UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

Relatórios Técnicos
do Departamento de Informática Aplicada
da UNIRIO
n°0012/2009

Arquitetura Orientada a Serviço - Conceituação

Camille Furtado
Vinícios Pereira
Leonardo Azevedo
Fernanda Baião
Flávia Santoro

Departamento de Informática Aplicada

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
Av. Pasteur, 458, Urca - CEP 22290-240
RIO DE JANEIRO – BRASIL

Projeto de Pesquisa

Grupo de Pesquisa Participante

Patrocínio

PETROBRAS

Relatórios Técnicos do DIA/UNIRIO, No. 00012/2009
Editor: Prof. Sean W. M. Siqueira Julho, 2009

Arquitetura Orientada a Serviço - Conceituação *

Camille Furtado[1], Vinícios Pereira[1], Leonardo Azevedo[1,2], Fernanda Baião[1,2], Flávia Santoro[1,2]

[1]Núcleo de Pesquisa e Prática em Tecnologia (NP2Tec)
[2] Departamento de Informática Aplicada (DIA) – Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
camillefurtado@gmail.com, viniciospereira@gmail.com, azevedo@uniriotec.br, fernanda.baiao@uniriotec.br, flavia.santoro@uniriotec.br

Abstract. Service Oriented Architecture is presented as being more flexible and capable to support services independent of platform and protocol in a distributed environment. In this work, we present the main concepts related to SOA in order to provide the main concepts related to an initiative of SOA.

Keywords: Service Oriented Architecture, service.

Resumo. A arquitetura orientada a serviços (SOA – Service Oriented Architecture) apresenta-se como sendo mais flexível e capaz de suportar serviços independentes de plataforma e protocolo em um ambiente distribuído. Neste trabalho, apresentamos os principais conceitos relacionados à abordagem SOA.

Palavras-chave: Arquitetura orientada a serviço, serviço.

* Trabalho patrocinado pela Petrobras.

Sumário

1 Introdução

1.1 Motivação

Organizações modernas precisam responder de forma efetiva e rápida às oportunidades do mercado. Uma organização de médio e grande porte possui diversos departamentos (ou áreas), os quais em geral utilizam diferentes aplicações para realizar suas atividades. Estas aplicações necessitam se comunicar de forma integrada com o objetivo de atingir agilidade e simplificar processos de negócio, tornando-os mais produtivos, frente à crescente e a intensa competitividade do mercado.

O uso de serviços e de seus padrões de suporte à integração automatizada de negócios levou a grandes avanços na integração de aplicações. Mais notavelmente, a arquitetura orientada a serviços (SOA – Service Oriented Architecture) apresenta-se como sendo mais flexível e capaz de suportar serviços independentes de plataforma e protocolo em um ambiente distribuído. "Serviços" são módulos de negócio ou funcionalidades das aplicações que possuem interfaces expostas, e que são invocados via mensagens. Por exemplo, em um sistema bancário teríamos os seguintes serviços: serviço de nomes e endereços, serviço de abertura de conta, serviço de balanço de contas, serviço de depósitos etc. Serviços correspondem a recursos de software bem definidos através de uma linguagem padrão, são auto-contidos, provêem funcionalidades padrões do negócio, independentes do estado ou contexto de outros serviços [Erl, 2005].

SOA tem o propósito de tratar os requisitos de baixo acoplamento, desenvolvimento baseado em padrões, computação distribuída independente de protocolo, mapeamento dos sistemas de informação da organização para todos os seus fluxos de processos de negócios, integração de aplicações, gerência de transações, políticas de segurança e coexistência de sistemas em múltiplas plataformas e também sistemas legados [Papazoglou *et al.*, 2007]. Apesar da abordagem orientada a serviços requerer mais disciplina e planejamento, o retorno de investimento é elevado [Marks e Bell, 2006].

O objetivo de SOA é permitir às organizações realizarem seus negócios e ter vantagens tecnológicas por meio da combinação de inovação de processos, governança eficaz e estratégia de tecnologia, as quais giram em torno da definição e reutilização de serviços. Um dos benefícios mais importantes que SOA fornece é a melhora da produtividade e agilidade tanto para o negócio quanto para TI, e conseqüentemente, a redução de custos no desenvolvimento e manutenção dos sistemas envolvidos.

Existem muitos casos de sucesso de implantação de SOA em organizações, como apresentado em [SOA-Consortium, 2008a]. Por exemplo, a empresa de energia Valero Energy [Valero Energy, 2008] adotou a arquitetura SOA a fim de tratar suas necessidades de mudanças em óleo e gás devido a limitações no acesso a oportunidades de reservas e exploração remota, além dos riscos resultantes da volatilidade do mercado [SOA-Consortium, 2008b]. A estratégia adotada foi de executar ciclos curtos de aquisições de novas empresas, aumentando rapidamente sua presença e sua cadeia de valor através de estratégias de M&A (*Merge and Acquisition*). O objetivo em se empregar SOA foi de minimizar custo e separações forçadas, além de reduzir os ciclos de integração de 6 meses para de 2 a 8 semanas. Como resultados do uso de Arquitetura Orientada a Serviços, a Valero Energy alcançou flexibilidade e velocidade na mudança de processos de negócio. Como exemplos temos: clientes passaram a obter o preço rapidamente

nos terminais; mitigação de riscos e otimização do negócio, por exemplo, precisão nos dados comerciais, financeiros e de ganhos através da cadeia de valor; confiabilidade no sistema pela simplificação de interfaces pela redução de redundância de dados.

Outro caso de sucesso apresentado pela SOA Consortium é de uma importante empresa automobilística [SOA-Consortium, 2008c]. A empresa necessitava aumentar a satisfação de seus clientes e diminuir o tempo do ciclo de operações. Além disso, a empresa tinha como objetivo obter flexibilidade para executar mudanças rapidamente. A fim de alcançar estes objetivos, a empresa adotou uma estratégia de implantação de SOA definindo passos a longo prazo; iniciou uma mudança cultural através de melhorias de habilidades existentes, conhecimento e políticas; aplicou mudanças organizacionais; e utilizou uma gerência de transição ativa baseada em comunicação efetiva. Como resultado da implantação de SOA, a empresa melhorou a satisfação dos clientes através da redução de dados duplicados dos clientes e acesso às informações de veículos praticamente em tempo real. Além disso, a empresa aumentou a agilidade pela manutenção do foco em governança através de infraestrutura e serviços compartilhados na organização, facilidade de integração com parceiros (por exemplo, facilidade de acesso dos vendedores às informações de veículos e clientes).

1.2 Objetivos

A fim de desenvolver e implementar serviços em uma arquitetura SOA, é necessária a utilização de uma metodologia eficaz para análise, especificação, desenvolvimento e governança de serviços. Esta metodologia deve permitir aos interessados tirar o melhor proveito deste novo tipo de abordagem, diminuindo os riscos inerentes ao projeto. Todavia, SOA não pode ser comprado a partir de uma prateleira ou adquirido da internet, construído durante a noite. SOA é uma jornada. SOA não é nova, pois o uso de tecnologia para direcionar os objetivos do negócio vem desde os primórdios [Carter, 2007].

Portanto, é importante que seja definida a abordagem mais adequada à organização e mais efetiva para implantação de SOA. Vários aspectos têm que ser considerados, tais como, o entendimento dos conceitos relacionados a abordagem, definição dos propósitos da organização na implantação de SOA, definição de políticas e governança para SOA, definição de indicadores, definição do modelo organizacional e de comportamento, definição de tecnologia a ser empregada e definição da metodologia para desenvolvimento de serviços, contemplando identificação, modelagem, projeto e codificação de serviços [Marks e Bell, 2006].

Existem diferentes abordagens para identificação, análise e projeto de serviços. Serviços podem ser identificados a partir de análise dos processos do negócio, análise das entidades principais do negócio, oportunamente via iniciativas orçadas, entrevista com pessoas que detêm conhecimento especializado do negócio ou do domínio do negócio ou mesmo a partir de serviços pré-existentes [Marks e Bell, 2006]. De uma forma ou de outra todas estas propostas estão relacionadas à necessidade de identificar serviços a partir das atividades realizadas pelos diferentes departamentos da organização, ou seja, dos processos existentes na organização. Uma abordagem adequada para identificação e análise de serviços é a partir da análise de processos de negócio. A modelagem de processos do negócio promove o entendimento do negócio e explicita as atividades do negócio além do fluxo de execução destas atividades. A automatização destas atividades pode ser realizada diretamente através de serviços interoperáveis e reutilizáveis em SOA. De outra forma, a representação de conceitos do domínio do negócio e

das relações semânticas existentes entre os mesmos torna-se essencial e extremamente importante, apesar de não trivial.

O objetivo deste relatório é apresentar os principais conceitos relacionados à abordagem SOA a fim de prover o embasamento necessário para um estudo mais aprofundado dos principais conceitos relacionados a uma iniciativa de implantação de uma Arquitetura Orientada a Serviços.

Este relatório foi produzido pelo Projeto de Pesquisa em SOA como parte das iniciativas dentro do contexto do Projeto de Pesquisa do Termo de Cooperação entre NP2Tec/UNIRIO e a Petrobras/TIC-E&P/GDIEP.

1.3 Estrutura do Relatório

Os próximos capítulos desse relatório estão divididos como a seguir. No capítulo 2 a definição de SOA e dos principais elementos que suportam essa arquitetura. Os elementos mais importantes são detalhados em seções específicas desse capítulo. Além disso, são apresentados os principais desafios que devem ser superados por uma abordagem SOA e as tecnologias que podem apoiar a sua implantação.

No capítulo 3 , os conceitos relacionados a *Entreprise Service Bus* é apresentada com destaque, dado sua importância na implantação de uma iniciativa SOA. O ciclo de vida dos serviços levantados em uma iniciativa com essa arquitetura é apresentado no capítulo 4 .

O capítulo 5 apresenta os benefícios que uma iniciativa de gerenciamento de processos de negócio pode trazer a uma iniciativa SOA. Uma metodologia recomendada pela empresa IDS Scheer para obtenção desses benefícios através da sua ferramenta ARIS é apresentada no capítulo 6 .

O conceito de serviços de informação é introduzido no capítulo 7 .

Os capítulos 9 e 11 são responsáveis, respectivamente, pelas conclusões obtidas na pesquisa realizada para elaboração desse relatório, e por apresentar um glossário dos principais termos utilizados ao longo do texto.

2 Principais elementos de SOA

2.1 Definição de SOA

Primeiramente, SOA é um conceito. Existem diversas definições para SOA na literatura, mas é importante deixar claro algumas definições de SOA que NÃO procedem. SOA não é um produto, não é uma solução, não é uma tecnologia, não pode ser reduzido a produtos de software e, finalmente, não atende todos os desafios tecnológicos aos quais estão submetidos os negócios de hoje.

SOA é um estilo de projeto que guia todos os aspectos de criação e uso de serviços de negócio através de todo o ciclo de vida de desenvolvimento (desde a fase de concepção até a aposentadoria de serviços), bem como trata da definição e do provisionamento da infra-estrutura de TI que permite que diferentes aplicações troquem dados e participem de processo de negócio independente dos sistemas operacionais onde estas aplicações estão executando ou linguagens de programação utilizadas para suas implementações [Newcomer e Lomow, 2005].

2.2 Elementos de SOA

Os elementos essenciais para alcançar o sucesso em SOA são: Visão conceitual de SOA, serviços, tecnologia, governança SOA e estratégia SOA, indicadores e cultura e comportamento [Marks e Bell, 2006].

2.2.1 Visão conceitual de SOA

SOA é uma proposta de como as funcionalidades de TI podem ser planejadas, projetadas e entregues como serviços de negócio modulares para atingir um determinado benefício de negócio. Para alcançar os objetivos SOA, considerações organizacionais e de comportamento devem ser entendidas e alteradas primeiro, de forma gradual e ao longo do tempo.

2.2.2 Serviços

Os serviços são os artefatos centrais de SOA, os ativos primários de arquitetura. Juntamente com os serviços deve ser definido um modelo de projeto de serviços que assegure reutilização, interoperabilidade e integração por todos os processos de negócio e plataforma de tecnologias.

2.2.3 Tecnologia

A tecnologia é essencial para apoiar e alcançar SOA, mas a iniciativa SOA não se resume em uma tecnologia. A tecnologia deve ser disponibilizada para atingir os seguintes objetivos:

- permitir que os serviços operem de forma confiável e segura apoiando os objetivos do negócio; e,
- permitir evoluir na arquitetura de TI existente, possibilitando, por exemplo, que sistemas legados possam ser utilizados em processos de negócio a fim de apoiar os objetivos SOA. Em muitas organizações sistemas legados são os principais contribuidores de serviços para o SOA.

2.2.4 Governança SOA e estratégia SOA

SOA é uma estratégia que afeta a organização como um todo. A arquitetura conceitual SOA requer que sua visão e objetivos sejam comunicados aos envolvidos: usuários do negócio, desenvolvedores, arquitetos de TI, executivos de negócio e de TI, analistas de negócio e parceiros do negócio devem estar de acordo.

SOA é alcançado de forma incremental, ao longo do tempo por continuamente definir e garantir padrões em que ele será baseado. Portanto, o apoio à estratégia SOA é importante para tomar decisões apropriadas, e garantir tempo e recursos financeiros suficientes. Estes padrões formam as políticas SOA. O modelo de governança define processos de governança, regras e responsabilidades organizacionais, padrões e políticas que devem estar aderentes ao SOA. É necessária uma equipe central para determinar aspectos gerais de SOA que serão específicos para a organização.

2.2.5 Indicadores

Os indicadores são utilizados para medir os resultados alcançados. Uma das regras fundamentais de TI nos negócios é a medição. Deve-se estar apto a medir para se poder gerir. Logo, para os negócios tudo que TI produz deve ser mensurável. A documentação de informações que agregam valor tem sido o cerne de TI há alguns anos. Para alguns significa requerimentos inatingíveis que não provêem benefício algum. Em SOA, algumas coisas interessantes que eram difíceis ou impossíveis de se criar antes, se tornam viáveis. Se você puder disponibilizar alguma funcionalidade em serviço, em tecnologia, em um pedaço de hardware ou software, então você pode medir esse serviço. Obviamente isso levará a SLAs (*Service Level Agreements*) e a SLOs (*Service Level Objectives*), mas também levará a áreas tais como conformidade, auditoria e indicadores chave de desempenho (KPIs). E, medindo conformidade, auditorias e KPIs, a TI pode produzir medições que não são nem impossíveis nem sem valor. Essas medições podem incluir métricas de desempenho ou estatísticas de utilização de serviços, mas podem incluir outras questões, tais como, que dados estão transitando, quem pode acessar esses dados, se o dado contém determinada informação, entre outras. É nesse ponto que começamos a mudar medição (tempo de operação do serviço) para visibilidade (trinta e cinco usuários acessaram os dados desse serviço que continham a palavra "companhia"). Essa visibilidade possui um valor próprio e na criação de sua definição foi esboçada a necessidade do negócio (para entender quem está acessando os dados sob quais condições) em uma solução de TI mensurável.

Uma das coisas mais importantes em SOA é que qualquer coisa pode ser um serviço. Se qualquer coisa pode ser um serviço, então a visibilidade pode estar em todo lugar. Se um processo de negócio pode ser transformado em serviços de negócio, então podemos criar visibilidade nesses serviços, e fazendo isso, estamos criando visibilidade para a organização como um todo. A possibilidade de se criar visibilidade através de medições de serviços dentro da organização capacita seus gerentes a produzir mais do que antes.

Os indicadores devem ser planejados desde cedo e sempre ser avaliados para explicitar os resultados alcançados [Marks e Bell, 2006].

2.2.6 Cultura e comportamento

Iniciativas SOA são direcionadas a processos e abrangem toda a organização trazendo questões e desafios organizacionais, tais como:

- Quem são os proprietários de serviços?
    - É o departamento que desenvolveu o serviço?
    - É o cliente que mais utiliza o serviço?
    - É o cliente que utiliza o serviço no processo mais crítico?
- Como deve ser o relacionamento entre áreas do negócio e áreas de TI?
- Qual é a melhor prática de investimentos?
- Quem paga o desenvolvimento de um novo serviço?
    - O primeiro consumidor do serviço?
    - Todos os consumidores dos serviços?

- Provedor do novo serviço, e define-se um modelo de cobrança do uso do serviço?
- Deve-se reservar recursos financeiros específicos para a criação de novos serviços?

Para alcançar sucesso na implantação SOA, inicie com características comportamentais, culturais e outros fatores que garantirão o sucesso SOA [Marks e Bell, 2006].

2.3 Desafios

Existem vários desafios que devem ser considerados quando da implantação de uma arquitetura orientada a serviços [Marks e Bell, 2006].

2.3.1 O Modelo da arquitetura da organização pode necessitar ser melhorado para permitir SOA

O estado das arquiteturas de TI atualmente pode ser caracterizado como rígido, constituído de sistemas legados pesados, aplicações inflexíveis e um portifólio de aplicações que demandam softwares de integração. Porém, as questões mais importantes são flexibilidade e reutilização que poderão dar suporte aos processos de negócio que mudam freqüentemente com as mudanças e forças globais. SOA não vai alcançar o sucesso a menos que o processo de arquitetura mude de sua forma estática para um que se molde ativamente, com implementações flexíveis e reutilizáveis que dêem o suporte apropriado ao processo de negócio.

2.3.2 SOA tem distribuição espacial e temporal.

Um dos maiores desafios de SOA é que ele não é implementado de uma vez só. Ao invés disso, ele é alcançado através de muitos projetos distintos que ao longo do tempo e em diferentes localizações (ou departamentos). Essa distribuição temporal e espacial dos projetos de SOA faz da governança o fator mais crítico para o sucesso de SOA. A governança e as políticas de execução são as chaves para a gestão de conformidade de SOA pelos horizontes de tempo e espaço.

2.3.3 SOA é organizacionalmente complexo e desafiante do ponto de vista cultural.

A implantação de SOA é organizacionalmente, comportamentalmente e culturalmente desafiador à maioria das organizações.

Uma arquitetura de TI é um resultado de anos de comportamentos da organização, decisões de negócio, e escolhas estruturais. Para alcançar os objetivos SOA considerações organizacionais e de comportamento devem ser entendidas e alteradas primeiro, de forma gradual e ao longo do tempo. Modelos organizacionais novos e modelos de comportamentos serão essenciais para o sucesso SOA.

2.3.4 SOA requer governança para ser executado e gerenciado.

SOA requer um modelo robusto e claro de governança e uma forma de se implementar esse modelo por todo os ciclos de vida dos processos da organização: arquiteturas da organização, projeto de serviços, publicação, descoberta, e tempo de execução.

2.3.5 Desafios de identificação, modelagem e projeto de serviços.

Um processo de desenvolvimento para determinar a granularidade, gerenciamento de versão e a definição de políticas para desenvolvimento de projetos de serviços ao longo do tempo são essenciais. A principal unidade em uma arquitetura orientada a serviços é o serviço. Entretanto, o processo de identificação e análise de serviços não é simples. Por exemplo, como saber que os serviços corretos foram identificados? Como saber se um uma funcionalidade deve ser implementada como um serviço ou apenas como uma funcionalidade de uma aplicação? A resposta para estas perguntas traz questões a respeito de granularidade, reutilização de serviços e outras questões de modelagem e projeto de serviços.

2.4 Definição de serviços

Serviços são módulos de negócio ou funcionalidades das aplicações que possuem interfaces expostas, e que são invocados via mensagens.
Em SOA, recursos de software são empacotados como serviços bem definidos, autocontidos, provendo funcionalidades padrões do negócio, independentes do estado ou contexto de outros serviços.

Os serviços são expostos via suas interfaces e o projetista do cliente do serviço não tem acesso direto à implementação do serviço, mas apenas à sua interface.

[Marks e Bell, 2006] fazem a distinção entre dois tipos de serviços: serviços de negócio e serviços técnicos (ou de infra-estrutura).

Os serviços de negócio refletem o conceito do negócio e estão tipicamente associados à execução de uma função de negócio de uma organização. Eles ampliam naturalmente a linguagem do negócio e podem se tornar a ponte de terminologia entre TI e os usuários do negócio durante a análise e o projeto.

Os serviços técnicos ou de infra-estrutura são reutilizáveis por todos os processos de negócio. Eles são serviços de TI que devem ser nivelados por todas as linhas do negócio (por exemplo, serviços de segurança, de impressão, de log, de auditoria, de monitoramento de estatísticas em tempo de execução, de verificação de interfaces). Estes serviços tipicamente utilizam a infra-estrutura de serviços para prover alguma funcionalidade adicional, que não é focada no negócio.

De um ponto de vista puro de SOA, serviços técnicos não representam serviços, porque serviços deveriam representar sempre uma funcionalidade do negócio. Por outro lado, se temos a infra-estrutura de disponibilização de serviços, podemos utilizá-la para nos ajudar em outros aspectos. Na realidade, esta é uma questão de terminologia. O importante é deixar claro que estes serviços têm um propósito específico, a fim de que as equipes do negócio não tenham a impressão de que serviços não necessariamente precisam representar funcionalidades do negócio. É importante manter uma camada de abstração clara que encapsula detalhes técnicos [Josuttis, 2007].

2.5 Papéis de serviços

2.5.1 Requisitante de serviço (ou consumidor ou cliente).

O requisitante (cliente ou consumidor) é o serviço que invoca a funcionalidade provida por outro serviço.

2.5.2 Provedor de serviço.

O provedor oferece uma funcionalidade implementada através de uma interface.

2.5.3 Agregador de serviço.

O agregador de serviços combina a funcionalidade de provedor e requisitante. O agregador age como uma camada intermediária entre serviços consumidores e serviços provedores. Ele encapsula o acesso às funcionalidades providas por diferentes serviços provedores. Dessa forma, o agregador atua como provedor ao disponibilizar funcionalidades de outros serviços, oferecendo uma solução completa de serviços pela criação de serviços compostos e em alto nível, os quais são providos à aplicação cliente. O agregador atua como requisitante na medida em que ele requisita a execução de funcionalidades providas pelos serviços provedores.

2.5.4 Intermediário (*broker*).

O intermediário age entre o solicitante e o provedor de serviços possibilitando a localização de serviços oferecidos pelo provedor e provendo informações adicionais sobre esses serviços, tais como: Confiabilidade, nível de segurança, qualidade do serviço, SLA: contratos entre serviços (que especificam, níveis de disponibilidade, desempenho, operações, valor monetário para usar serviço, penalidades no caso de violação do contrato).

2.6 Características de serviços

[Marks e Bell, 2006] apresentam as seguintes características para serviços:

2.6.1 Granularidade adequada

Serviços devem representar funções, processos ou transações do negócio e encapsular outros componentes ou serviços de granularidade mais fina. Serviços pouco granulares podem trazer problemas de desempenho e baixo índice de reutilização. Serviços muito granulares podem trazer problemas de escopo limitado não estando de acordo com requisitos de múltiplos processos do negócio. Além disso, muitos serviços finos podem trazer problemas de tráfego excessivo de mensagens.

São questões do projeto de serviços: definir a granularidade adequada a fim de resolver o problema do negócio, possibilitar o reuso e permitir que os serviços sejam implementados de um ponto de vista tecnológico.

2.6.2 Contratos de serviços bem definidos

Contrato de serviço especifica o que o serviço faz e como proceder para utilizá-lo. Ele separa a funcionalidade das características tecnológicas. No mundo de *web services*, contratos são definidos pelo documento WSDL (Web Service Description Language) em conjunto com outros metadados, tais como políticas, XML *schema*, semântica de documentos etc.

2.6.3 Acoplamento fraco

Acoplamento fraco é o conceito tipicamente utilizado para tratar requisitos de escalabilidade, flexibilidade e tolerância a falha. O principal objetivo é minimizar dependências a fim de que modificações ou falhas em um serviço cause poucos impactos em outros serviços [Josuttis, 2007].

No projeto de serviços, acoplamento fraco significa projetar serviços de forma que implementações específicas dos serviços possam ser substituídas, modificadas e evoluídas sem corromper os serviços, por exemplo, quebra de comunicação com os clientes.

Acoplamento fraco é um princípio que não depende de ferramenta e nem está relacionado a um *checklist* listando um conjunto de atividades que devem ser executadas. Depende da abordagem que se deseja seguir para definir a quantidade de acoplamento fraco que será introduzida na arquitetura.

A tabela 1 abaixo lista alguns tópicos típicos que devem ser considerados quando se deseja obter baixo acoplamento no seu sistema (trata-se de uma extensão de uma lista publicada em [Krafig *et al.*, 2004]).

Tabela 1 – Possíveis formas de se obter baixo acoplamento em SOA

	Alto acoplamento	Baixo Acoplamento
Conexões Físicas	Ponto a ponto	Via mediador
Estilo de comunicação	Síncrono	Assíncrono
Modelo de dado	Tipos complexos	Apenas tipos simples
Tipos de variáveis	Forte	Fraco
Padrão de interação	Navegação por árvores complexas	Mensagens auto-contidas, centradas no dado
Controle da lógica do processo	Controle central	Controle distribuído
Conexão	Estática	Dinâmica
Plataforma	Dependências fortes de plataforma	Independente de plataforma
Transações	2PC (two_phase commit)	Compensação
Desenvolvimento	Simultâneo	Em tempos diferentes
Versionamento	Atualizações explícitas	Atualizações implícitas

Essa tabela está longe de estar completa, mas é bastante comum para grandes sistemas distribuídos. Apesar de não ser um *checklist*, seria muito estranho se nenhuma dessas formas de baixo acoplamento fossem usadas em uma implementação em SOA.

2.6.4 Capacidade de serem localizados

Serviços devem ser bem projetados e seus contratos devem ser publicados e estarem visíveis para os possíveis clientes poderem acessá-los. Serviços podem ser publicados através de: registros de serviços, repositórios de metadados, subdiretórios, ou uma localização qualquer conhecida.

Após serem publicados, os serviços devem ser notificados aos usuários potenciais (propaganda dos serviços). Eles devem ter clientes (ou consumidores) identificados e padrões de reuso identificados antes de serem criados ou expostos.

Um cliente para qualquer dispositivo, usando qualquer sistema operacional, em qualquer linguagem de programação, pode acessar um serviço SOA a fim de criar um novo processo do negócio.

2.6.5 Durabilidade

Os serviços devem existir ao longo do tempo, sendo mapeados a temas de processos duradouros. Os serviços podem ser alterados, mas o negócio ou tema do processo que o serviço implementa deve persistir. Por exemplo, uma empresa de seguros que deve oferecer indenização por sinistro. Os serviços de negócio para o processo de indenização podem sofrer manutenção, mas a indenização por sinistro sempre existirá.

2.6.6 Composição

Serviços devem ser projetados a fim de permitir que possam ser utilizados por outros serviços. Eles devem permitir seu acoplamento a fluxos de processos orquestrados, não devem manter estados e devem ser atômicos (outras estruturas devem ser utilizadas para manutenção de estado e de contexto).

Novas funcionalidades ou novos padrões podem ser adicionados aos serviços sem quebrar o contrato com os consumidores atuais do serviço, mantendo interoperabilidade e as funcionalidades existentes anteriormente.

2.6.7 Alinhamento ao negócio

Serviços devem representar conceitos do negócio e estar de acordo com as necessidades do negócio como definido na estratégia e no plano do negócio, além de estarem associados aos processos de negócio identificados.

2.6.8 Reuso

Serviços devem ser implementados com reuso claro em processos de negócio. Deve-se estar ciente de que o reuso de um serviço pode levar à utilização do serviço por um novo cliente ou consumidor (não determinado previamente). Deve-se considerar a infra-estrutura a fim de garantir desempenho e tamanho apropriado de hardware e de largura da banda de rede.

2.6.9 Interoperabilidade

A interoperabilidade deve ser garantida pela aplicação de políticas, padrões e outros critérios de projeto durante o ciclo de vida do serviço: identificação de serviço, análise de serviços, projeto, implementação, teste e integração e implantação.

2.7 Política e governança SOA

Governança SOA é um processo contínuo de alinhamento dos objetivos estratégicos, novas oportunidades táticas e uso de experiência adquirida na implantação de iniciativas SOA. Essa governança é essencialmente diferente da de TI porque necessita do envolvimento de gestão de processos e de pessoas ligadas ao negócio da organização [Chepers *et al.*, 2008].

A governança em SOA é essencial para a implantação de SOA. Quanto maior o universo SOA, maior a necessidade de governança, e mais complexos devem ser os papéis e mecanismos relacionados. Governança leva tempo para ser projetada e instalada, e não é fácil de ser garantida, mas sem ela, todo projeto estará em risco. [Malinvero, 2006].

2.7.1 Definição de Governança

[Manes, 2007] define governança como sendo aquilo que garante que as pessoas façam que é certo. [Manes, 2007] define as seguintes bases para governança:

- Políticas definem o que é certo;
- Processos obrigam o uso das políticas;
- Métricas provêem visibilidade e validação das obrigatoriedades políticas;
- Organização deve estabelecer uma cultura que dê suporte ao processo de governança.

Não há dúvidas de que a governança SOA foca inicialmente em aspectos não técnicos. Em determinado momento torna-se necessário envolver-se com detalhes técnicos sobre a infra-estrutura (por exemplo, ESB) e ferramentas (tais como repositórios), mas as considerações sobre esses detalhes devem ser embasadas por informações que deixem claro o porquê de usá-los, quando usá-los e quem deveria usá-los.

A governança em SOA pode ser definida como uma tarefa que trata das seguintes questões não técnicas:

- **Modelos de visões, objetivos, caso de negócio e financeiros**: Os modelos de visões, objetivos e caso de negócio são questões que introduzem o por quê de se usar SOA. Mas é também importante decidir um modelo financeiro que determine como as despesas iniciais para a introdução de SOA e dos novos serviços serão pagas.
- **Arquitetura de referência**: É necessária a definição de uma arquitetura de referência que demonstre as decisões fundamentas referentes à arquitetura, incluindo a tecnologia escolhida, padrões de troca de mensagem, meta-modelos.
- **Papéis e responsabilidades**: É necessário saber quem se preocupa e quem é responsável por que questões. Por exemplo, é preciso deixar claro quem toma as decisões de arquitetura de SOA, onde a responsabilidade do ESB termina e

de quem é o dever de criar soluções de projeto de alto nível para identificação de novos serviços.

- **Políticas, padrões e formatos**: Decisões sobre arquitetura, artefatos e tecnologias levam a políticas e requerimentos que devem ser definidos utilizando-se formatos padrões e proprietários.
- **Processos e ciclo de vida**: Atrelado a papéis, responsabilidades e políticas devem ser definidos processos e ciclos de vida para as soluções e serviços.

Aspectos técnicos de governança também devem ser considerados a fim de ajudar a dar suporte aos aspectos não técnicos. Entre eles incluem-se:

- **Documentação**: Ajudando a promover as questões não técnicas de governança, tais como, processos, responsabilidades, políticas etc.
- **Gerenciamento de serviços**: Nesse caso, ferramentas como repositórios e registros devem ajudar.
- **Monitoramento**: Para verificação de regras, políticas e contratos. Além disso, pode-se verificar quais serviços não estão mais em uso.
- **Gerenciamento de mudança e configuração**: As ferramentas de gerenciamento de configuração podem ajudar na gerência de softwares de SOA, artefatos de SOA e documentações de SOA.

Ferramentas de gerência de configuração e monitoramente podem auxiliar muito em um projeto SOA. Isto é explicitado pelo fato de muitos fornecedores de ferramenta focarem nestas características técnicas. Todavia, existem muito mais aspectos para governança SOA do que o que é apresentado nesta lista.

2.7.2 Papéis em Governança SOA

[Josuttis, 2007] apresenta que governança SOA exige a criação de equipes formadas por pessoas que sejam capazes de coordenar todos os passos da implementação de SOA e suas estratégias.

Normalmente existe um time central responsável por isso. Esse time pode ser chamado, por exemplo, de SCC (*SOA Competency Center* – Centro de Competência de SOA), SCoE (*SOA Center of Excellence* – Centro de Excelência de SOA) ou ainda ICC (*Integration Competency Center* – Centro de Competência Integrada), o qual pode ser direcionado por um comitê que encontra, controla e decide sobre as estratégias SOA como um todo. As principais funções dessa equipe são: coordenar, consolidar, padronizar e controlar a estratégia como um todo (em alguns casos).

Além disso, uma equipe virtual formada por desenvolvedores, especialistas do negócio e especialistas de TI pode ser criada para discussão e disseminação dos aspectos tecnológicos da governança SOA.

É importante garantir que todas as pessoas envolvidas em qualquer nível da governança SOA tenham tempo e recursos para resolver questões envolvidas à suas atividades de governança.

2.7.3 Tecnologias para governança SOA

O mercado para tecnologias de governança SOA está apenas iniciando o seu desenvolvimento. Entre fusões e aquisições no mercado de governança SOA, fornecedores estão tentando assumir a liderança em todos os aspectos da governança SOA.

Uma tecnologia de sucesso para governança SOA deve incluir as seguintes características, como ilustrado na Figura 1:

- **Gerenciamento da política SOA:** provê a tecnologia para criar, descobrir, referenciar e, algumas vezes, forçar políticas relacionadas a instrumentos SOA, como controle de acessos, desempenho e níveis de serviços.
- **Registro / Repositório SOA:** ajuda a gerenciar os metadados relacionados a artefatos SOA (por exemplo, serviços, políticas, processos e perfis). Recentemente esta característica tem evoluído a ponto de permitir a criação e documentação dos relacionamentos (configurações e dependências) entre vários metadados e artefatos.
- **Garantia e validação da qualidade SOA**: valida os artefatos SOA individualmente e determina o relacionamento entre eles, no contexto de uma distribuição SOA. Por exemplo, estas tecnologias irão testar e validar uma composição de serviços que executa processos específicos, garantindo que estejam de acordo com políticas específicas para esse tipo de composição.

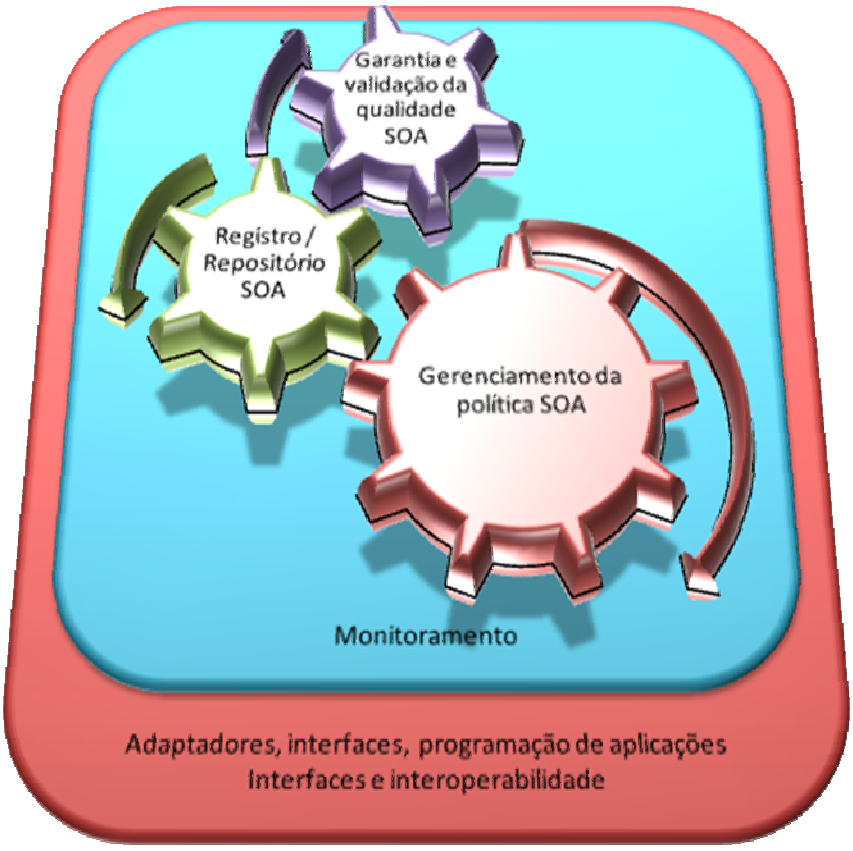

Figura 1 - Suite de tecnologias de governança SOA

2.8 Padrões

As pessoas tendem a pensar e se comunicar em padrões. Christopher Alexander, autor de vários livros sobre linguagens de padrões, definiu padrões como "uma abstração de uma forma concreta que continua se repetindo em contextos específicos, não-arbitrários" [Microsoft, 2005].

Os padrões e as linguagens de padrões são formas de descrever melhores práticas, projetos testados e capturar experiências passadas de forma que os outros aprendam com essas experiências. Os padrões são abordagens testadas para entender rapidamente as diretrizes de projeto e os vários contextos em que devem ser aplicadas.

Os potenciais benefícios de SOA são claros, como a reutilização dos ativos existentes, mas o panorama dos padrões ainda é nebuloso. Em seu estudo mais recente sobre o assunto, a Forrester Research contabilizou cerca de 115 padrões flutuando ao redor de SOA e *web services*. Descobriu também que é quase impossível confirmar quais fornecedores apóiam quais padrões [Violino, 2008].

A existência de diversos padrões relacionados à SOA mostra que a indústria de software ruma para a ampla adoção de SOA. Executivos de tecnologia e especialistas da indústria aconselham o monitoramento atento do cenário dos padrões até que eles realmente amadureçam. Porém, aconselham também que não se fique restrito às opções de padrões existentes e que não se adie os projetos SOA essenciais.

Padrões como SOAP e WSDL, por exemplo, já foram amplamente adotados, e outros, incluindo WS-Security, estão prontos para adoção disseminada, diz Randy Heffner, analista da Forrester Research. Mas outras especificações necessárias para criar *web services* que operam com alta qualidade de serviço (como os padrões para gerenciamento, transações e segurança avançada) só amadureceram o suficiente para usuários agressivos de tecnologia, segundo Heffner [Heffner, 2007].

O ideal é a utilização de SOA baseada em padrões ao invés de protocolos nativos. Porém, deve-se ficar atento para que não se sacrifique a necessária qualidade de serviço (QoS) para uma determinada aplicação só para usar padrões.

Muitas organizações buscam soluções temporárias — *middleware*, por exemplo — para superar a falta de padrões amadurecidos. Elas devem ficar atentas para não ficarem presas a essas plataformas, utilizando *middleware* não intrusivos, pois depois o processo de mudança será muito mais difícil.

2.9 Web services

Os *web services* podem ser definidos como programas modulares, geralmente independentes e auto-descritivos que podem ser localizados e invocados através da internet ou de uma intranet corporativa. Eles tipicamente são construídos com especificações de XML, SOAP, WSDL e UDDI [W3C, 2004].

Web services oferecem padrões de desenvolvimento para implementar funções de negócios que possam ser invocadas remotamente. Atualmente, a maioria das companhias de software estão implementando ferramentas baseadas nesses novos padrões [IBM, 2000]. Considerando a velocidade com que novos padrões estão sendo criados e disponibilizados, graças ao forte empenho de várias companhias de software, é provável que em pouco tempo eles sejam tão populares quanto é o HTML hoje.

A plataforma básica dos *web services* é HTML com XML. O HTML é o protocolo mais utilizado na internet. O XML provê uma linguagem que pode ser utilizada em

diferentes plataformas e linguagens de programação e mesmo assim disponibilizar mensagens e funções complexas. Os elementos padrão dos *web services* são SOAP, UDDI e WSDL.

A arquitetura de *web services* envolve muitas camadas e inter-relacionamento de tecnologias. Existem muitas maneiras de visualizarmos essas tecnologias, pois existem muitas maneiras de se construir e se usar s *web services*. A Figura 2 mostra uma ilustração de algumas famílias dessas tecnologias.

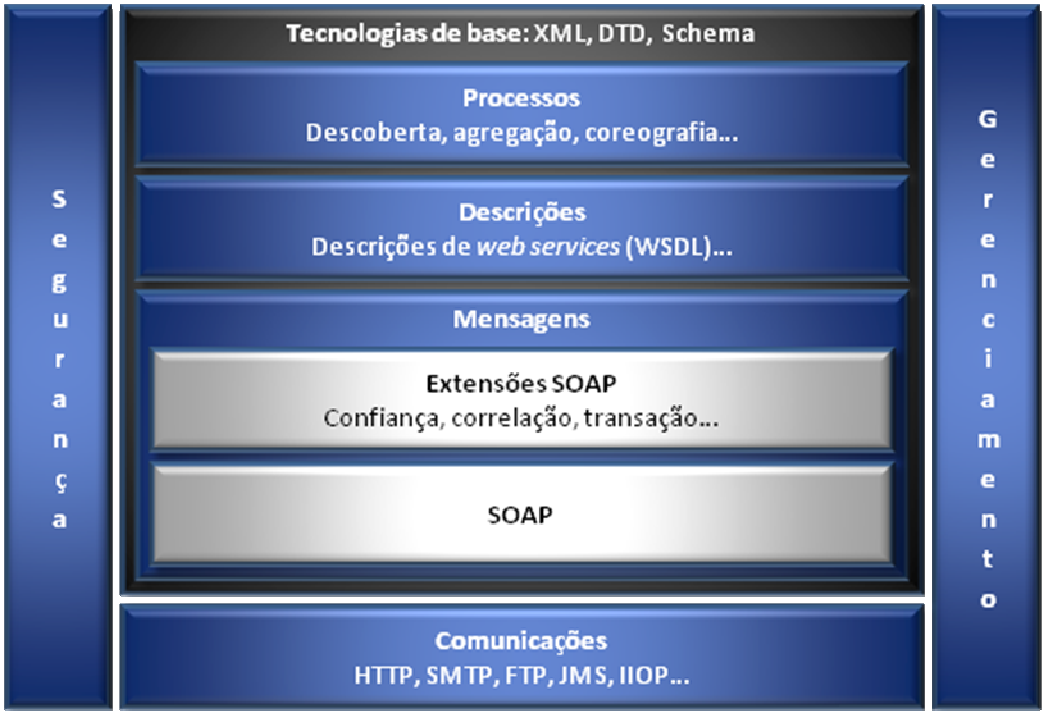

Figura 2– Arquitetura de web services

São necessários quatro passos para se empregar os *web services*, de acordo com a Figura 3. Embora eles sejam necessários, possivelmente eles não serão suficientes: muitos cenários necessitarão de passos adicionais, ou refinamentos significativos desses passos. Além disso, a ordem em que os passos são executados pode variar de situação a situação [W3C, 2004].

- Passo 1 - O solicitante de serviço e o provedor de serviço tomam conhecimento um do outro. No caso típico, o solicitante iniciará a comunicação. Logo, ele deverá, de alguma forma, tomar conhecimento do endereço do provedor. Ele pode obter esse endereço diretamente com o próprio provedor ou ele pode usar um serviço de descoberta para localizar as descrições dos serviços disponíveis, as quais contêm o endereço dos provedores disponíveis.
- Passo 2 – O solicitante e o provedor de serviços concordam com a descrição do serviço (um documento WSDL) e semânticas que irão governar a interação entre eles. Isso não significa que eles deverão se comunicar um com o outro. Isso também pode ocorrer de outras formas, como, por exemplo, o provedor publica a descrição e a semântica do serviço como um contrato e o solicitante aceita ou não, mas não poderá modificá-las. Ou então, a descrição e a semântica dos serviços estão definidas como um padrão por uma organização e são utilizadas por muitos solicitantes e provedores e não podem ser alteradas por nenhum dos dois lados.

- Passo 3 – A descrição e semântica dos serviços são associadas apropriadamente aos agentes em ambos os lados (tanto no solicitante como no provedor do serviços).
- Passo 4 – Os agentes solicitante e provedor de serviço trocam mensagens SOAP.

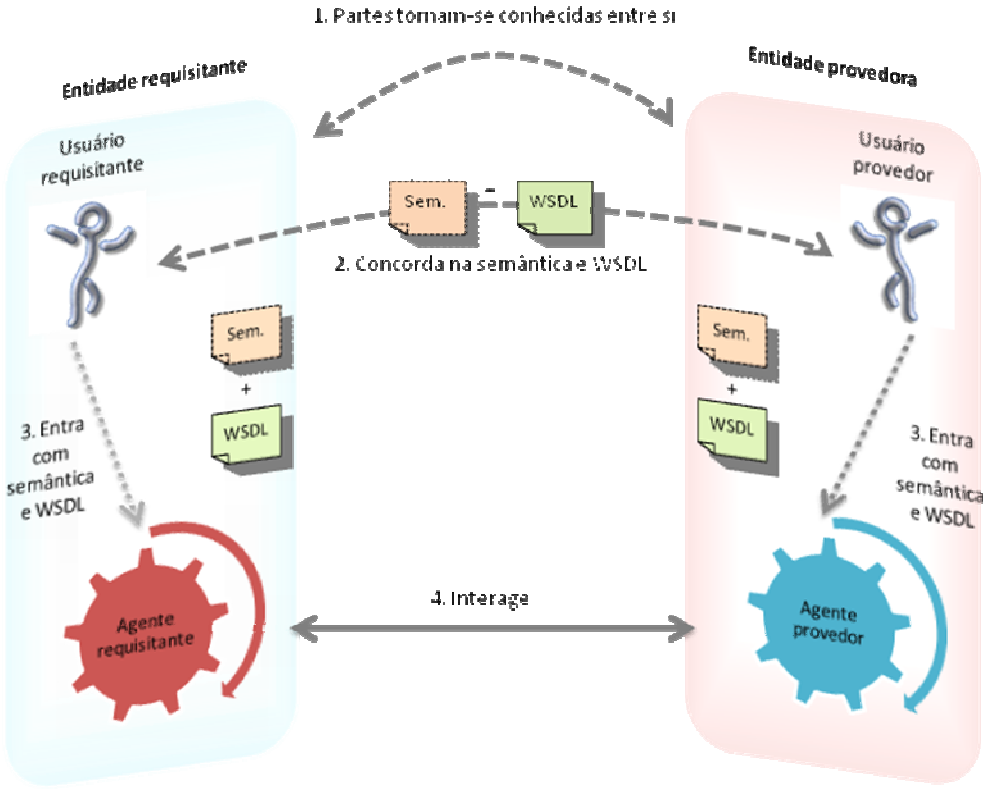

Figura 3– Processo genérico de se empregar os *web services*

Em suma, é conveniente dizer que o solicitante e o provedor de serviços devem concordar com a semântica e descrição que irão governar a interação entre eles. Porém, seria mais correto dizer que eles simplesmente precisam ter uma visão congruente e não conflitante das semânticas e descrições durante a interação.

A plataforma dos *web services* é simples, interoperável, e baseada em troca de mensagens. Mas, alguns aspectos ainda estão pendentes, como segurança, privacidade e roteamento. Esses aspectos estão sendo cobertos à medida que SOA está avançando (W3CSCHOOLS).

2.10 WSDL

O WSDL (Web Services Description Language – Linguagem de Descrição de Serviços Web) é uma linguagem baseada em XML utilizada para descrever Web Services. Trata-se de um documento escrito em XML que além de descrever o serviço, especifica como acessá-lo e quais são as operações ou métodos disponíveis.

O WSDL é dividido nas seguintes camadas:

- Descrição de interface do serviço: a interface pode consistir de uma ou mais operações, contendo parâmetros de entrada e saída;

- Conexão do serviço: protocolo e formato nos quais as operações dos serviços são providas;
- Localização física (endereço, URL): onde o serviço está disponível.

A Figura 4 apresenta um exemplo de um arquivo WSDL o serviço *CustomerService*. Esse arquivo define um serviço chamado CustomerService, que provê uma operação chamada getCustomerAddress(). Seu parâmetro de entrada é o ID do cliente (do tipo long), e a saída é o endereço do cliente com atributos a rua, a cidade e o CEP do tipo string.

Essa é uma versão WSDL 1.1:

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <definitions name="CustomerService"
3 targetNamespace="http://soa-in-practice.com/wsdl"
4 xmlns:tns="http://soa-in-practice.com/wsdl"
5 xmlns:xsd1="http://soa-in-practice.com/xsd"
6 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
8 xmlns="http://schemas.xmlsoap.org/wsdl/">
9
10 <types>
11 <xsd:schema
12 targetNamespace="http://soa-in-practice.com/xsd"
13 xmlns="http://soa-in-practice.com/xsd">
14
15 <xsd:element name="getCustomerAddress">
16 <xsd:complexType>
17 <xsd:sequence>
18 <xsd:element name="customerID" type="xsd:long"/>
19 </xsd:sequence>
20 </xsd:complexType>
21 </xsd:element>
22
23 <xsd:element name="getCustomerAddressResponse" type="Address"/>
24 <xsd:complexType name="Address">
25 <xsd:sequence>
26 <xsd:element name="street" type="xsd:string"/>
27 <xsd:element name="city" type="xsd:string"/>
28 <xsd:element name="zipCode" type="xsd:string"/>
29 </xsd:sequence>
30 </xsd:complexType>
31
32 </xsd:schema>
33 </types>
```

```
34
35 <message name="getCustomerAddressInput">
36 <part name="params" element="xsd1:getCustomerAddress"/>
37 </message>
38 <message name="getCustomerAddressOutput">
39 <part name="params" element="xsd1:getCustomerAddressResponse"/>
40 </message>
41
42 <portType name="CustomerInterface" >
43 <operation name="getCustomerAddress">
44 <input message="tns:getCustomerAddressInput" />
45 <output message="tns:getCustomerAddressOutput" />
46 </operation>
47 </portType>
48
49 <binding name="CustomerSOAPBinding"
50 type="tns:CustomerInterface" >
51 <soap:binding style="document"
52 transport="http://schemas.xmlsoap.org/soap/http" />
53 <operation name="getCustomerAddress">
54 <soap:operation
55 soapAction="http://soa-in-practice.com/getCustomerAddress" />
56 <input>
57 <soap:body use="literal" />
58 </input>
59 <output>
60 <soap:body use="literal" />
61 </output>
62 </operation>
63 </binding>
64
65 <service name="CustomerService" >
66 <port name="CustomerPort"
67 binding="tns:CustomerSOAPBinding">
68 <soap:address
69 location="http://soa-in-practice.com/customer11"/>
70 </port>
71 </service>
72
73 </definitions>
```

Figura 4– Exemplo de WSDL para o serviço *CustomerService*

Analisando esse exemplo de baixo pra cima nós temos:

- A seção <service> no final (linhas 65 a 71) define um serviço chamado CustomerService, disponível na URL http://soa-in-practice.com/customer11, e que é provido por uma conexão chamada CustomerSOAPBinding. O prefixo tns: é o *namespace* onde podemos encontrar detalhes sobre o identificador. Isso é definido no início do documento e tem o mesmo nome do *namespace* alvo, o qual é o *namespace* que todos os identificadores identificam nesse arquivo (veja linhas 3 e 4). Isso significa que CustomerSOAPBinding é um identificador definido nesse arquivo.
- A seção <binding> (linhas 49 a 63) define o protocolo e formato que são usados para prover o serviço. Aqui está a definição de CustomerSOAPBinding. Ele começa identificando para qual tipo de interface é essa conexão (CustomerInterface). Sem entrar em detalhes, a conexão especificada aqui é uma conexão SOAP baseada no protocolo HTTP de baixo nível (veja linhas 51 e 52). A seção também define qual estilo de SOAP é usado (é possível mais de um tipo de conexão SOAP).
- A seção <portType> (linhas 42 a 47) define a interface CustomerInterface. Ele consiste de uma operação, chamada getCustomerAddress, e especifica a mensagem que será enviada pelo barramento de serviço quando a operação for chamada. Esse serviço envia uma requisição e recebe uma resposta, as quais são definidas pela especificação de uma mensagem de entrada (getCustomerAddressInput) e uma especificação de mensagem de saída (getCustomerAddressOutput). Outras mensagens poderiam ser mensagens de erro.
- As seções <message> (linhas 35 a 37 e 38 a 40) definem as mensagens individuais, usando os identificadores referenciados na seção <portType>. Tanto getCustomerAddressInput quanto getCustomerAddressOutput usam os tipo de dados definidos na seção <types>.
- A seção <types> (linhas 10 a 33) definem os tipos de dados usados: nesse caso, o parâmetro de entrada customerID do tipo long e o parâmetro de saída address, o qual é uma estrutura de árvore com atributos do tipo string. Todos os tipos têm seu próprio *namespace*, xsd1.

A seção <binding> informa que o estilo SOAP usado é document/literal, o qual é uma combinação do atributo de estilo na linha 51 e as definições no corpo nas linhas 57 e 60.

2.11 SOAP

SOAP (*Simple Object Access Protocol* – Protocolo Simples de Acesso a Objeto) é independente de plataforma e independente de implementação. Permite baixo acoplamento entre requisitante e provedor e permite comunicação entre serviços de diferentes organizações.

SOAP é uma possibilidade para implementação do paradigma SOA, porém não são obrigatórios. Ao invés de SOAP, por exemplo, organizações utilizam Websphere MQ da IBM para troca de documentos XML, CORBA, TIBCO.

Em uma requisição SOAP o serviço "Listener" aceita a mensagem SOAP, extrai o XML do corpo da mensagem, transforma a mensagem XML em um protocolo nativo,

delega a requisição ao processo de negócio corrente dentro da organização e retorna a resposta também em formato XML.

As versões atuais em que se encontra o SOAP são 1.1 e 1.2, mas elas não são muito diferentes entre si. As especificações são principalmente relevantes para os adaptadores SOAP e não são visíveis para os programadores de *web services*.

A Figura 5 apresenta um exemplo de solicitação SOAP de acordo com a especificação do arquivo WSDL da Figura 4 (usando o estilo *document/literal*).

```
<?xml version='1.0' ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  ...
  </soap:Header>
  <soap:Body>
    <getCustomerAddress xmlns="http://soa-in-practice.com/xsd">
      <customerID>12345678</customerID>
    </getCustomerAddress >
  </soap:Body>
</soap:Envelope>
```

Figura 5– Exemplo de solicitação SOAP

Uma mensagem de resposta correspondente poderia ter o conteúdo apresentado na Figura 6.

```
<?xml version='1.0' ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
  ...
  </soap:Header>
  <soap:Body>
    <getCustomerAddressResponse xmlns="http://soa-in-
practice.com/xsd">
      <address>
        <street>Gaussstr. 29</street>
        <city>Braunschweig</city>
        <zipCode>D-38106</zipCode>
      </address>
    </getCustomerAddressResponse>
  </soap:Body>
</soap:Envelope>
```

Figura 6 – Exemplo de respostas SOAP

Nos exemplos da Figura 5 e da Figura 6 observa-se que a mensagem SOAP é descrita em formato XML e contém um elemento raiz chamado <Envelope>. Além disso, a mensagem SOAP pode possuir um elemento opcional <Header> e um obrigatório <Body>. Enquanto o elemento body contém a informação útil (requisição, resposta, ou dados padrão), o header pode conter informações adicionais que ajudem a

infra-estrutura a tratar a mensagem (tais como sugestões de roteamento, segurança e muitos outros).

2.12 REST (Representational State Transfer)

O estilo REST, também conhecido como RESTful http, especifica uma série de restrições de arquitetura com o objetivo de melhorar o desempenho, escalabilidade e abstração de recursos em sistemas distribuídos. Refere-se a uma coleção de princípios de arquiteturas de rede que foca em acesso a recursos de formas simples e sem manutenção de estado (stateless) [Josuttis, 2007].

Os princípios de arquitetura incluem [Hansen, 2007]:

- Serviços RESTful são auto-contidos, ou seja, cada requisição de um cliente ao servidor deve conter toda a informação necessária para entender a requisição, e não pode se beneficiar de qualquer contexto armazenado no servidor.
- Serviços RESTful possuem interface uniforme. Isso significa os recursos possuem a mesma interface que o cliente e que as únicas operações permitidas são operações HTTP: GET, POST, PUT e DELETE.
- Arquiteturas baseadas em REST são construídas com recursos unicamente identificados pelas URIs. Por exemplo, em um sistema RESTful de compras, cada compra possui uma URI única.
- Componentes REST manipulam recursos através da troca de representações de recursos. Por exemplo, um recurso de compra pode ser representado por um documento XML. Em um sistema de compras RESTful, uma compra poderia ser atualizada através do envio de um documento XML contendo a alteração da compra para a sua URI.

No modelo REST os únicos métodos permitidos são GET, POST, PUT e DELETE.

Esses são os conceitos básicos por trás de REST. Entretanto, quando as pessoas falam sobre os benefícios de sistemas RESTful hoje em dia, elas não estão aplicando estritamente esses princípios. Por exemplo, manter os dados de cartão de compras no servidor e manter a sessão relacionada a esse processo que está usando o cartão de compras é aceitável, embora armazenar informações de sessão ou dados de cartões de compras no servidor esteja claramente violando os conceitos de REST, que diz que o serviço deve ser auto-contido).

Desvios mais significativos dos princípios de REST envolvem a restrição de "interface uniforme" através do embutimento de métodos e parâmetros dentro das URIs. Algumas interfaces já não têm mais uma única URI para cada recurso. Esses tipos de sistemas, embora nomeados como RESTful, parecem com RCP usando XML sobre HTTP.

A tabela 2 abaixo ilustra as principais diferenças entre *web services* com REST e *web services* com SOAP. REST usa XML sobre HTTP sem uma definição de interface WSDL.

Tabela 2 – Web services RESTful contra web services SOAP

	REST	SOAP
Formato da mensagem	XML	XML em um envelope SOAP
Definição da interface	Nenhuma	WSDL
Transporte	HTTP	HTTP, FTP, MIME, JMS, SMTP, etc.

2.13 Segurança

Em SOA, os aspectos e técnicas mais comuns para sistemas distribuídos são [Josuttis, 2007]:

- SOA força a uma alta interoperabilidade, o que prejudica a segurança padrão. Frequentemente isso é uma motivação para introduzir os conceitos de segurança no ESB.
- SOA tem que lidar com conceitos de segurança heterogêneos dos sistemas existentes.
- Processos distribuídos transferem dados sobre múltiplas camadas, logo soluções ponto a ponto não são suficientes para a segurança. Ao invés disso, é necessária uma segurança fim a fim. Normalmente é preciso segurança na camada de mensagem, o que significa lidar com segurança dentro da mensagem.
- Capacitações de multi-clientes força uma verificação de segurança em tempo de execução.

Os requerimentos de segurança podem ser categorizados da seguinte maneira:

- Autenticação: Verificação de identidade, podendo essa ser um usuário, uma fonte física ou um solicitante de serviço externo.
- Autorização: Determinação se o serviço solicitante pode chamar e/ou ver os resultados da execução do serviço.
- Confidenciabilidade: Assegurar que ninguém, além do solicitante do serviço, possa ver os dados do serviço enquanto eles estão sendo transferidos.
- Integridade: Garantir que os dados não sejam manipulados, nem suas autenticações ou autorizações.
- Disponibilidade: Bloquear ataques que tornem os dados indisponíveis.
- Auditoria: Inclui monitoração, *log* e rastreamento de todo o fluxo relevante de segurança.

Deve ser definida e implementada uma abordagem de segurança estratégica que cubra intra-estrutura, arquitetura e aplicações. A melhor abordagem é introduzir a segurança como um serviço.

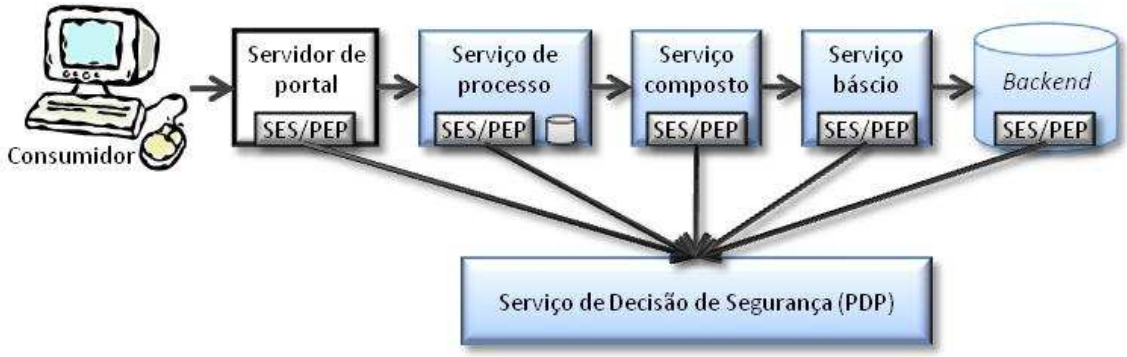

Figura 7 – Segurança via serviço

A Figura 7 ilustra a idéia de como introduzir um componente ou sistema central que gerencia identidades e suas regras e perfis associados. Isso normalmente é chamado de provedor de identificação (IdP). Ele provê serviços que tomam decisões de segurança.

Existem vários padrões para XML e *web services*. Esses padrões não acrescentam novas tecnologias ou procedimentos de segurança, ao invés disso, definem como aplicar tecnologias e procedimentos existentes no momento da troca de arquivos XML ou via *web services*.

Para XML existem os seguintes padrões: *XML Signature*, *XML Encryption*, *XML Key Management* [Josuttis, 2007].

Para os *web services*, os padrões mais importantes são:

- *WS-Security*: Define como aplicar diferentes técnicas para autorização, integridade e privacidade com os *web services* (SOAP). Descreve uma forma padrão de embutir informações de segurança, tais como palavras chave, criptografias, assinaturas, etc, num cabeçalho SOAP [WS-SECURITY, 2004].
- *WS-SecurityPolicy*: Permite encontrar quais padrões de segurança um provedor de serviço precisa e/ou suporta.
- *WS-Trust*: Permite publicar, renovar e validar palavras chaves de segurança.
- *WS-Federation*: Define mecanismos que permitem a integração e a federação de diferentes domínios de segurança
  permitindo e bloqueando a confiança de identidades, atributos, e autenticações.

A segurança influi no desempenho (chamadas a serviços adicionais de decisão, criptografia e decriptografia de dados, validação de integridade, verificação de disponibilidade, etc), portanto é necessário encontrar um equilíbrio entre o risco e o desempenho. O recomendado é manter aspectos de segurança desde o começo, o que não significa que seja necessário tentar lidar com todos os aspectos de segurança imediatamente.

3 Enterprise Service Bus

Enterprise Service Bus (ESB) corresponde à infra-estrutura utilizada para disponibilizar serviços de diferentes sistemas em diversos ambientes computacionais que estão distribuídos em uma rede.

O ESB é um barramento de mensagens baseado em padrão aberto, projetado para possibilitar a implementação, desenvolvimento e gerenciamento de soluções baseadas em SOA com o foco no empacotamento, desenvolvimento e gestão de serviços distribuídos [Papazoglou *et al.*, 2007]. É responsabilidade do ESB permitir que consumidores (clientes) invoquem os serviços oferecidos por provedores de serviços, o que envolve as seguintes tarefas:

- Prover conectividade
- Transformação de dados
- Roteamento inteligente de mensagens
- Tratar segurança
- Tratar confiabilidade

- Gerência de serviços
- Monitoramente e *log* de serviços

Estas tarefas devem ser consideradas para diferentes plataformas de software e hardware para diferentes *middleware* e protocolos.

O principal papel do ESB é prover interoperabilidade. Como o ESB integra diferentes plataformas e linguagens de programação, uma parte fundamental deste papel é o de transformação de dados. Uma abordagem usual é introduzir um formato específico para o qual todas as plataformas e APIs específicas são mapeadas. O ESB agrupa aplicações e componentes para criar composição de serviços e formar processos de negócio compostos, o que por sua vez, automatiza funções de negócio na organização.

A Figura 8 descreve uma arquitetura simplificada de um ESB que integra uma aplicação J2EE usando JMS (Java Message Services), uma aplicação .NET em um cliente C#, uma aplicação MQ (Message Queue) que tem interface com aplicações legadas, aplicações externas e fontes de dados sendo acessadas via Serviços Web em um módulo de consultas distribuídas. O ESB provê uma integração mais eficiente entre diferentes componentes de aplicações. Isso é possível, pois o ESB posiciona os componentes atrás de uma fachada orientada a serviço, utilizando, por exemplo web services. Na Figura 8, o componente o módulo de consultas distribuídas é usado para abstrair a complexidade de fontes de dados heterogêneas. Todos os pontos finais apresentados na figura provêem abstração de destino físico e informações de conexão. Um portal na Figura 8 é um ponto de interface com o usuário de agregação de variados recursos representados como serviços, por exemplo, vendas, funcionalidades departamentais, recursos humanos, além de portais de parceiros do negócio.

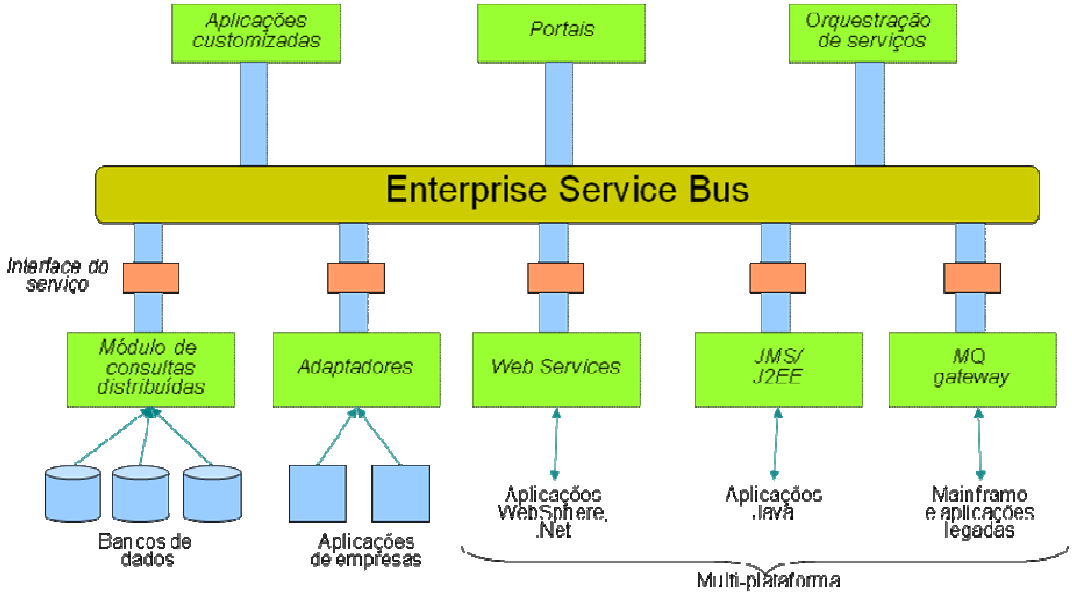

Figura 8 – Exemplo de Enterprise Service Bus

É importante definir quais partes da arquitetura serão de responsabilidades por quais tarefas, e quais equipes serão responsáveis por prover e manter estas partes. Por exemplo, deve-se deixar claro se o ESB deverá prover tecnologia para lidar com processos de negócio, tal como BPEL ser parte do ESB para orquestração de serviços. Por outro lado, pode-se concluir que BPEL é apenas uma plataforma específica que ajuda a prover serviços compostos e, portanto, não deveria ser considerado como parte do ESB.

O ESB permite a comunicação usando nomes lógicos, os quais são mapeados em destinos físicos da rede, em tempo de execução. A independência de destino permite atualizar (upgrade), mover ou substituir serviços sem ter que modificar o código ou corromper aplicações existentes no ESB. Além disso, a duplicação de processos pode ser utilizada para tratar falhas se um serviço não está disponível, sendo tarefa do ESB rotear a solicitação para o serviço disponível.

Conceitualmente ESB combina tecnologias de EAI, web services, XSLT, orquestração de serviços e coreografia de serviços.

Os aspectos básicos para SOA em um ESB são:

- Disponibilização de serviço: cada aplicação deve ser exposta como um serviço;
- Orquestração de serviço: serviços distribuídos podem ser configurados e orquestrados em processos de negócio unificados e claros;
- Implantação: disponibilização do serviço para o ambiente de produção, levando em consideração aspectos de segurança, confiabilidade e escalabilidade;
- Gerência de serviços: serviços executando no ESB podem ser monitorados, auditados, mantidos e reconfigurados. No último caso, mudanças no processo podem ser feitas sem necessidade de reescrever serviços ou aplicações subjacentes, dependendo das modificações necessárias e dos serviços existentes.

4 Ciclo de vida de serviços

Serviços são "pedaços" de software, como outro software qualquer, logo o ciclo de vida usual para desenvolvimento de software também se aplica. Assim como sabemos que um ciclo de vida em cascata não deve ser utilizado. Por outro lado, um ciclo de vida iterativo e incremental será mais adequado.

Os passos de um ciclo de vida de desenvolvimento de serviços são (Figura 9):

- Identificação de serviços de negócio
- Análise de serviços de negócio
- Implementação
- Integração e testes
- Implantação

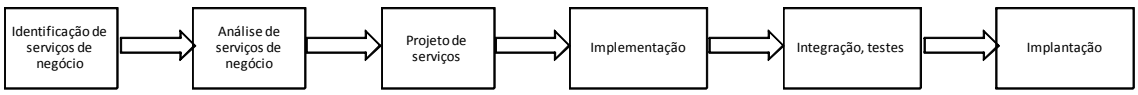

Figura 9 – Passos de um ciclo de vida de desenvolvimento de serviços

Durante o desenvolvimento de serviços, assim como ocorre no desenvolvimento de software, mudanças podem ocorrer. A necessidade de mudanças resulta de erros ou enganos cometidos, ganho de experiência e maior conhecimento do domínio que levam à necessidade de melhoria de artefatos já produzidos, ou mesmo devido a mudanças do negócio. Portanto, o processo de desenvolvimento de serviços deve ser iterativo.

Um serviço é utilizado em um processo de negócio para executar um requisito de negócio automatizado. Uma das características importante de serviços é a capacidade de ser reutilizado, podendo ser parte de uma ou mais atividades de processos distin-

tos, por exemplo. Logo, a mudança de um serviço para atender às necessidades de um processo de negócio pode impactar a execução dos outros processos que utilizam aquele serviço.

A ligação entre processos de negócio e serviços é feita através de suas interfaces. A princípio, uma interface bem projetada reduz os riscos de retrabalho devido a modificações futuras. Contudo devemos aceitar que serviços podem mudar, por mais experiente que seja o responsável pelo projeto do serviço. Mudanças podem ocorrer por falta de algum conhecimento nas fases iniciais do projeto. [Josuttis, 2007] apresenta que menos de 50% das interfaces projetadas mantêm-se estáveis durante a fase de implementação.

Em SOA temos a combinação de serviços novos sendo desenvolvidos e serviços existentes sofrendo manutenção. Modificações em serviços são mais críticas eles possuem diferentes proprietários e são utilizados em diferentes processos de negócio. Além disso, deve-se considerar que em produção podem estar executando processos críticos. Logo, modificar um serviço (interface ou implementação) pode trazer problemas. A manutenção do serviço deve ser feita com cautela.

A boa prática é de serviços em produção serem estáveis. Sempre que for necessário alterar o comportamento de um serviço, a melhor abordagem é criar um serviço novo ou criar uma nova versão do serviço – independente da anterior. Uma nova versão do serviço, na realidade, corresponde a um novo serviço. Todavia, o que fazer quando uma nova versão de um serviço é necessária para corrigir um erro? Neste caso, a solução é substituir a versão do serviço existente pela nova implementação. Esta substituição deve ser feita, entretanto, se a correção não impactar a interface do serviço. Caso a interface do serviço seja impactada, é melhor criar um novo serviço.

Outro caso a ser considerado é o tratamento de serviços obsoletos (como apresentado na Figura 10, que não mais são necessários. Estes serviços devem ser removidos de produção, aposentados. O primeiro passo para a remoção de um serviço obsoleto é marcá-lo como obsoleto ("deprecated"), a fim de indicar que o serviço será removido e que não é aconselhável utilizá-lo. Em seguida, monitora-se o uso do serviço, com o intuito de avaliar o uso do serviço. Este monitoramento pode ser feito pelo barramento de serviços (ESB) ou por um código implementado com esta finalidade. Se o serviço não estiver mais sendo utilizado, então ele pode ser removido. Caso contrário, deve-se definir um cronograma para aposentadoria do serviço e comunicar este cronograma para todos os clientes do serviço, aqueles que podem ser impactados.

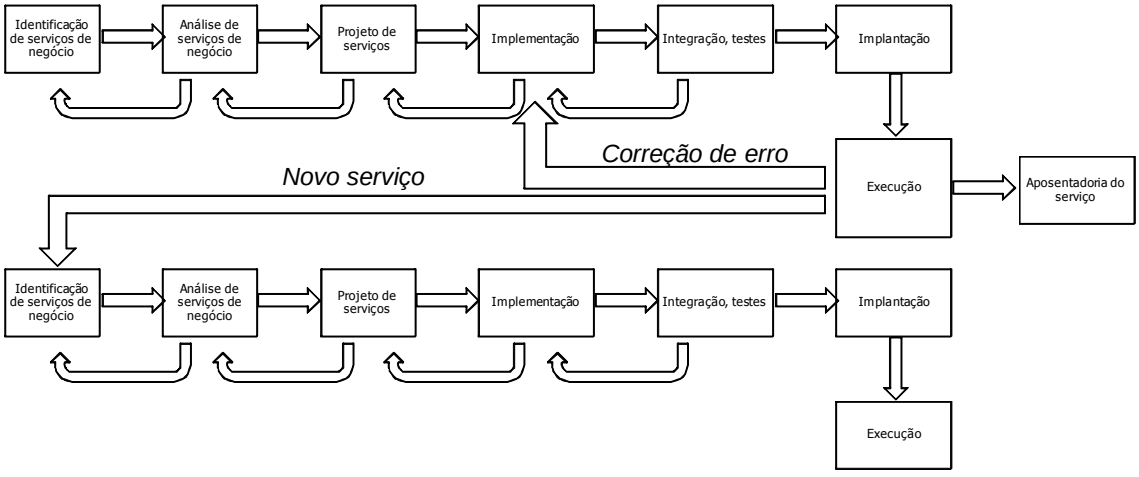

Figura 10 – Passos do ciclo de vida para desenvolvimento de serviços explicitando correção de erro, implementação de uma nova versão do serviço (novo serviço) e

5 Business Process Management e SOA

Existe uma grande relação entre os conceitos Business Process Management e Service Oriented Architecture. Muitos autores indicam que ambas as iniciativas devem ser consideradas em conjunto. Todavia, pode-se ter BPM sem SOA ou SOA sem BPM. Por outro lado, possuir os processos modelados quando da implantação de uma SOA pode trazer grandes benefícios.

Esta seção trata dos conceitos de BPM e da sua importância para uma SOA.

5.1 Definição de BPM

Existe uma redundância no significado da sigla BPM. Em geral ela significa gerenciamento de processos de negócio (*Business Process Management*), que diz respeito a todas as atividades referentes ao gerenciamento e aprimoramento dos processos de negócio de uma organização. Segundo [Bloomberg e Schmelzer, 2006], gerenciamento de processos de negócio normalmente se refere aos meios tecnológicos para a organização adquirir a visibilidade e o controle sobre os processos de negócios que envolvem sistemas e pessoas em uma ou mais organizações.

Entretanto, existe outro significado usado para BPM, que é o de modelagem de processo de negócios (*Business Process Modeling*), que abrange apenas as atividades referentes à modelagem dos processos de negócio. [Bloomberg e Schmelzer, 2006] definem esse segundo significado como o conjunto de práticas ou tarefas que as organizações podem executar para descrever visualmente todos os aspectos de um processo de negócio, incluindo o seu curso, controle e pontos de decisão, gatilhos e condições para execução das atividades, o contexto em que uma atividade executa e os recursos associados. Nesse caso a modelagem é apenas uma das várias etapas do gerenciamento de processos de negócio.

Como o título da seção sugere, nesse trabalho consideramos BPM como gerenciamento de processos de negócio.

Um processo de negócio é um conjunto estruturado de atividades modeladas para produzir uma saída específica para um cliente ou mercado particular. Ele Implica em uma forte ênfase em como o trabalho é feito por dentro de uma organização, em contraste com o foco apenas no produto. Um processo é, portanto, uma ordenação específica das atividades do trabalho ao longo do tempo e espaço, com um início, um fim, e uma identificação clara das entradas e saídas: uma estrutura para ação [Davenport, 1992].

Um processo de negócio pode, portanto, ser subdividido em subprocessos menores e mais concretos, que em algum nível de abstração essas atividades, ou passos do processo, são partes de uma camada que descreve como um determinado resultado pode ser alcançado.

5.2 Relação entre SOA e BPM

A decomposição dos processos de negócio, em um nível mais baixo de suas atividades, permite a identificação dos serviços de negócio. O objetivo desses serviços é executar

as funções necessárias ao negócio. A identificação dos seus relacionamentos com os serviços técnicos oferecidos pelos vários sistemas da organização permite agilizar a execução dos processos de negócio.

A identificação desses serviços, no entanto, não é uma tarefa simples e imediata. Em geral são utilizadas duas abordagens para resolver esse problema: *top-down* e *bottom-up*. A abordagem *top-down* permite que um processo ou sistema seja gradativamente decomposto em partes menores até atingir um nível de serviços básicos. Por outro lado, a abordagem *bottom-up* parte dos serviços já identificados ou oferecidos pelos sistemas da organização, compondo-os gradativamente em serviços mais complexos e generalizados [Perepletchikov *et al.*, 2005].

No entanto, todos os autores concordam que a aplicação de apenas uma das abordagens pode trazer sérios problemas na identificação de serviços. Uma abordagem *top-down* ajuda a entender a separação das atividades de uma forma mais lógica e otimizada. Por outro lado, essa abordagem pode levar a ignorar as realidades existentes na estrutura atual da organização, levando a altos custos para implantar o modelo desejado. Uma abordagem exclusivamente *bottom-up* pode levar ao engessamento do processo de negócio com base nas restrições técnicas existentes na organização.

[Bloomberg e Schmelzer, 2006] defendem que uma abordagem SOA deve ser tanto *top-down*, através da decomposição dos processos de negócio, quanto *bottom-up*, expondo as funcionalidades existentes como serviços e compondo-os dentro do processo. Se apenas a abordagem *top-down* for considerada, provavelmente a construção dos serviços poderá se basear em serviços que são tecnicamente difíceis ou complexos para implementar. Caso apenas a abordagem *bottom-up* seja empregada poderão ser produzidos serviços que não sejam necessários à organização, ou até pior, serviços que não atendam aos requisitos do negócio.

Portanto, uma iniciativa BPM em uma organização permite que os processos de negócio mais importantes que necessitam de um serviço que os suportem sejam identificados com a decomposição dos macroprocessos, em uma abordagem *top-down*. Além disso, uma iniciativa BPM permite que as atividades do nível mais baixo da modelagem sejam relacionadas a serviços técnicos existentes e que estes sejam orquestrados em processos mais complexos, facilitando assim uma abordagem bottom-up. Podemos dizer, portanto, que BPM oferece o apoio necessário para a implantação de SOA em uma iniciativa conhecida como *middle-out* (Figura 11). [Arsanjani, 2004] e [Perepletchikov *et al.*, 2005] chamam essa abordagem de *meet-in-the-middle*, o que parece mais apropriado, já que ela representa o encontro das estratégias *top-down* e *bottom-up* em algum ponto no meio de suas execuções.

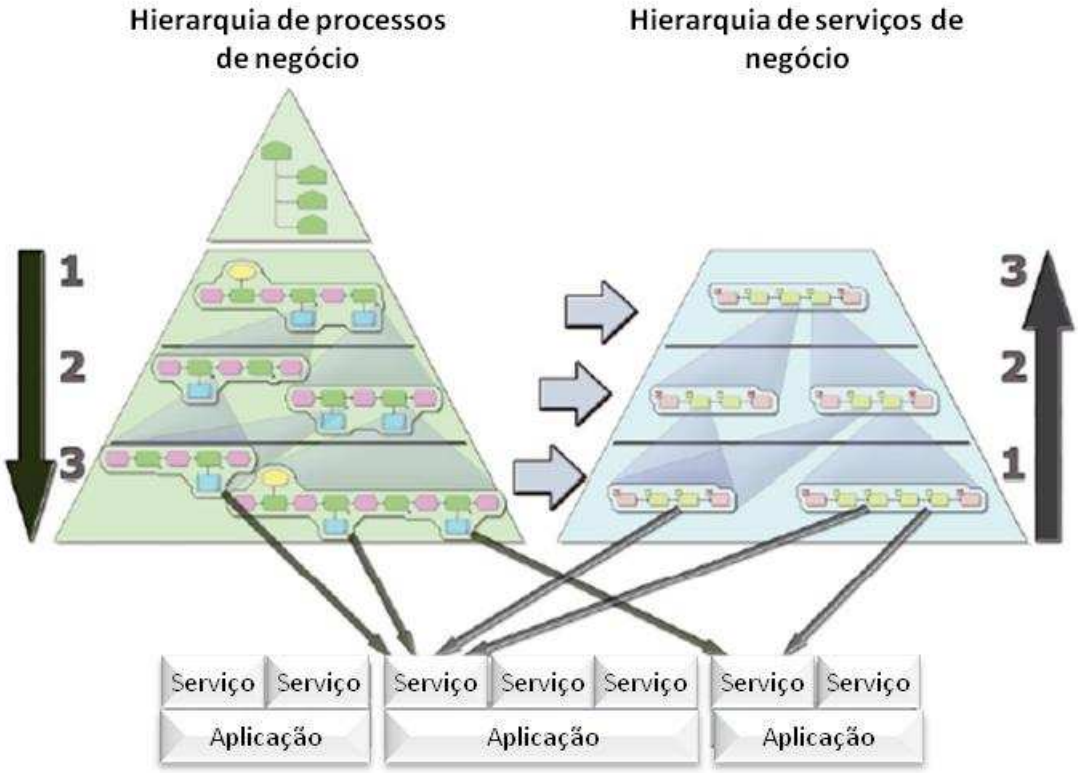

Figura 11 – Hierarquia de serviços de negócio baseada na hierarquia de processos de negócio

5.3 BPM/BR e SOA

Muitas linguagens de composição baseadas em workflows – como BPEL [Curbera, 2003], WSCI [Arkin, 2002], BPML [Arkin, 2002b], etc. – surgiram para agilizar a composição de serviços. Estas linguagens definem um processo de negócio que determina a dependência lógica entre os serviços compostos. O processo especifica a ordem de invocação (fluxo de controle) e as regras para transferência de dados (fluxo de dados).

[Charfi e Mezini, 2004] defendem que esta abordagem baseada em processos para definição de serviços compostos mostra pouca preocupação com o suporte à integração de regras de negócio.

De acordo com o *Business Rules Group* [The Business Rules Group, 2000], uma regra de negócio é uma expressão que define ou restringe alguns aspectos do negócio. Sua intenção é garantir a estrutura do negócio ou controlar o seu comportamento. Normalmente, regras de negócio são expressas como restrições ou em forma de condições. A abordagem por regras de negócio abrange um conjunto de termos (definições), fatos (conexões entre termos) e regras (cálculos, restrições e condições lógicas) [Von Halle, 2001]. Termos e fatos são expressões que contém observações relevantes ao negócio, ao passo que regras são expressões usadas para descobrir novas informações ou guiar tomadas de decisões.

Regras de negócio são especialmente úteis na decisão. Elas provêem os meios para expressar, gerenciar e atualizar pedaços de conhecimentos de domínio de negócio de forma independente do restante da aplicação. Um *Business Rule System* é um siste-

ma onde as regras de negócio são logicamente (e possivelmente, também fisicamente) separadas das outras partes.

Um dos problemas das linguagens orientadas a processos é que toda a lógica de negócio que sustenta a composição de um serviço, é expressa como um bloco monolítico, chamado de especificação do processo. Cada restrição ou política de negócio que deve ser sustentada pelo processo tem que ser expressada através de atividades, e deve ser integrada com a especificação do processo. Essa falta de modularidade dos processos dificulta a reusabilidade. Uma vez que toda lógica do negócio é definida em uma unidade, não é possível reutilizar suas partes.

Outro problema desse tipo de linguagens é a falta de flexibilidade, uma vez que elas assumem que a composição é predefinida e não se desenvolve. A única forma de acomodar mudanças é através da modificação da definição de processos. Isso significa que não há suporte para mudanças dinâmicas do processo. Flexibilidade e adaptabilidade são características muito importantes, principalmente no contexto altamente dinâmico da modelagem de regras de negócio para serviços. Isso porque organizações envolvidas em composições de serviços podem mudar suas regras de negócio, padrões e condições de colaboração com alguma frequência.

Se for possível quebrar a lógica do negócio, sustentando a composição em várias partes ou módulos, esta composição torna-se mais flexível desde que cada uma dessas partes possa desenvolver-se independente do restante. Isto leva à necessidade de uma abordagem de granularidade mais fina, com suporte apropriado para as partes da composição lógica para que possam ser criadas, modificadas ou apagadas dinamicamente em tempo de execução. Para alcançar esses objetivos é necessária uma abordagem híbrida que combine processos de negócio com regras de negócio [The Business Rules Group, 2000].

A idéia é que a parte principal da especificação da composição defina apenas o controle básico e o fluxo de dados entre os serviços compostos com a abordagem baseada em processo. Os aspectos da composição mais sensíveis à política da organização, por exemplo, regras de negócio que são sujeitas a mudanças, são modularizadas em unidades separadas. Dessa forma, quando as políticas do negócio mudam, é necessário modificar apenas as unidades correspondentes à modularização da implementação das regras de negócio. Além disso, esta separação reduz a complexidade da composição e auxilia na adaptabilidade.

De acordo com [Von Halle, 2001], existem quatro tipos de regras de negócio:

- Uma regra de restrição é uma sentença que expressa uma circunstância incondicional que deve ser verdadeira ou falsa. Por exemplo, "*um pedido de viagem deve ter um aeroporto de partida e um aeroporto de destino*".
- Uma regra de facilitação de ação é uma sentença que checa condições e, uma vez que estas sejam verdadeiras, inicia uma ação. Por exemplo, "*se não é encontrado nenhum vôo, não procurar por acomodação*".
- Uma regra de cálculo é uma sentença que checa uma condição e, quando o resultado é verdadeiro, provê um algoritmo para calcular o valor de um termo. Por exemplo, "*se mais de 2 pessoas viajam juntas, o terceiro paga apenas metade do preço*".
- Uma regra de interface é uma sentença que testa condições e, uma vez que estas sejam verdadeiras, estabelecem a veracidade de um novo fato. Por exemplo, "*se um cliente é assíduo, ele obtém um desconto de 5%*".

Um sistema de regra de negócio enfatiza especialmente a expressão, o gerenciamento e a automação das regras de negócio. [Von Halle, 2001] apresenta os princípios *STEP* da abordagem de regra de negócio: separa (*Separate*), investiga (*Trace*), externaliza (*Externalize*) e posiciona (*Position*) as regras para mudança. Deve-se focar nas regras de negócio que são relevantes para a composição de serviços, por exemplo, condições e/ou ações levam a vários serviços técnicos. Como exemplo, podemos considerar a sentença: "*se não forem encontrados vôos para as datas solicitadas pelo cliente, não procure por acomodação*", envolve três serviços: um serviço da companhia aérea, um com o hotel e uma composição desses serviços.

5.4 Ciclo de vida de integração SOA e BPM

[Klückmann, 2006] apresenta quatro passos cíclicos a serem seguidos na integração entre BPM e SOA (Figura 12): encontrar uma estratégia (Business Process Strategy); criar a base para a modelagem (modelagem de processos); implementação (transferência para TI); e controle (medição e análise). Esses passos devem ser cíclicos e contínuos para garantirem vantagens competitivas sustentáveis. As etapas do ciclo de vida serão complementadas com modelagem, implementação e controle de uma arquitetura orientada a serviço.

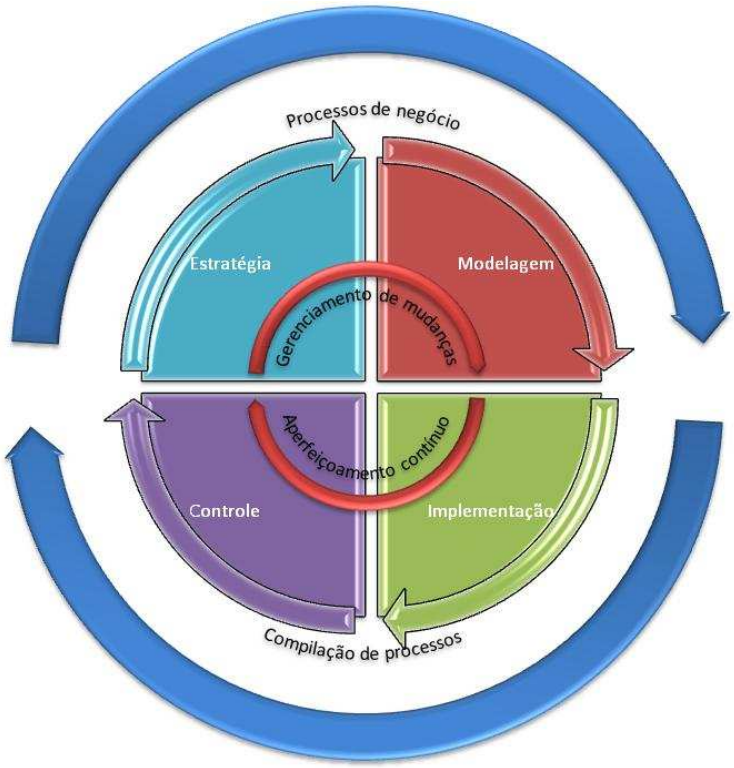

Figura 12 – Ciclo de vida BPM

1. **Fase de estratégia:** Nesta fase são determinados quais objetivos devem ser alcançados através das mudanças de processos desejados. Durante essa fase os objetivos funcionais ainda são independentes da sua implementação de TI. Sem um objetivo corporativo claro, será difícil derivar os processos de forma que qualquer objetivo seja alcançado.
2. **Fase de modelagem:** Nesta fase os processos são descritos independentemente do cenário futuro de TI. No entanto, o nível de granularidade da modelagem pode ser coordenado mais cedo, caso já exista alguma orientação em relação aos serviços

técnicos. Os processos funcionais são considerados consumidores de serviços, e as aplicações de software, provedores de serviços. Um planejamento da arquitetura de serviços consistente previne a redundância de serviços em um segundo momento.

3. **Fase de implementação:** Nesta fase os serviços técnicos existentes são orquestrados em relação ao o que foi modelado no processo de negócio funcional. Para tanto são seguidos os seguintes passos:

   3.1. Checar se existem serviços apropriados para os componentes funcionais que serão suportados.

   3.2. Observar os requisitos técnicos existentes para um repositório de serviços, pois isso é um dos fatores determinantes para a definição da granularidade dos serviços.

   3.3. Desenvolver serviços que são necessários e ainda não são oferecidos pelos fabricantes dos sistemas. Nesse ponto o autor apresenta MDA (*Model-Driven Architecture*) [Mukerji e Miller, 2003] como uma boa abordagem para implementação de serviços baseada em UML.

   3.4. Explicitar as regras de negócio da definição do processo. Essas regras freqüentemente definem alternativas no fluxo de trabalho, por isso são fortes candidatas a serem disponibilizas como serviços para que sejam reutilizadas em outras aplicações.

4. **Fase de controle:** Nesta fase as capacidades de desempenho da arquitetura orientada a serviço são mensuradas na execução do sistema. Monitoração de processos e *Business Activity Monitoring* (BAM) são fundamentais para a avaliação do sucesso do projeto SOA e identificar fraquezas na arquitetura de TI. As categorias de medição incluem a freqüência das chamadas dos serviços, a duração da execução dos serviços e a intercomunicação entre serviços.

5.5 Exemplo de BPM com serviços

Em geral, uma vez que é necessário implementar um novo processo de negócio, a organização parte de um cenário inicial onde possui serviços básicos oferecidos por sistemas já existentes. O próximo passo é iniciar uma decomposição *top-down* onde serão decididos quais processos serão manuais e quais serão suportados pela TI. Em seguida o processo é dividido em partes menores de acordo com o momento em que são executadas e quais sistemas são responsáveis por elas. A partir desse momento os aspectos complexos são quebrados em passos mais gerenciáveis. A Figura 13 apresenta um esboço de decomposição de processos *top-down*.

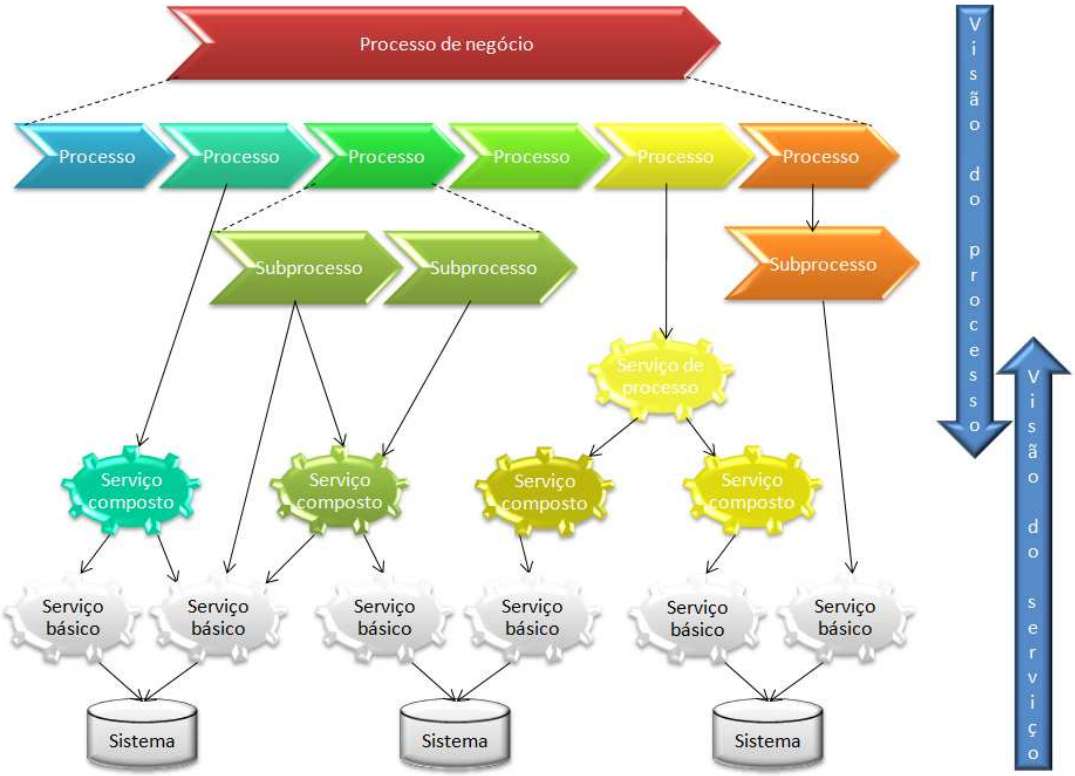

Figura 13 – Integração entre processos, subprocessos e atividades e serviços

Nesse ponto o produto gerado é uma modelagem de alto nível que permite que sejam determinados quais sistemas de TI são responsáveis por quais partes do processo e, a partir dessa informação, saber quais pessoas devem ser envolvidas na modelagem dessa solução distribuída.

O nível de abstração em que os subprocessos e/ou atividades devem ser modelados para que sejam associados aos serviços será definido caso a caso. Mas ao decompor um processo é fundamental levar em consideração os serviços existentes para que a modelagem não se distancie do cenário tecnológico atual da organização.

5.6 Orquestração × Coreografia

Além da orquestração, existe outra forma de compor serviços existentes em serviços e processos de níveis mais altos, a coreografia. Nessa abordagem as partes do processo colaboram entre si, tomando a responsabilidade por um ou mais passos. Isso significa que nenhuma dessas partes, ou os setores por ela responsáveis, controlam o processo com um todo.

Em geral a coreografia assemelha-se mais com o funcionamento do processo na prática. Por exemplo, no momento que um determinado setor de uma organização recebe um documento, os demais setores não têm controle sobre como o portador do documento naquele momento irá executar sua tarefa, até que o documento seja entregue ao departamento que deve recebê-lo em seqüência. Uma vez entregue o documento para o setor seguinte, o setor atual deixa de ter controle sobre o documento.

A vantagem dessa abordagem sobre a orquestração é que, como não centraliza o controle, ela permite maior escalabilidade para o processo. Um processo muito grande e complexo poderá sobrecarregar um serviço ou mesmo uma equipe responsável por controlá-lo. Por outro lado, a descentralização da coreografia cria grandes dificuldades

para identificar o estado atual em que um processo se encontra, ou até mesmo identificar a razão de um comportamento errado. Na orquestração o controlador mantém o estado processo e identifica qual passo do processo falhou em caso de erro.

A coreografia, portanto, é extremamente dependente de colaboração e especificação de regras gerais que definam a integração entre as partes do processo.

6 ARIS × BPM × SOA

De acordo com a metodologia SOA da *IDS Scheer* [Klückmann, 2007] a hierarquia de serviço técnico é construída ao mesmo tempo em que a hierarquia de processos de negócio. Enquanto os processos de negócio são compilados usando uma abordagem *top-down*, os serviços são orquestrados em processos técnicos usando uma abordagem *bottom-up*. Essa metodologia recomenda 10 passos para uma implementação SOA orientada a negócio, apresentada na Figura 14 e detalhada a seguir.

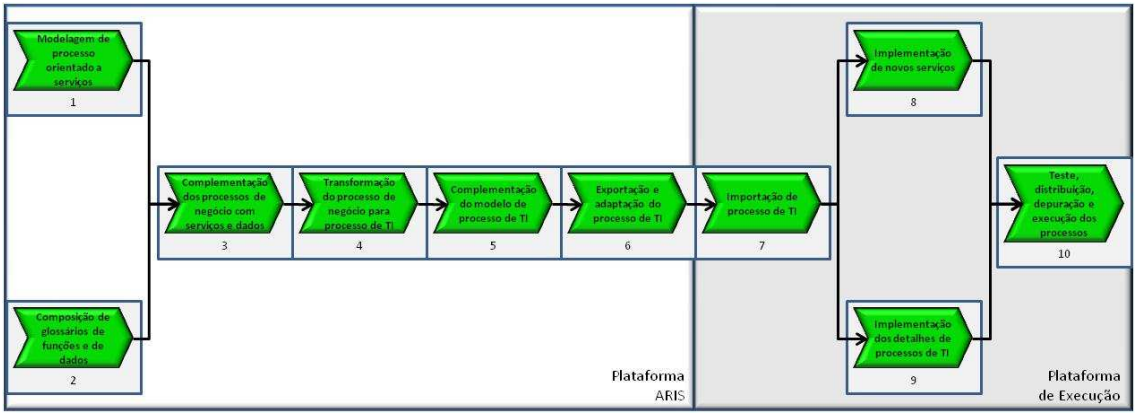

Figura 14 – 10 passos para processos para SOA orientado a negócio

1. **Modelagem de processo orientado a serviços:** Esta primeira etapa envolve a criação de modelos de negócio de alto nível que gradualmente são substituídos por modelos de processos orientados a serviços, segundo uma abordagem top-down. Essa modelagem revela as propriedades que os serviços técnicos precisam ter para, em um segundo momento, suportar as atividades de negócios definidas. A Figura 15 mostra um EPC (*Event Driven Process Chain*, cadeia de processos dirigida por eventos) de um processo de vendas simples.

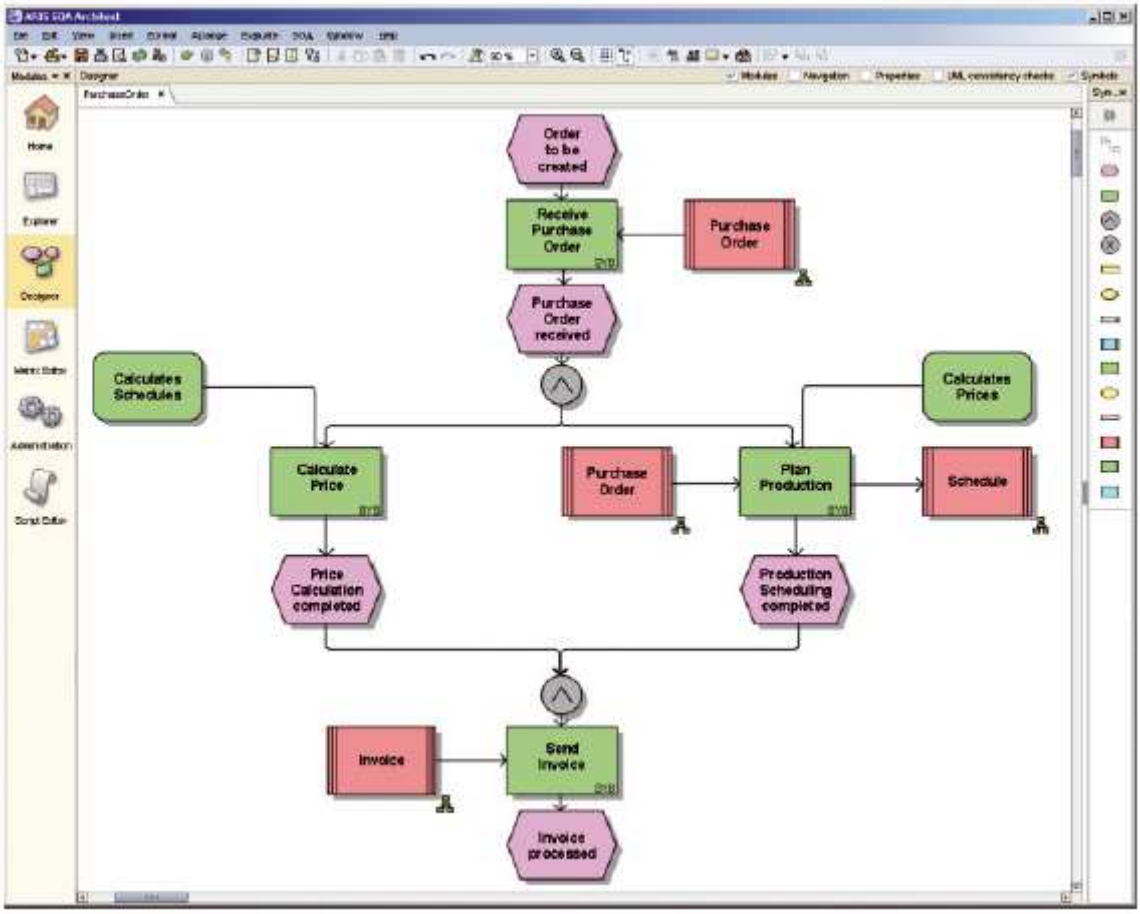

Figura 15 – Event-driven process chain (EPC)

2. **Composição de glossários de funções e de dados:** Nessa fase os WSDLs dos serviços técnicos existentes são importados. O ARIS permite que os serviços importados sejam divididos em blocos de negócios e em blocos técnicos simultaneamente. Dessa forma, através dos serviços classificados em blocos dos dois tipos, é possível visualizar a interface entre a descrição dos serviços de negócios e a descrição de serviços técnicos. A Figura 16 mostra um arquivo WSDL com a descrição de um serviço sendo importado para o ARIS.

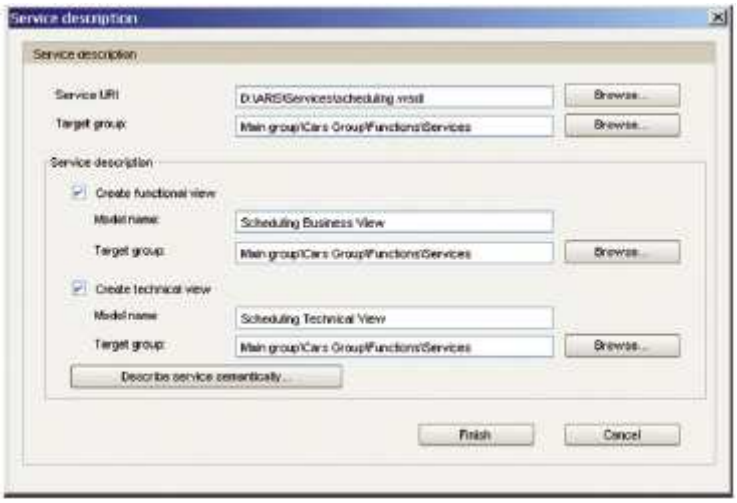

Figura 16 – Caixa de diálogo de importação de serviço

3. **Complementação dos processos de negócio com serviços e dados:** A equipe SOA

identifica os serviços mais adequados para automatizar processos nesta etapa. Esta equipe deve garantir que cada atividade que deve ser automatizada seja suportada por um serviço. Processos mapeados em serviços em um nível hierárquico mais elevado precisam implementar uma transação de negócio concreta. Segundo o [Klückmann, 2007], o ARIS SOA Architect permite o mapeamento automático de uma atividade de processo ao serviço mais adequado, baseado nos requisitos de negócio e/ou na modelagem de dados no ARIS e na descrição oferecida pelo WSDL do serviço. Caso um serviço adequado não seja encontrado, a equipe pode modificar a definição do processo ou um dos serviços existentes, ou ainda criar um novo serviço. No último caso, a ferramenta ARIS UML Designer permite a definição de serviços. O objeto "*Scheduling*" na Figura 17 representa o serviço que irá automatizar a função "*Plan Production*".

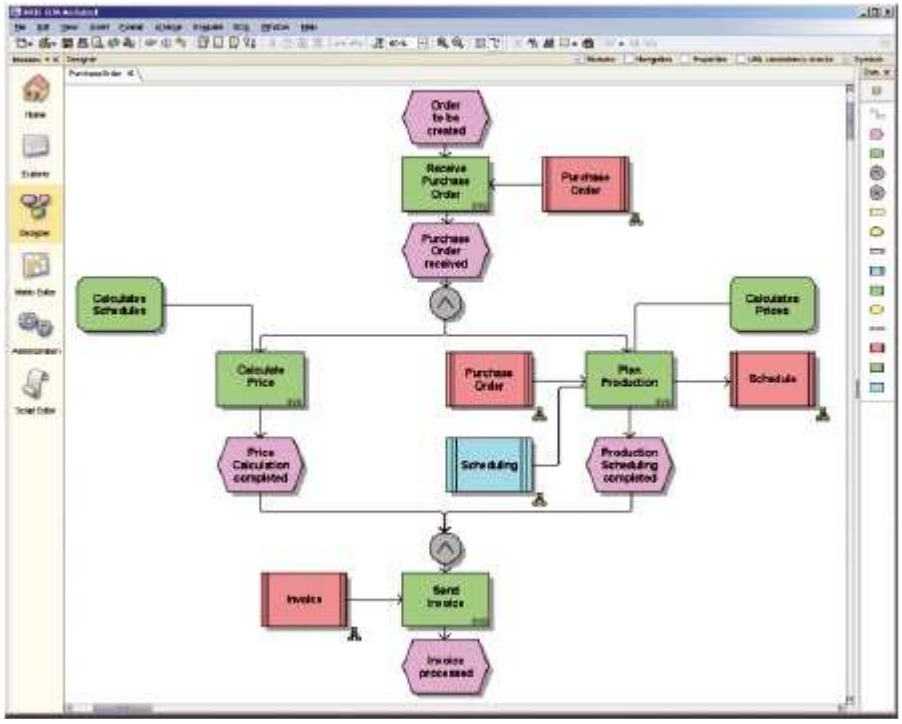

Figura 17 – Serviço no modelo de processo

4. **Transformação do processo de negócio para processo de TI:** Nesta etapa os processos de negócio acrescidos dos serviços técnicos são transformados em processos BPEL. Para isso, informações técnicas necessárias para o processo BPEL devem ser adicionadas no modelo. Segundo o autor, o conteúdo do processo de negócio que não é relevante para a construção do processo BPEL pode ser escondido do modelo. A Figura 18 mostra um processo BPEL gerado a partir do EPC apresentado na Figura 17.

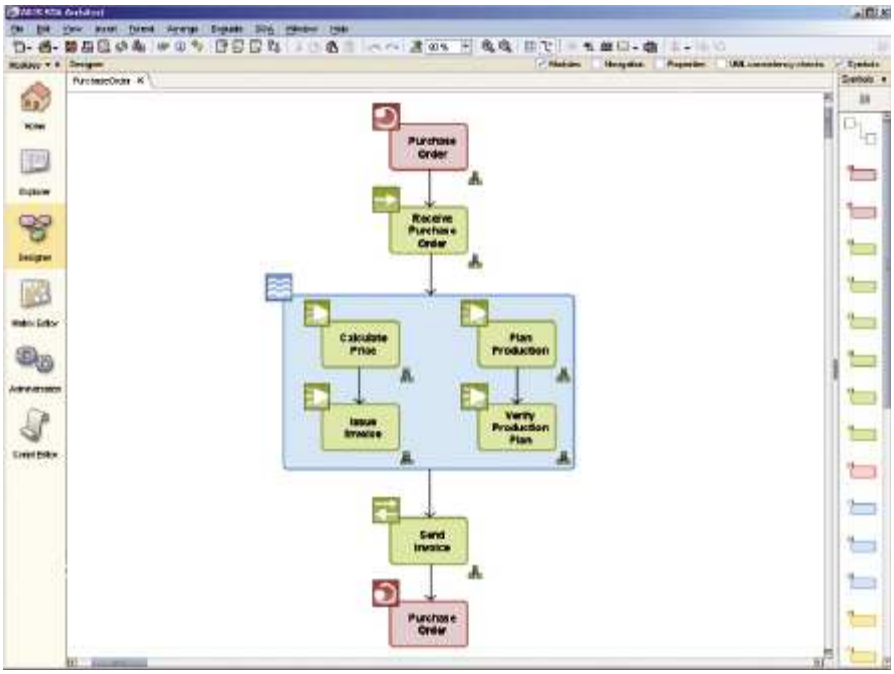

Figura 18 – Processo BPEL gerado automaticamente

5. **Complementação do modelo de processo de TI:** Nessa etapa são modelados detalhes técnicos como exceções e *correlation sets.Um correlation set* é um mecanismo de BPEL para prover a correlação entre mensagens assíncronas baseado no conteúdo da mensagem (Figura 19).

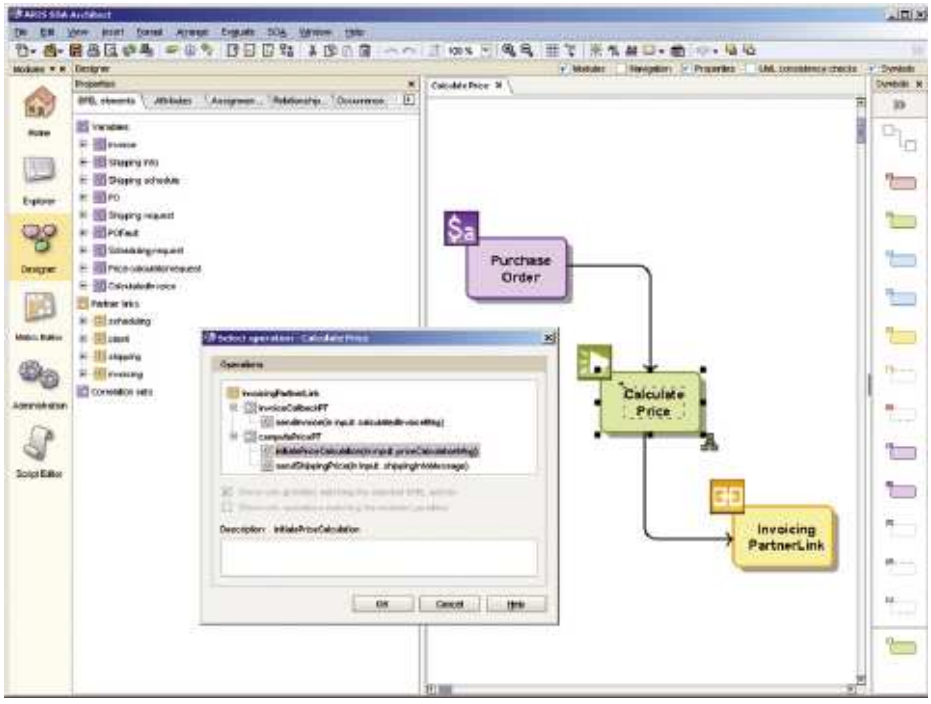

Figura 19 – Completando o processo de modelagem BPEL

6. **Exportação e adaptação do processo de TI:** Uma vez criado e detalhado o modelo de processo BPEL pode ser exportado como um arquivo XML. Junto com esse arquivo é criado um WSDL que permite que esse novo serviço seja disponibilizado no ARIS para que seja reutilizado em um novo ciclo de modelagem. Como os arquivos seguem padrões eles podem ser importados pela maioria das plataformas de aplicações. A Figura 20 mostra um código BPEL exportado pela ferramenta A-

RIS.

Figura 20 – Código BPEL

7. – 10. **Execução do serviço:** Segundo o artigo, as plataformas de aplicações introduziram extensões específicas no BPEL nos últimos anos, o que pode exigir a inclusão de novos elementos no modelo de processo BPEL depois que este for importado para uma plataforma específica. Estes elementos não são modelados no ARIS para que este seja mantido independente de plataforma. Atividades de teste e correção de falhas devem ser executadas antes da execução e disponibilização dos serviços. Estas atividades são consideradas nos passos 7 a 10.

[Klückmann, 2007] defende que há um aumento de custos, por conta de uma falta de transparência e reusabilidade, quando são usados repositórios diferentes para os elementos SOA de negócio e tecnológicos. Com o armazenamento dessas informações no ARIS é possível identificar facilmente todas as interdependências entre esses elementos.

7 Serviços de Informação

Em um projeto SOA deseja-se identificar serviços e suas especificações. Em outras palavras, deseja-se saber quais funções e dados devem ser expostos como serviços e como devem ser definidos e modelados os serviços identificados [Byrne *et al.*, 2008d].

Muitas metodologias para desenvolvimento de serviços baseiam-se fortemente em análise de processos de negócio, projeto de casos de uso e projeto de bancos de dados para projetar interfaces de serviços com o nível de granularidade apropriada, estabelecendo reuso etc. A maioria destas metodologias está preocupada em tratar serviços que tratam de regras de negócio, não tendo foco em se definir uma camada de abstração para acesso aos dados.

Como o mesmo conjunto de dados pode ser acessado por diferentes serviços, é importante que o acesso aos dados não seja replicado em cada serviço que o acessa. Dessa forma, as regras e formas de acesso aos dados ficam transparentes, bem como ganha-se com flexibilidade de manutenção.

Uma solução geralmente adotada é tratar serviços de informação como um número pequeno de serviços que realizam consultas na base de dados. Para suportar escalabilidade, consistência e reuso do acesso à informação, a solução SOA deve incluir conceitos, padrões e estratégias estabelecidos de arquitetura de informação. A informação deve ser disponibilizada a fim de melhor suportar objetivos técnicos e do negócio em uma solução SOA, tais como:

- Reuso de serviços;
- Informações do negócio expostos de forma precisa, completa e em tempo hábil;
- Informações compartilhadas devem ter uma estrutura e significado comum para todos;
- Dados integrados;
- Entidades de dados que ligam domínios de negócio da organização devem ser consistentes e confiáveis;
- Organização deve conseguir obter o maior valor a partir dos dados.

Deve-se aplicar uma abordagem estruturada para análise, modelagem e projeto de dados em um projeto SOA que implicam uma implementação da solução que melhor atenda aos requisitos de negócio existentes e uma melhor preparação para se adaptar a novos requisitos. Existem diferentes padrões para tratar a perspectiva da informação que podem ser utilizados em diferentes situações, tais como: serviços de acesso a informações básicas; federação de dados; consolidação de dados; limpeza de dados; integração de conteúdo; etc. Neste capítulo trataremos dos principais aspectos relacionados ao tratamento da informação em uma arquitetura SOA.

7.1 Níveis de abstração para serviços de informação

Assim como temos os níveis de abstração de banco de dados (nível conceitual, nível lógico e nível físico) [Elmarsi e Navathe, 2007], podemos considerar os mesmos níveis de abstração para serviços de informação. No caso do nível conceitual, podemos dividi-lo em 3 partes: semântica, estrutura e conteúdo. Semântica pode ser representada através de um glossário de termos do negócio. Estrutura pode ser representada através de um modelo canônico como, por exemplo, através de um modelo Entidade-Relacionamento ou modelo de classes. Já o conteúdo trata da qualidade dos dados que serão armazenados. A Figura 21 mostra o relacionamento entre os diferentes níveis de abstração.

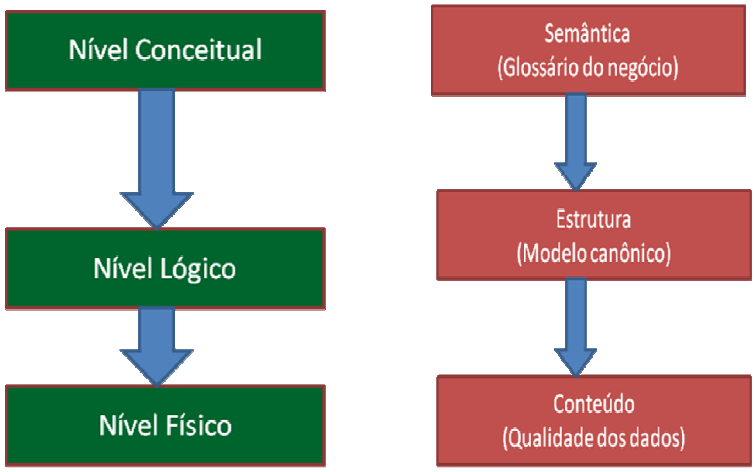

Figura 21 – (a) Níveis de abstração propostos para serviços de informação, (b) Sub-níveis de abstração para o caso do nível conceitual.

7.1.1 Nível conceitual: Perspectivas da informação

Os analistas do negócio e equipe técnica devem ter um entendimento comum a respeito das terminologias do domínio, incluindo processos, serviços e dados. O Glossário elimina ambigüidade e esclarece o entendimento da semântica dos conceitos. Cada termo deve ser definido com uma descrição e posicionado em uma taxonomia. Deve-se definir responsáveis por manter termos, a fim de definir e suportar a governança destes termos. O Glossário deve ser acessível; deve estar ligado a outros artefatos de modelagem; e deve ser utilizado ativamente na fase de projeto.

7.1.1.1 Glossário de dados (semântica)

Os analistas do negócio e equipe técnica devem ter um entendimento comum a respeito das terminologias do domínio, incluindo processos, serviços e dados. O Glossário elimina ambigüidade e esclarece o entendimento. Cada termo deve ser definido com uma descrição e posicionado em uma taxonomia. Deve-se definir responsáveis por manter conceitos. O Glossário deve ser acessível; deve estar ligado a outros artefatos de modelagem; e deve ser utilizado ativamente na fase de projeto [Byrne *et al.*, 2008c].

7.1.1.2 Modelo canônico (estrutura)

O modelo canônico provê entendimento claro de como as informações do negócio estão estruturadas (principais entidades, seus atributos e relacionamentos). Ele auxilia na definição dos parâmetros de entrada e saída de serviços e facilita a identificação de conjuntos de dados reutilizáveis em múltiplos domínios, resultando em definições de serviços que podem ser utilizados por diferentes consumidores, o que reduz a quantidade de serviços duplicados [Byrne *et al.*, 2008b].

7.1.1.3 Qualidade dos dados (conteúdo)

Dados que estão de acordo com regras e descrições dos seus repositórios originais podem não satisfazer os requisitos a nível da organização. Por exemplo, identificador único de registros em bases de dados diferentes. Os identificadores são únicos em suas bases locais, mas não são únicos quando o domínio é extrapolado para a organização. Os critérios de qualidade dos dados em uma aplicação local podem não ser tão forte quando o dado é disponibilizado ao nível da organização. Por exemplo, valores não preenchidos, dados redundantes e em formatos inconsistentes [Byrne *et al.*, 2008a].

Devem ser conduzidas atividades para garantia de qualidade de dados na análise e projeto SOA. Após catalogar os sistemas, a partir do qual serão definidos os serviços, deve-se avaliar a qualidade dos dados dos mesmos, por exemplo:

- Verificar se os dados estão de acordo com as regras de integridade que os definem;
- Verificar se há replicação de dados e como isto pode ser resolvido durante correspondência dos dados e agregação de dados.

Deve-se executar ações necessárias para garantir que implementações de serviços atendam níveis de precisão e significado dos dados.

7.1.2 Nível Lógico

O nível lógico representa uma visão especializada para as fontes de dados existentes e serviços que irão acessá-las. O nível lógico inclui definições de:

- Interfaces dos serviços (parâmetros de entrada e de saída);
- Quais fontes de dados serão acessadas por quais serviços;
- Transformações de dados;
- Orquestração.

O nível lógico é representado através de modelos gráficos e especificações textuais.

7.1.3 Nível Físico

O nível físico de um serviço corresponde à implementação dos serviços de informação a partir do entendimento dos requisitos de dados necessários para a organização.

7.2 Desafios para serviços de Informação

Serviços de informação são serviços que dependem da arquitetura da informação, onde a separação da informação de aplicações e processos provê benefícios.

Alguns desafios em SOA são problemas conhecidos em arquiteturas de informação tradicionais, tais como [Byrne *et al.*, 2008d]:

- Existência de diversos conjuntos de dados e variadas tecnologias para armazenar e processar dados;
- Informação geralmente aparece espalhada e replicada em muitos sistemas;
- Informação é atualizada segundo regras locais e não de acordo com regras globais, da organização como um todo.

O entendimento destes problemas pode ser feito através do modelo canônico, o qual permite identificar inconsistências nas definições dos dados. As informações estratégicas e reutilizáveis da organização devem ser vistas como um conjunto de enti-

dades padronizadas para reuso por toda organização; de acordo com padrões estruturais e semânticos da indústria; e de acordo com contratos dos serviços que expõem estas informações. O objetivo é criar um conjunto de serviços que sejam um caminho oficializado, único e consistente para acessar as informações da organização.

7.3 Princípios para serviços de informação

Deve haver uma fonte lógica e única de informações consistentes e completas através de interfaces de serviços. A heterogeneidade e complexidade devem ser encapsuladas em serviços, os quais fazem o mapeamento de entidades lógicas do negócio para os dispositivos de armazenamento reais. As fontes de dados oficiais devem ser claramente identificadas e utilizadas por toda a organização.

Deve haver disponibilização de metadados sobre os serviços de informação a fim de que:

- A estrutura e semântica da informação sejam conhecidas;
- A informação seja disponibilizada com qualidade e atendendo aos requisitos do negócio;
- A "Idade" da informação seja conhecida e existam mecanismos para disponibilizar a informação com o tempo desejado.

Os serviços de informação podem ser governados baseados em processos, políticas e estruturas organizacionais adequadas para garantir, por exemplo:

- Segurança
- Auditoria das mudanças dos serviços
- Facilidade de localização dos serviços

7.4 Arquitetura para serviços de informação

[Byrne *et al.,* 2008] propõe uma arquitetura para serviços de informação em SOA com as seguintes camadas, a qual é ilustrada na Figura 22:

- Serviços de dados: responsáveis por obter o dado no formato desejado e expô-los como serviços. Corresponde a funcionalidades de get e set.
- Serviços de metadados: responsáveis por disponibilizar metadados de serviços;
- Serviços de conteúdo: responsáveis por prover mecanismos de acesso a dados distribuídos e heterogêneos;
- Serviços de integração de informações: provê mecanismos para tratar qualidade, limpeza, transformação e disponibilidade dos dados integrados;
- Serviços de gerência de dados principal: fonte principal oficial de registros para dados no nível da organização: acesso consistente, completo, contextual e preciso;
- Serviços analíticos: consolida, agrega e sumariza dados estruturados e não estruturados e calcula informações analíticas: índices, tendências, e predições.

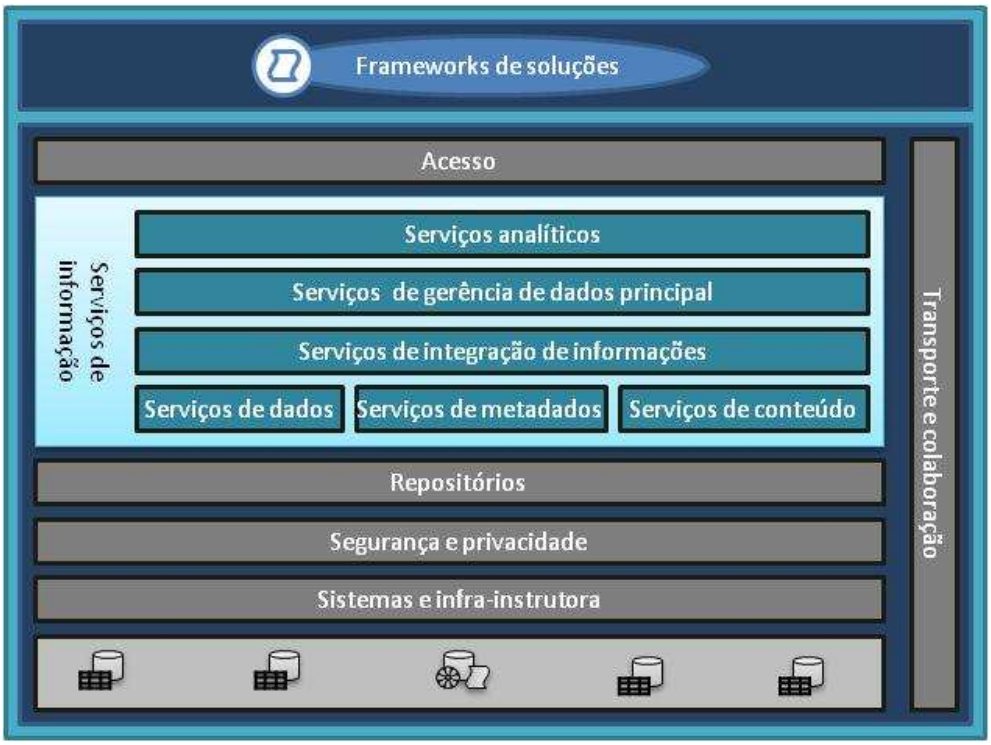

Figura 22 – Proposta de arquitetura para serviços de informação

As principais características que os serviços de informação devem possuir são:

- A qualidade da informação deve ser conhecida e sua integridade garantida quando da consulta e atualização em diferentes repositórios de informação;
- Metadados armazenam esta sobre disponibilidade das fontes de dados;
- Heterogeneidade deve ser transparente:
  - Serviços consumidores não precisam tratar diversidade de fontes de dados e diversidade de formatos
- A Estrutura da informação deve ser conhecida
- Mudanças nos serviços e nas informações subjacentes devem ser feitas de maneira uniforme e consistente.

7.5 Padrões para serviços de informação em SOA

Existem um conjunto de padrões para tratar informação como serviço. De acordo com as características da organização e dos problemas sendo enfrentados, estes padrões podem ser adotados [Byrne *et al.*, 2008d].

7.5.1.1 Modelo de ciclo de vida

Em uma organização, podem ocorrer diferentes problemas relacionados à informação, tais como:

- Terminologia não estar clara. Por exemplo: Qual o significado da entidade cliente para os diferentes departamentos? Cliente É quem compra, é quem deseja comprar, é quem se cadastrou?

- Dados não estarem com a qualidade adequada. Por exemplo, o campo endereço do cliente pode ter diferentes formatos em diferentes fontes de dados.
- Modelos inconsistentes de serviços. Por exemplo, o modelo de mensagens que descreve entrada e saída dos serviços estar inconsistente.

Uma solução que pode ser adotada para resolver estes problemas é empregar um modelo de ciclo de vida, considerando os diferentes níveis de abstração de serviços de informação, como apresentado na seção 7.1 :

- Dicionário de dados: usado para esclarecer terminologias, estabelecer vocabulário comum que controla a definição dos termos e definir gerente do conceito.
- Modelo canônico de dados: usado para prover consistência nas definições de entidades, atributos e seus relacionamentos nos diferentes sistemas.
- Análise da qualidade dos dados, usada para endereçar problemas de duplicação, inconsistência e precisão dos dados.
- Projeto das interfaces e orquestrações de serviços.
- Implementação de serviços.

A Figura 23 ilustra os diferentes artefatos do modelo de ciclo de vida de serviços.

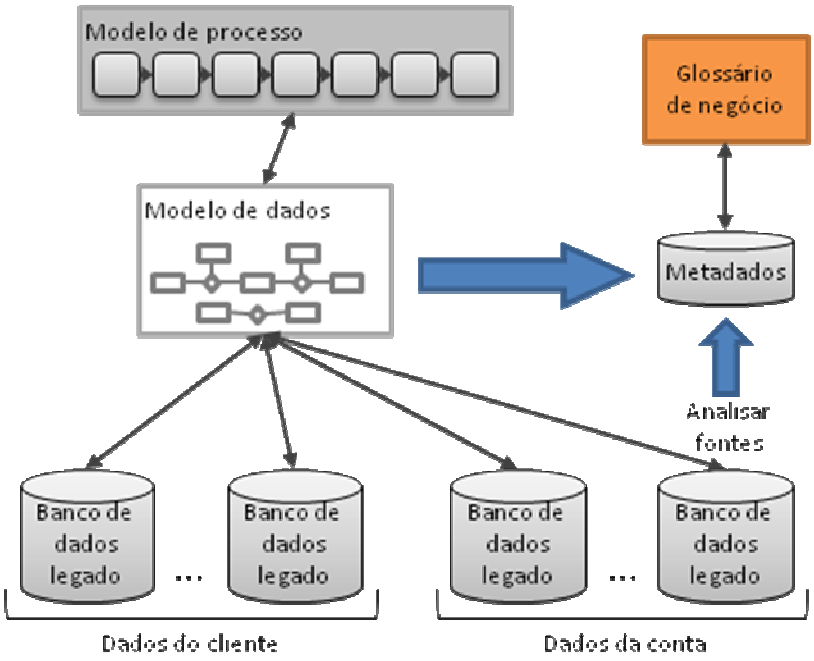

Figura 23 – Modelo de ciclo de vida

7.5.1.2 Serviços de acesso a informações básicas

Serviços de acesso a informações básicas podem ser utilizados como qualquer outro serviço. Os problemas mais freqüentes são que tradicionalmente, aplicações acessam diretamente as fontes de dados através de ODBC, JDBC e há acoplamento com fontes de dados e modelos de dados. Os serviços de informação provêem baixo acoplamento de aplicações com fontes de dados, o que traz agilidade.

O padrão serviços de acesso às informações propõe a disponibilização de serviços que tratem as seguintes questões:

- Gerência de metadados
- Monitoramento
- Gerenciamento
- Mapeamento a nível de segurança
- Escalabilidade
- Balanceamento de carga
- Ambiente de desenvolvimento integrado

A Figura 24 ilustra a implantação de uma camada de abstração chamada de Disponibilização de Serviços de Informação que trata destes aspectos. Serviços que tratam dos processos de negócio, portais e outras aplicações acessam os dados via esta camada.

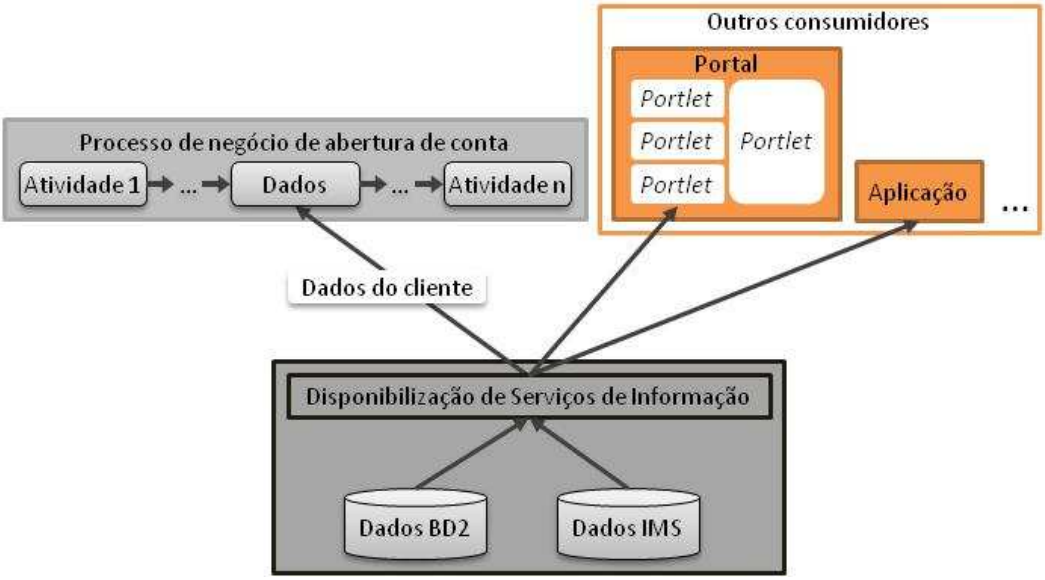

Figura 24 – Serviços de acesso a informações básicas

7.5.1.3 Federação de dados

Em uma organização frequentemente existem dados em diferentes formatos, armazenados nos mais diversos tipos de bancos de dados (sejam bancos de dados XML, bancos de dados relacionais, bancos de dados objeto-relacionais etc). O termo federação de dados corresponde à de diferentes informações bem como de informações tratando do mesmo conceito em diferentes fontes de dados, tendo que ser acessadas em um curto período de tempo. Além disso, a informação está distribuída por toda a organização. Neste caso, não é viável integrar os dados copiando-os em uma única fonte de dados, pois a necessidade de acesso rápido à informação causará replicação de dados e problemas de atualização dos dados replicados.

A solução seria a utilização de um servidor de dados federados responsável por:

- Receber a consulta em uma visão integrada
- Transformar a consulta em sub-operações e enviar para as fontes apropriadas
- Reunir os resultados de cada fonte

- Montar e retornar o resultado integrado

O processamento ocorre de forma síncrona, em tempo real e é transparente para o usuário.

A Figura 25 ilustra um Servidor de Dados Federados a partir do qual existem serviços de acesso a informação permitindo acesso integrado aos dados.

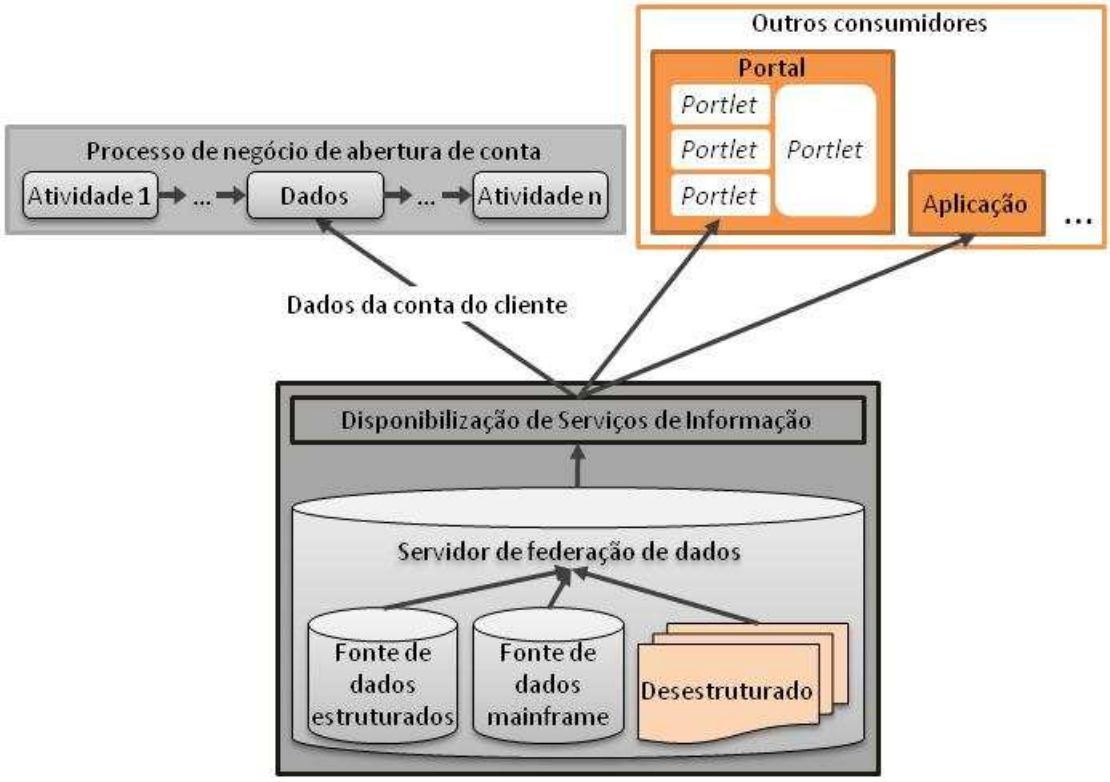

Figura 25 – Federação de dados

7.5.1.4 Consolidação de dados

Em uma organização com grandes volumes de dados, é comum a necessidade de acesso a dados históricos (por exemplo, de clientes) para análise e auditorias. A existência de dados espalhados em diferentes fontes de dados e o fato da informação não precisar ser obtida em tempo real, pode permitir o uso do padrão de consolidação de dados. Este padrão propõe o uso de um servidor de dados consolidados, o qual é responsável por:

- Consolidar dados de diferentes fontes de dados, resolvendo conflitos e juntando dados
- Permitir que dados consolidados sejam acessados independentes de processos de transformação e integração
- Permitir que dados sejam acessados concorrentemente, garantindo escalabilidade e alto desempenho.

A Figura 26 ilustra o padrão de consolidação de dados.

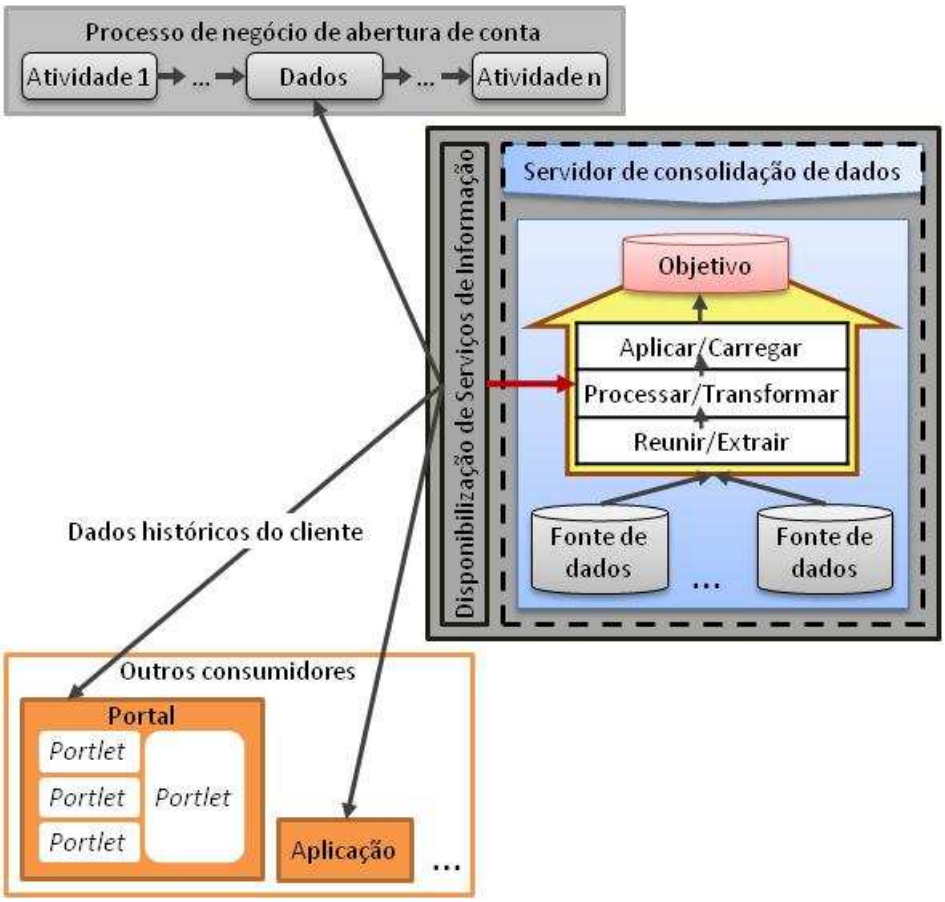

Figura 26 – Consolidação de dados

7.5.1.5 Limpeza de dados

Dados sendo acessados devem estar consistentes e serem válidos. Por exemplo, informações inconsistentes sobre clientes em diferentes fontes de dados ou dificuldade de entrar em contato com o cliente devido a erros nos dados, podem ocorrer se questões de limpeza de dados não forem consideradas.

Dessa forma, torna-se importante o uso de um servidor de limpeza de dados que permita a construção de regras de limpeza de dados com checagens, normalizações e transformações, no qual as informações possam ser corrigidas quando da entrada de dados ou nas próprias fontes de dados. Além disso, o servidor de limpeza de dados permite ligar registros e duplicar registros nas diferentes fontes de dados aplicando sofisticadas regras de *parsing* e funções estatísticas de semelhança. Possibilita também a seleção de um único registro a fim de popular dados não preenchidos ou dados incompletos.

A Figura 27 ilustra o padrão de limpeza de dados.

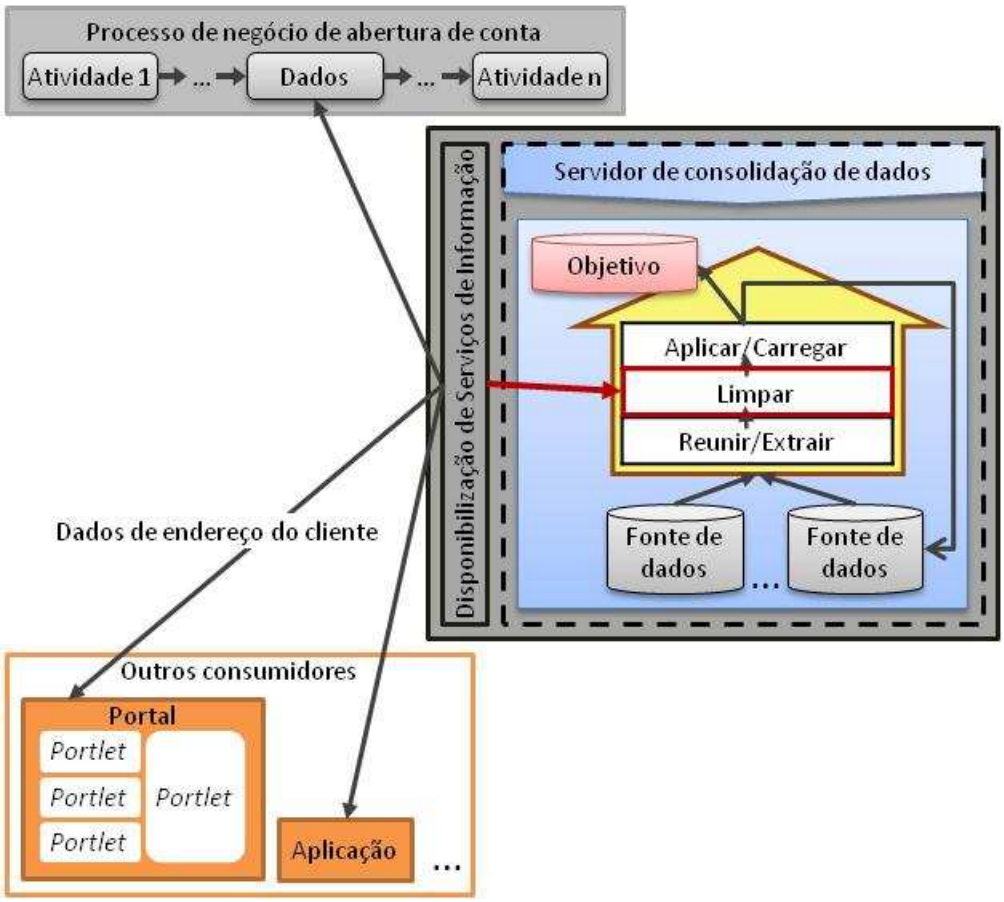

Figura 27 – Limpeza de dados

7.5.1.6 Integração de conteúdo

A integração de conteúdo trata da possibilidade de acessar informações não estruturadas de forma integrada. Por exemplo, dados armazenados arquivos manuseados manualmente em excel ou word.

O padrão de integração de conteúdo propõe o uso do servidor de integração de conteúdo, permitindo que dados sejam armazenados em um repositório de conteúdo juntamente com metadados que os descrevem. O servidor de integração disponibiliza serviços para acessar os conteúdos a partir dos metadados.

A Figura 28 ilustra o padrão de integração de conteúdo.

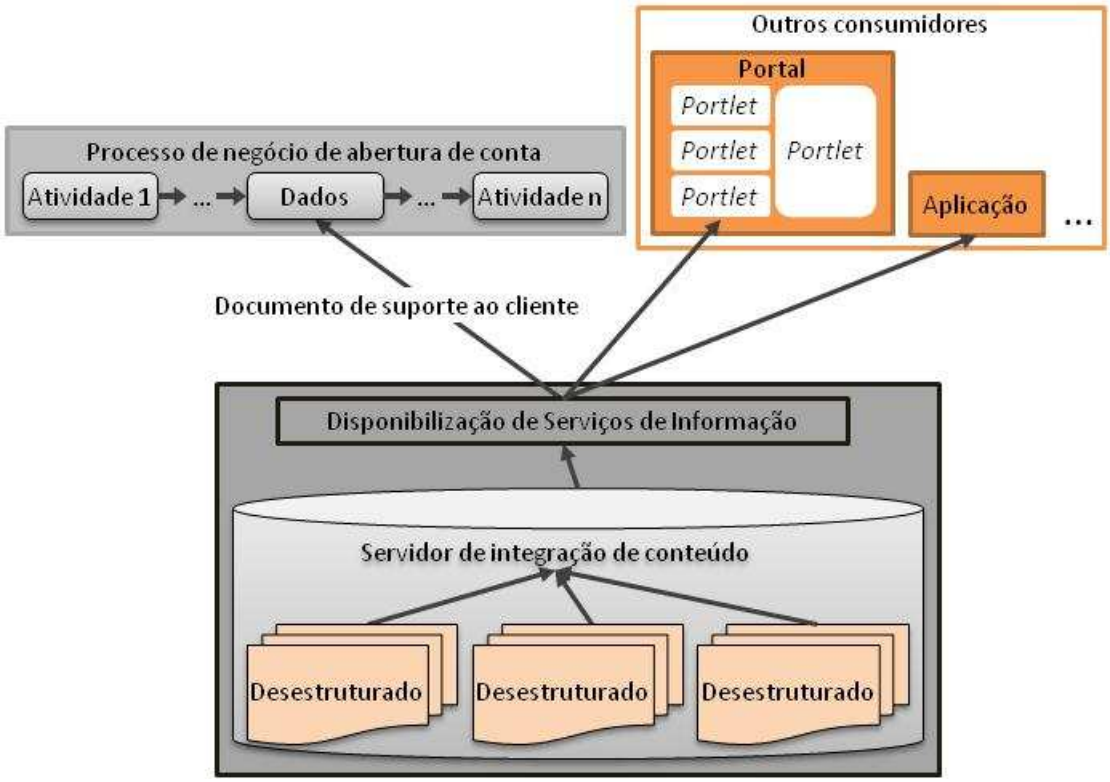

Figura 28 – Integração de conteúdo

7.5.1.7 Master data management

O padrão Gerente de Dados Principal (*Master Data Management*) permite a gerência robusta a diferentes fontes de dados. Por exemplo, informações de clientes, produtos, e fornecedores.

Master data management considera padrões de consolidação de dados e limpeza de dados, além de regras de controle de eventos e ferramental para gerência de dados. Ele permite garantir modelo de dados padrão e garante que dados chaves estejam sempre completos e precisos.

Além disso, provê um conjunto de serviços centralizados para gerenciar dados chaves para o negócio. O *Master Data Management* atua como a fonte oficial de dados, executando sincronização de dados com outras fontes e gerência dos dados.

A Figura 29 ilustra o padrão *Master data management*.

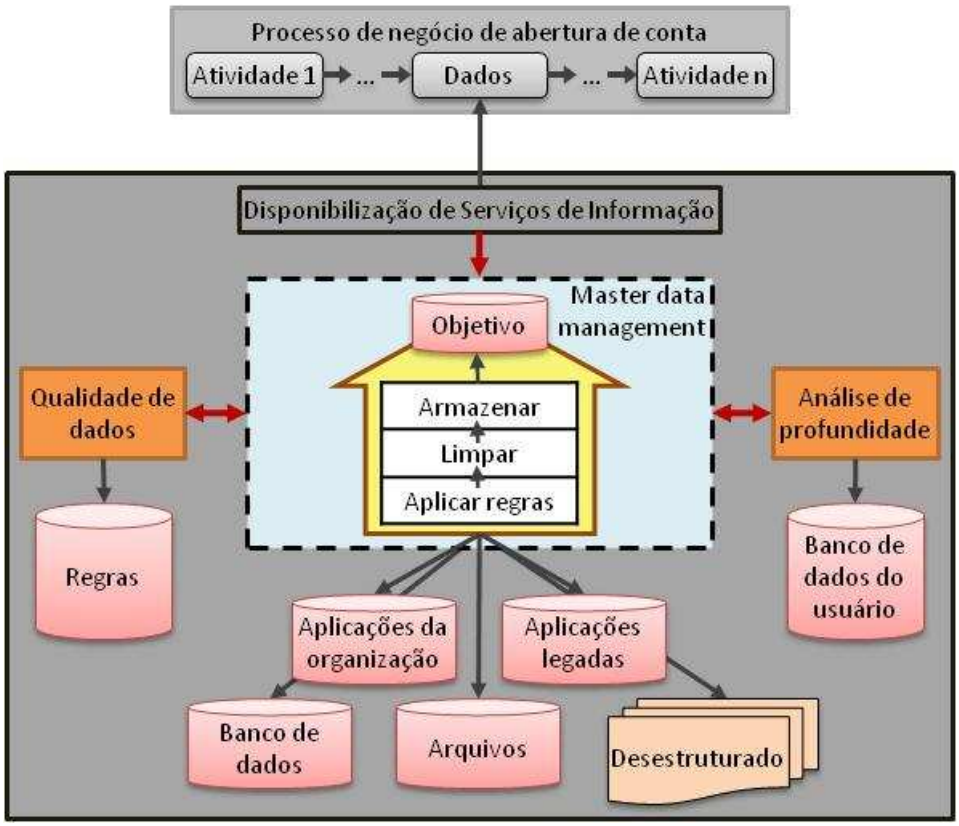

Figura 29 – Master data management

7.5.1.8 Gerência do ciclo de vida

É importante gerenciar o uso de recursos, o uso dos serviços e o desempenho dos serviços. Para isso, algumas atividades devem ser adotadas:

- isolar problemas;
- trabalhar com atividades preventivas a fim de determinar o problema, isolar e resolver;
- entender dependências dos serviços com recursos em particular;
- manter e acompanhar SLA (*Service Level Agreement* – nível de acordo de serviços);
- monitorar processos de negócio a fim de saber o desempenho dos serviços.

8 Data Service Layer

O conceito de SOA foi introduzido a fim de obter baixo acoplamento, gerenciamento mais adequado, controle operacional, encapsulamento de heterogeneidade, coexistência de sistemas em multi-plataformas etc. Por outro lado, muitas aplicações estão se tornando cada vez mais complexas.

Em uma organização, podemos encontrar dados em diferentes formatos: XML, texto delimitado, arquivos binários, dados relacionais, dados objeto-relacionais. Dis-

ponibilizar para as aplicações todos estes tipos de dados dispersos e heterogêneos não é algo fácil. São grandes os desafios para encapsular sistemas de informação da empresa (EIS – *Enterprise Information Systems*) ou bancos de dados locais. O que se deseja é ter uma arquitetura para gerência de dados que trate dos aspectos de integração expondo dados através de diferentes serviços autônomos, com o objetivo de obter simplicidade e acelerar o projeto [Davydov, 2005].

8.1 Opções de integração de dados tradicionais

Em geral, um departamento de TI cria um framework para tratar os requisitos de integração de dados baseado nas seguintes opções:

- Aplicações do negócio acessam as fontes de dados através de *stored procedures* (Figura 30);
- Aplicações do negócio acessam aplicações legadas ou outras aplicações construídas explicitamente para acessar as fontes de dados e disponibilizar os dados (Figura 31);
- Aplicações acessam as bases de dados via componentes de integração (EAI – *Enterprise Application Integration*) (Figura 32).

Figura 30 – Aplicações acessam diretamente as fontes de dados através de *stored procedures*

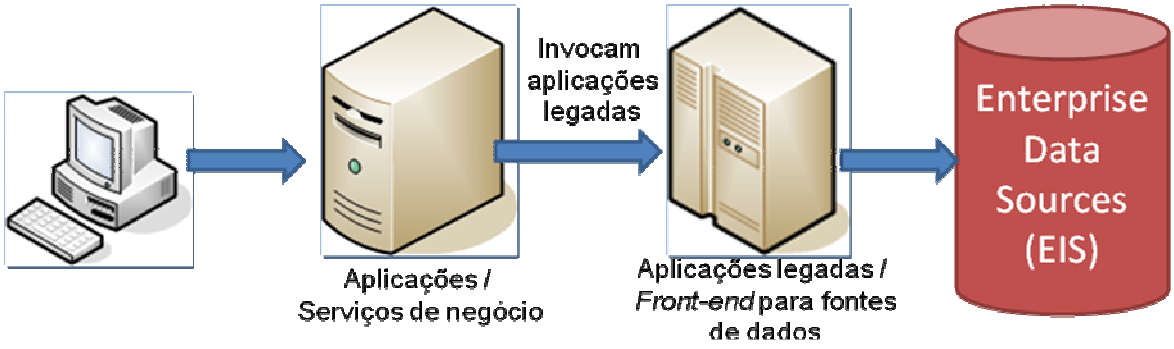

Figura 31 – Aplicações acessam diretamente aplicações legadas que acessam os dados

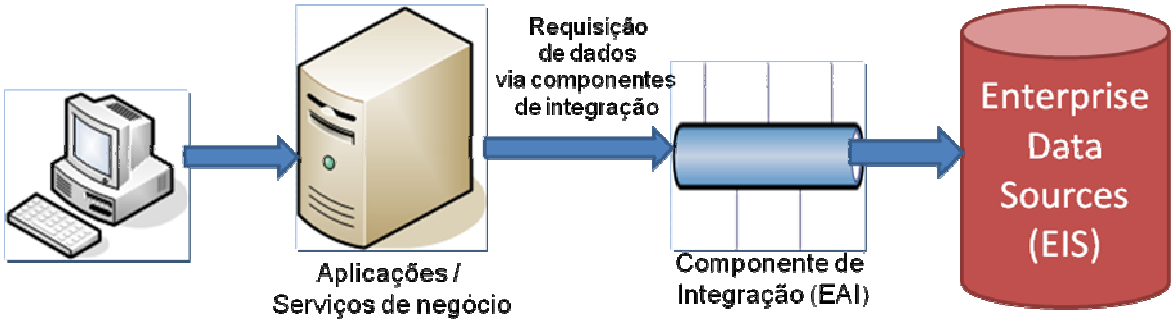

Figura 32 – Aplicações se conectam a um componente de integração

A reengenharia de acesso a dados não aparece usualmente na implantação de abordagens SOA. A tendência é continuar usando *frameworks* tradicionais de integração de dados e incluir uma nova camada de *web services* para acessá-los. As formas mais utilizadas são:

- Aplicações do negócio acessam *web services* que acessam aplicações legadas ou outras aplicações construídas explicitamente para acessar as fontes de dados (Figura 33);
- Aplicações acessam *web services* que acessam as bases de dados via componentes de integração (EAI – *Enterprise Application Integration*) (Figura 34).

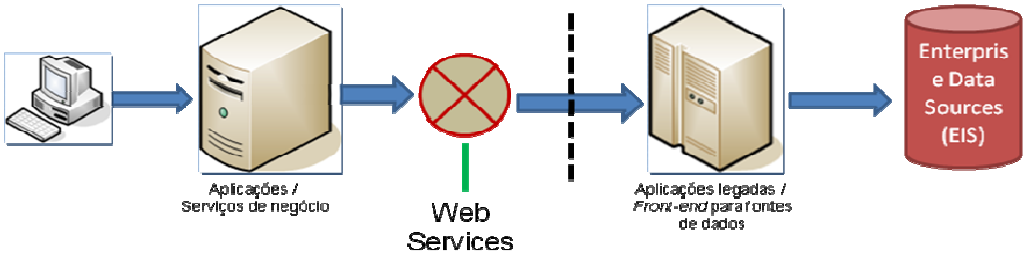

Figura 33 – Aplicações acessam *web services* que disponibilizam os dados através de aplicações legada

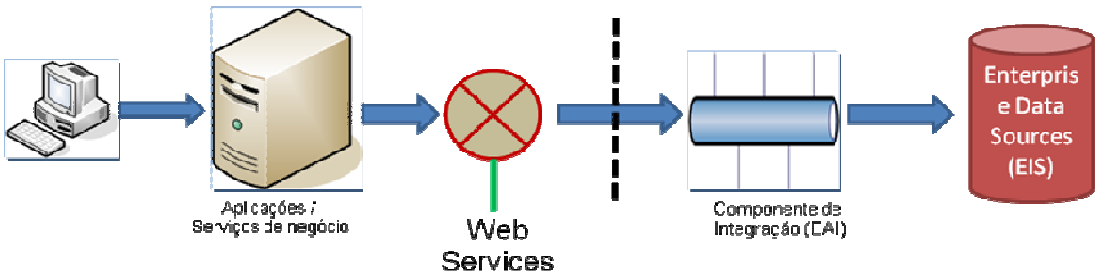

Figura 34 – Aplicações acessam *web services* que disponibilizam os dados através de componentes de integração

Encapsular acesso a dados via *web services* utilizando frameworks tradicionais de integração deixa transparente modelo de dados e mecanismo de consultas. A aplicação do negócio não precisa saber como o acesso a dados está implementado, bem como em quais plataformas os dados residem. Todavia, esta abordagem inclui desafios, tais como:

- Utilizar SOAP para acesso aos dados;

- Mapear XML para estrutura de dados, e vice-versa;
- *Parser* e mapeamento de documentos XML, ao invés de trabalhar com conexões às fontes de dados via JDBC e ODBC e processar conjuntos de registros.

Ao se empregar SOA surgem novos clientes, antigamente inesperados (pela maior facilidade de acesso), pois para acessar os dados, basta se registrar como cliente dos serviços. A otimização do desempenho de acesso passa a não depender mais apenas de otimizar o banco de dados.

Acessar dado, nesta abordagem, é igual a acessar outro serviço. A lógica da aplicação continua consumindo muito recurso de desenvolvimento (serviços de negócio e serviços de acesso a dados). Deve-se ter cuidado com a disponibilização dos web services para acesso aos dados a fim de não ter uma solução que gerem uma grande quantidade *web services*. Isto pode trazer problemas de desempenho e escalabilidade. Deve-se ficar atento, pois a reengenharia de acesso a dados quando indo para SOA deve ser gerenciada apropriadamente. É neste ambiente que surge a proposta do uso da camada de acesso a dados (DSL – *Data Service Layer*).

8.2 Definição de Data service layer

DSL (*Data Service Layer*) é uma camada de software que gerencia a disponibilização de dados entre serviços de negócio e as múltiplas fontes de dados que estes serviços utilizam. Com DSL, serviços de negócio vêem os dados através de um modelo de dados comum (por exemplo, esquema relacional) usando algum padrão de linguagem, ou padrão de interface ou padrão de transporte. A Figura 35 abaixo ilustra o DSL.

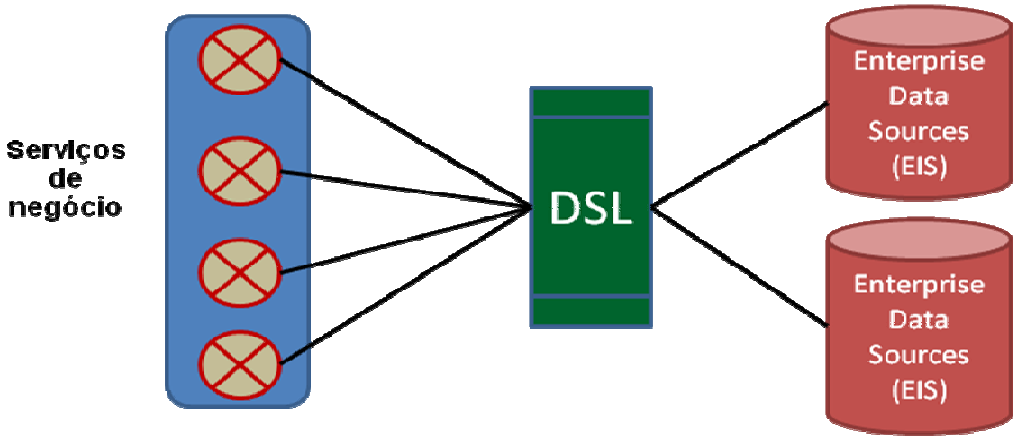

Figura 35 – *Data service layer*

O objetivo da DSL em SOA é prover um ponto único de acesso a todas as operações de leitura e escrita de dados, garantir que dados sejam obtidos sempre de uma fonte única e segura, garantir que dados sejam interpretados corretamente e garantir que dados continuem íntegros após passarem por todos os serviços. Os níveis de abstrações de serviços de dados e os padrões de acesso a dados apresentados na seção 7 devem ser considerados na construção da DSL.

A DSL é uma ponte entre a camada de serviços e a camada de persistência e provê visão dos modelos de dados embutidos nas aplicações.

A DSL pode ajudar a reduzir a complexidade, pois encapsula conectividade às fontes de dados, segurança das fontes de dados, mapeamento de dados e resolução de diferenças taxonômicas e semânticas entre fontes de dados, e modelos de dados subja-

centes. Ela trata fontes de dados não estruturados, trata todos os detalhes para prover dados aos serviços de negócio, executa todas as operações padrões (criar, ler, pesquisar e remover), garante integridade de transações e trata exceções e relata as mesmas.

Além disso, a DSL centraliza todo código de serviços de dados, resultando em uma solução SOA adaptável e de alto nível. Ela simplifica o projeto, pois todo o código para prover determinado dado aos serviços de negócio é feito apenas uma vez e o acesso a dados é centralizado em um tipo uniforme de acesso a dados para todas as aplicações e não para cada *web service* que se conecta às fontes de dados (o que gera redução do número de interações entre *web services* e o número de conexões abertas ao mesmo tempo).

Por fim, proporciona ganho de escalabilidade, pois permite o aumento do número de *web services* executando concomitantemente, podendo aumentar o número de clientes às bases de dados.

Todavia, a DSL deve usada quando o número de serviços e fontes de dados têm certo tamanho. Usualmente podemos considerar que mais do que 50 serviços ou mais do que 10 fontes de dados seria um ambiente com complexidade suficiente para se empregar uma DSL. Logicamente, isto pode variar de organização para organização. Caso contrário, continuar usando conectores às fontes de dados ou *web services*.

8.2.1 Arquitetura da DSL

Uma DSL pode ser dividida em duas camadas: camada de integração e camada de orquestração. A Figura 36 ilustra a arquitetura da DSL.

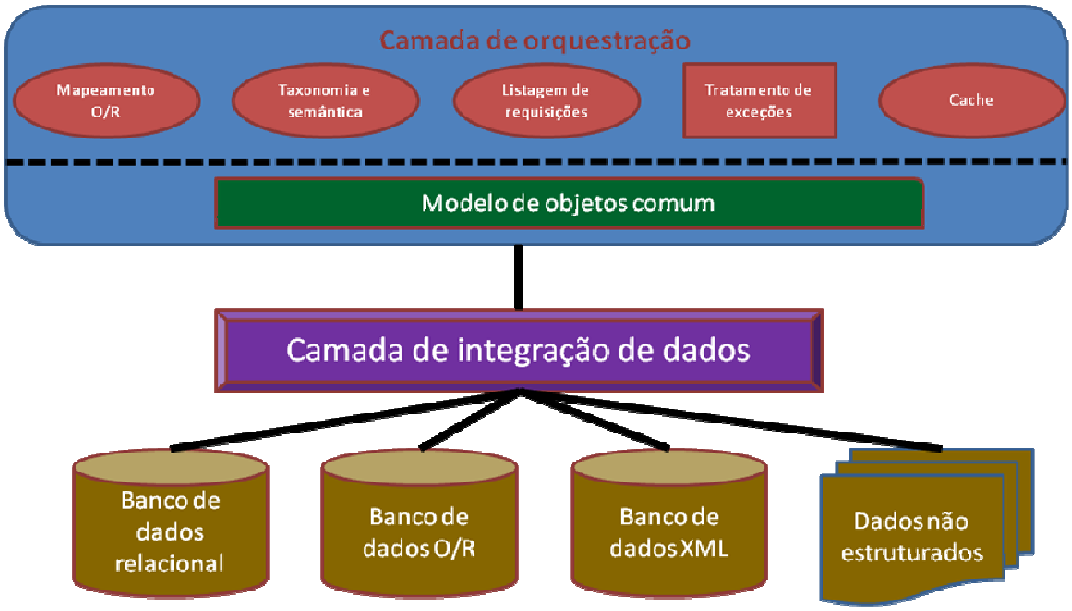

Figura 36 – Arquitetura Data service layer

A camada de integração pode ser implementada de duas formas: utilizando o padrão de consolidação de dados ou utilizando o padrão de federação de dados.

Na primeira opção de implementação 1, objetiva-se consolidar dados em uma única fonte de dados (ODS – *Operation Data Store*). Esta abordagem trás as seguintes vantagens: rapidez, alta disponibilidade, acesso integrado às informações, desempenho, integridade. Por outro lado, esta abordagem pode ter as seguintes desvantagens: necessidades adicionais de administração, servidores e recursos de armazenamento;

desafio de se ter que sincronizar dados da cópia com as fontes de dados; dificuldade de tratar dados não estruturados, áudio e imagens, neste cenário; pode gerar alto acoplamento entre DSL, ODS e fontes de dados.

Na segunda opção de implementação (federação de dados), os dados são distribuídos, sendo acessados como uma visão única. Esta abordagem pode trazer os mesmos benefícios de uma base consolidada, mas traz questões de desempenho: disponibilização de *cache* de dados a fim de reduzir número de acessos para gerenciar grandes volumes de dados, disponibilizar compressão de dados que trafegam na rede, codificar arquivos XML em binário.

A camada de orquestração tem o objetivo de simplificar a camada de integração e provê funcionalidades especiais a fim de facilitar as implementações SOA, tais como: mapeamento Objeto/Relacional, componentes para tratar taxonomia e semântica, gerência de requisições, tratamento de exceções e cache.

As principais questões a serem consideradas quando pretende-se empregar uma camada de acesso a dados são a definição do modelo de dados apropriado, o levantamento e definição de serviços de dados que proporcione reuso e a definição do modelo de objetos comum, que normalize múltiplos domínios da aplicação. Além disso, é importante que serviços tenham a granularidade adequada, a fim de garantir características de desempenho.

8.3 Administração de serviços de informação

Os serviços de informação precisam ser administrados para agilizar a manipulação da informação, garantir que dados estejam atualizados, consistentes, íntegros e disponíveis, permitir reuso do acesso à informação, suportar escalabilidade e garantir padrões e alinhamento à estratégia da organização no desenvolvimento de serviços de acesso à informação. Neste ambiente surge a necessidade do papel para administração de serviços.

A administração de serviços pode ser definida como a função responsável por desenvolver e administrar de forma centralizada as estratégias, políticas, procedimentos e práticas para o processo de gerência dos serviços de informações, incluindo planos para sua definição, padronização, organização, proteção, localização, divulgação e utilização. As vantagens trazidas pela administração dos serviços de informação são:

- Reuso de serviços;
- Informações expostas de forma precisa, completa e em tempo hábil;
- Informações compartilhadas com significado (glossário) e estrutura (modelo canônico) bem definidos trazendo facilidade no entendimento da informação e facilidade no desenvolvimento de novas aplicações;
- Qualidade da informação;
- Informações de fontes heterogêneas sendo acessadas por meio de uma interface única (serviços);
- Dados integrados, através do acesso via serviços;
- Informações consistentes e confiáveis;
- Reuso dos serviços facilita o desenvolvimento e a manutenção de sistemas.

Definir uma camada de serviços para acesso aos dados é de suma importância para conseguir encapsular fontes de dados, facilitar integração de dados e possibilitar reuso. Além disso, deve-se decidir qual a melhor arquitetura a ser empregada na organização e quais padrões de acesso a dados devem ser empregados. Neste ambiente, surge a importância da figura do administrador de serviço de dados, o qual garantirá que todos os objetivos serão alcançados quando da implantação da camada de acesso a dados.

9 Conclusão

Este relatório apresentou os principais conceitos relacionados à Service Oriented Architecture (SOA – Arquitetura Orientada a Serviços). As motivações para a implantação de uma Arquitetura Orientada a Serviços foram apresentadas.

Os principais conceitos de SOA necessários para o entendimento desta abordagem foram caracterizados, tais como: definição de SOA, caracterização detalhada de serviços e ciclo de vida de serviços, principais elementos de SOA, desafios, padrões, Enterprise Service Bus (principal infra-estrutura para SOA) e tecnologias relacionadas.

Como o principal objetivo do projeto de pesquisa está relacionado ao desenvolvimento de serviços a partir da modelagem de processos, este assunto foi abordado no Capítulo 5.

No Capítulo 6 apresentamos uma metodologia SOA da *IDS Scheer* [Klückmann, 2007] na qual a hierarquia de serviço técnico é construída ao mesmo tempo em que a hierarquia de processos de negócio.

Outro conceito importante no escopo deste projeto é o de serviço de informação, o qual foi detalhado no Capítulo 7 . Além disso, no Capítulo 8 , apresentamos uma proposta de arquitetura para construção de serviços de informação denominada de DSL (Data Service Layer, Camada de Serviços de Dados).

10 Agradecimentos

Este trabalho não seria possível sem a contribuição de pesquisadores em Sistemas de Informação e da parceria com a Petrobras, principalmente a área TIC/TIC-E&P/GDIEP. Em especial, agradecemos aos professores e alunos que colaboraram nas discussões e desenvolvimento de pesquisas, testes e desenvolvimentos necessários ao projeto. Dentre os agradecimentos à academia, se destaca o papel dos profissionais do NP2Tec[1] que contribuíram, técnica ou administrativamente, para o sucesso de nossas atividades.

A condução e os resultados deste trabalho são uma exemplar evidência de como a relação entre as universidades e as empresas pode contribuir para a geração de conhecimento útil e, desta forma, contribuir para nossa sociedade.

[1] Site do NP2Tec: http://www.uniriotec.br/~np2tec

11 Glossário

2PC – Uma abordagem para manter a consistência entre múltiplos sistemas. Na primeira fase, todos os nós são consultados para confirmar uma requisição de alteração. Na segunda fase, a confirmação das atualizações é executada. De acordo com os princípio de baixo acoplamento, a compensação em SOA é geralmente usada, ao invés do 2PC.

API - (Application Program Interface) - um conjunto de rotinas, protocolos e ferramentas para a construção de aplicativos de software. Uma API de boa qualidade deve tornar fácil o desenvolvimento de um programa, através do uso de blocos já construídos, bastando para o programador pôr estes blocos em sua aplicação. Muitos sistemas operacionais, como o MS-Windows, disponibilizam APIs para que os programadores possam criar aplicações consistentes com o sistema operacional. Embora as APIs sejam feitas para programadores, elas são boas para os usuários finais, pois garantem que todos os programas que usem uma API em comum tenham uma interface similar, facilitando o aprendizado de usuários de programas novos.

Correlation set – É um mecanismo de BPEL para prover a correlação entre mensagens assíncronas baseado no conteúdo da mensagem.

HTTP – Do Inglês HyperText Transfer Protocol. Protocolo usado para transferir páginas Web entre um servidor e um cliente (por exemplo, o browser).

Macroprocessos – É o nível mais alto de agregações de ações que a organização desenvolve para projetar, produzir, comercializar, entregar e sustentar seus produtos e/ou serviços, realizando sua missão e os objetivos estratégicos para obtenção e manutenção da vantagem competitiva.

MDA (Model-Driven Architecture) - é um arcabouço para desenvolvimento de sistemas. Ele visa separar as decisões orientadas ao negócio das decisões de plataforma permitindo, assim, maior flexibilidade durante as fases de especificação e de desenvolvimento de sistemas [Mukerji e Miller, 2003].

Processo de negócio – É um conjunto estruturado de atividades modeladas para produzir uma saída específica para um cliente ou mercado particular. Ele Implica em uma forte ênfase em como o trabalho é feito por dentro de uma organização, em contraste com o foco apenas no produto [Davenport, 1992].

Regra de negócio – É uma expressão que define ou restringe alguns aspectos do negócio [The Business Rules Group, 2000].

Serviço composto – É um serviço complexo, criado a partir da composição de serviços simples ou de outros serviços compostos.

Serviço de negócio – O termo serviço de negócio é usado para representar os produtos ou serviços que um componente de negócio oferece para outro componente de negócio (interno ou externo) [Cherbakov *et al*., 2005].

Serviço técnico – É um requisito não funcional que pode ser atendido em tempo de execução através da adaptação da configuração dos recursos selecionados [Caromel *et al*., 2006].Serviço básico: É um serviço atômico, responsável por cumprir uma tarefa simples e específica em um processo de negócio.

SOA – Do inglês Service-oriented Architecture. É um estilo de arquitetura de software cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços.

SOAP – Do Inglês Simple Object Access Protocol. É um protocolo para intercâmbio de mensagens entre programas de computador. SOAP é um dos protocolos usados na criação de Serviços Web. Geralmente servidores SOAP são implementados utilizando-se servidores HTTP pré-existentes, embora isto não seja uma restrição para funcionamento do protocolo. A princípio, as mensagens SOAP podem ser convertidas usando qualquer protocolo contanto que as conexões sejam definidas. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C. Elas são recebidas por um serviço em tempo de execução (um SOAP *listener*), que aceita a mensagem, extrai o corpo da mensagem XML, transforma a mensagem XML em um protocolo nativo e delega a requisição ao processo de negócio atual dentro da organização.

UDDI – Do Inglês Universal Description, Discovery and Integration. É um protocolo baseado em XML que provê um diretório distribuído com listas de negócios na Internet e descoberta destes serviços.

Web service (Serviço Web) – É um conjunto de protocolos e padrões que permitem que aplicações se comuniquem, mesmo que rodando em distintas plataformas.

WSDL – Do inglês Web Service Definition Language. Padrão baseado em XML para descrever o web service (operações oferecidas, dados de entrada e saída, etc.), além de ser usado para a validação das chamadas das operações.

XML – É a sigla usada para eXtensible Markup Language, ou seja, Linguagem de Marcação Expansível. A linguagem XML foi criada pelo W3C (World Wide Web Consortium) como um substituto mais poderoso para a HTML. Com o tempo viu-se que ela era muito mais do que isso. Hoje o mundo caminha rumo ao XML para a troca de informações.

Referências Bibliográficas

ARKIN A., et al., 2002, **Web Service Choreography Interface 1.0, W3C**.

ARKIN A., et al., 2002b, **Business Process Modeling Language-BPML 1.0.**

ARSANJANI, A., 2004, **Service-oriented modeling and architecture: how to identify, specify, and realize services for your SOA**. IBM – whitepaper. Disponível em <ftp://www6.software.ibm.com/software/developer/library/ws-soa-design1.pdf>. Acesso em: 10 jun. 2008.

BLOOMBERG, J., SCHMELZER, R., 2006. **Service Orient or Be Doomed!: How Service Orientation Will Change Your Business.** Hoboken, NJ: John Wiley & Sons.

BYRNE, B., KLING, J., MCCARTY, D., SAUTER, G., SMITH, H., Worcester, P., 2008a, **The value of applying the data quality analysis pattern in SOA**. Disponível em <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0804sauter/?S_TACT=105AGX11&S_CMP=ART>. Acesso em: 10 jun. 2008.

BYRNE, B., KLING, J., MCCARTY, D., SAUTER, G., WORCESTER, P., 2008b, **The value of applying the canonical modeling pattern in SOA**. Disponível em <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0803sauter/?S_TACT=105AGX11&S_CMP=ART>. Acesso em 10 jun. 2008.

BYRNE, B., KLING, J., SAUTER, G., WORCESTER, P., 2008c, **The value of applying the business glossary pattern in SOA**. Disponível em <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0802sauter/?S_TACT=105AGX11&S_CMP=ART>. Acesso em: 10 jun. 2008.

BYRNE, B., MCCARTY, D., SAUTER, G., WORCESTER, P., KLING, J., 2008d, **Introduction to the information perspective of a Service Oriented Architecture**. Disponível em <http://www.ibm.com/developerworks/db2/library/techarticle/dm-0801sauter/>. Acesso em: 10 jun. 2008.

CAROMEL, D., DELBE, C., DI COSTANZO, A., 2006, **Peer-to-peer and fault-tolerance: Towards deployment based technical services**.

CARTER, S., 2007, **The new language of business: SOA & Web**, IBM Press.

CHARFI, A., MEZINI, M., 2004, **Hybrid Web Service Composition: Business Processes Meet Business Rules**, ICSOC'04, New York, New York, USA.

CHERBAKOV, L., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G., 2005, **Impact of service orientation at the business level**, Ibm Systems Journal, v. 44, n. 4. Disponível em <http://www.hec.unil.ch/myoo/soc/support/lectures/soa_business_level.pdf>. Acesso em: 10 jun. 2008.

CURBERA F., et. al., 2003, **Business Process Execution Language for Web Services, version 1.1**, May.

DAVENPORT, T., 1992: **Process Innovation: Reengineering work through Information Technology, Harvard Business School**.

DAVYDOV, M., "**How robust data layers accelerate SOA implementations developerWorks**", 2005. Disponível em <http://www.ibm.com/developerworks/webservices/library/ws-soa-adventure3/>. Acesso em: 10 jun. 2008.

ELMASRI, R., NAVATHE, S. B., 2007, **Fundamental of Database Systems**, Addison Wesley.

ERL, T., 2005, **Service-Oriented Architecture: concepts, technology, and Design**, Prentice Hall.

HANSEN, M., 2007, **Basic SOA Using REST.** Disponível em <http://www.webreference.com/programming/basic_soa/index.html>. Acesso em: 10 jun. 2008.

HEFFNER, R., FORRESTER RESEARCH - Making leaders successful every day, 2007, **How to build your SOA Plataform**. Disponível em <http://www.forrester.com/rb/research>. Acesso em: 10 jun. 2008.

IDS SCHEER, **ARIS Solutions**. Disponível em <http://www.ids-scheer.com/en/Solutions/ARIS_Solutions/5740.html>. Acesso em: 10 jun. 2008.

IBM, 2000, **The IBM WebSphere software platform and patterns for e-business - invaluable tools for IT architects of the new economy**, IBM White Paper. Disponível em <http://www4.ibm.com/software/info/websphere/docs/wswhitepaper.pdf>. Acesso em: 10 jun. 2008.

JOSUTTIS, N. M., 2007, **SOA in practice: The Art of Distributed System Design**. O'Reilly.

KLÜCKMANN, J., 2006, **On the way to SOA. ARIS Expert Paper**. Disponível em <http://www.ids-scheer.com/set/6473/ARIS_Expert_Paper-SOA-Way_to_SOA_-_Klueckmann_2006-09_en.pdf>. Acesso em: 10 jun. 2008.

KLÜCKMANN, J., 2007, **10 Steps to Business-Driven SOA, ARIS Expert**. Disponível em <http://www.ids-scheer.com/set/6473/ARIS_Expert_Paper-SOA-10_Steps_to_SOA_Klueckmann_2007-03_en.pdf>. Acesso em: 10 jun. 2008.

KRAFZIG, D., BANKE, K., SLAMA, D., 2004, **Enterprise SOA: Service-Oriented Architecture Best Pratices**.

MALINVERO, P., 2006, **Service-Oriented Architecture Craves Governance**, Gartner. Disponível em <http://www.gartner.com/DisplayDocument?id=488180>. Acesso em: 10 jun. 2008.

MANES, A. T., 2007, Handouts of a talk at OOP 2007. SIGS Datacom.

MARKS, E. A.; BELL, M., 2006, **Service-Oriented Architecture: a planning and implementation guide for business and techonology**, John Willey & Sons Inc.

MICROSOFT, **A Blueprint for Building Web Sites Using the Microsoft Windows DNA Platform**, Microsoft White Paper. Disponível em <http://www.microsoft.com/commerceserver/techres/whitepapers.asp.>. Acesso em: 10 jun. 2008.

MICROSOFT, **Princípios do design de serviço: padrões e antipadrões de serviço**, 2005. Disponível em <http://www.microsoft.com/brasil/msdn/Tecnologias/arquitetura/SOADesign_US.mspx>. Acesso em: 10 jun. 2008.

MUKERJI, J., MILLER, J., 2003, **MDA Guide**. Disponível em <http://www.omg.org/docs/omg/03-06-01.pdf>. Acesso em: 10 jun. 2008.

PAPAZOGLOU, MIKE P.; HEUVEL, WILLEM-JAN, 2007, **Service oriented architectures: approaches, technologies and research issues**, VLDB Journal, Springer-Verlag.

PEREPLETCHIKOV, M., RYAN, C., TARI, Z., 2005, **The Impact of Software Development Strategies on Project and Structural Software Attributes in SOA**, On the Move to Meaningful Internet Systems 2005: OTM Workshops, 442-451, Lecture Notes in Computer Science, 11 de outubro. Disponível em

<http://goanna.cs.rmit.edu.au/~caspar/downloads/INTEROP2005paper.pdf>. Acesso em: 10 jun. 2008.

SCHEPERS, T., IACOB, M., VAN ECK, P., 2008, **A lifecycle approach to SOA governance**, SAC'08, Fortaleza, Ceará, Brasil. Disponível em <http://eprints.eemcs.utwente.nl/12165/01/Schepers_SAC2008_final.pdf>. Acesso em: 10 jun. 2008.

SOA CONSORTIUM, 2008a. Disponível em www.soa-consortium.org. Acesso em: 10 jun. 2008.

SOA CONSORTIUM, 2008b, **Valero Energy: Mergers and Acquisitions: Bringing Acquisitions Online Faster**. Disponível em <http://www.soa-consortium.org/cs/Business_Lead/valero-energy.htm>. Acesso em: 27 jun. 2008.

SOA CONSORTIUM, 2008c, **Automobile #1: SOA Governance Increasing Agility Through a Governance Focus**. Disponível em <http://www.soa-consortium.org/cs/Business_Lead/automobile.htm>. Acesso em: 10 jun. 2008.

THE BUSINESS RULES GROUP, 2000, Defining **Business Rules, What are they really?** Disponível em <http://www.businessrulesgroup.org>. Acesso em: 01 jul. 2008.

VALERO ENERGY. Disponível em <http://www.valero.com/>, Acesso em: 10 jun. 2008.

VIOLINO, B., CIO – Gestão, estratégias e negócios em TI para líderes corporativos, 2008, **Como navegar no mar de padrões SOA**. Disponível em <http://cio.uol.com.br/gestao/2008/01/09/idgnoticia.2008-01-09.3145720442/>. Acesso em: 10 jun. 2008.

VON HALLE, B., 2001, **Business Rules Applied: Building Better Systems using the Business Rules Approach**, Wiley.

W3C, BOOTH, D., HAAS, H., McCABE, F., NEWCOMER, E., CHAMPION, M., FERRIS, C., ORCHARD, D., 2004, **Web Services Architecture**, W3C Working Group Note 11 February 2004. Disponível em < http://www.w3.org/TR/ws-arch/#engaging>. Acesso em: 10 jun. 2008.

W3C, CHRISTENSEN, E., CURBERA, F., MEREDITH, G., WEERAWARANA, S., 2001, **Web Services Description Language (WSDL) 1.1**, World Wide Web Consortium Note.

Disponível em <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>. Acesso em: 10 jun. 2008.

W3CSCHOOLS, **Web Services Tutorial**, Disponível em <http://www.w3schools.com/webservices/default.asp>. Acesso em: 10 jun. 2008.

WS-SECURITY, NADALIN, A., KALER, C., MONZILO, R., HALLAM-BAKER, P., 2004, **Web Services Security v1.1**. Disponível em <http://www.oasis-open.org/committees/download.php/16790/wssv1.1-spec-os-SOAPMessageSecurity.pdf>. Acesso em: 10 jun. 2008.