“Integração de dados corporativos: uma proposta de arquitetura baseada em serviços de dados”

Joyce Otsuka Côrtes Degan

Trabalho Final de Mestrado Profissional em Computação

Instituto de Computação
Universidade Estadual de Campinas

“Integração de dados corporativos: uma proposta de arquitetura baseada em serviços de dados”

Joyce Otsuka Côrtes Degan

19/12/2005

Banca Examinadora:

- **Prof[a]. Dr[a]. Claudia Maria Bauzer Medeiros (orientadora)**
  Instituto de Computação - Unicamp
- **Prof. Dr. Renato Fileto**
  CNPTIA - EMBRAPA
- **Prof[a]. Dr[a]. Ariadne Maria Brito Rizzoni de Carvalho**
  Instituto de Computação - Unicamp
- **Prof. Dr. Célio Cardoso Guimarães (Suplente)**
  Instituto de Computação – Unicamp

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Degan, Joyce Otsuka Côrtes

D363i Integração de dados corporativos: uma proposta de arquitetura baseada em serviços de dados / Joyce Otsuka Côrtes Degan – Campinas, [S.P. :s.n.], 2005.

Orientador : Claudia M. Bauzer Medeiros

Trabalho final (mestrado profissional) – Universidade Estadual de Campinas, Instituto de Computação.

1. Serviços na Web. 2. Banco de dados. 3. Engenharia de software. I. Medeiros, Claudia Maria Bauzer. II. Universidade Estadual de Campinas, Instituto de Computação. III. Título.

“Integração de dados corporativos: uma proposta de arquitetura baseada em serviços de dados”

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigido e defendido por Joyce Otsuka Côrtes Degan e aprovado pela Banca Examinadora.

Campinas, 19 de Dezembro de 2005.

Prof[a]. Dr[a]. Claudia Maria Bauzer Medeiros
(Orientadora)

Trabalho Final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na área de Engenharia de Computação.

A meus filhos,

Danielle e Douglas

Agradecimentos

Ao meu marido, Valdemir, pelo carinho e por ter estado comigo todo o tempo, acreditando no meu sucesso e me fortalecendo com o seu apoio e dedicação.

A minha mãe, Mei, pelo apoio constante e amoroso durante todos os momentos da minha vida e, principalmente, por ter me ensinado a importância dos estudos e da persistência na perseguição de nossos ideais.

A meus filhos, Danielle e Douglas, pelo incentivo, pela compreensão, por serem a base sólida na qual eu me apóio e tiro forças para crescer e melhorar sempre.

A meu irmão Jonathas, pela força carinhosa, não me deixando desanimar nunca.

À minha orientadora, Prof[a]. Dr[a]. Claudia Bauzer Medeiros, pela dedicação, ensinamentos e confiança depositada.

Aos colegas e amigos da Unicamp, em especial Luiz Kacuta, Raquel Maranhão e Mozart Costa, pela solidariedade e suporte, inclusive logísticos, durante as jornadas até Campinas.

Aos meus amigos, por sempre torcerem pelo meu sucesso e compreenderem as minhas ausências para dedicar-me aos estudos.

Á Claudia Regina da Silva da secretaria de extensão do IC pela ajuda e cordialidade no atendimento.

E, finalmente, ao Instituto de Computação da Unicamp pela oportunidade oferecida para a realização de um desejo há muito acalentado.

Resumo

A necessidade da integração de dados em ambientes empresariais, apesar de antiga, ainda é um problema crucial a ser resolvido para a maioria das empresas, permitindo integração entre clientes, parceiros e fornecedores. Fusões e aquisições corporativas conjugadas com sistemas legados resultantes de diferentes implementações, por diferentes fornecedores em diferentes momentos tecnológicos, resultam em uma base distribuída, redundante, heterogênea e de difícil gerenciamento. A integração necessita de mecanismos confiáveis e procedimentos integrados para assegurar a consistência, segurança e controle dos dados corporativos. As soluções para integração de dados disponíveis no mercado ainda têm uma visão fragmentada da integração nesse nível.

Este trabalho analisa problemas encontrados na prática em ambientes empresariais para integração de dados que considera todos esses fatores e propõe uma arquitetura para a solução desses problemas. A arquitetura combina enfoques de bancos de dados, sistemas distribuídos e soluções atuais do ponto de vista de serviços e sistemas Web.

Abstract

The need for data integration in enterprises dates back to several decades. However, it is still a pressing problem for most environments, since it is seen as a means to allow integration among customers, partners and suppliers. Besides needs that arise from fusion of companies, there is always the issue of legacy systems that result from distinct implementations in different technologies. The resulting scenario is a distributed set of files and databases, which are redundant heterogeneous and hard to manage. Data integration requires reliable mechanisms, as well as an integrated set of procedures to ensure consistency, security and control of corporate data. Off the shelf solutions still provide fragmented views of data integration.

This work analyzes problems found in enterprises during data integration processes, taking all previously mentioned factors into consideration. It proposes an architecture to solve these problems. The solution combines research in databases, distributed systems, and Web services and systems.

Índice

Índice de Figuras

Capítulo 1

INTRODUÇÃO

Empresas necessitam integrar seus dados por diversas razões. Uma dessas razões é estender a utilização da tecnologia já existente, ganhando flexibilidade, agilidade e redução de custos na implementação de novos serviços. Outra razão é permitir a integração com clientes, fornecedores e parceiros objetivando ampliar o alcance dos serviços e produtos da empresa inclusive com ampliação para a Web e o mercado eletrônico. Ainda outras razões incluem o entendimento e compartilhamento de informações comuns, distribuídas por entre diferentes bases de dados resultantes de legados, fusões e aquisições.

A quantidade de informação a ser integrada cresce continuamente, assim como a dificuldade de controle sobre o uso, características e distribuição dessas informações. A tarefa de integração é dificultada por diversos motivos. O legado existente na empresa geralmente não é amigável para integração, tendo sido implementado ou adquirido de diversos fornecedores de produtos e serviços em diferentes momentos tecnológicos. Clientes, fornecedores e parceiros muitas vezes utilizam tecnologias díspares que dificultam a integração da cadeia comercial. Empresas que foram fundidas ou adquiridas não raro utilizam diferentes abordagens para implementação de um mesmo conceito. Além da dificuldade imposta por tecnologias ou metodologias, políticas internas sobre a responsabilidade do fornecimento e controle da informação também impõem barreiras à integração.

Os produtos voltados para integração encontrados no mercado geralmente consideram separadamente vários aspectos que deveriam ser vistos de forma integrada. A diversidade de implementações a partir de uma única especificação normalmente não é considerada pelas ferramentas CASE. Além disso, detalhes adicionais sobre o uso dos dados, como fragmentação e redundância, e de comportamento dos dados, como seus aspectos dinâmicos e ativos, não são facilmente visualizados. Tais detalhes ficam encapsulados, entre outros, em bancos de dados, produtos de replicação e aplicativos. As dificuldades relacionadas ao controle sobre o uso dos dados manipulados por diferentes implementações, mesmo utilizando plataformas homogêneas, impedem uma visualização unificada para o gerenciamento adequado desses dados. Aspectos

políticos são ainda diluídos, embutidos em produtos de diversas finalidades, como aqueles voltados para sistemas operacionais, rede, aplicações e bancos de dados.

O objetivo desta dissertação é estudar os problemas mais relevantes para integração de dados em um ambiente empresarial e especificar uma arquitetura considerando tais problemas. A idéia principal é que a solução de integração deve ser unificada e abranger vários aspectos sobre os dados, como especificação, implementação, gerenciamento de metadados, comportamento, qualidade e controle sobre uso e distribuição. Esta dissertação fornece as seguintes contribuições:

- Análise das questões relacionadas à integração de dados em ambientes empresariais.
- Discussão sobre arquitetura orientada a serviços (Service Oriented Architecture - SOA) e serviços Web, sob uma visão prática, abordando questões como segurança, integridade, gerenciamento e outros quesitos de qualidade de serviço.
- Especificação de uma arquitetura para permitir a integração de dados em um ambiente empresarial. Esta arquitetura está centrada em três conceitos básicos: a) considerar o ambiente como uma federação de grupos organizacionais responsáveis pela informação; b) tratar dos problemas de integração em diferentes camadas com funções claramente definidas; c) utilizar a noção de arquiteturas orientadas a serviços, facilitando assim expansão e interoperabilidade.

Os principais conceitos associados são a noção de domínio de dados, o uso de serviços de dados e a proposta da arquitetura propriamente dita. O âmago da contribuição é a arquitetura descrita no capítulo 4. Ela mostra claramente como permitir a integração de domínios separados em uma empresa através de serviços de dados e sua comunicação. Esta comunicação é estabelecida por uma camada denominada CMID (Camada Mediadora de Integração de Dados) que encapsula diferentes funcionalidades garantindo independência de dados. Esta arquitetura nasceu de uma necessidade prática constatada em diferentes empresas.

O texto está organizado da seguinte forma. O capítulo 2 aborda os conceitos básicos necessários para a compreensão do texto, discutindo distribuição e integração de dados. O capítulo 3 descreve arquiteturas orientadas a serviços e serviços Web. Em seguida, o capítulo 4 descreve a solução proposta, a qual abrange uma federação de domínios, a inclusão de uma camada intermediária de integração de dados e a arquitetura dessa camada. Finalmente, o capítulo 5 apresenta as conclusões deste trabalho.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

Este capítulo tem a finalidade de fazer uma revisão dos conceitos que serviram de base para este trabalho. A seção 2.1 introduz os tópicos a serem detalhados nas próximas seções, as quais relacionam distribuição (seção 2.2) e integração de dados (seção 2.3) dentro do contexto de uma empresa. Finalmente a seção 2.4 conclui o capítulo.

2.1 Introdução

A quantidade de informação disponível em uma empresa cresce continuamente; por isso, é importante ter ferramentas adequadas para simplificar a manipulação e o gerenciamento dessas informações. Um dos problemas atuais relacionados aos dados é a dificuldade que a organização tem de enxergá-los corporativamente. Um outro aspecto preocupante é que muito do código para operar os dados, que antes se encontrava somente em aplicativos externos, passou a ser armazenado nos SGBDs (Sistema Gerenciador de Banco de Dados), e a ficar junto com os dados, na forma de *stored procedures*, *triggers* e *rules*. Além disso, a linguagem de implementação desses códigos geralmente é proprietária do fabricante do SGBD, o que fornece um fator a mais de complexidade se os dados estiverem distribuídos por SGBDs heterogêneos.

Os dados de uma empresa são geralmente armazenados em bases de dados heterogêneas, seja utilizando tecnologias modernas ou mais antigas – estas, muitas vezes resquícios de um legado ainda ativo e importante para o funcionamento do negócio. Estas bases de dados podem ser de vários tipos: estruturadas, como bancos de dados relacionais; semi-estruturadas, como arquivos XML (eXtensible Markup Language) e páginas HTML (Hypertext Markup Language); ou não-estruturadas como sons, imagens e fotos.

Uma base de dados centralizada, compartilhada por todas as aplicações corporativas, geralmente distribuídas por uma rede, acaba por criar problemas de otimização de acesso à base de dados, principalmente devido ao desempenho da rede. Um outro problema que geralmente surge é o esgotamento da capacidade física do servidor de banco de dados de atender a tantas solicitações e de armazenar eficientemente todos os dados corporativos. Esses problemas

geralmente levam a soluções que visam à distribuição física das bases corporativas por vários servidores.

Com todos esses cenários, administrar e integrar os dados de uma organização, que já não era simples nos tempos do processamento centralizado, tornou-se uma tarefa ainda mais complexa devido à distribuição de dados, a inclusão de tantos ambientes heterogêneos e tanta diversidade de implementações.

Um aspecto a ser considerado é a independência de dados, em que mudanças de implementação não afetam a visão lógica ou conceitual de aplicações. As noções de esquemas e modelo são importantes em tal contexto.

Um esquema é uma descrição, baseada em um certo modelo de dados, de uma determinada base de dados. Um modelo de dados é um conjunto de conceitos usados para descrever aspectos estáticos dos dados em uma base, como sua estrutura, operações, semântica de dados e regras de consistência. Um modelo de dados pode também descrever o comportamento dos dados, e portanto, seus aspectos dinâmicos e ativos. Um modelo de dados pode ainda ser usado para especificar alguns detalhes adicionais relevantes ao uso dos dados, tais como distribuição e redundância. Os parâmetros de distribuição relacionam-se à fragmentação dos dados, e os de redundância, às especificações de cópias explícitas ou sobrepostas de dados [58]. A figura 2.2, adaptada de [58], esquematiza o processo de projetos de esquemas de bases de dados.

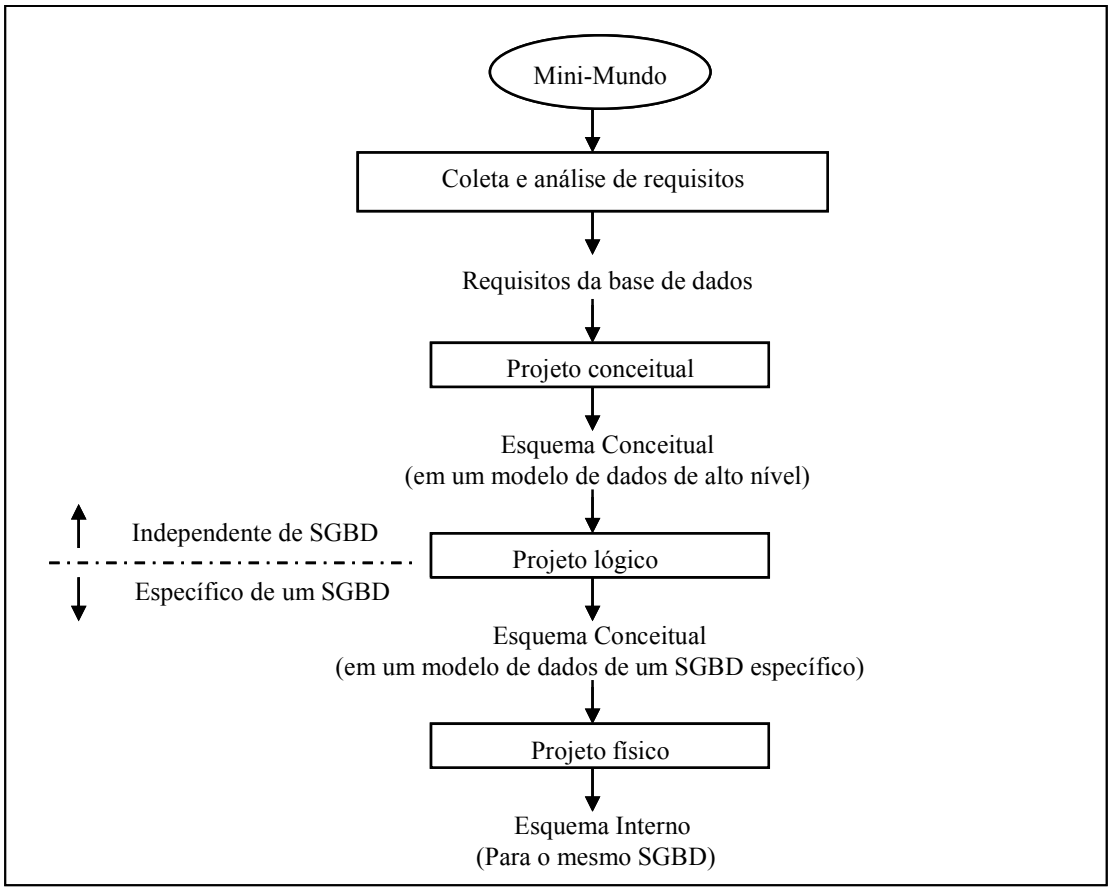

Figura 2. 1 – Processo de projeto de esquemas de bases de dados

Um projeto de base de dados utiliza modelos de dados para mapear os requisitos dos dados e dos aplicativos em esquemas ao longo das etapas de projeto conceitual, lógico e físico. Muitas vezes, durante o projeto lógico, uma base de dados é estruturada usando um determinado modelo, visando uma melhor eficiência no acesso aos dados e uma maior independência dos programas em relação à implementação física.

Durante os últimos anos surgiram diversos modelos de dados, e os SGBDs também evoluíram para se adequarem aos novos paradigmas. Para maiores detalhes sobre esse assunto, uma boa fonte é Navathe [58] que discute a evolução dos modelos de dados para bancos de dados abrangendo desde os modelos hierárquicos e orientados a rede até o modelo para bancos de dados ativos e dinâmicos. Atualmente, os modelos mais usados são o relacional e o orientado a objetos. No modelo orientado a objetos, em vez de enxergar o banco de dados como um conjunto de arquivos, os usuários enxergam o banco de dados como um conjunto de objetos que podem exigir vários arquivos para sua representação[13]. Esse tipo de abordagem introduz uma maior

independência lógica de dados, pois o tratamento da estrutura lógica da base é armazenado junto com os dados.

2.2 Distribuição de dados

Ozsu e Valduriez, em [09], afirmam que, em projetos de distribuição de dados, as duas questões fundamentais são: a separação do banco de dados em fragmentos (fragmentação), e a distribuição ótima desses fragmentos. Dado um esquema relacional, a fragmentação subdivide cada relação em fragmentos (ou partições) que podem ser horizontais (através de uma operação de seleção), verticais (através de uma operação de projeção) [62] ou mista. Há duas alternativas básicas para posicionar os dados: particionados (ou não-replicados) e replicados. No esquema particionado, o banco de dados é dividido em uma série de partições disjuntas, cada uma das quais colocada em um site diferente. No projeto replicado, há a opção de replicação total ou parcial. Na replicação total, o banco de dados é totalmente replicado e armazenado em cada site. Na replicação parcial, cada partição do banco de dados é armazenada em mais de um site, mas não em todos eles [09].

A replicação de dados visando acesso local fornece um melhor desempenho, principalmente para os dados que são acessados freqüentemente; porém, há o perigo da inconsistência de dados. Por outro lado, o acesso direto às bases remotas, que diminui o perigo da inconsistência dos dados, requer uma arquitetura mais forte em relação à tolerância a falhas e desempenho da rede. A seção 2.3.3 abordará com mais detalhes a replicação de dados.

Quando as várias bases de dados que estão distribuídas por uma rede de computadores são logicamente inter-relacionadas, elas formam um sistema de banco de dados distribuídos (SBDD). É importante observar que a distribuição física implica que a comunicação entre as diversas bases é feita através da rede, ou seja, esses SGBDs não precisam estar necessariamente distantes geograficamente [09]. Quando os dados são distribuídos, mas todos os servidores têm o mesmo SGBD, temos um SBDD homogêneo. Se as diferentes bases de dados são controladas por diferentes SGBDs, essencialmente autônomos, porém conectados de alguma forma que permita acesso aos dados a partir dos diferentes locais, temos os SBDDs heterogêneos (SBDH), também conhecidos como sistema de multibancos.

Os SGBDDs tornam a distribuição, a fragmentação e a replicação transparentes para os usuários. Esse tipo de facilidade traz certa complexidade de gerenciamento para algumas funcionalidades do sistema, entre outras, processamento e otimização de consultas, controle de

concorrência, controle de transações e controle de replicação. Os programas não precisam se preocupar com detalhes sobre conectividade e interoperabilidade. Portanto, esses SGBDs fornecem independência também em relação às interfaces de comunicação e operação entre sistemas distribuídos e heterogêneos.

Para maiores detalhes sobre SGBDDs, duas boas fontes são Date [04] e Ozsu e Valduriez [09] que discutem amplamente sistemas de bancos de dados distribuídos abordando várias questões, entre elas, as implicações na arquitetura e no projeto de tais sistemas.

2.3 Integração de Dados

Hoje em dia, as empresas têm necessidade de integrar seus dados e aplicações por diversos motivos, entre outros: expor informações de seus sistemas na *Web*, participar no mercado eletrônico, integrar sua cadeia de fornecedores, e compartilhar informações comuns distribuídas entre suas bases de dados e sistemas corporativos[19].

A tarefa de integrar os dados não é somente uma questão tecnológica, abrangendo também o conhecimento dos requisitos atuais e futuros do domínio do problema. Esse conhecimento engloba, entre outros, metadados, lógica de negócios, interfaces, gerenciamento de desempenho, *workflow*, processamento da informação, bases de dados, desenho da aplicação e tecnologia de *middleware*[19]. Mesmo que o projeto de integração envolva integração em outros níveis, como serviços ou processos, ainda se faz necessário o entendimento do significado dos dados para a empresa e alguma forma de integração no nível de dados. Portanto, esta seção abordará a integração de dados como um problema dentro do domínio da integração de aplicações. Linthicum[19] discute detalhadamente vários dos aspectos relacionados à integração de aplicações tratados nesta seção. Outras fontes sobre esse assunto são Cummins[03] com um enfoque maior em arquiteturas, Krafzig et al [18] e Erl [20] que focam a integração através da arquitetura orientada a serviços e Baragoin et al [63] que discutem integração da informação dentro do contexto de produtos e soluções IBM.

A integração de aplicações pode tomar muita formas, incluindo a integração de aplicações internas – Enterprise Application Integration (EAI) – e a integração de aplicações externas – Business-to-Business Integration (B2B). Seja qual for o tipo de integração, as soluções tecnológicas geralmente envolvem, entre outros, alguma forma de transformação para resolução de diferenças diversas, roteamento para assegurar que a informação chegue ao seu destino correto e regras de processamento que definem o comportamento na integração.

Embora abordagens para integração de aplicações variem, é possível criar algumas categorias. Na abordagem utilizada em [19], a integração incorpora um ou mais dos seguintes tipos:

Orientada a informação (ou a dados) – o objetivo é promover uma integração na camada de dados a ser usada em conjunto com as camadas de aplicação e processos, evitando as complexidades associadas com o conhecimento do negócio e implementação das aplicações.

Orientada a processos de negócios – permite a colocação de um processo de negócio como uma entidade de controle, capaz de acessar tanto as informações como os processos encapsulados em vários sistemas, na ordem correta, fornecendo o gerenciamento e execução de processos comuns que existem - nas e entre as - aplicações.

Orientada a serviços – permite que aplicações compartilhem métodos e lógica de negócios. Isto é alcançado ou pela definição de métodos, que podem ser compartilhados e integrados, ou pelo fornecimento da infra-estrutura para tal compartilhamento e integração. Métodos podem ser compartilhados através de: a) armazenamento em um servidor central, b) acessos interaplicação ou c) mecanismos padronizados para serviços Web.

Orientada a Portal – permite uma visão de vários sistemas – tanto os internos à empresa, como os externos – através de uma única interface de usuário ou aplicação. O benefício trazido por este tipo de integração evita os problemas de integração do *backend*; ele adapta a interface do usuário de cada sistema para uma interface comum – geralmente um *browser Web*. Como resultado, há a integração de todos os sistemas participantes através dessa interface, embora as aplicações não estejam diretamente integradas.

2.3.1 A integração da informação e a integração de aplicações

Historicamente, as aplicações eram desenhadas com o esquema de dados fortemente integrado com o código da aplicação, misturando as fronteiras entre dados e aplicações. Abordagens mais recentes misturam ambos os tipos, integrando no nível de dados em alguns casos, e no de aplicações, em outros. A integração (ou conectividade) entre aplicações geralmente é usada quando o problema é a comunicação de eventos simples ou de negócios. A integração da informação geralmente é usada quando o problema é o acesso do estado do negócio, na forma como ele está representado nas bases de dados [63]. A integração no nível da informação, permite, dentre outros: acesso aos dados como se eles estivessem em uma única base, explorar

características disponíveis dentro do servidor de dados ou melhorar o desempenho durante uma operação de recuperação de dados e redução da complexidade de desenvolvimento de programas.

2.3.2 Produtores e consumidores de dados

No mundo da integração de dados os sistemas de origem e destino são sempre entidades que produzem e consomem informação, podendo ser, entre outros [19]:

Sistemas Gerenciadores de Banco de dados – Atualmente são os produtores e consumidores de dados mais utilizados. Os SGBDs foram desenhados para exercerem essas funções fornecendo uma excelente interface para troca de informações. A interação com os SGBDs é feita através de linguagens nativas como o Structured Query Language (SQL), por meio de interfaces como Call-Level Interface (CLI), por exemplo, JDBC (Java Database Connectivity) e ODBC (Open Database Connectivity). Os SGBDs no mercado podem diferir entre outros, quanto a: linguagens de interação, linguagem de programação procedural, formatos produzidos, e modelos (relacional, objeto, objeto-relacional, XML, orientado a rede, e hierárquico).

Interfaces de aplicativos – interfaces de aplicativos são mais complexas que os bancos de dados devido a diferenças na abordagem de como os dados são consumidos e produzidos. Tais interfaces geralmente não compartilham padrões comuns. Cada pacote de aplicação precisa ser endereçado individualmente, através de programação ou através de adaptadores fornecidos pelos fabricantes de produtos para integração. As interfaces de aplicação fornecem acesso tanto à informação encapsulada, quanto aos serviços encapsulados na aplicação. Também permitem acesso tanto aos processos de negócios, como acesso direto aos dados.

Interfaces do usuário – A utilização de interfaces do usuário como um ponto de integração da informação geralmente é feita através de um processo chamado *screen scraping*. Esse processo consiste em acessar informações contidas na interface do usuário, como um terminal 3270, através de mecanismos de programação. Esse tipo de mecanismo geralmente é usado para acessar dados de sistemas legados ou proprietários, onde é difícil, às vezes impossível, obter essas informações de outra maneira, seja por falta de entendimento da base de dados, ou por indisponibilidade dos códigos dos aplicativos. Outra forma de integração via interface é o uso de semântica no *design* e opções de mapeamento entre conceitos.

Dispositivos embutidos (*Embedded device*) – A informação também pode vir de dispositivos, tais como, sensores de temperatura ou dispositivos sem fio. Lidar com dispositivos é similar a lidar com aplicações. As interfaces são tipicamente API (Application Programming

Interface) proprietárias e muitas vezes a informação obtida é pontual, fluindo sem que o dispositivo armazene os dados.

2.3.3 Soluções para Integração de dados

Há várias funções que precisam ser fornecidas por uma plataforma de integração de informação [63]:

- Acesso de leitura/gravação a todas as formas de informação, tais como: estruturadas, semi-estruturadas ou não-estruturadas e heterogêneas, vindas de sistemas legados, pacotes e Web;
- Acesso unificado aos dados locais e remotos, em tempo real ou não.
- Gerenciamento de metadados através de uma ferramenta comum para acesso e gerenciamento dos dados e suas descrições.
- Escolha de modo de armazenamento
- Escolha de entrega de dados: interface de aplicação tais como CLI (Call Level Interface), ODBC, JDBC e Web Services.
- Gerenciamento da colocação dos dados: replicação, ETL (Extract, Transform, Load) e *caching* de dados heterogêneos.

As soluções para integração orientada à informação podem ser agrupadas nas categorias federação, replicação e processamento de interface[19].

Federação é o conceito que envolve uma coleção de fontes de dados que podem ser visualizadas e manipuladas como se elas fossem uma única fonte, enquanto retêm sua autonomia e integridade. Os recursos podem ser homogêneos ou heterogêneos, locais ou distribuídos, dependendo da implementação. Possibilita a integração de diversas informações de negócios tanto da empresa como de seus parceiros.

A ***replicação*** foca na captura de alterações nas bases de dados de origem, aplicando as alterações em uma ou mais bases de dados de destino. Essas bases podem ser de diferentes fabricantes e/ou de diferentes modelos. O fundamental é que a replicação possibilite as transformações entre modelos de dados e esquemas de bancos de dados e que forneça a infra-estrutura para troca de dados. A replicação possibilita a centralização dos dados (por exemplo, em *data warehouses*), e fornece a infra-estrutura necessária para o gerenciamento eficiente dos caches de dados. Os caches de dados (tabelas de consultas materializadas) suportam a alocação e

gerenciamento de dados em múltiplos pontos na hierarquia dos dados visando desempenho, podendo ser definidas sobre as fontes de dados federadas.

A replicação pode ser feita de várias maneiras, desde uma aplicação customizada que leia os dados de uma base e atualize os dados em outra até produtos sofisticados específicos para essa funcionalidade. Os produtos que implementam a replicação geralmente fornecem a capacidade de gerenciamento fim-a-fim da transferência. Esta capacidade permite a realocação incremental ou total em uma variedade de combinações, além de permitir várias opções de políticas de atualizações, por exemplo, atualização contínua ou a cada hora.

A replicação pode ser síncrona ou assíncrona. No caso síncrono, as atualizações são propagadas dentro da mesma transação. A vantagem, nesse caso, é a diminuição da possibilidade de inconsistências nos dados; por outro lado, cada atualização nos dados primários pode acarretar problemas de desempenho e indisponibilidade das cópias em todos os sites (origem e destinos). Esse tipo de replicação geralmente é implementado através de aplicações customizadas ou de *triggers*. No caso assíncrono, as atualizações se propagam em um momento posterior, fora da transação que efetuou a atualização. Nesse caso, a atualização dos dados é mais rápida e não indisponibiliza o acesso às cópias, porém cria-se um período de latência em que os dados não estão idênticos. Esse tipo de replicação geralmente é implementada através da leitura de registros de *logs* de transações ou de filas de atualizações a serem propagadas.

As maiores vantagens da replicação de dados são: simplicidade e baixo custo[19]. No entanto, se houver necessidade de que métodos sejam acoplados ou compartilhados junto com os dados, devem ser usadas soluções como replicação de *stored procedures*, que muitos produtos de replicação fornecem, ou utilizar uma solução orientada a serviço. Uma dificuldade no gerenciamento de replicação de dados é garantir a integridade dos dados. Os dados devem ser protegidos contra alterações quando eles forem simples cópias. No caso da replicação ser bi-direcional, é preciso que haja um bom gerenciamento na resolução de eventuais conflitos.

O ***processamento de interface*** foca na integração entre pacotes (como SAP) e as aplicações customizadas. Agentes (*brokers*) de integração suportam o processamento de interface através de adaptadores para conexão com as aplicações, externalizando a informação dessas aplicações através de suas interfaces proprietárias. Eles também se conectam a soluções tecnológicas que incluem *middleware* e *screen scrapers* como pontos de integração. O processamento de interface

da aplicação resolve diferenças entre esquemas, conteúdo, e semânticas da aplicação através da tradução *on the fly* da informação que flui entre os sistemas.

A desvantagem do uso deste tipo de solução é que ela geralmente não considera a lógica e os métodos dos sistemas de origem e destino, que podem ser relevantes dentro da abordagem da integração que estiver sendo usada. Nesses casos, soluções orientadas a serviço podem ser uma opção mais adequada[19].

Diante das soluções descritas acima, destacam-se as abordagens mais comuns para integração de dados em termos de arquitetura – bancos de dados federados, *data warehouse* e mediadores.

Bancos de dados federados permitem acesso a múltiplos bancos de dados conectados na empresa através de uma interface única e bem definida. Podem apresentar diferentes graus de autonomia, heterogeneidade e distribuição de dados. Cada banco conhece cada um dos outros participantes, e pode solicitar que eles lhe forneçam informações. Diferente da replicação de dados, essa solução não requer alterações nas bases de dados, porém requer alterações nas aplicações para que elas suportem o *software* da base de dados federada.

O *software* que implementa a federação de bases de dados coloca uma camada entre as bases de dados físicas distribuídas e as aplicações que vêem os dados. Essa camada conecta os bancos de dados usando vários tipos de interfaces, mapeando as bases de dados físicas ao modelo virtual que existe somente no software. A aplicação utiliza a base virtual para acessar a informação necessária. Larson *et al*, em [06], discutem e comparam diversas arquiteturas para federação de bancos de dados.

Armazéns de dados (Data Warehouses) contêm cópias de dados de várias origens armazenadas (materializadas) num banco de dados único, após sofrerem processamento ou não. O armazém de dados é atualizado periodicamente, e mantém histórico dos dados para análise dos mesmos.

Mediadores (Middleware) são componentes de software que contêm os mecanismos e estruturas necessários para integração dos dados oriundos de fontes diferentes. A integração é virtual. Os dados são acessados de forma integrada mas permanecem no seu estado original nas fontes ou bases de dados de origem. Os adaptadores complementam os mediadores, transformando os dados quando eles são extraídos em sua origem.

2.3.4 Tecnologias para Integração

Uma integração pode ser categorizada por produtos de implementação de padrões e tecnologias middleware que ajudam no processo de compartilhamento de dados, serviços e processos. Um *middleware* pode ser classificado usando os seguintes modelos:

Ponto-a-ponto – permite uma ligação direta entre duas aplicações. O envolvimento de mais de duas aplicações na integração requer ligações ponto-a-ponto entre todas as aplicações envolvidas, resultando em uma configuração muito complexa e inadequada[19]. Este tipo de configuração não permite processamento na camada intermediária.

Muitos-para-muitos – muitas aplicações são ligadas a várias outras aplicações, constituindo uma melhor opção para EAI (Enterprise Application Integration) [19].

Um middleware tipicamente emprega um ou mais dos seguintes estilos de comunicação:

Síncrono ou Assíncrono – No primeiro, o software de *middleware* é fortemente acoplado às aplicações. A aplicação é dependente do *middleware* para processar uma ou mais funções de chamada em uma aplicação remota. Como resultado, a aplicação que inicia a chamada precisa interromper o processamento para esperar que a aplicação remota responda. Problemas com a rede ou com o servidor remoto podem interromper o processamento da aplicação. No modo assíncrono, o software de middleware consegue se desacoplar da aplicação fonte ou destino. O término de uma aplicação não depende do estado das outras aplicações.

Orientado a conexão ou sem conexão – O primeiro possibilita a conexão, troca de mensagens e desconexão entre duas aplicações. Geralmente é um processo síncrono, embora possa ser também assíncrono. No sem conexão, o programa que executou a chamada não faz conexão com o processo destino. A aplicação que recebe a chamada simplesmente atua na requisição, respondendo quando requisitada.

Comunicação direta ou através de filas (queued communications) – Na comunicação direta, a camada *middleware* aceita uma mensagem de um programa que executou uma chamada e a passa diretamente para o programa remoto. A comunicação direta, usualmente síncrona, é geralmente usada pelos *middlewares* que habilitam RPC (Remote Procedure Call). A comunicação através de filas geralmente requer um gerenciador para colocar uma mensagem na fila. Uma aplicação remota então recupera a mensagem, sem levar em consideração o tempo. Se a aplicação que iniciou a chamada requer uma resposta, a informação retorna pelo mecanismo de fila. A comunicação através de filas não requer nem que o programa remoto esteja ativo para que

o programa chamador envie a mensagem, e nem que os programas envolvidos interrompam o seu processamento. Esse tipo de comunicação é geralmente usado pelos produtos baseados em MOM (Message Oriented Middleware).

Publica/subscreve (Publish/subscribe) – a aplicação não precisa entender algo sobre a aplicação destino. A aplicação envia (publica) a informação para um *broker* de publicação/subscrição que a redistribui para as aplicações interessadas (subscritoras).

Request/response(reply), fire and forget e modo conversacional – No primeiro caso, uma solicitação é feita para uma aplicação usando middleware de request/response, que responde à solicitação. Em *fire and forget*, o usuário do middleware envia uma mensagem e esquece, sem se preocupar quem recebe, ou mesmo se a mensagem foi recebida por alguém. Esta é uma abordagem assíncrona que permite que uma aplicação fonte ou destino anuncie tipos específicos de mensagens para múltiplos receptores. O modo conversacional suporta troca bi-direcional de informação através de uma ou mais aplicações lembrando uma conversa.

Os *middlewares* podem ser de um ou mais dos seguintes tipos:

Baseados em RPC (Remote Procedure Call) – utilizam o modelo síncrono e fornecem a capacidade para uma aplicação invocar um procedimento em uma máquina remota através de uma função de chamada. O procedimento é executado na máquina remota de forma transparente para o usuário, escondendo as implicações de rede e sistema operacional, dando a aparência de processamento local. O processamento no cliente que invoca a função remota precisa ser suspenso até que o procedimento na máquina remota seja completado. O RPC pode limitar o desempenho e usar maiores recursos de largura de banda da rede, mas provê maior controle de integridade de dados. Padrões como CORBA (Common Object Request Broker Architecture) e COM (Component Object Model) usam RPC para comunicação entre os objetos distribuídos.

Orientado a mensagem (Message Oriented Middleware - MOM) – usa mecanismo de mensagens para mover informações de um ponto a outro. Seguem o paradigma de comunicação assíncrona e suportam os modelos ponto-a-ponto e fila de mensagens. Garante a entrega da informação mesmo quando o destino não está disponível no momento do envio. O produto MQSeries, da IBM, é um exemplo de MOM.

Objetos distribuídos – São pequenos programas que usam interfaces e protocolos padrões para comunicação entre si. Estão classificados como *middleware* pois facilitam a comunicação entre aplicações. Todavia, eles também são mecanismos para desenvolvimento de aplicações,

fornecendo uma infra-estrutura para compartilhamento de métodos usados por vários serviços. Geralmente são utilizados em cenários de integração em um modelo de computação distribuída e onde um grande número de métodos comuns precisam ser compartilhados. Um exemplo são programas de aplicação em JAVA usando protocolo RMI (Remote Method Invocation) ou CORBA IIOP (Internet Inter-Orb Protocol) para comunicação.

Orientado a banco de dados – *middlewares* que facilitam a comunicação com bases de dados, não importando o modelo empregado, ou plataforma utilizada. Esse tipo de middleware é utilizado para extrair informações em bases de dados locais ou remotas. Também são utilizados para fornecer uma camada para alocação de dados como uma base virtual. Por exemplo, permitem visualizar os dados em uma base relacional como objetos, ou misturar e combinar diferentes modelos de dados. Podem ser: a) nativos, através de APIs fornecidas pelo fabricante para acesso a uma base de dados específica; b) CLI (Call Level Interface), tais como ODBC e JDBC, que fornecem uma única interface para acesso a diferentes bases de dados, utilizando *drivers* de diferentes fornecedores, traduzindo conjuntos de respostas para uma representação comum.; c) *gateways* para banco de dados, com APIs que usam uma interface única para acesso a bases que residem em diferentes tipos de plataformas, geralmente difíceis de serem acessadas, como mainframes.

Orientado a transação – Coordenam a movimentação de informações e o compartilhamento de métodos entre diferentes recursos. São baseados no conceito de transação, controlando a integridade transacional envolvendo recursos locais ou remotos. Funcionam como uma API em uma aplicação, sendo flexíveis para misturar e combinar uma variedade de camadas middleware e servidores de recursos. Nesta categoria, estão incluídos os servidores de aplicação e monitores de transações.

Os monitores de transações fornecem um lugar para a lógica da aplicação e um mecanismo para facilitar a comunicação entre uma ou mais aplicações. O produto CICS (Customer Information Control System) da IBM é um exemplo de monitor de transações. Servidores de aplicação possibilitam o compartilhamento e processamento da lógica e interface da aplicação facilitando a conexão com os recursos de segundo plano (backend), tais como, bases de dados, aplicações ERP e aplicações no mainframe. Os servidores de aplicação permitem um ambiente integrado de desenvolvimento. Através da colocação da maior parte da lógica da aplicação em uma camada intermediária, pode-se aumentar o controle dessa lógica através da centralização. O

modelo tradicional de arquitetura de aplicação em três camadas prevê a utilização de servidores de aplicação e caracteriza-se pelas seguintes camadas básicas: a) apresentação, camada cliente responsável por apresentar a informação ao cliente; b) aplicação, camada middleware com a lógica da aplicação e c) dados, camada onde residem os dados.

Message Broker – alguns autores, como Lithicum, em [19], denominam esse tipo de middleware de servidores de integração. Facilitam o movimento de informações (por exemplo, gerenciando mensagens) entre dois ou mais recursos (aplicações fonte e destino). Além disso, os *messages brokers* podem integrar muitas aplicações usando regras comuns e mecanismos de roteamento. *Message brokers* também fazem o mapeamento/transformação das mensagens que trafegam entre as aplicações para resolução de diferenças diversas. Um exemplo de *message broker* é o WebSphere MQ Integrator da IBM. *Message Brokers* geralmente incluem adaptadores que atuam como um *gateway* entre o message broker e o ambiente nativo da aplicação. Através de um conjunto de bibliotecas que mapeiam as diferenças entre duas interfaces distintas, os adaptadores são implementados como uma camada entre a interface do message broker e a aplicação, escondendo as complexidades da interface. Há várias topologias possíveis, tais como:

- **Hub-and-spoke** – O *message broker* fica entre as aplicações fonte e destino que são integradas em uma configuração que lembra uma estrela.
- **Bus** – O *message broker* fica no bus da rede e fornece seus serviços para outros sistemas nesse bus.
- **Multihub** – vários *message brokers* são ligados ou federados, de forma que as aplicações fonte e destino ficam ligadas a quaisquer *brokers* (remotos ou locais) disponíveis na configuração. A topologia multihub é escalável, tornando possível a integração de um número virtualmente ilimitado de aplicações.

Serviços Web – suportam o envio de mensagens XML, encapsuladas em envelopes SOAP (Simple Object Access Protocol), para serem processadas em um servidor. Dependendo do tipo de transporte, por exemplo, HTTP (Hypertext Transfer Protocol), MOM, SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) os serviços Web podem ser implementados usando diferentes tecnologias *middleware* e estilos de comunicação. Os serviços Web implicam em um overhead adicional por causa da inclusão de uma camada de abstração acima da abordagem tradicional de objeto/método. O benefício dessa camada de abstração é que o código lógico não é limitado a uma implementação específica (por exemplo, COBOL - COmmon

Business Oriented Language - ou JAVA) [63]. Serviços Web serão abordados com mais detalhes no capítulo 3.

2.3.5 Problemas para a Integração de dados

Nesta seção abordaremos os principais problemas que dificultam o processo de integração de dados. Os itens mencionados seguiram os seguintes critérios de classificação: administração de dados, diferenças entre modelos de dados, diferenças de semântica, aspectos funcionais e os relacionados ao legado.

Administração de dados – Se refere à falta de um sistema que possibilite a administração de forma centralizada e eficaz das informações consolidadas sobre os dados a serem integrados. Em decorrência dessa situação, podem surgir problemas relacionados à segurança de acesso integrada, onde falta controle sobre quem tem acesso aos dados; de desempenho, onde acessos externos indiscriminados podem prejudicar os usuários do domínio local; ou de inconsistência, onde os dados são processados sem o devido conhecimento de seu significado. É importante que haja um repositório para o controle administrativo dos dados que forneça as seguintes informações:

- Descrição geral - Qual a origem de cada dado, quem é o responsável pela informação; quais os sistemas que a utilizam; quais as suas dependências.
- Distribuição de dados – Se um dado está distribuído, qual a forma de distribuição e o fluxo que o dado percorre. Se as cópias replicadas forem somente para leitura, de que maneira estão protegidas de serem atualizadas. Se sofrerem atualização, como a atualização é refletida na fonte.
- Forma de armazenamento dos dados em cada fonte - Em que ambientes os dados estão armazenados, em quais meios, quais os formatos.
- Descrição de cada fonte – Qual o tipo de fonte de dados, a plataforma, protocolos, servidor, número da porta, entre outros.
- Propriedades dinâmicas (comportamentais) de operação sobre os dados – Quais as regras e gatilhos envolvidos em cada operação de atualização dos dados.
- Formas de acesso – Quem está autorizado a acessar os dados, de que forma, como é feito o controle de acesso. Se o critério de acesso na fonte é seguido pelas cópias, ou seja, um

dado tem um determinado nível de segurança em sua origem, mas depois que é distribuído ele tem o seu acesso liberado sem a devida proteção.

- Semântica do dado – Algumas informações relativas ao significado e representação do dado no contexto do seu domínio.

Diferenças de SGBD – Um sistema corporativo pode utilizar SGBDs diferentes, que acarreta problemas quanto a linguagens de manipulação de dados, estruturas de armazenamento, facilidades de gerenciamento e interfaces não padronizadas.

Diferenças entre modelo de dados – Problemas de integração devido a diferenças de modelo são relativos à presença de determinados conflitos que aumentam a complexidade do mapeamento entre os esquemas de banco de dados, mesmo que os modelos sejam semelhantes. Para maiores detalhes sobre tais conflitos, uma boa fonte é Battini e Navathe[01] que discutem os principais problemas para integração de esquemas de bancos de dados e fazem uma análise comparativa entre diversas metodologias existentes para essa área. Por exemplo, um conceito pode estar representado de forma diferente em dois ou mais esquemas, devido a um conceito estar representado de forma diferente, quer seja parcial ou total, em dois ou mais esquemas. Divergências podem surgir no momento da agregação ou comparação com outras fontes, tais como conflitos de nomenclatura, de estrutura, de restrições comportamentais e de linguagem de manipulação. Os seguintes problemas podem ocorrer:

- Conflitos de nomenclatura – Os diversos sistemas em uma organização utilizam uma ontologia própria para dar nomes e classificar os dados que compõem o seu universo. Portanto, é natural que no momento da integração com outros sistemas possam surgir alguns conflitos. A divergência mais comum é relativa ao vocabulário empregado, não somente no nível conceitual, mas também no nível da implementação. Este último ainda tem certas restrições relativas às regras de nomenclatura das fontes de dados. Algumas fontes não aceitam determinados caracteres, outras obrigam que o nome comece com um caractere alfanumérico, além de restrições quanto ao tamanho do nome do dado. Outra fonte de divergência entre nomenclaturas pode inclusive surgir da metodologia de padronização de nomes. Muitas vezes é usada uma codificação que inclui informações no nome do dado a respeito do sistema ou entidade ao qual ele está vinculado. Há, ainda, os problemas de homônimos e sinônimos. Problemas de homônimos ocorrem quando o

mesmo nome é usado com conceitos diferentes. Os de sinônimo ocorrem quando o mesmo conceito é descrito por diferentes vocábulos.

- Conflitos estruturais – são os problemas que surgem devido a diferentes modelagens ou restrições de integridade. Nesse caso, as diferenças que podem surgir são relativas: ao tipo, à cardinalidade, à chave de identificação e ao comportamento.
  a) Tipo - diferentes tipos podem ser associados a um mesmo dado conceitual. Um dado pode ter sido modelado como entidade em um esquema e como atributo em outro, por exemplo.
  b) Cardinalidade – as dependências entre um grupo de conceitos são mapeadas com diferentes cardinalidades. Por exemplo, em um esquema o relacionamento entre duas entidades pode estar mapeado como 1:1 e em outro 1:N.
  c) Chave de identificação – chaves de identificação diferentes são definidas para o mesmo conceito em esquemas distintos.
  d) Comportamento – quando o modelo de dados permite uma representação comportamental dos objetos, ou seja, de suas propriedades dinâmicas, pode haver conflitos quanto ao comportamento em relação a uma determinada operação. Por exemplo, diferentes comportamentos podem ser associados à exclusão de um contrato.
- Restrições de integridade – modelos de dados podem suportar diferentes tipos de restrições. Por exemplo, um modelo em rede pode não ter a equivalência das restrições de integridade em um esquema relacional e vice-versa.

Diferenças de Semânticas – Divergências que surgem devido ao significado, interpretação, ou intenção de uso de um mesmo conceito de dado. Por exemplo, o preço de um produto em uma fonte de dados F1 pode ter embutido o imposto e em F2 não. Cliente em F1 pode ser qualquer pessoa (física ou jurídica) que tenha comprado um produto, e em F2 pode ser aquela que tenha enviado uma proposta, embora não tenha efetivado nenhuma compra.

Aspectos funcionais – refletem os problemas operacionais devido, principalmente, à qualidade de serviço na distribuição de dados. Podem envolver:

- Redundância de informações na mesma fonte de dados – necessidade de manutenção de informações redundantes, que muitas vezes são referentes à mesma entidade, no mesmo SGBD, porém vinculadas a sistemas diferentes. Por vezes, na mesma tabela existem dois

campos com a mesma informação, mas com formatos diferentes para melhorar o desempenho de junção com tabelas de outros sistemas.

- Desempenho dos sistemas locais – usuários ou processos remotos podem fazer acessos demorados indiscriminadamente, concorrendo com os usuários locais dos sistemas, dificultando a disponibilidade das informações.
- Inconsistência da informação – processos de atualização são vulneráveis a diversas falhas a cada vez que arquivos são selecionados, processados, descarregados, replicados ou transferidos.
- Validação da integridade da informação – falta de mecanismos que permitam verificar a consistência dos dados em todas as suas fontes.
- Duplicação de procedimentos – duplicação de procedimentos para transferência ou manutenção de dados devido a problemas de interoperabilidade. A mesma informação pode ser processada várias vezes por diferentes *interfaces* para conversão e transferência dos dados. Quando a informação muda na fonte é preciso atualizar todas elas; havendo falta de controle do que existe, alguns podem ser deixados de fora.
- Medições – dificuldades em medir custo, quantidade de tempo e recursos para manter os dados distribuídos na rede. Em decorrência, também há dificuldade para dividir os custos entre os que compartilham os dados.
- Dependência de terceiros – dados são gerados e manipulados por pacotes de terceiros, ou estão sob responsabilidade de usuários que os mantêm em planilhas ou bases pessoais. Isto é sempre uma dificuldade adicional para a integração desses dados, pois se enfrentam barreiras comerciais, políticas ou pessoais quanto ao domínio da informação.

Legado – Dificuldades relativas a dados legados, entre outros motivos, podem ser causadas por uma tecnologia que foi descontinuada, ou que não possua mais suporte do fabricante, ou ainda originadas da falta de conhecimento técnico para interagir com ela. Mesmo quando as barreiras tecnológicas podem ser transpostas, podem surgir problemas que dificultam o entendimento da funcionalidade interna do sistema devido à falta de documentação e conhecimento do significado das estruturas existentes. Outras vezes, pode até mesmo ocorrer ausência de código fonte dos aplicativos.

2.3.6 Estratégias para a Integração de dados

Uma abordagem prática para a integração de dados em uma empresa precisa considerar estratégias que englobem atividades de modelagem de dados, aplicações e processos. Essas estratégias levam em conta conceitos e termos tais como metadados, esquemas, padrões e análise de requisitos. Geralmente cada empresa adota sua própria metodologia para o projeto de integração de dados. Nesta seção, será apresentada uma relação de atividades, parcialmente adaptadas de [19], que normalmente são consideradas neste tipo de projeto.

Entendimento do negócio da empresa e do domínio do problema – esse tipo de atividade requer interação com várias pessoas da empresa e familiarização com a estrutura e conteúdo dos sistemas de informação, assim como dos requisitos de negócio de cada organização. É preciso identificar como os negócios são realizados, e o que é importante ou não para cada assunto. Requer interagir com pessoas, papéis, e sistemas computacionais para determinar as informações necessárias visando: análise correta, modelagem, ajustes e implementação da solução.

Entendimento das bases de dados já existentes – Engloba a identificação dos dados do ponto de vista do negócio e da tecnologia, ou seja as características lógicas e físicas das bases a serem integradas. Envolve informações diversas, tais como, quem são os responsáveis pelos dados, onde eles estão fisicamente alocados, como foram modelados e como os dados são usados dentro do contexto do negócio. Nessa fase, os dicionários de dados existentes são extremamente úteis para a compreensão das regras necessárias para a construção das bases de dados e dos controles de integridade para a integração correta com outras bases. É necessário levantar as regras que estão acopladas com os dados nas bases, encapsuladas em SGBDs (em *stored procedures*, *triggers* e *rules)* ou embutidas nos aplicativos. A latência dos dados também é uma característica relevante, pois define o quão atualizados os dados precisam estar no contexto de cada aplicação, para saber qual a freqüência que os dados precisam ser copiados ou movidos. Finalmente, é preciso fazer um levantamento de como a informação está estruturada e as propriedades dos elementos de dados tais como: tamanho, tipo do dado (numérico ou caractere), nome do elemento, tipo de informação armazenada (binário, texto, espacial, etc.).

Entendimento dos processos – Possibilita a visão da empresa no nível de processos e serviços através do entendimento e documentação de todos os processos de negócios e como eles se relacionam uns com os outros.

Catalogação dos dados e processos – reúne metadados tradicionais como formato, nome e descrição dos atributos com outras informações pertinentes à integração, como por exemplo, sistemas, segurança, processos e mecanismos de comunicação. O catálogo global da empresa definirá a base da solução, contendo informações tanto sobre os dados como sobre o negócio.

Construção do meta-modelo de dados e de negócios – nessa atividade há a definição de todas as estruturas de dados e negócios das empresas e como esses modelos irão interagir com o domínio da solução da integração adotada. O modelo de metadados pode ser dividido em lógico e físico. Sempre que for viável, o modelo deve ser normalizado, eliminando-se redundâncias, para que a empresa possa ter, tanto quanto possível, uma única fonte para cada informação.

Identificação das interfaces das aplicações – É importante conhecer as interfaces das aplicações disponíveis para suporte à integração. É preciso reunir essas informações em um repositório ou diretório, junto com a documentação de cada interface, estendendo o modelo de metadados para refletir não somente os dados, mas também os métodos que atuam sobre esses dados. Deve refletir a semântica da aplicação, estabelecendo a forma como uma determinada aplicação se referencia às propriedades dos processos de negócios. Essas propriedades se referem às listas de funções e métodos fornecidos pela aplicação. Essas informações, junto com o metamodelo de negócios e o de dados, darão o entendimento dos pontos de integração necessários para a solução do problema.

Identificação dos eventos de negócio – É preciso identificar quais as ações resultantes da ocorrência de um evento. É importante entender o que disparou o evento, o que acontece durante o evento, e quais os eventos que precisam ser disparados como conseqüência do evento inicial. A solução final da integração deve automatizar todos os eventos inter-relacionados.

Identificação dos cenários de transformação de dados – Já com o entendimento da semântica e estrutura dos dados e aplicações dentro do domínio da integração, é preciso identificar as transformações necessárias na movimentação dos dados, para que as semânticas e estruturas se mantenham consistentes em todos os sistemas.

Mapeamento da movimentação dos dados – É preciso identificar quais elementos de dados e interfaces estão sendo movimentados, identificando suas origens e destinos. Identificar onde a informação está fisicamente armazenada, mecanismos de segurança, o tipo de fonte de armazenamento, e como a informação é extraída e carregada em outro local. É necessário

também identificar quais os eventos que estão acoplados com os dados que serão movimentados (por exemplo *stored procedure* ou *trigger*).

Definição da arquitetura e tecnologia a ser usada na solução – A solução escolhida pode ser adquirida de um ou vários fornecedores, ou customizada, - desenvolvida na própria empresa ou por terceiros. Seja qual for a decisão, é importante que o produto final tenha, entre outras, as seguintes características:

- Prover conectividade entre os sistemas de origem e destino através de algum tipo de ponto de integração, fornecendo suporte para conexões orientadas a informação, orientadas a serviços e orientadas a transação. A conectividade também deve fornecer suporte para abstrações física e lógica, para que as informações sejam independentes de sua localização física ou estrutura lógica.
- Transformação – deve prover suporte para transformação de forma a resolver diferenças na semântica, conteúdo e tipos de dados.
- Roteamento da informação – junto com a transformação, o roteamento da informação é uma característica importante na movimentação dos dados de um sistema para outro. É importante que o roteamento possa ser baseado no conteúdo da mensagem, identificando a aplicação de origem e roteando a informação para a aplicação destino apropriada, fazendo a devida tradução. Esse serviço deve a) rotear a informação vinda de uma ou várias origens, b) dividir a informação que vem de uma origem enviando para vários destinos, c) combinar a informação vinda de várias origens enviando para um único destino, d) filtrar a informação antes de enviá-la para o seu destino.
- Suporte para transações – Deve suportar tanto a noção de transações ACID (Atomicity, Consistency, Isolation, Durability) como não-ACID, prevendo tanto o suporte para transações longas como as de curta duração.
- Suporte para configuração e gerenciamento da integração – Deve suportar monitoração, controle da segurança de acesso, otimização de acesso, registros de logs, rastreamento, etc., de forma parametrizada e de fácil configuração.

Implementação da solução de integração – Além de escolher (ou desenvolver) e testar exaustivamente o produto para solucionar o problema da integração, é preciso definir parâmetros e procedimentos para o ambiente de produção. Essa atividade envolve questões tais como, desempenho, segurança, monitoração e plano de desastre/recuperação. Estes requisitos são muito

importantes para que a integração, ao invés de ser uma solução, não se transforme em um gargalo, ou ponto de falha trazendo um impacto negativo no ambiente de produção da empresa.

2.4 Conclusão

Este capítulo abordou alguns conceitos que servem como base ao resto da dissertação, em especial os relativos à distribuição e integração dos dados em uma empresa. O próximo capítulo detalha questões de arquiteturas orientadas a serviços, uma solução cada vez mais disseminada para facilitar interoperabilidade.

Capítulo 3

ARQUITETURA ORIENTADA A SERVIÇOS

Este capítulo discute aspectos de arquiteturas orientadas a serviços (SOA) e serviços Web, a abordagem atualmente mais comum na implementação SOA. Sendo assim, falaremos sobre tecnologias para serviços Web, como SOAP, WSDL (Web Services Description Language), UDDI (Universal Description, Discovery and Integration) e XML, assim como outros itens importantes que envolvem questões de qualidade de serviços, segurança e gerenciamento de serviços Web. As noções descritas servem de base para a proposta de arquitetura deste trabalho, descrita no capítulo 4.

3.1 Introdução

A arquitetura orientada a serviços, também conhecida pelo acrônimo em inglês SOA (Service Oriented Architecture), apresenta uma abordagem para construção de sistemas distribuídos [12].

A arquitetura orientada a serviços é uma abordagem para definição de arquiteturas de integração baseadas no conceito de serviço [31]. Dentro dessa abordagem, um serviço incorpora uma funcionalidade que é definida na sua descrição. Essa descrição também conterá informações sobre o serviço relativas ao comportamento, à sintaxe e à semântica. Além disso, os serviços também são descritos e organizados de maneira que haja suporte para sua descoberta dinâmica em tempo de execução [24]. Serviços possuem completa autonomia em relação a outros serviços. Cada serviço é responsável por seu próprio domínio, que geralmente está limitado ao escopo de uma função específica (ou um grupo de funções relacionadas) definida por uma *interface* independente de plataforma. Além disso, um serviço encapsula funções reusáveis e é fracamente acoplado e invocado através de protocolos de comunicação que permitem transparência de localização e interoperabilidade.

Serviços são criados como unidades isoladas com funcionalidades específicas e reusáveis, que se comunicam por protocolos padrões interagindo de uma maneira fracamente acoplada. Além disso, a independência que esses serviços necessitam ter dentro do conceito de uma SOA

acaba por determinar que a programação por eles encapsulada não se preocupe em obedecer a uma determinada plataforma ou tecnologia.

A arquitetura orientada a serviços é uma abordagem que agrega técnicas e estilos já provados de outras arquiteturas precedentes, como a orientada a objetos ou a baseada em componentes. As raízes da orientação a serviço podem ser encontradas em três diferentes áreas [18]:

- Paradigmas de programação – fornecem a plataforma de implementação para os diferentes elementos de uma SOA e têm influência nas técnicas de *interface* e nos padrões de interação que são empregados entre os provedores e consumidores de serviço.
- Tecnologias de distribuição – fornecem a tecnologia para acesso remoto dos serviços fornecidos por diferentes aplicações em diferentes plataformas.
- Evolução da computação empresarial – possibilita um grande número de aplicações empacotadas, proprietárias e heterogêneas que hoje fornece o conteúdo (dados e lógica do negócio) que leva à necessidade de um estilo de integração que tenha uma abordagem como a da SOA.

As implementações baseadas em uma SOA, envolvendo interação em ambiente distribuído e fracamente acoplado, trazem preocupações adicionais. Portanto, em um projeto que envolva a adoção de uma SOA, questões sobre qualidade de serviço, principalmente segurança e gerenciamento adequado, são fatores que precisam ser tratados de maneira diferenciada e aprofundada.

3.2 Conceitos básicos de uma SOA

Nesta seção serão discutidos os conceitos básicos de uma SOA, englobando aspectos gerais da arquitetura (seção 3.2.1) e seus artefatos, operações, participantes e relacionamentos (seção 3.2.2).

3.2.1 Visão Geral

De acordo com a documentação do W3C (World Wide Web Consortium) [21], uma arquitetura orientada a serviços é tipicamente caracterizada pelas seguintes propriedades:

- Visão lógica – O serviço é uma visão abstrata e lógica de programas reais, bases de dados, processos de negócios, etc., definido em termos de sua funcionalidade.
- Orientada a mensagem – O serviço é formalmente definido em termos das mensagens trocadas entre os agentes provedores e os agentes requisitantes, e não das propriedades dos

agentes em si. Não há necessidade de se saber como o agente que implementa um serviço foi construído, pois a sua estrutura interna, linguagem de implementação ou quaisquer outras características são abstraídas na SOA.

- Orientada à descrição – Um serviço é descrito por metadados processáveis por máquina. A descrição suporta a natureza pública da SOA: somente os detalhes que serão expostos para o público e importantes para o uso do serviço devem ser incluídos na descrição. A semântica de um serviço deve ser documentada, direta ou indiretamente, através de sua descrição.
- Granularidade – Serviços tendem a usar um número pequeno de operações com mensagens relativamente longas e complexas.
- Orientada à rede – Serviços tendem a ser orientados ao uso de uma rede; todavia isso não é um requisito absoluto.
- Independência de plataforma – Mensagens são enviadas em um formato padrão, independente de plataforma e entregues através de *interfaces*. O formato XML é o que mais se adapta a essas características.

A SOA não possui ainda um modelo de referência, sendo que o comitê técnico OASIS Service Oriented Architecture Technical Comitee [29] está designado para essa tarefa. A figura 3.1, adaptada de [12], destaca os elementos funcionais e não-funcionais de uma arquitetura orientada a serviços que satisfaz os critérios propostos para estruturar este trabalho.

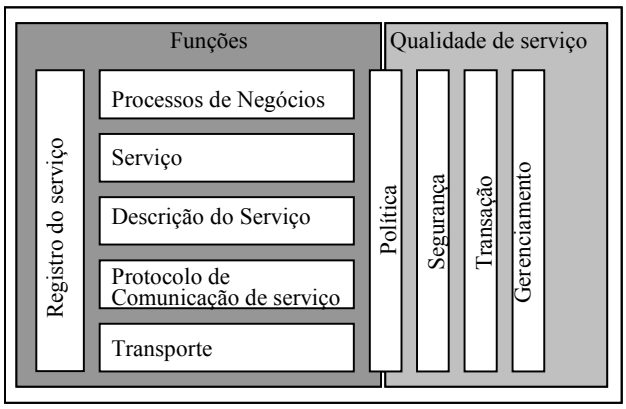

Figura 3.1 – Elementos de uma Arquitetura orientada a serviços (SOA)

Os aspectos funcionais incluem:

- Transporte – mecanismo usado para movimentar as requisições e respostas de serviço entre os consumidores de serviço e os provedores de serviço.

- Protocolo de comunicação de serviço – mecanismo acordado entre o consumidor e o provedor do serviço e usado para comunicar o que está sendo requisitado e o que está sendo retornado.
- Descrição do serviço – é um esquema combinado para descrever o serviço, a forma como deve ser chamado e quais dados são necessários para executá-lo. A descrição do serviço é considerada um artefato da SOA e como tal será descrita na seção 3.2.2.
- Serviço – descreve um serviço real que está disponível para uso. Também é um artefato da SOA e será descrito na seção 3.2.2.
- Processo de negócio – coleção de serviços invocados em uma seqüência particular com um conjunto de regras específicas para adequar-se a um requisito de negócio. Um processo de negócio poder ser considerado um serviço, pois pode ser composto de serviços de diferentes granularidades.
- Registro de serviço – contém o repositório de serviços e descrição dos dados associados. O registro é usado por a) provedores de serviços para publicarem seus serviços e b) consumidores de serviços para descobrirem os serviços disponíveis. O registro de serviços é um dos participantes de uma interação na SOA e será descrito com mais detalhes na seção 3.2.2.

Os aspectos referentes à qualidade de serviço são:

- Política – conjunto de condições ou regras sob as quais um provedor de serviço torna o serviço disponível aos consumidores. Há aspectos da política que são funcionais, e aspectos que se relacionam à qualidade de serviço. Por exemplo, um provedor de um determinado serviço pode ter como política: a) que todos os consumidores precisam ser autenticados antes de acessarem quaisquer funcionalidades de seu serviço, b) que todas as mensagens sejam criptografadas, c) que sejam guardados registros das mensagens trocadas. Políticas também podem estar associadas ao negócio e não somente à infra-estrutura, como por exemplo, o horário de disponibilização do serviço.
- Segurança – conjunto de regras que precisam ser aplicadas para identificação, autorização e controle de acesso dos consumidores de serviço que invocam os serviços. Por exemplo, utilização de HTTPS, HTTP via SSL (Security Socket Layer), que permite a autenticação via certificados. Nesse caso, um conjunto de regras possível seria: a) o provedor e o consumidor devem negociar a versão do protocolo SSL, b) se o certificado for

desconhecido para o consumidor, ele pode aceitar ou rejeitar o certificado, c) será usada uma estrutura PKI (Public Key Infrastructure) de entidades certificadoras independentes para validação dos certificados usados no processo de criptografia das mensagens trocadas.

- Transação – conjunto de atributos que precisam ser aplicados a um grupo de serviços para a entrega de um resultado consistente. Por exemplo, se um grupo de três serviços é necessário para completar uma função de negócio, todos precisam completar ou nenhum pode ser completado.
- Gerenciamento – conjunto de atributos que precisam ser aplicados para gerenciamento dos serviços fornecidos ou consumidos. Por exemplo,

Em relação à integração de aplicações e dados, a arquitetura orientada a serviços introduz uma nova camada lógica em uma plataforma de computação distribuída. A camada de integração de serviço estabelece um ponto comum de integração entre camadas de aplicação ou entre aplicações, como visto nas figuras 3.2 e 3.3, respectivamente, adaptadas de [20].

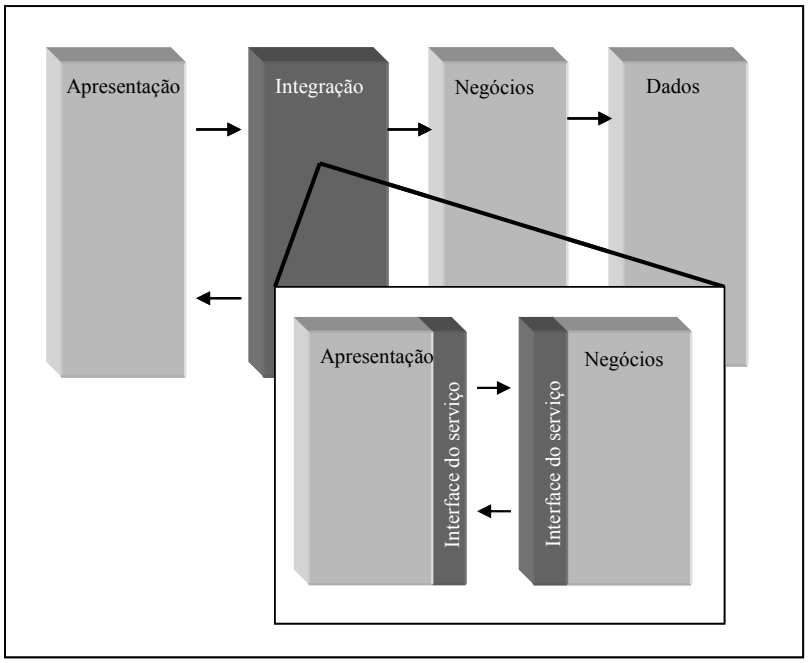

Figura 3.2 – Camada de integração de serviço em uma aplicação

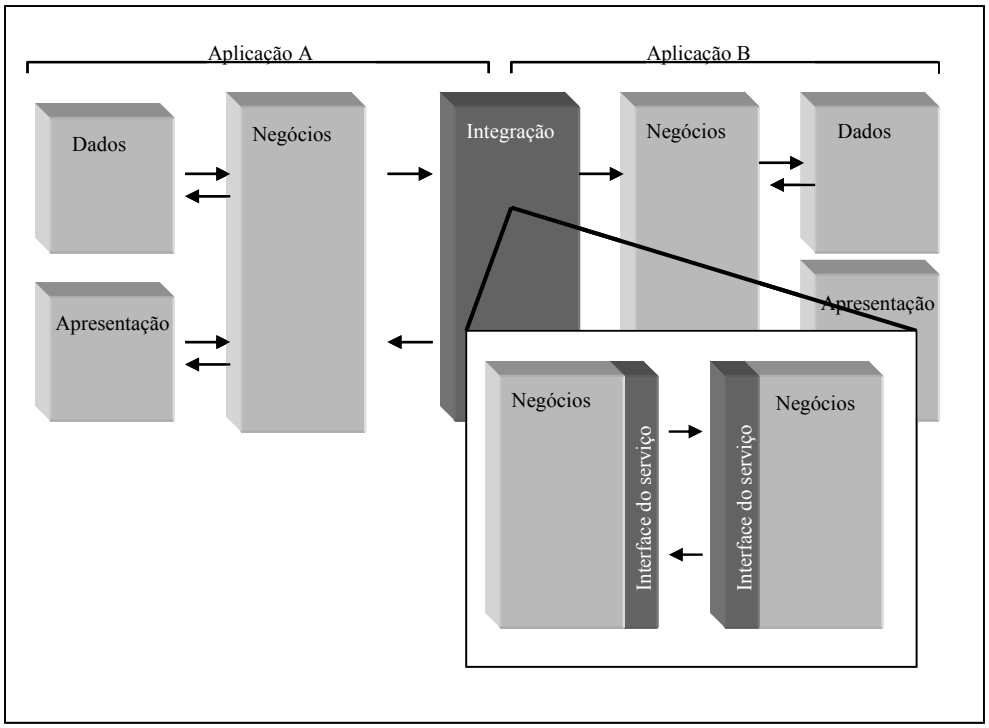

Figura 3.3 – Camada de integração de serviço entre aplicações

Hoje em dia é comum que um ambiente computacional dentro de uma empresa englobe várias tecnologias e conceitos de computação distribuída. Um ponto importante em relação à SOA, ou qualquer arquitetura que tenha como objetivo a integração de aplicativos e dados, é a capacidade de incorporar essa diversidade e heterogeneidade.

Para possibilitar a integração entre aplicativos e fontes de dados, as empresas precisam executar a tarefa de habilitação de serviço. A habilitação de serviço é o processo que cria um serviço para encapsular a funcionalidade fornecida por uma aplicação existente. O tipo de aplicação encapsulada pode ser qualquer um, desde uma aplicação de *mainframe* até uma aplicação distribuída construída em cima de CORBA ou DCOM (Distributed Component Object Model) [18].

As duas abordagens mais comuns para a habilitação de serviço são as *proxies* de serviço e os adaptadores de serviço [32]. *Proxies* e adaptadores já são largamente utilizados no contexto de integração de aplicativos e também podem ser usados para habilitarem fontes de dados como serviços.

A finalidade de uma *proxy* de serviço é criar um *wrapper* externo por meio do qual é executado o acesso à fonte de dados. As *proxies* de serviço são usadas para esconder os detalhes sobre o que uma entidade ou consumidor precisa fazer para estabelecer uma sessão com a fonte

de informação, transformar a requisição do serviço em um conjunto apropriado de interações de forma a retornar ou atualizar as informações e então formatar a informação que será retornada ao consumidor [32]. No modelo de *proxy* de serviço há uma conexão de rede entre o consumidor e a *proxy*, assim como uma conexão entre a *proxy* e a fonte de dados.

Adaptadores de serviço são uma forma integrada ou embutida de uma *proxy* de serviço [32]. A principal diferença entre as duas abordagens é que o adaptador é integrado na lógica do processamento "requisição/resposta" de uma fonte de dados em vez de externamente na fonte de dados. Diferente do modelo de *proxy* de serviço, o modelo utilizado por adaptadores somente requer a conexão entre o consumidor e o adaptador. A fonte de dados é diretamente mediada pelo adaptador.

Embora a exposição de fontes de dados como serviços seja importante, fica faltando ainda solucionar uma série de problemas, tais como visão unificada da informação ou transformação de dados [32]. Este trabalho de mestrado propõe no capítulo 4 uma abordagem para endereçar especificamente esses problemas, típicos de integração de dados dentro de uma empresa com um legado tecnológico heterogêneo e distribuído.

Uma outra questão muito importante para a integração no ambiente empresarial está relacionada com a abordagem adotada para a integração de dois componentes de *software* distribuídos. É preciso analisar e decidir sobre questões que têm um impacto no acoplamento dos dois sistemas, tais como: infra-estrutura de comunicação, tipo de chamada de semântica, semântica de *interface* (por exemplo, RPC) ou semântica de *payload* (por exemplo, MOM), uso de intermediários para o acoplamento físico, padrão de interação (por exemplo, foco nos dados ou orientado a objetos), estilo de comunicação (síncrono ou assíncrono), descoberta e ligação dos serviços (estático ou dinâmico).

O acoplamento tipicamente se refere ao grau em que os componentes de um *software* são dependentes uns dos outros. Em comunicações tradicionais programa-a-programa fortemente acoplados, os programadores precisam dizer à aplicação "A" onde achar a aplicação "B". Isso exige definir e programar conexões e relacionamentos entre as aplicações envolvidas. Por outro lado, é mais fácil considerar confiabilidade, segurança e gerenciamento, pois ambos os sistemas finais são conhecidos.

Aplicações fracamente acopladas, por sua vez, não requerem que a comunicação entre elas seja programada. Apesar de fornecerem benefícios como busca dinâmica, flexibilidade e

interoperabilidade entre ambientes heterogêneos, podem requerer mecanismos suplementares para garantir confiabilidade, segurança e gerenciabilidade. Além disso, essa integração flexível entre sistemas faz com que as definições de padrões para a troca de dados independentemente de plataforma seja um fator relevante. Implementações de SOA baseadas em Web são um exemplo desse tipo de abordagem.

Com a implementação de uma arquitetura orientada a serviços, podemos visualizar alguns benefícios para ajudar as organizações a serem bem-sucedidas na integração de seus sistemas dentro do atual cenário dinâmico e heterogêneo. A SOA fornece uma camada de abstração que possibilita o reaproveitamento de aplicações legadas, encapsulando-as como serviços em vez de reconstruir tudo de novo. Além disso, o ponto de integração em uma SOA é a especificação do serviço, e não a sua implementação. Isso fornece transparência de implementação e minimiza o impacto quando ocorrem mudanças na infra-estrutura ou na implementação. Tudo isso leva a uma diminuição do tempo e dos custos no desenvolvimento e manutenção do *software*.

3.2.2 Interações em uma SOA

As interações em uma SOA seguem o paradigma "localiza, vincula e invoca" e envolvem basicamente três tipos de participantes: o provedor do serviço, o registro do serviço e o consumidor do serviço.

Os provedores são participantes que oferecem serviços. Eles definem as descrições de seus serviços e as publicam em um registro, que é um repositório que suporta pesquisas. Os consumidores usam uma operação de pesquisa para localizar os serviços de seu interesse. O registro retorna a descrição do serviço e o consumidor usa as informações contidas nessa descrição (por exemplo, a localização na rede) para invocar e usar o serviço. A figura 3.4, adaptada de [23], mostra as interações entre os participantes da arquitetura, bem como seus artefatos e operações, que serão detalhados a seguir.

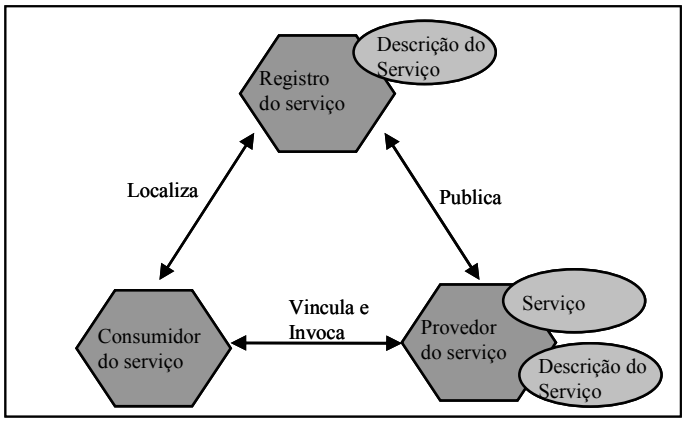

Figura 3.4 – Interações entre os participantes de uma SOA e seus artefatos e operações

Na arquitetura orientada a serviços, cada entidade pode atuar em um ou mais dos três papéis descritos a seguir:

- **Provedor de serviços** – Sob a perspectiva do negócio, é o proprietário do serviço. Sob a perspectiva da arquitetura, é a plataforma que hospeda o acesso ao serviço [23], sendo uma entidade endereçável na rede que aceita e executa pedidos de consumidores. O provedor publica seus serviços e seus contratos de *interface* no registro de serviços de forma que os consumidores possam descobri-los e acessá-los.
- **Consumidor de serviços** – Sob a perspectiva do negócio, é aquele que necessita de certas funções. Sob a perspectiva da arquitetura, é a aplicação que está procurando e chamando (ou iniciando) uma interação com um serviço [23]. O consumidor do serviço pode ser uma aplicação, um módulo de *software* ou até mesmo um outro serviço.
- **Registro de serviços** – É um facilitador para a descoberta de serviços. Ele contém um repositório com serviços disponíveis e permite busca das *interfaces* dos serviços. Os requisitantes de serviços procuram serviços obtendo informações para a ligação (nas descrições dos serviços) que poderá ser: ligação estática, durante o desenvolvimento, ou ligação dinâmica, durante a execução. Para requisitantes de serviços com ligação estática, o registro de serviços é um participante opcional na arquitetura porque um provedor de serviços pode enviar a descrição diretamente para os requisitantes [23]. Os requisitantes de serviços podem obter uma descrição de outras fontes, como um arquivo local, um *site* FTP ou um *Web site.*

As operações em uma arquitetura orientada a serviços são:

- **Publicação** – A descrição de um serviço necessita ser publicada para que os consumidores dos serviços possam encontrá-la, acessá-la e usá-la.
- **Localização** – O cliente do serviço pode recuperar diretamente a descrição do serviço ou efetuar uma busca no registro de serviço para localizar o tipo de serviço desejado. A operação de pesquisa pode ser feita a) em tempo de desenvolvimento da aplicação, para recuperar as descrições das *interfaces* do serviço, ou b) em tempo de execução, para recuperar as descrições de localização (e ligação) do serviço para poder efetuar a sua chamada.
- **Ligação** – O cliente do serviço invoca ou inicia uma interação com o serviço em tempo de execução, usando a descrição do serviço para localizar, contatar e chamar o serviço.

Os artefatos em uma arquitetura orientada a serviços são:

- **Serviço** – Um serviço é um módulo de programa desenvolvido em uma plataforma acessível via rede, fornecido pelo provedor do serviço e disponível para uso através de uma *interface* publicada, permitindo que ele seja chamado por um consumidor de serviço. Ele também pode funcionar como um cliente, usando outro serviço em sua implementação.
- **Descrição do serviço** – A descrição do serviço contém os detalhes da *interface* e implementação do serviço especificando como um consumidor do serviço vai interagir com o provedor do serviço. Isso inclui seus tipos de dados, operações, informações de ligação, localização na rede e o formato das requisições e respostas dos serviços. Também pode incluir uma série de condições, níveis de qualidade de serviços, classificação por categoria e outras subdivisões para facilitar a descoberta e utilização pelos clientes de serviço.

O modelo orientado a serviços simplificado é ilustrado na figura 3.5, adaptada de [21], e está baseado primordialmente em:

- Um serviço é realizado por um agente e usado por um outro agente.
- Serviços são mediados através de mensagens trocadas entre os agentes requisitantes e os agentes provedores.
- Serviços são implementados para oferecerem funcionalidades no mundo real. Isso é modelado através da elaboração do conceito de um proprietário do serviço, o qual tem uma responsabilidade no mundo real por esse serviço. Por exemplo, serviços bancários

(como extrato de movimentação ou saldo), serviço de emissão de proposta de seguro, serviço de reserva de passagem aérea.

- Metadados são fundamentais, pois são usados para documentar vários aspectos dos serviços. O fornecimento de descrições robustas é vital para uma implementação bem-sucedida e para o uso de serviços na integração de dados e aplicações. Exemplo

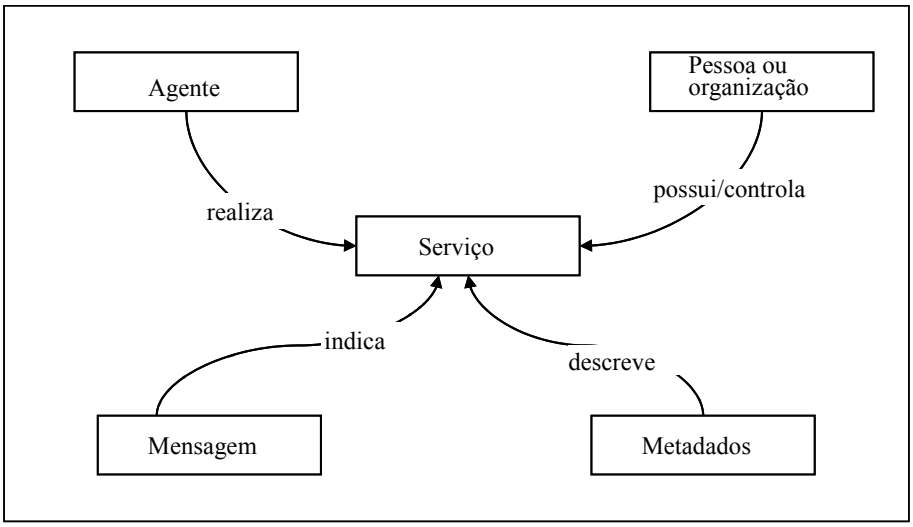

Figura 3.5 – O modelo orientado a serviços simplificado

3.3 Serviços Web

Os serviços Web são uma tecnologia que se ajusta bem em uma implementação de SOA [12]. Na essência, os serviços Web são aplicações modulares e autodescritivas que expõem lógica de negócios como serviços e que podem ser publicados, descobertos e invocados através da Internet. Baseados em padrões XML, os serviços Web podem ser desenvolvidos como componentes de aplicações fracamente acopladas usando qualquer linguagem de programação, protocolo ou plataforma.

No entanto, é importante destacar que há uma distinção entre uma aplicação que usa serviços Web e uma aplicação baseada em uma arquitetura orientada a serviços [20]. Uma SOA é um modelo com um conceito de encapsulamento da lógica da aplicação na forma de serviços que interagem através de um protocolo de comunicação comum. Quando os serviços Web são usados para estabelecer esse *framework* de comunicação, eles representam uma implementação de uma SOA baseada em Web. Porém os serviços Web não são a única tecnologia que pode ser usada para implementar uma arquitetura orientada a serviços, assim como os serviços Web também

podem ser usados para implementar outros tipos de arquiteturas que não a SOA. Neste trabalho, todavia, focaremos o uso de serviços Web como forma de implementação de uma SOA.

3.3.1 Visão geral

A estrutura e implementação da arquitetura de serviços Web é particular a cada empresa. No entanto, para que a interoperabilidade seja garantida, certos conceitos, relacionamentos e restrições são importantes. Nesse caso, para uma real interoperabilidade, um item muito importante é a existência de padrões definidos pela indústria. Existe um esforço muito grande para padronizar os serviços Web, liderado principalmente por consórcios de empresas. Todas essas organizações estão tentando gerar um consenso para a aceitação de padrões em comum. Em [30] está disponível uma planilha com um resumo dos padrões mais importantes, as empresas e consórcios envolvidos em cada especificação, bem como o *status* atual de cada trabalho. Entre essas organizações se encontram:

- **W3C (World Wide Web Consortium)** – desenvolve tecnologias interoperáveis (especificações, direcionamentos, *softwares* e ferramentas), visando o máximo aproveitamento da Web. O W3C é um fórum para informações, comércio, comunicação e entendimento coletivo. *Home page* oficial: www.w3.org.
- **OASIS (Organization for the Advancement of Structured Information Standards)** – é um consórcio global, sem fins lucrativos, que tem como missão o desenvolvimento, a convergência e a adoção dos padrões para *e-business*. *Home page* oficial: http://www.oasis-open.org.
- **WS-I (Serviços Web Interoperability Organization)** – organização cujo foco é promover a interoperabilidade dos serviços Web através de plataformas, sistemas operacionais, e linguagens de programação. *Home page* oficial: www.ws-i.org.

O W3C define um serviço Web como "um sistema de software projetado para suportar interação máquina-a-máquina através de uma rede. Ele tem uma interface descrita em um formato processável por máquina (especificamente WSDL). Outros sistemas interagem com o serviço Web da maneira prescrita por sua descrição usando mensagens SOAP, tipicamente transportadas usando HTTP com serialização XML juntamente com outros padrões Web".

A arquitetura para serviços Web proposta pelo W3C envolve muitas tecnologias em camadas que se inter-relacionam. A figura 3.6, adaptada de [21], ilustra algumas dessas tecnologias.

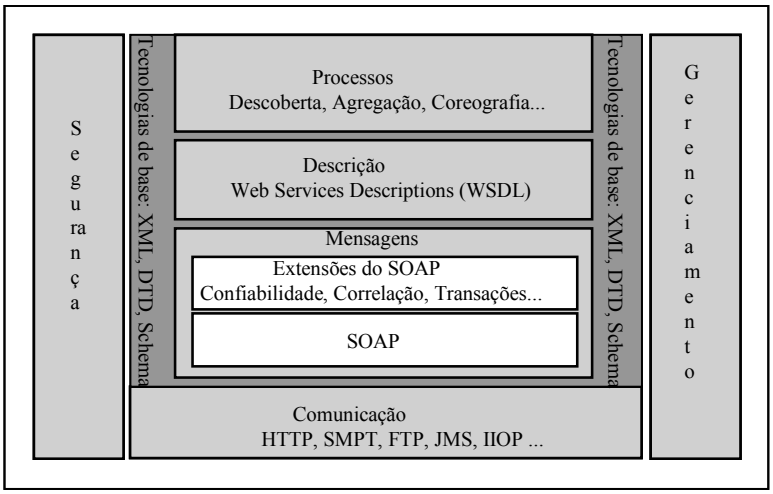

Figura 3.6 – Camadas da arquitetura para serviços Web

Serviços Web podem ser usados em vários cenários de integração de aplicações: do compartilhamento de dados internos, simples, *ad hoc*, ao comércio em larga escala através da Internet [47]. Além disso, serviços Web também estão sendo aplicados em cenários de computação móvel e de *Grid Computing*, como no projeto Global Grid [33].

Do ponto de vista do negócio, os serviços Web podem ser usados para: melhorar a eficiência operacional através da comunicação entre processos, diminuir os custos operacionais devido ao uso de infra-estrutura já existente, promover ou melhorar a integração no mundo corporativo tanto interno quanto externo ou melhorar a flexibilidade organizacional através da exposição de serviços em novos canais e nos já existentes [36].

Do ponto de vista tecnológico, os serviços Web podem trazer:

- interoperabilidade a baixo custo, principalmente quando não se pode assumir nenhum detalhe específico sobre a implementação das aplicações clientes, ou quando as aplicações a serem integradas são heterogêneas, ou ainda quando a infra-estrutura não pode ser mudada;
- redução da complexidade por meio do encapsulamento de *interfaces* e componentes;
- prolongamento da utilização de aplicações legadas.

3.3.2 O modelo dos serviços Web

As interações na arquitetura de serviços Web seguem a de uma SOA. A figura 3.7, adaptada de [23], ilustra as operações e os participantes das interações de serviços Web .

As tecnologias básicas de serviços Web são geralmente descritas em termos de SOAP, WSDL e UDDI, que serão definidas nas próximas seções. Porém, cada um desses padrões pode ser usado isoladamente. Por exemplo, existem muitas implementações que usam somente SOAP, outras somente SOAP e WSDL.

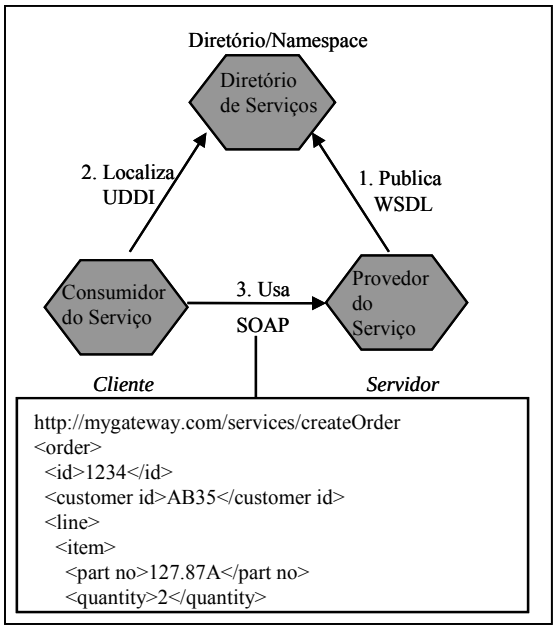

Figura 3.7 – Operações e participantes das interações de serviços Web

Publicação – O local de publicação pode variar dependendo dos requisitos da aplicação [23]. Há várias formas de publicação de serviços, com diversos graus de dinamismo e complexidade de implementação. A maneira mais simples (e estática) é a publicação direta, como um *attachment* de *e-mail*, um *site* FTP ou uma distribuição através de CD-ROM. Para uma publicação mais dinâmica, pode-se usar DISCO (Discovery of Web Services) ou ADS (Advertisement and Discovery of Services), tecnologias da Microsoft e IBM, respectivamente, para publicação e descoberta de serviços Web. Estas definem um mecanismo simples de HTTP para retornar descrições de serviços a partir de um determinado URL (Uniform Resource Locator) [23]. Porém, a forma mais dinâmica (e complexa) é o uso de UDDI, que podem ser públicos ou privados.

Localização – As implementações de localização também vão depender do grau de dinamismo desejado. Se a publicação é do tipo direta, o requisitante faz o *cache* da descrição em tempo de desenvolvimento e usa a informação em tempo de execução. A descrição pode estar estaticamente representada na lógica do programa ou armazenada em um arquivo ou repositório de descrições de serviços local. Quando da utilização de repositórios de WSDL, registro simples

ou nó UDDI, o mecanismo de busca deve suportar uma forma de consulta que forneça, por exemplo, busca por tipo de *interface* (por exemplo, baseado em um *template* WSDL), informações de ligação (ou seja, protocolos), propriedades (parâmetros de qualidade de serviço), tipos intermediários, taxonomias dos serviços ou informações de negócios. Quando for necessário escolher um serviço Web entre vários que satisfizeram a busca, é necessário que haja uma forma de selecionar o serviço desejado baseado em critérios como desempenho, classificação de qualidade de serviço, proximidade geográfica ou balanceamento de carga.

A figura 3.8, adaptada de [12], sumariza o que foi discutido sobre abordagens para implementação, publicação e descoberta de serviços.

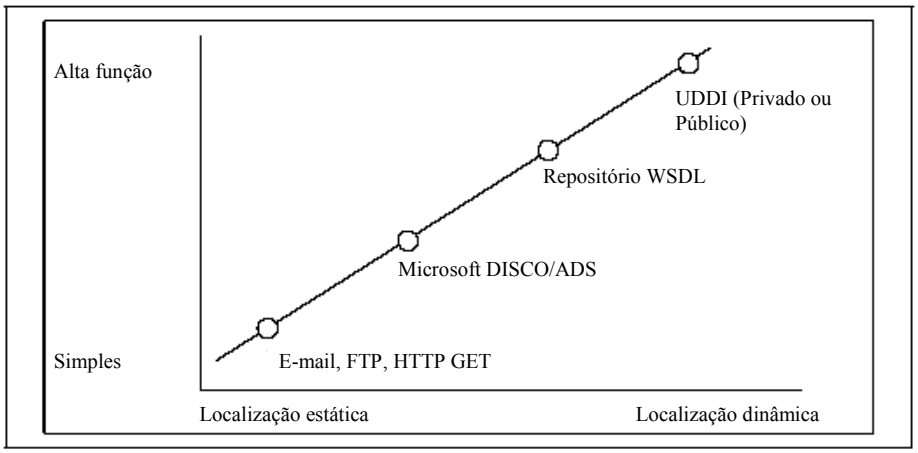

Figura 3.8 – Implementações para publicação e descoberta de serviços X complexidade

3.3.3 Tecnologias para serviços Web

O *framework* do W3C para serviços Web consiste em uma base construída no topo de três especificações XML: Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) e Universal Description, Discovery, and Integration (UDDI) [20]. Esses padrões desenvolvidos com base nos princípios de SOA formam uma SOA baseada em serviços Web. A figura 3.9, adaptada de [20], ilustra os relacionamentos entre as especificações de serviços Web da 1ª geração.

Outras tecnologias para serviços Web chamadas de 2ª. geração não serão cobertas neste trabalho, como WS-Security, WS-Policy, WS-Reliability. Para maiores informações sobre essas especificações, acessar www.w3c.org.

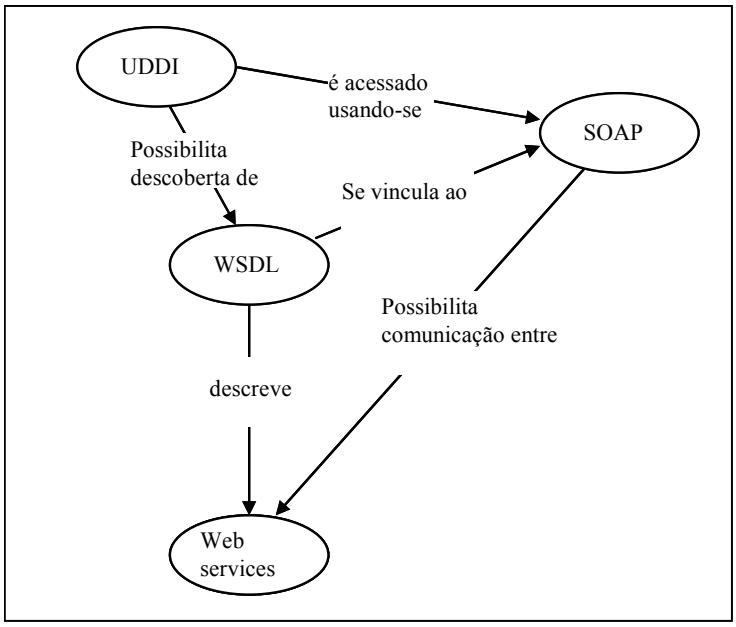

Figura 3.9 – Relacionamentos entre as especificações de serviços Web da 1ª. geração

a) SOAP (Simple Object Access Protocol)

O SOAP é um protocolo leve para troca de informações estruturadas em um ambiente descentralizado e distribuído [25]. As mensagens SOAP podem ser transportadas por uma variedade de protocolos de rede, como HTTP, SMTP, FTP, RMI/IIOP, ou protocolos proprietários de troca de mensagens, e são independentes de quaisquer modelos de programação.

O *framework* de troca de mensagens SOAP consiste nos seguintes elementos XML:

- *Envelope* – elemento raiz do documento XML que representa a mensagem SOAP.
- *Header* (cabeçalho) – elemento opcional, porém se estiver presente precisa ser o primeiro elemento. Contém informações de controle, como autenticação, identificação de intermediários, gerenciamento de transação, etc.
- *Body* (corpo) – elemento obrigatório, contém a carga da mensagem, ou seja, as informações a serem transportadas.
- *Fault* – elemento aninhado dentro do elemento *body*, traz informações de erros.

O SOAP estabelece dois estilos de mensagens: documento e RPC. O estilo documento indica que o corpo simplesmente contém um documento XML cujo formato está acordado tanto pelo remetente quanto pelo receptor. O estilo RPC indica que o corpo contém uma representação XML de um método de chamada. Nesse caso a infra-estrutura necessita de informações como:

URI (Uniform Resource Identifier), nome do método, nomes e valores de parâmetros, assinatura do método (opcional) e dados do *header* (opcional) [26]. Para saber mais sobre SOAP ver [25].

b) WSDL (Web Services Description Language)

A WSDL é uma linguagem para especificação e implementação de descrições de serviços. Foi criada com uma gramática XML específica, desenvolvida para informar metadados sobre um serviço Web de maneira uniforme para todos os clientes em potencial [27].

Um serviço Web pode possuir várias portas, sendo capaz de aceitar vários protocolos. Tais protocolos podem ser definidos com a WSDL. Esta fornece a definição de implementação do serviço, que é uma forma padronizada para descrições abstratas das operações e mensagens. Essas descrições são vinculadas a um formato de mensagem e protocolo de rede específicos para criar um *endpoint* que é a definição da implementação do serviço. Sendo assim, a definição abstrata de cada extremidade pode, então, ser vinculada a um protocolo de comunicação concreto [05]. Uma definição de *interface* do serviço pode ser instanciada e referenciada por várias implementações de serviço.

Em um documento WSDL, os serviços são definidos usando seis elementos principais [28] divididos entre a definição de *interface* do serviço e a definição da implementação do serviço. A definição da *interface* do serviço contém os elementos que formam a porção reusável da descrição:

- *Types* – fornece as definições dos tipos de dados usados para descrever as mensagens trocadas entre as portas.
- *Message* – representa uma definição abstrata dos dados que estão sendo transmitidos. Uma mensagem consiste em partes lógicas, e cada uma é associada a uma definição de tipo.
- *PortType* – conjunto de operações abstratas. Cada operação se refere a uma mensagem de entrada e mensagens de saída.
- *Binding* – especifica um protocolo concreto e especificações de formato de dados para as operações e mensagens definidas por um *portType* específico.

A definição da implementação do serviço contém os elementos que descrevem como uma *interface* particular é implementada por um dado provedor de serviço.

- *Endpoint* – associa um *endpoint* (por exemplo, um URL ou endereço de rede) a um elemento *WSDL:binding* de uma definição de *interface* de serviço.

- *Service* – usado para agregar um conjunto de portas relacionadas. Um serviço Web é modelado conforme esse elemento.

A definição da *interface* do serviço junto com a definição da implementação do serviço contêm informação suficiente para descrever para o requisitante do serviço como invocar e interagir com o serviço Web. Sendo assim, desde que o remetente e o receptor tenham a mesma descrição do serviço (isto é, o arquivo WSDL), as implementações por trás dos serviços Web podem ser qualquer coisa. Para maiores informações sobre WSDL ver [28].

c) UDDI (Universal Description, Discovery, and Integration)

Atualmente o principal padrão para registros de serviços é o UDDI, definido pela organização UDDI, que faz parte do OASIS. O UDDI pretende agir como um agente de informação entre consumidores e provedores de serviços. Ele especifica a maneira de armazenar e recuperar informações sobre serviços, especialmente o nome do provedor e as *interfaces* técnicas[12].

Vimos que a definição WSDL de um serviço informa ao requisitante como invocar e interagir com o serviço. Contudo, o requisitante do serviço poderia precisar de outras informações sobre o *endpoint* do provedor, tais como requisitos de qualidade de serviço, ou taxonomias associadas a um produto. Essas informações não constam das descrições na WSDL, mas podem ser recuperadas do UDDI que também fornece um mecanismo para armazenar descrições dos serviços Web. Portanto, o UDDI não é somente um mecanismo de diretório, ele também define um padrão de estrutura de dados para representar informações de descrição de serviços em XML. Há quatro estruturas de dados fundamentais em uma entrada de UDDI [23]:

- *BusinessEntity* – modela informações da empresa, como nome, informação de contato, tipo de negócio, informações de identificação e os elementos opcionais para categorização. Contém uma coleção de elementos *BusinessServices*, um para cada serviço Web que a empresa deseja publicar.
- *BusinessService* – contém informações do serviço como o nome do serviço, o identificador e os elementos opcionais para categorização. Cada *BusinessService* contém uma coleção de elementos *BindingTemplate*.

- *BindingTemplate* – descreve informações de acesso (por exemplo, um URL) ou opções de roteamento. Esses elementos também contêm ponteiros para especificações técnicas. Uma especificação técnica é modelada com um Tmodel (modelo técnico).
- *Tmodel* – descreve diferentes conceitos da *interface*, como um tipo de serviço, uma plataforma tecnológica ou uma taxonomia. O *Tmodel* é uma parte importante para a estratégia de busca do UDDI fornecendo metadados sobre especificações adicionais, seu nome, publicador e o URL do seu esquema. Diferentes serviços Web que processarem as mesmas mensagens podem ter o mesmo Tmodel.

A figura 3.10, adaptada de [12], ilustra o mapeamento entre as entradas do UDDI e de um documento WSDL.

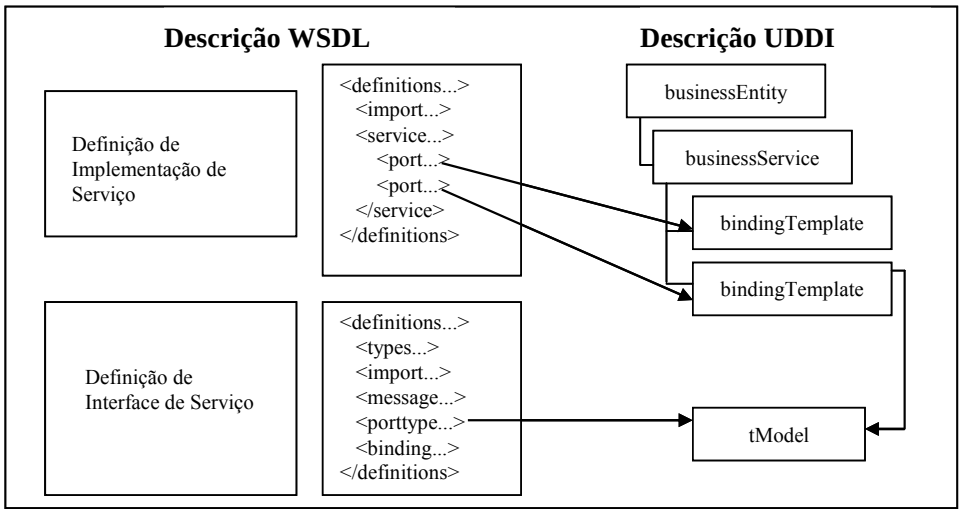

Figura 3.10 – Mapeamento entre UDDI e descrição WSDL

UDDI públicos

Os UDDI públicos são diretórios globais. Instâncias desses diretórios podem residir em grandes corporações (também chamadas de "nós operadores"). Os "nós operadores" suportam um conjunto comum de APIs que vão assegurar a integridade e disponibilidade da informação fornecida. O registro só precisa ser feito em um único operador, pois os dados são replicados regularmente para todos os outros operadores. Quando um registro é efetuado em um operador, esse operador fica sendo o chamado "Custodian". Existem empresas que executam serviços de registro em nome de outras empresas. Essas empresas são conhecidas por "Registrar" e auxiliam outras a compor suas informações de registro, como a lista de serviços e os modelos técnicos que descrevem tais serviços.

UDDI privados

As implementações particulares de registros UDDI seguem a especificação UDDI e residem em intranets, extranets ou redes privadas na Internet. Entre os vários tipos de UDDI privado podemos ter [23]:

1) Nó UDDI interno às aplicações da empresa – geralmente usado dentro dos limites de uma empresa para fins de integração de aplicações. O escopo pode ser aplicação, departamento ou corporação. Esse tipo de UDDI fica nos limites internos de *firewalls* e permite que publicadores de serviços tenham mais controle sobre seus registros de serviços e sobre quesitos, como acessibilidade, disponibilidade e desempenho.
2) Portal – utilizado por empresas para integração com parceiros externos. Pode residir tanto fora como entre os limites de *firewalls*. Permite que as empresas retenham controle sobre suas descrições de serviços, do acesso ao nó UDDI e de qualidade de serviços.
3) Catálogo de parceiros – é usado por uma empresa para manter descrições de parceiros de negócios. Essas descrições residem nos limites internos de um *firewall* e já foram aprovadas, testadas e validadas.
4) *E-marketplaces* – geralmente utilizado por um provedor de serviços que pretende competir por requisitantes de serviços com outros provedores de serviços Web. Esses nós residem em uma organização ou consórcio de indústria e contêm descrições de serviço de uma indústria, ou segmento em particular. Geralmente fornecem algum tipo de filtro para evitar entradas ilegítimas e podem garantir alguns quesitos de qualidades de serviços.

Os tipos de UDDI privado mais significativos dentro da abordagem deste trabalho são os privados de tipos 1, 2 e 3. Para saber mais sobre UDDI e sobre soluções que ajudem a implementar um registro UDDI privado, sugere-se a leitura de [56].

3.3.4 Qualidade de serviço

O fornecimento de qualidade de serviço para serviços Web é um grande desafio, em razão de natureza dinâmica dos serviços. Entre os principais requisitos de qualidade de serviço estão incluídos: disponibilidade, acessibilidade, integridade, interoperabilidade, desempenho e segurança. Este último requisito será tratado na próxima seção. Os demais são apresentados nessa seção, segundo abordagens tratadas em [35, 36 e 40]:

Disponibilidade é o aspecto da qualidade que diz se o serviço está pronto para uso imediato. O TTR (*Time-to-Repair*) está associado à disponibilidade e representa o tempo que leva para que

o serviço seja disponibilizado novamente após uma falha. No caso de serviços Web, fica muito caro construir sistemas tolerantes a falhas com alta disponibilidade pelo fato de esse tipo de ambiente ser diversificado, dinâmico e distribuído.

Acessibilidade é a qualidade que diz se o serviço Web é capaz ou não de atender às requisições dos clientes. Para aumentar a acessibilidade podemos construir sistemas altamente escaláveis, ou seja, sistemas com habilidade de responder às requisições consistentemente, não importando as variações no volume delas. Boas práticas para melhorar o fornecimento de acessibilidade é o *Service pooling* e o balanceamento de cargas.

Integridade é o aspecto da qualidade de serviço que diz respeito a como os serviços Web mantêm a exatidão das informações e a execução apropriada de transações. A abordagem mais popular para prover integridade de transações em ambiente distribuído é o *two-phase commit*. As transações envolvendo serviços Web adicionam uma dificuldade a mais para essa abordagem, pois elas podem ter uma duração muito longa, talvez até de dias, envolvendo organizações autônomas, heterogêneas e geograficamente distantes.

Nesse caso, uma boa prática é a adoção de padrões como WS-Coordination[53], WS-Transaction[54] e o BTP (Business Transaction Protocol) [50]. O OASIS Business Transactions TC, comitê técnico do OASIS, liberou o BTP, que usa um protocolo de coordenação em duas fases, para assegurar que a aplicação como um todo alcance um resultado consistente [50]. No entanto, o BTP melhorou a aplicabilidade das transações distribuídas fracamente acopladas com o necessário enfraquecimento das garantias transacionais "tradicionais" (por exemplo, as propriedades ACID) [51]. Ele permite dois tipos de transações de negócios: a) transações atômicas (*atomic business transactions*) e b) transações coesivas (*cohesive business transactions*) [50].

As transações atômicas BTP são similares às transações em sistemas fortemente acoplados. Nesse caso, o trabalho da transação como um todo é confirmado ou cancelado; no entanto, a propriedade de isolamento é relaxada. Nas transações de negócio coesivas, além do relaxamento da propriedade de isolamento, o protocolo permite ainda a) a intervenção da aplicação na seleção do que pode ser confirmado [50] e b) a determinação da consistência através de acordo e interação entre o coordenador (*coordinator*) e o cliente (*iniciator*) [51].

Participantes de uma transação BTP podem reverter os efeitos de uma transação atômica. Para isso, podem usar abordagens como registro de imagens antes/depois ou compensação [50].

Interoperabilidade em serviços Web significa que os programadores não tenham de se preocupar com qual linguagem de programação ou em que sistema operacional eles serão usados. A maioria dos serviços Web é definida sob padrões que, infelizmente, demoram um pouco para ser implementados. Devido à natureza dinâmica do mercado, os fornecedores acabam por implementar parcialmente essas especificações nos seus produtos, muitas vezes com resultados insuficientes em termos de interoperabilidade. O ponto-chave para interoperabilidade em serviços Web é a habilidade de um conjunto de serviços consumir documentos WSDL gerados por outros.

O **desempenho** de um serviço Web é medido em termos de *throughput*, latência, tempo de execução e tempo de transação. *Throughput* representa o número de requisições atendidas em um dado período. Latência é o RTT (*round trip time*) entre o envio de uma requisição e o recebimento de sua resposta. Tempo de execução é aquele que o serviço Web levou para processar uma seqüência de atividades. O tempo de transação é o período que o serviço Web levou para completar uma transação por inteiro. O desempenho global depende da lógica da aplicação, do estado da rede e de protocolos de transporte e troca de mensagens, tais como SOAP e HTTP.

O protocolo SOAP usa um processo envolvendo vários passos para completar um ciclo de comunicação. Uma requisição SOAP começa com a lógica do negócio na aplicação descobrindo o método e os parâmetros para chamar um documento WSDL. O processo todo consome tempo, que requer vários níveis de análise e validação do XML, podendo comprometer o desempenho do serviço Web. Algumas maneiras de melhorar o desempenho em serviços Web envolvem:

- **Uso de serviços de mensagens assíncronas** – produtos como o JMS (Java Messaging Service) ou o MQSeries para serviços Web da IBM podem ser usados internamente em uma empresa. Eles são confiáveis e flexíveis para troca assíncrona de dados em uma corporação.
- **Uso de wans/extranets e redes de serviços Web privadas** – pode ser uma boa opção quando o aspecto do negócio é crítico. Essas redes privadas fornecem baixa latência, baixa congestão e entrega garantida, além de *non-repudiation*.
- **Uso de SOAP** – o desempenho do SOAP é degradado devido à extração do envelope do pacote SOAP. Também há o tempo de análise do conteúdo da informação XML dentro do envelope SOAP usando o analisador XML. O papel do analisador XML no desempenho do SOAP é crítico devido ao tamanho do código, tempo de processamento e memória

necessária para suportar uma quantidade grande de atribuições, como verificação e conversão de tipo, verificação de formato, ou resolução de ambigüidades. As implementações podem ser baseadas em analisadores DOM (Document Object Model), que armazenam a estrutura completa do documento na memória. Uma outra possibilidade é usar o SOAP baseado em SAX (Simple API for XML), que não armazena o documento XML inteiro na memória.

- **Compressão do XML** – a representação dos dados em XML ocupa mais espaço do que os mesmos dados na forma binária, aumentando o tempo de transmissão de dados. Algumas aplicações devem considerar a possibilidade de compressão do XML, principalmente se o *overhead* de CPU requerido pela compressão for menor que a latência da rede.

3.3.5 Segurança

Esta seção apresenta uma visão geral da arquitetura, administração e estratégia de segurança como são abordados em [21, 23, 36, 41 e 48]. Não é o foco deste trabalho se aprofundar nesse assunto.

Para tornar os serviços Web seguros, alguns mecanismos são necessários para solucionar principalmente problemas relacionados à autenticação, controle de acesso baseado em papéis, aplicação de políticas de segurança distribuídas e segurança da camada de mensagem onde ficam os intermediários.

Implementações de serviços Web requerem mecanismos de segurança ponto-a-ponto e/ou fim-a-fim, dependendo do grau de ameaça e risco. No entanto, mecanismos de segurança tradicionais, geralmente orientados à conexão ponto-a-ponto, podem não suprir os requisitos de segurança fim-a-fim dos serviços Web pois eles requerem maiores granularidades. Em geral, os serviços Web usam uma abordagem baseada em mensagens que habilitam interações complexas podendo incluir o roteamento de mensagens através de vários domínios [21].

O quesito segurança é preocupante para empresas, fornecedores, organizações e consórcios de padronização. A Microsoft Corporation, a IBM, a VeriSign Inc. e outras empresas anunciaram a publicação de uma especificação de segurança, a Web Services Security (WSS), que fornece funções de segurança, como integridade e confidencialidade em mensagens, visando a implementação de um melhor nível de aplicações de serviços Web [49]. A especificação WSS

define um *framework* de segurança fim-a-fim que também fornece suporte para o processamento de segurança intermediária [21]. Maiores detalhes em [49].

Além dos problemas tradicionais relativos à segurança, um ambiente que englobe serviços Web precisa também prover segurança para as mensagens que estejam em trânsito (com ou sem a presença de intermediários), bem como a segurança dos dados armazenados. Do ponto de vista de quem administra, as características desse tipo de arquitetura de segurança, além das já conhecidas, enfoca questões como política de segurança, monitoramento e administração de acesso entre ambientes com múltiplos sistemas heterogêneos. Outros pontos incluem parceiros fracos no quesito segurança e liberação de uma aplicação legada que não tenha sido projetada para ser exposta ao público via Web.

Cada *interface* de aplicação de serviço Web pode ter centenas de operações que podem ser acessadas expondo informações sigilosas para intrusos mal-intencionados. Todos os aspectos de serviços Web, inclusive roteamento, gerenciamento, publicação e descoberta, devem ser executados de uma maneira segura [23]. Na realidade, a maior parte dos problemas já conhecidos com o tráfego da Web também são concernentes aos serviços Web, porém é preciso prestar maior atenção a certos detalhes, tais como autenticação, automação, integridade, *sign on* e assinaturas. Tais detalhes são discutidos a seguir.

Autenticação – Como no tráfego normal da Web, os requisitantes de serviço precisam ser autenticados pelo provedor do serviço antes que a informação seja enviada. Tecnologias padrão da Web como *passwords*, certificações, Kerberos, LDAP (Lightweight Directory Access Protocol) e Active Directory podem ser usadas para autenticar os requisitantes. Todavia, quem requisita o serviço também precisa autenticar o provedor do serviço. Informações confidenciais podem estar sendo enviadas não somente na resposta, mas também na requisição. Além disso, um arquivo WSDL pode ser falsificado fazendo com que o requisitante se comunique com um serviço Web mal-intencionado. Portanto, arquivos WSDL também precisam ser autenticados para assegurar que seus dados sejam íntegros.

Autorização – É crítica por causa dos novos níveis de acesso disponibilizados com os serviços Web. Como serviços são *interfaces* programadas, são difíceis de ser monitorados quanto à atividade suspeita. Serviços Web permitem uma integração que pode envolver muitos parceiros, inclusive competidores entre si; por isso, direitos de acesso precisam ser muito bem controlados e atualizados.

Integridade e confidencialidade dos dados – É preciso assegurar que as informações contidas nas mensagens cheguem íntegras ao seu destino, ou seja, que não tenham sido deliberada ou acidentalmente modificadas. Além disso, o conteúdo das mensagens não pode ter sido disponibilizado ou acessado por programas ou pessoas não autorizadas. Técnicas de criptografia e assinatura digital podem ser usadas para esse propósito [21].

***Sign-on* único** – Diversos sistemas precisam se comunicar entre si e fica difícil para cada sistema manter listas de controle de acesso uns dos outros. Uma solução é dar a todos a mesma chave de acesso; todavia, isso representa um problema sério quando é preciso eliminar um dos membros. Novas senhas precisam ser enviadas para todos os membros remanescentes. O padrão SAML (Security Access Markup Language) é um *framework* XML para troca de informações de autenticação e autorização. Maiores detalhes podem ser adquiridos em [57].

Assinaturas e *Non-repudiation* – Um problema importante é assegurar que a mensagem recebida é válida e não foi falsificada. Assinaturas digitais podem ser usadas para certificar documentos. Um requisitante de serviços pode certificar um documento com a senha privada de quem o enviou e enviá-la junto com os dados da mensagem. O provedor do serviço pode então verificar a assinatura com a senha pública para ver se alguma porção do documento foi comprometida. O padrão "XML Signature" fornece meios para certificar partes de documentos XML, fornecendo integridade fim-a-fim através de sistemas múltiplos. A especificação do "XML Signature" pode ser encontrada em [55].

Outro ponto importante é o de *non-repudiation* para garantir que quem enviou a mensagem não poderá mais tarde negar tê-la enviado. Quando as transações são executadas, geralmente é bom ter um *log* provando que uma ação em particular foi executada. Com assinaturas, os provedores de serviço conseguem registrar as transações em *logs* de auditoria certificados. Uma vez que um *log* de auditoria foi certificado, não poderá mais ser modificado sem a alteração da assinatura, evitando que *hackers* modifiquem esses *logs* para despistar seus passos. Quando há necessidade de *non-repudiation* para uma terceira parte envolvida, recibos digitais de que certas transações ocorreram podem fornecer uma verificação independente.

Proteção contra ataques – XML trafega pelas portas 80 (HTTP) e 443 (HTTPS) abertas para o tráfego normal da Web. Os arquivos WSDL e as entradas do UDDI podem ser usados por código mal-intencionado. Além disso, o formato do XML, que geralmente é autodescritivo, mostra claramente que tipo de conteúdo carrega. Portanto, mecanismos e políticas para

publicação e descoberta seguras devem ser aplicados [21]. Mesmo dentro dos limites internos de uma empresa, é preciso cuidado com a publicação e descoberta seguras. Por exemplo, programadores de aplicações SOA de um departamento podem não ter permissão para descobrir serviços disponíveis para outro departamento. Para isso, é bom que sejam usadas políticas de identificação tanto no momento da publicação quanto no de descoberta dos serviços.

3.3.6 Gerenciamento

Esta seção apresenta alguns quesitos esperados para uma arquitetura de gerenciamento de serviços Web mais significativos no contexto deste trabalho, conforme abordagem em [37, 38 e 39].

O gerenciamento de serviços Web engloba uma série de facilidades de gerenciamento que possibilita monitoração, controle e relatórios tanto da qualidade de serviço como da utilização desses serviços (freqüência, duração, escopo, autorização de acesso). O gerenciamento de serviços Web requer uma semântica comum para gerenciamento das políticas e condições de gerenciamento, e para serem compreendidos pelos consumidores e provedores de serviço [21]. Questões como requisições, dinamismo, alerta na ocorrência de problemas e problemas gerais referentes ao gerenciamento de serviços Web são muito importantes.

O gerenciamento envolve questões de controle de acesso, logging, monitoração e auditoria. O log deve registrar requisições de serviços Web, respostas, informações relativas às sessões dos clientes, e outros eventos. Esses *logs* podem ser usados para várias outras operações, tais como provisionamento de serviços, monitoramento, avaliação de desempenho, etc. Monitoramento e auditoria são complementares e essenciais ao bom gerenciamento de serviços. Permitem igualmente gerenciamento de desempenho como *throughput*, uso e estatísticas de tempo de respostas para a plataforma e para os serviços Web individuais, incluindo trilhas de auditoria sobre o uso dos serviços.

Ainda outros fatores relativos ao gerenciamento incluem provisionamento de serviços, gerenciamento de aplicativos, mediação e faturamento. O provisionamento de serviços é responsável pela definição do comportamento do sistema em termos das funcionalidades e técnicas dos negócios. Por exemplo, os clientes dos serviços podem assiná-los, e eles serão autenticados e autorizados de acordo com suas contas de assinaturas. Clientes têm acesso aos serviços de acordo com os contratos que firmaram no momento da assinatura dos serviços. Um sistema de medição deve coletar as estatísticas de uso, pois através delas o sistema de

faturamento poderá enviar a conta para o cliente. Nesse caso, é desejável que o gerenciamento de contrato e a aplicação de medição sejam tratados como entradas da aplicação de faturamento. Contratos são definidos para a ligação entre os clientes dos serviços Web e os termos de prestação de serviços (que incluem faturamento e pacotes de avaliação). Esse contrato é associado a cada assinatura dos clientes. Quando os clientes usarem os serviços Web, os sistemas de autenticação, autorização e identificação verificam o contrato associado, permitindo acesso apenas no caso de contratos válidos.

Ainda outros fatores a considerar incluem o gerenciamento de políticas e o gerenciamento de eventos. No primeiro caso, deve haver integração entre SLA (Service Level Agreement) e sistemas de gerenciamento de políticas. Nesse caso, o gerenciamento de dados é usado para derivar os valores reais do SLA e dos parâmetros de política que podem ser usados para compará-los com o que foi acordado bem como as políticas pré-definidas, permitindo o monitoramento/garantia das políticas e dos SLAs. O gerenciamento de eventos considera eventos relacionados a regras de negócio.

Além dessas funcionalidades, é desejável que uma arquitetura de gerenciamento de serviços Web possa também fornecer acesso remoto. O gerenciamento deve poder ser feito inclusive pela Internet, trafegando através de protocolo HTTP. Além disso, as informações de gerenciamento devem poder ser processadas por outra aplicação usando APIs abertas ou manipuladores de dados. Isso possibilita ligar componentes independentes para executar as ações apropriadas, baseadas no estado geral dos ambientes onde residem os serviços Web.

O OASIS tem um comitê técnico com o propósito de definir o gerenciamento de serviços Web, o WSDM (Web Services Distributed Management), incluindo o uso da arquitetura e tecnologia de serviços Web para gerenciar recursos distribuídos. Esse comitê também pretende desenvolver o modelo de um serviço Web como um recurso gerenciável.

3.4 Conclusão

A arquitetura orientada a serviços foi concebida para fornecer interoperabilidade entre aplicações distribuídas. A abordagem dessa arquitetura permite definir aplicações corporativas como serviços que interagem por entre diferentes plataformas de *hardware*, *software* e sistemas operacionais. Da mesma forma, possibilita também a integração de sistemas remotos entre diferentes corporações, através da Web. Assim, promove a integração de aplicações e dados internos à empresa e também externos a ela, provenientes de parceiros, clientes e fornecedores.

Atualmente, uma das formas mais populares de implementação dos conceitos e técnicas da SOA são os serviços Web, cujas especificações iniciais, SOAP, WSDL e UDDI, fornecem mecanismos abertos baseados em XML para interoperabilidade, descrição e descoberta de serviços.

Como as implementações baseadas em SOA envolvem interação em um ambiente distribuído e fracamente acoplado, é importante que haja também um modelo padronizado e integrado para fornecimento de segurança, gerenciamento e interoperabilidade entre outros quesitos de qualidade de serviço. Nesse caso, é de fundamental importância o trabalho que importantes consórcios tais como W3C, OASIS e WS-I estão desenvolvendo para a definição de padrões de segurança, interoperabilidade, suporte a *multi-devices,* troca de mensagens, roteamento e toda infra-estrutura necessária para suportar implementações de uma SOA.

No momento, as tecnologias para serviços Web ainda estão em fase de evolução, e muitas ainda não estão maduras, principalmente no que se refere à interoperabilidade entre diferentes empresas através da Internet. No entanto, os primeiros padrões, chamados de 1ª. geração, como SOAP, WSDL e UDDI, abordados neste capítulo, já são amplamente utilizados dentro do contexto interno da empresa ou na sua integração com parceiros confiáveis.

O próximo capítulo descreve a arquitetura proposta pelo mestrado e seu mapeamento parcial para uma solução em serviços Web.

Capítulo 4

ARQUITETURA PROPOSTA

Este capítulo tem o objetivo de apresentar a arquitetura para acesso aos dados através de uma federação de domínios proposta por este trabalho, de forma a endereçar muitos dos problemas mencionados no capítulo 1.

A seção 4.1 introduz o contexto deste capítulo. Em seguida, a seção 4.2 apresenta os objetivos da arquitetura. As seções 4.3 a 4.6 abordam a federação de domínios: como ela foi idealizada (seção 4.3), seus níveis e esquemas (seção 4.4), como ela permite a interoperabilidade entre os domínios (seção 4.5), sua administração (seção 4.6) e sua configuração (seção 4.7). As seções 4.8 a 4.11 apresentam a arquitetura de integração de dados: sua visão geral (seção 4.8), o detalhamento de sua camada mediadora integradora de dados (seção 4.9), como ela é implementada utilizando serviços web (seção 4.10) e como as aplicações legadas podem ser migradas (seção 4.11). A seção 4.12 conclui o capítulo.

4.1 Introdução

Os executivos de TI geralmente enfrentam desafios como conciliar corte de custos e maximização da utilização da tecnologia já existente. Ao mesmo tempo, são obrigados a oferecer serviços melhores e mais eficientes para atender às estratégias de negócios da empresa. Para poderem atingir suas metas, eles precisam equacionar vários problemas comuns à maioria das empresas. Um desses problemas é o imposto pelos vários graus de heterogeneidade no ambiente tecnológico. A integração de processos, aplicativos e dados, que não é uma tarefa fácil, constitui um desafio ainda maior se esses estiverem distribuídos em produtos de diferentes fornecedores através de múltiplas plataformas e concebidos em diferentes períodos no tempo.

A integração de dados bem realizada é de vital importância. A informação é um dos bens mais preciosos que uma empresa pode ter e sua imprecisão em uma base tem um alto potencial de propagação, podendo rapidamente afetar diversos setores da organização. Portanto, uma empresa que tenha seus dados distribuídos, em ambientes heterogêneos ou não, tem de estar sempre preocupada com a administração, a integração e principalmente o controle desses dados.

Como os negócios precisam se adaptar rapidamente às mudanças para sobreviver em um ambiente tão dinâmico e competitivo, a infra-estrutura tecnológica de uma empresa também precisa disponibilizar um ambiente favorável a essas mudanças. As organizações evoluíram de divisões verticais e isoladas, como eram antes dos anos 80, passando por uma estrutura horizontal, focada em processos de negócios, entre os anos 80 e 90, para finalmente, nos dias atuais, chegar a um novo paradigma de ecossistema de negócios. As figuras 4.1 e 4.2, adaptadas de [12], mostram respectivamente a evolução estrutural das empresas e as respostas de TI para acompanhar essas mudanças, evoluindo de ambientes monolíticos até arquiteturas de serviços distribuídos.

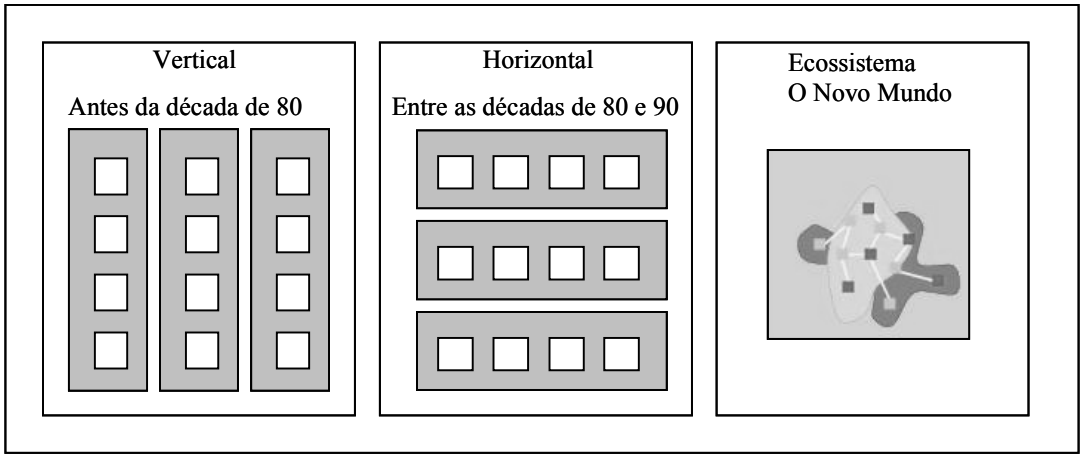

Figura 4.1 – Evolução estrutural das empresas

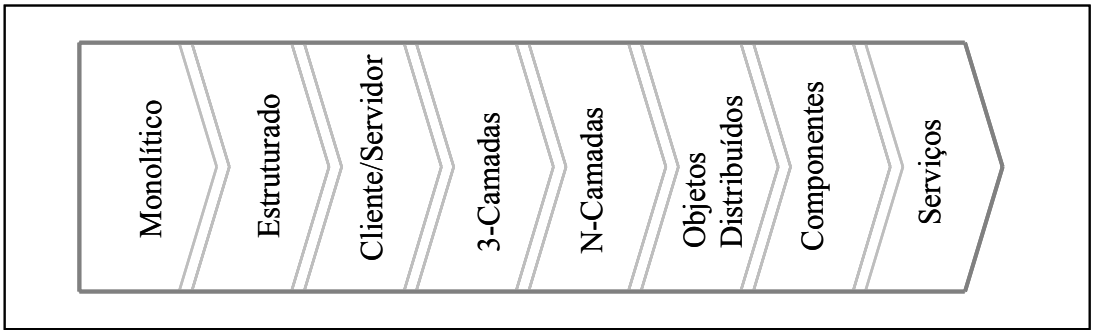

Figura 4.2 – Evolução de ambientes

Políticas organizacionais também têm um importante peso no processo de integração de dados dentro de uma empresa. Isso se torna ainda mais evidente quando empresas participam de fusões ou são adquiridas por outras empresas. Unidades de negócios que outrora eram independentes precisam ser agregadas e compartilhar ou integrar dados e processos com outras

unidades. Esse tipo de situação pode gerar conflitos de interesse. A necessidade dos responsáveis pelo gerenciamento das informações de ter o controle do processo de compartilhamento para saber quanto, quando e de que forma essas informações serão disponibilizadas para a corporação torna-se um elemento-chave para o processo de integração entre diferentes unidades de negócios.

Muitas vezes, durante o compartilhamento de dados e processos entre sistemas que não foram inicialmente concebidos para cooperarem entre si, pode surgir sobreposição de conceitos divergentes sobre os mesmos dados, com diferentes necessidades. Essa situação precisa ser levada em consideração e tratada com extremo cuidado.

Para contornar os problemas de heterogeneidade, interoperabilidade e requisitos de negócios em constante mudança, é necessária uma arquitetura para a construção de aplicações baseada em uma plataforma que seja independente de protocolo, com componentes fracamente acoplados e que forneça transparência de localização dos dados corporativos. Para se manter uma política interna livre de problemas em relação ao controle dos dados dentro de uma empresa, é preciso que a visão corporativa sobre esses dados seja baseada na cooperação entre os diversos grupos organizacionais, ao mesmo tempo em que cada interesse envolvido seja preservado.

4.2 Objetivos da Arquitetura Proposta

A proposta deste trabalho é sugerir uma arquitetura para a integração de dados corporativos com os seguintes objetivos:

1) Administrar os dados corporativos de uma forma global e unificada. Geralmente esses dados estão formatados de diversas maneiras, distribuídos em diferentes meios de armazenamento por plataformas e ambientes diversificados.
2) Prover meios através dos quais o responsável pela informação possa controlar os acessos externos ao seu domínio. Entre outros, esses controles possibilitam:
    - evitar queda de desempenho forçando o acesso externo através de serviços padronizados e específicos;
    - auditar a utilização dos dados por usuários e aplicações;
    - dividir os custos da utilização e manutenção dos dados por todos os domínios que acessam os dados;
    - ter serviços padronizados e homologados, compartilhando processos unificados de controle de acesso, administração e gerenciamento dos dados.

3) Permitir o acesso aos dados distribuídos pela corporação através do modelo de especificação desses dados, e não somente o de implementação. Dessa maneira, possibilita:
    - Certo grau de independência lógica de dados; algumas mudanças na estrutura lógica dos dados ficam transparentes para as aplicações.
    - Independência física dos dados através de serviços de acesso aos dados que deixam transparente para a aplicação a real localização desses dados, além de abstrair as mudanças na estrutura física.
4) Fornecer um sistema de integração que possibilite transparência de localização dos dados, que seja flexível na sua forma de implementação e que forneça qualidade no acesso aos dados.
5) Prover facilidade para inclusão de novas fontes de dados residentes em plataformas diversificadas.
6) Estender a vida útil dos dados armazenados em sistemas legados possibilitando a sua integração com sistemas mais novos.

Para prover os aspectos mencionados no item 1, a arquitetura prevê um repositório de metadados que abriga informações referentes aos dados corporativos e aos serviços que os disponibilizam.

Este trabalho propõe um sistema que represente os diferentes grupos corporativos, os quais foram aqui chamados de *domínios*, organizando-os em uma federação. Uma característica primordial de uma federação é a cooperação entre sistemas independentes, respeitando a autonomia e a heterogeneidade de seus integrantes. Uma federação também possui uma política organizacional com regras definindo deveres e direitos que precisam ser respeitados por todos os seus integrantes.

O objetivo da Federação de domínios é ter procedimentos e regras consistentes dentro de uma organização para assegurar que o fluxo de informações entre os diversos domínios participantes seja controlado de forma que o resultado seja a disponibilidade, segurança e integridade das informações corporativas, ao mesmo tempo em que preserva a autonomia de cada domínio sobre seus dados e operações a eles relacionadas. Dessa maneira, esta arquitetura endereça o objetivo mencionado no item 2.

Para fornecer independência de dados, objetivo apontado no item 3, a arquitetura proposta baseou-se nos princípios da arquitetura de referência *ANSI/SPARC*. Essa arquitetura de referência foi estendida neste trabalho para suportar as dimensões da Federação de domínios – distribuição, autonomia, heterogeneidade e interações SOA (publicação, descoberta e ligação de serviços).

Os objetivos definidos nos itens 4 a 6 foram alcançados disponibilizando-se a visão federada para as aplicações, através de uma camada mediadora de integração de dados (CMID), cuja arquitetura atende aos requisitos de uma arquitetura orientada a serviços (SOA).

A arquitetura proposta prevê uma abordagem flexível em relação à sua forma de implementação. Através da implementação da CMID com base nos conceitos de orientação a serviços utilizando serviços web, consumidores e provedores dessa camada poderão interagir independentemente de plataforma, protocolo ou linguagem de programação que estiverem utilizando para acesso às informações dos diversos domínios. Os serviços poderão interagir com outros serviços e com consumidores de serviço independentemente de linguagem de programação, protocolo, ou plataforma. Essa flexibilidade é habilitada pelo uso de padrões abertos como XML, SOAP e HTTP.

4.3 Federação de domínios

Dentro do contexto deste trabalho, domínios são uma abstração, uma maneira lógica de organizar conjuntos de dados com forte relacionamento entre si. Eles podem ser internos ou externos à empresa e refletir organizações parceiras que mantêm algum fluxo de informações que precise ser incorporado à organização. Cada organização pode ter seu próprio critério para definir seus domínios. Eles podem ser definidos, por exemplo, por processos de negócio, sistemas ou departamentos.

Um domínio tem responsabilidades e direitos sobre as informações que gera e administra. Representado pelo seu administrador, um domínio é responsável por definir regras de acesso e manipulação dessas informações e tem o direito de saber quem as acessa essas informações e como elas são manipuladas.

Cada conjunto de informações é de responsabilidade de um domínio e pode ser agrupado logicamente em uma ou mais entidades de negócio, como, por exemplo, Produto ou Pedido. Essas entidades podem ter subconjuntos de informações que são associadas a elas espalhadas por diversos domínios. Por exemplo, a entidade Pessoa pode ter fragmentos de informações sob responsabilidade do domínio Vendas (informações sobre as pessoas que adquiriram produtos da

empresa), outro fragmento sob o domínio de Marketing (informações sobre as pessoas que podem vir a adquirir um produto). Muitas vezes esses subconjuntos podem ter uma sobreposição com informações em comum. A solução que a Federação de domínios oferece a esse tipo de situação é tratada na seção 4.7.

As informações são disponibilizadas através de serviços de acesso a dados. Um serviço de acesso a dados é um recurso abstrato que simboliza uma habilidade para a execução de uma tarefa, devendo representar a mesma funcionalidade para toda a Federação. As informações e os serviços são relacionados através das mensagens de entrada e saída trocadas pelos serviços.

Os serviços de acesso aos dados podem ser de dois tipos: específico e genérico. Nos serviços específicos, as mensagens de entrada e saída relacionam-se às informações de forma estática. Esse tipo de serviço impõe maior autonomia ao controle dos domínios responsáveis pelas informações. Por exemplo: um certo serviço disponibiliza os dados básicos do cliente e tem como mensagem de entrada o "cpf do cliente" e como mensagens de saída, o "nome do cliente", "data de nascimento do cliente" e "local de nascimento do cliente". Esse serviço pode ser implementado por um aplicativo que faça, por exemplo, uma chamada a um procedimento armazenado no SQL Server que acesse os dados do cliente através da coluna cpf_cli e retorne entre outros parâmetros, nm_cli, dt_nasccli e nm_locnasccli desse cliente acessando a tabela "Dados_cliente". Note que esse mesmo serviço pode também ser implementado por um outro aplicativo que tenha os mesmos parâmetros de entrada e saída, mas que acesse esses dados diretamente usando o DB2 no mainframe, por exemplo.

Já nos serviços de tipo genérico, as mensagens de entrada e saída relacionam-se às informações de forma dinâmica, sendo indicados para acessos dentro do próprio domínio, ou entre domínios que sejam parceiros confiáveis. Nesse tipo de serviço, a mensagem de entrada inclui um *string* com o comando a ser executado. Seguindo o exemplo anterior, o comando teria o nome, a data e o local de nascimento do cliente selecionado a partir de uma determinada entidade. Sendo assim:

- O comando pode usar diretamente a forma como os dados e entidades foram especificados, referenciando os nomes únicos conhecidos pela federação ("nome do cliente", "cpf do cliente", etc.).
- A mensagem pode referenciar os dados e entidades em sua estrutura de armazenamento. Sendo assim, o comando estaria em uma linguagem entendida pela fonte de dados que

será acessada e os dados e entidades teriam os nomes conhecidos nesse local. Na realidade, para esse tipo de comando ser aceito, o domínio necessita ter grau 2 de visualização, como detalhado na seção 4.5.

Para prover autonomia de visualização aos domínios, a federação permite que os domínios criem restrições de visualização, que podem ser funcionais ou não-funcionais. Essas restrições são definidas por cada domínio para descrever sua visão particular sobre a visão corporativa (maiores detalhes serão fornecidos na seção 4.5).

As informações e os serviços podem ser implementados de diversas maneiras pelos domínios. Um serviço pode ser implementado por um ou mais aplicativos. Um aplicativo pode conter acesso aos dados de várias maneiras, por exemplo, através de uma visão ou uma *stored procedure* em um sistema gerenciador de banco de dados (SGBD).

Os dados são implementações das informações e estão agrupados e armazenados sob algum formato em algum tipo de estrutura de armazenamento. Estas geralmente são mantidas ou têm seu acesso gerenciado por fontes de dados.

Na abordagem deste trabalho, as fontes de dados podem ser diferentes recursos de computação, tais como, SGBDs, sistemas gerenciadores de arquivos ou uma aplicação. Uma fonte de dados pode permitir acesso a dados de diferentes domínios. Da mesma maneira, um domínio pode conter dados distribuídos em diversos tipos de fontes de dados.

Os domínios protegem suas informações através de uma política de segurança sustentada pela Federação. Esta deve cuidar para que nenhuma informação ou serviço seja visível para domínios não-autorizados.

A Federação, representada por seu administrador, estabelece uma política gerencial para a sua utilização, que deve ser seguida por todos os seus integrantes. Essa política gerencial visa garantir a qualidade dos acessos aos dados mantidos pelas diversas fontes de dados, usando dados gerenciais colhidos dos domínios.

A Federação de Domínios centraliza e armazena informações em seu repositório de metadados que é global à Federação e não pertence a nenhum domínio específico. O repositório de metadados armazena informações específicas para prover suporte para os programadores de aplicação em tempo de desenvolvimento e para as aplicações utilizarem em tempo de execução. O repositório também armazena informações específicas para a configuração e administração da Federação. O repositório de metadados armazena os seguintes tipos de informações:

- De especificação dos dados e serviços dos domínios integrantes da federação. São basicamente as descrições de dados e serviços, sua *interface*, as mensagens de entrada e saída, padrões de troca de mensagens que estão envolvidas na interação com o serviço, juntamente com quaisquer condições implicadas por essas mensagens. Também descreve as restrições de visualização que cada domínio associou à sua visão particular.
- De implementação dos dados e serviços dos domínios integrantes da federação. São as informações de relacionamento entre a especificação e a implementação de cada dado e serviço tais como: protocolo de transporte e endereços de localização, parâmetros de entrada e saída de cada aplicativo, comportamentos de exceção (exemplos: nomes de serviços de recuperação em caso de falhas, de serviços de mensagens de erros, etc.). Mantém também informações sobre as estruturas de armazenamento, as fontes de dados e os ambientes em que elas residem.
- De qualidade no acesso aos dados disponibilizados via federação. São informações para implementação da política de segurança e da política gerencial. As informações de segurança agregam as autorizações de acesso e níveis de autonomia de visualização de cada domínio. As informações gerenciais incluem parâmetros para os níveis de serviço, tais como limiares para tempo de execução de aplicações ou quantidade máxima de linhas a serem retornadas por um aplicativo. Além disso, o repositório ainda mantém os dados recolhidos das fontes de dados e ambientes para fins diversos, tais como: contabilizações, monitoramento, avaliação de desempenho, rastreamento, log, auditoria, parâmetro de restrição de visualização e outros. O repositório também mantém a referência cruzada das aplicações que acessam a federação via CMID e das informações que elas acessam para facilitar o gerenciamento e a manutenção de programas e dados.
- Administrativas. São informações necessárias para o funcionamento da própria federação. Por exemplo, parâmetros de configuração, informações sobre os domínios e sobre os usuários da federação (administradores dos domínios, da federação e as aplicações que acessam a federação).

4.4 Os níveis e esquemas da Federação

A figura 4.3 demonstra os diferentes níveis e esquemas e o relacionamento entre eles.

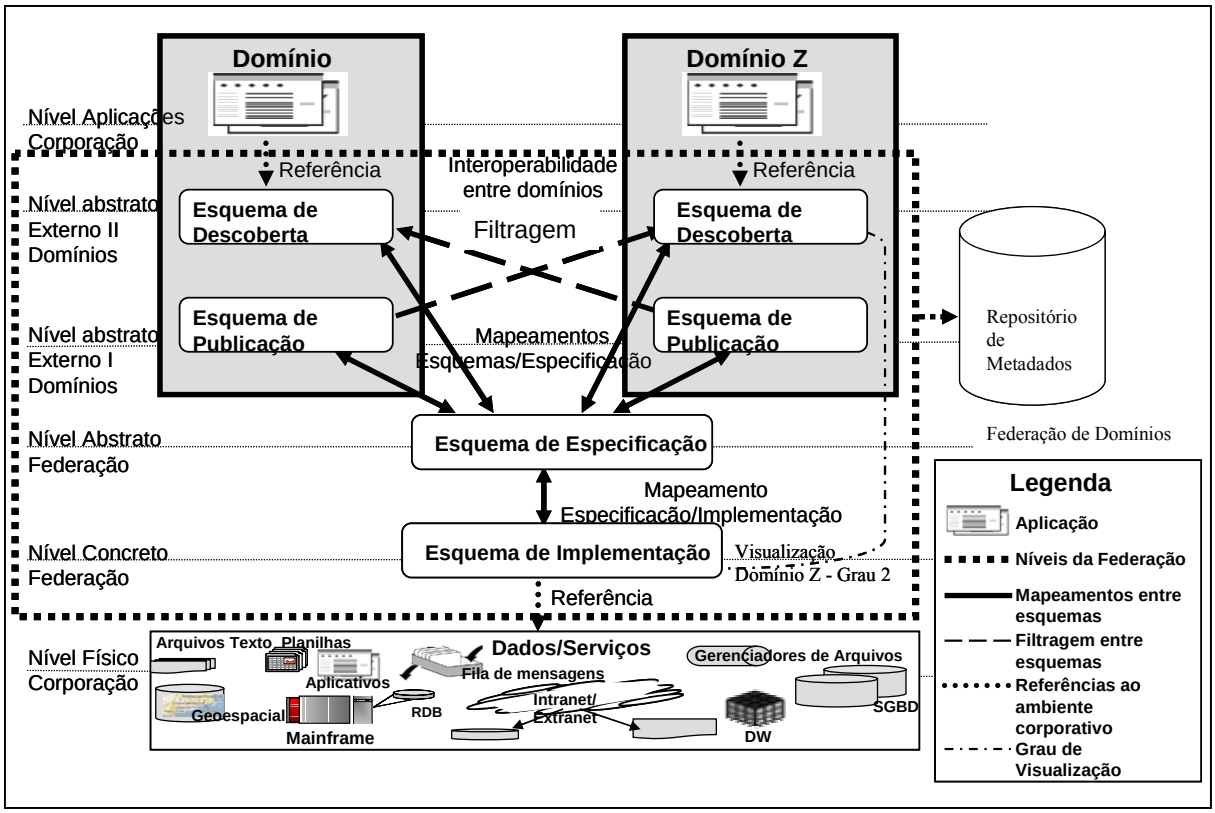

Figura 4.3 – Níveis e esquemas da federação

O nível concreto corresponde a uma percepção da Federação. Esse nível é uma referência à implementação dos aplicativos e dados corporativos que estão fisicamente distribuídos, armazenados e gerenciados por outras fontes de dados. Ele é descrito por meio do esquema de implementação que agrega as informações de implementação descritas anteriormente na seção 4.3.

O nível abstrato também é uma percepção da Federação, representando as informações e serviços especificados por cada domínio integrante da Federação. Esse nível é descrito por meio do esquema de especificação, que ignora a representação física das informações e serviços. O esquema de especificação agrega as informações de especificação descritas na seção 4.3.

Os níveis abstratos externos I e II são percepções individuais de cada domínio. O primeiro nível é uma representação das informações e serviços publicados por um domínio para serem compartilhados com outros domínios. Ele é descrito por meio dos esquemas de publicação. Como ele é uma visão do esquema de especificação definida para um domínio, ele também não contém referências quanto à implementação das informações e serviços.

O segundo nível representa as informações e serviços que um domínio pode descobrir e utilizar. Este nível é descrito por meio dos esquemas de descoberta (vide seção 4.5). Os esquemas de descoberta também podem visualizar partes dos esquemas de publicação (vide seção 4.5) de outros domínios, e essa capacidade proporciona interoperabilidade entre os domínios. Mais tarde se verá que esta visibilidade é condicional à aprovação dos administradores de domínios.

As aplicações dos domínios vão referenciar os dados e serviços corporativos através dos esquemas de descoberta. A Federação também armazena dados sobre as aplicações, assim como a referência cruzada entre elas e os dados que elas acessam.

Além dos quatro níveis, a Federação também mantém certos mapeamentos: um mapeamento especificação/implementação e vários mapeamentos publicação/especificação e descoberta/especificação.

O mapeamento especificação/implementação define a correspondência entre a visão de especificação e a visão de implementação. Havendo uma mudança na definição da implementação de um dado ou serviço, como, por exemplo, mudança de sua localização de uma estrutura para outra, ou de uma fonte de dados para outra, o mapeamento especificação/implementação deverá ser alterado de acordo, porém os esquemas dos níveis de especificação, publicação e descoberta não sofrem alteração. Dessa forma a Federação preserva a independência de dados física, pois as mudanças na estrutura física que armazena os dados e serviços são transparentes para os programas.

Da mesma forma, os programas são imunes a alguns tipos de alteração na estrutura lógica, como, por exemplo, alguns tipos de alteração de formato de dados, alteração de nomenclatura de dados e aplicativos, inclusão/exclusão tanto dos parâmetros de aplicativos como dos campos das estruturas de armazenamento. Os efeitos dessas mudanças são isolados abaixo, no nível de implementação, proporcionando um certo grau de independência de dados lógica.

Os mapeamentos publicação/especificação e descoberta/especificação definem a correspondência entre a visão externa abstrata, específica de cada domínio, e a visão abstrata federada. Esse nível de abstração, principalmente do nível externo II (mapeamento descoberta/especificação) aumenta ainda mais a capacidade de independência de dados lógica. Através das restrições de visualização, que mapeiam inclusive as diferenças semânticas, os domínios podem ter uma visão externa particular independentemente de alterações lógicas dos níveis abaixo.

As filtragens publicação/descoberta definem a visão de um domínio sobre os dados publicados por outros domínios através de seus esquemas de publicação. Esse tipo de interação permite o compartilhamento dirigido dos dados. Nenhum domínio visualiza os dados de outro domínio sem o controle deste último.

4.5 Interoperabilidade entre domínios

Os domínios, através de suas aplicações, são produtores e consumidores dos dados da Federação. Domínios podem precisar consumir dados de outros domínios, podendo, portanto criar relacionamentos com outros domínios. Para a Federação funcionar apropriadamente, o compartilhamento de dados entre domínios precisa ser feito de forma que estes preservem sua autonomia o máximo possível. Para isso é preciso que:

- cada domínio possua isolamento administrativo – de seus dados e serviços, de outros domínios.
- os domínios não manipulem diretamente quaisquer dados de outros domínios. Isso é feito somente através dos serviços de acesso a dados.
- os dados e operações internos a um domínio sejam transparentes para outros domínios.
- cada domínio tenha autonomia de operação, ou seja, os domínios é que determinam os dados e serviços que querem compartilhar, com quem querem compartilhar e de que forma querem compartilhar.
- um domínio tenha autonomia de associação, incluindo e retirando acesso aos seus dados e serviços, contanto que estes não estejam em uso por outros domínios.
- cada domínio tenha autonomia de visualização, ou seja, tem o direito de reestruturar as informações para a forma mais apropriada para as suas necessidades. Essa reestruturação pode ser feita através de serviços de restrição de visualização e são particulares para o domínio que as criou.

Para a interoperabilidade entre os domínios, é necessário que haja uma interpretação correta da informação. Para isso é preciso que tanto o domínio provedor da informação quanto o domínio consumidor não apenas falem a mesma língua, mas também compartilhem o mesmo conceito da informação. Isso inclui um vocabulário compartilhado, garantindo que os termos tenham um mesmo significado para ambas as partes.

Cada domínio deve descrever, com uma semântica clara, cada informação que estiver publicando na Federação, juntamente com os serviços que as acessam. Esse conjunto de descrições é chamado de esquema de publicação e é composto de:

- descrições das especificações das informações e serviços sob responsabilidade do domínio;
- relacionamentos entre cada mensagem de entrada e saída de um serviço e as informações especificadas.

O esquema de descoberta de um domínio descreve:

- especificações das informações e serviços sob sua responsabilidade;
- especificações das informações e serviços de outros domínios que foram disponibilizados para ele;
- relacionamentos entre cada mensagem de entrada e saída de cada serviço do domínio com as informações especificadas;
- restrições de visualização e dos serviços que as efetuam (se for o caso).

As restrições de visualização são uma forma da federação garantir que os domínios mantenham sua autonomia de visualização. Elas podem ser funcionais ou não-funcionais.

Uma restrição de visualização funcional é aquela que impõe uma transformação a que os dados precisam ser submetidos para se adequarem às necessidades de um domínio. Por exemplo, a visão federada de um determinado dado é expressa em uma unidade de medida e uma visão particular do domínio "B" sobre esse mesmo dado é expressa em outra unidade de medida.

As restrições de visualização não-funcionais geralmente estão relacionadas às preferências de acesso de um determinado domínio. A Federação precisa saber qual serviço acessar no caso de uma informação ter vários serviços que a disponibilizem, ou ter sido implementada de várias maneiras.

Como demonstrado na figura 4.4, a Federação oferece dois graus de visualização para os esquemas de descoberta. Isso tem por objetivo relaxar um pouco o trabalho de mediação da camada integradora de dados para os acessos efetuados pelas aplicações dentro do próprio domínio.

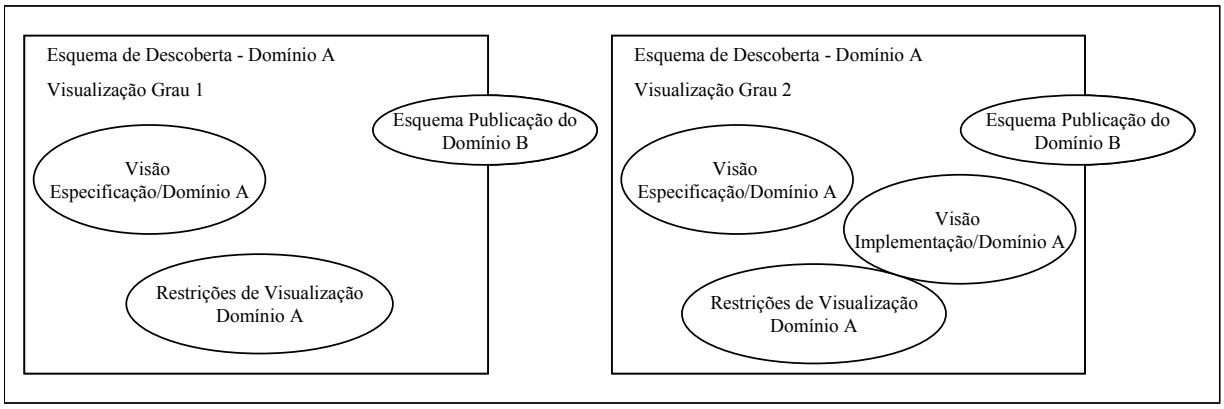

Figura 4.4 – Graus de visualização

1) Grau 1 – Visão somente das especificações. Nesse tipo de visualização, as aplicações de um domínio visualizam o esquema de especificação e as restrições de visualização dos dados sob controle do próprio domínio e partes dos esquemas de publicação de outros domínios aos quais tenha acesso. Nesse caso, todos os acessos devem referenciar os nomes dos dados e entidades como eles foram especificados (e não os nomes implementados).

2) Grau 2 – Permite acesso à especificação (grau 1) e implementação. Neste caso, as aplicações de um domínio podem ter acesso direto às estruturas implementadas sem a necessidade de passar por tradução especificação→ implementação. Este tipo de visualização é indicado para melhorar o desempenho de acesso aos dados em aplicações que não necessitam efetuar transformações em seus dados, ou aplicações legadas.

4.6 Administração da Federação

A Federação prevê dois tipos de administradores conforme a abrangência de sua atuação: os administradores de domínio e o da federação.

Os administradores de domínio têm a responsabilidade de:

- especificar os objetos sob responsabilidade de seus domínios. Entre outros, definir a conceitualização de cada informação, serviços e entidades, as mensagens trocadas pelos serviços, os relacionamentos entre as mensagens e as informações;
- determinar quem pode acessar os dados publicados e de que forma, por exemplo: se através de serviços do tipo genérico ou específico;
- disponibilizar para a Federação, através de esquemas de publicação, a parte de suas bases de dados que quer compartilhar;

- providenciar a geração de *stored procedures*, aplicativos e outros para implementar os serviços de acesso aos dados publicados;
- definir quais informações e serviços publicados por outros domínios serão incorporados em seu esquema de descoberta, contanto que obtenha autorização;
- associar as restrições de visualização necessárias para a aderência de dados externos às suas necessidades;
- providenciar a geração de serviços necessários para atender às restrições de visualização;
- providenciar para que os serviços disponibilizados atendam aos níveis de serviço determinados pela Federação, entre outros, quanto à segurança e desempenho.

O administrador da federação é responsável por:

- coordenar a visão federada, garantindo que os domínios respeitem as regras da federação quanto aos níveis de serviço exigidos;
- administrar os dados federados através da configuração dos diversos esquemas na federação de dados;
- configurar parâmetros para a administração da federação, para fins de segurança, monitoração e desempenho de uma forma global;
- monitorar o funcionamento da federação. Por exemplo, verificar logs e mensagens geradas, efetuar rastreamento para identificação de problemas ou validar os dados em redundância envolvidos em replicação, de maneira a certificar a consistência desses dados;
- configurar parâmetros para os domínios quanto ao tipo de controle que a camada mediadora dos dados terá sobre suas aplicações (vide seção 4.9 para mais detalhes).

4.7 Configuração da visão federada

A visão Federada permite que a aplicação trabalhe com múltiplos domínios. Como se verá na seção 4.8, a visão federada é garantida e gerenciada pela camada CMID da arquitetura proposta. O objetivo é que a organização possa garantir a integridade de dados e aplicações. Uma visão federada é formada por dois conjuntos básicos:

- esquemas de especificação;
- regras de mapeamento entre os esquemas de especificação/implementação.

A visão deve, além disso, ser submetida a restrições globais da organização, tais como:

- obediência às regras de negócios globais;

- estabelecimento de padrões de semântica para descrição, entre outros, de serviços, dados, mensagens, restrições de visualização, para a organização como um todo. Todos os domínios que fizerem parte da Federação devem concordar, entender e utilizar esses padrões;
- identificação com um nome único, com semântica clara e compreendida pela organização, de todos os objetos pertencentes à federação, tais como dados, serviços e fontes de dados.

Conflitos entre modelos, tipos e semânticas dos objetos devem ser resolvidos entre os administradores de domínios e o da federação. A federação não determina a forma como esses conflitos serão resolvidos; ela determina somente que haja um acordo comum para a adoção de um padrão para o objeto de discordância. O padrão adotado será incorporado à visão federada. As visões particulares divergentes da visão federada serão resolvidas através das restrições de visualização dos domínios e farão parte de seus esquemas de descoberta.

O fluxo de eventos para a configuração de um novo domínio de forma a incluí-lo na Federação segue os seguintes passos básicos:

1) Cada novo domínio que queira pertencer à Federação precisa incorporar os dados e serviços sob seu controle ao esquema de especificação da federação.
2) Se houver subconjuntos com sobreposição de informações em comum, os administradores dos domínios envolvidos devem determinar quem é o responsável primário sobre essas informações. Eventualmente os administradores dos domínios podem decidir que esses dados precisam estar sincronizados para que a federação possa prover o sincronismo dessas informações de forma a manter a integridade dos dados. Uma maneira de isso ser implementado é associar a execução de uma operação a outras operações, onde cada uma atualiza os dados em um ambiente.
3) Os administradores devem refazer as regras de mapeamento entre os novos integrantes com o resto da visão federada já existente.
4) O novo domínio vai publicar os dados que queira compartilhar.
5) O novo domínio definirá o que precisa acessar através de seu esquema de descoberta.
6) Se o esquema de descoberta do novo domínio envolver dados de outros domínios, será necessário obter autorização de acesso do dono dos dados.

7) Se houver alguma incompatibilidade entre a sua visão particular sobre os dados e a visão federada, o novo domínio precisa definir as suas restrições de visualização em cima dessa visão divergente e, se for o caso, fornecer serviços que as implementem.
8) Se houver mais de um meio de conseguir os dados que implementam as informações às quais o domínio tem acesso, ele pode definir parâmetros para facilitar a escolha de qual dado será acessado. Esses parâmetros podem ser, por exemplo: a) o nome de uma fonte de dados ou b) uma característica do ambiente em que a fonte de dados residir, como o tempo de resposta ou a localização geográfica. Se nada for definido, a configuração *default* será acionada, ou o domínio pode escolher que as aplicações passem esse parâmetro em tempo de execução.
9) O administrador da federação configura os parâmetros para o domínio, como tipo de autonomia de visualização, tipo de autonomia em relação ao controle da camada mediadora, parâmetros gerais de nível de serviço que o domínio deve seguir e parâmetros para controle de acesso.

4.8 Visão geral da Arquitetura

A Arquitetura de Integração de Dados (AID) proposta neste trabalho foi idealizada para permitir acesso controlado e transparente de dados compartilhados entre os diversos domínios dentro de uma organização – vide figura 4.5.

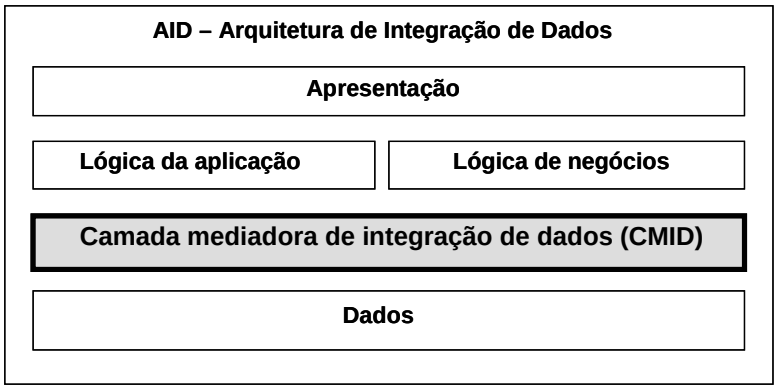

Figura 4.5 – Arquitetura proposta – visão geral

As camadas da figura 4.5 têm as seguintes funcionalidades:

- A camada de apresentação é responsável pela formatação e controle dos dados necessários para a *interface* com o usuário.
- A camada de lógica da aplicação é responsável pelo controle da seqüência de operações.

- A camada de lógica do negócio é responsável por aplicar as regras de negócios do domínio da aplicação.
- A camada mediadora de integração de dados (CMID) é responsável por: a) viabilizar acesso integrado às fontes de dados participantes da federação de domínios, b) reforçar a aplicação das regras de acesso e manipulação dos dados entre domínios e c) fornecer qualidade de acesso aos dados. A CMID é detalhada na seção 4.9.
- A camada de dados é responsável por manter as diversas fontes de dados. Os dados podem ser fornecidos por aplicativos, por sistemas gerenciadores de arquivos ou de banco de dados, ou por serviços. As fontes de dados podem ser estruturadas, semi-estruturadas ou não-estruturadas.

4.9 A camada mediadora de integração de dados (CMID)

A Arquitetura de Integração de Dados proposta sugere a inclusão de uma camada mediadora de integração de dados. O usuário e/ou aplicação não precisam se preocupar com diferenças de modelo, de tipo, de semântica, de tecnologia nem mesmo com a real localização de dados ou de como acessá-los. O resultado retornado para o solicitante pode ser um combinado de várias fontes de dados localizadas em qualquer lugar da rede interna ou externa a que ele tenha acesso.

A figura 3.5 utilizada para ilustrar o modelo orientado a serviço simplificado foi adaptada, neste capítulo (figura 4.6) para mostrar o modelo orientado a serviço da arquitetura proposta. Utilizou-se o mesmo procedimento para a figura 3.4 (interações em uma SOA), adaptada neste capítulo para a CMID (figura 4.7).

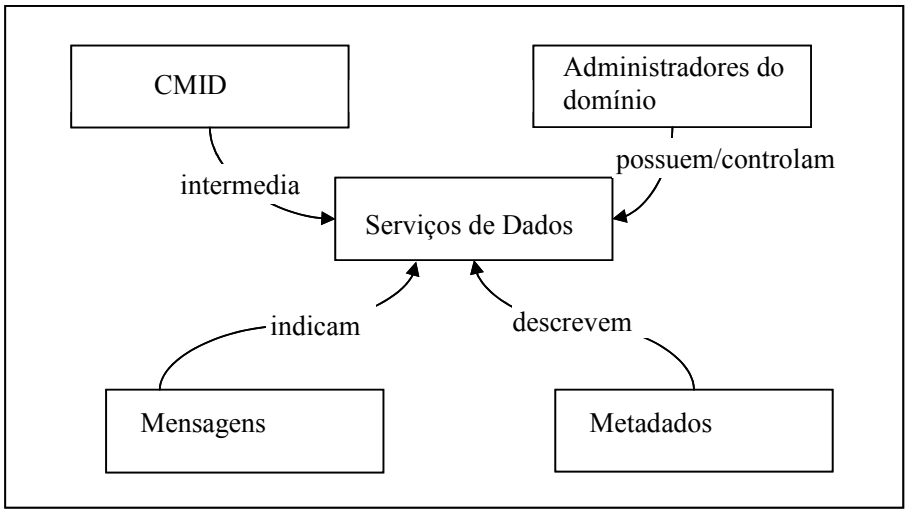

Figura 4.6 – Modelo orientado a serviços da arquitetura proposta

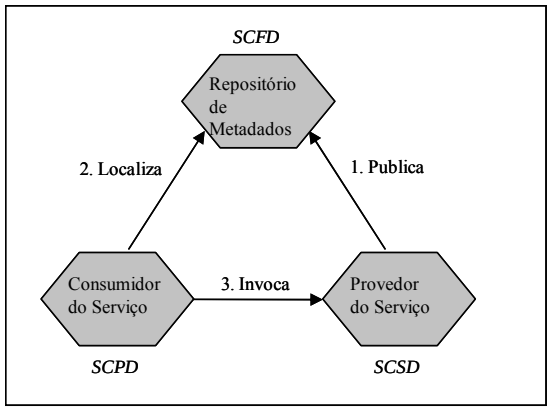

Figura 4.7 – Interações SOA na CMID

A camada de integração (CMID) e suas subcamadas estão na figura 4.8. A CMID baseia sua funcionalidade em três subcamadas responsáveis pelo processo de mediação: subcamada de recebimento e entrega de dados (SCRED), subcamada de processamento de dados (SCPD) e subcamada de serviços de dados (SCSD). Ortogonalmente, são fornecidas camadas adicionais, agregando suporte, viabilização e qualidade aos serviços de integração. A subcamada de federação de dados (SCFD) agrega tanto a parte de mediação como a de qualidade. As camadas de controle de acesso (SCCA) e a de gerenciamento de dados (SCGD) agregam qualidade aos dados.

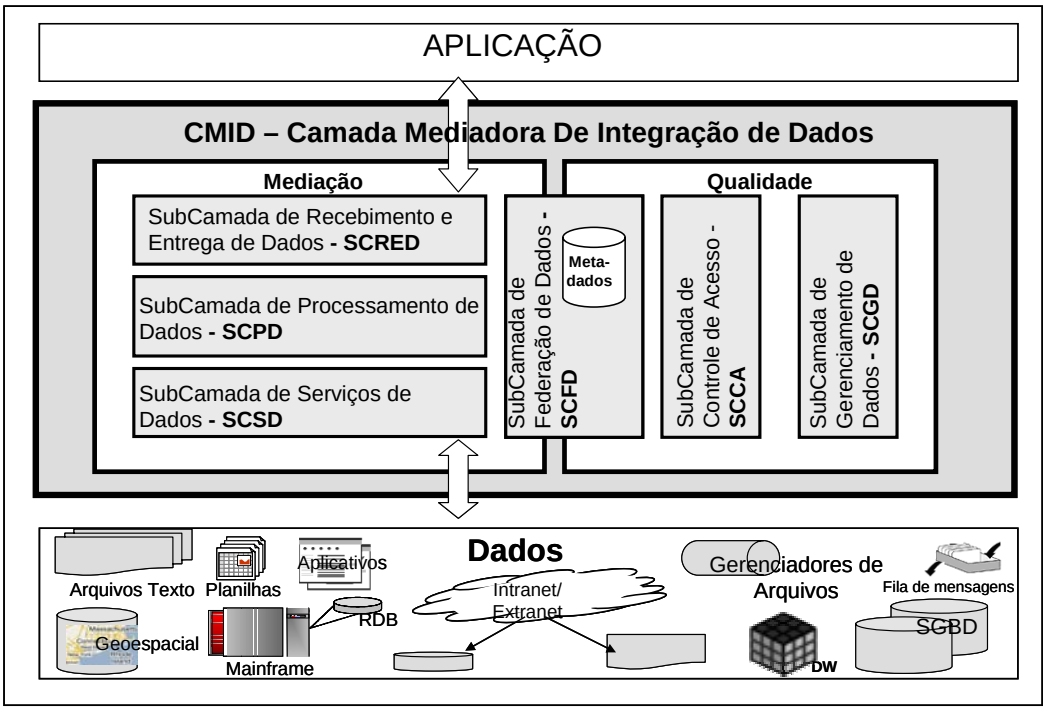

Figura 4.8 – Arquitetura proposta – Detalhamento da CMID

As subcamadas da CMID fornecem as seguintes funcionalidades:

A subcamada de recebimento e entrega de dados (SCRED) é responsável por receber e validar os comandos enviados pelas camadas de lógica da aplicação e/ou de negócio e repassá-los para a subcamada de processamento de dados (SCPD). Ela é também responsável por entregar os dados que foram formatados de acordo com a solicitação do cliente.

A subcamada de processamento de dados (SCPD) tem a responsabilidade de processar tanto os comandos de entrada como os resultados providos pelas fontes de dados e efetuar as decomposições, agregações, transformações, junções, filtragem, necessárias para processar os dados.

A subcamada de serviços de dados (SCSD) é responsável por disponibilizar serviços que buscam os dados em cada fonte de dados envolvida na operação. A subcamada de federação de dados (SCFD) é responsável por manter e gerenciar o repositório global e a federação de domínios. A subcamada de controle de acesso (SCCA) é responsável pela implementação da política de segurança de acesso adotada pela federação.

A subcamada de gerenciamento de dados (SCGD) é responsável pela política de gerenciamento adotada pela federação, incluindo serviços de auditoria, medição, monitoramento, avaliação de desempenho, entre outras facilidades, para agregar qualidade aos dados disponibilizados de uma forma global e integrada.

As seções 4.9.1 a 4.9.6 detalham as subcamadas da CMID e a seção 4.9.7 descreve como a CMID pode ser configurada na Federação de Domínios.

4.9.1 Subcamada de Recebimento e Entrega de Dados (SCRED)

Esta subcamada é a *interface* entre a CMID e as camadas de lógica da aplicação e/ou de negócios. A SCRED tem a funcionalidade de receber e validar as solicitações enviadas pelo consumidor da camada de integração de dados (CMID) e repassá-las para a subcamada de processamento (SCPD). Ela é responsável também por entregar os dados de acordo com os critérios definidos na solicitação do consumidor da CMID. Para atingir seus objetivos, essa camada fornece as seguintes funcionalidades:

- Recebimento e validação da mensagem recebida pela camada consumidora. Esta mensagem deve conter, no mínimo, a identificação do usuário, do domínio, da aplicação

chamadora e a operação a ser realizada. Adicionalmente, a mensagem pode também indicar critérios adicionais para a manipulação dos dados, tais como:

a) Especificação do serviço de acesso a dados a ser executado.
b) Especificação de algum critério de escolha, caso haja mais de um serviço fornecendo os dados solicitados. Por exemplo: selecionar a partir da fonte de dados original, ou da fonte de dados que estiver com melhor tempo de resposta no momento.
c) Especificação de critério de escolha quando houver mais de uma ocorrência para o mesmo dado, porém com valores diferentes por problemas de redundância da informação implementada em mais de um local. Por exemplo: selecionar o que estiver com data de atualização mais recente, ou o que tiver maior valor.

- Validação da sintaxe da operação solicitada. A operação a ser realizada pode estar identificada no formato de implementação (na linguagem da fonte de dados destino) ou no de especificação (na linguagem da federação).
- Validação da solicitação quanto ao atendimento às regras impostas pelo domínio possuidor do dado. Por exemplo: se o acesso for de um determinado domínio D1, o domínio D2, possuidor do dado, impôs a obrigatoriedade de ser informado de um valor para o parâmetro P, ou então do acesso ser somente através do serviço S, ou da operação ser somente do formato F.
- Entrega do comando para a SCPD e recebimento dos dados dela, entregando-os para a camada solicitante.

4.9.2 Subcamada de processamento de dados (SCPD)

Processa as solicitações provenientes da SCRED e os dados retornados pela SCSD. Esta camada entende, através de interação com a SCFD, o formato de todas as informações sendo transmitidas e trata a informação *on the fly*. Fornece as seguintes funcionalidades:

- Traduzir os comandos que estiverem no formato de especificação para o de implementação.
- Decompor os comandos a serem executados pelos serviços de acesso a dados na SCSD.
- Descobrir os serviços apropriados e tomar decisões, de acordo com critérios previamente definidos, quando houver mais de um que atenda à solicitação.
- Invocar os serviços de dados mantidos pela SCSD.

- Processar os dados entregues pelos serviços sempre que for necessário ou solicitado. Nesses casos a SCPD é responsável por:
  a) Converter os dados para diferentes formatos.
  b) Aplicar critérios de manipulação de dados que foram solicitados na mensagem de entrada.
  c) Relacionar as informações de entrada referentes aos dados solicitados às do esquema de especificação, efetuando as devidas conversões.
  d) Efetuar transformações para adequação à visão particular do domínio sobre os dados. Quando o domínio associa um serviço a uma restrição de visualização, a SCPD executa esse serviço como parte das transformações dos dados, como se fosse uma função. Outros tipos de transformações envolvem mudanças tais como: formatos de datas, unidades de medida, de moeda, etc...
  e) Agregar, filtrar, ordenar ou juntar os dados vindos de diferentes fontes, heterogêneas ou não.
  f) Propagar atualizações de dados (replicar), se configurado na federação.

4.9.3 Subcamada de serviços de dados (SCSD)

Esta subcamada mantém as implementações de todos os serviços de dados e pode estar fisicamente distribuída em diversos ambientes. Os serviços de dados podem disponibilizar tanto os dados corporativos como os externos à empresa, através da Extranet e Internet, como, por exemplo, serviços de operadoras de cartão de crédito ou central de informações de crédito ao consumidor.

Os serviços podem ser implementados de várias maneiras e as escolhas dependerão do grau de autonomia de controle que o domínio quiser exercer sobre os seus dados. Serviços do tipo genérico, de maneira geral, retornam dados de uma fonte de dados específica, ou um tipo de fonte de dados.

Os serviços podem acessar os dados mantidos em suas respectivas fontes de dados através de diversos protocolos de comunicação e tecnologias de acesso aos dados, tais como: acesso nativo, via *Call Level Interfaces* (ODBC, JDBC), ou através de *gateways*.

4.9.4 Subcamada de federação de domínios (SCFD)

Essa camada mantém o repositório de metadados. Os serviços oferecidos possibilitam a administração e configuração da federação de domínios e o acesso à visão federada através da CMID.

As informações mantidas no repositório são necessárias para as subcamadas que processam a mediação dos dados, que descobrem e executam os serviços necessários tanto para a sua própria funcionalidade como para o acesso aos dados propriamente dito. As informações mantidas no repositório também são necessárias para que as subcamadas responsáveis para prover qualidade aos dados possam exercer suas funções de controle de acesso e gerenciamento de uma forma global.

Para atingir os seus objetivos a SCFD fornece as seguintes funcionalidades:

- Publicação de dados e serviços na federação.
- Descoberta de dados e serviços mantidos pela Federação – através desses serviços é que as outras subcamadas interagem com a SCFD.
- Uma *interface* para que os usuários (administradores de domínio, da federação e programadores de aplicação) possam administrar, configurar e consultar os objetos tratados pela Federação.

4.9.5 Subcamada de controle de acesso (SCCA)

Os dados muitas vezes estão espalhados por diversas fontes. Algumas possuem seu próprio mecanismo de controle de acesso, como os SGBDs, outras precisam que o sistema operacional implemente esse tipo de serviço, como é o caso de diretórios. Muitas vezes a tarefa de administrar o controle de acesso fica sob a responsabilidade de diferentes setores como banco de dados, desenvolvimento de sistemas ou administradores da rede. Esse tipo de situação pode levar a conflitos ou inconsistências na implementação de um controle apropriado para a corporação de uma forma global.

A proposta da SCCA é prover um meio único de controle de acesso independentemente das fontes de dados. Outra finalidade é isolar as aplicações *frontend* e os serviços *backend* de eventuais alterações nos mecanismos de controle de acesso. Essa facilidade provida pela CMID pode ser agregada com outras, como, por exemplo, regras de *firewall* para que os provedores das

fontes de dados só aceitem tráfego a partir da CMID, para aumentar ainda mais a segurança no acesso aos dados. A SCCA fornece as seguintes funcionalidades básicas:

- Autenticação – valida as credenciais do usuário, tais como usuário/senha ou um certificado digital.
- Autorização – verifica se o usuário/aplicação *frontend*/domínio pode acessar os recursos desejados (dados e serviços). Isso envolve, entre outros:
  a) Validação do domínio quanto aos dados e serviços que serão acessados – eles devem fazer parte do esquema de descoberta do domínio.
  b) Validação do tipo de serviço utilizado (genérico ou específico) para aquele domínio. Se o tipo de serviço for genérico, então é preciso validar se ele está adequado ao grau de visualização do domínio.

4.9.6 Subcamada de gerenciamento de dados (SCGD)

Assim como ocorre com o controle de acesso, o gerenciamento de acesso a dados é dificultado devido à distribuição desses dados por diversos ambientes e fontes de dados. A SCGD foi idealizada para fornecer condições para um gerenciamento corporativo do acesso aos dados.

Todas as subcamadas da CMID devem gerar eventos com informações tais como: o nome da aplicação, usuário, domínio, data e hora inicial e final de acesso, se foi um acesso autorizado ou não, se houve término normal ou não, tipos de erros que ocorreram, e outros, que servem de entrada para o controle via SCGD. Alguns desses eventos são registrados no repositório de metadados, servindo para, entre outras finalidades, referência cruzada de aplicações versus serviços, histórico de ocorrências de erros, ou estatísticas de utilização.

Além disso, uma importante funcionalidade da SCGD é prover mecanismos de coleta de informações estatísticas, desempenho e log em outros ambientes de forma a prover uma visão global, possibilitando um controle gerencial unificado.

A SCGD também se vale de parâmetros de configuração definidos pelo administrador da Federação, como, por exemplo, tempo máximo de execução de serviços. Nesse caso, dependendo do que foi configurado, a SCGD pode somente registrar o evento ou também pode interromper a execução do serviço. Outras funcionalidades incluem:

- Serviços de *Logging* – as requisições dos serviços, erros e outros eventos devem ser registrados em um *log*. Esses *logs* podem ser usados, entre outros, para monitoramento, auditoria e gerenciamento de desempenho. No caso de serviços distribuídos por vários ambientes, os *logs* podem ser locais, mas é necessário haver uma facilidade para sua consolidação.
- Monitoramento – visa detectar falhas e avisar diversos tipos de ocorrências de eventos previamente configurados na Federação de domínios, como por exemplo, tentativa de acesso não-autorizado, limiares que foram atingidos – de tempo máximo de execução, quantidade máxima de linhas, entre outros.
- Gerenciamento de desempenho – coleta de estatísticas de *throughput*, uso e tempo de resposta. Essas informações precisam ser registradas no repositório de metadados, pois podem servir de base para o serviço de descoberta. Diante de uma informação disponível em mais de uma fonte, com base em estatísticas, é possível escolher uma fonte de dados que está com um tempo de resposta melhor em detrimento de uma outra que estiver com tempo de resposta pior ou indisponível no momento. Uma outra utilidade é o balanceamento entre as diversas fontes de dados evitando sobrecarga de uma fonte.
- Medição – a partir dos registros dos eventos são gerados relatórios de utilização para fins administrativos ou de cobrança individualizada pela utilização de serviços.
- Serviços de validação – responsável por validar os dados replicados. A propagação de atualizações de dados pode ser configurada através da Federação. Nesse caso, o administrador da federação pode programar a execução de serviços para validação da integridade desses dados. Através do confronto entre as diversas ocorrências dos dados replicados é possível capturar inconsistências devido a falhas no processo de replicação.

4.9.7 Configuração da CMID na Federação de domínios

Os domínios podem ser configurados em três diferentes graus de autonomia em relação ao controle exercido pela CMID sobre suas aplicações em Grau 0, 1 e 2, como mostra a figura 4.9.

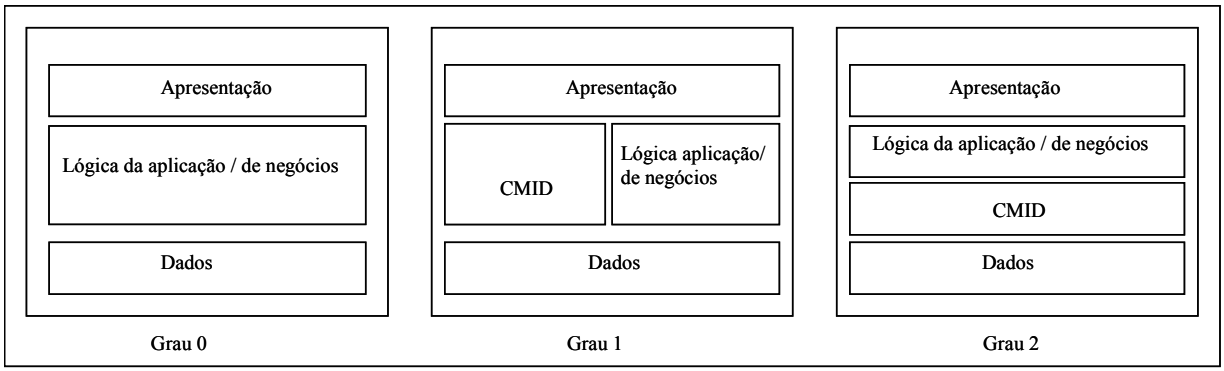

Figura 4.9 – Graus de autonomia de controle da CMID

O grau 0 corresponde à autonomia total, sem controle da CMID. As aplicações não usam a visão federada através da CMID para acesso aos dados. Indicado para:

a) Fins administrativos – aproveitamento da federação como uma forma de administração centralizada de dados e programas. O repositório de metadados provê uma visão corporativa com informações agregadas de diferentes ambientes, fontes de dados, estrutura de armazenamento.
b) Padronização – os programadores de aplicação podem utilizar os metadados armazenados na Federação como auxiliar no desenvolvimento das aplicações. Isso pode ser implementado, por exemplo, utilizando uma ferramenta que gere os serviços de acesso aos dados corporativos que serão incorporados ao programa, e não processados pela CMID.
c) Aplicações que não acessam dados de outros domínios.
d) Aplicações legadas que ainda estão se preparando para implementar a visão federada.

O grau 1 corresponde à autonomia total no acesso aos dados de seu domínio, podendo ou não ser intermediado pela CMID. O acesso aos dados de outros domínios é feito através da CMID. Indicado para:

a) As mesmas indicações dos itens “a” e “b” do Grau 0, exceto que a CMID também **pode** intermediar o acesso aos dados do próprio domínio.
b) Aplicações que não acessavam dados de outros domínios e agora passaram a acessar, e não querem ou não podem visualizar detalhes de implementação dos dados que não são da sua esfera de decisão.
c) Evitar ter inconsistências e responsabilidades devido a alterações nos dados de outros domínios.

d) Aplicações legadas que ainda estão iniciando a implementação da nova arquitetura através do acesso via visão federada.

Finalmente, o grau 2 visa o controle total da CMID. Quaisquer acessos a dados precisam ser intermediados pela CMID. Indicado para:

a) Mesmas indicações dos itens “a”, “b” e “c” do Grau 1, exceto que a CMID **precisa** intermediar o acesso aos dados dentro do próprio domínio.
b) Aplicações que acessam, em conjunto, dados provenientes de fontes heterogêneas.
c) Aplicações que necessitam transformar os dados antes de integrá-los em seu contexto.
d) Evitar trabalho de manutenção de vários programas devido a alterações físicas e lógicas nas estruturas de dados.
e) Obtenção de um ambiente controlado e padronizado no acesso aos dados.

4.10 Implementação baseada em serviços web

Tecnologias para serviços web não são necessariamente a melhor escolha para implementação de uma SOA. De modo geral, serviços web são mais apropriados para aplicações [21]:

- que precisem operar através da Internet onde confiança e rapidez não podem ser garantidos;
- onde não há habilidade para gerenciamento de distribuição de instalação (*deployment*), portanto todos os requisitantes e provedores têm suas versões atualizadas de uma vez;
- onde os componentes dos sistemas distribuídos são executados em diferentes plataformas e fornecedores de produtos;
- Onde uma aplicação existente precisa ser exposta para ser usada através da rede, e pode ser empacotada (*wrapped*) como um serviço web.

A arquitetura da CMID pode ser baseada em serviços web por razões de flexibilidade e neutralidade de plataforma através da utilização de padrões abertos. Outra vantagem é a possibilidade de haver apenas uma versão da CMID a ser utilizada por todas as diferentes plataformas que abrigarem os dados corporativos. Nesse caso, uma possível implementação utilizando serviços web seria a seguinte:

A SCFD pode fornecer mecanismos para a publicação e descoberta de serviços utilizando tecnologias como WSDL e UDDI. Na WSDL são descritos os tipos para os elementos de dados

usando esquema XML, as operações suportadas e suas definições, bem como as definições de mensagens, acoplamento e operações; enfim, as informações básicas necessárias para invocar o serviço a partir de um outro processo chamador. A localização da especificação WSDL pode ser especificada de duas maneiras: uma URI, que aponta para o arquivo WSDL, ou através da publicação em um UDDI.

A SCPD pode transformar dados XML para o formato apropriado através de ferramentas como o XML Style Language Transformation (XSLT).

A SCSD pode fornecer serviços web de acesso a dados que em conjunto com adaptadores de interfaces, podem prover interoperabilidade com diversas fontes através de tecnologias como LU6.2 (Logical Unit 6.2), CORBA, MQSeries, MSMQ (Microsoft Message Queuing**)**, COM+, RMI, etc. Hoje em dia, até mesmo os fornecedores de tecnologias mais antigas estão provendo facilidades de serviços web em seus produtos. A IBM, por exemplo, adicionou suporte de serviços web para Cobol e PL/1 (Programming Language One). Ela proveu o runtime do CICS e do IMS (Information Management System**)** para fornecer suporte a serviços web no CICS 2.2 e 2.3. O SOAP for CICS habilita programas CICS a serem tanto consumidores como provedores de serviços web [11]. Também é possível expor certas funcionalidades de sistemas legados através do wrapping do legado como um serviço web, ou através de *façades* ou adaptadores para serviços web [2, 3 , 8, 17, 18, 19, 20].

A SCCA pode conseguir seus objetivos através de padrões como SAML (Security Assertion Markup Language), que é um framework XML para troca de informações de autenticação e autorização ou WS-Security, que trata da troca de mensagens SOAP para fornecimento de qualidade de proteção através da confidencialidade e integridade da informação, assim como mensagem única de autenticação.

A SCGD pode ser implementada com base nas especificações do comitê técnico WSDM (WS Distributed Management) do OASIS. O WSDM está definindo dois conjuntos de especificações: a) MUWS (Management Using Web Services), que define como representar e acessar a gerenciabilidade de interfaces de recursos como serviços web, e b) MOWS (Management Of Web Services), que define o modelo de gerenciabilidade para gerenciamento de serviços web como um recurso e como descrever e acessar essa gerenciabilidade usando o MUWS.

4.11 Lidando com aplicações legadas

As aplicações legadas, no contexto deste trabalho, referem-se a quaisquer aplicações que, não importando a sua idade ou arquitetura, possuem um código, estão em uso e foram implementadas anteriormente à CMID.

Uma estratégia para a implementação da CMID envolvendo a organização como um todo e implementada de uma vez, abordagem conhecida por *cold turkey*, não é adequada à maioria das empresas. A abordagem mais adequada seria a *chicken little*, que envolve uma implementação incremental. Nesse caso, os domínios seriam incorporados um de cada vez.

Enquanto os domínios não estivessem migrados, as aplicações legadas utilizariam a federação somente para fins administrativos e de preparação para a migração. Enquanto a migração fosse preparada, a configuração da autonomia em relação ao uso da CMID seria de grau 0. Nessa fase, as informações disponíveis no repositório de metadados são importantes para que cada domínio possa ter uma visão geral e centralizada dos dados corporativos que utiliza, entender o conteúdo semântico de cada dado e conhecer o padrão de uso desses dados pelos outros domínios.

Na próxima fase, os domínios cujas aplicações acessem dados de outros domínios já participantes da federação passariam a ser de grau 1. Nesse caso, as aplicações utilizariam a CMID para acessos aos domínios externos. Para isso, é necessário que esses acessos sejam identificados e substituídos por chamadas à CMID.

A passagem para o grau 2 de autonomia é opcional com todos os acessos sendo intermediados pela CMID. O custo/benefício das alterações de todos os acessos aos dados do próprio domínio, que geralmente é em número bem maior do que os acessos externos, deve ser um importante fator para essa tomada de decisão. Nesse caso, quanto maior o grau de heterogeneidade das fontes dos dados acessadas, maior será o benefício obtido em razão das aplicações terem uma visão federada e transparente dos dados, em termos de localização e manipulação das diferentes tecnologias envolvidas.

4.12 Exemplo de aplicação

O exemplo dado nesta seção é de ordem didática e não esgota as possibilidades de configuração dos domínios e nem de utilização das funcionalidades da CMID. Outras

configurações de domínios na Federação seriam possíveis, assim como os serviços de dados disponibilizados poderiam ser definidos com maior ou menor granularidade.

Como exemplo de utilização da CMID para fins de integração de dados provenientes de diferentes fontes de dados de forma transparente para a aplicação, utilizaremos o ambiente corporativo de uma empresa fictícia do ramo de seguros, "Seguradora A". Essa empresa configurou na sua Federação de domínios os seguintes domínios: Auto, Vida e Sinistros.

O domínio "Sinistros", responsável pelos dados relativos aos sinistros ocorridos, possui uma aplicação que necessita dos dados das apólices de seguros com ocorrência de sinistro. Os domínios "Auto" e "Vida", responsáveis respectivamente pelos dados das apólices de seguros de automóveis e seguros de vida da empresa, publicaram na Federação as descrições de seus dados e serviços. O domínio "Vida" disponibiliza os dados de suas apólices através de uma visão e o domínio "Auto" disponibiliza os seus dados através de uma *stored procedure*. A aplicação do domínio "Sinistros" solicita à CMID as informações sobre as apólices. A SCRED recebe e valida essa solicitação, repassa o solicitado para a SCPD que localiza os serviços e consome os serviços que servem essas informações. A figura 4.10 ilustra essa situação. Durante todo o processo as subcamadas SCCA e SCGD se responsabilizam pela conformidade às políticas de segurança e gerenciamento ditadas pela Federação.

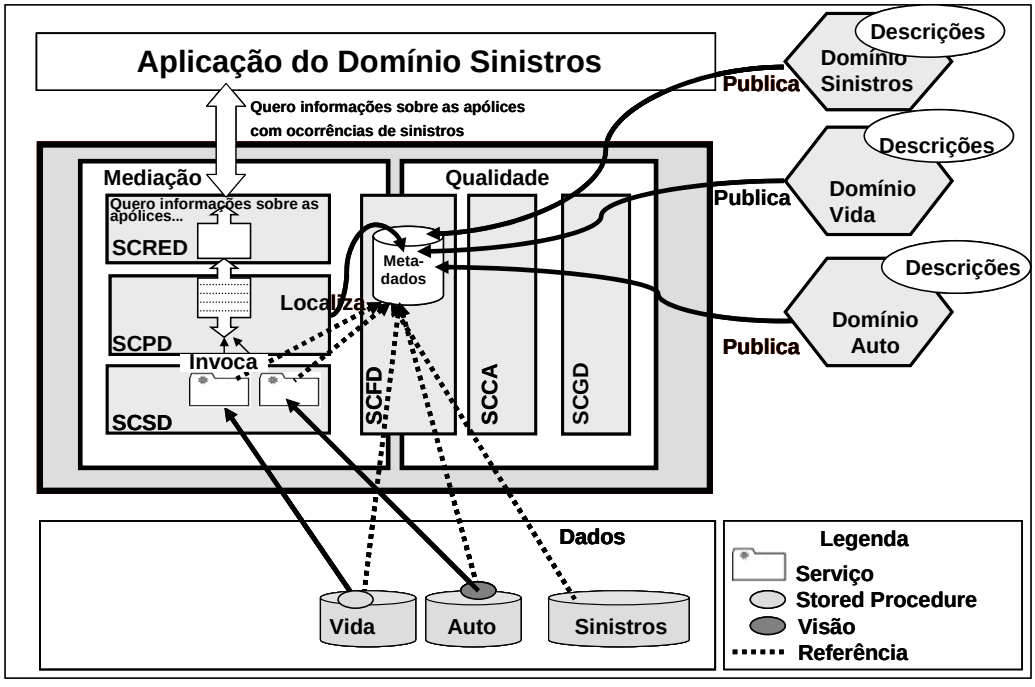

Figura 4.10 – Exemplo de aplicação da empresa "Seguradora A" utilizando a CMID

Em um momento posterior, a empresa "Seguradora A" adquire uma outra empresa de seguros, "Seguradora B", que comercializa somente seguros saúde. As bases de dados da "Seguradora B" residem no *mainframe* e os dados sobre as apólices gerenciadas pelo domínio "Saúde" são publicados na Federação e disponibilizados através de um aplicativo. Quando a aplicação do domínio "Sinistros" solicitar os dados sobre as apólices de seguros com ocorrências de sinistro, as informações referentes ao domínio Saúde farão parte do resultado retornado à aplicação. A aplicação não precisa saber da localização das bases de dados e nem dos mecanismos de acesso às mesmas. A figura 4.11 ilustra a aplicação do domínio "Sinistros" acessando a CMID após a inclusão do novo domínio "Saúde".

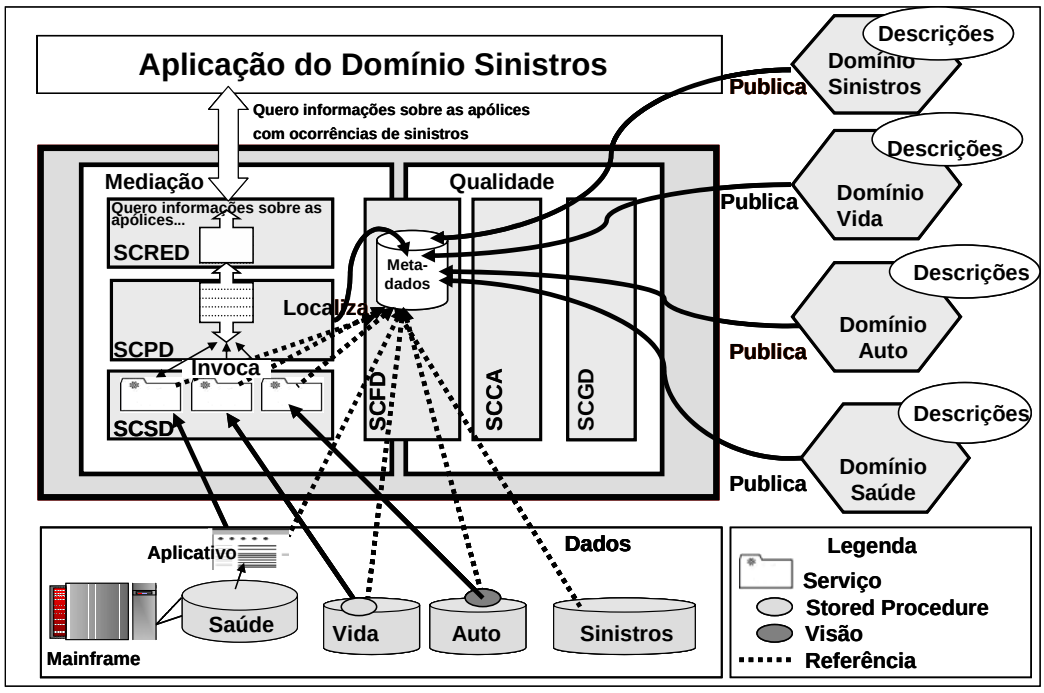

Figura 4.11 – Aplicação da empresa "Seguradora A" utilizando a CMID após a inclusão de um novo domínio.

4.13 Conclusão

A maior parte das empresas enfrenta problemas para integrar e controlar suas bases de dados. Estas geralmente ficam distribuídas, sem o controle de seu uso e com novos fatores dificultando a tarefa de integrá-los. No entanto, mesmo que bem definida, não basta ter uma arquitetura para a integração dos dados inter/intra-empresa. A integração entre pessoas é também necessária, principalmente entre aquelas responsáveis pelas informações. Outro aspecto essencial é o

controle sobre a utilização da arquitetura de integração. Finalmente, é importante que ela seja também flexível e possibilite a incorporação de novas regras, novas abordagens e novos componentes, além de permitir integração com o legado.

A proposta apresentada por este trabalho, detalhada neste capítulo, visa atender aos requisitos mencionados acima, endereçando vários problemas na integração e controle dos dados corporativos. Podemos dividi-la em três partes:

Primeiro, ela determina que os vários grupos de interesses dentro de uma empresa sejam organizados em uma Federação de domínios. Esses domínios vão cooperar entre si para a configuração da visão federada e preservação da autonomia e heterogeneidade de seus participantes;

Segundo, ela propõe a inclusão de uma camada mediadora de integração de dados (CMID) na arquitetura de aplicações em n-camadas. Essa camada disponibiliza, de forma flexível, a visão federada para as aplicações;

Terceiro, ela propõe que a arquitetura da CMID seja fundamentada nos conceitos de orientação a serviços, possibilitando tanto a mediação como o controle de qualidade no acesso aos dados corporativos.

Através da implementação da CMID seguindo a arquitetura orientada a serviços (SOA), consumidores e provedores dos dados corporativos poderão interagir independentemente de plataforma, protocolo ou linguagem de programação para acessarem as informações dos diversos domínios existentes. As aplicações consumidoras dessa camada não precisarão se preocupar com a forma de acesso nem com a real localização dos dados, pois a CMID se responsabiliza por executar essas tarefas de forma transparente e controlada.

Capítulo 5

CONCLUSÕES e EXTENSÕES

Este trabalho apresentou uma arquitetura para a integração dos dados corporativos, de uma forma unificada, abrangendo diversos aspectos relacionados à integração de dados, como metadados, especificação, implementação, comportamento, qualidade e controle sobre os acessos aos dados. Esta arquitetura, AID, foi motivada por necessidades encontradas em ambientes empresariais reais.

O objetivo principal dessa arquitetura foi facilitar o acesso aos dados e a interoperabilidade entre vários domínios no ambiente empresarial através do modelo de especificação, aumentando a consistência e o reúso através de serviços de dados. Além disso, garantir independência tanto da localização dos dados como da plataforma utilizada para implementação desses dados.

As principais contribuições dessa dissertação foram, assim:

a) análise de problemas encontrados na prática em ambientes empresariais para integração de dados;
b) discussão sobre arquitetura orientada a serviços (SOA) e serviços Web, sob uma visão prática, abordando questões como segurança, integridade, gerenciamento e outros quesitos de qualidade de serviço.
c) proposta de uma arquitetura que considerou a solução desses problemas, combinando enfoques de bancos de dados, sistemas distribuídos e soluções atuais do ponto de vista de serviços e sistemas Web.

Este trabalho utiliza uma extensão do conceito de "serviços de dados" abordados em [07] e [45]. Em [07], o projeto IBHIS (Integration Broker for Heterogeneous Information Sources), criado para a integração de dados do serviço de saúde do Reino Unido, utiliza o conceito de DaaS (Data as a Service) para "expor fontes complexas de dados como serviços". A arquitetura proposta no artigo [07], SODIA (Service Oriented Data Integration Architecture), é baseada em serviços Web e conceitos de bancos de dados federados, porém, como observado em [45], sem suportar serviços de mediação. O trabalho concluiu que serviços Web fornecem uma boa camada de infra-estrutura, porém ainda não estão maduros o suficiente para a finalidade complexa de um

sistema de integração como o IBHIS, entre outros, em relação à semântica suportada para descrição e descoberta dinâmica de serviços.

O trabalho em [45] propõe uma abordagem unificada para EAI (Enterprise Application Integration) que explora a mediação de ontologia e serviços Web. Essa abordagem, chamada de ODSOI (Ontology-Driven Service-Oriented Integration), objetiva estender a arquitetura de serviços Web inserindo uma camada de semântica. A arquitetura utilizada, ODSOA, também prevê *data services* para expor fontes de dados como serviços.

O conceito de domínios proposto nesta dissertação utiliza abordagem semelhante à utilizada em [46]. Esse trabalho descreve a arquitetura para integração implementada pela AGL (Australian Gas & Light Company). Nesse caso, as aplicações são particionadas em domínios. Os domínios se comunicam por mensagens através de uma infra-estrutura baseada em MOM, chamada "Federal Highway".

Extensões possíveis à AID podem ser tanto teóricas quanto práticas. Exemplos de extensões teóricas são:

a) Estudo sobre interoperabilidade semântica, abordando aspectos como linguagens e mecanismos, incluindo a viabilidade da abordagem vista em [45] para a federação e a CMID.
b) Análise de mecanismos para prover integridade em transações longas, abrangendo especificações como o BTP e sua aplicabilidade na CMID.
c) Análise comparativa sobre metodologias para integração entre XML e bancos de dados relacionais, com indicação ou proposta da abordagem mais adequada à implementação da CMID.
d) Análise das conclusões de [07], em relação ao UDDI e WSDL, e sua aplicabilidade na abordagem deste trabalho, focado em um ambiente de integração mais controlado, limitado às fronteiras da empresa e com parceiros confiáveis.
e) Incorporação de noções de interoperabilidade de processos

A principal extensão prática é a implementação da arquitetura usando algumas das características definidas ao final do capítulo 4. Ainda outras extensões do ponto de vista de implementação envolvem:

a) Plug-in para ambiente IDE (Integrated Development Environment), de forma a facilitar o desenvolvimento das aplicações fornecendo serviços como listagem, a partir do nome do

dado, de todos os serviços de acesso a esse dado publicados na federação e geração de *templates* para chamada à CMID.

b) Sistema de administração de dados para gerar e gerenciar o repositório de metadados da federação. Esse sistema deve prover formas de extrair dados externos como os contidos em catálogos de sistemas gerenciadores de bancos de dados, ferramentas CASE e ferramentas IDE.
c) Implementação de componentes para acesso à CMID.
d) Proposta de interface gráfica para gerenciamento das facilidades providas pela federação/CMID e para consultas *ad hoc*.

Lista de Acrônimos

ACID – Atomicity, Consistency, Isolation, Durability

ADS – Advertisement and Discovery of Services

API – Application Programming Interface

B2B – Business-to-Business

BTP – Business Transaction Protocol

CICS – Customer Information Control System

CLI – Call Level Interface

COBOL – COmmon Business Oriented Language

COM – Component Object Model

CORBA – Common Object Request Broker Architecture

DCOM – Distributed Component Object Model

DISCO – Discovery of Web Services

DOM – Document Object Model

EAI – Enterprise Application Integration

ETL – Extract, Transform, Load

FTP – File Transfer Protocol

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

IIOP – Internet Inter-Orb Protocol

JDBC – Java Database Connectivity

LDAP – Lightweight Directory Access Protocol

MOM – Message Oriented Middleware

MOWS – Management Of Web Services

MUWS – Management Using Web Services

OASIS – Organization for the Advancement of Structured Information Standards

ODBC – Open Database Connectivity

RMI – Remote Method Invocation

RPC – Remote Procedure Call

SAML – Security Access Markup Language

SAX – Simple API for XML

SBDD – Sistema de Banco de Dados Distribuído

SBDH – Sistema de Banco de Dados Heterogêneos

SGBD – Sistema Gerenciador de Banco de Dados

SLA – Service Level Agreement

SMTP – Simple Mail Transfer Protocol

SOA – Service Oriented Architecture

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

UDDI – Universal Description, Discovery and Integration

URI – Uniform Resource Identifier

URL – Uniform Resource Locator

W3C – World Wide Web Consortium

WSDL – Web Services Description Language

WSDM – Web Services Distributed Management

WS-I – Web Services Interoperability Organization

WSS – Web Services Security

XML – eXtensible Markup Language

BIBLIOGRAFIA

[01] Batini, C., Lenzerini, M., Navathe, S.B., "A Comparative Analysis of Methodologies for Database Schema Integration", ACM Computing Surveys, v.18 n.4, pp.323-364, Dec. 1986, ISSN: 0360-0300.

[02] Botto, R., "Arquitetura Corporativa de Tecnologia da Informação", 1a. Edição, Brasport, 2004.

[03] Cummins, F.A. "Integração de Sistemas", 1a. Edição, Editora Campus, 2002. Tradução de "Enterprise Integration", John Wiley & Sons, Inc.

[04] Date, C.J., "Introduction to Database Systems", 8a. Edition, Prentice-Hall Inc, 2003.

[05] Daum, B., Merten, U., "Arquitetura de Sistemas com XML", 1a. Edição, Editora Campus, 2002. Tradução de "System Architecture with XML", Morgan Kaufmann Publishers.

[06] Larson, J.A., Sheth, A.P., "Federated Database Systems for managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, v. 22 n. 3, pp.183-236, Sep. 1990, ISSN: 0360-0300.

[07] Zhu, F. et al, "Dynamic Data Integration using Web Services", IBHIS Project, Version 1. Versão Final publicada em - Proc. 2004, IEEE International Conference on Web Services (ICWS 2004), pp. 262-269, IEEE Computer Society Press. 2004.

[08] Microsoft Corporation, "Legacy Application Integration Inside the Firewall", Microsoft Windows Server 2003 White Paper, November 2002, disponível em http://www.microsoft.com/windowsserver2003/techinfo/serverroles/appserver/legappint.mspx, acessado em 05/2005.

[09] Ozsu, M.T., Valduriez P., "Princípios de Sistemas de Bancos de Dados Distribuídos", tradução da 2ª. Edição Americana, Editora Campus, 2001.

[10] Ramakrishnan, R., Gehrke, J., "Database Management Systems", 3rd. Edition, McGraw-Hill, 2003.

[11] Sharman, G., Cocker, M., "Delivering e-business access to CICS: strategic options", Cics Solutions Whitepaper, IBM Software Group, June 2004, disponível em http://www-

306.ibm.com/software/htp/cics/library/whitepapers/CICS_strategic_options_G224-7324-00_Final.pdf, acessado em 06/2005.

[12] Mark, E. et al, "Patterns: Service Oriented Architecture and Web Services", IBM Redbooks, April 2004, disponível em http://www.redbooks.ibm.com/abstracts/sg246303.html?Open, acessado em 05/2005.

[13] Silberschatz, A., Korth, H.F., Sudarshan, S., "Sistemas de Banco de Dados", 3a. Edição, Pearson Education do Brasil, 2004. Tradução de "Database System Concepts".

[14] Costa, G., "O Modelo de Web Services", Promon Tecnologia Business & Technology Review, Ano 02, no.04, disponível em http://www.promon.com.br/portugues/a_promon/download/PBTR4.pdf, acessado em 05/2005.

[15] Fisher, M., "The JDBC Tutorial", 05/1999, disponível em http://java.sun.com/developer/Books/JDBCTutorial/, acessado em 17/07/2005

[16] Shirolkar, P., "Data Access Technologies Road Map", disponível em http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmdac/html/mdac25.asp, acessado em 07/2005.

[17] Yajaman, N., "Web Services Façade for Legacy Applications", Microsoft Corporation White Paper, June 2003, disponível em http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/WSFacadeLegacyApp.asp, acessado em 05/2005.

[18] Krafzig, D., Banke, K., Slama, D., "Enterprise SOA: Service-Oriented Architecture Best Practices", Prentice Hall PTR, 2004.

[19] Linthicum, D. S., "Next Generation Application Integration: From Simple Information to Web Services", Addison Wesley, 2003.

[20] Erl, T., "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services", Prentice Hall PTR, 2004.

[21] W3C Working Group Note 11 February 2004, "Web Services Architecture", disponível em http://www.w3.org/TR/ws-arch/

[22] Kendall, S. C., Waldo, J., Wollrath, J. Wyant, G. "A Note On Distributed Computing", disponível em http://research.sun.com/techrep/1994/abstract-29.html, acessado em 08/2005.

[23] Kreger, H., "Web Services Conceptual Architecture", May, 2001. disponível em www-4.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, acessado em 09/2003.

[24] Cervantes, H., Hall, R.S., “Autonomous Adaptation to Dynamic Availability Using a Service-Oriented Component Model”, 2004, IEEE, Proceedings of the 26th International Conference on Software Engineering (ICSE’04), 0270-5257/04, pp 614-623.

[25] W3C XML Protocol Working Group, disponível em http://www.w3.org/2000/xp/Group/

[26] Skonnard, A., “Understanding SOAP”, March 2003, disponível em http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsoap/html/understandsoap.asp, acessado em 06/2005.

[27] Potts, S., Kopack, M. “Aprenda Web Services em 24 horas”, Editora Campus, 2003. Tradução de “Teach yourself Web services in 24 hours”, Sams Publishing.

[28] W3C Web Services Description Working Group, disponível em http://www.w3.org/2002/ws/desc/

[29] OASIS SOA Reference Model TC, home page: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

[30] Wilkes, L., “The Web Services Protocols Stack”, Report setembro/2003, disponível em http://roadmap.cbdiforum.com/reports/protocols/summary.php, acessado em 07/2005.

[31] Keen, M. et al, “Patterns: Implementing an SOA using Enterprise Service Bus”, IBM Redbooks, July 2004, disponível em http://publib-b.boulder.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg246346.html?Open, acessado em 04/2005.

[32] Patrick, P., “Impact of SOA on Enterprise Information Architectures”, Proceedings of the 2005 ACM Sigmod International Conference on Management of Data, pp 844-848, 2005, ISBN: 1-59593-060-4.

[33] Global Grid Forum, home page em http://www.gridforum.org/

[34] Bennet, K., "Legacy Systems: Coping with Success", IEEE Software, v.12 n.1, pp 19-23, Jan. 1995.

[35] Mani, A., Nagarajan, A. “Understanding quality of service for Web services”, Janeiro/2002, disponível em http://www-106.ibm.com/developerworks/library/ws-quality.pdf, acessado em 09/2003.

[36] Bloor Research North America, “Web Services Gotchas”, 2002, disponível em ftp://ftp.software.ibm.com/software/websphere/pdf/bloor_gotchas.pdf, acessado em 09/2003.

[37] Kothari, P., Trivedi, R., "Web Services Management: A Standards-Based Common Architecture", Outubro, 2002. disponível em http://www.developer.com/services/article.php/10928_1478281_2, acessado em 09/2003.

[38] Seth, M. "Introduction to Web Services Management", artigo disponível em http://www.developer.com/services/article.php/10928_1583511_1, acessado em 09/2003.

[39] Stephenson, J., Wilkes, L., "The Business Services Server – Approaches and products for Web Services Management", 2003, disponível em http://www.cbdiforum.com/downloads/bss.pdf, acessado em 09/2003.

[40] Sumra, R., Dhesiaseelan, A., "Quality of Service for Web Services -Demystification, Limitations, and Best Practices for Performance", Agosto, 2003. disponível em http://www.developer.com/java/web/article.php/2248251, acessado em 09/2003.

[41] Yang, A. "XML Web Services – The long term security risks", 2002, disponível em http://www.westbridgetech.com/downloads/XMLWebServicesSecurityRisks.pdf versão áudio em http://www.eaixpo.com/Symposium.asp?SymposiumId=34, acessados em 09/2003.

[42] Delphi Group "Web Services 2002 – Market Milestone Report", white paper, disponível em http://www.delphigroup.com/research/whitepapers.htm, acessado em 09/2003.

[43] Fremantle, P.; Weerawarana, S.; Khalaf, R. "Enterprise Services", Communications of the ACM, v. 45 n. 10, pp.77-82, Oct. 2002, ISSN: 0001-0782.

[44] Gorton, I.; Liu, A., "Architectures and Technologies for Enterprise Application Integration", IEEE, Proceedings of the 26th International Conference on Software Engineering (ICSE'04),0270-5257/04, pp. 726-727, May 2004.

[45] Izza, S.; Vincent, L.; Burlat, P., "Ontology-Based Approach for Application Integration", 2004, First International Conference on Interoperability of Enterprise Software and Applications, disponível em http://interop-esa05.unige.ch/INTEROP/Proceedings/Doctoral/PerPaper/II-1-Izza.pdf, acessado em 09/2005.

[46] Wijegunaratne, I.; Fernandez, G.; Valtoudis, J., "A Federated Architecture for Enterprise Data Integration", IEEE, 2000 Australian Software Engineering Conference, 0-7695-0631-3/00, pp.159, April 2000.

[47] Ferguson, D. et al, “Secure, Reliable, Transacted Web Services”, October 2003, disponível em http://www-128.ibm.com/developerworks/webservices/library/ws-securtrans/ , acessado em 09/2005.

[48] Dixon, B. et al “Security in a Web Services World: a proposed Architecture and Roadmap”, A joint security whitepaper from IBM Corporation and Microsoft Corporation. April 2002, disponível no web site http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/, acessado em 09/2003.

[49] OASIS Web Services Security (WSS) TC. Home page em http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[50] OASIS Business Transaction Protocol Specification. Disponível em http://www.oasis-open.org/committees/business-transactions/documents/specification/2002-06-03.BTP_cttee_spec_1.0.pdf

[51] OASIS Business Transaction Protocol Primer. Disponível em http://www.oasis-open.org/committees/business-transactions/documents/primer/Primerhtml/BTP%20Primer%20D1%2020020602.html#_Toc10821003

[52] Business Process Execution Language for Web Services, May 2003, disponível em ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, acessado em 10/2003.

[53] Web Services Coordination (WS-Coordination), August 2005, disponível em ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf, acessado em 09/2005.

[54] Web Services Atomic Transaction (WS-AtomicTransaction), August 2005, disponível em ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf, acessado em 09/2005.

[55] XML Signature WG, Home page em http://www.w3.org/Signature/

[56] OASIS UDDI, home page em http://www.uddi.org

[57] OASIS Security Services (SAML) TC, Home page em http://www.oasis-open.org/committees/security.

[58] Navathe, S.B., “Evolution of Data Modeling for Databases”, Communications of the ACM, v.35 n. 9, pp.112-123, Sep. 1992, ISSN: 0001-0782.

[59] Ullman, J.D., Widom, J., ”A first course in database systems”, Prentice Hall, 1997.

[59] Teorey, T., "Database Modeling and Design", 3ª. Edition, 1999, Lecture Notes, Michigan University, disponível em http://www.eecs.umich.edu/~teorey/lec.notes.pdf, acessado em 02/2004.

[60] Powell, G., "Oracle High Performance Tuning for 9i and 10g", 2004, Digital Press.

[61] Bagui, S., Earp R., "Database Design using Entity-Relationship Diagrams", 2003, Auerbach Publications.

[62] Ozsu, T., Valduriez, P., "Distributed and Parallel Database Systems", ACM Computing Surveys, v. 28 n.1, pp. 125-128, March 1996, ISSN: 0360-0300.

[63] Baragoin, C. et al, "Getting Started on Integrating Your Information", IBM Redbooks, February 2003, disponível em http://www.redbooks.ibm.com/abstracts/sg246892.html, acessado em 09/2005.