VINICIUS DIAS OLIVEIRA

ABORDAGEM ORIENTADA A SERVIÇOS PARA UMA IMPLEMENTAÇÃO DE SISTEMAS DISTRIBUÍDOS

ASSIS
2009

ABORDAGEM ORIENTADA A SERVIÇOS PARA UMA IMPLEMENTAÇÃO DE SISTEMAS DISTRIBUÍDOS

VINICIUS DIAS OLIVEIRA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Ms. Douglas Sanches da Cunha

Analisador: Dr. Almir Rogério Camolesi

ASSIS
2009

VINICIUS DIAS OLIVEIRA

ABORDAGEM ORIENTADA A SERVIÇOS PARA UMA IMPLEMENTAÇÃO DE SISTEMAS DISTRIBUÍDOS

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

ORIENTADOR: Ms. Douglas Sanches da Cunha

ÁREA DE CONCENTRAÇÃO: Java, Sistemas Distribuídos

ASSIS
2009

DEDICATÓRIA

Dedico este trabalho à minha família e amigos,
pois sempre deram a força necessária para
que eu não desistisse no meio do caminho.

AGRADECIMENTOS

Ao professor e amigo, Douglas Sanches da Cunha, pela orientação e pelo estímulo transmitido durante o trabalho.

Aos amigos, Jabes Felipe Cunha, Larissa Piovezani, Rafael Bevilacqua Mello, Rodrigo Merlin e Simone Cardoso da Silva e a todos que colaboraram direta ou indiretamente, na execução deste trabalho.

Aos familiares, Ana Cláudia Dias e Cleny de Lourdes Sant'Ana Dias.

RESUMO

O que se desejou investigar é o desenvolvimento de projetos orientados a serviços. Tal assunto necessita certa base para ser conhecido, no caso, Sistemas Distribuídos, Web Services e SOA (*Service Oriented Architecture*). O mundo dos Sistemas Distribuídos oferece diversas soluções para a comunicação entre diferentes dispositivos para diferentes propósitos, mas, para uma aplicação ser distribuída, deve atender a algumas necessidades, a fim de evitar problemas durante seu funcionamento. Este trabalho se preocupou em estudar a implementação de tais sistemas através do uso de Web Services, devido à grande utilização da tecnologia no mercado de trabalho. O estudo de serviços com Web Services, leva ao estudo de SOA, pois é necessário entender como fazer o design de projetos orientados a serviços, para que todos os requisitos sejam atendidos pelos serviços. Então, como estudo de caso, a implementação de uma aplicação simples foi desenvolvida, buscando dados através de serviços.

Palavras-chave: Sistemas Distribuídos, Web Service, SOA, Arquitetura Orientada a Serviços, Orientação a Serviços.

ABSTRACT

The aim of this research is studying the development of service oriented projects. For that, it is needed a certain base of knowledge of things like Distributed Systems, Web Services and SOA (Service Oriented Architecture). The world of Distributed Systems offers various solutions for communication of different types of machines, but, for an application to be distributed, it must follow some requirements, in order to avoid problems during runtime. This research worried about studying the development of such applications using Web Services, once it's something that is being used a lot by development companies. The study of services with Web Services, takes to the study of SOA, because understanding how to design service oriented projects is necessary, so that all the requirements are fulfilled by the services. So, as study case, the development of a simple application that uses services to acquire data was realized.

Keywords: Distributed Systems, Web Service, SOA, Service Oriented Architecture, Service Orientation.

LISTA DE ILUSTRAÇÕES

Figura 01. Passos para iteração com um Web Service 20
Figura 02. Modelos da arquitetura Web Service.....21
Figura 03. Relacionamento das entidades do modelo orientado à mensagens.....22
Figura 04. Relacionamento das entidades do modelo orientado a serviços.....23
Figura 05. Relacionamento das entidades do modelo orientado a recursos.....24
Figura 06. Relacionamento das entidades do modelo orientado à política.....24
Figura 07. Domínios de lógica de negócio e aplicação.....27
Figura 08. Camada de serviço entre as demais melhora lógica de negócio e de aplicação.....28
Figura 09. A camada de serviço abstrai a conectividade de ambientes de implantação de serviços.....29
Figura 10. Fases em comum de projetos SOA.....30
Figura 11. Fases comuns da estratégia TOP-DOWN.....33
Figura 12. Fases comuns da estratégia BOTTOM-UP.....35
Figura 13. Fases comuns da estratégia AGILE.....37
Figura 14. Arquitetura da aplicação do estudo de caso.....40
Figura 15. Diagrama UML das classes de mapeamento da aplicação.....40
Figura 16. Tela inicial da aplicação web.....42
Figura 17. Tela de busca de currículos.....42
Figura 18. Tela de gerenciamento do currículo.....43
Figura 19. Tela de atualização dos dados do currículo.....43
Figura 20. Tela de inclusão e alteração dos dados do currículo.....44
Figura 21. Tela de atualização dos dados do ex-aluno.....44
Figura 22. Tela de busca da aplicação cliente.....45
Figura 23. Currículo gerado pela aplicação.....45

SUMÁRIO

1. INTRODUÇÃO.....10
2. SISTEMAS DISTRIBUÍDOS.....12
2.1. HETEROGENEIDADE.....12
2.2. SISTEMAS ABERTOS.....13
2.3. SEGURANÇA.....14
2.4. ESCALABILIDADE.....15
2.5. TRATAMENTO DE FALHAS.....15
2.6. CONCORRÊNCIA.....16
2.7. TRANSPARÊNCIA.....17
3. WEB SERVICES.....19
3.1. ARQUITETURA DE UM WEB SERVICE.....19
3.1.1. Visão Geral do Funcionamento de um Web Service.....20
3.1.2. Modelos Arquiteturais de um Web Service.....21
4. SOA – ARQUITETURA ORIENTADA A SERVIÇOS.....26
4.1. ORIENTAÇÃO A SERVIÇOS E O AMBIENTE EMPRESARIAL.....26
4.2. DESENVOLVIMENTO DE UM PROJETO ORIENTADO A SERVIÇOS. .30
3.2.1. Fases Básicas de um Projeto SOA.....30
4.2.2. Estratégias de implantação de um projeto SOA.....32
4.2.2.1. Estratégia TOP-DOWN.....32
4.2.2.2. Estratégia BOTTOM-UP.....34
4.2.2.3. Estratégia AGILE.....36
5. ESTUDO DE CASO.....39
5.1. VISÃO GERAL DA APLICAÇÃO.....39
5.2. DESCRIÇÃO DO DESENVOLVIMENTO.....41
6. CONCLUSÃO.....46
REFERÊNCIAS BIBLIOGRÁFICAS.....47

1. INTRODUÇÃO

Durante a evolução da espécie humana, a comunicação entre aqueles que nela pertencem, tem desempenhado um papel importante, contribuindo com o seu desenvolvimento.

Para tal, a comunicação tem sido feita de diversos modos, um deles sendo através da tecnologia desenvolvida pelo homem. Dentro dessa tecnologia, um dos meios utilizados foram as redes de computadores, que possibilitam a troca de informações entre pontos remotos do planeta em tempo real.

Mas, uma vez que o crescimento da espécie e das redes de computadores tem sido grande, a necessidade de uma abordagem de implementação diferente das redes de computadores tornou-se indispensável, surgindo assim os Sistemas Distribuídos.

Após um tempo de sucesso dos Sistemas Distribuídos, as empresas enxergavam a oportunidade de oferecer serviços e ampliar o seu horizonte de mercado. O problema era que a implementação de tal não era tão produtiva, pois levava tempo e também existiam alguns problemas de comunicação em diferentes aplicações, o que gerou a padronização das mensagens enviadas e com isso os Web Services.

Com a grande satisfação que os Web Services trouxeram para as empresas, essa área de serviços começou a se expandir, e com ela a necessidade de organização dos mesmos. Uma arquitetura totalmente voltada a serviços foi criada, chamada de SOA (*Service Oriented Architecture*).

Essa área se expandiu de tal forma, que a quantidade de profissionais não tem sido suficiente, tornando interessante o estudo e preparação para o mercado neste contexto.

Para que uma boa preparação seje alcançada, um embasamento apenas teórico não é suficiente, tendo a prática como o aperfeiçoamento necessário. Este trabalho pesquisa a parte teórica de Sistemas Distribuídos, Web Services e SOA, focando os assuntos necessários para o aprendizado de serviços, e então a implementação de uma aplicação utilizando SOA, por meio de da implementação de Web Services.

Para o estudo teórico e prático, pesquisas em livros, artigos e documentações oficias das tecnologias serão utilizadas.

A implementação será de uma aplicação responsável pela busca e atualização dos currículos de ex-alunos de uma instituição. A busca dos currículos será realizada através de uma interface de serviço, o que possibilita que uma organização mantenha um contrato com a instituição para buscar currículos em sua base de dados. A aplicação será desenvolvida em um ambiente Java EE (*Enterprise Edition*), utilizando as ferramentas oferecidas pela tecnologia Java.

Este trabalho foi dividido em seis capítulos:

- **Introdução;**
- **Capítulo 2 - Sistemas Distribuídos:** descreve algumas características básicas da distribuição de sistemas, as quais um desenvolvedor deve se preocupar;
- **Capítulo 3 - Web Services:** a arquitetura de um Web Service é apresentada e suas partes descritas, mostrando o motivo de ser uma ótima tecnologia para o desenvolvimento de serviços;
- **Capítulo 4 - SOA - Arquitetura orientada a serviços:** como a estrutura da TI de uma empresa irá se comportar em um ambiente voltado a serviços, e a descrição das fases para o desenvolvimento de um projeto SOA, apresentando algumas estratégias, são o foco deste capítulo;
- **Capítulo 5 - Estudo de caso:** após a apresentação teórica do assunto, este capítulo se preocupa em descrever a parte prática deste trabalho, no qual uma aplicação foi desenvolvida com o objetivo de gerenciar os currículos dos ex-alunos de uma instituição;
- **Capítulo 6 - Conclusão:** as conclusões deste trabalho, expondo os resultados alcançados após o estudo teórico e prático de um projeto orientado a serviços;

2. SISTEMAS DISTRIBUÍDOS

Os Sistemas Distribuídos surgiram para melhorar o desempenho das redes de computadores, em um ambiente com grande número de usuários e trocas de informações. Baseando-se na idéia de componentes independentes e comunicação através de mensagens, os Sistemas Distribuídos oferecem uma abordagem suficientemente capaz de cumprir a sua proposta (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Para o sucesso na implementação de um Sistema Distribuído, alguns desafios devem ser levados em consideração pelos projetistas. Para cada desafio, soluções foram desenvolvidas com sucesso, o que os tornam características, e não mais desafios, dos Sistemas Distribuídos. Essas características encapsulam vários tipos de falhas que poderiam ocorrer no ambiente, e se resumem em:

- Heterogeneidade;
- Sistemas Abertos;
- Segurança;
- Escalabilidade;
- Tratamento de falhas;
- Transparência.

2.1. HETEROGENEIDADE

Os participantes de um Sistema Distribuído devem ser capazes de cumprir suas responsabilidades independentemente das suas diferenças. As diferentes características dos participantes podem ocorrer desde a concepção física até a lógica. Assim sendo, um Sistema Distribuído deve ser capaz de oferecer serviços em um nível abstrato o suficiente para superá-las.

As diferenças físicas e lógicas se resumem em tipos de hardware, software, sistemas operacionais, linguagens de programação e implementações de diferentes desenvolvedores.

Almejando uma abstração de nível suficiente o bastante para superar essas diferenças, algumas soluções foram desenvolvidas através de middleware e migração de código.

O middleware oferece uma abstração de programação, surgindo o CORBA (*Common Object Request Broker*) e Java RMI (*Remote Method Invocation*). O CORBA, oferece abstração para diferentes linguagens de programação, diferentemente do Java RMI, que suporta apenas uma linguagem de programação.

A migração de código trata a abordagem do envio de código, para que este seja executado no local de destino. Um modo de fazer isso é através de applets em Java, que independem de sistema operacional e hardware para que seja executado, necessitando apenas de uma JVM (Java *Virtual Machine*).

2.2. SISTEMAS ABERTOS

Outra característica importante de um Sistema Distribuído, é a possibilidade de ser estendido e reimplementado de diversas maneiras. Novos serviços podem ser adicionados e disponibilizados para o uso, sem a necessidade de existir apenas um fornecedor, sendo assim independentes.

Para que diversos clientes acessem os serviços existentes, é necessário que a especificação e documentação do serviço seja disponibilizada. Isso ocorre através de interfaces publicadas, as quais especificam o seu funcionamento e objetivo.

A padronização da publicação da documentação de um serviço e do modo que este será transmitido, possibilita a adição de novos participantes no ambiente com diferentes tipos de hardware e software.

2.3. SEGURANÇA

Assim como em qualquer tipo de comunicação, certas informações devem ser protegidas para que não sejam interceptadas. Em um Sistema Distribuído não é diferente, as mensagens enviadas entre os participantes devem chegar ao seu destino sem que aconteça nenhuma interceptação. A segurança em um ambiente distribuído foca três componentes: confidencialidade, integridade e disponibilidade.

Várias das informações disponíveis são destinadas a usuários específicos, o que é conhecido como confidencialidade, a qual traz consigo a idéia de autorização para acessar tais informações.

Como integridade, é interessante lembrar da existência de troca de mensagens, pela qual as informações devem chegar ao seu destino sem que tenham sofrido alterações ou danos.

A disponibilidade das informações deve ser capaz de superar as interferências com os meios de acesso aos recursos, evitando que ocorra problemas na requisição de informações.

A combinação desses três componentes de segurança, conclui que um serviço provedor de informações, possui alguns requisitos para que as informações sejam enviadas com sucesso. Dentre esses requisitos estão, a necessidade de conhecer a identificação do usuário e criptografar as mensagens, para que não ocorra nenhuma interceptação indesejada.

Além dos elementos de segurança citados acima, existem dois que não foram totalmente resolvidos, são eles o ataque de negação de serviço e a segurança de código móvel.

O ataque de negação de serviço, se trata da interrupção de um serviço por um usuário. Isso pode ocorrer através de inúmeras requisições inúteis ao mesmo serviço, causando a indisponibilidade do mesmo aos usuários que, realmente necessitam das informações providas por aquele serviço.

Para a existência de boa segurança em código móvel, sua manipulação deve ser

feita com cuidado. Por exemplo, códigos executáveis que são distribuídos em uma rede, seja através de e-mail ou outros serviços, não devem fingir que fazem uma coisa enquanto, na verdade, estão acessando recursos locais ou fazendo parte de um ataque de negação de serviço.

2.4. ESCALABILIDADE

Para o crescimento de um Sistema Distribuído, é necessário o aumento no número de seus participantes, que se tratam, basicamente, de clientes e servidores, os que requisitam e os que fornecem informações. Lembrando que um Sistema Distribuído utiliza a troca de mensagens para a comunicação, o aumento do número de participantes não deve ser um problema para a sua performance, que para manter-se consistente, deve superar alguns obstáculos.

De acordo com a necessidade de novos clientes ou servidores em um sistema, deve ser possível que o desejado seja adicionado ao sistema a um custo razoável, assim fazendo um controle do custo dos recursos físicos. Além disso, também existe a preocupação da perda de desempenho, pois não pode existir demora excessiva para a resposta à uma requisição.

A partir do momento em que as requisições aumentam, as informações não devem se esgotar, de forma a sempre atender a necessidade do cliente. Para que as requisições de informações sejam realizadas, mensagens são enviadas a diferentes destinos, o que cresce com o aumento de participantes em um sistema, assim gerando maiores listas de destinos alcançáveis, o que pode gerar gargalos de desempenho se mantidas em um local centralizado.

2.5. TRATAMENTO DE FALHAS

Sistemas formados por hardware e software estão sempre expostos à falhas. Em um Sistema Distribuído essas falhas acontecem parcialmente, afetando apenas alguns

componentes enquanto outros continuam funcionando.

Para que as falhas sejam superadas, existem maneiras de detecção e mascaramento de falhas. A detecção de falhas pode ser feita através da verificação da consistência de dados em mensagens, mas ainda deixa um desafio a ser superado, que é o gerenciamento de falhas que podem não ser detectáveis, mas que, ainda sim, são suspeitas. O mascaramento de falhas ocultam algumas delas ou a tornam menos sérias. Por exemplo, se uma mensagem encontra problemas e não é enviada, ela pode ser reenviada ou, dados que necessitam ser gravados, podem ser gravados em mais de um disco, assim caso um esteja danificado, ainda existe outro.

Mas, ainda com métodos como os apresentados, mecanismos como tolerância e recuperação de falhas são necessários. Supondo que uma mensagem encontra erros consecutivos no seu envio, ela será reenviada frequentemente ou, os diferentes discos em que um dado foi gravado podem estar danificados.

A tolerância a falhas explora a idéia de que tanto clientes quanto servidores devem ser tolerantes, assim deixando a decisão de tentar de novo para os mesmos. A recuperação de falhas trabalha com a recuperação do estado dos dados após a falha de um servidor, controlando assim a consistência dos dados, e logo, das mensagens.

2.6. CONCORRÊNCIA

O compartilhamento de serviços e aplicativos apresenta um sério problema no contexto de Sistema Distribuído, a concorrência. As características desse ambiente possibilitam que mais de um cliente tente acessar o mesmo recurso ao mesmo tempo, o que torna necessário o tratamento de uma requisição por vez.

Essa abordagem supera o problema, mas cria outro, pois esse tipo de tratamento, levaria um tempo longo e de grande influência no desempenho, uma vez que o número de requisições aumenta.

Para evitar tal situação e ainda solucionar o problema de acesso a dados compartilhados, os serviços e aplicativos devem ser capazes de processar pedidos concorrentemente, mantendo suas operações sincronizadas, garantindo a consistência dos dados.

2.7. TRANSPARÊNCIA

Apesar de um ambiente distribuído ser composto por componentes independentes, sua visão como um todo é importante para o usuário final ou desenvolvedor de aplicações. A transparência tem isso como foco, ocultando a separação dos componentes do Sistema Distribuído. A transparência pode ser dividida em oito tipos:

- Transparência de acesso: é possível, através de operações idênticas, o acesso à recursos locais e remotos.
- Transparência de localização: independentemente da localização física ou na rede de um recurso, o mesmo pode ser acessado.
- Transparência de concorrência: vários processos ocorrem concorrentemente, através de recursos compartilhados sem interferência entre eles.
- Transparência de replicação: sem conhecimento dos usuários e desenvolvedores de aplicativos, várias instâncias de um recurso são utilizadas para aumentar a confiabilidade e o desempenho.
- Transparência de falhas: as tarefas executadas por usuários ou aplicativos são capazes de serem concluídas, independentemente da falha em algum hardware ou software.
- Transparência de mobilidade: a movimentação de recursos e clientes dentro de um sistema não deve afetar a operação de usuários ou aplicações.
- Transparência de desempenho: de acordo com as alterações no sistema,

este deve ser reconfigurado para melhorar o desempenho.

- Transparência de escalabilidade: sem que ocorra alterações na estrutura do sistema ou nos algoritmos de aplicativos, o sistema e os aplicativos podem se expandir em escala.

Hoje em dia esses desafios estão sendo muito bem contornados através de diversos métodos. Dispositivos de hardware e plataformas de desenvolvimento já fornecem soluções para cada desafio, o que facilita o trabalho de um desenvolvedor e aumenta a satisfação do usuário final.

Do ponto de vista do desenvolvimento de aplicativos, diversos tipos de implementações têm sido utilizados, um dos que teve maior satisfação, foram os Web Services, focando a comunicação entre aplicações de diferentes plataformas.

3. WEB SERVICES

A grande aceitação dos Sistemas Distribuídos resultou na necessidade de uma implementação mais conveniente, que transpareça, as diferenças entre os fornecedores e requisitantes de serviços.

Seguindo a componentização dos Sistemas Distribuídos, os web services oferecem uma arquitetura que permite a troca de mensagens entre aplicativos por meio de uma interface padronizada, de acordo com um protocolo.

3.1. ARQUITETURA DE UM WEB SERVICE

Para se desenvolver um web service, é necessária a existência de alguns componentes. Dentre esses componentes estão:

- Agentes: um agente é responsável por enviar e receber mensagens, podendo ser uma parte de um software ou hardware.
- Serviços: se trata do conjunto de funcionalidades oferecidas pelos agentes.
- Requisitante: uma entidade requisitante, se trata da pessoa ou organizaçao que deseja fazer uso do web service de um fornecedor.
- Fornecedor: uma entidade fornecedora, se trata da pessoa ou organização que oferece um web service.
- Descrição de serviço: os mecanismos da troca de mensagens é documentado como Web Service *Description (WSD).* O WSD se trata de um arquivo escrito em WSDL (Web Service *Description Language*), o qual define o formato das mensagens, tipos de dados, protocolos de transporte e formato de serialização de transporte que devem ser usados na iteração entre os requisitantes e fornecedores.
- Semântica: enquanto o WSD representa um contrato relacionado aos

mecanismos de iteração, a semântica representa um contrato relacionado ao significado e propósito de uma iteração. A semântica pode ser explícita ou implícita, documentada em um arquivo ou em uma maneira orientada a humanos.

3.1.1. Visão Geral do Funcionamento de um Web Service

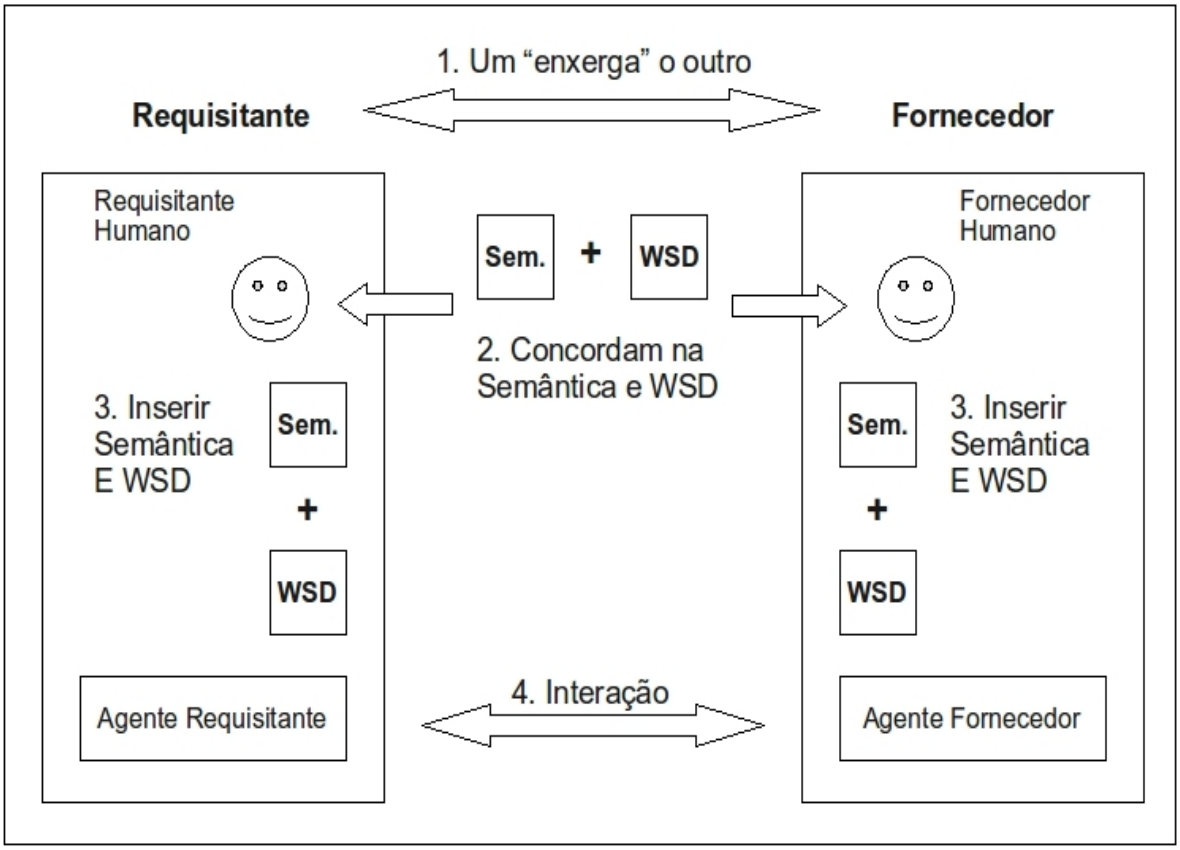

Figura 01. Passos para iteração com um Web Service (BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

Dentre os diferentes modos possíveis de se acessar um web service, em geral, quatro passos são necessários para que tal seja realizado, como na figura 01.

(1). O requisitante e o fornecedor se tornam visíveis um para o outro;

(2). O requisitante e o fornecedor concordam com a descrição do serviço e com a

semântica que coordenará a iteração entre eles;

(3). O requisitante e o fornecedor realizão a descrição do serviço e a semântica;

(4). O requisitante e o fornecedor trocam mensagens, executando tarefas particulares.

3.1.2. Modelos Arquiteturais de um Web Service

Existem alguns modelos dentro da arquitetura de um web service, os quais fornecem uma visão melhorada de como a implementação de tal pode ser orientada. Os modelos se dividem em quatro, como ilustra a figura 02.

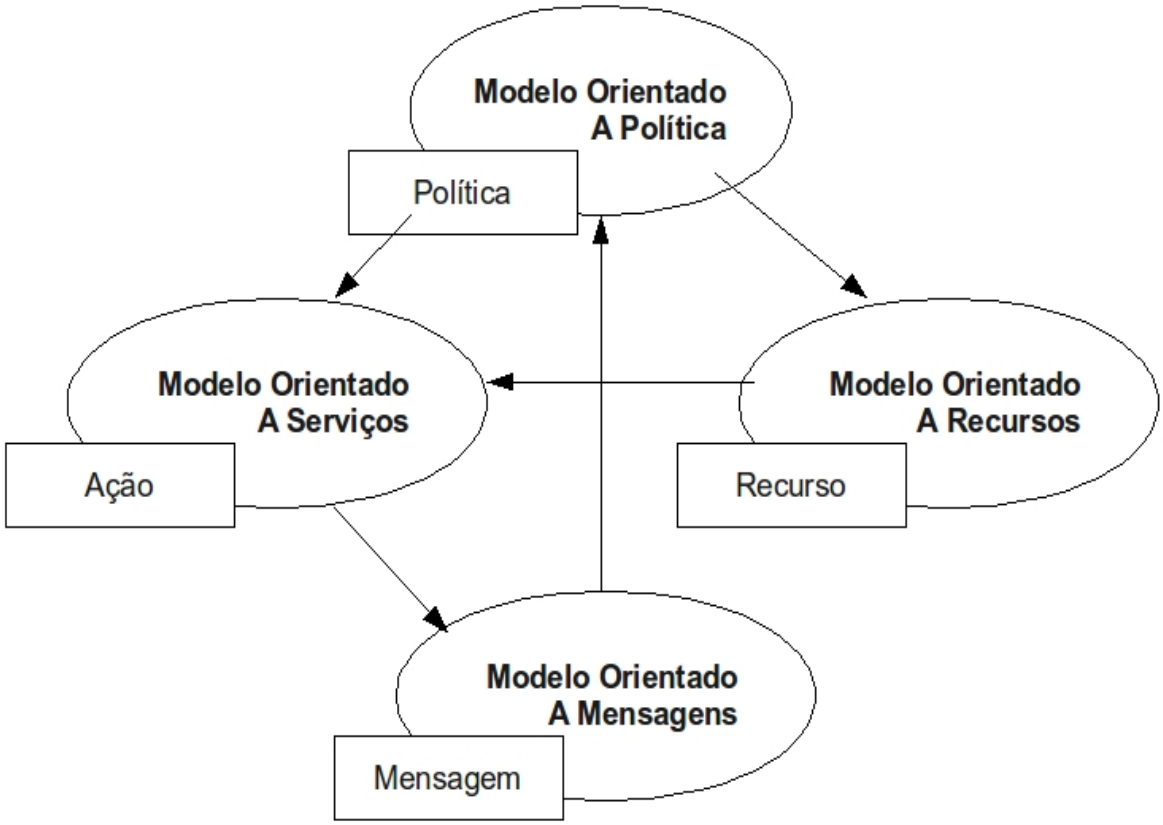

Figura 02. Modelos da arquitetura Web Service
(BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

- **Modelo orientado a mensagens:** foca nas mensagens, sua estrutura e transporte, sem se preocupar com a sua importância ou significado. Esse modelo segue alguns conceitos. O agente que envia e recebe mensagens, a estrutura da mensagem, como seus cabeçalhos e corpos, e o mecanismo utilizado para a entrega. A figura 03 ilustra como esses conceitos se relacionam.

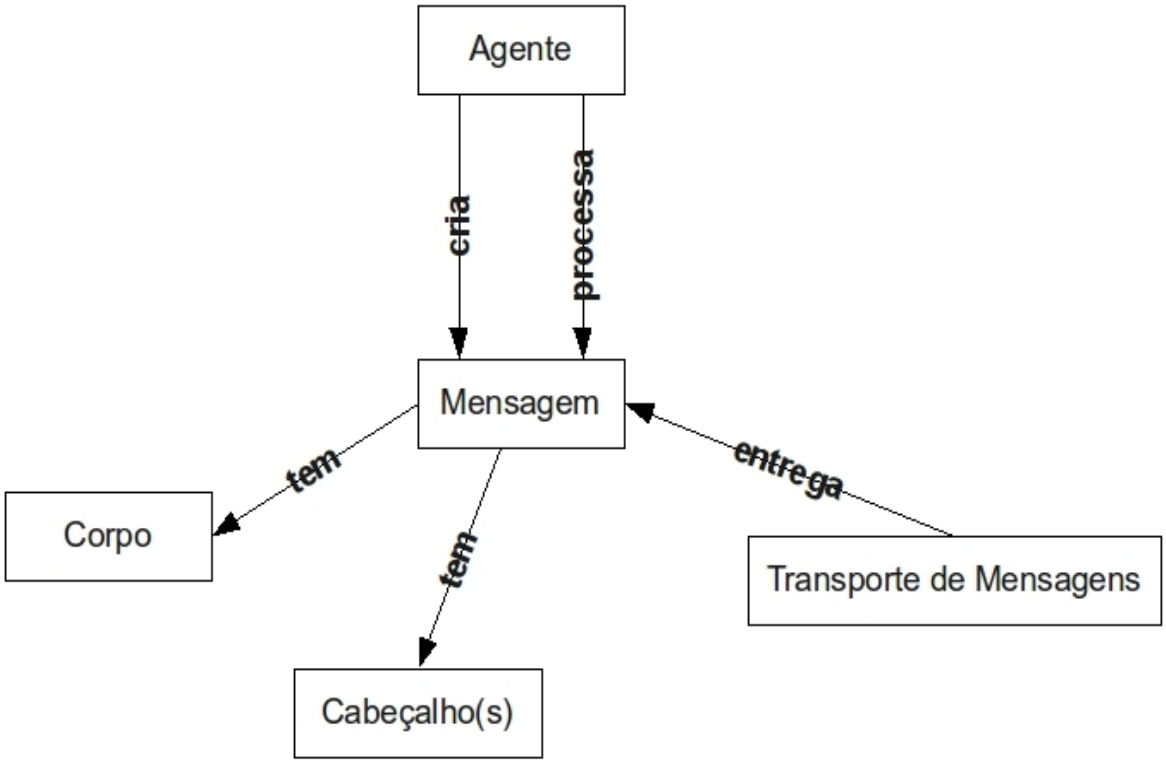

Figura 03. Relacionamento das entidades do modelo orientado à mensagens (BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

- **Modelo orientado a serviços:** foca nos aspectos e ações do serviço. Embora em um sistema distribuído um serviço não é realizado sem mensagens, as mesmas não precisam estar relacionadas com este.

Os conceitos desse modelo descrevem que um serviço é realizado por um agente e utilizado por outro. Os serviços são mediados por mensagens trocadas pelo agente requisitante e fornecedor. Os serviços possuem um dono, que seria uma pessoa ou organização, o que torna possível que ofereça funcionalidades para o mundo real. Enfim, os serviços utilizam meta dados para documentar seus aspectos, como detalhes de sua interface, semântica de transporte ou qualquer restrição imposta por seu dono. Essa descrição é ilustrada na figura 04.

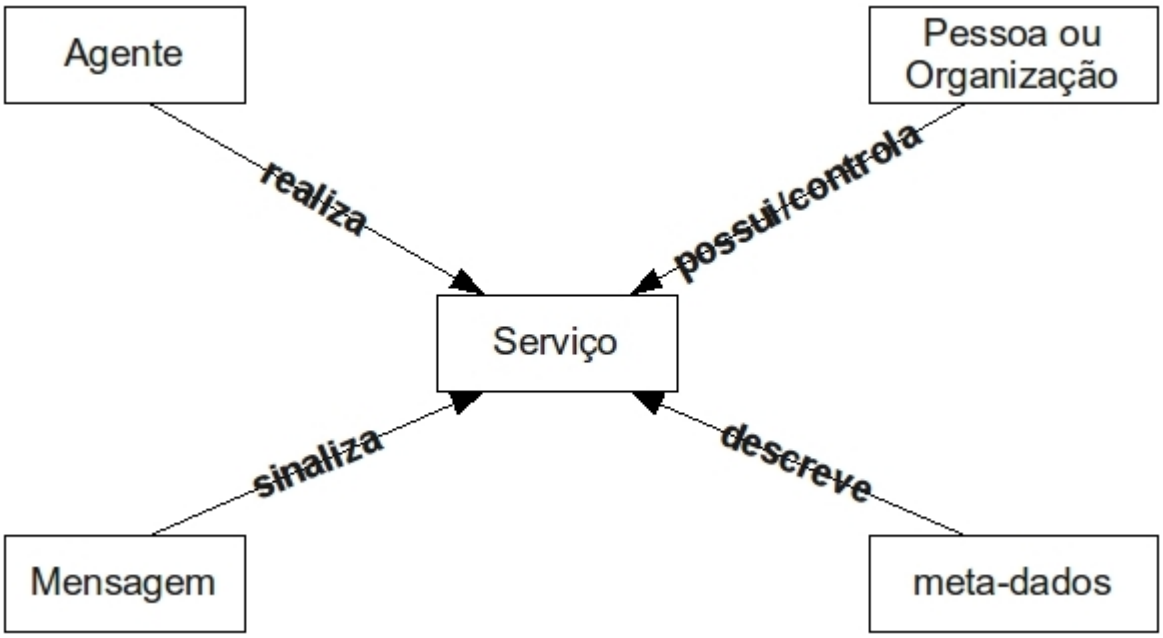

Figura 04. Relacionamento das entidades do modelo orientado a serviços (BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

- **Modelo orientado a recursos:** foca nos recursos que existem e em seus donos, seu conceito é o mesmo da arquitetura web. O recurso tem um dono, uma URI de identificação e uma representação. Ilustra-se com a figura 05.

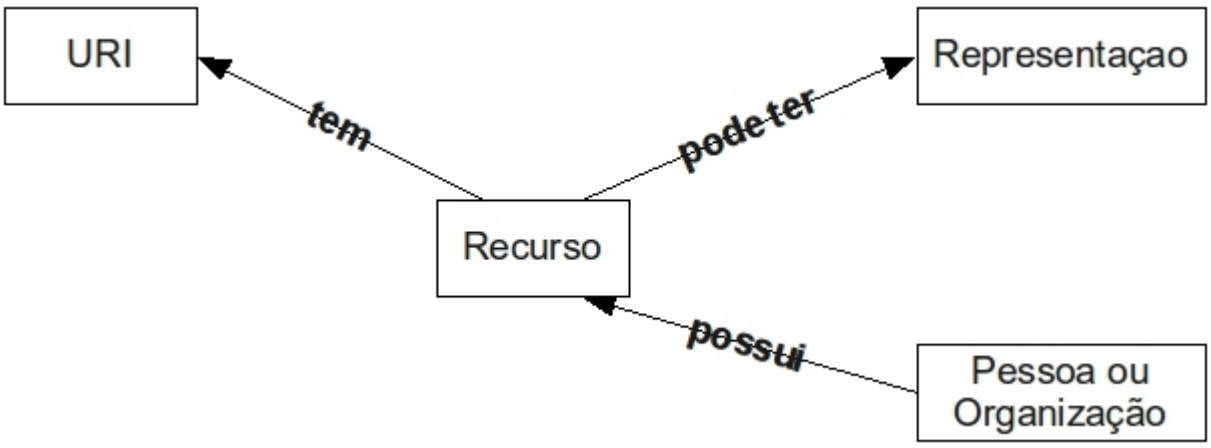

Figura 05. Relacionamento das entidades do modelo orientado a recursos (BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

- **Modelo orientado a política:** foca nas regras dos comportamentos dos serviços. Políticas são estabelecidas pos pessoas a recursos, logo em agentes que desejam acessar tal recurso. Também podem representar regras de segurança, qualidade de serviço, gerenciamento e aplicações. Isso é descrito pela figura 06.

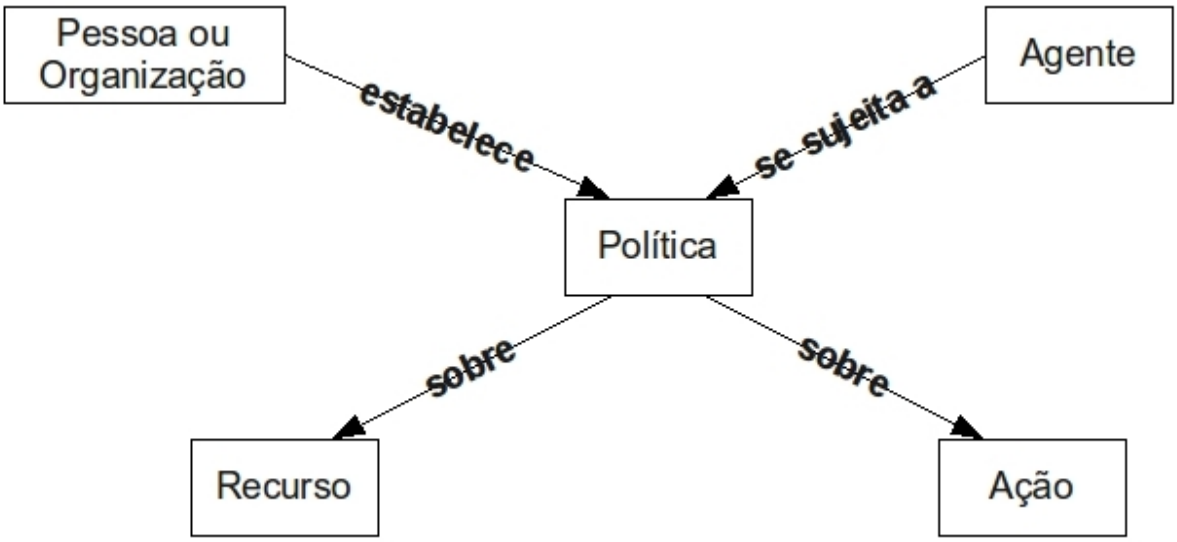

Figura 06. Relacionamento das entidades do modelo orientado à política (BOOTH, David et al. Web Services Architecture. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.)

Assim, um Web Service se mostra totalmente voltado a serviços, separando cada um de seu participante e lhe atribuindo uma responsabilidade. Serviços se relacionam não apenas com aplicações, mas também com outros serviços, o que logo realça a visão de um ambiente repleto de Web Services, então surge a necessidade de organizá-los corretamente e estabelecer regras aos seus comportamentos no ambiente, surgindo assim SOA.

4. SOA – ARQUITETURA ORIENTADA A SERVIÇOS

Dentre as publicações existentes relacionadas com SOA, diversas definições são encontradas, tendo algumas coisas em comum. O importante a ressaltar, é que SOA se trata de um paradigma, ou seja, uma maneira de pensar e organizar processos para alcançar maior flexibilidade. A forma como é feita a busca pela flexibilidade, se dá através de serviços, o que leva ao *design* de uma arquitetura orientada a serviços (ERL, 2005).

A implementação de SOA deve respeitar os conceitos de Sistemas Distribuídos, mantendo suas características, o que gera, automaticamente, à flexibilidade desejada e ao baixo acoplamento.

4.1. ORIENTAÇÃO A SERVIÇOS E O AMBIENTE EMPRESARIAL

A lógica que dirige uma empresa está em constante evolução, sofrendo modificações de acordo com as influências tanto internas como externas. Do ponto de vista de TI (Tecnologia da Informação), a lógica da empresa pode ser dividida em duas partes:

- Lógica de negócio: trata-se da implementação documentada dos requisitos do negócio necessários de acordo com as áreas de atuação da empresa. Geralmente é estruturada em processos que representam os requisitos, relacionados com qualquer dependência ou influência externa.

- Lógica de aplicação: trata-se da implementação automatizada da lógica da empresa, organizada em diversas soluções tecnológicas. Representa o fluxo de atuação dos processos através de sistemas computacionais.

Ambas existem em diferentes mundos, e cada uma representa uma parte necessária da estrutura organizacional de uma empresa.

A orientação a serviços se aplica, na lógica da empresa, introduzindo novos conceitos que aumentam a maneira de que esta lógica é representada, visualizada,

modelada e compartilhada.

Esses conceitos são introduzidos através de serviços, estabelecendo uma grande forma de abstração entre as tradicionais camadas de negócio e aplicação, podendo encapsular a lógica de ambos.

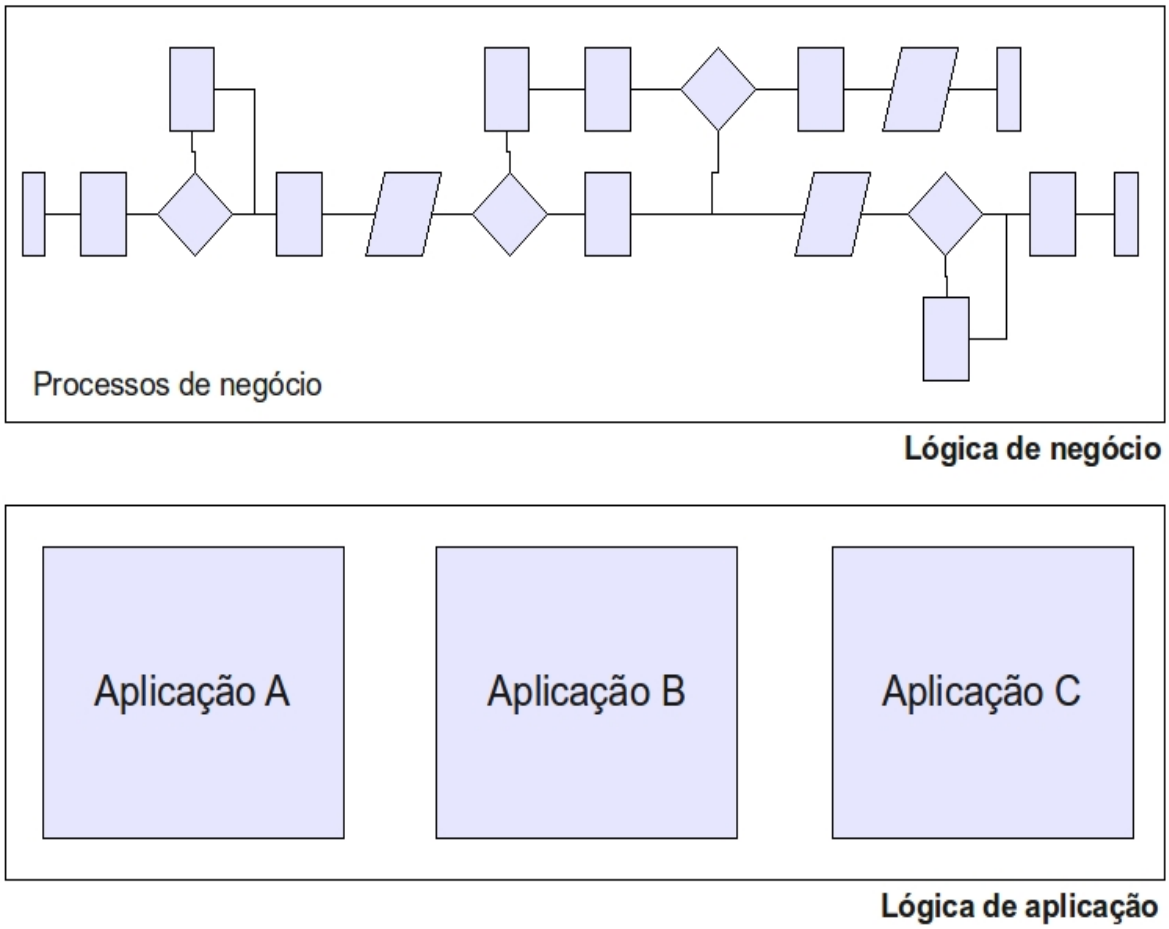

Figura 07. Domínios de lógica de negócio e aplicação (ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

Modularizando a empresa, os serviços formam uma unidade de lógica existente em uma camada de conectividade. Essa camada permite que serviços pais encapsulam serviços filhos, possibilitando a criação de diferentes camadas de abstração.

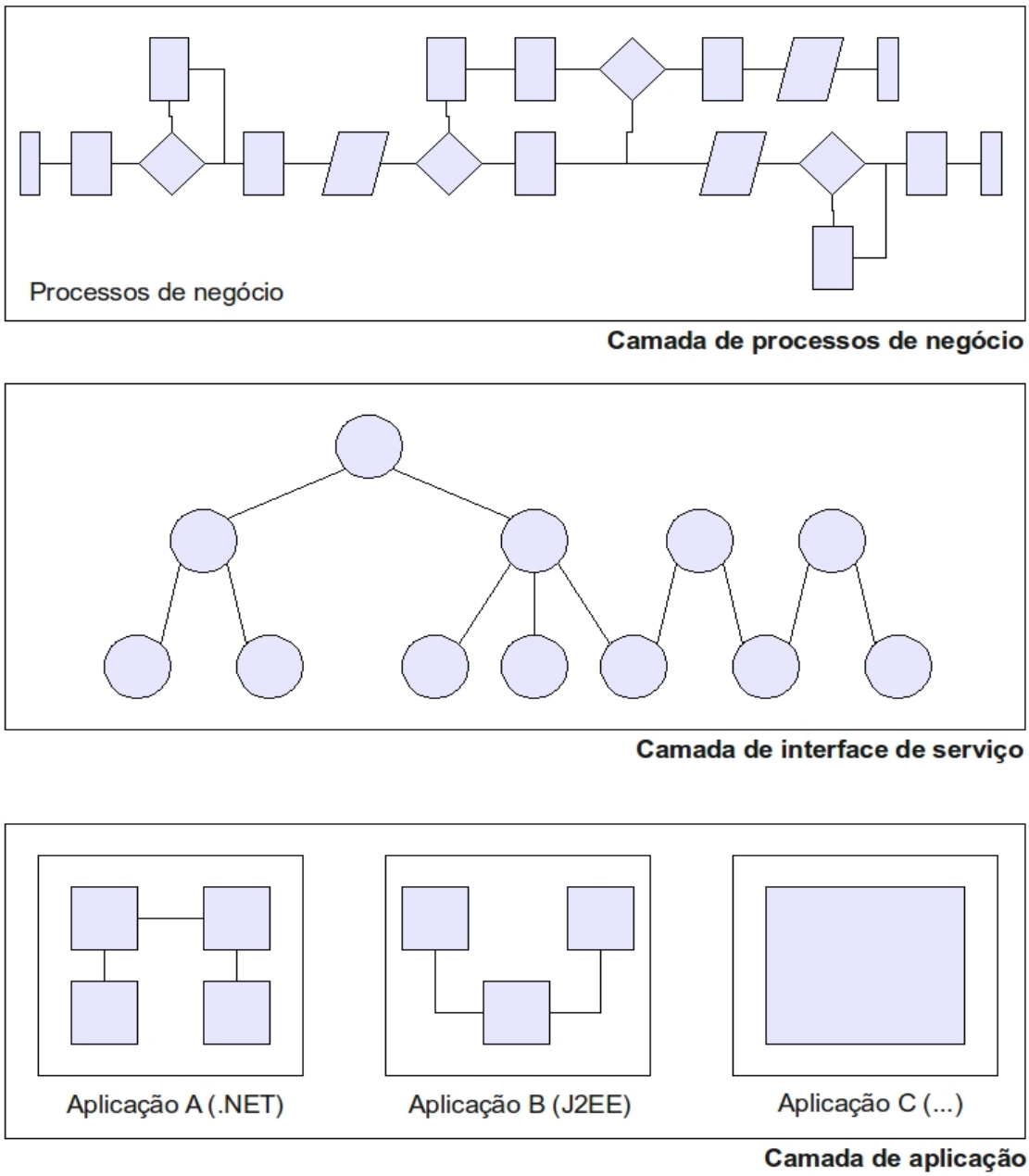

Figura 08. Camada de serviço entre as demais melhora lógica de negócio e de aplicação
(ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

Na figura 08, é possível observar que a camada da aplicação está fragmentada, categorizada por tecnologias, e a camada de serviços é uma camada contínua. Visto isso, é possível dizer que os serviços, além de encapsularem lógicas de aplicações desenvolvidas com diferentes tecnologias, se relacionam independentemente do fato, pois suas interfaces oferecem esta vantagem.

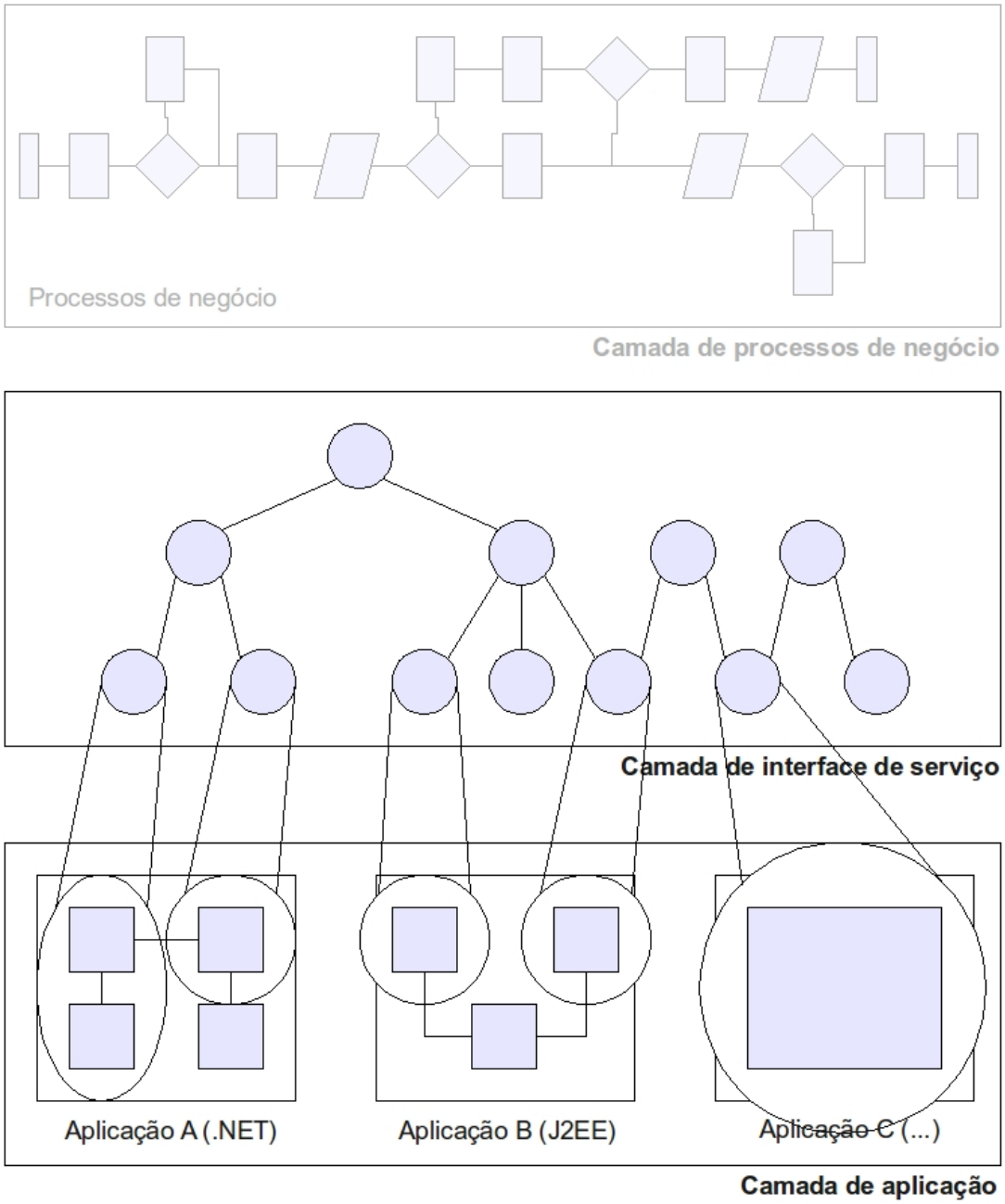

Figura 09. A camada de serviço abstrai a conectividade de ambientes de implantação de serviços
(ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

A figura 09 expõe a vantagem das interfaces dos serviços. Apesar dos serviços serem executados em ambientes de uma tecnologia específica, eles podem se relacionar sem essa preocupação.

4.2. DESENVOLVIMENTO DE UM PROJETO ORIENTADO A SERVIÇOS

Para que um projeto orientado a serviços seja desenvolvido, é necessário seguir algumas fases para alcançar a solução de serviços desejada.

3.2.1. Fases Básicas de um Projeto SOA

No geral, o desenvolvimento de um projeto orientado a serviços, é basicamente um desenvolvimento personalizado de projeto para sistemas distribuídos. Web Services são desenvolvidos e implantados, juntamente com os outros componentes e tecnologias utilizados na aplicação. Mas, entrando um pouco mais a fundo no mundo da orientação a serviços, é visto que ajustes em algumas fases do projeto sejam realizados.

Na figura 10 estão expostas as fases que projetos SOA tem em comum. É importante notar que, as duas primeiras fases do projeto têm seus nomes iniciados com "*service-oriented*" e outras apenas "*service*". Isso se dá devido à relação entre SOA e as fases de análise e *design* dos serviços, nas quais ocorre a orientação a serviços, nas outras fases, ocorrem ações relacionadas com o resultado das duas primeiras, assim se preocupando com a implementação dos serviços definidos pela análise e design.

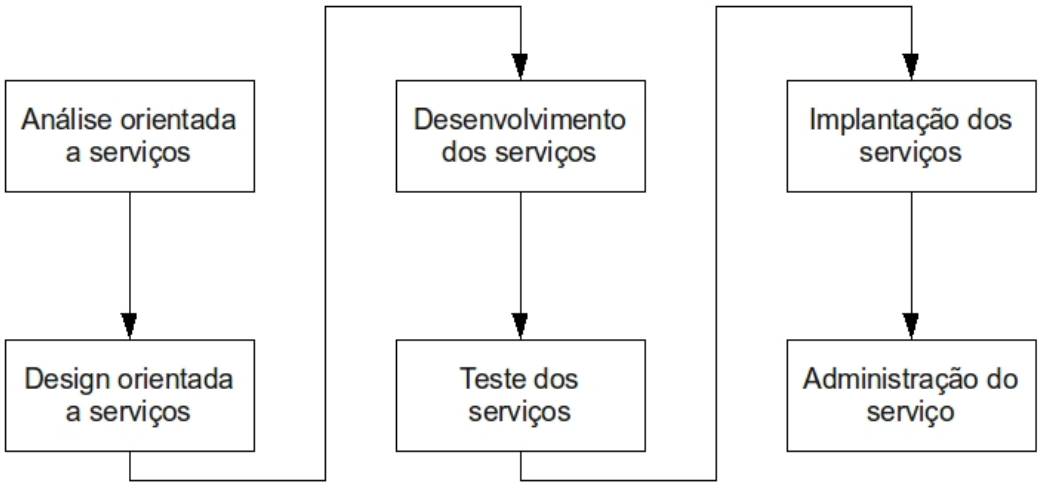

Figura 10. Fases em comum de projetos SOA
(ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

- **Análise orientada a serviços:** fase inicial, na qual o escopo da solução SOA é determinado. Camadas de serviços são definidas, e serviços individuais são modelados como candidatos a um SOA inicial.

- **Design orientado a serviços:** nesta fase a maneira que os serviços são construídos é definida, seguindo convensões do mercado e princípios orientados a serviços. Diversas decisões também são tomadas, uma delas é qual a lógica que será encapsulada em cada serviço, formando assim a definição do processo de negócio.

- **Desenvolvimento dos serviços:** ocorre a definição da plataforma que será utilizada, assim também definindo a linguagem de programação e o ambiente de desenvolvimento.

- **Testes dos serviços:** uma vez que os serviços devem ser independentes e reutilizáveis em diferentes situações, diversos testes devem ser feitos em um ambiente de produção. Alguns pontos chaves que devem ser testados são: que tipo de requisitante acessará o serviço, quais exceções podem ocorrer, qual a facilidade de composição os serviços têm, etc.

- **Implantação do serviço:** após o desenvolvimento e testes do serviço, é necessário instalá-lo e configurá-lo em um servidor de produção. Como os serviços serão distribuídos, quais as configurações de segurança necessárias, a infraestrutura é adequada para atender o processamento dos serviços, são algumas das questões resolvidas nesta fase.

- **Administração do serviço:** uma vez implantado, o serviço deve ser administrado. O processo de administração é o mesmo para aplicações distribuídas baseadas em componentização. Dentre as preocupações desta fase estão: como a utilização do serviço será monitorada, como as mensagem serão rastreadas e gerenciadas, como gargalos de desempenho serão detectados.

4.2.2. Estratégias de implantação de um projeto SOA

As fases apresentadas na sub-seção acima descrevem um simples modo de desenvolver serviços individuais. É necessário organizar essas fases em um processo que seja capaz de:

- acomodar as preferências do projeto de acordo com as camadas de serviço a serem implantadas;
- coordenar a implantação da aplicação e do negócio;
- suportar a transação para um SOA padronizado, o que ajuda a alcançar os requisitos do projeto.

O grande desafio se encontra no último item da lista. O sucesso de SOA dentro de uma empresa depende, geralmente, do modo em que é padronizado quando é dividido em domínios de lógica e aplicação. Entretanto, o sucesso de uma solução orientada a serviços, é medido pelo atendimento às expectativas dentro de um espaço de tempo e dinheiro.

Tal problema para ser contornado, precisa de uma estratégia. Esta estratégia deve ser baseada nas prioridades da organização em estabelecer um balanço correto entre objetivos de longo prazo com o alcance dos requerimentos de curto prazo. Três estratégias emergiram, cada uma abordando o problema de forma diferente.

4.2.2.1. Estratégia TOP-DOWN

Para ser orientada a serviços, a estratégia top-down não necessita apenas dos processos de negócio da empresa, mas também da criação de um modelo de negócio da mesma. Esse processo está intimamente ligado com a lógica de negócio já existente da empresa. É comum nessa estratégia, a criação de vários serviços de aplicações reutilizáveis.

Tipicamente, essa estratégia irá conter alguns ou todos os passos ilustrados na

figura 11. É importante notar que esse processo assume que os requisitos de negócio já tenham sido coletados e definidos.

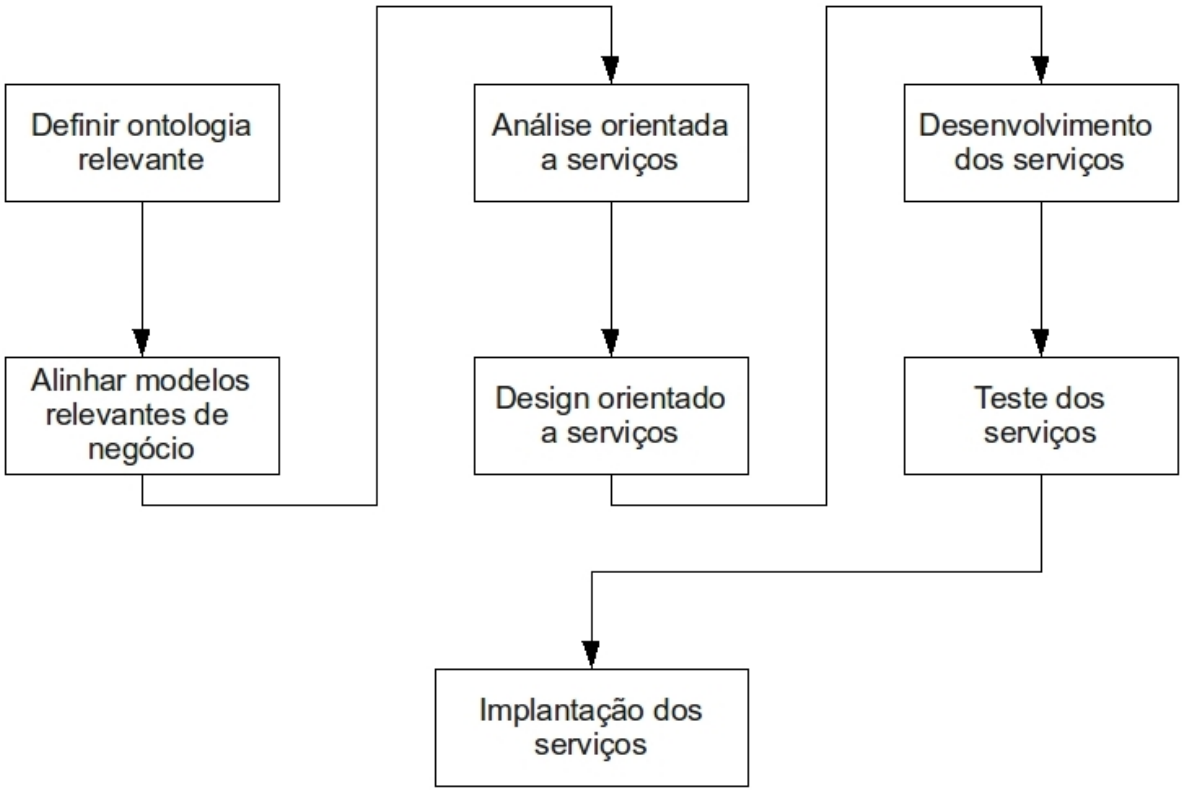

Figura 11. Fases comuns da estratégia TOP-DOWN
(ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

- **Definir uma ontologia empresarial relevante:** classificar os conjuntos de informação processados pela empresa, resultando em um vocabulário comum de como esses conjuntos se relacionam.

- **Alinhar modelos relevantes de negócio:** ajustes podem ser necessários para seguir o vocabulário estabelecido pela ontologia. Importante focar modelos de entidades, pois eles podem, mais tarde, ser usados como base em serviços de lógica centrada em entidades.

- **Realizar a análise orientada a serviços:** fase onde ocorre uma análise orientada a serviços da lógica de negócio.

- **Realizar o design orientado a serviços:** definição das camadas de serviço que serão criadas.
- **Desenvolver os serviços necessários:** desenvolvimento dos serviços de acordo com as especificações definidas no design.
- **Testar os serviços e suas operações:** os serviços precisam passar por diversos testes de qualidade. O serviço deve ser capaz de lidar com diferentes situações.
- **Implantar os serviços:** os serviços são implantados nos servidores de produção. Alguns podem necessitar de um processamento maior que outros, devido à quantidade de requisições, assim podendo existir diferentes medidas de segurança e acessibilidade.

A estratégia top-down para construir SOA, geralmente resulta em uma arquitetura de serviços de alta qualidade. O design e parâmetros entre os serviços são analisados um a um, maximizando seu potencial reutilizável e facilitando a composição com outros serviços.

Alguns obstáculos dessa estratégia estão relacionados com tempo e dinheiro. As empresas devem investir pesado na análise, o que pode levar uma grande quantidade de tempo, sem obter resultados em curto prazo.

4.2.2.2. Estratégia BOTTOM-UP

A preocupação desta estratégia, é criar serviços para atender as necessidades centradas em aplicações. Web services são criados conforme a necessidade, encapsulando a lógica da aplicação, para melhor servir os requisitos da solução. O principal motivador aqui é a integração.

Um processo bottom-up típico segue as ilustrações da figura abaixo. É preciso que os requisitos da lógica já tenham sido coletados e definidos.

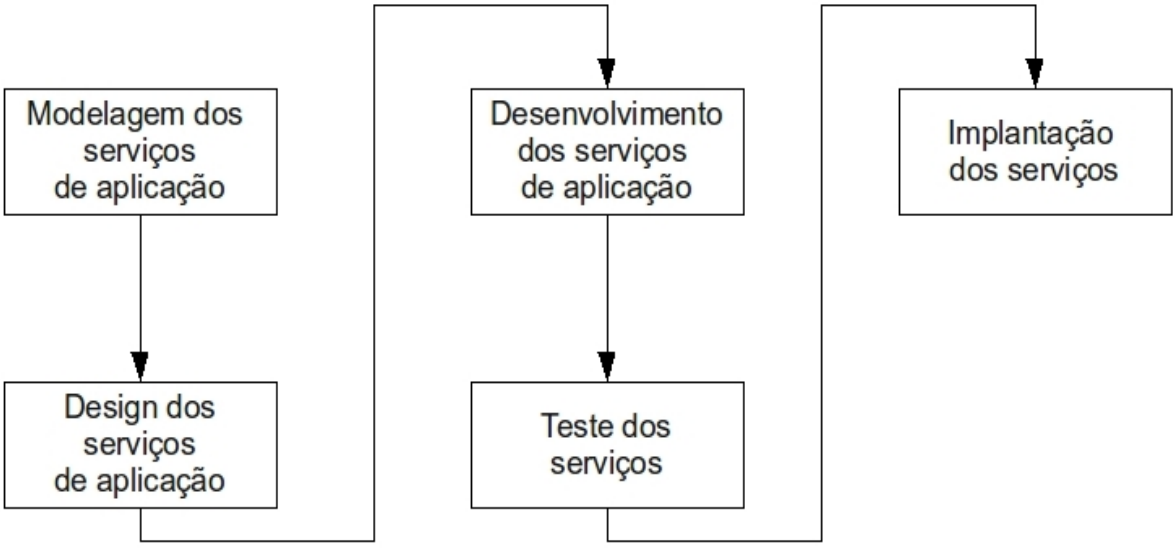

Figura 12. Fases comuns da estratégia BOTTOM-UP (ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

- **Modelar serviços de aplicações necessários:** resultado da definição dos requisitos da aplicação que podem ser atendidos por Web Services. Os serviços são modelados para incluir lógica e regras de negócio específicas, assim, surgem duas camadas de serviços de aplicação, contendo serviços de diferentes utilidades.

- **Realizar o design dos serviços de aplicação necessários:** alguns serviços que fazem papel de proxy para a aplicação, não oferecem muitas opções de design, diferentemente de serviços personalizados que necessitam garantir um certo level de consistência.

- **Desenvolvimento dos serviços de aplicação necessários:** serviços de aplicação são desenvolvidos de acordo com suas respectivas *services descriptions* e especificações de design.

- **Testar os serviços:** a associação com o ambiente da solução e a lógica encapsulada dos serviços são testados para garantir que os requisitos serão atendidos. Testes de desempenho e segurança também são realizados.

- **Implantar os serviços:** os serviços são implantados nos servidores de produção, levando em consideração a necessidade de desempenho e segurança.

A maioria das empresas que desenvolvem Web services utilizam a estratégia bottom-up. Um dos motivos para tal, é o fato de que o que precisa ser feito é apenas adicionar Web services nos ambientes de aplicações existentes.

Como resultado, conceitos da orientação a serviços são raramente levados em consideração. A estratégia bottom-up, na verdade, não é uma estratégia válida para se alcançar SOA. Apesar de levar a uma criação eficiente de Web services de acordo com as necessidades das aplicações. A implementação de um SOA mais correto pode resultar na introdução de novas camadas padronizadas, posicionadas no topo das camadas geradas por essa estratégia.

4.2.2.3. Estratégia AGILE

O desafio de encontrar um balanço aceitável entre incorporar princípios de design orientados a serviço no ambiente de análise de negócio, sem ter que esperar antes integrar tecnologias de Web services em ambientes técnicos, ainda persiste. Para várias empresas, é melhor visualizar essas duas abordagens como extremos e encontrar uma solução entre eles.

Isso é possível quando um novo processo é definido, o qual permite que a análise do negócio ocorra concorrentemente com o design e desenvolvimento de serviços.

Os passos ilustrados na figura abaixo demonstram como a estratégia agile pode ser usada para alcançar os objetivos das estratégias top-down e bottom-up.

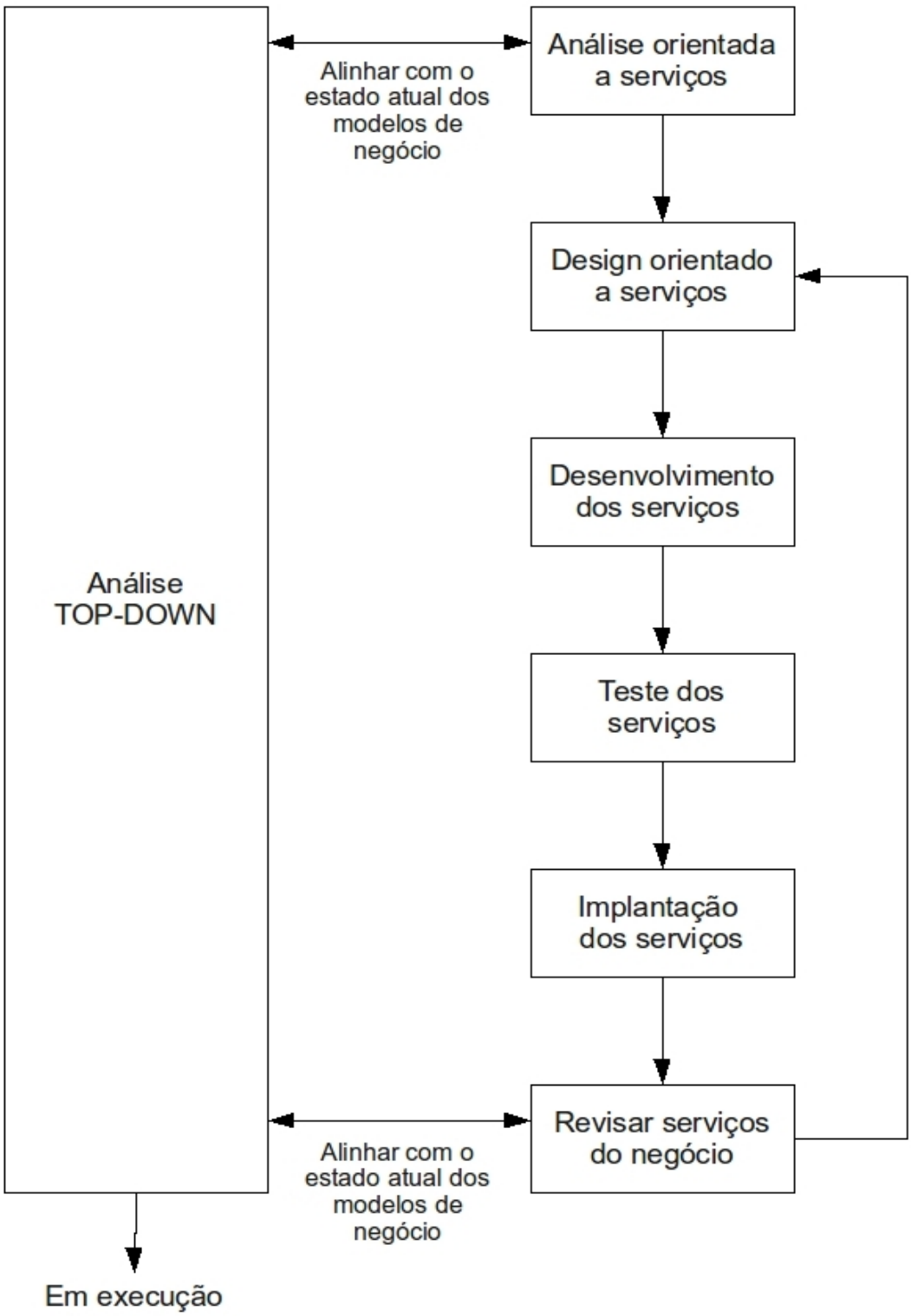

Figura 13. Fases comuns da estratégia AGILE
(ERL, Thomas. Service-Oriented Architecture: Concepts, Technology, and Design. Editora Prentice Hall PTR, 2005. 792 p.)

- **Início da análise top-down, focando primeiramente em partes chave da ontologia e entidades relacionadas com o negócio:** a análise padrão da estratégia top-down é iniciada, mas com um certo foco. As partes do modelo de negócio diretamente relacionadas com a lógica de negócio, recebem maior prioridade.

- **Após a análise top-down, realizar a análise orientada a serviços:** Ainda com o passo 1 em execução, uma nova fase de análise orientada a serviços é iniciada. Ocorre uma decisão entre realizar outro processo de análise além da análise top-down.

- **Realizar o design orientado a serviços:** novas camadas de serviços e serviços individuais são definidos.

- **Desenvolver, testar e implantar os serviços:** desenvolvimento, testes e implantação dos serviços em produção.

- **Ainda com a análise top-down em progresso, realizar os passos novamente:** revisar os passos do processo periodicamente e comparar seu design com o do estado atual dos modelos de negócio.

A estratégia agile, define uma combinação dos elementos necessários para realizar um projeto SOA levantando e resolvendo os requisitos de curto e longo prazo do projeto. A revisão dos serviços é feita proporcionalmente com a maneira em que o número dos mesmos aumenta.

Como qualquer processo de software, um projeto SOA também possui suas particularidades, tendo como principal objetivo a criação de novas camadas, nas quais a comunicação é realizadas apenas através de serviços. Isso encapsula a lógica de diferentes aplicações, facilitando, principalmente, a integração das mesmas.

No desenvolvimento de um projeto SOA, é importante adotar uma estratégia, seguindo passos bem definidos, essenciais para o desenvolvimento do projeto.

5. ESTUDO DE CASO

Além do estudo teórico de uma tecnologia, é importante entender como ela se comporta na prática, para tal, é necessário um estudo de caso para criar primeiros contatos com a tecnologia. Neste trabalho, o estudo de caso será a implementação de uma aplicação, responsável pela busca e atualização de currículos dos ex-alunos de uma instituição. A aplicação possui características distribuídas, pois para a criação do serviço, é preciso um certo grau de independência entre seus componentes.

5.1. VISÃO GERAL DA APLICAÇÃO

O objetivo da aplicação aqui apresentada, é manter contato entre os ex-alunos de uma instituição com as organizações do mercado de trabalho. Na visão desta aplicação, um currículo é composto por:

- dados dos ex-alunos: nome, endereço, e-mail;
- categorias: graduação, eventos, pós-graduação;
- atividades: atividades desenvolvidas durante uma categoria.

Logo, sendo uma aplicação orientada a objetos, existem classes que mapearam os objetos ex-aluno, currículo, categoria e atividades, além de outras classes responsáveis por oferecerem diversas funcionalidades necessárias pela aplicação, inclusive o serviço.

A organização da aplicação é feita através de módulos, existindo primeiramente três:

- Módulo de interface web: aplicação web responsável pela comunicação usuário/aplicação;
- Módulo de modelos: classes responsáveis por mapear os objetos do mundo real;

- Módulo de componentes distribuídos: comunicação com a base de dados e a interface do serviço.

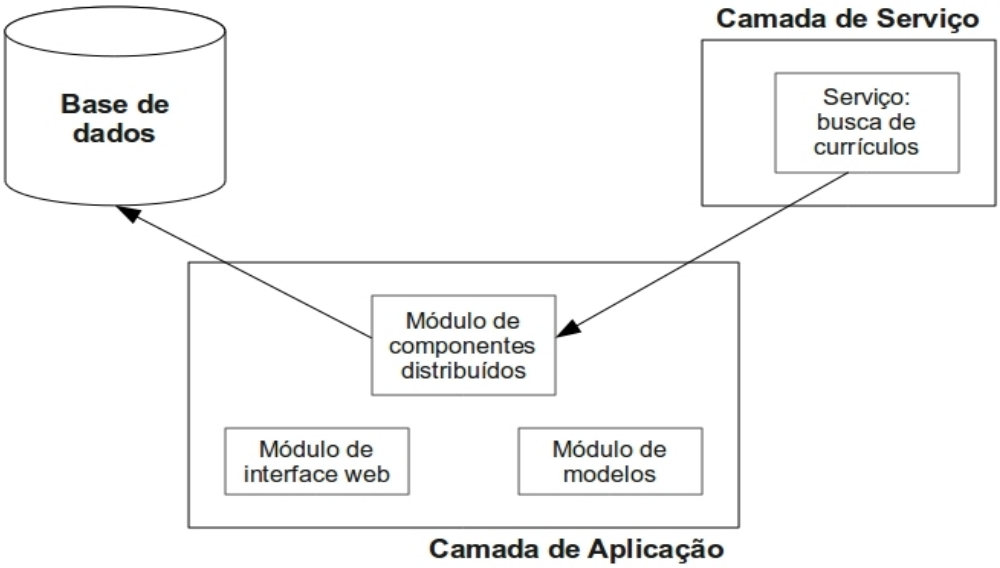

Figura 14. Arquitetura da aplicação do estudo de caso

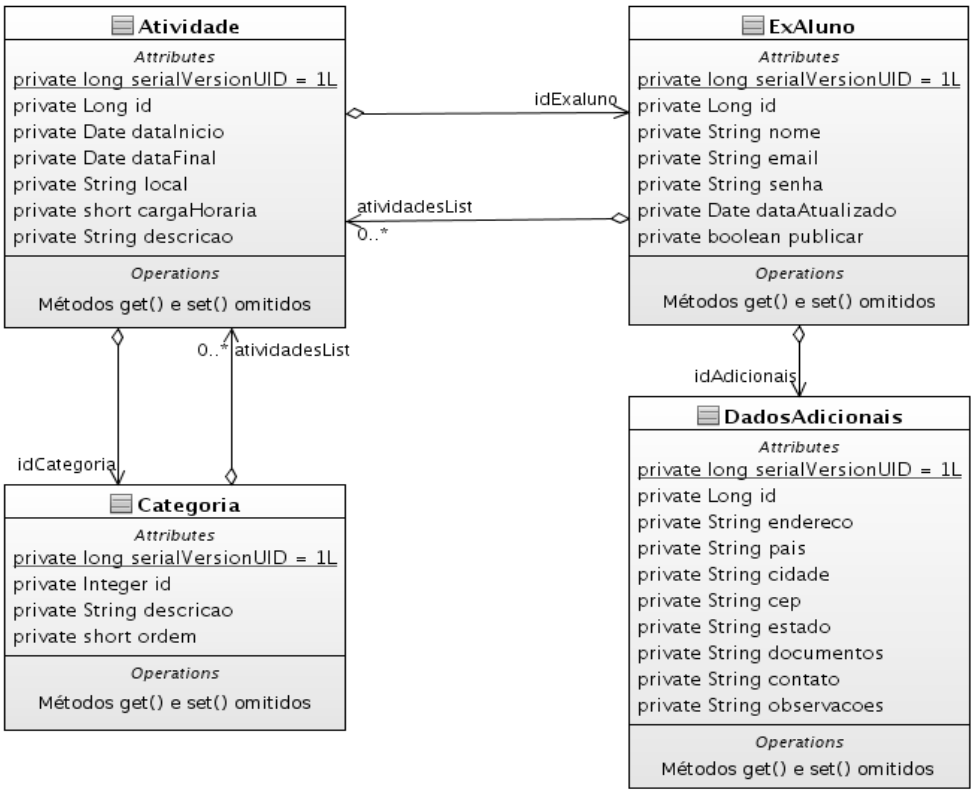

Figura 15. Diagrama UML das classes de mapeamento da aplicação

5.2. DESCRIÇÃO DO DESENVOLVIMENTO

Durante o desenvolvimento, para melhor organização dos códigos-fonte, facilitar a manutenibilidade e oferecer uma boa reutilização, as classes foram divididas em pastas diferentes, chamadas de pacotes. Os códigos-fonte foram escritos seguindo padrões de nomeação e de design.

Para nomeação de classes, atributos e métodos, foi utilizado o padrão *CamelCase*, no qual as palavras compostas são iniciadas com uma letra maiúscula ou minúscula, dependendo da situação.

Como padrão de design da aplicação, foi utilizado o MVC (Modelo – Controle – Visão), o qual oferece melhor suporte para a reutilização das classes e facilidade na manutenibilidade.

- Módulo de modelos: foram criadas dois pacotes, um para armzenar as classes de mapeamento dos objetos do mundo real e outro para as classes responsáveis pela ordenação e classificação dos dados;

- Módulo de interface web: existem também dois pacotes, um para armazenar as classes responsáveis pelo fluxo da aplicação web, ou seja, a navegação do usuário e outro, que contém a classe responsável pela geração do currículo como um arquivo disponível para impressão;

- Módulo de componentes distribuídos: mais dois pacotes aqui foram criados, um que contém as classes responsáveis pela comunicação da aplicação com a base de dados e outro, que armazena as classes responsáveis por expor os serviços necessários para a busca de currículos;

Seguem algumas imagens da aplicação web e da aplicação cliente, a qual acessa os serviços disponíveis pelo módulo de componentes distribuídos.

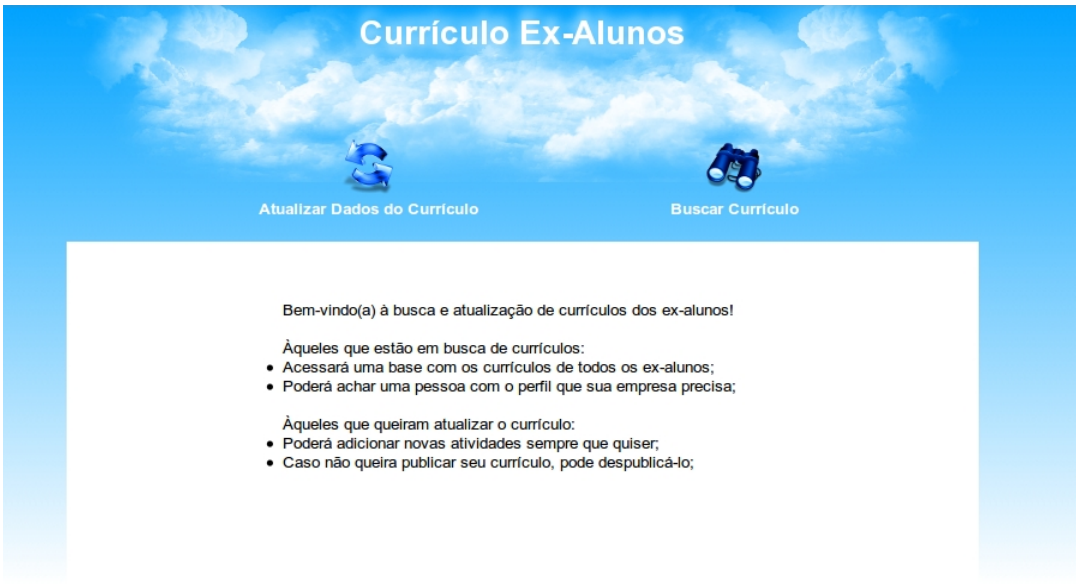

Figura 16. Tela inicial da aplicação web

Figura 17. Tela de busca de currículos

Figura 18. Tela de gerenciamento do currículo

Figura 19. Tela de atualização dos dados do currículo

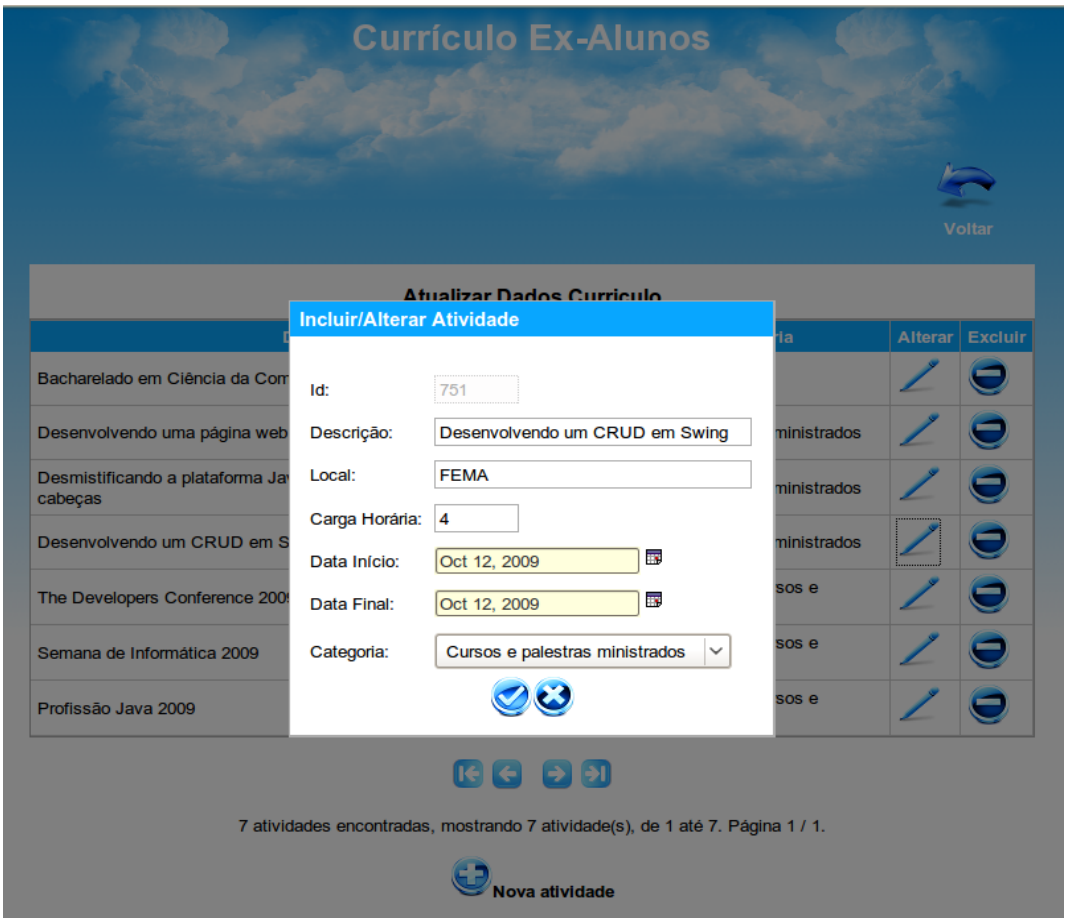

Figura 20. Tela de inclusão e alteração dos dados do currículo

Figura 21. Tela de atualização dos dados do ex-aluno

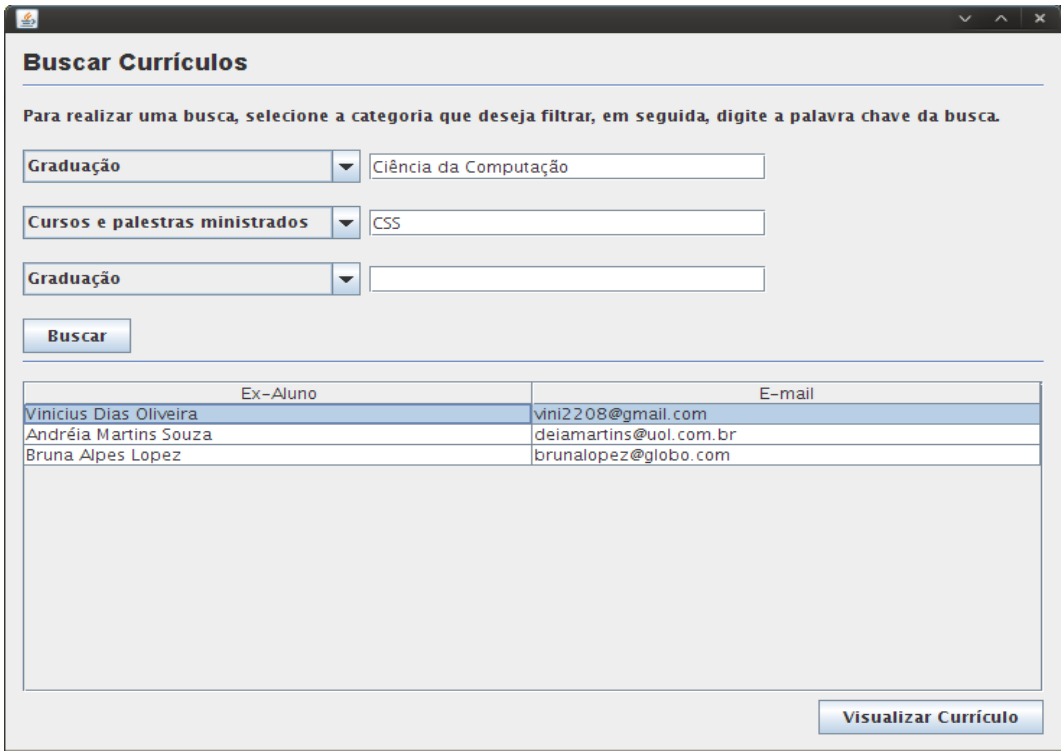

Figura 22. Tela de busca da aplicação cliente

Figura 23. Currículo gerado pela aplicação

6. CONCLUSÃO

O desenvolvimento de software com características distribuídas é um dos principais focos do mercado de trabalho nos dias de hoje. A comunicação entre diferentes dispositivos é essencial para a distribuição de informações. Uma das maneiras de desenvolvimento distribuído é a utilização de Web Services, os quais encapsulam as características distribuídas.

Este trabalho possibilitou a criação de uma base para o desenvolvimento de aplicações com uma arquitetura orientada a serviços. A principal etapa aqui, foi o estudo de como diferentes camadas são criadas para integrar ainda mais a lógica de negócio da empresa, com as aplicações da mesma, de modo independente.

Para uma preparação ainda melhor no desenvolvimento orientado a serviços, existem diversos assuntos a serem tratados. Estudos mais detalhados sobre Web Services e sua arquitetura ajudariam bastante para um melhor estudo de caso, o qual também poderia ser diferente, ou seja, uma integração entre diferentes aplicações, mostraria uma diferente abordagem de serviços, mas também, muito utilizada.

REFERÊNCIAS BIBLIOGRÁFICAS

BOOTH, David et al. **Web Services Architecture**. <http://www.w3.org/TR/ws-arch/>. Acesso em: 15 mar. 2009.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.. **Sistemas Distribuídos: Conceitos e Projeto**. 4. ed. Editora Bookman, 2007. 784 p.

ERL, Thomas. **Service-Oriented Architecture: Concepts, Technology, and Design**. Editora Prentice Hall PTR, 2005. 792 p.

ERL, Thomas. **SOA Principles of Service Design.** Prentice Hall, 2008. 573p.

JOSSUTIS, Nicolai. **SOA in Practice**. 1. ed. Editora O'Reilly, 2007. 324p.