CARLOS ROBERTO ALVES DE OLIVEIRA

ARQUITETURA DE INTEGRAÇÃO DE BANCOS DE DADOS UTILIZANDO WEB SERVICES

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Especialista em Engenharia de Software.

Área de Concentração:
Objetos Distribuídos

Orientador:
Prof. Dr. Jorge Luís Risco Becerra

SÃO PAULO
2005

FICHA CATALOGRÁFICA

Oliveira, Carlos Roberto Alves de

Arquitetura de Integração de Bancos de Dados Utilizando Web Services
São Paulo, 2005. 66p.

Dissertação (MBA) – Escola Politécnica da Universidade de São Paulo.
PECE – Engenharia de Software – MBA USP

1. Integração de bancos de dados 2. Aplicações Cliente / Servidor

Ao meu pai e ao meu irmão (*In memorian*).

AGRADECIMENTOS

Ao Prof. Dr. Jorge Luís Risco Becerra por suas diretrizes, apoio e incentivo na elaboração deste trabalho.

RESUMO

A necessidade de integração de dados e sistemas, é um problema antigo que vem se tornando cada vez mais fácil devido ao surgimento de novas tecnologias. Os desenvolvimentos de aplicações que fazem acesso a bancos de dados precisam estar aptos a utilizar qualquer produto do mercado e deles poder extrair todo seu potencial.

A infra-estrutura existente em ambientes corporativos deve ser aproveitada evitando-se assim custos desnecessários, neste cenário, a utilização de *Web Services* como serviço facilitador de integração entre aplicações de *software* e os bancos de dados se adapta de maneira exemplar.

Neste trabalho apresentamos uma proposta de arquitetura baseada em *Web Services* como um *middleware* capaz de integrar aplicações cliente aos diversos bancos de dados do mercado, facilitando assim o desenvolvimento e a manutenção do *software* durante seu ciclo de vida.

Palavras chaves: *Web Services*, *Middleware*, Arquitetura, Protocolo, Integração, Bancos de Dados, Cliente e Servidor.

ABSTRACT

The necessity of integration of data and systems is a old problem that has become easier with the increase of new technologies. The development of procedures to access the various databases in the marketplace must be precise and able to be applicable with any product on the market in order to obtain its full potential.

The corporative environment infra-structure requires that unnecessary cost be avoided. In this case the utilization of Web Services as facilitator service of integration between software and databases works perfectly.

In this work we present a proposal of architecture based on Web Services as one middleware capable to integrate applications customer to the various databases of the market, thus facilitating the development and the maintenance of software during its cycle of life.

Key words: Web Services, Middleware, Architecture, Integration, Protocol, Databases, Client Server.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACID -	Atomicity, Consistency, Isolation, Durability
API -	Application Program Interface
CLOB -	Character Large Object
COM -	Component Object Model
CORBA -	Common Object Request Broker Architecture
CPU -	Central Processing Unit
CSS -	Cascade Style Sheet
DCE -	Distributed Computing Environment
DCOM -	Distributed Common Object Model
DOM -	Document Object Model
DTD -	Document Type Definition
EAI -	Enterprise Application Integration
ERP -	Enterprise Resource Planning
FORTRAN -	Formula Translator
FTP -	File Transfer Protocol
GUI -	Graphical User Interface
HTML -	Hypertext Markup Language
HTTP -	Hypertext Transfer Protocol
IDL -	Interface Definition language
IIOP -	Internet Inter-ORB Protocol
J2EE -	Java 2 Enterprise Edition
JDBC -	Java Database Connectivity
LAN -	Local Area Network
ODBC -	Open Database Connectivity
OLE -	Object Linking and Embedding
OMG -	Object Management Group
OOP -	Object Oriented Programming
ORB -	Object Request Broker
RAD -	Rapid Application Development
SAX	Simple API for XML

SGBD -	Sistema Gerenciador de Banco de Dados
SGML	Standard Generalized Markup Language
SOA -	Service-Oriented Architecture
SOAP -	Simple Object Access Protocol
SQL -	Structured Query Language
TCP/IP -	Transmission Control Protocol / Internet Protocol
TI -	Tecnologia da Informação
UDDI -	Universal Description, Discovery, and Integration
UML -	Unified Modeling Language
URL -	Uniform Resource Locator
W3C -	Word Wide Web Consortium
WAN -	Wide Area Network
WSDL -	Web Service Description Language
XDR -	XML Data Reduced Language
XML -	Extensible Markup Language
XSD -	XML Schema Definition language
XSL -	Extensible Style Sheet
XSLT -	Extensible Stylesheet Language Transformation

1 – INTRODUÇÃO

As aplicações de *software* destinadas a atender um número grande de clientes necessitam de padronização ao requisitar informações em bancos de dados, e ao mesmo tempo extrair o máximo do poder oferecido pelos diversos produtos.

Os bancos de dados relacionais são utilizados por muitas empresas, desde a grande como também a média e a pequena, são utilizados produtos de diversos fabricantes e também soluções de código aberto. Uma aplicação de *software* destinada a atender um mercado com tamanha heterogeneidade antes de tudo deve tomar decisões de projeto que possibilite o amadurecimento da aplicação, a facilidade de manutenção e integração.

A integração de bancos de dados, se apresenta assim, como atividade cada vez mais necessária quando se pensa em desenvolvimento de sistemas tanto para a indústria, comércio e instituições financeiras. Os aplicativos precisam estar preparados para interagir com os diversos bancos de dados do mercado e para minimizar os custos, utilizar-se da infra-estrutura já existente.

Os desenvolvedores de SGBD's, acrescentam novas características e peculiaridades a cada nova versão de seus produtos com o intuito de conquistar fatias de mercado tornando a integração um empreendimento de porte.

A SQL transformou-se na linguagem de bancos de dados predominante para *mainframes* e servidores de rede: ela é o centro das atenções num cenário de competição por fatias de mercado.[Orfali,96]. Conjuntamente com este fato, tecnologias recentes têm se apresentado como uma opção à integração, dentre elas os chamados *Web Services*.

O contexto deste capítulo, está composto de tópicos referentes aos objetivos, abrangência do trabalho, justificativa, motivação e a metodologia de pesquisa utilizada.

O tópico a seguir apresenta as considerações iniciais sobre o presente trabalho.

1.1 – Considerações Iniciais

O principal objetivo por trás da próxima geração de plataformas de linguagens de computador é a necessidade de integrar de maneira transparente aplicações e dispositivos com a plataforma.

O paradigma da orientação a objetos, hoje amadurecido está pronto para ser substituído pela próxima geração de linguagens e plataformas de computação. As plataformas de quinta geração foram propostas, mas nenhuma teve um impacto expressivo na industria.

A independência de hardware prevista pelas mais avançadas plataformas de linguagens como JAVA não é tão completa como foi apregoada no início para esses sistemas. Para o caso da plataforma .NET o problema de independência ainda é maior pois a plataforma é proprietária. Apesar de incorporarem um conjunto completo de recursos orientados para objetos, ainda estão sujeitas a vários problemas.

O avanço da *Internet*, iniciou uma revolução na arquitetura de *software,* uma nova estratégia de projetos de aplicações chamada de cliente / servidor apareceu, e as empresas começaram a rever seus negócios para obter vantagens e

produtividade das aplicações cliente / servidor. As aplicações cliente / servidor permitem uma operação mais eficiente por facilitar trabalho colaborativo e compartilhamento de informações.

Com a utilização de tecnologias *Web* em aplicações desenvolvidas em várias camadas, os desenvolvedores agora precisam desenvolver e manter apenas uma versão da camada cliente, melhor que suportar várias versões algumas vezes incompatíveis, nesse contexto as aplicações de comércio eletrônico podem ser criadas num padrão único.

O futuro que antes estava distante já é realidade, uma nova mudança de geração irá conduzir a sistemas acoplados livremente que interagem com outros sistemas através de redes sem levar em conta ou se importar com *hardware* ou sistema operacional do lado cliente.

Essa nova visão vai tornar possível níveis totalmente novos de interação dos sistemas de computação entre empresas, estamos falando dos XML *Web Services*, pedaços de código de programas destinados a resolver uma tarefa específica, que ficam hospedados no servidor aguardando a requisição por parte do usuário o qual poder ser um outro programa, em determinado local da rede.

Porém essas novas tecnologias, não serão os responsáveis por resolver todos os problemas de acoplamento de sistemas quando se pensa num acoplamento geral de aplicações legadas no *mainframe*, ou aplicações na plataforma baixa.

Com toda a tecnologia de linguagens de programação e as facilidades de acoplamento oferecidas, quando pensamos em desenvolvimento e manutenção de sistemas sejam eles para qualquer plataforma esbarramos sempre em alguns problemas comuns: limites de orçamento e prazos.

Com estas considerações iniciais, o presente trabalho propõe o desenvolvimento de uma aplicação de *software,* precisamente um *middleware* o qual utilizará as tecnologias recentes como XML e *Web Services,* protocolos como SOAP e HTTP para integrar aplicações aos bancos de dados com o objetivo de alcançar alto grau de acoplamento e obter níveis de padronização e excelência em desenvolvimento na camada de acesso a dados.

O trabalho visa também a obtenção de conclusões importantes sobre aplicações que utilizam bancos de dados, especialmente aplicações comerciais, que poderão ser estendidas às outras áreas dos sistemas distribuídos, e assim gerar contribuições na comunidade científica e técnica.

1.2 – Objetivos

A proposta deste trabalho surgiu ao se pesquisar a necessidade de integração de bases de dados genéricas em aplicações de *software*.

O objetivo principal do trabalho é propor uma integração entre aplicações de *software* que fazem acesso à bancos de dados de diferentes distribuidores utilizando um *Web Service* como um *middleware*, o qual se encarregará das solicitações aos SGBD's, tornando as aplicações mais eficientes e padronizadas.

As aplicações clientes solicitam o serviço utilizando os protocolos padrões, e através dos módulos internos o *Web Service* identificará qual a fonte de dados deverá ser utilizada e baseado em regras pré-estabelecidas, controle de transações e concorrências, validação de usuários, controle de processos e utilização de *parsers*, direcionará a solicitação ao módulo de processamento de consulta, e estas retornarão ao cliente em XML.

As peculiaridades do servidor de banco de dados SQL e as extensões provocam grandes dores de cabeça para os fornecedores de ferramentas para bancos de dados cliente / servidor de plataformas múltiplas. Como resultado, o suporte a extensões de servidores tende a ser altamente desigual.[Orfali,1996]

Embora nos refiramos à linguagem SQL como uma "linguagem de consulta", ela possui muitos outros recursos além da consulta ao banco de dados, como meios para a definição da estrutura de dados, para modificação de dados no banco de dados e para a especificação de restrições de segurança. [Silberschatz,1999]

1.3 – Abrangência

O escopo deste trabalho de monografia, estará restrito ao contexto definido pelo projeto de aplicações cliente / servidor, a solução poderá ser hospedada em um servidor de rede local ou mesmo uma rede mundial, levando-se em consideração aspectos de segurança, confiabilidade e tempo de resposta, para tal as consultas aos bancos de dados deverão ser as mais otimizadas possíveis.

A implementação do sistema limitar-se-á ao nível de especificação, e esta deverá definir as tecnologias utilizadas para compor a arquitetura. A abrangência reside em disponibilizar o serviço em uma rede corporativa (*Intranet*) ou mesmo a *Internet* onde o *software* cliente possa utilizar este *middleware* através de TCP/IP, SOAP e XML.

Uma vez que a solução proposta faz uso da tecnologia de *Web Services* e este esteja hospedado em um servidor na rede utilizando os protocolos TCP/IP e também o protocolo SOAP, a integração tende a se tornar acoplada. A decisão de

qual camada de apresentação será utilizada, ficará a cargo da equipe de desenvolvimento como estratégia de projeto.

1.4 – Justificativa

A busca por padronização em integração de *software* é antiga, tornar a integração de tais aplicações através de ferramentas que utilizam a infra-estrutura de rede já existente, cria a justificativa necessária para a realização deste trabalho.

A escolha do tema vem de encontro à necessidade dos desenvolvedores de *software* de poder projetar suas aplicações para um mercado heterogêneo sem que haja surpresas na medida em que a aplicação amadurece. Manter grandes equipes de desenvolvimento com conhecimentos específicos de cada produto de banco de dados pode acarretar custos desnecessários.

O artigo publicado por [Mensah, 2004] que propõe a utilização de bancos de dados como consumidor e provedor de serviços através de *Web Services* forneceu os subsídios necessários para o desenvolvimento da pesquisa. O trabalho de [Martin, 2004], apresentou a modelagem orientada a objetos, para o projeto de aplicações *Web* envolvendo gerenciamento de *Web Services* em larga escala.

Na medida em que a solução proposta é capaz de interpretar requisições a fontes de dados de diferentes fabricantes e ainda assim extrair o máximo de cada produto, a equipe de desenvolvimento, passa a focar mais nas novas solicitações, pois não haverá a preocupação com as peculiaridades do SGBD utilizado pelos seus clientes.

A motivação deste trabalho de monografia, foi encontrada no desafio de propor uma solução de integração de banco de dados, que seria muito útil às empresas de desenvolvimento de *software* caso implementada. A utilização de novas tecnologias, como *Web Services* e também as ferramentas de desenvolvimento que facilitam sua implementação, torna o assunto ainda mais apaixonante.

O *software* desenvolvido na atualidade, precisa estar preparado para facilmente integrar-se aos diversos dispositivos e sistemas existentes, fazer uso de informações proveniente do *mainframe*, conectar-se aos vários produtos de banco de dados comerciais de maneira transparente tanto para o usuário final quanto para o desenvolvedor, faz com que o projeto e desenvolvimento nesse paradigma seja outra fonte de motivação.

O motivo principal está na utilização de *Web Services* como um *middleware* para integrar aplicações de *software* aos bancos de dados relacionais, de maneira que as consultas sejam enviadas ao SGBD não importando qual o fabricante do produto e suas peculiaridades. A solução resolverá onde, quando e como a requisição deverá ser executada. Outra fonte de motivação, é a pesquisa e o desenvolvimento de aplicações distribuídas.

1.5 – Metodologia

Para o desenvolvimento deste trabalho e com o objetivo de gerenciar a evolução do processo visando atingir as metas e a qualidade, foi utilizada uma metodologia cujas fases são definidas a seguir:

- Pesquisa: fase que engloba a seleção de informações relevantes de várias fontes, bibliografia específica como fontes primárias, livros acadêmicos, revistas especializadas, e *sites* da *Internet*. Esta atividade é realizada ao

longo do processo de desenvolvimento diminuindo à medida que o projeto chegue perto de seu final.

- Estruturação: fase que abrange as atividades referentes à formação da proposta e a definição da metodologia de projeto baseado na integração de dados utilizando *Web Services*.

- Análise: fase que engloba as atividades do sistema proposto, a modelagem utilizando UML e a especificação do mesmo.

- Definição da arquitetura: fase que engloba a proposta de uma arquitetura que represente a integração de bancos de dados utilizando *Web Services*.

2 – TECNOLOGIAS DISTRIBUÍDAS

A utilização de sistemas disponibilizados em redes de computadores, tem proporcionado inúmeros benefícios aos seus usuários, sejam corporativos quando falamos de redes empresariais ou mesmo um conjunto de redes interligadas como a *Internet*. O uso destes sistemas ou aplicações, tem sido de grande auxílio e imprescindível na realização das mais diversas tarefas em ambientes de trabalho os mais diversos.

Com o crescimento da *Internet*, as tecnologias de telecomunicações e informática evoluíram a níveis inesperados, e continuam a evoluir, pesquisas em novos protocolos mais simples e de fácil implementação estão sendo feitas, tecnologias como XML e *Web Services*, propõem solucionar problemas que num passado não tão distante eram simplesmente classificados como sem solução, ou de difícil implementação devido a custos e falta de tecnologia adequada.

No contexto atual, com a utilização de tais tecnologias é possível atender milhões de usuários, cujos interesses e necessidades de uso são suportados por aplicações de rede, principalmente as relacionadas ao fornecimento de serviços de informações.

As redes empresariais evoluíram com o advento da *Internet*, o mesmo modelo passou a ser adotado principalmente por utilizar-se de protocolos abertos os quais puderam e podem ser implementados para diversos sistemas operacionais. Desde o início de suas conexões, esse tipo de rede recebeu atenção especial no que tange ao controle e monitoramento dos dispositivos de rede. Este cenário vem mudando significativamente, inúmeras pesquisas e produtos comerciais da atualidade, estão sendo desenvolvidos para o gerenciamento das camadas superiores, onde residem as aplicações de rede.

2.1 – Sistemas Cliente / Servidor

O tópico a seguir, apresenta os conceitos das aplicações cliente / servidor e sua relação com o *Web Services*.

Um programa servidor, na maior parte do tempo fica em estado de espera, aguardando requisições dos clientes, essas requisições chegam em forma de mensagens, através da sessão de comunicação estabelecida. É comum um servidor estabelecer uma sessão dedicada com um determinado cliente, em outros casos os servidores criam um *pool* de sessões que são reutilizáveis.

Os servidores, para realizar um bom trabalho, precisam estar sempre solícitos aos clientes e estar preparados para os horários de maior tráfico, quando muitos clientes solicitam serviços ao mesmo tempo [Orfali, 1996]. A figura 1 ilustra uma arquitetura cliente / servidor em duas camadas.

Figura 1 - Arquitetura Cliente / Servidor duas camadas.

O papel de um programa servidor é servir muitos clientes, nesse contexto, um cliente é um programa que pede um determinado serviço (por exemplo, a transferência de um arquivo) a um servidor, outro programa ou computador. O

cliente e o servidor podem estar em duas máquinas diferentes, sendo esta a realidade para a maior parte das aplicações que usam este tipo de interação. É um processo ou programa que requisita serviços a um servidor.

O conceito cliente / servidor é utilizado para descrever comunicações entre processos de computação que são classificados como consumidores de serviços (clientes) e provedores de serviços (servidores). Alguns exemplos são *softwares* que acessam serviços ou bancos de dados localizados num servidor a partir de uma máquina *desktop*. [Umar, 1997]. Um *Web Service* pode se encaixar no mesmo exemplo uma vez que o serviço reside em uma máquina servidora, e outros programas ou até mesmo outros *Web Services* podem utilizar-se da aplicação.

2.2 – Middleware

O tópico a seguir apresenta os conceitos de *middleware,* base para elaboração do presente trabalho.

Embora um *middleware* seja difícil de se definir com precisão, basicamente é um *software* que permite que aplicações e usuários interajam uns com os outros através de uma rede.[Umar, 1997]

Middleware é a camada de *software* entre a rede propriamente dita e as aplicações, é o responsável pelos serviços como: identificação, autorização, serviços de diretório e segurança. Um *middleware* não é propriamente uma aplicação, mas facilita o uso de ambientes ricos em tecnologia da informação. Em outras palavras o *middleware* é a designação genérica utilizada para os sistemas de *software* que são executados entre as aplicações e os sistemas operacionais.

O objetivo do *middleware* é facilitar o desenvolvimento de aplicações, tipicamente aplicações distribuídas, assim como facilitar a integração de sistemas legados ou desenvolvidos de forma não integrada.

Aplicações tradicionais implementam vários serviços contidos no *middleware*, tratados de forma independente por cada uma delas. As aplicações modernas, no entanto, delegam e centralizam estes serviços na camada de *middleware*. Ou seja, o *middleware* serve como elemento que aglutina e dá coerência a um conjunto de aplicações e ambientes. [Carvalho, 2004]. A figura 2 ilustra a comunicação entre aplicações distribuídas através de um *middleware.*

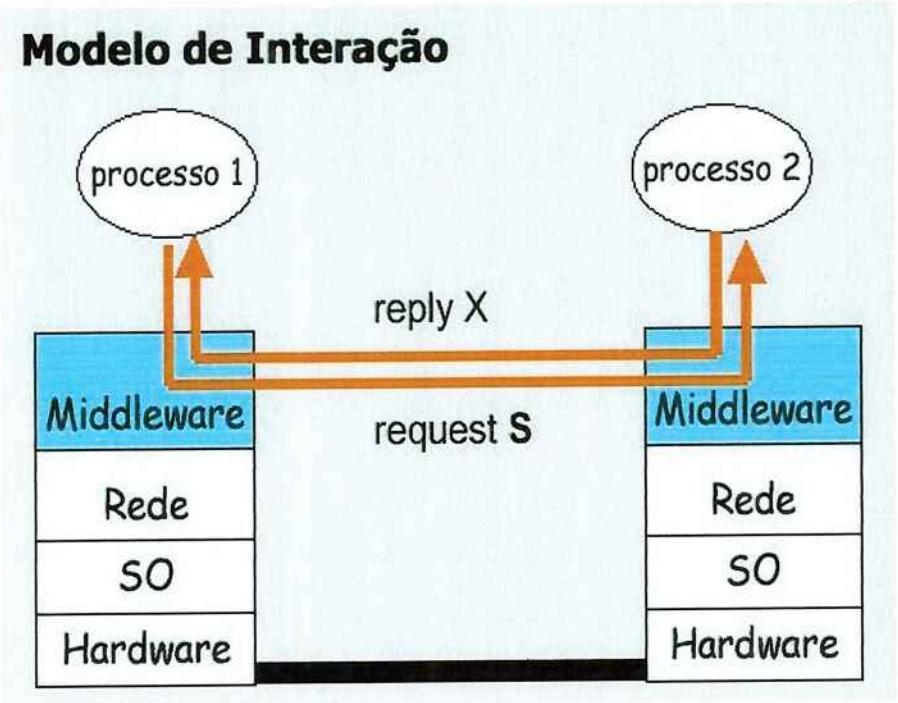

Figura 2 – Comunicação através de *middleware* [Rosa, 2004].

Segundo [David 2004], *middleware* é um conjunto robusto de serviços on-line que suportam uma ampla gama de aplicações institucionais e habilitadas a mudanças, é uma tecnologia padrão e bem definida, com interfaces bem definidas. O objetivo principal de um *middleware*, é diminuir a complexidade e heterogeneidade entre

os diversos sistemas existentes possibilitando a comunicação entre aplicações distribuídas.

O *middleware* está localizado entre as camadas de aplicação e transporte, implementa protocolos próprios que suportam os serviços fornecidos, geralmente protocolos de autenticação. A figura 3 ilustra o contexto do *middleware,* este utiliza API's do sistema operacional para facilitar a comunicação.

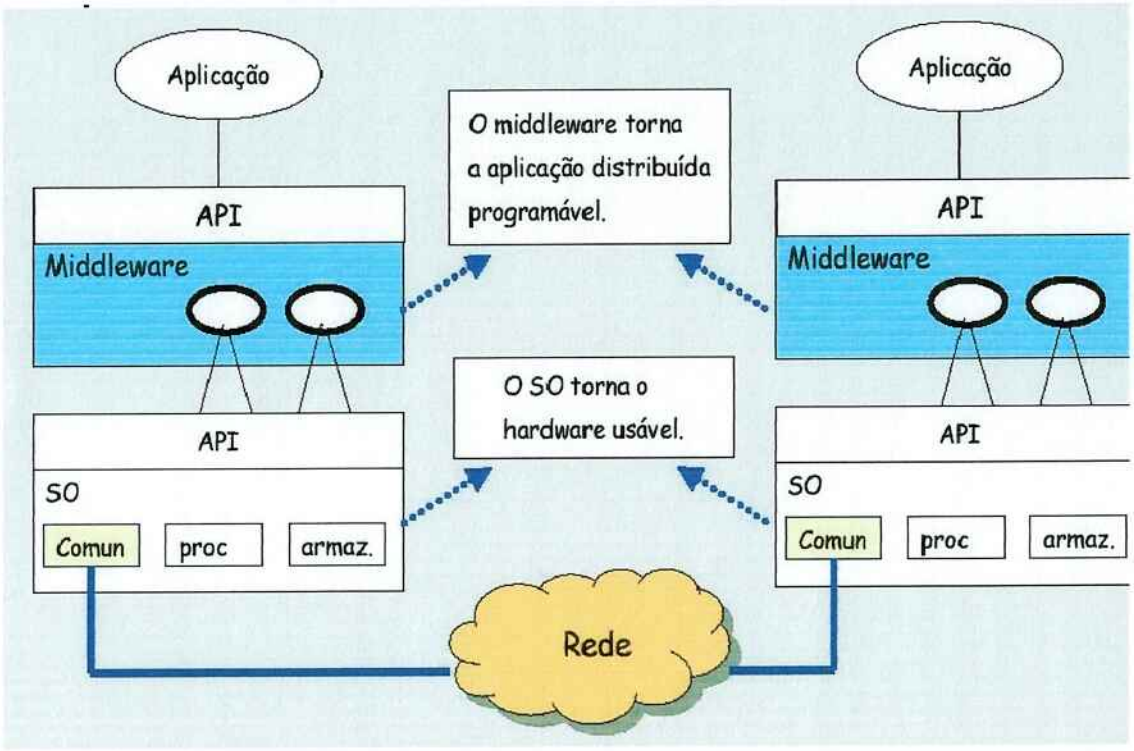

Figura 3 – Contexto do *midleware* [Rosa,2004].

As aplicações que demandam rapidez de processamento na execução de transações remotas, geralmente utilizam um *middleware* orientado a transações, estes provêem chamadas a procedimentos remotos e controle de transações, os serviços comuns são: comunicação síncrona / assíncrona e transação. A figura 4 ilustra um monitor de transações gerenciando as chamadas provenientes de aplicações clientes.

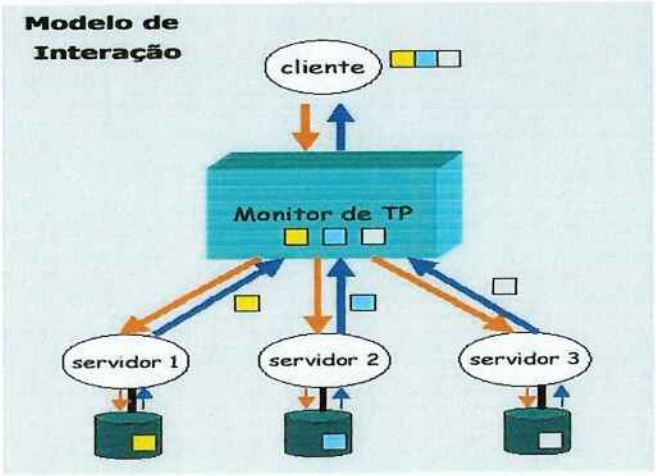

Figura 4 – *Middleware* orientado a transação [Rosa,2004].

As Chamadas de Procedimentos Remotos (RPC), foram uma das primeiras formas de comunicação entre aplicações remotas, mantém comunicação síncrona e geram vários problemas que devem ser tratados pelo programador, em sua maioria falhas de comunicação. A figura 5 ilustra o modelo de interação do *middleware* RPC.

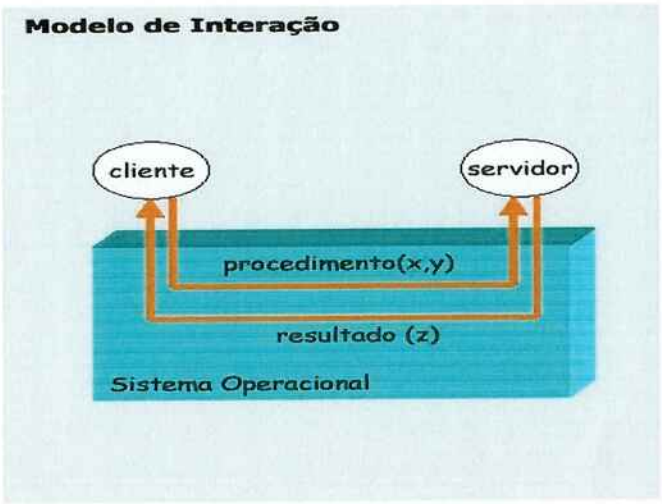

Figura 5 – *Middleware* RPC [Rosa,2004].

O *Middleware* Orientado a Mensagens, (MOM), provê comunicação através de mensagens entre aplicações de maneira assíncrona, não assume um transporte

confiável; entre os serviços estão: segurança, comunicação assíncrona propriamente dita, priorização de mensagens e replicação. A figura 6 ilustra o modelo de interação do *middleware* orientado a mensagem.

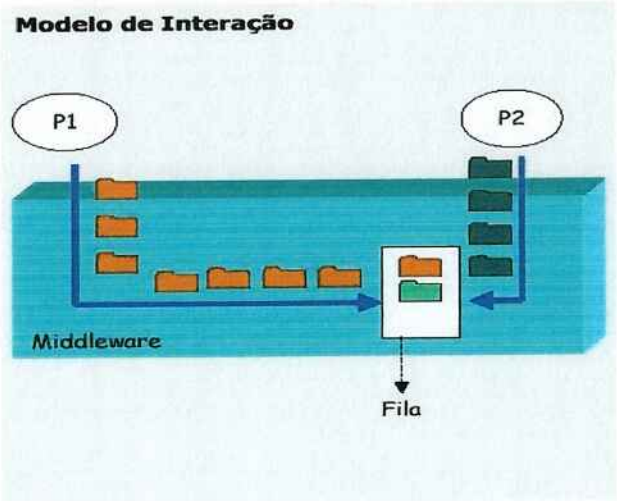

Figura 6 – *Middleware* orientado a mensagens [Rosa,2004].

O *middleware* baseado em objetos, mantém comunicação síncrona e assíncrona, assume um transporte confiável e o ambiente de programação é orientado a objetos. A figura 7 ilustra o modelo de interação do *middleware* orientado a objetos.

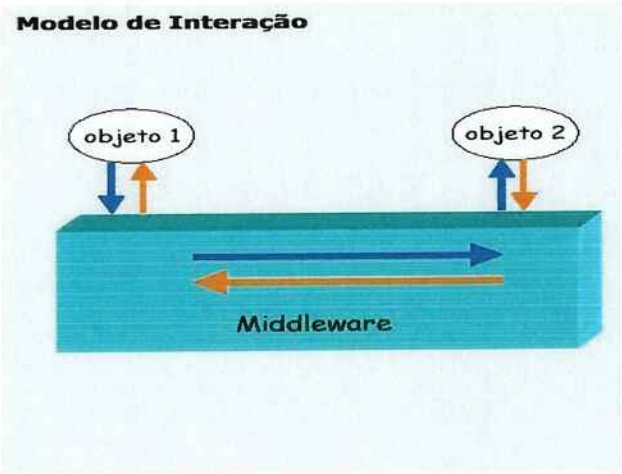

Figura 7 – *Middleware* orientado a objetos [Rosa,2004].

Os *Web Services*, são os *middlewares* característicos em aplicações *Web*. Atuam como interfaces para acessar outros serviços fornecidos por outros *middlewares*. Entre as tecnologias e protocolos utilizados estão: HTTP livre acesso através de *firewalls*, XML para codificação dos dados, SOAP transporte de dados (XML / HTTP), WSDL descrição do serviço e UDDI publicação e busca do serviço. A figura 8 ilustra um *middleware* para *Web*.

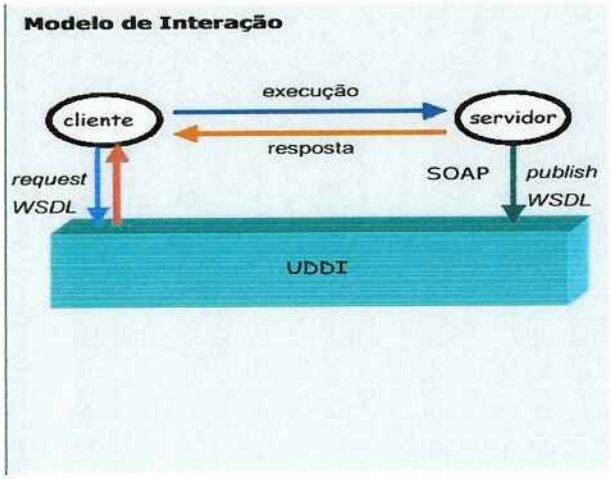

Figura 8 – *Middleware* para *Web* [Rosa,2004].

A figura 9 ilustra um *Web Service* utilizando serviços providos por outros *middlewares*.

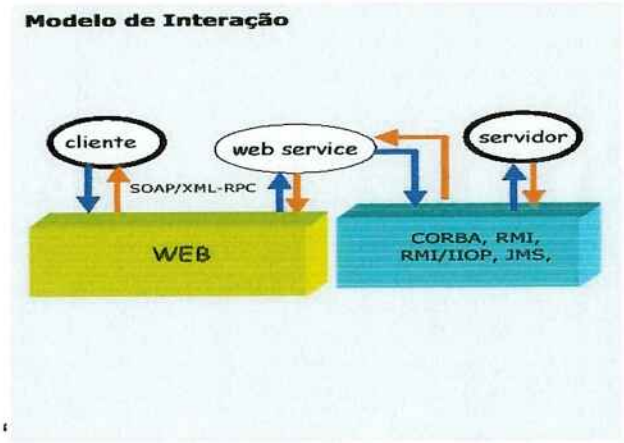

Figura 9 – *Web Services* e outros *middlewares* [Rosa,2004].

O tópico a seguir apresenta os principais *middlewares* atualmente utilizados.

2.2.1 – CORBA

A arquitetura CORBA, disponibiliza serviços básicos utilizados por todos os objetos distribuídos, as interfaces são independentes do domínio de aplicação, e suportam o ORB. Dentre os serviços: o serviço de persistência controla e armazena os estados do objeto, o serviço de *Trading,* controla o repositório dos serviços do sistema corporativo (páginas amarelas), o serviço de identificação permite encontrar os objetos baseados em nomes.

Os objetos de suporte a aplicações, *Common Facilities,* implementam serviços utilizados no nível de aplicação em domínios específicos. Os serviços horizontais implementam serviços de interface com o usuário, gerenciamento da informação, gerenciamento de sistemas, gerenciamento de tarefas e formação de documentos com componentes distribuídos. A figura 10 ilustra a arquitetura CORBA.

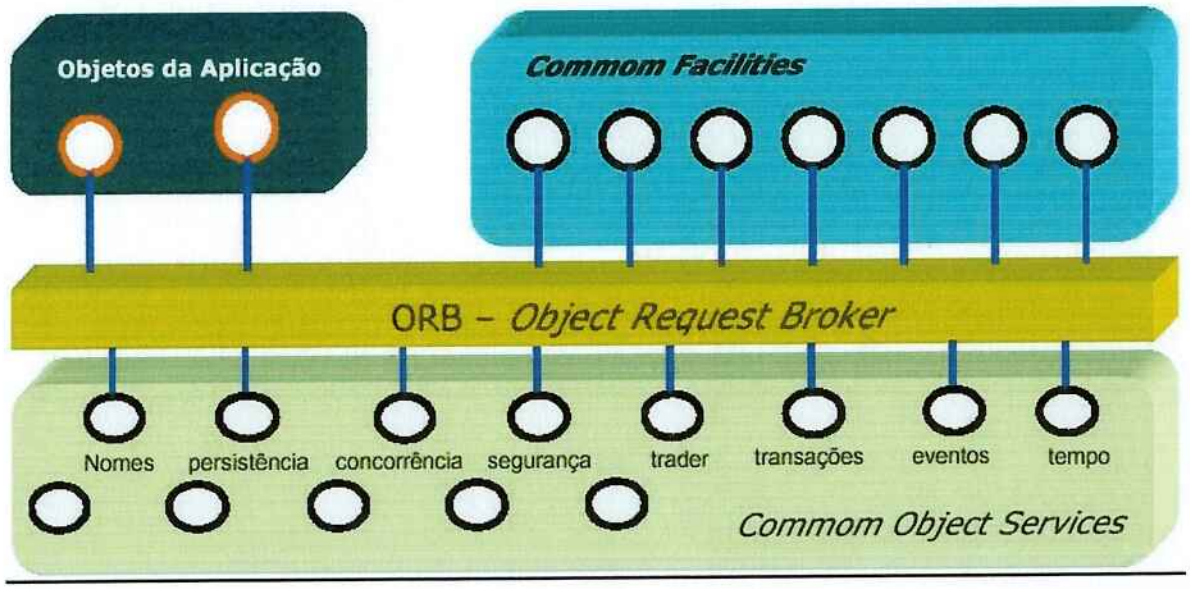

Figura 10 - Arquitetura CORBA [Rosa,2004].

Os serviços verticais, implementam serviços especializados, orientados a aplicações de domínios específicos: telecomunicações (TMN, Multimídia), manufatura (CIM, *Product Data Management*), médica (indexação de pacientes), negócios (definição de objetos para suportar processos de negócios).

O objeto aplicação (cliente), é composto de um conjunto de objetos básicos, comuns ou proprietários, em geral não são padronizados, e implementam aplicações específicas da corporação: sistemas de comércio eletrônico, sistemas de *Workflow* específicos e sistemas de gestão de documentos.

O núcleo do ORB, encaminha os pedidos entre o cliente e o objeto implementação, realiza a localização dos objetos recursos, envio de mensagens e a conexão, suporta vários mecanismos de manipulação de objetos: criação de objetos e serviços de diretório. A figura 11 ilustra o ORB.

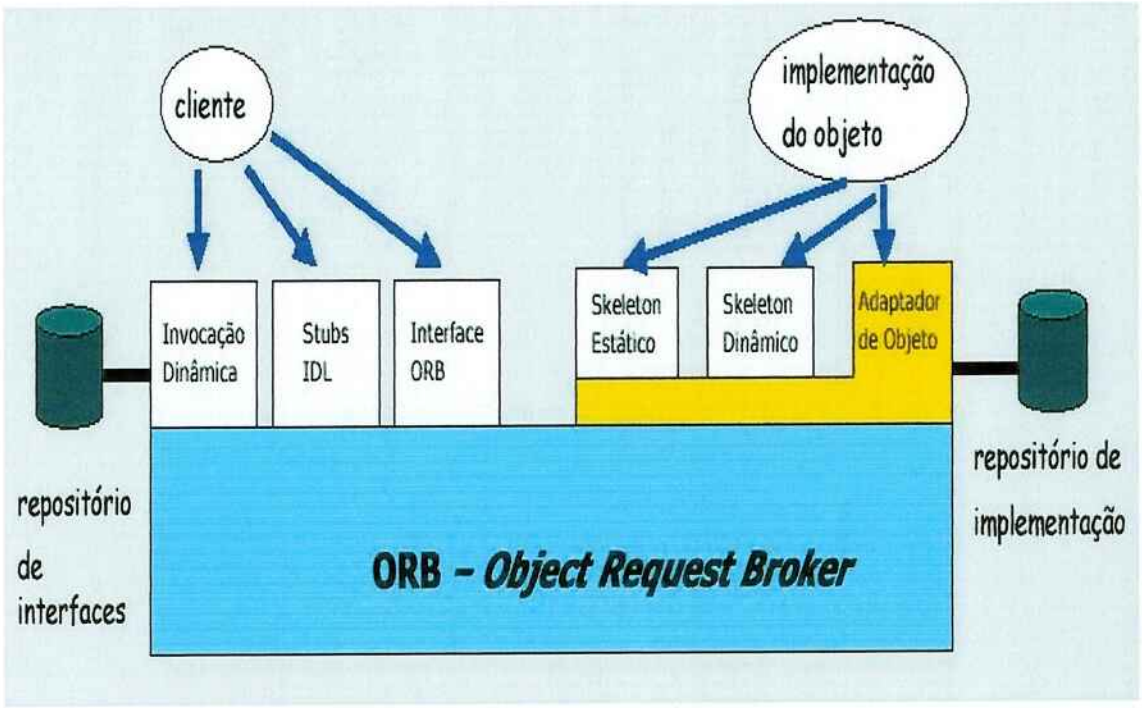

Figura 11 – Núcleo ORB [Rosa,2004].

A figura 12 ilustra o modelo de interação do ORB, o servidor registra o serviço, o cliente solicita o serviço, ORB retorna para o cliente, cliente invoca operação direto ao servidor, o servidor responde a solicitação.

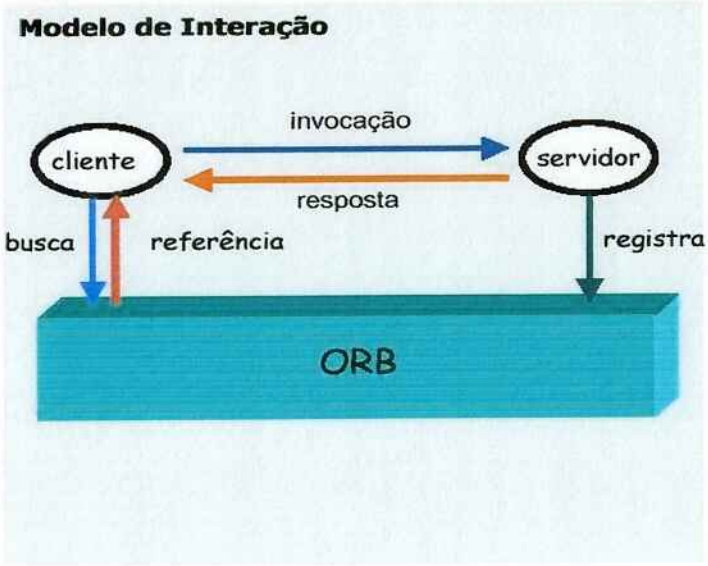

Figura 12 – Modelo de Interação ORB [Rosa,2004].

2.2.2 – DCOM

Em arquitetura de objetos distribuídos, distribuir componentes de *software* consiste em distribuir objetos. DCOM e CORBA, são opções de arquiteturas de objetos, o DCOM desenvolvido pela *Microsoft* é construído sobre as tecnologias ActiveX e OLE é uma tecnologia proprietária, de outro lado CORBA oferece uma arquitetura de objetos mais completa que o DCOM e a promessa de implementar objetos para serem compartilhados entre máquinas com diferentes sistemas operacionais.

O DCOM (*Distributed Component Object Model*) é um protocolo que possibilita a comunicação entre componentes de *software* diretamente sobre uma rede de

maneira confiável, segura e eficiente. Anteriormente chamado "*Network* OLE", DCOM é designado para uso em múltiplos protocolos de transporte de rede, incluindo protocolos de *Internet* tais como HTTP. DCOM é baseado na especificação DCE-RPC da *Open Software Foundation* e trabalhará com *applets* Java e componentes ActiveX através do uso do *Component Object Model* (COM) [Umeda, 1998].

O DCOM é utilizado na construção de soluções de software em três camadas, de forma a centralizar regras de negócio e processos, obter escalabilidade e facilitar a manutenção. DCOM funciona de forma transparente tanto para a aplicação cliente quanto para o servidor, que são codificados de acordo com o COM. A figura 13 ilustra a arquitetura DCOM.

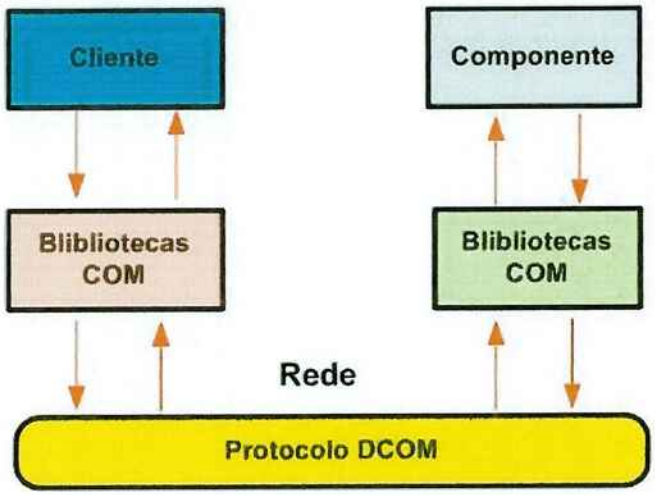

Figura 13 – Arquitetura DCOM.

2.2.3 – J2EE

O *software* moderno necessita de ferramentas modernas de desenvolvimento. Da mesma forma que existem as negociações entre bens e serviços, envolvendo dinheiro, ou outra forma de pagamento, existe também a negociação entre

aplicações de *software* as quais trocam dados entre si, mas para isso é necessário plataformas e ferramentas de apoio que possibilitem essa interoperação. A função principal da plataforma J2EE é oferecer um novo padrão para dar suporte as novas necessidades do *software* comercial, facilitando a troca de dados e informações.

A plataforma J2EE, possibilita a criação de aplicações distribuídas ou seja aplicações de *software* que são executadas em vários sistemas de computador ao mesmo tempo e que são complementares entre si, por ser uma plataforma de *software* moderna, utiliza-se de tecnologias recentes e também modernas que facilitam a integração, dentre elas XML e *Web Services*.

2.2.4 – .NET

O *Framework* .NET, oferece a infra-estrutura de desenvolvimento, distribuição e execução de aplicações, componentes e *Web Services*. Os serviços da *Web* baseados em XML permitem que os aplicativos se comuniquem e compartilhem dados através da *Internet* ou por uma *Intranet*.

O *Framework* .NET é dividido em três partes principais: a *Common Language Runtime* (CLR), ASP.NET e as classes do *Framework* a qual é uma grande biblioteca extensível onde incorpora todas as funcionalidades de uma plataforma operacional.

Com a plataforma .NET é possível trabalhar com independência na linguagem de programação, é possível trabalhar com várias linguagens diferentes no mesmo projeto devido à chamada IL (Intermediate Language), ou seja, uma linguagem intermediária para a qual todos os fontes são compilados e executados.

A plataforma .NET também suporta as novas tecnologias como XML e *Web Service* tecnologias promissoras na integração de dados e sistemas.

2.3 – Conexão aos Sistemas Legados

O tópico seguinte, apresenta os *Web Services* como opção de integração aos sistemas legados.

A maioria dos desenvolvedores, se ressente quando solicitados a trabalhar com sistemas legados. É comum a confusão de conceito ou seja por ser antigo não quer dizer que sejam ruins, outro fator é a sensação de que se está trabalhando com tecnologias ultrapassadas que não agregam nada em matéria de conhecimento.

Ocorre porém que um sistema que esteja funcionando por um longo tempo tem um valor alto, por mais ultrapassada que seja a tecnologia utilizada em seu desenvolvimento. Em geral os clientes não estão preocupados com a tecnologia utilizada, na verdade eles querem a informação correta e os produtos solicitados, não importando o quão moderno é o programa que controlou o processo.

A idéia de substituir uma aplicação que funciona bem ou seja escrever, testar e implementar é difícil de se vender à alta gerência das empresas em geral. Haja visto exemplos de aplicações financeiras como sistemas de contas correntes e fundos de investimentos que em sua maioria foram escritos em cobol e funcionam direito nos *mainframes*, sem contar o alto custo que um empreendimento dessa magnitude envolveria.

As novas tecnologias, fazem com que os profissionais de TI (Tecnologia da Informação), fiquem algumas vezes tentados a estabelecer a troca, apresentam as novas facilidades, porém existe um grande risco em se ter uma nova aplicação que substituirá um sistema já testado que funciona bem por algo que certamente rodará com problemas nos primeiros meses senão por anos.

O fato de ser difícil a substituição de sistemas legados, é apenas um pedaço de um problema maior que tende a mudar com o aparecimento de novas tecnologias de integração. Acessar informações proveniente de sistemas no *mainframe* para serem utilizados na plataforma baixa, é tarefa que demanda tempo, custos recursos e cronogramas apertados.

Disponibilizar e integrar tais informações no *site* da empresa também é empreendimento de porte. A simples utilização de um arquivo texto com informações de aplicações financeiras envolve diversas etapas no processo de desenvolvimento. Os *Web Services* [Potts,2003] envolvem perfeitamente os sistemas legados, independentemente da linguagem em que são escritos.

Os *Web Services* podem ser escritos de uma maneira que exija pouca ou nenhuma mudança na base de código legada. Eles podem ser escritos para interagir com o código antigo de modo semelhante ao que faria uma interface gráfica com o usuário (GUI). O resultado final é um moderno sistema distribuído que conserva toda eqüidade que a empresa construiu em sua base de código legada.[Potts, 2003]

Em situações concretas, é possível que os *Web Services* atuem freqüentemente como empacotadores em sistemas legados, como os sistemas de bancos de dados e aplicativos empresariais. Portanto, a dificuldade de distribuir um *Web Service* não está em distribuir o serviço propriamente dito, mas em garantir que funcione bem juntamente com os sistemas legados.[Jorgensen, 2002]

2.3.1 – Outras Aplicações de Web Services

As facilidades de integração através das novas tecnologias, em particular os *Web Services*, não poderiam deixar de incluir em seu leque de abrangência uma parte da computação que vêm crescendo consideravelmente: a computação móvel. Muitas [Umar, 1997] tecnologias emergentes estão sendo utilizadas no desenvolvimento de novas aplicações, e tais aplicações precisam integrar sistemas legados com as novas aplicações em desenvolvimento. As aplicações que utilizam rede sem fio por exemplo estão crescendo para atender a demanda de pessoas em trânsito.

A utilização das redes sem fio para acessar informações coorporativas, têm crescido rapidamente devido ao também crescimento de computadores móveis (*laptops*, e *pda's*). Por outro lado, muitos usuários destes dispositivos, não conseguem acesso a muitas aplicações coorporativas distribuídas tais como solicitações a bancos de dados remotos através de uma LAN. Isso se deve tipicamente a lentidão das redes sem fio, o congestionamento constante da rede e a maior suscetibilidade a erros. A computação móvel precisa de um *middleware* para agilizar o processo para que as aplicações possam rodar bem com ou sem fio. [Umar, 1997]

É grande o número de aplicações desenvolvidas e em desenvolvimento para atender necessidades de negócios e de entretenimento. Alguns produtos de *software* estão sendo utilizados para permitir que pessoas acessem desde planilhas de cálculo, *e-mail* e aplicações gráficas. Esta facilidade de integração é possível devido ao crescimento da *internet* e das tecnologias que apareceram para suportá-la, dentre elas XML e *Web Services*.

A computação móvel vem surgindo como uma nova proposta de paradigma computacional advinda da tecnologia de rede sem fio e dos sistemas distribuídos.

Nela o usuário, portando dispositivos móveis, como *palmtops* e *notebooks*, tem acesso a uma infra-estrutura compartilhada independente da sua localização física. Isto fornece uma comunicação flexível entre as pessoas e um acesso contínuo aos serviços de rede. É esperado que a computação móvel revolucione o modo como os computadores são usados hoje.[ISAM, 2004]

3 – WEB SERVICES

Um *Web Service* é uma arquitetura distribuída que utiliza os protocolos padrões, um dos protocolos utilizados é o SOAP (*Simple Object Access Protocol*). A especificação do protocolo SOAP foi submetida a W3C em maio de 2000 para ser utilizada como padrão na transferência de informações através de redes TCP/IP utilizando HTTP e XML.

Web Service é um sistema de *software* cuja interface pública é definida e descrita utilizando-se XML. Demais sistemas interagem com *Web Services* de acordo com as definições baseadas nas mensagens XML. Os *Web Services* oferecem protocolos de *internet* baseados em computação distribuída. [Holst, 2003]

Ao nível de negócio, os *Web Services* representam um maravilhoso mundo novo que, ao facilitar tremendamente a integração de aplicações, permite a curto prazo interligar as novas aplicações com as aplicações existentes e no médio prazo dará origem a um novo conceito de aplicações. [Silva, 2004]

Um *Web service*, portanto, é um componente de *software*, ou uma unidade lógica de aplicação, que se comunica através de tecnologias padrões de *Internet*. Esse componente provê dados e serviços para outras aplicações.

Essa tecnologia combina os melhores aspectos do desenvolvimento baseado em componentes e a *Web*. Como componentes, representam uma funcionalidade implementada em uma 'caixa-preta', que pode ser reutilizada sem a preocupação de como o serviço foi implementado. As aplicações acessam os *Web Services* através de protocolos e formatos de dados padrões, como HTTP, XML e SOAP.[Dextra,2003]

Os usuários de um *Web Service*, não precisam saber nenhuma informação a respeito da plataforma ou da linguagem de programação utilizada para a implementação do serviço, o que realmente se precisa saber é como emitir e receber mensagens SOAP (HTTP e XML).

Uma aplicação pode utilizar um *Web Service* instalado em uma *intranet*, para atender uma demanda interna, ou ser acessado através da *internet* por várias aplicações. Por ser acessível atendendo um padrão de mercado, um *Web Service* permite que sistemas heterogêneos trabalhem em conjunto como se fosse uma única aplicação.

Uma característica central dos *Web Services* é o alto grau de abstração que existe entre a implementação e o consumo do serviço. Utilizando o mecanismo de mensagens baseados em XML, o qual é o responsável pela criação e acesso do serviço, tanto o cliente de um *Web Service* quanto o provedor, não precisam ter conhecimento das informações que trafegam entres si além das entradas e saídas.

Os *Web Services*, estão proporcionando uma nova era no desenvolvimento de aplicações distribuídas. Quando os sistemas são fortemente acoplados, utilizando infra-estruturas proprietárias, os custos para as aplicações são extremamente altos. Os *Web Services*, possibilitam a interoperabilidade das aplicações em um novo nível. Com o avanço revolucionário da *Internet*, os *Web Services* serão a estrutura fundamental para ligar todos os dispositivos de computação.

Os *Web Services* foram projetados para ser simples acima de tudo, simples de implementar e de usar. Porém, a simplicidade tem seu preço. Existem vários recursos que os *Web Services* não têm, recursos que fazem parte de outros protocolos de troca de dados mais antigos e estabelecidos como COM / DCOM e

CORBA. Esses recursos envolvem gerenciamento de estado, segurança e processamento de transação.[Jorgensen, 2002]

3.1 – Arquitetura

O tópico seguinte apresenta a arquitetura básica de um *Web Service* e seus principais componentes.

O objetivo dos *Web Services* é a comunicação aplicação para aplicação através da *Internet / Intranet*. Mais especificamente, essa comunicação é realizada com a idéia de facilitar a integração de aplicação empresarial (EAI) e o comércio eletrônico, particularmente de empresa para empresa.[Potts, 2003]. A figura 14 ilustra a arquitetura geral de *Web Services,* cliente pesquisa em um *Web Service* (UDDI) os serviços disponíveis utilizando o protocolo SOAP.

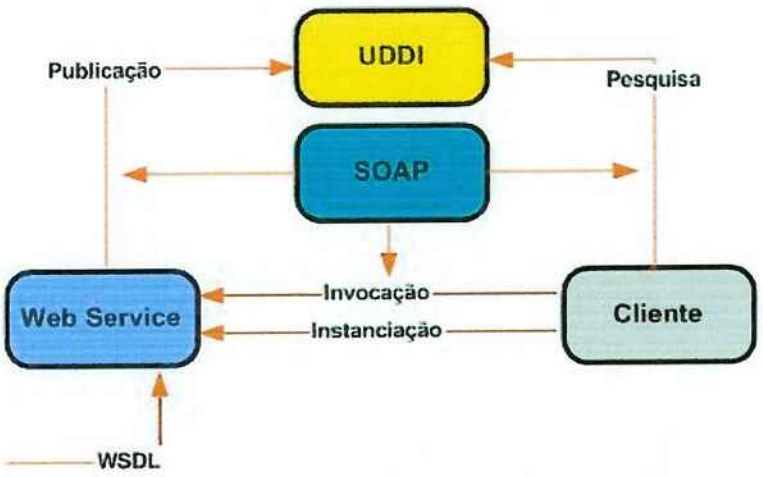

Figura 14 - Arquitetura geral de *Web Services*.

Os *Web Services* são construídos de forma a evitar muitos problemas que ocorrem com CORBA e DCOM.

- A idéia central dos *Web Services* desde o início foi a integração de aplicações o que dá suporte a transações síncronas e assíncronas.

- Os dados são codificados em XML.

- Os documentos XML são legíveis pelas máquinas e pelas pessoas.

- XML representa dados arbitrários, para esquemas XML complexos pode ser utilizado WSDL e sua descrição.

- *Web Services* são interoperáveis, e sua arquitetura permite aumento de interoperação com o tempo.

- Os fornecedores de *software* cooperam entre si, em torno dos *Web Services*, ou seja muitas ferramentas podem ser utilizadas para sua implementação.

Os *Web Services* são construídos sobre uma fundação de especificações e padrões diferentes mas cooperantes. Algumas razões para isso são históricas, mas a principal razão de existirem vários padrões é que nenhuma pessoa ou grupo possui um monopólio sobre o processo de especificação. Todos na comunidade têm voz para determinar a direção que a tecnologia tomará. Atualmente, as seguintes especificações são consideradas as principais especificações de *software* que, quando tomadas como um todo, compõem o que conhecemos como *Web Services*: [Potts, 2003]

- HTTP/1.1
- RFC 2965 : HTTP *State Management Mechanism* (*cookies*)
- SOAP 1.1
- UDDI versão 2.04 API e as outras especificações inferiores à UDDI 2.03
- WSDL 1.1

- XML 1.0 (segunda edição)
- XML *Schema* Part 1: Estruturas
- XML *Schema* Part 2: Tipos de dados

A figura 15 ilustra o fluxo do processo em *Web Services*.

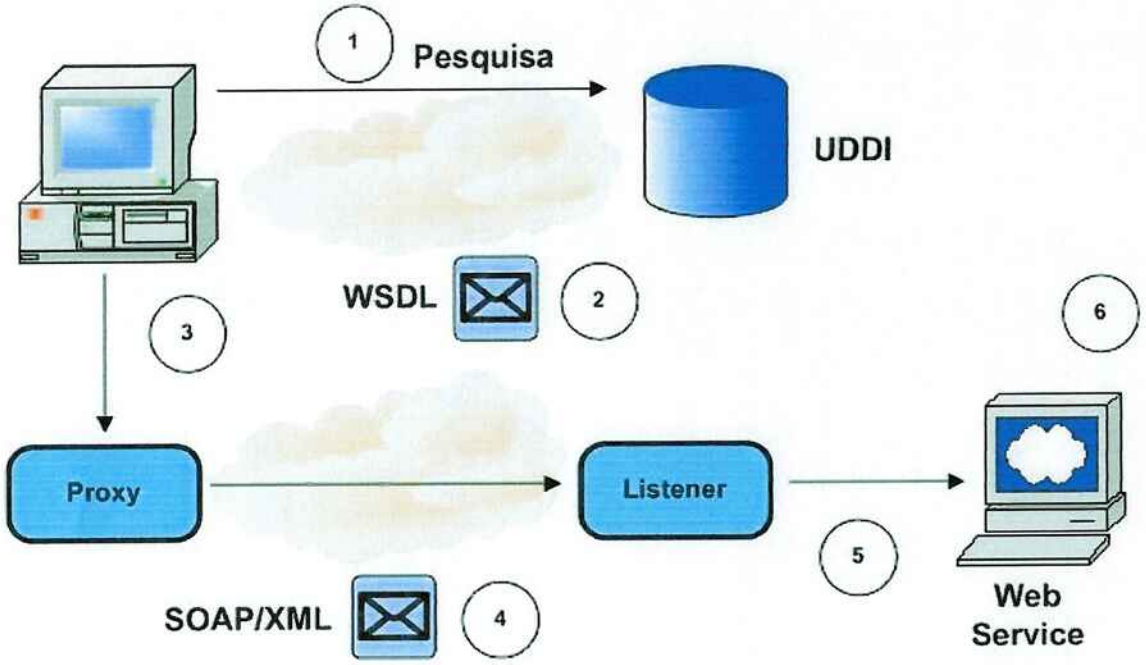

Figura 15 - Fluxo do processo em *Web Services*.
Fonte: http://www.zdnet.co.uk/i/z/tu/illo/WebServices.gif

3.2 – Tecnologias

O tópico a seguir apresenta as tecnologias utilizadas em conjunto com os *Web Services*.

As novas tecnologias vêm promovendo ultimamente uma nova abordagem para os sistemas de informação corporativos. Os grandes produtores de *software*

proclamam uma série de nomes diferentes para seus produtos que no fim todos chegam a um consenso ou seja *Web Services*. A idéia central dessa nova abordagem é que as corporações logo irão comprar suas tecnologias de informação como serviços providos através da *Internet*.

A nova tecnologia, que é genericamente conhecido por *Web Services*, tem alguns atributos peculiares. Diferentemente dos sites tradicionais, projetados para as pessoas interagirem com informação, os *Web Services* conectam aplicações diretamente com outras aplicações. E a idéia básica é que essa conexão se dê sem que seja necessário efetuar grandes customizações nas próprias aplicações. Além disso, uma das premissas fundamentais é que o padrão usado pelas conexões seja aberto e independente de plataforma tecnológica ou linguagens de programação.[Dextra, 2003]

O XML é extremamente simples. Ele define as regras para marcação de um documento, o fechamento apropriado de *tags*, a distinção entre maiúsculas e minúsculas, o aninhamento correto de elementos etc. No entanto, o documento XML sozinho não é tão útil. Ele funciona juntamente com uma série de tecnologias que permitem que os programadores manipulem documentos XML acompanhando a especificação XML. Essas tecnologias incluem a imposição da estrutura do documento, como o DTD (*Document Type Definition*) e o XML *Schema*; recuperação de dados como o Xpath (XML Path), DOM (*Document Object Model*) e SAX (*Simple API for XML*); transformação de documentos como o XSLT (*Extensible Stylesheet Language Transformation*).[Lee, 2004]

O aperfeiçoamento do processo de desenvolvimento de *software*, exige que novas tecnologias sejam pesquisadas e desenvolvidas, seja na forma de apoio automatizado, ou na forma de novas metodologias de trabalho. A constante busca do setor por ferramentas de apoio causa num primeiro momento ansiedade e preocupação aos profissionais de informática. A nova tecnologia a ser adotada

deve antes de tudo procurar prover acoplamento ao sistema existente e ao mesmo tempo facilitar seu aprendizado pelos recursos disponíveis.

O desenvolvimento de produtos inovadores necessitam de ferramentas de trabalho também inovadoras. A cegueira causada pela adoção de uma nova tecnologia é comum para os mais aficionados, antes de tudo deve haver sempre a grande preocupação de acoplamento ao legado, acompanhar as mudanças do mercado, pesquisando as novas tecnologias que sejam adequadas ao negócio desde que o mesmo não seja comprometido.

As mudanças que vêm ocorrendo nos processos e tecnologias de desenvolvimento de *software*, tais como a chamada plataforma .NET e J2EE, prometem facilidade e rapidez ao desenvolver *software* para qualquer dispositivo, como também integração total através de XML e *Web Services*. A razão principal por trás de tanto esforço é a *internet*, ou seja, a criação de aplicativos que possam ser executados em diversos tipos de equipamentos e interagir através da rede mundial.

3.2.1 – Web Service Description Language – WSDL

A implementação de uma interface que irá se comunicar com um Web Service, necessariamente precisa conhecer a sua estrutura para efetuar essa comunicação, nesse caso se torna necessário o documento WSDL (*Web Service Description Language*). Não existe em outras tecnologias nada análogo ou que tenha toda a eficiência de um documento WSDL. O documento WSDL é "escrito" utilizando XML como metalinguagem, descreve a interface da implementação de um determinado código, incluindo o nome dos métodos, os parâmetros e seus tipos, o retorno, o protocolo que o serviço vai se comunicar, as estruturas dos dados complexos, as URL's onde se encontra o serviço.

O WSDL é uma especificação que diz como um *Web Service* deve ser descrito, um documento WSDL contém todas as informações necessárias para a contratação do serviço. Os desenvolvedores, ou mesmo uma aplicação são capazes de ler os documentos WSDL criando mensagens claras que podem acessar um método ou métodos em determinado serviço.

De acordo com [Potts,2003] agora, temos uma especificação que mostra uma maneira de como descrever precisamente qualquer aplicação de *software*, inclusive que não seja um *Web Service*. Isto significa que é possível escrever um *software* capaz de gerar mensagens com base na lógica combinadas com as informações no documento WSDL.

Podemos afirmar que um documento WSDL, que é representado em XML descreve uma série de mensagens SOAP e a forma como funciona a troca dessas mensagens, em outras palavras, o WSDL está para o SOAP assim como o IDL está para o CORBA ou para o COM.

O próprio documento WSDL é um arquivo criado com uma gramática XML específica, desenvolvida para fazer a comunicação de metadados sobre os *Web Services* para atender todos os clientes potenciais. Um [Potts,2003] documento WSDL é um documento XML que obedece as regras de uma especificação. Todos os metadados sobre o *Web Service* são incluídos em algum lugar nesse arquivo. A estrutura existe para facilitar, descobrir o que os dados significam.

As aplicações desenvolvidas para utilizar-se da tecnologia de *Web Services*, seja na *Internet* ou numa *Intranet* corporativa, precisam utilizar um documento WSDL para descobrir como se vincular ao serviço. A WSDL é criada com alguma ferramenta adequada e publicada no servidor enviando o endereço a um serviço de diretórios. Os consumidores do serviço utilizam o registro para identificar se o *Web Service* atende suas necessidades.

O documento WSDL [Potts, 2003] é subdividido logicamente em dois agrupamentos diferentes: as descrições concretas e as abstratas. As quais podem ser chamadas de descrição funcional e não funcional. A descrição concreta é composta dos elementos que são orientados para vincular fisicamente o cliente ao serviço. A descrição desempenha as mesmas tarefas da *Interface Definition language* (IDL) na CORBA. A descrição abstrata é composta dos elementos que são orientados para descrever as capacidades do *Web Service*. A figura 16 ilustra um documento WSDL.

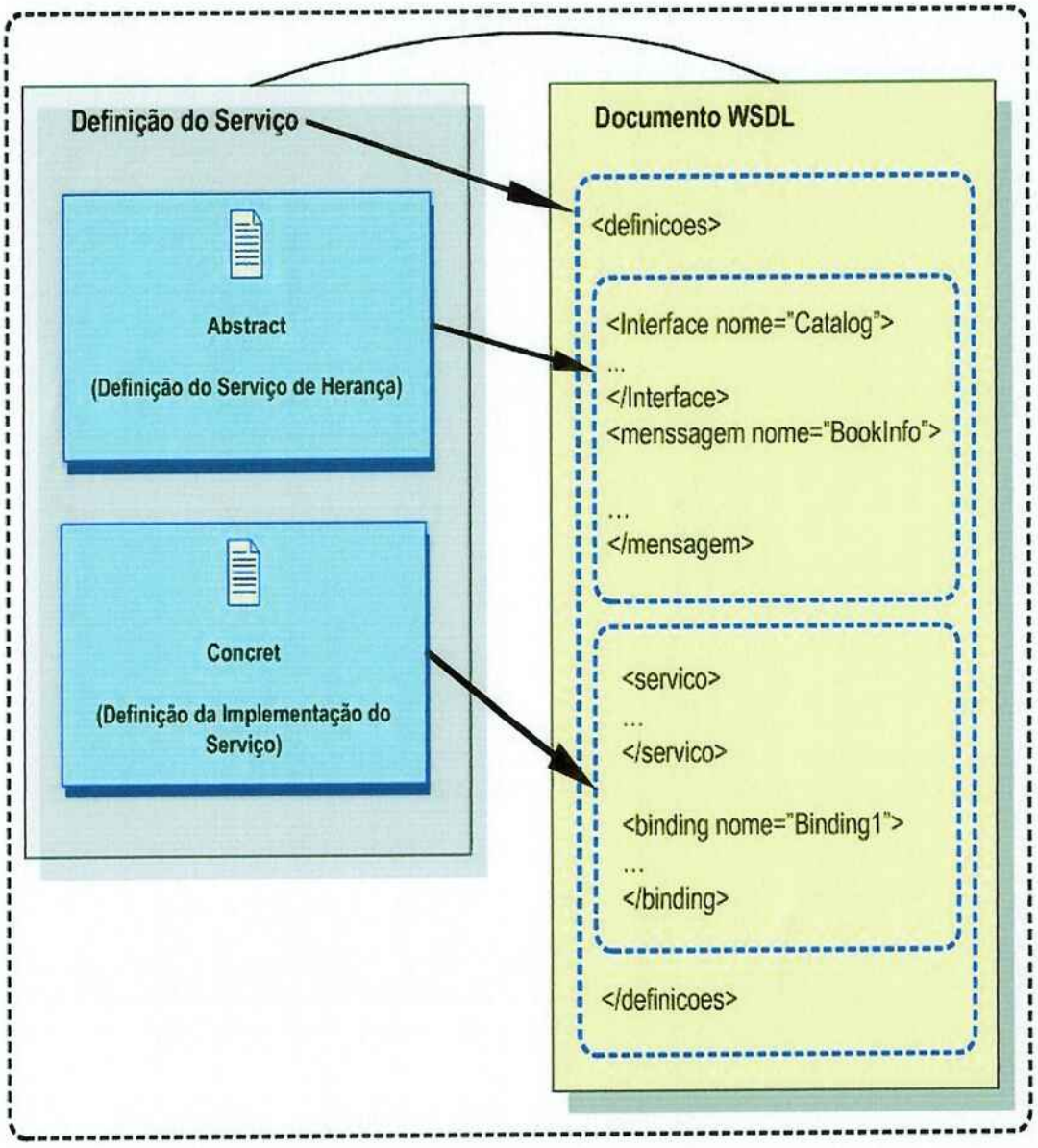

Figura 16 – Documento WSDL.
Fonte.: http://www.xmltc.com/ws-standards/wsdl.asp

3.2.2 – Extensible Markup Language – XML

O XML, devido a sua representação estruturada dos dados, tem se mostrado de fácil desenvolvimento e amplamente implementável. Os desenvolvimentos industriais na linguagem SGML (*Standard Generalized Markup Language*) mostraram a qualidade intrínseca e a força do formato estruturado em árvore dos documentos XML. O XML é um subconjunto do SGML, o qual é otimizado para distribuição através de redes baseadas em TCP/IP, e é definido pelo *Word Wide Web Consortium* (W3C), garantindo que os dados estruturados serão uniformes e independentes de aplicações e ou fornecedores.

O XML é capaz de codificar o conteúdo, as semânticas e as esquematizações para uma grande variedade de aplicações desde simples até as mais complexas, um simples documento, um registro estruturado tal como uma ordem de compra, um objeto com métodos e dados como objetos Java ou controles *ActiveX*, registros de dados tais como o resultado de uma consulta a bancos de dados, apresentação gráfica, como interface de aplicações de usuário, entidades e tipos de esquema padrões e URL's entre informações e pessoas em uma rede.

O dado em XML recebido pelo *software* cliente, pode ser manipulado, editado e visualizado sem a necessidade de uma nova viagem ao servidor, diminuindo assim a sobrecarga, reduzindo a necessidade de computação e reduzindo também a largura de banda para as comunicações entre cliente e servidor. A grande importância do XML, reside na capacidade de interoperação dos computadores em rede, por possuir um padrão flexível e aberto, independente de dispositivo. As aplicações podem ser construídas e atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados.

A crescente aceitação do XML e seu uso cada vez maior tendem a gerar uma quantidade enorme de dados nesse formato. Muitos desses dados são

transientes (passageiros) e não precisam ser armazenados. Em função disso, a tecnologia de bancos de dados tem se deparado com dois novos desafios: armazenar documentos XML internamente no SGBD, criando mecanismos para consultá-los e atualizá-los de forma eficiente, e integrar dados já existentes (legados) com dados no formato XML. [Bezerra, 2004]

3.2.3 – Simple Object Access Protocol - SOAP

O SOAP é um protocolo para troca de informações em ambientes de computação distribuídos. O protocolo SOAP é baseado em XML e consiste de três partes: o envelopamento que define a estrutura para descrever o que deve conter uma mensagem e como processá-la, um conjunto de regras específicas para codificação e uma convenção para representar chamadas e respostas remotas. O SOAP pode ser utilizado em combinação com uma infinidade de protocolos entre eles HTTP.

A característica mais convincente do protocolo SOAP é a possibilidade de ser implementado em diferentes plataformas de *hardware* e *software*. Isto significa que uma aplicação implementada com SOAP é capaz de ligar sistemas diferentes numa *Intranet* ou também na *Internet*. No passado foram feitas muitas tentativas de se implementar protocolos de comunicação comuns para o uso na integração de sistemas, mas nenhum deles foi tão difundido quanto o SOAP.

O SOAP é enxuto e de fácil implementação do que a maioria dos protocolos anteriores, DCE (*Distributed Computing Enviroment*) e CORBA levaram alguns anos para serem implementados e apenas poucas aplicações foram disponibilizadas. O SOAP por outro lado utiliza analisadores gramaticais (*Parses*) XML e bibliotecas HTTP para realizar o trabalho árduo, isso faz com que uma

implementação com SOAP seja concretizada em pouco tempo, já que existem mais de 30 implementações de SOAP disponíveis.

Os pedidos SOAP podem ser feitos em três padrões: GET, POST e SOAP. Os padrões GET e POST são idênticos aos pedidos feitos por navegadores *Internet*. O SOAP é um padrão semelhante ao POST, mas os pedidos são feitos em XML e permitem recursos mais sofisticados como passar estruturas e *arrays*. Independente de como seja feito o pedido, as respostas são sempre em XML. O XML descreve perfeitamente os dados em tempo de execução e evita problemas causados por inadvertidas mudanças nas funções, já que os objetos chamados têm a possibilidade de sempre validar os argumentos das funções, tornando o protocolo muito robusto.[Sant'Anna, 2004]

3.2.4 – Universal Description, Discovery, and Integration - UDDI

O UDDI na verdade é um tipo de *Web Service*, onde é possível publicar os *Web Services* e disponibilizá-los para uso geral. O UDDI funciona como as páginas amarelas dos *Web Services*. Através desse tipo de serviço, podemos procurar empresas que disponibilizem *Web Services* específicos para resolver um determinado problema, acessar informações a respeito do serviço oferecido e até contatar alguém para obter mais informação.

Apesar de ser um padrão de registro patrocinado por fornecedores, o UDDI vem se firmando como uma solução interessante para a publicação de *Web Services*. Existem grandes empresas de *software* por trás do assunto e estas estão cooperando entre si o que certamente consolida o serviço como padrão e o melhor: é grátis. O UDDI é uma coleção de *Web Services* duplicados, os diretórios públicos duplicam as informações para garantir que ao acessar um *Web*

Service, na verdade estamos acessando todos. A figura 17 ilustra o UDDI como um *Web Service*.

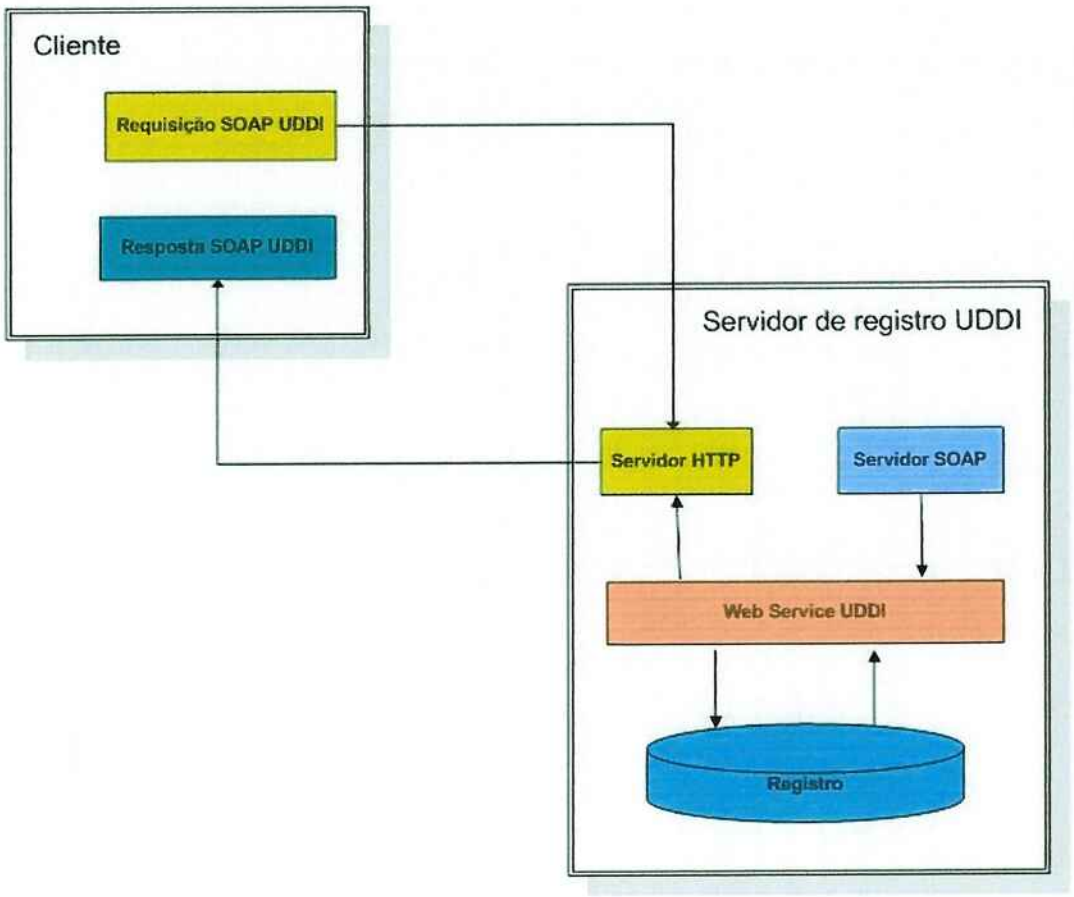

Figura 17 - UDDI como *Web Service*.

Segundo [Potts, 2003] o UDDI é descrito da seguinte forma: É um mecanismo padronizado e transparente para descrever serviços, descreve métodos simples para solicitar o serviço, especifica um registro central para solicitar o serviço. Outro recurso é que os registros são baseados em um modelo confederado pelo qual copiam uns aos outros.

O registro de um *Web Service* em determinado registro, será espalhado para outros registros no momento da próxima sincronização. Isso quer dizer que todos os *Web Services* registrados publicamente podem ser encontrados acessando

qualquer um dos registros públicos. A razão de vários fornecedores de *software* aceitarem o UDDI de comum acordo reside no fato de ele ser construído sobre os mesmos protocolos padrões SOAP dos *Web Services*.

O procedimento de registro de um *Web Service* no UDDI não é obrigatório, podemos criar os nossos *Web Services* e não os registrá-los no UDDI, o que pode ocorrer quando do desenvolvimento de um serviço para atender uma demanda interna da organização. Por outro lado uma aplicação de *e-commerce* para atingir um grande mercado certamente utilizará o serviço de registro público.

3.2.5 – Service-Oriented Architecture - SOA

Quando os *Web Services* começaram a ser disponibilizados, tornou-se imprescindível pensar em boas práticas para sua utilização. Começou-se então a falar em SOA. O fato de se levar em conta apenas os protocolos não é suficiente, é preciso saber as boas práticas e a abordagem para montar e explorar a infra-estrutura dos *Web Services* de forma correta. SOA na verdade, consiste em se ter a certeza de que se pode construir *Web Services* de qualidade, sendo estes a tecnologia que faz com que a SOA funcione.

A tecnologia em alguns aspectos não tem nada de novo. Em um passado não muito longe, olhava-se para SOA através de ferramentas como CORBA, que não foi universalmente adotada pois o custo era alto. Porém é primeira vez em que se vê toda a comunidade de TI de acordo sobre algo. Os gigantes da indústria como *Microsoft*, IBM, *Sun* e HP, todos estão de acordo em dar suporte a esses padrões.

A exploração dos *Web Services* através de SOA tende a se tornar tarefa complexa num futuro próximo, atualmente os *Web Services* estão sendo

utilizados para atender demandas simples, a complexidade aumentará na medida em que os ambientes se tornam mais ricos e complexos. A outra complexidade reside em como desenvolver aplicações com serviços ágeis de modo correto, que implementem transações em um ambiente heterogêneo.

3.3 – Bancos de Dados e XML

O próximo tópico apresenta os bancos de dados como consumidores e fornecedores de serviços através de *Web Services*.

A necessidade de integração de dados diversos, possibilitou o aparecimento de produtos de bancos de dados que tratam a informação em XML no modo nativo, é o caso do Tamino Servidor XML nativo. Este produto tem condições de armazenamento e troca de documentos XML, devido a essa característica o desempenho e a escalabilidade são melhores do que nos outros produtos de bancos de dados que trazem o tratamento a XML agregado. O Tamino, armazena dados multimídia e tem condições de fazer integração de informações em outras plataformas.

A decisão de projeto ao se adotar uma ferramenta de banco de dados que faz uso nativo XML passa a ser interessante em desenvolvimentos novos, por outro lado para aproveitar a infra-estrutura já existente, a utilização de produtos que fazem uso de XML agregado e a utilização de *Web Services* para a integração, pode se tornar uma solução atraente e de baixo custo

Uma vantagem dos SGBD's XML nativos é o fato de eles serem adequados para manipular documentos XML que não estão em conformidade com um esquema predefinido. Ou seja, no caso de documentos XML que não possuem uma estrutura rígida, os SGBD's XML nativos parecem ser uma escolha mais

apropriada do que os SGBD's habilitados a XML. Porém é necessário levar em conta o investimento em um novo tipo de SGBD e o fato de que esses SGBD's não são adequados para manipular dados relacionais. [Bezerra, 2004]

Os principais SGBD's do mercado têm se esforçado para ajustar o seu modelo de dados para oferecer suporte ao gerenciamento de dados XML, tais produtos passaram a ser denominados: SGBD's habilitados para XML. Os produtos habilitados para XML armazenam os documentos por meio de mapeamento de seus elementos para uma estrutura de dados definida especificamente para XML. Os SGBD's habilitados para XML definem novos tipos de dados abstratos específicos a XML que facilitam o armazenamento e o processamento dos documentos.

O armazenamento de documentos XML em SGBD puramente relacionais, pode ser feito de duas maneiras: utilizando um campo do tipo CLOB, onde o SGBD se comporta apenas como um repositório de dados sem o conhecimento da estrutura de um documento XML, ou através da leitura de um documento XML juntamente com seu esquema (DTD), nesse caso o documento é analisado por um processador XML e os dados são armazenados em tabelas relacionais.

As *Stored Procedures* são um modelo essencial de programação em bancos de dados, elas permitem uma separação clara da lógica que deve rodar nos bancos de dados da lógica do negócio a qual se localiza na camada intermediária. *Stored Procedures* podem ser programadas utilizando a linguagem específica de cada banco de dados, (e.g. PL/SQL), ou em uma linguagem portável e independente de banco de dados como JAVA.

A beleza dos *Web Services* é que se torna possível alavancar investimentos em *Stored Procedures*, disponibilizá-las como *Web Services* sem ter que se

preocupar com a linguagem em que estas são implementadas.[Mensah, 2004]. A figura 18 ilustra um *framework* genérico de banco de dados e *Web Services*.

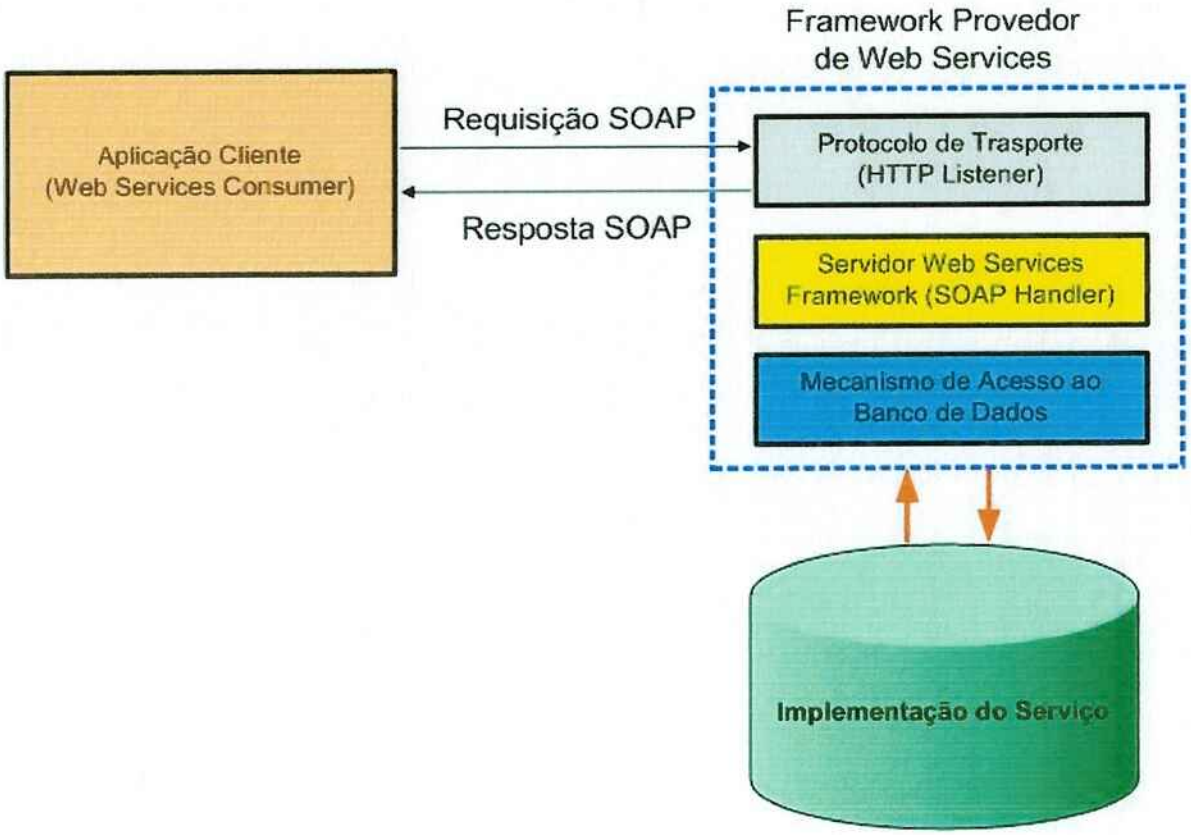

Figura 18 - *Framework* de Banco de Dados e *Web Service*.
Fonte.: http://www.sys-con.com/webservices/articleprint.cfm?id=515

4 – ARQUITETURA DE INTEGRAÇÃO

O modelo ora proposto visa exclusivamente facilitar o desenvolvimento e integração de aplicações as quais fazem solicitações a bancos de dados. Para obter sucesso em um mercado heterogêneo e competitivo, um projeto de *software* profissional precisa ser pensado com cautela, levando-se em conta seu futuro ciclo de vida, simplesmente atender uma necessidade emergente, ou resolver uma tarefa complexa não basta, as necessidades de futuros acoplamentos e mesmo integrações com sistemas legados e com novas tecnologias deve fazer parte do projeto inicial.

Todo o desenvolvimento de *software* pode ser caracterizado como um ciclo, no qual são encontrados quatro estágios distintos: situação atual, definição do problema, desenvolvimento técnico e integração da solução. [Pressman,2002]

Os componentes propostos na arquitetura de integração, executam tarefas distintas, os módulos envolvidos cada um têm sua responsabilidade para o funcionamento do todo. O módulo de controle responsável pela administração do *middleware* controlará cada parte e estas individualmente executarão as tarefas solicitadas, ao processar uma consulta o módulo de controle terá condições, entre outras coisas de verificar se a ação obteve sucesso mantendo uma trilha de auditoria.

4.1 – Requisitos Básicos

O tópico seguinte apresenta os requisitos básicos da proposta, ou seja aplicações clientes fazendo acesso aos bancos de dados via *Web Service*.

A utilização de ambientes de desenvolvimento comerciais como *front-end* para desenvolvimento de aplicações cliente / servidor oferecem dispositivos de software os quais proporcionam acesso muitas vezes nativo aos bancos de dados de diversos fabricantes, além de suportar conexões via ODBC e JDBC.

Algumas aplicações destinadas ao mercado corporativo precisam estar preparadas para utilizar o ambiente oferecido pelos seus clientes, os quais podem optar pelos diversos produtos de bancos de dados oferecidos pela indústria.

A figura 19 ilustra a proposta do trabalho, uma arquitetura de integração de bancos de dados utilizando *Web Service* como um *middleware*. As aplicações clientes provenientes de diversas plataformas de *software*, fazem a conexão e acesso aos bancos de dados utilizando uma camada intermediária, **WSIntegraBancoDeDados,** o *middleware* baseado em *Web Services* propriamente dito para interpretação das consultas.

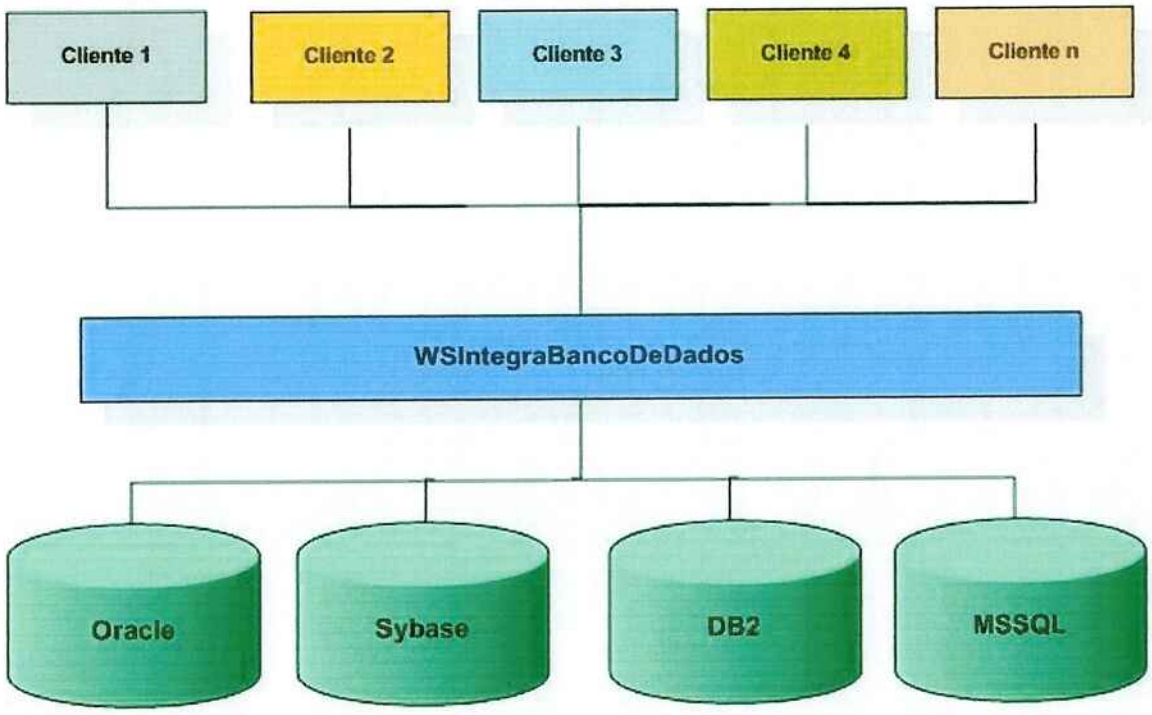

Figura 19 - Arquitetura de Integração.

As solicitações aos bancos de dados utilizam a SQL (*Structered Query Language*), uma linguagem simples que provê os serviços necessários para a manutenção das informações.

Apesar do poder e simplicidade e do esforço para tornar a SQL uma linguagem padronizada, ainda assim muitos fabricantes de bancos de dados acrescentam características específicas à linguagem em seus produtos, dificultando a escrita de uma consulta capaz de se executada em qualquer banco de dados.

A seguir a lista de requisitos básicos para o desenvolvimento da arquitetura proposta.

Requisitos Não funcionais.

Requisitos	Descrição
REQ001	O sistema deve ser hospedado em um servidor *Web*.
REQ002	O sistema deve trabalhar desconectado.
REQ003	O sistema deve utilizar os protocolos padrões.
REQ004	O sistema deve ser escalável.
REQ005	O sistema deve estar disponível a qualquer momento.
REQ006	O sistema deve ficar imune a acessos indesejados.
REQ007	O sistema deve ser acessado por usuários com perfil de operador ou superior.
REQ008	Os clientes devem ser identificados a partir do *log in*.

REQ009	O tempo de resposta das requisições devem ser inferiores à 5 segundos.

Requisitos Funcionais

Requisitos	Descrição
REQ001	O sistema deve identificar o usuário.
REQ002	O sistema deve validar o usuário.
REQ003	O sistema deve autenticar o usuário.
REQ004	O sistema deve abrir seções por usuário.
REQ005	O sistema deve controlar o número de usuários.
REQ006	O sistema deve controlar o número de conexões por usuário.
REQ007	O sistema deve controlar concorrência.
REQ008	O sistema deve controlar fila de requisições.
RQE009	O sistema deve priorizar solicitações.
REQ010	O sistema deve validar metadados.
REQ011	O sistema deve processar metadados.
REQ012	O sistema deve controlar transações.
REQ013	O sistema deve se certificar se a transação ocorreu com sucesso.
REQ014	O sistema deve identificar as regras
REQ015	O sistema deve validar as regras.
REQ016	O sistema deve processar as regras.
REQ017	O sistema deve validar a comunicação.
REQ018	O sistema deve identificar a fonte de dados.
REQ019	O sistema deve abrir conexão com a fonte de dados.
REQ020	O sistema deve executar consulta SQL

	genérica.
REQ021	O sistema deve executar *procedure* armazenada genérica.
REQ022	O sistema de executar atualizações genéricas.
REQ023	O sistema deve tratar consultas genéricas.
REQ024	O sistema deve tratar *procedure* armazenada genérica.
REQ025	O sistema deve tratar atualizações genéricas.
REQ026	O sistema deve fechar conexão ao final da solicitação.
REQ027	O sistema deve solicitar consultas genéricas.
REQ028	O sistema deve solicitar *procedure* armazenada genérica.
REQ029	O sistema deve solicitar atualizações genéricas.
REQ030	O sistema deve manter trilha de auditoria.
REQ031	O sistema deve tratar erros de conexões.
REQ032	O sistema deve de tratar erros de solicitações.
REQ033	O sistema deve tratar erros de conversão.

4.2 – Projeto Orientado a Objetos

O tópico a seguir apresenta o modelo orientado a objetos da proposta.

A proposta do presente trabalho exige uma abordagem orientada a objetos, não por ser o jargão da moda e sim por permitir uma visualização mais detalhada do serviço proposto. O modelo orientado a objetos, nos permite um acoplamento mais preciso às aplicações clientes que farão uso do serviço, as classes internas de cada pacote, fazem a comunicação entre si, delegando tarefas específicas para cada método em particular.

A figura 20 ilustra o diagrama de caso de uso da arquitetura, um usuário comum ou seja o que solicita o serviço, faz acesso ao *Web Service,* o mesmo através do módulo de controle se inter-relaciona com os demais componentes da arquitetura. Um ator, usuário administrador se encarregará de monitorar a aplicação durante seu tempo de vida em memória.

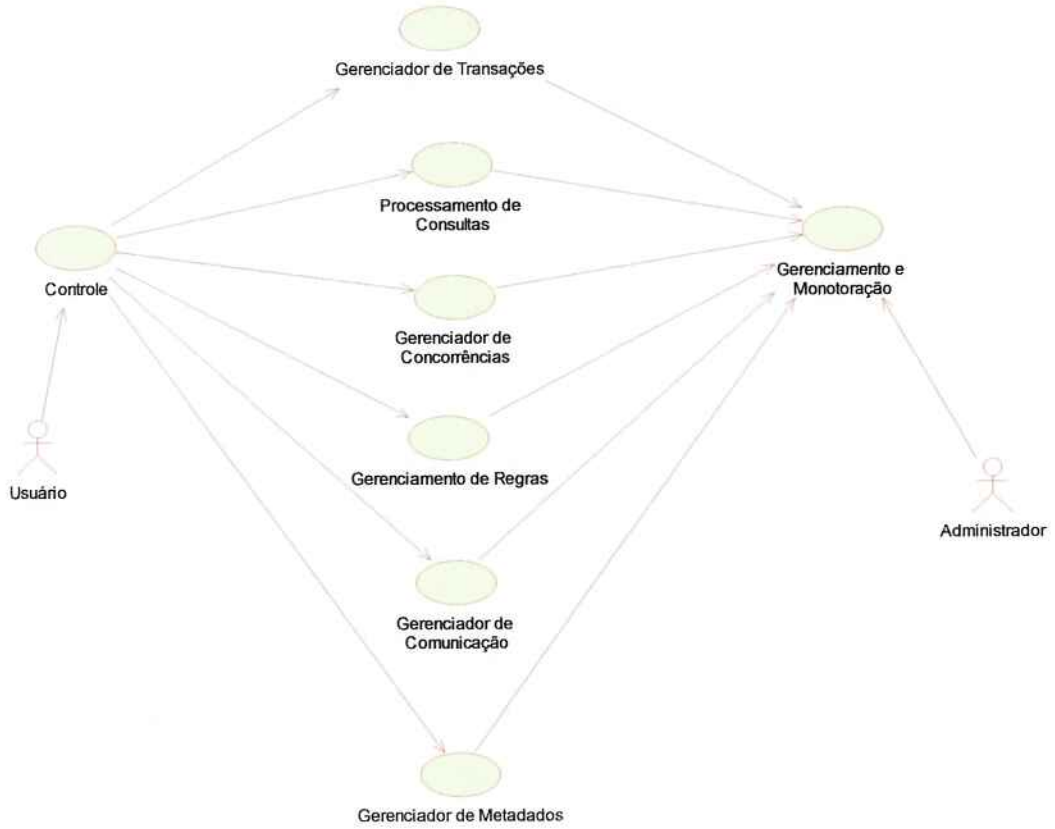

Figura 20 – Caso de Uso - Geral.

A figura 21 ilustra um diagrama de caso de uso de um pacote em particular: o processamento de consultas, este recebe as ordens do módulo de controle e opera as ações solicitadas, no módulo de consulta as solicitações são tratadas, para atender cada banco de dados específico.

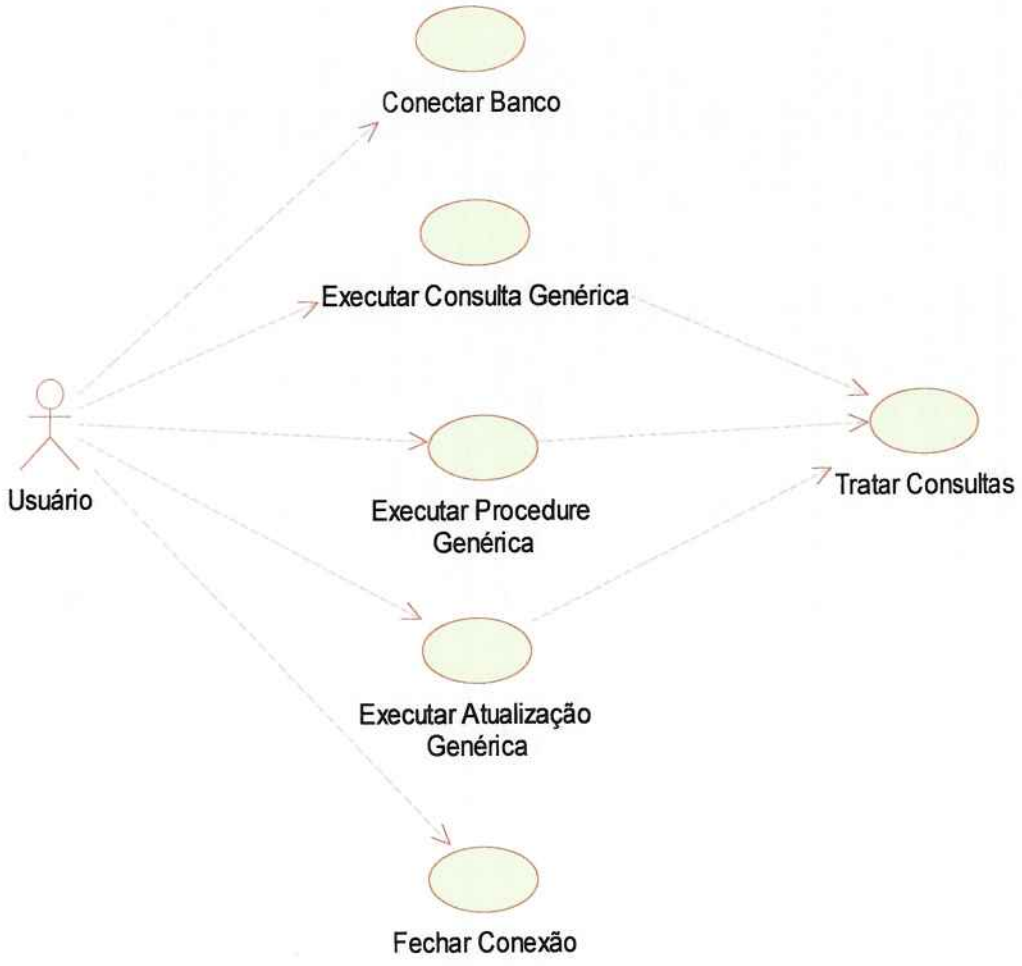

Figura 21 – Caso de Uso – Processador de Consultas.

A figura 22 ilustra um o diagrama de classes da arquitetura, apresenta as funções e seus relacionamentos, ressaltando sempre o módulo de controle com suas classes internas delegando tarefas aos outros módulos, e estes por sua vez executando as ordens e retornando ao chamador.

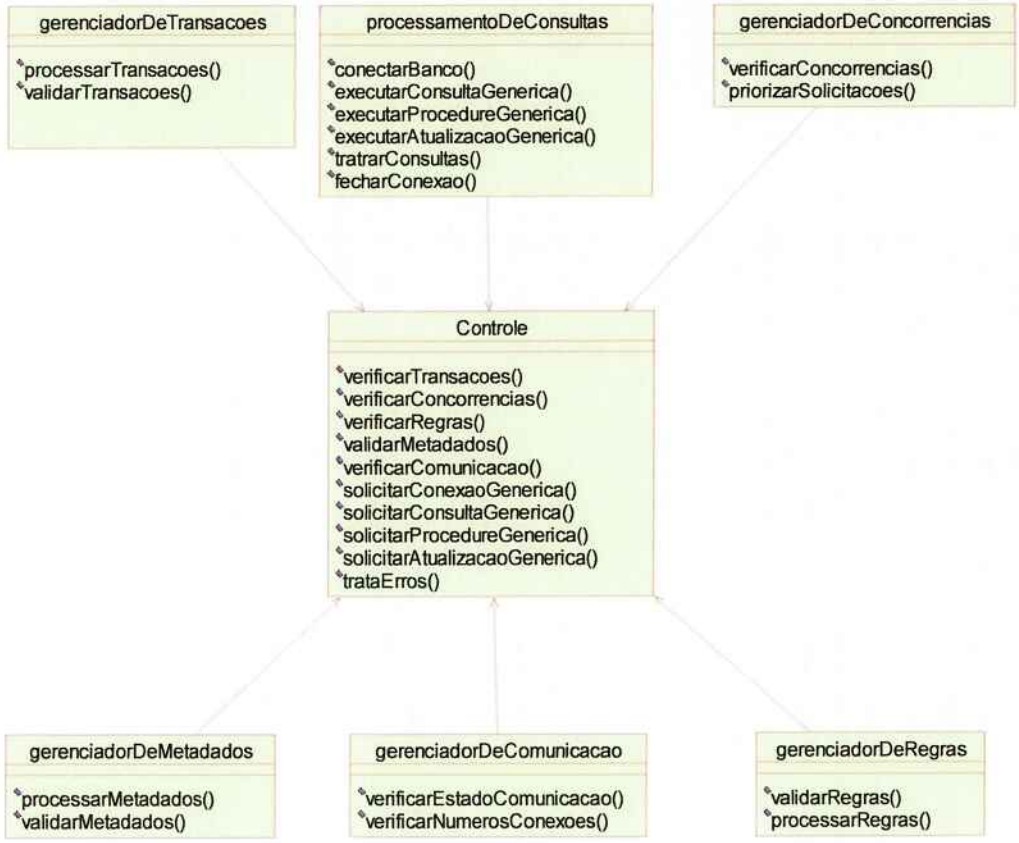

Figura 22 – Diagrama de Classes.

Figura 23 – Diagrama de Estados.

A figura 23 ilustra um diagrama de estados da arquitetura, as solicitações são feitas pelos clientes, é solicitado o serviço através do *Web Service* o cliente ficará aguardando caso haja muitas solicitações, o estado pode alternar entre sucesso da operação ou um retorno por parte *software* informando o cliente que a solicitação foi cancelada.

Para concluir a análise, através de um modelo orientado a objetos, é possível uma visualização em nível macroscópico, de como poderia ser implementada a arquitetura proposta, é bem verdade que mais artefatos e ferramentas da UML podem e devem ser utilizadas para que o desenvolvimento da aplicação não incorra em surpresas desagradáveis.

4.3 – Arquitetura da Proposta

O próximo tópico apresenta a arquitetura da proposta propriamente dita.

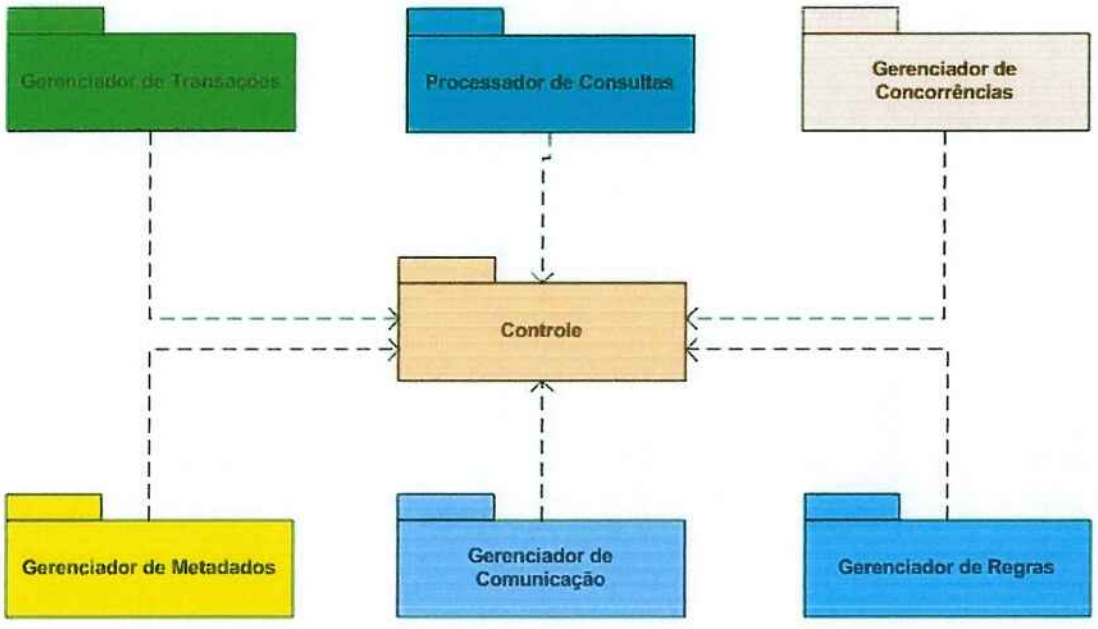

Figura 24 – Componentes da arquitetura.

A figura 24 ilustra a arquitetura proposta, o módulo de controle faz o gerenciamento do *middleware* delegando tarefas aos demais módulos. Estes por sua vez executam as solicitações retornando ao módulo de controle um aviso de ocorrência de sucesso ou impossibilidade de execução.

O modelo proposto, através do módulo de controle, será capaz de gerenciar os demais módulos controlando assim todo o fluxo de informações mapeando os comandos aos componentes de uma dada configuração.

O gerenciamento de transações será o responsável pela comunicação ao sistema de gerenciamento de transações do banco de dados utilizado, certificando-se assim se uma dada transação ocorreu com sucesso ou não, possibilitando a volta ao estado anterior e posteriormente informando o módulo de controle.

O gerenciamento de concorrências, será o responsável por administrar a fila de solicitações ao serviço, em um ambiente corporativo onde haja processamento em grande quantidade este módulo terá com função principal garantir que todas as solicitações sejam executadas com sucesso.

O módulo encarregado pelo gerenciamento da comunicação, terá como principal tarefa, monitorar tanto a disponibilidade da rede quanto a do banco de dados. No momento em que haja número excessivo de conexões ao serviço este enviará a informação ao módulo de controle que por sua vez informará a aplicação cliente.

O gerenciador de metadados, será o encarregado de manter uma documentação relacionada às consultas e os tipos de dados, de posse dessas informações, que serão enviadas ao módulo de controle para serem utilizadas pelo processador de consultas e posteriormente envidas aos bancos de dados executando-se assim solicitações com os tipos de dados corretos.

O gerenciador de regras será o módulo com a função de definir qual o servidor de banco de dados deverá ser utilizado, quais os direitos de cada usuário, quais as características específicas de cada produto e suas peculiaridades, e assim comunicar o módulo de controle que deverá acionar uma rotina no módulo de processamento de consultas, capaz de traduzir as solicitações e enviá-las ao destino correto, o retorno da solicitação recebido através do processamento de consultas será enviado ao módulo de controle e este passará o resultado em XML para a aplicação cliente.

O processador de consultas, será o encarregado de receber as consultas, adaptá-las e convertê-las com base nas regras existentes no módulo gerenciador de regras, acionar o módulo de conexão, informando a este qual o SGBD será utilizado, e fazer a solicitação já adaptada para aquele produto específico, obtendo os dados em XML como já mencionados.

4.4 – Funcionamento da Proposta

O próximo tópico apresenta o funcionamento da proposta, um *Web Service* foi desenvolvido para mostrar o funcionamento da arquitetura.

A proposta principal deste trabalho, é possibilitar que uma aplicação cliente utilize todo o poder de processamento dos bancos de dados independente de fabricante, para isso utilizaríamos um *Web Service* como *middleware* para a integração.

O *Web Service* ficará hospedado em um nó da rede e será o responsável por direcionar as solicitações aos bancos de dados provenientes de aplicações clientes, obtendo o resultado em XML. A infraestrutura de rede existente e a

utilização dos protocolos padrões, TCP/IP, HTTP e SOAP contribuem para facilitar o desenvolvimento do *middleware* em questão.

A figura 25 nos mostra um *Web Service* desenvolvido na plataforma .NET utilizando a linguagem de programação C# (*C Sharp*) fazendo acesso a um banco de dados.

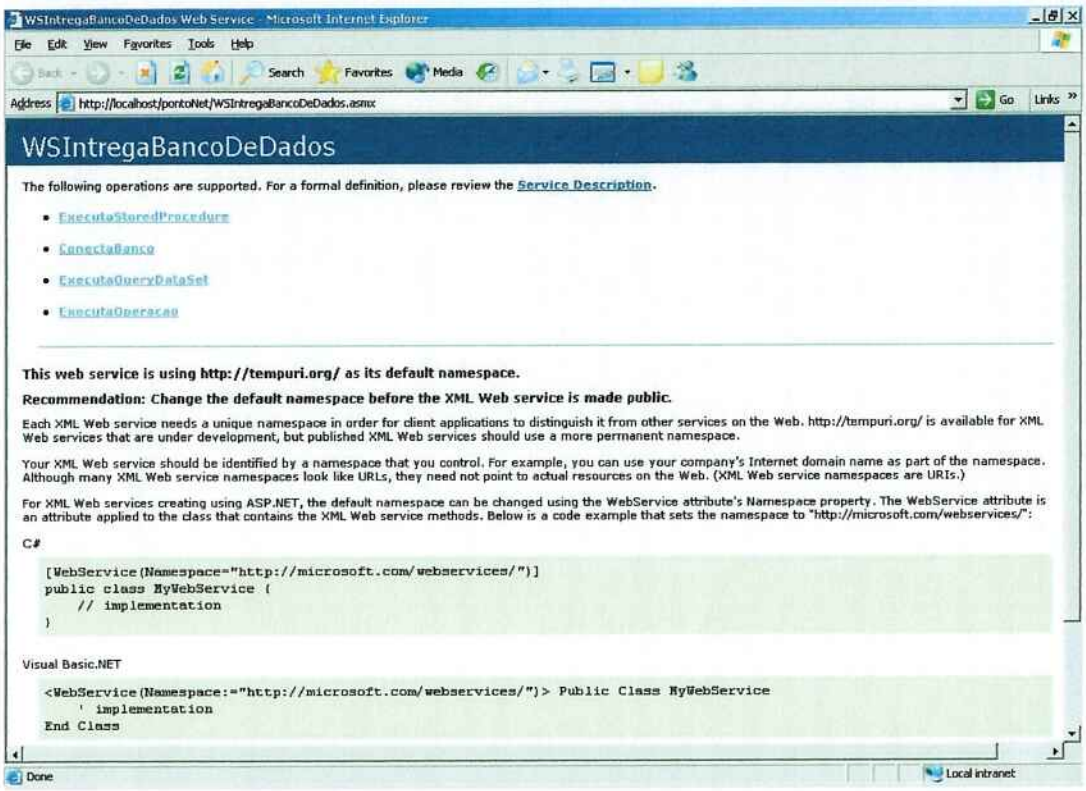

Figura 25 - *Web Service* na plataforma .NET.

A aplicação cliente, através de parâmetros contidos em um arquivo XML de configuração, tais como: qual o banco de dados a ser utilizado, qual a localização do servidor de banco de dados na rede e demais regras as quais serão acessadas pelo módulo de regras, fará o acesso ao *Web Service* e este por sua vez direcionará a solicitação para o banco de dados específico, traduzindo toda e qualquer peculiaridade para aquele produto obtendo o máximo de performance.

A figura de número 26 nos mostra o *Web Service* em ação, é passando uma *string* contendo uma solicitação SQL, o *Web Service* através dos protocolos padrões fará acesso ao banco de dados e o resultado surgirá em XML. Vale afirmar que apesar de utilizar o *framework* da plataforma .NET para escrever este *Web Service* ilustrativo, os mesmos podem ser desenvolvidos utilizando-se outras plataformas como J2EE.

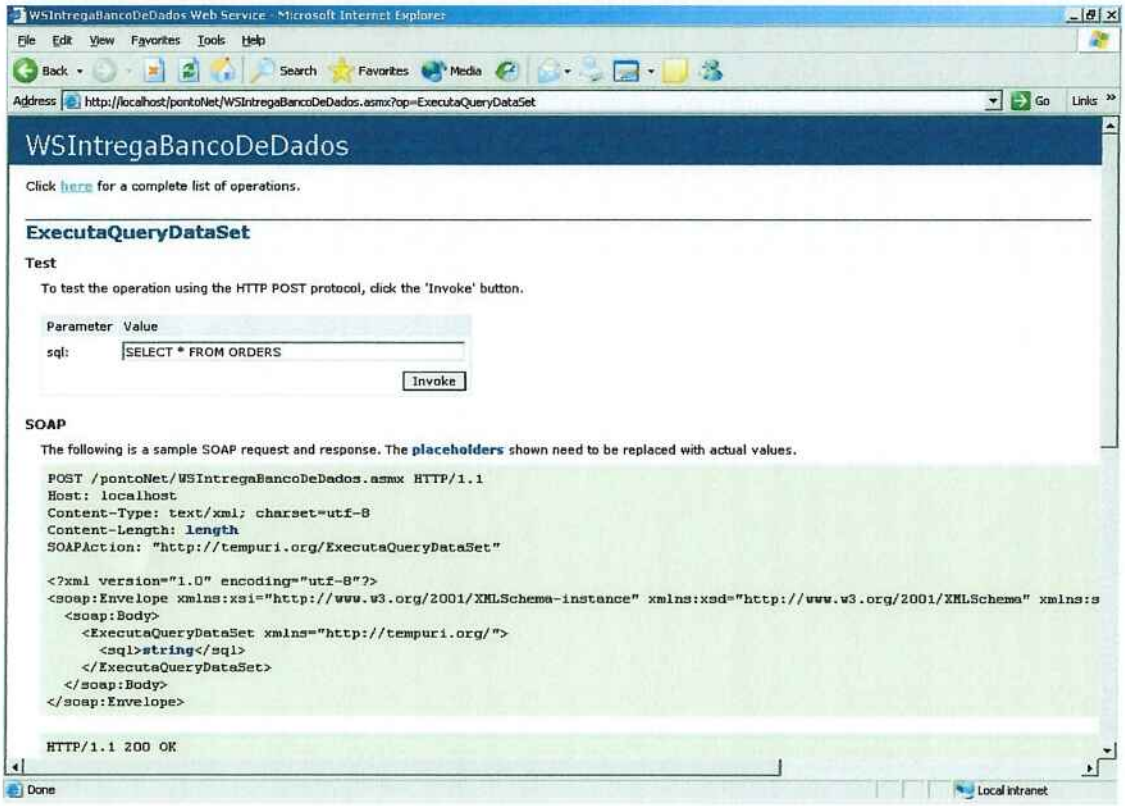

Figura 26 – Solicitação de serviço via *Web Service*.

A aplicação cliente utilizará o serviço disponibilizado pelo *Web Service* seja numa rede interna coorporativa ou externa (*Internet*) como já mencionado, e caso seja a rede mundial, o mesmo deverá ser publicado em um UDDI.

A figura 27 mostra o resultado da solicitação ao banco de dados, o retorno aparece em XML, padrão atualmente aceito em qualquer plataforma de *software* e que pode ser tratado e visualizado em diversos meios. A transformação de

documentos XML para vários tipos de meios de apresentação depende do estabelecimento de regras de formato descritas em documentos denominados *Extensible Style Sheet* (XSL), com a utilização do XSL, pode-se criar documentos de estilo que são processados para a exibição de documentos de acordo com a necessidade.

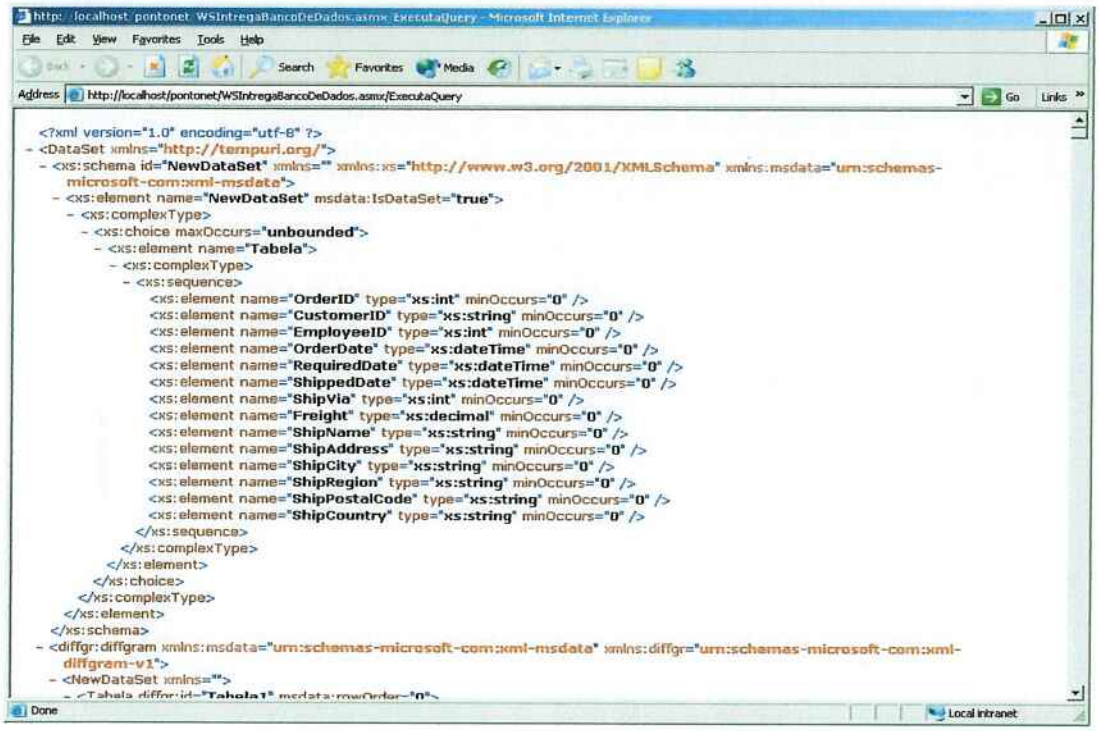

Figura 27 - Resultado da solicitação em XML.

O *Web Service*, enviará as solicitações aos devidos bancos de dados utilizando-se da API de cada produto. As consultas passarão por um decodificador o qual traduzirá determinados comandos específicos de cada linguagem SQL, pois uma consulta totalmente ANSI não resolve muitos dos problemas que temos na prática quando a aplicação faz acesso a vários produtos de bancos de dados de diferentes fabricantes.

Em aplicações de porte, a equipe de desenvolvedores precisa conhecer todas as características de cada banco de dados, uma vez que a solução proposta possa

traduzir as consultas para cada produto em particular estas, ficariam padronizadas e com facilidade de manutenção.

Para atender uma solicitação específica de determinado cliente, muitas vezes o código é duplicado pois uma consulta ou um procedimento armazenado é executado apenas em um produto em particular, tornando a aplicação difícil de se manter. As consultas e procedimentos armazenados devem ser escritas com cautela, tendo em mente principalmente a performance e escalabilidade.

4.5 – Gerenciamento e Monitoração

O tópico seguinte apresenta o módulo de gerenciamento e monitoração do *middleware* proposto.

O gerenciamento e monitoração do *middleware* proposto pode ser feito com o desenvolvimento de uma aplicação sem grandes complexidades. O *software* de monitoração visa identificar tentativas de acessos indevidos, número de seções abertas controle de usuários e transações.

O próprio módulo de gerenciamento é um exemplo de uma aplicação a qual poderá utilizar o serviço proposto. A camada de apresentação, fica como decisão de projeto ou seja pode-se optar pelo estilo janelas ou mesmo o estilo *Web*, o retorno será sempre em XML e este pode ser formatado utilizando-se tecnologias com XSL. O *software* de monitoração utilizará o *Web Service* para fazer a conexão às fontes de dados, e através do modulo de controle interno do *middleware* gerenciar as aplicações que estejam utilizando o serviço em determinado momento.

O módulo de controle interno do serviço, além gerenciar os acessos e solicitações das aplicações cliente, conterá métodos que deverão auxiliar o administrador do *middleware*, um exemplo seria o tratamento de erros quem além de retornar determinados erros que dizem respeito aos acessos clientes, também deverá reportar erros ocorridos entre o serviço em si e a camada de banco de dados, porém apenas para a aplicação de monitoração.

Uma vez que os métodos internos do módulo de controle possam executar tarefas como auditoria, registro de log entre outras, o desenvolvimento de um módulo de gerenciamento passa a ser nada mais do que uma aplicação cliente utilizando-se do serviço proposto.

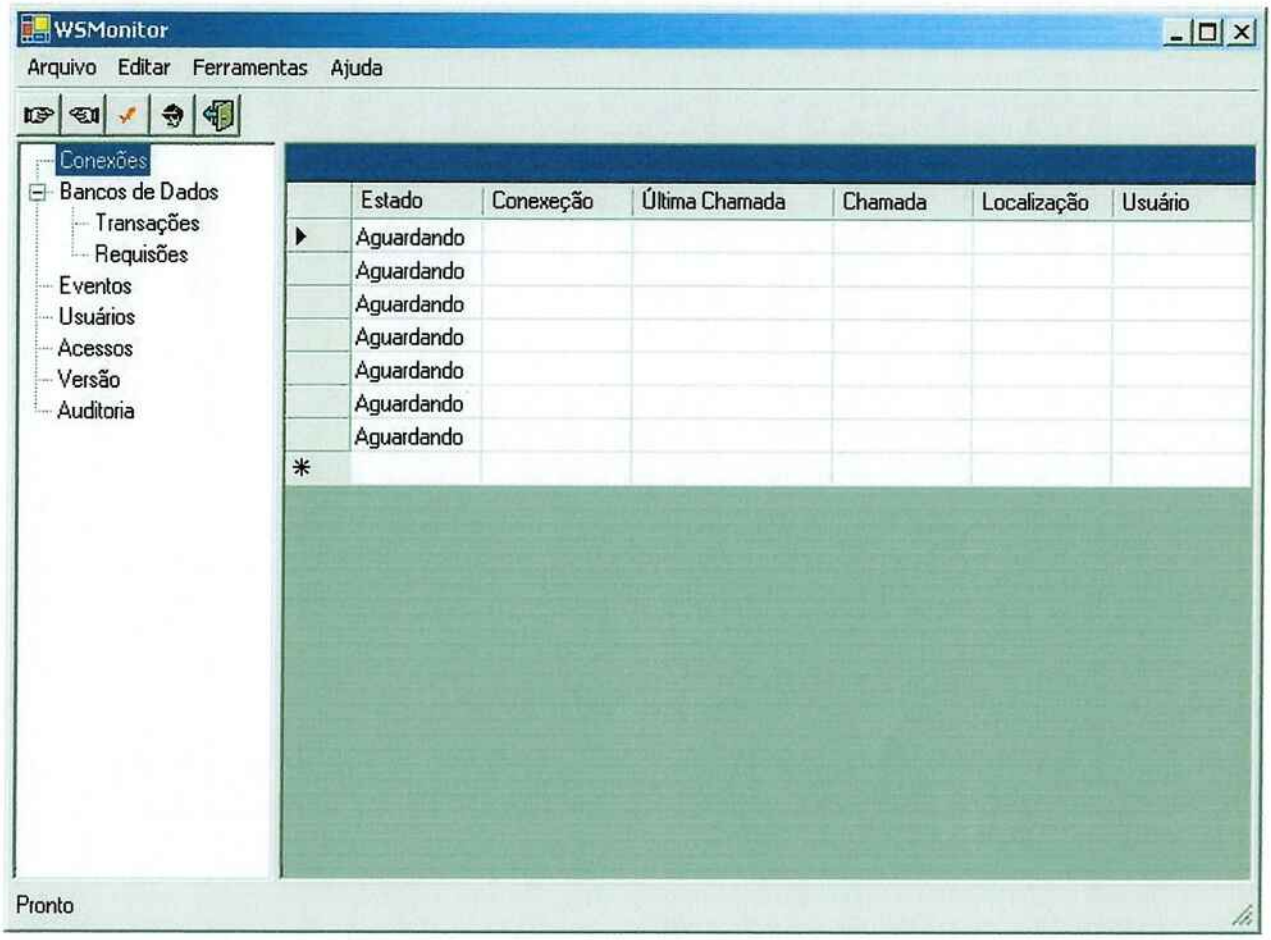

Figura 28 – Módulo de Gerenciamento.

A ferramenta de gerenciamento, é de acesso restrito ao administrador do *middleware*, que a utilizará para identificar possíveis gargalhos de rede, disponibilidade das fontes de dados, o números de requisições e transações executadas. O módulo deverá ainda manter uma trilha de auditoria de todos o eventos ocorridos em determinada seção tais como data e hora de conexão, números de vezes que o serviço foi solicitado e tempo de resposta. A figura do 28 ilustra a proposta de uma ferramenta de gerenciamento do *middleware*, foi utilizado o estilo de janelas como camada de apresentação.

5 – CONSIDERAÇÕES FINAIS

Neste capítulo final apresenta-se as principais conclusões encontradas, alguns comentários gerais, e uma possível contribuição acadêmica, ressaltando que um estudo com mais profundidade seria de grande auxílio para a implementação de um modelo prático e utilizável. A proposta de uma arquitetura baseada em padrões abertos e em tecnologias emergentes de comum acordo entre os grandes desenvolvedores de *software*, podem impulsionar novas pesquisas, e trabalhos mais abrangentes, os quais poder gerar uma contribuição acadêmica com bases ainda mais sólidas.

5.1 – Conclusões

O presente trabalho propôs a utilização de *Web Services* como um *middleware* para a integração de aplicações de *software* e bancos de dados, com o intuito de facilitar tanto o desenvolvimento quanto a manutenção das aplicações. Os *Web Services* e as tecnologias que o acompanham, tornam sua utilização a ferramenta ideal para definir a arquitetura de integração entre bancos de dados de diferentes fabricantes.

A utilização de XML e *Web Services* nos projetos de integração de sistemas da forma como foi apresentado não seria atividade simples e rápida, apesar da infra-estrutura e das ferramentas de desenvolvimento de *software* existentes, é necessário rever a real necessidade e a aplicabilidade da solução.

A troca de informações em XML, facilita a integração por ser uma linguagem amplamente aceita em todas as plataformas e também por ser um padrão o qual os grandes produtores de *software* estão de acordo.

Os *Web Services*, vêm recebendo significativa atenção da industria de desenvolvimento de *software*, é grande a expectativa principalmente em torno das oportunidades oferecidas pelos mesmos. Os grandes fabricantes de SGBD's e ambientes de desenvolvimento já incorporam a tecnologia dos *Web Services* aos seus produtos.

A parceria entre *Web Services* e bancos de dados promete para um futuro próximo uma crescente integração através de ferramentas que suportem tipos de dados mais complexos. Apesar da promessa e do empenho dos grandes produtores de *software* em tornar a integração realidade, existem diversos pontos que ainda precisam ser analisados e estudados com maior profundidade, entre eles questões relativas a desempenho, tolerância a falhas, segurança e transações ACID.

Apesar das considerações e conclusões acima, a arquitetura proposta, dentro do contexto da solução de problemas de integração, precisa definir o tipo de aplicação que deve fazer uso da arquitetura, demandar recursos para desenvolver uma arquitetura como a proposta no presente trabalho para integração de uma *interface* simples entre o *mainframe* e a plataforma baixa não faz sentido.

Uma conclusão importante está na utilização da arquitetura proposta para integrar aplicações de *software* que acessam fontes de dados heterogêneas de porte médio a grande. O tipo de arquitetura proposta engloba muitos aspectos que podem tornar difícil a definição dos requisitos. A utilização de plataformas de *software* como J2EE e .NET tornam o desenvolvimento da arquitetura tarefa mais simples e completa pois suportam as tecnologias necessárias à implementação.

Finalmente, o sucesso da implementação da arquitetura proposta, depende basicamente de uma infra-estrutura que esteja preparada para suportar as tecnologias necessárias tais como XML e *Web Services* esta é uma relação muito

forte, portanto, o projeto do *middleware* de integração deve ser realizado metodologicamente.

5.1 – Comentários Gerais

Um comentário que merece destaque, é a visão do autor como desenvolvedor de *software* para empresas de *asset management* onde o convívio diário com tarefas de manutenção e mesmo desenvolvimento de novas implementações em aplicações de *software* que acessam diversos produtos de banco de dados tem sido tarefa estafante, pois muitas vezes, o código é duplicado para atender uma solução específica de determinado cliente o qual tem um produto de banco de dados ao qual a aplicação não estava preparada para atender. Nesse contexto a aplicação além de manter muitas linhas de código exige altos custos de manutenção e mesmo desenvolvimento.

5.1 – Continuidade da Pesquisa

Este trabalho de monografia inicia mais um impulso na linha de pesquisa de integração de sistemas a fontes de dados heterogêneas. A integração de dados em geral e também a integração de aplicações de *software* diversos, merecem especial atenção devido a larga utilização na comunidade industrial.

Para tanto, dar continuidade a uma pesquisa mais profunda se faz necessário, para isso um estudo acadêmico metódico, em conjunto com pesquisadores em nível de mestrado ou doutorado e o desenvolvimento de um modelo utilizando as plataformas de software mais expressivas da atualidade, a ser aplicado na prática possibilitaria aparar quaisquer arestas que possam prejudicar principalmente a performance e a escalabilidade da arquitetura proposta.

5.1 – Contribuição Acadêmica

A contribuição acadêmica deste trabalho está no incentivo de poder propor uma solução para resolver um problema antigo existente nas corporações e nas aplicações de *software* destinadas a atender mercados variados, ou seja a integração de dados.

O ponto mais importante da contribuição acadêmica, está na utilização de tecnologias como *Web Services* e XML, as quais a comunidade industrial está de comum acordo, e incentivar um estudo mais profundo do assunto que promete ser num futuro muito próximo a solução para a integração tanto de sistemas como de fontes de dados heterogêneas.

Outra contribuição está na geração de outros trabalhos que possam ser utilizados na prática.

REFERÊNCIAS BIBIOGRÁFICAS

[Orfali,1996] Cliente / Servidor Guia de Sobrevivência, IBPI Press, 1996.

[Umar,1997] Object Oriented Client / Server Internet Enviroments, 1997.

[Silberschatz et al, 1999] Silberschatz, Abraham; Korth, F. Henry; Sudarshan,S Sistemas de Banco de Dados, Makron Books, 1999.

[Saens, 2004] Saens Artola, Esmilda; Internet (TCP/IP e UDP. Disponível em http://penta2.ufrgs.br/Esmilda/origem.html Data de acesso:18/10/2004.

[David, 2004] David L. Wasley; Developing Middleware Services for UC. Disponível em http://www.ucop.edu/irc/projects/UC-Middleware/sld002.htm Data de acesso: 18/10/2004.

[Carvalho, 2004] Carvalho, Osvaldo; GT Middleware. Disponível em http://www.rnp.br/pd/gts2004-2005/middleware.html Data de acesso: 18/10/2004.

[Silva, 2004] Silva, Miguel Mira da: Integração de Sistemas da Informação; Disponível em http://www.fca.pt/livros-html/391_5.html#pagonline. Data de acesso: 18/10/2004.

[Holst, 2003] Holst, Sebastian; XML Databases: An Idea Whose Time has Finally Come, Software AG The XML Company, 01/06/2003.

[Bezerra, 2004] Bezerra, Eduardo; XML e sua integração com Banco de Dados; artigo publicado na revista: SQL Magazine, edição 8, página 10.

[Parand, 2002] Parand Tony Darugar; Keeping Web Services Simple; New Architect: Disponível em http://www.newarchitectmag.com/documents/s=7221/na0702j/index.html Data de acesso: 26/09/2004.

[Lombardi, 2002] Lombardi Victor; Designing for Web Services; New Architect:, Disponível em http://www.newarchitectmag.com/documents/s=7053/new1015627350101/index.html Data de acesso: 26/09/2004.

[Sholtz, 2004] Sholtz, Paul; SOAP: Clean and Secure New Architect:, Disponível em http://www.newarchitectmag.com/documents/s=4292/new1013636066/index.html Data de acesso: 26/09/2004.

[Gitman, 2004] Gitman, Mitch; Web Services for Interoperable Management Extend J2EE and .NET enterprise management with Web services. XML and Web Service Magazine, Disponível em http://www.fawcette.com/special/opsmgmt/gitman/ Data de acesso: 26/09/20004.

[Meehan, 2004] Meehan, Michael; Data Integration and Web services; COMPUTERWORLD; Disponível em http://www.computerworld.com/databasetopics/data/story/0,10801,70043,00.html Data de acesso: 26/09/20004.

[Mensah, 2004] Kuassi Mensah; Web Services enable Your Database, Web Service Journal, Disponível em http://www.sys-con.com/webservices/articleprint.cfm?id=515 Data de acesso: 24/10/2004.

[Sant'Anna, 2004] Sant'Anna,Mauro, SOAP e WebServices, Disponível em http://www.linhadecodigo.com.br/artigos.asp?id_ac=38 Data de acesso: 26/10/2004.

[Turban, 2000] Turban, Efrain, Lee,Jae, King David, Chung, Michal H., Eletronic Commerce A Managerial Perpective, Prentice Hall 2000.

[Potts, 2003] Potts, Stephen, Kopack, Mike, Aprenda em 24 horas Web Services, Editora Campos 2003.

[Jorgensen, 2002] Jorgensen, David, Desenvolvendo .NET Web Services com XML, Editora Atlas 2002.

[ISAM, 2004] Infra-Estrutura de suporte às Aplicações Móveis; Disponível em http://www.inf.ufrgs.br/~isam/paginaDefComputacaoMovel.html Data de acesso: 03/11/2004.

[Lee, 2004] Lee, Wei Meng, Web Services: Comunique-se em Diferentes Plataformas, Disponível em http://www.linhadecodigo.com.br/artigos.asp?id_ac=61&pag=1 Data de acesso: 03/11/2004.

[Pressman, 2002] Pressman, Roger S., Engenharia de Software 5º Edição Rio de Janeiro MacGraw-Hill, 2002.

[Dextra, 2003] Web Services na Integração de Sistemas Corporativos, Disponível em http://www.dextra.com.br/empresa/boletim/0302-02/02tecnologia.htm Data de acesso: 24/11/2004.

[Martin, 2004] Martin Gaedke; Hans-W. Gellersen; Albrecht Schmidth; UlfStegemüller, Wolfgang Kurr, Object-oriented Web Engineering for Large-scale Web Service Management, Disponível em http://www.computer.org/proceedings/hicss/0001/00015/00015029.PDF Data de acesso: 26/09/2004.

[Rosa, 2004] Rosa, Nelson Souto, Ambientes de *Middleware.* Universidade Federal de Pernambuco, Disponível em: http://www.cin.ufpe.br/~sd/disciplinas/sd/pos/aulas/Middleware.pdf Data de acesso: 03/01/2005.

[Umeda, 1998] Umeda Nelson Naoki; DCOM x DSOM, Disponível em http://www.pr.gov.br/batebyte/edicoes/1998/bb72/dcom.htm Data de acesso: 05/01/2005.