamento ($banco, $codigo_usuario, $codigo_conta, $valor ) {

        $instrucao  = "select numero from lancamento ";
        $instrucao .= "where cod_usuario = $codigo_usuario ";
        $instrucao .= "and cod_conta = $codigo_conta ";
        $instrucao .= "and valor = $valor; ";
        $resultado     = @mysql_db_query("despesas_com_br_-_despesas",$instrucao,$banco);
        if (@mysql_num_rows($resultado) == 0)
           return 0;
        return @mysql_result( $resultado, 0, 0);

}

?>
```