Daniel Massaki Kurokawa

Metodologia para planejamento, análise, especificação e desenvolvimento de Arquitetura Orientada a Serviços

São Paulo

2011

Daniel Massaki Kurokawa

Metodologia para planejamento, análise, especificação e desenvolvimento de Arquitetura Orientada a Serviços

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de MBA em Gestão de Tecnologia da Informação

Área de Concentração:
Tecnologia da Informação

Orientador:
Professor Eduardo de Oliveira

São Paulo

2011

AGRADECIMENTOS

Ao meu orientador Professor Eduardo de Oliveira por todos os conselhos, diálogos, troca de conhecimentos e pela paciência durante a realização deste trabalho.

A todos os meus amigos de turma que fizeram o curso de MBA um verdadeiro sucesso.

A todos os professores do curso de MBA com os quais tive oportunidade de aprender.

À minha namorada, que sempre me apoiou e me incentivou.

RESUMO

O objetivo de uma arquitetura orientada a serviços é a integração dos setores corporativos através de um canal de comunicação comum devido à crescente complexidade dos requisitos a serem atendidos e à necessidade de agilidade nos negócios. Essa integração é primordial pela dependência de cada componente organizacional. Para desenvolver e implementar uma arquitetura orientada a serviços (SOA), é necessário a utilização de uma metodologia de planejamento, análise, especificação e desenvolvimento que permita aos interessados ter um melhor rendimento para criação de uma nova arquitetura, diminuindo os riscos. O desenvolvimento deste método abrangente é o foco desta monografia. Este trabalho de monografia apresenta os conceitos básicos da arquitetura orientada a serviços, os benefícios que a SOA traz para as corporações que adotem este modelo estratégico, tecnologias e ferramentas relacionadas a SOA. Em seguida, é apresentada uma metodologia de planejamento (em que são levantadas as justificativas do projeto e as tarefas a serem executadas), análise (fase que foca no levantamento de requisitos que deverão ser atendidos pelo projeto), especificação (criação de todos os artefatos que servirão de guia na implementação do sistema) e desenvolvimento (implementação e testes da arquitetura orientada a serviços) da arquitetura-alvo. Finalmente, os resultados apresentam a nova postura que a empresa deverá adotar após a implantação de uma SOA, buscando os benefícios propostos e minimizando possíveis riscos existentes. Isso possibilita um desenvolvimento mais acelerado, atendimento de demandas mais complexas que exigem integração de setores corporativos e diminuição de risco de problemas como atrasos, falta de planejamento, qualidade abaixo das expectativas e retrabalho.

Palavras-chave: Arquitetura corporativa. Arquitetura orientada a serviços. Metodologia. Planejamento. Análise de requisitos. Especificação. Desenvolvimento.

ABSTRACT

The purpose of a service-oriented architecture is the integration of corporate sectors through a common communication channel due to the increasing complexity of requirements to be met and the need of business agility. This integration is crucial for the dependence of each organizational component. To develop and implement a service-oriented architecture (SOA), it is necessary to use a planning methodology, analysis, specification and development that allow the teams to have a better performance in creating a new architecture, reducing the risks. The development of this comprehensive method is the focus of this monograph. This monograph presents the basic concepts of service-oriented architecture, the benefits that SOA brings to the corporations that adopt this strategic model, technologies and tools related to SOA. Next, this monograph presents a methodology of planning (that justifications of the project and the tasks to be performed are presented), analysis (which focuses on the requirements identification that must be achieved by the project), specification (creation of all artifacts that will guide the implementation of the system) and development (implementation and testing of service-oriented architecture) of the target architecture. Finally, the results showed the new stance that the company will adopt after the deployment of an SOA, seeking the proposed benefits and minimizing potential risks. This allows for faster development, comply with demands that require more complex integration of the corporate sectors and decrease the risk of problems such as delays, lack of planning, quality below expectations and rework.

Keywords: Enterprise architecture. Service-oriented architecture. Methodology. Planning. Requirements analysis. Specification. Development.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API: *Application Programming Interface*

B2B *Business to Business*

B2C: *Business to Consumer*

B2E *Business to Employee*

BPEL: *Business Process Execution Language*

EA: *Enterprise Architecture*

ESB: *Enterprise Service Bus*

FCS: Fatores Críticos de Sucesso

HTTP: *HyperText Transfer Protocol*

IDE: *Integrated Development Environment*

IDL: *Interface description language*

IT: *Information Technology*

JCA: *Java Connector Architecture*

JVM: *Java Virtual Machine*

PMBOK: *Project Management Body of Knowledge*

PMI: *Project Management Institute*

RUP: *Rational Unified Process*

SaaS: *Software as a Service*

SOA: *Service Oriented Architecture*

SoaML: *Service Oriented Architecture Modeling Language*

SOAP: *Simple Object Access Protocol*

TI: Tecnologia da informação

UDDI: *Universal Description, Discovery and Integration*

UML: *Unified Modeling Language*

WBS: *Work Breakdown Structure*

WSDL: *Web Services Description Language*

XML: *eXtensible Markup Language*

XSD: *XML Schema Definition*

SUMÁRIO

1. INTRODUÇÃO

1.1. Importância do tema

Há estudos onde foram feitos levantamentos apontando que as Arquiteturas Orientadas a Serviços (SOA), dentro de alguns anos, terão grande influência sobre o desenvolvimento de novos sistemas. O *Gartner Institute*[1] estima que até 2011, mais do que 75 por cento das organizações usarão SOA no desenvolvimento e integração de aplicações e processos. Isso mostra que as empresas estão cada vez mais preocupadas com a integração de sistemas e a adoção de um paradigma que aprimore os processos de negócios corporativos.

O conceito de Arquitetura Orientada a Serviços é relativamente recente e tem despertado globalmente o interesse de muitas organizações. Atualmente há uma grande quantidade de material sobre SOA, mas sem uma visão integrada deste tipo de arquitetura. Há dificuldades para consolidação de um modelo que estabeleça métodos consistentes para implementar SOA.

1.2. Objetivos

O objetivo desta monografia é fazer a apresentação de uma metodologia que ajude nas fases de planejamento, análise, especificação e desenvolvimento de arquiteturas orientadas a serviços. Para alcançar tal objetivo, será proposto um método que abranja vários aspectos, como planejamento, negócio, desenvolvimento de aplicações e operações. Estes aspectos são pertencentes às arquiteturas corporativas.

O estudo feito aborda:

- Planejamento do projeto SOA;
- A definição do escopo da análise orientada a serviço;
- A identificação e a modelagem de um serviço;
- A definição de requisitos não funcionais;
- A especificação da tecnologia e a escolha da plataforma de desenvolvimento da arquitetura orientada a serviços;

[1] Também conhecido como GartnerGroup, é centro de pesquisa tecnológica voltada aos impactos dos negócios com a tecnologia.

- O desenvolvimento da arquitetura.

1.3. Justificativa e Motivação

A agilidade do mundo corporativo é um ponto que cada vez mais importante para as empresas que querem se adequar com a dinâmica do mercado de tecnologia da informação, mas tal agilidade faz com que os requisitos de negócio não sejam atendidos conforme o esperado. Problemas originados dessa alta necessidade de atender os requisitos em menos tempo são provenientes de problemas de comunicação e baixa interação entre as áreas corporativas envolvidas. Dessa forma, é necessário possuir uma arquitetura corporativa bem robusta e eficaz capaz de garantir que os requisitos de negócio sejam completamente atendidos.

Por sua vez, a arquitetura orientada a serviços (SOA) é uma tecnologia recente que chegou a possuir maior visibilidade nos últimos anos. SOA possibilita a implementação de serviços responsáveis por atender os requisitos de negócio devido aos conceitos de arquitetura corporativa com os quais estão fortemente ligados. De acordo com Thomas Erl (2005), SOA é um paradigma altamente recomendado para integração de sistemas, independentemente das tecnologias utilizadas para atingir os objetivos estratégicos corporativos. Segundo Botto (2008), SOA chega para mudar os processos internos das corporações a fim de aumentar a produtividade e agilidade. O *Gartner Institute* (2009) aponta SOA como uma das soluções focadas para redução de gastos e que, pela evolução nos últimos anos, deixa de ser uma aposta, mas sim um produto consolidado responsável por contribuir na integração de aplicativos e sistemas legados.

Um dos benefícios mais importantes que SOA fornece é a melhora da produtividade e agilidade tanto para o negócio quanto para TI, e, portanto, a redução de custos no desenvolvimento e manutenção dos sistemas envolvidos neste tipo de arquitetura.

Daniel Batista Fernandes[2] afirma que "o projeto estrutural de implementação de SOA requer mudanças profundas de metodologia", o que traz organização ao projeto de desenvolvimento. Já Socorro Cavalcante[3] afirma que a "metodologia é a base do trabalho executado na organização", porém não são utilizadas com frequência, pois há o "pensamento que existe na maior parte das organizações é imensamente perturbador", como, por exemplo, metodologia pode atrapalhar ou atrasar os processos corporativos. Vijay Narayanan[4] afirma que as metodologias provêm técnicas para identificar, especificar e criar serviços que compõem a arquitetura orientada a serviços da corporação.

Este trabalho de monografia irá abordar as fases de planejamento, análise, especificação e desenvolvimento, que abrangem desde o estudo de verificar se SOA é a melhor solução para a corporação até a remodelagem dos processos de negócio para ser base da arquitetura desenvolvida. A cada etapa, a metodologia destaca as tarefas e os entregáveis de cada uma delas, apontando o motivo e a importância dos passos e dos artefatos criados e desenvolvidos.

---

[2] Daniel Batista Fernandes é autor do livro "Análise de Sistemas Orientada ao Sucesso - Rio de Janeiro, Editora Ciência Moderna – 2005". Retirado de http://imasters.com.br/artigo/8140/gerencia/a_onda_soa/.

[3] Socorro Cavalcante é analista sênior líder da SeedTS, empresa especializada em SOA (www.seedts.com). Retirado de http://www.linhadecodigo.com.br/artigo/2528/Metodologia-SOA-qual-a-necessidade.aspx.

[4] Vijay Narayanan é líder técnico e contribuidor da SOA Magazine. Retirado de http://www.soamag.com/I42/0810-2.php.

2. REVISÃO DA LITERATURA

A revisão da literatura irá abordar conceitos relacionados à arquitetura orientada a serviços (conceito base desta monografia), *Web Services* (que representa uma possível implementação dos serviços), as etapas de planejamento, análise, especificação e desenvolvimento de projeto (que servirão como base das fases propostas pela metodologia) e as linguagens *Business Process Execution Language* (BPEL) e *Unified Modeling Language* (UML) a serem utilizadas na fase de especificação da metodologia.

2.1. Arquitetura Orientada a Serviços

SCHULTE & NATIS (1996), então analistas do *Gartner Institute*, definiram SOA como "uma configuração de computação multicamadas que ajudam as organizações compartilhar lógica e dados através de múltiplas aplicações e modelos de uso".

Organization for the Advancement of Structured Information Standards (OASIS, 2006) define SOA como:

> Arquitetura Orientada a Serviços é um paradigma para organização e utilização de competências distribuídas as quais estão sob o controle de diferentes domínios proprietários. Ela fornece um meio uniforme de oferecer, descobrir, interagir e usar capacidades para produzir efeitos desejáveis consistentes com pré-condições e expectativas mensuráveis. (OASIS, 2006)

Erl (2005) define os seguintes princípios de uma arquitetura orientada a serviços:

- Reusabilidade do serviço (ou *reusability*): A lógica é dividida dentro dos serviços com a intenção de promover o seu reuso;
- Fraco acoplamento: Serviços mantêm um relacionamento que minimiza dependências e somente requer que eles mantenham o conhecimento da existência um do outro;
- Contrato de serviço: Os serviços aderem a um acordo da comunicação, como definidos coletivamente por um ou mais documentos de descrição do serviço;

- Abstração: Além do que está descrito no contrato de serviços, serviços escondem a lógica para quem está olhando de fora;
- Agregabilidade (ou *composability*): Coleções de serviços podem ser coordenados e montados para formar o menu de serviços;
- Autonomia: Serviços têm o controle sobre a lógica que eles encapsulam;
- Sem estado (ou *stateless*): Serviços minimizam a retenção da informação específica para uma atividade;
- Existência de mecanismos de descoberta de serviço (ou *discoverability*): Os serviços são projetados para serem descritivos de modo que podem ser achados e podem ser avaliados via mecanismos disponíveis de descoberta.

2.1.1. *Enterprise Service Bus*

> Consiste em uma camada intermediária que conecta um conjunto de serviços de negócios reutilizáveis disponíveis dentro de uma empresa. Em outras palavras o ESB é responsável em organizar, interconectar logicamente os serviços e principalmente compartilhar estes com as demais aplicações da empresa. (David A. Chappell[5])

"O ESB fornece um conjunto de capacidades implementadas pela tecnologia de *middleware*, que permite a integração de serviços dentro de uma SOA" (Botto, 2008). O ESB pode ser considerado um middleware, pois é utilizado para o transporte de informações entre programas baseados em plataformas distintas. É importante notar que, dentro da arquitetura, o ESB possui três responsabilidades importantes:

- Fornecer informações necessárias para permitir o roteamento das interações de serviço;
- Fornecer uma taxonomia e detalhes de serviços disponíveis (interfaces) para sistemas participantes de uma arquitetura orientada a serviço;

[5] **David A. Chappell**, fundador da Sonic Software e criador do conceito de ESB, que foi aceito e divulgado pelo *Gartner Group* e hoje é uma realidade nas implementações dos grandes de muitos fornecedores

- Fornecer um ponto controlado de acessos externos aos serviços (propriedade de *gateway*).

2.1.2. As Camadas de uma Arquitetura Orientada a Serviços

Segundo Botto (2008), uma camada representa um conjunto de artefatos com responsabilidade definida dentro da arquitetura. No caso da arquitetura orientada a serviços, as camadas existentes são camada de aplicação, camada de interfaces de serviços e a de processos de negócio. Erl (2005) apresenta um "alto nível de abstração da camada de serviços posicionada entre as camadas dos processos de negócio e de aplicação que demonstra a conectividade aberta entre as camadas de serviço".

> No nível físico, serviços são desenvolvidos em ambientes proprietários, onde eles são individualmente responsáveis pelo encapsulamento (ou acoplamento de informações) de uma aplicação lógica específica para que as informações sejam acessadas a partir de suas interfaces. (Erl, 2005)

A Figura 2.1 mostra como serviços individuais dentro da camada lógica de serviços representam aplicações lógicas originadas de diferentes plataformas.

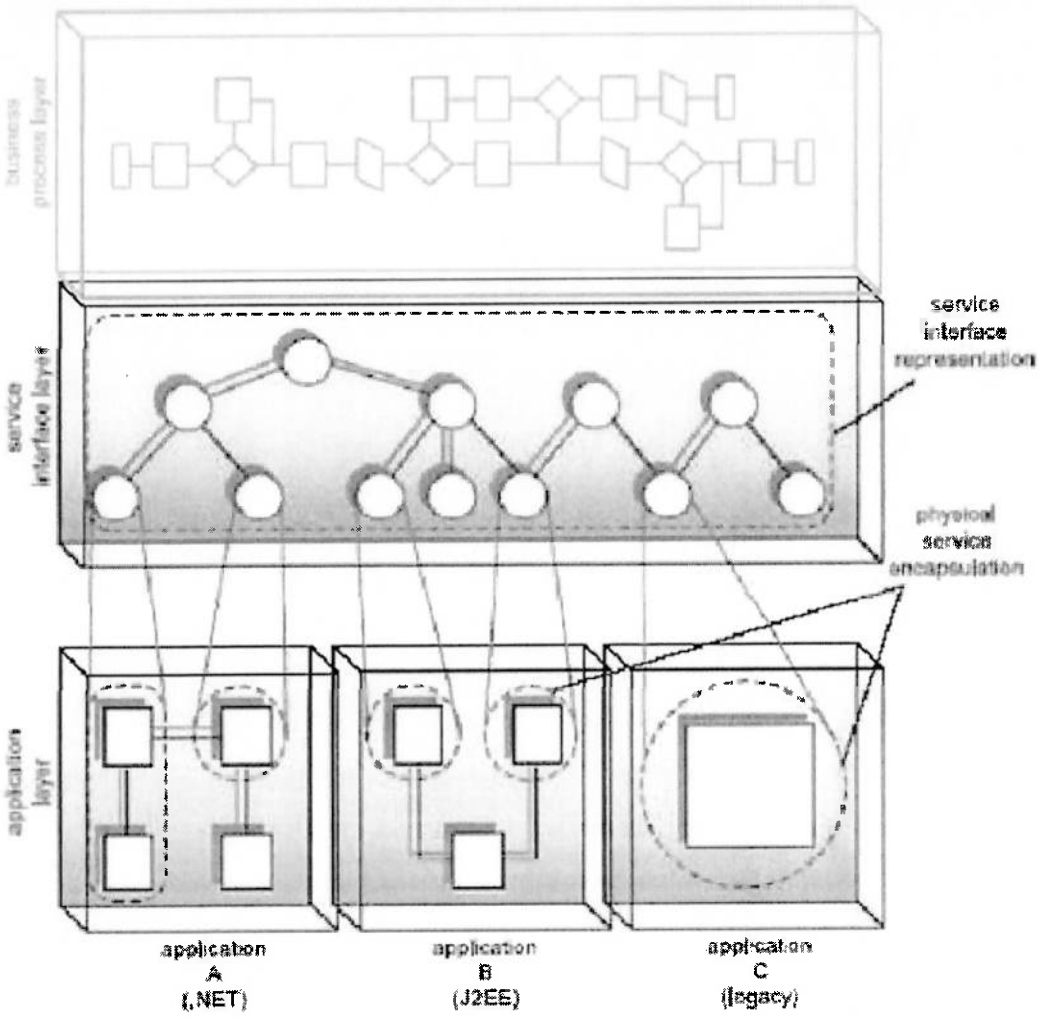

Figura 2.1: Camada de serviços posicionada entre as camadas de processos de negócio e de aplicação (Erl, 2006)

Segundo Erl (2005), um dos objetivos da arquitetura SOA é justamente eliminar a integração *spaghetti* (conhecida pela desorganização da disposição dos serviços com dependências cíclicas e desnecessárias) que existem em grande parte das empresas, através da reutilização dos serviços. A SOA fornece a integração entre processos e serviços, independentemente da plataforma tecnológica, resultando assim no que é chamado de integração com fraco acoplamento, conforme mostra a Figura 2.2.

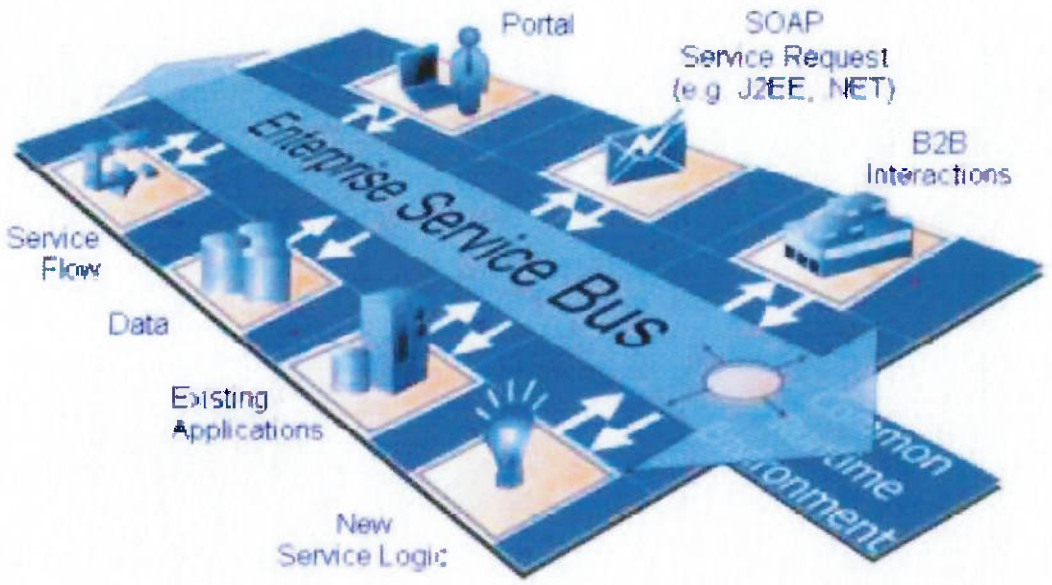

Figura 2.2: Integração com fraco acoplamento permite rapidamente se adaptar às mudanças do negócio (Retirado de *WebSphere SOA and JEE in Practice*. Disponível em: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/woolf/date/200601>.)

De acordo com a figura 2.2, as diversas aplicações (por exemplo, o Portal e o banco de dados) são integradas somente a partir do ESB, e não diretamente com uma ligação direta entre elas. O *Enterprise Service Bus* é o componente de software responsável pela comunicação essas aplicações

2.2. Web Services

De acordo com W3C, o *Web Service* é um componente de *software* desenhado para suportar a interoperabilidade entre computadores em uma rede de computadores. O *Web Service* tem uma interface descrita em uma linguagem chamada WSDL. Outros sistemas se comunicam com o *Web Service* utilizando o protocolo SOAP (*Simple Object Access Protocol*) tipicamente serializado em XML (*Extensible Markup Language*) veiculado utilizando o protocolo HTTP (*HyperText Transfer Protocol*).

> Um *Web Service* é uma abstração que deve ser implementada por um agente concreto. O agente é uma parte concreta de *software* ou *hardware* que manda e recebe mensagens enquanto o serviço é o recurso caracterizado pelo conjunto de funcionalidades que são providas. (W3C)

Segundo Botto (2008), a base do padrão *web services* relevantes ao SOA incluem:

- XML: É uma linguagem de marcação para descrever dados em cargas de mensagens em um formato de documento;
- SOAP: É um protocolo baseado em XML utilizado para trocas de informações num ambiente distribuído;
- WSDL: É um documento XML que descreve um conjunto de mensagens SOAP e a forma como essas mensagens são trocadas;
- UDDI: É uma das formas de localizar um *web service* num registro, similar a um catálogo de páginas, de tal forma que um programa em busca de um determinado serviço possa facilmente localizar e entender o que ele faz.

A relação entre esses conceitos podem ser mais bem demonstrados conforme a Figura 2.3.

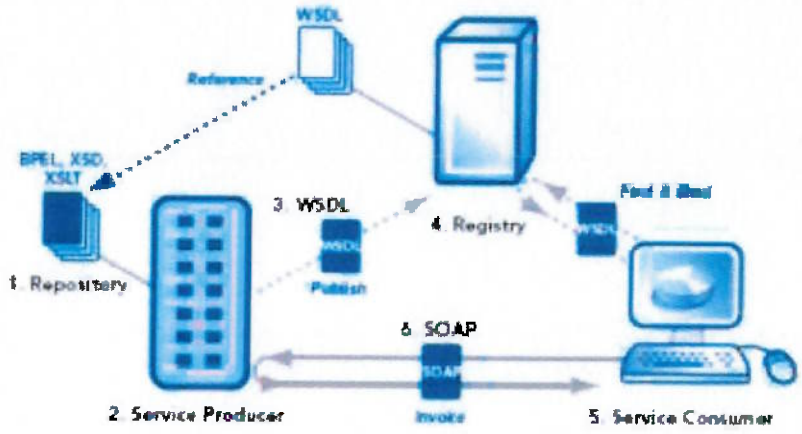

Figura 2.3: Relação dos conceitos envolvidos para os *web services* (Arsanjani, 2004)

Segundo Arsanjani (2004) a SOA pode ser bem representada a partir do seguinte modelo envolvendo três partes principais: o fornecedor do serviço, o usuário e o registro do serviço, também chamado de "*find-bind-execute paradigm*" o que significa paradigma de "procurar-consolidar-executar".

2.3. Business Process Execution Language

De acordo com OASIS[6], o BPEL (*Business Process Execution Language*) é uma linguagem para especificação de processos de negócio com *web services*. Os processos descritos pelo BPEL exportam e importam informação dos serviços utilizando as interfaces dos *web services*.

Ainda segundo OASIS, o BPEL utiliza a WSDL (*Web Services Description Language*) para descrição das informações de entrada e de saída do serviço consumido. Ele ainda suporta a linguagem XML, que o torna flexível. Ele é focado na orquestração dos serviços de negócio, ou seja, especifica processos que envolvem troca de mensagens com outros sistemas, sendo que a sequência das mensagens é controlada por um orquestrador, sendo que o ESB é o responsável pela orquestração das mensagens no caso da arquitetura SOA.

Segundo Botto (2008), o ESB possui as seguintes características:

- Apoiar-se em SOA;
- Possuir funções de transformação, roteamento e gerência para comunicação entre aplicações e serviços;
- Suportar diversos padrões de interfaces e conectores;
- Se apoiar em padrões abertos, sem deixar de lado a segurança;
- Não possuir um ponto integrador centralizado (propriedade conhecida como "operação e gerência distribuídas").

O grande objetivo desta linguagem é, de acordo com *Gartner Institute*, organizar, estruturar e disponibilizar um processo de negócio que possa ser utilizado por várias aplicações. Ainda de acordo com o *Gartner Institute*, esta linguagem fornece várias vantagens para o desenho dos serviços, tais como:

- Definição dos processos de negócio que integram com entidades externas via *web services*;

---

[6] O OASIS (*Organization for the Advancement of Structured Information Standards*) é um "consórcio global que conduz o desenvolvimento, convergência e adoção de padrões abertos para a sociedade de informação global". (http://www.oasis-open.org/home/index.php, tradução nossa)

- Definição dos processos de negócio usando uma linguagem baseada em XML;
- Utilização de conceitos relacionados à orquestração de serviços;
- Utilização de recursos gráficos e de desenho para representação de serviços e sua orquestração;
- Utilização de funções de manipulação de dados consumidos pelos serviços;
- Recursos para apresentar a criação e término de um processo de negócio;
- Utilização de padrão *web services* para apresentar decomposição de processos.

2.4. Planejamento de Projeto

Segundo o PMI[7], o projeto é um esforço que possui início, duração e fim com finalidade de criação de um produto. O planejamento do projeto remete ao processo de estabelecer quais as metas que o projeto almeja alcançar, qual o tempo de duração, quais os custos totais estimados e como as equipes envolvidas serão mensuradas durante o ciclo de vida do projeto. O plano de projeto é baseado nas variáveis tempo (referente ao período de duração do projeto), custo (referente aos custos e investimento a serem feitos) e escopo do projeto (de acordo com as exigências especificadas pelo projeto). Para auxiliar o planejamento de projeto, o PMI provê o guia PMBOK (*Project Management Body of Knowledge*).

Segundo o PMI, o PMBOK é um conjunto de boas práticas em gestão de projetos. O PMBOK possui áreas de conhecimento da gerência de projetos que se aplicam ao projeto inteiro, porém as áreas que estão diretamente relacionadas à fase de planejamento de projetos são:

- Gerenciamento de integração do projeto: Conjunto de processos responsáveis por garantir que as diferentes equipes do projeto trabalhem de forma sincronizada. É importante para o planejamento para estabelecer as metas que o produto deverá atingir;

[7] O PMI (*Project Management Institute*) é uma entidade global voltada ao gerenciamento de projetos.

- Gerenciamento de escopo do projeto: Conjunto de processos responsáveis por definir qual o escopo do projeto. Um dos pilares do planejamento é o escopo definido;
- Gerenciamento de tempo de projeto: Conjunto de processos responsáveis pelo estabelecimento de prazos. A fonte mais importante para a criação do cronograma das tarefas é o tempo de duração do projeto;
- Gerenciamento de custos do projeto: Conjunto de processos responsáveis pelo planejamento de orçamento e custos do projeto;
- Gerenciamento de aquisições do projeto: Conjunto de processos responsáveis pelo controle de aquisições de produtos e serviços consumidos pelo projeto. É importante na parte de contratações necessárias para o projeto;
- Gerenciamento de risco: Conjuntos de processos responsáveis por controlar os riscos e ameaças que podem ocorrer no projeto. É importante no planejamento para identificar os riscos e antecipar possíveis respostas para eles;
- Gerenciamento da qualidade do projeto: Conjunto de processos responsáveis por garantir a qualidade dos produtos entregues pelo projeto. O planejamento da qualidade dos produtos desenvolvidos é o processo executado nesta fase;
- Gerenciamento de recursos humanos do projeto: Conjuntos de processos responsáveis por identificar as responsabilidades existentes no projeto. O planejamento dos recursos humanos é importante para identificar quais os recursos mais especializados para cada papel dentro do projeto;
- Gerenciamento das comunicações do projeto: Conjunto de processos responsáveis por garantir a comunicação entre as equipes envolvidas no projeto. O planejamento das comunicações visa verificar como as equipes irão se interagir.

De acordo com o PMI, algumas das atividades a serem feitas durante o planejamento são:

- Definir as tarefas a serem realizadas: Os gerentes do projeto definem a WBS (*Work Breakdown Structure*) e, a partir dela, as macro atividades são especializadas até o ponto de determinar uma tarefa (menor unidade de trabalho dentro do projeto);
- Identificar os recursos necessários: A partir das definições das tarefas, são identificados os perfis dos profissionais para atender as demandas;
- Definir dependência entre tarefas distintas: Com a definição das tarefas, é possível verificar as dependências entre elas (caso existam) para que possa ser verificado o sequenciamento das atividades
- Criar um cronograma base do projeto: Com base nas tarefas e suas dependências, o cronograma é montado com as devidas atribuições de atividades para os responsáveis.

Dentro do planejamento, é realizada análise dos Fatores Críticos de Sucesso (FCS) para reforçar o motivo de sua implementação.

2.4.1. Fatores Críticos de Sucesso

Segundo Rockart[8] (1979), os Fatores Críticos de Sucesso (FCS) consistem em um conjuntos de itens contendo um número limitado de pontos chave que irão determinar o sucesso ou o fracasso de uma meta definida por um planejamento de determinada corporação. "Quando os FCS são definidos, eles se tornam um ponto de referência para toda a organização em suas atividades voltadas para atingir o objetivo.". (Rockart, 1979).

Ainda de acordo com Rockart (1979), os Fatores Críticos de Sucesso de qualquer indústria devem satisfazer três critérios genéricos:

1. Aplicabilidade a todos os concorrentes de um determinado segmento de negócio;
2. Relevância decisiva do objetivo dentro do contexto da corporação;
3. Possibilidade de controle do objetivo alcançado pelas empresas.

[8] John F. Rockart estuda o papel da tecnologia da informação e a implementação de integração de sistemas globais. (http://mitsloan.mit.edu/faculty/detail.php?in_spseqno=SP000111&co_list=F, tradução nossa)

2.5. Fase de Análise

Nesta fase, são levantados os requisitos funcionais que serão considerados como base para o desenvolvimento de um projeto.

O RUP[9] (2004) define as seguintes tarefas para ser realizadas na fase de análise:

- Transformar os requisitos em um projeto do que o sistema vai ser a partir das informações passadas pelo cliente;
- Construir uma arquitetura robusta para o sistema;
- Adaptar o projeto para as limitações do ambiente de execução.

Ainda segundo o RUP (2004), são produzidos artefatos (também conhecidos como artefatos ou *deliverables*) que serão utilizados nas fases seguintes do projeto. Entre eles, os que serão abordados nesta monografia são o escopo, os requisitos do projeto e os cenários de testes integrados.

2.5.1. Escopo de Projeto

Escopo é definido como um conjunto limitado de metas e objetivos a serem alcançados. No contexto de Projetos, o Escopo pode ser definido como um conjunto de requisitos que deverão ser atendidos pelo produto desenvolvido depois de pronto.

2.5.2. Requisitos do Projeto

Requisitos de projetos podem ser definidos como "(...) uma condição ou capacidade necessitada por um usuário para resolver um problema ou alcançar um objetivo" (RUP). Os requisitos podem ser classificados em funcionais e não funcionais.

De acordo com o RUP (2004), os requisitos funcionais são os que descrevem o comportamento do sistema, ou seja, indicam quais as funcionalidades que o produto deve oferecer ao usuário.

---

[9] O RUP (*Rational Unified Process*) "é uma plataforma de processo de desenvolvimento de software configurável que oferece melhores práticas comprovadas e uma arquitetura configurável." (retirado de <http://www-01.ibm.com/software/br/rational/rup.shtml>).

Já os requisitos não funcionais são os que descrevem como o sistema deve ser feito, estando mais relacionados com a qualidade do produto. Segundo Erl (2005), dentro do contexto do projeto SOA, os requisitos não funcionais mais comuns são:

- Segurança: Também conhecido como Segurança da Informação, está relacionado com a proteção de um conjunto de dados. O nível de segurança de acesso desejado pode impactar a especificação e a construção da Arquitetura Orientada a Serviços;
- Desempenho: Tempo total entre o momento em que o cliente submeteu a requisição até o momento em que a resposta foi retornada por completo;
- Vazão de requisições: Quantidade de requisições executadas por unidade de tempo em uma SOA;
- Interoperabilidade: A arquitetura deverá estar preparada para operar a comunicação entre aplicações desenvolvidas em diversas tecnologias;
- Manuteniblildade: A arquitetura deverá fornecer mecanismos para minimizar problemas de disponibilidade caso haja a manutenção da mesma;
- Disponibilidade: Fração de tempo em que o site está operacional para o cliente;
- Escalabilidade: Habilidade de alterar o esforço fornecido pela infraestrutura de acordo com a demanda corrente. A adição de novos recursos pode se dar de duas formas: aumento do número de servidores (escalabilidade horizontal) ou incremento do poder de servidores existentes, através do aumento de memória e processamento (escalabilidade vertical). Isso é importante para deixar o desempenho da aplicação praticamente o mesmo, independentemente da vazão de requisições.

2.5.3. Cenários de Testes Integrados

Segundo o RUP (2004), testes integrados consistem nos testes responsáveis pela verificação se os componentes de *software* estão funcionando de forma adequada quando combinados com outras peças do sistema, garantindo assim

a comunicação entre sistemas ou componentes distintos. Esses testes deverão abordar todas as situações (casos ou cenários) em que o produto está sujeito a passar (incluindo casos de sucesso ou falha da aplicação). Os cenários de teste deverão descrever:

- Quais componentes estão sendo testados (por exemplo, se é um serviço compartilhado específico ou uma funcionalidade da arquitetura);
- Quais as entradas necessárias para execução de cada condição de teste;
- Quais as saídas esperadas após a execução do cenário de teste;
- Quais os passos necessários para realização da condição de teste, incluindo pré-requisitos;
- Quais os pós-requisitos gerados a partir da realização do teste.

2.6. Fase de Especificação

Na fase de especificação, a principal meta é produzir quaisquer artefatos que têm o objetivo de ser base na fase de desenvolvimento do sistema-alvo. Esses artefatos são denominados especificações e são geradas a partir de informações recolhidas na fase de análise. De acordo com o RUP (2004), a linguagem mais utilizada para descrever as especificações é o UML (*Unified Modeling Language*). As especificações deverão abordar tanto requisitos funcionais (segundo o RUP, aqueles que descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, é aquilo que descreve o que tem que ser feito pelo sistema) e não funcionais (ainda de acordo com o RUP, aqueles que expressam como deve ser feito e como se relacionam com padrões de qualidade como confiabilidade e desempenho).

De acordo com o RUP (2004), é necessário criar mais de um tipo de especificação, pois cada um deles pode se destinar a públicos-alvo diferentes e, dada a complexidade do sistema, as especificações podem ser segmentadas, dando foco nos diversos componentes a serem implementados.

Neste ponto do processo, de acordo com Botto (2008), há preocupações de a visão geral do produto estar estável, de o planejamento do projeto estar confiável e viável e de os custos, admissíveis.

Entre as especificações, este trabalho de monografia irá abordar os diagramas de classes e de sequência (baseados no *Unified Modeling Language*) e diagramas baseados em *Business Process Execution Language* (BPEL). Essas especificações serão utilizadas na fase de desenvolvimento do sistema.

2.7. Fase de Desenvolvimento

Segundo o RUP (2004), a fase de desenvolvimento corresponde à construção do componente de software a partir de uma determinada especificação. O desenvolvimento é realizado utilizando ferramentas de construção também conhecidas como IDEs (*Integrated Development Environment*). Nesta fase, atuam os profissionais com perfil mais técnico que estejam em contato diretamente com a tecnologia para a criação do sistema. Segundo a IBM[10] (2010), os profissionais de fábrica de *software* são muito utilizados devido aos conhecimentos técnicos e específicos que possuem.

Sob o ponto de vista do RUP (2004), o principal produto desta fase é a construção da solução e as documentações dos testes realizados.

2.8. Unified Modeling Language

De acordo com o RUP (2004), a modelagem é uma parte essencial da engenharia, arte e construção por séculos. Desenhos de sistemas de software complexos que seriam dificilmente descritos de forma textual podem ser convertidos em forma de diagramas para facilitar sua leitura. A modelagem provê três principais benefícios: visualização, gerenciamento da complexidade e comunicação clara. O UML (*Unified Modeling Language*) é uma linguagem visual para especificação, construção e documentação dos artefatos de sistemas. O UML pode ser utilizado em todos os processos do ciclo de vida do desenvolvimento de software, e na implementação de diferentes tecnologias. O UML foi aprovado como um padrão em 1997 pelo OMG[11] (2001) e, desde então, tem sofrido algumas alterações em sua linguagem.

---

[10] A IBM é a empresa que mais investe na arquitetura orientada a serviços. A fonte desta informação foi retirada de <http://www.ibm.com/developerworks/rational/library/253.html>.

[11] O OMG (*Object Management Group*) é uma empresa global responsável por desenvolver padrões de integração que possam prover valor às corporações. (http://www.omg.org/gettingstarted/gettingstartedindex.htm, tradução nossa)

2.8.1. Diagrama de Caso de Uso

De acordo com o OMG (2001), o Diagrama de Caso de Uso possui o objetivo de apresentar uma visão externa do sistema, representando a interação entre um agente externo (por exemplo, o usuário do sistema) e a aplicação.

O agente externo ao sistema representado no diagrama de caso de uso é definido como Ator. É muito importante saber que o Ator nem sempre é uma pessoa (por exemplo, um recepcionista de hotel que acessa o sistema de reservas), mas também pode ser outro sistema (este é o caso mais comum desta monografia, que visa à comunicação entre sistemas distintos).

Os atores interagem com o sistema a partir de um caso de uso. O caso de uso é responsável por representar uma funcionalidade do sistema que será acessado pelo Ator. Desta forma, existem relações entre os Atores e os Casos de Uso. O Caso de Uso não apresenta detalhes de como a funcionalidade relacionada está construída.

A figura 2.4 apresenta um exemplo de diagrama de caso de uso:

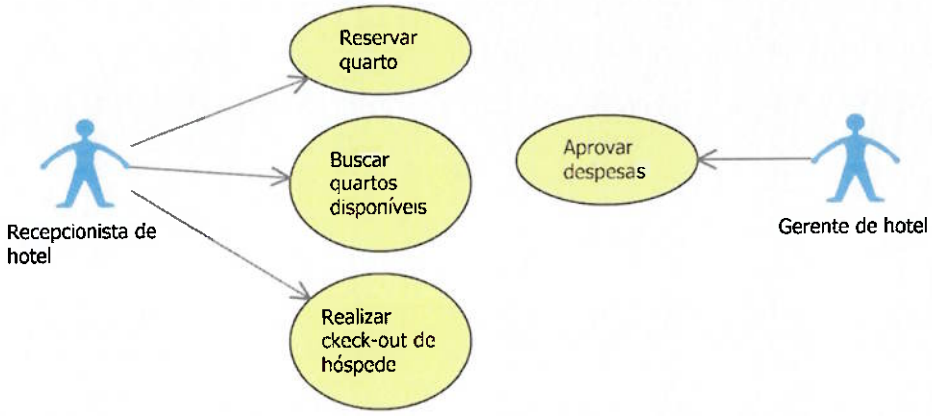

Figura 2.4: Exemplo de diagrama de caso de uso

2.8.2. Diagrama de Classes

De acordo com o OMG (2001), o Diagrama de Classes apresenta uma visão da estrutura estática interna ao sistema. Ele é composto por Classes (responsável por descrever um conjunto de objetos pelas suas características e operações)

e seus relacionamentos (que indicam como as classes se relacionam, apresentando o tipo de relacionamento, a navegabilidade e a multiplicidade).

Os relacionamentos podem ser Associação (uma classe está associada a outra), Agregação (uma classe possui um conjunto de objetos de outra classe), Composição (uma classe é composta por objetos de outras classes) ou Herança (uma classe herda as características de outra classe).

A figura 2.5 apresenta um exemplo de diagrama de classes.

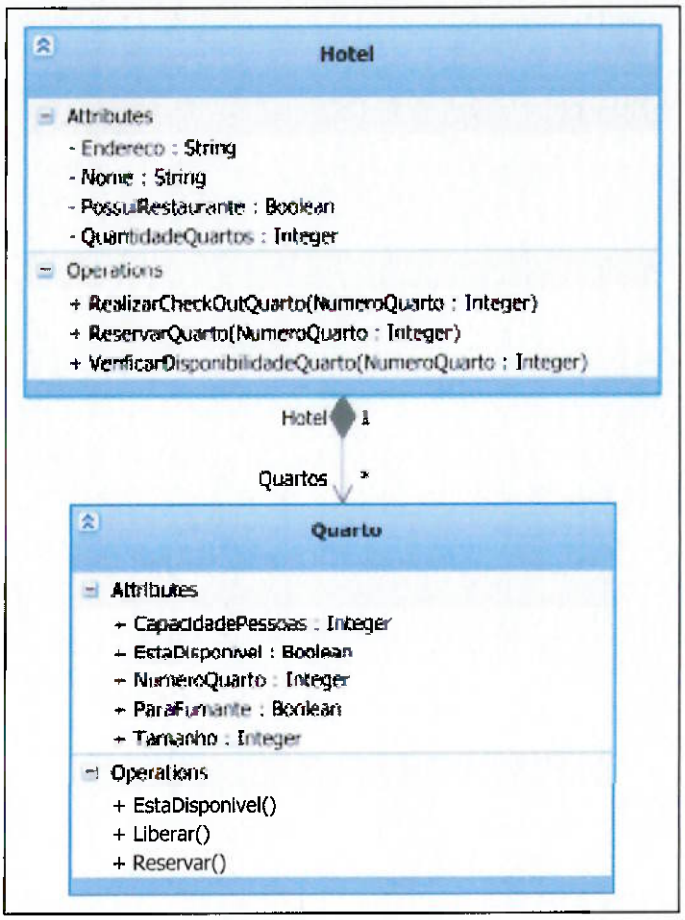

Figura 2.5: Exemplo de diagrama de classes

2.8.3. Diagrama de Sequência

De acordo com o OMG (2001), ao contrario do Diagrama de Classes, o Diagrama de Sequência apresenta uma visão dinâmica do sistema. Ela é centrada nos comportamentos dos objetos, apresentando suas interações, as sequências das mensagens trocadas (em ordem temporal), o acionamento de

métodos e o ciclo de vida dos objetos. É possível representar loops e condicionais neste diagrama.

A figura 2.6. apresenta um exemplo de diagrama de sequência:

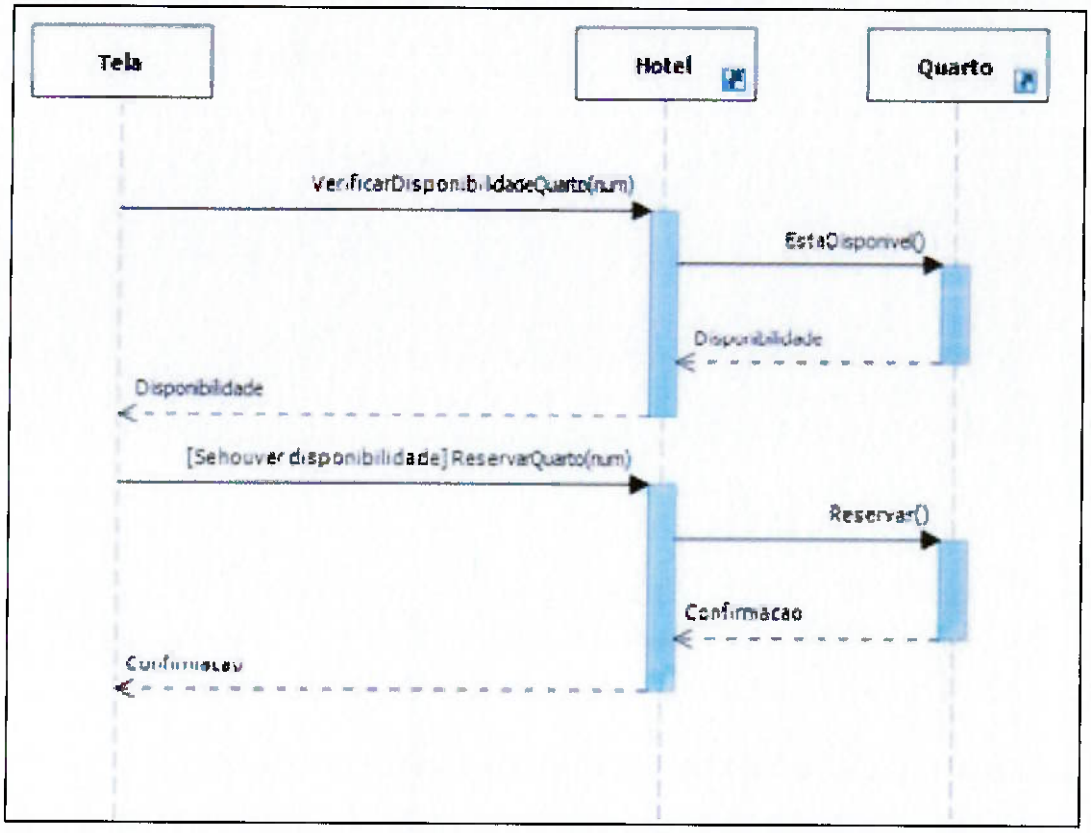

Figura 2.6: Exemplo de diagrama de sequência

3. MATERIAIS E MÉTODOS

3.1. Desafios de metodologia

Os maiores desafios de implementar uma Arquitetura Orientada a Serviços podem ser resumidas nos seguintes pontos:

- Curto prazo de implementação;
- Baixa tolerância para fracasso;
- Levantamento de requisitos dinâmicos.

Visando esses pontos, é necessária uma metodologia capaz de fornecer uma sequência lógica de processos durante a fase inicial de planejamento SOA, não deixando de ser adaptável e ágil para realizar as mudanças, quando necessárias.

3.2. Proposta da metodologia

De acordo com Botto (2008), projetos de desenvolvimento de uma Arquitetura Orientada a Serviços possuem mais complexidade do que projetos de desenvolvimento de produtos comuns de software, devido à necessidade de um maior tempo de projeto e equipes multidisciplinares que requerem comunicação constantemente durante o processo de desenvolvimento. Segundo o *Gartner Institute*, a visão dos resultados esperados não é clara no começo de um projeto SOA, mas ficam claras durante a fase de desenvolvimento, devido ao objetivo de integração de sistemas através da arquitetura desenvolvida. Muitas vezes, a implantação de uma SOA em uma empresa é muito benéfico, desde que ela passe a seguir uma série de regras para adaptar as aplicações dentro da arquitetura-alvo.

Como proposta desta monografia, será apresentada uma metodologia contendo uma série de fases e, para cada uma delas, há uma lista de atividades. A metodologia proposta é baseada em quatro fases apresentadas na figura 3.1:

Figura 3.1: Fases da metodologia proposta

As fases descritas nesta monografia são:

- **Fase de Planejamento de Projetos Orientado a Serviços**: Fase responsável pelo planejamento do projeto SOA;
- **Fase de Análise Orientada a Serviços**: Fase responsável por abordar os requisitos para a construção da arquitetura orientada a serviços;
- **Fase de Especificação Orientada a Serviços**: Fase responsável pela criação dos desenhos e especificações para construção da arquitetura;
- **Fase de Desenvolvimento Orientado a Serviços**: Fase de desenvolvimento e construção da arquitetura.

Antes de aplicar a metodologia proposta, é necessária a análise dos seguintes pontos para verificar se ela é aplicável ou não para um determinado projeto:

- Existência de uma arquitetura orientada a serviços: Caso o projeto envolva uma corporação que já possua uma SOA consolidada com os objetivos estratégicos já alinhados, a metodologia proposta não é aplicável. Nos casos de a corporação não possuir uma arquitetura orientada a serviço ou até mesmo possuir, porém sem ter os objetivos estratégicos alinhados com os processos de negócios baseados na SOA, a metodologia é aplicável;
- Mapeamento dos processos de negócio: Um dos pré-requisitos do projeto é o mapeamento dos processos de negócio dentro do contexto corporativo. A metodologia é aplicável a partir deste mapeamento existente;
- Existência de sistemas críticos: A metodologia é aplicada a empresas que possuem mais de um sistema crítico que necessite de integração para atender novas demandas. Caso não haja, a metodologia não é aplicável.

3.3. Planejamento Orientado a Serviços

De acordo com Erl (2005), SOA tem um grande potencial para promover a eficiência dos processos de TI em uma organização, mas, para implementá-la, é necessário muito mais do conhecimento técnico: conhecimentos e habilidades em gerenciamento de projeto, gestão de pessoas e percepção do impacto causado pela implantação de uma arquitetura orientada a serviços também são fundamentais. O planejamento do projeto é importante para que os objetivos de negócio nos níveis estratégico, funcional e operacional sejam atendidos e possibilite o atendimento dos requisitos do sistema. A partir do planejamento, é possível verificar o que fazer, como fazer, quem deve fazer e como medir o esforço.

Nesta fase, as tarefas a serem realizadas são estabelecer uma visão SOA para a corporação e criar o cronograma do projeto, conforme a figura 3.2.

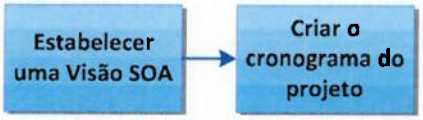

Figura 3.2: Etapas da fase de planejamento orientado a serviços

3.3.1. Etapa 1: Estabelecer uma Visão SOA

Os clientes nem sempre compreendem os benefícios que uma arquitetura orientada a serviços pode trazer em seus negócios, portanto, as equipes que definem a estratégia SOA necessitam influenciar as equipes do projeto para ajudá-los a identificar os problemas no negócio. Os gerentes e os analistas de projeto precisam analisar os negócios dos clientes para determinar as vantagens que uma solução SOA pode trazer para a corporação, mesmo que necessite da reformulação dos processos de negócio. Os fatores-chave que os analistas devem examinar são as operações internas dos clientes, interações com seus sócios, fornecedores, e consumidores e seu modelo completo de negócio, fazendo com que os analistas adotem a iniciativa de desenvolver e possuir uma estratégia e uma visão de SOA.

Para criar uma arquitetura orientada a serviços, é necessário estabelecer uma clara visão do que o SOA será, quais os limites ou fronteiras que ela atingirá na

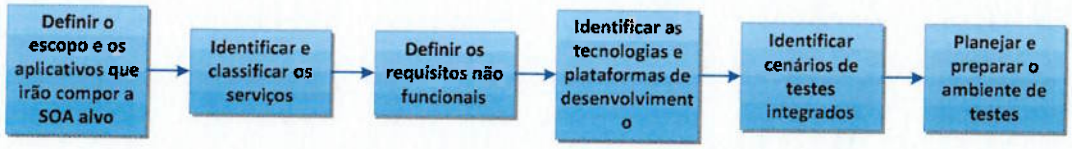

Figura 3.3: Etapas da fase de análise orientada a serviços

3.4.1. Etapa 1: Definir o escopo e os aplicativos que irão compor a SOA alvo

O escopo inicial da SOA alvo é o conjunto de requisitos extraídos e funcionalidades-alvo que o projeto de desenvolvimento da arquitetura orientada a serviços deverá atender. Para determinar esse escopo, é necessário considerar quais unidades de negócio são relevantes no contexto da análise (de acordo com a criticidade e o nível de dependências de outros setores corporativos). De acordo com o modelo de negócio corporativo, com que frequência uma aplicação é acessada, por quais outras aplicações é acionada e qual a disponibilidade necessária, é possível saber se um sistema é crítico para uma organização e se deve ser construído um serviço para acessá-lo.

Uma vez levantado o escopo, são definidos os requisitos que a SOA irá abordar. Neste ponto do processo, há interações com os envolvidos com o projeto (também conhecidos como *stakeholders*), incluindo representantes de cada setor da empresa impactado pela implantação da arquitetura orientada a serviços. Isso acarretará no levantamento dos processos mais críticos, quais deverão ser considerados de maior prioridade e, após a implantação, de como os setores irão se interagir a partir da SOA. O acompanhamento de um representante de cada setor é importante, pois o projeto envolve a integração entre os diversos setores, cujas aplicações serão providas de comunicação.

Os requisitos levantados serão utilizados nas próximas etapas desta metodologia.

3.4.2. Etapa 2: Identificar e classificar os serviços

Uma vez identificadas as aplicações as quais serão acessadas por serviços, o próximo passo é identificar os serviços que serão construídos e agrupar as operações de serviços em classificações de acordo com a severidade,

organização, bem como quais benefícios (tangíveis e intangíveis) a corporação-alvo será capaz de obter com sua implementação. "As empresas insistem em implementar uma arquitetura SOA sem claramente identificar o valor obtido ou o resultado ideal a ser alcançado para o negócio com sua implementação" (Botto).

Assim como dito na teoria, os Fatores Críticos de Sucesso (FCS) são as habilidades e os recursos que a empresa ou algum projeto precisa necessariamente ter para atingir os objetivos propostos. Na visão SOA, devem ser incluídos os Fatores Críticos de Sucesso para ressaltar os objetivos e benefícios que uma implantação bem sucedida de SOA pode trazer à corporação. Alguns dos Fatores Críticos de Sucesso presentes em projetos SOA são o retorno financeiro a longo prazo, a facilidade de integração de novos produtos desenvolvidos e o atendimento de demandas e necessidades mais complexas.

Para estabelecer uma visão SOA para a empresa, é importante que as vantagens que a arquitetura pode trazer para a corporação sejam apresentadas para todos os interessados no projeto. As vantagens podem ser classificadas em dois grupos: vantagens para o desenvolvimento de projetos e vantagens para a estratégia de negócio da SOA.

Visando o desenvolvimento de novos projetos, a propriedade de reutilização de serviços é a base das vantagens que a arquitetura pode trazer para a gestão de TI. O reuso de *software* traz como consequências:

- Diminuição de redundância de código: Como os serviços são tratados como "caixas pretas", ou seja, somente as suas interfaces são expostas sem dar detalhes de suas implementações, a codificação fica centralizada em um único lugar, fazendo com que a lógica de programação seja centralizada;
- Redução de tempo de construção: Com os serviços já disponibilizados, partes das aplicações já estão construídas, fazendo com que seja necessária somente uma chamada ao serviço publicado para acessar uma determinada funcionalidade. Essa vantagem também pode ser

considerada como aumento de produtividade da equipe de desenvolvimento;

- Redução de tempo de manutenção: Caso seja necessária a manutenção em um determinado serviço, a quantidade de pontos a serem modificados é menor que o número de aplicações que o acessam, reduzindo assim o tempo de manutenção dos sistemas;
- Redução de tempo de testes: Como o código do serviço seja separado das aplicações que somente os acessam, não é necessário realizar testes unitários do serviço e partindo diretamente para os testes de integração da aplicação com o serviço publicado.

Visando as vantagens para a estratégia de negócio, a implantação de uma arquitetura orientada a serviços faz com que as corporações remodelem os processos de negócio para se adequarem com a SOA. Isso forçará os serviços a ficarem mais alinhados com os processos de negócio. O mapeamento entre as aplicações e os processos corporativos é de vital importância para a empresa para atender as demandas solicitadas e identificar possíveis gargalos dentro do macro processo corporativo.

A implantação de uma arquitetura orientada a serviços faz com que a área de TI, mais especificamente a de desenvolvimento de sistemas, sofra impactos e seja forçada a mudar a metodologia de construção de novas aplicações. O maior impacto é a mudança da forma que novas aplicações e serviços serão desenvolvidos com a implantação de uma nova arquitetura que provê um paradigma diferente de integração de sistemas, baseada nos processos de negócio presentes na corporação. Esta deverá trabalhar como uma linha de montagem, ou seja, cada equipe de desenvolvimento é responsável por cada componente do sistema e, quando integrados, os componentes formam o sistema. Esse processo seria similar a um processo de construção de um carro: cada máquina constrói um componente comum aos diversos carros. No caso do desenvolvimento de sistemas, a equipe especializada será responsável por construir os serviços para ser consumidos pelas outras aplicações. Na área de TI como um todo, isso é benéfico tanto para a equipe responsável por desenvolver os serviços (aumento a produtividade devido à especialização do trabalho), quanto para as equipes de desenvolvimento de

outras aplicações customizadas (que têm parte das soluções construídas, aumentando a produtividade).

Para fazer uma análise dos impactos na área de TI da corporação, é necessário fazer um estudo de como os processos estão no momento antes do projeto e como ficarão após a implantação da SOA, envolvendo os sistemas externos (que envolvem os clientes ou não). Certamente o processo de desenvolvimento de novos sistemas será alterado e acelerado devido à reutilização de serviços já publicados. Os setores da empresa que possuírem aplicações que estiverem no escopo do projeto sofrerão impactos, visando benefícios devido à integração com outros sistemas de forma automatizada.

3.3.2. Etapa 2: Criar o cronograma do projeto

Para criar o cronograma do projeto, é necessário fazer o planejamento focando no investimento. Segundo o *Business Performance Management Institute*[12], processos que envolvem clientes são os que dão maior retorno financeiro às empresas. "Aprimorar estes aplicativos (relacionados com clientes) em 10% é melhor do que aprimorar aplicativos de nível mais baixo em 50%" (*Gartner Institute*).

Desta forma, os processos que envolvem os clientes devem ser priorizados, entrando no escopo inicial do projeto com o objetivo de balançar os custos e os investimentos realizados no projeto. Após priorizar os processos relacionados aos clientes, são analisados os processos em que a corporação esteja assumindo o papel de cliente, ou seja, esteja consumindo serviço de serviços externos à corporação (por exemplo, processo de aquisição de passagens aéreas para alocações externas ao centro de custo) e, então, analisar processos internos.

A partir disso, são definidas as macro tarefas que devem ser executadas de acordo com os processos escolhidos. As dependências entre as macro tarefas são representadas através de diagramas de Gantt. Esse tipo de diagrama possibilita a visualização de caminhos críticos do projeto, o que indica quais

[12] O BPMI e o OMG possuem atividades de BPM (*Business Process Management*) que trazem padrões de liderança e industrialização para a TI como um todo.

atividades não poderão possuir atrasos para início de outras. Em um projeto SOA, o esforço destinado às fases depende da maturidade da empresa referente à visão de arquitetura orientada a serviços, ou seja, caso a empresa já possua uma estratégia SOA, a fase de análise poderá ser reduzida. Caso a empresa adote uma postura que tenha auxílio de parceiros que possam prover suporte (ou até mesmo terceirização na execução das tarefas) nas fases de especificação e desenvolvimento (incluindo testes), essas poderão ser reduzidas, porém o custo poderá ser maior devido ao suporte especializado dessas empresas parceiras.

3.4. Análise Orientada a Serviços

A fase de Análise Orientada a Serviços é a fase na qual os requisitos (ou seja, definições documentadas de uma propriedade ou comportamento que um produto ou serviço particular deve atender) necessários para o projeto de desenvolvimento de uma arquitetura orientada a serviços são levantados. Ela é a mais crítica do projeto SOA, pois serão definidos quais são, como devem ser e quais as formas de integração entre os serviços e as aplicações que irão utilizá-los. Esta análise não passa a ser definitiva, mas sim complementar, pois deve se juntar a outras existentes na corporação para que a arquitetura possa suportar todos os processos de negócio passíveis a serem encapsulados em serviços e que a arquitetura, caso exista, possa evoluir de acordo com as necessidades corporativas. Um dos pontos mais importantes é a reorganização dos processos de negócio, pois a arquitetura orientada a serviços será baseada nessa remodelagem.

Na fase de análise orientada a serviços, as tarefas a serem vistas no projeto são definir o escopo e os aplicativos que irão compor a SOA alvo, identificar e classificar os serviços, definir os requisitos não funcionais, identificar cenários de testes integrados e planejar e preparar o ambiente de testes, conforme a figura 3.3.

frequência de acesso e dependência dentro do contexto da corporação. Estas classificações acabam formando os candidatos a serviços que mais tarde irão compor um menu de serviços da arquitetura SOA. Inicialmente, é necessário identificar os serviços críticos que precisam ser desenvolvidos e as lógicas de negócio que deverão ser implementadas em cada serviço. Em seguida, é necessário classificar as operações de serviços e, se possível, identificar o reuso potencial das lógicas encapsuladas.

Dentro desta etapa, são realizadas as seguintes atividades:

- Decompor os processos de negócio da corporação;
- Descartar passos do processo não apropriados para encapsulamento do serviço;
- Criar os casos de uso para os candidatos a serviços;
- Aplicar os princípios da SOA aos serviços candidatos.

3.4.3.1. Decompor os processos de negócio da corporação

O escopo de um serviço de negócio pode variar, devido à importância, a abrangência e a criticidade dentro de um processo de negócio. Por exemplo, um serviço pode representar apenas um passo dentro de um processo de negócio, uma parte de um micro processo dentro de um processo maior, ou até mesmo um processo inteiro. Desta forma, torna-se necessário verificar a granularidade dos processos de negócio documentados (ou seja, o quanto os processos estão divididos no macroprocesso) e, caso não estejam no nível mais detalhado possível (por exemplo, representar um macroprocesso), este detalhamento será necessário até chegar à atomicidade do serviço, ou seja, o serviço possui maior granularidade possível. A atomicidade do serviço faz com que ele possa ser reutilizado por diversos sistemas distintos.

3.4.3.2. Descartar passos do processo não apropriados para encapsulamento do serviço

Algumas atividades dentro dos processos de negócio podem ser facilmente identificadas como não pertencente à lógica que deve ser encapsulada por um serviço candidato. A melhor forma de identificar os processos não apropriados

são os que não possuem características similares às propriedades que o SOA oferece. Alguns exemplos disso são processos utilizados exclusivamente por uma única aplicação, processos que armazenem estados (também chamados de *Stateful*), com forte acoplamento, processos sem mecanismos de *discoverability* e que se comunicam via APIs (*Application Programming Interface*). Todos os exemplos citados infringem as propriedades que a arquitetura orientada a serviços pode oferecer.

Caso os passos de processos não apropriados para o encapsulamento do serviço façam parte de um macroprocesso candidato a serviço, eles deverão ser forçados a incorporarem no candidato, mas sabendo que eles não representarão um serviço por si só. Caso contrário, é necessário verificar se os passos do processo deverão ser modificados caso haja interações com passos a serem candidatos a serviços. Nesta situação, eles deverão ser modificados não para representar um serviço, mas para acessar outro serviço e se adequar com a SOA pretendida.

3.4.3.3. Criar os casos de uso para os candidatos a serviços

Para cada candidato a serviço, é necessário definir os casos de uso que contextualizarão o serviço, partindo de suas funcionalidades (ou requisitos funcionais). Para isso, as funcionalidades referentes aos candidatos devem mapear os casos de uso, descrevendo o ator e fluxos.

Cada funcionalidade levantada irá representar um caso de uso dentro do contexto do sistema. É importante que o nome do caso de uso resuma a ação da funcionalidade (para isso, é sugerido que o nome do caso de uso inicie com um verbo, identificando a ação principal a ser descrita). Ao detalhar a sequência lógica do fluxo principal (e alternativos, caso existam) de cada caso de uso, os passos deverão destacar alguns pontos importantes para adequação ao paradigma de SOA:

- Ressaltar que o ator do caso de uso é um sistema externo ao serviço generalizando o solicitante, pois sistemas distintos podem acessar o serviço candidato;

- Acessar as interfaces de outros serviços, caso necessário (o que resulta em inclusões de outros casos de uso);
- Esperar que as informações de entrada estejam em formato de alguma linguagem extensível, assim como as informações de saída devem ser geradas partindo da mesma linguagem;
- O caso de uso não poderá apresentar pré e pós-condições, pois não pode armazenar estado (conforme visto que os serviços são *stateless*).

3.4.3.4. Aplicar os princípios da SOA aos serviços candidatos

Para que os serviços candidatos passem a ser considerados como serviços da SOA, é necessário realizar alguns ajustes neles, aplicando os princípios de orientação a serviços para que sejam utilizados de forma mais adequada dentro da arquitetura construída, minimizando problemas de integração e, para os considerados críticos, sejam disponibilizados e aproveitados de forma adequada. Segundo Botto (2008), os princípios a serem levados em consideração na adequação dos candidatos a serviços são a reusabilidade (por exemplo, o serviço de "Login" poderia ser utilizado por diversas aplicações corporativas que solicitem autenticação do usuário que as acessam), a padronização das formas de acesso (preferindo formatos extensíveis como o XML e descartando interfaces proprietárias, como as APIs – *Application Programming Interface*), a identificação de interfaces de serviços definidas e não armazenar estados da aplicação.

Para alcançar este objetivo, é necessário:

- Garantir que cada caso de uso que representa as funcionalidades de serviços comuns sejam incluídos por pelo menos um caso de uso diferente;
- Fazer com que linguagens com formatos extensíveis (por exemplo, o XML) sejam as utilizadas para comunicação entre sistemas distintos;
- Utilizar um catálogo de nomes de serviços, garantindo que cada serviço seja único dentro da SOA construída;
- Disponibilizar os *templates* das mensagens que são esperadas por cada serviço (por exemplo, no caso de utilização da linguagem XML, as

interfaces podem ser definidas a partir de XSDs – *XML Schema Definition* – que servem como definir os formatos das mensagens);

- Fazer com que as aplicações não armazenem o estado corrente do sistema, por exemplo, removendo variáveis estáticas usadas pela aplicação.

3.4.3. Etapa 3: Definir os requisitos não funcionais

Como qualquer tipo de aplicação, o levantamento de requisitos não funcionais é muito importante, pois são eles que descrevem a preocupação com o desempenho e outros atributos de qualidade, afetando diretamente a satisfação dos envolvidos com o projeto. Este levantamento deverá ser feito e documentado para ser um dos direcionadores da fase desenvolvimento e, como qualquer requisito, eles deverão ser atendidos.

Os requisitos não funcionais relevantes para o projeto de desenvolvimento SOA são:

- Segurança: É importante garantir que as informações trocadas estejam envolvidas por serviços e aplicações conhecidas pela arquitetura construída;
- Desempenho: A arquitetura deverá oferecer escalabilidade para que não haja prejuízos nos desempenhos das trocas de informações na arquitetura. A escalabilidade oferecida está diretamente relacionada com a vazão das mensagens solicitadas, ou seja, quanto maior a demanda, maior a escalabilidade necessária;
- Interoperabilidade: Os serviços que estiverem disponíveis na SOA devem dispor de um protoloco flexível para garantir a interoperabilidade. A linguagem XML é a mais indicada no caso da SOA;
- Manutenção: O processo de manutenção envolve a parada da arquitetura para manutenção. Neste caso, é interessante acionar a infraestrutura secundária para substituir a primária durante o período de manutenção, mas há a opção de deixar a arquitetura indisponível em períodos de pouca ou nenhuma utilização;

- Disponibilidade: É necessário haver um SLA (*service level agreement*) para a disponibilidade da arquitetura.

3.4.4. Etapa 4: Identificar as tecnologias e plataformas de desenvolvimento

De acordo com a Sun (2004), um dos fatores de sucesso de um projeto SOA está diretamente relacionado à escolha da tecnologia e ferramentas utilizadas para a entrega da solução. A ferramenta escolhida deve suportar o desenvolvimento integrado e os padrões de implementação, mais especificamente os protocolos de *Web Services.* Mas vale lembrar que a escolha das ferramentas e tecnologias apropriadas não é suficiente para o sucesso do projeto por si só.

Os principais fatores a serem analisados para a escolha das tecnologias apropriadas para um determinado projeto SOA são:

- Habilidades dos desenvolvedores: Quanto mais conhecimento o time de desenvolvimento possuir para uma determinada ferramenta e tecnologia, a fase de desenvolvimento será mais rápida e os requisitos serão mais bem atendidos de acordo com as expectativas de negócio;
- Habilidades da equipe de suporte: Assim que como o item anterior, haverá melhores resultados caso as habilidades do time de suporte da arquitetura estejam alinhadas com a tecnologia e a ferramenta utilizadas;
- Parque instalado: Caso a corporação possua uma estratégia trocar o parque instalado que envolva um ativo de TI, os gastos com a tecnologia deverão ser menores para se adequar a esta estratégia;
- *Cases* de sucesso: Nem sempre os casos de sucesso são divulgados devido a cláusulas contratuais, porém a existência desses casos faz com que as corporações invistam em tecnologias que não sejam consideradas como meras apostas, mas sim como tecnologias consolidadas de mercado;
- Orçamento disponível para o projeto: De acordo com Erl (2005), em geral, projetos SOA requerem grande investimento por parte dos

patrocinadores do projeto. De acordo com o orçamento disponível, é possível optar por tecnologias abertas de desenvolvimento (por exemplo, Java e .NET Framework) ou tecnologias específicas de fornecedores (por exemplo, Oracle);

- Parceria com fornecedores: Parcerias com fornecedores de tecnologia para SOA podem ajudar na escolha de tecnologia, reduzindo os custos e possibilitando suporte especializado;
- Políticas de investimento: As políticas de investimento podem influenciar a escolha da tecnologia caso haja uma política de investimentos agressiva ou se é mais conservadora, ou seja, se segue um modelo de, quanto menor o investimento, melhor;
- APIs de serviços disponíveis: De acordo com os requisitos do projeto, algumas APIs de uma determinada tecnologia podem ajudar na integração de componentes já existentes na corporação. Este ponto está diretamente relacionado com as habilidades dos desenvolvedores, pois é a equipe de integração que será responsável por estabelecer a comunicação com APIs pré-estabelecidas com componentes legados da companhia.

3.4.5. Etapa 5: Identificar cenários de testes integrados

Esta etapa consiste em descrever os cenários de testes integrados que serão utilizados na fase de execução de testes da arquitetura orientada a serviços.

Como a fase de criação dos cenários de testes é a de análise, as condições de testes deverão ser atualizadas com mudanças solicitadas no decorrer do projeto.

É fato que a realização de testes de aplicações distribuídas é complexa, pois são necessárias diversas máquinas e interações entre diversas equipes e não é evidente prever todas as possíveis condições de falha. Para minimizar o impacto dessas dificuldades, é recomendado mapear, além dos testes dos requisitos funcionais, os testes dos requisitos não funcionais (por exemplo, desempenho), acessar as interfaces de comunicação de outras aplicações remotamente (por ser mais simples, mas dificultando os testes diretos de

interfaces locais) e em servidores locais, pois, segundo Erl (2005), permitem verificar também as classes com interfaces locais, por exemplo com testes usando serviços falsos, também conhecidos como *mock*, mas faz com que os custos de infraestrutura sejam maiores.

3.4.6. Etapa 6: Planejar e preparar o ambiente de testes

Uma vez definida a tecnologia a qual a arquitetura será desenvolvida, é necessário planejar e preparar o ambiente de testes. As propriedades técnicas da tecnologia selecionada são os pontos-chave de como o ambiente será adequado para execução dos testes após a arquitetura ser construída. Alguns exemplos do que deve ser levado em consideração são:

- Sistema operacional: A escolha do sistema operacional está diretamente relacionada com a tecnologia selecionada. Caso a tecnologia seja baseada na plataforma .NET, o sistema operacional deverá ser o Windows;
- Captura de evidência de testes: O ambiente deverá possuir recursos para captura de eventos dos testes;
- Quantidade de servidores utilizados nos testes: Para se aproximar de um cenário real, é necessário levantar quantos servidores são necessários para o teste. Isso auxilia na verificação dos testes de escalabilidade, balanceamento de carga (também conhecido como *load balance*) e desempenho do sistema;
- Procedimento de instalação: O ambiente deverá prover quais as formas possíveis para instalação do sistema. É recomendado que o responsável pela publicação do sistema tenha permissão de administração do ambiente de testes. Consequentemente, a aplicação possuirá os acessos necessários no ambiente de testes;
- Permissão de acesso remoto ao ambiente de testes: Os envolvidos no desenvolvimento da arquitetura deverão ter acesso remoto ao ambiente de testes para verificação das evidências geradas durante o teste;
- Espaço em disco rígido disponível: Por se tratar de uma arquitetura orientada a serviços, o espaço em disco rígido disponível não pode ser pequeno para a coleta de evidências. Arquivos de *logging* são usados

como evidências de teste e não poderá haver riscos na criação desses arquivos.

3.5. Especificação Orientada a Serviços

A fase de Especificação Orientada a Serviços é a fase nas quais documentações e desenhos necessários são criados para serem utilizados na construção da arquitetura orientada a serviços. Nela os arquitetos de SOA precisam envolver os arquitetos do projeto-alvo para certificar que o projeto apresentado pela equipe de SOA irá ser funcional para outros projetos específicos da corporação.

Durante a fase de especificação orientada a serviços, as etapas a serem cumpridas são identificar as tecnologias e plataformas de desenvolvimento, definir as especificações dos serviços e orquestrar os serviços com os processos de negócio, conforme a figura 3.4.

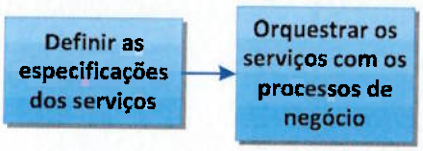

Figura 3.4: Etapas da fase de especificação orientada a serviços

3.5.1. Etapa 1: Definir as especificações dos serviços

Definir as especificações dos serviços envolve a criação de desenhos e documentações necessárias para o desenvolvimento e construção. As especificações são criadas a partir do conjunto de requisitos (funcionais e não funcionais) e dos casos de uso criados na fase de análise.

Para realizar esta etapa, é necessário focar em dois tipos de especificação: estáticas (ou estruturais) e comportamentais (ou dinâmicas).

As especificações estáticas são utilizadas para definir as entidades dos serviços a serem construídos. O UML (*Unified Modeling Language*) é uma linguagem que possibilita a modelagem das entidades através dos processos de negócio para especificar os serviços. O diagrama de classes é o recomendado para esse tipo de especificação, pois, além de representar a

abstração dos componentes a serem desenvolvidos, apresenta como é a relação entre eles. A partir dos fluxos dos casos de uso, é possível identificar quais passos representam interações entre aplicações distintas (o que identifica a comunicação entre serviços distintos). Desta forma, para cada comunicação, é necessário criar classes que representam o sistema ou serviço analisado e as interfaces com as quais serão integradas. Cada interface deverá ser implementada por uma classe concreta (ou seja, que tenha a finalidade de detalhar as implementações do serviço externo), mas este aspecto é opcional, uma vez que os serviços se comunicam somente a partir de suas interfaces para não detalhar regras de negócio internas ao sistema.

A figura 3.5 apresenta um exemplo de diagrama de classes com relação entre entidades, interfaces, operações e seus atributos:

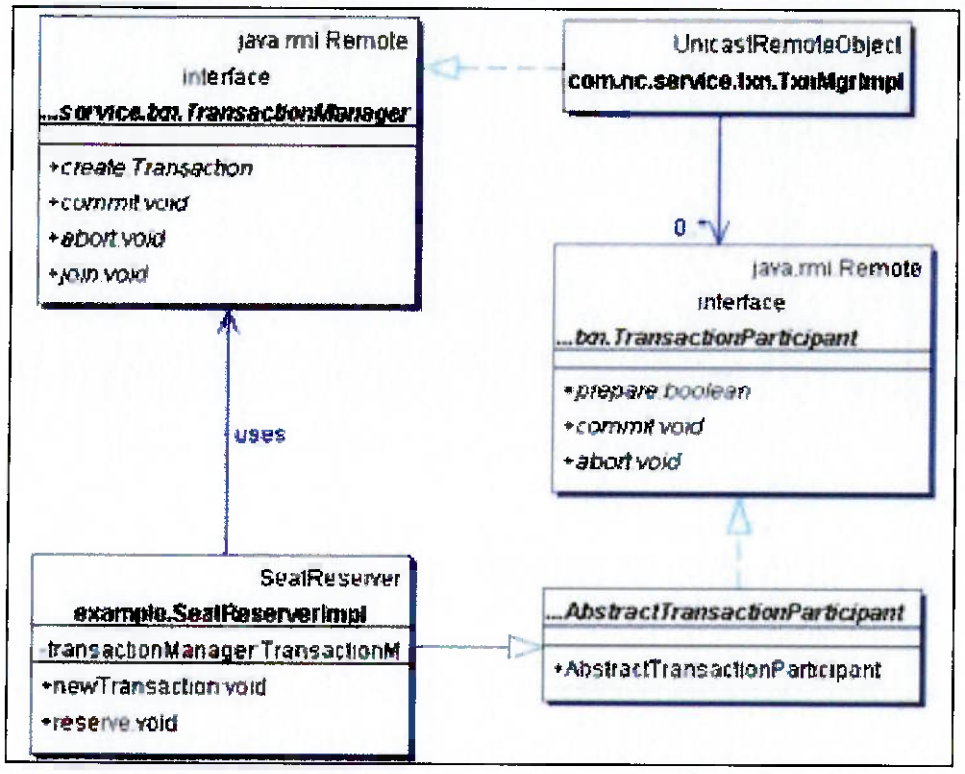

Figura 3.5: Exemplo de especificação de um serviço utilizando notação UML (retirado de *Taking Service-Oriented Architectures Mobile*. Disponível em <http://today.java.net/pub/a/today/2005/08/02/mobile2.html>)

Na modelagem de classes para o desenvolvimento da arquitetura orientada a serviços, é importante a utilização de interfaces que disponibilizam

mecanismos de acesso via serviços (no diagrama, estão representadas as interfaces *TransactionParticipant* e *TransactionManager*) e as imeplementações dos serviços (*example.SeatReserveImpl*). As classes são disponibilizadas a partir de suas interfaces, sem apresentar detalhes de suas implementações.

As especificações comportamentais são utilizadas para representar o sequenciamento dos serviços, podendo-se utilizar a notação da linguagem UML como o diagrama de sequência. O diagrama de sequência é um tipo de diagrama UML que representa a sequência de processos (mais especificamente, de mensagens passadas entre objetos) e também pode ser utilizada para representar o sequenciamento dos serviços. Para criar o diagrama, são necessários os casos de uso e os diagramas de classes criados previamente. Os casos de uso descrevem os fluxos que deverão ser atendidos e é no diagrama de sequência que o acionamento de serviços distintos deverá ser detalhado. O diagrama de classes também dá base ao diagrama de sequência, pois ele descreve os componentes que se comunicam entre si no nível de classe.

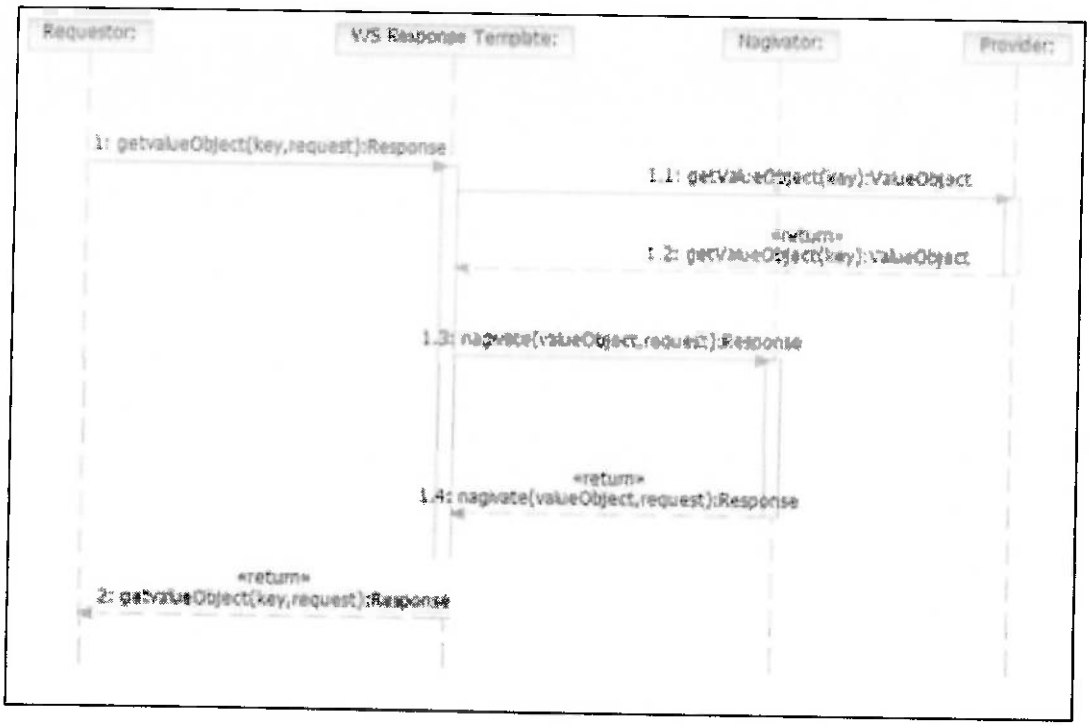

Figura 3.6: Exemplo de diagrama de sequências (retirado de *Web services response template pattern: a specification*. Disponível em <http://www.ibm.com/developerworks/webservices/library/ws-restemp/>)

É importante notar que, por definição de UML, o ator é um agente externo ao sistema desenvolvido. Dentro um contexto de SOA, as aplicações que acessarão os serviços são os próprios atores dos diagramas. É possível haver mais de um ator envolvido no diagrama de sequências, assim representando a comunicação inter-sistemas através de um barramento ou *middleware*. No diagrama de sequência, é importante haver uma interface que age entre o serviço e os agentes externos ao sistema (na figura 3.6, o objeto *WS Response Template*), a instância que representa a implementação do serviço (*Provider*) e o responsável por descobrir o sistema solicitante (*Navigator*), para que haja a comunicação síncrona entre os sistemas.

3.5.2. Etapa 2: Orquestrar os serviços com os processos de negócio

A orquestração dos serviços com os processos de negócio envolve a definição de quais serviços estão encapsulados (ou seja, somente com as suas interfaces de acesso expostas, e não suas lógicas de negócio) em quais

processos de negócio, a especificação de o que pode gerar erro na execução e a forma de como o sistema irá responder nas condições de finalização não esperadas. Para alcançar este objetivo, os serviços identificados na fase de análise devem ser inseridos nos fluxos dos processos com os quais estão associados. Isso garante que os serviços sejam acionados pelos processos, garantindo assim que sejam reutilizados pelos diversos processos. A linguagem BPEL (*Business Process Execution Language*) é utilizada nesta etapa do processo, pois ela permite trabalhar e modelar processos de negócio distribuídos por diversas aplicações, conceito este muito abordado na arquitetura orientada a serviços. Caso os processos já estejam mapeados em BPEL, é necessário alterá-los para que utilizem os serviços identificados na fase de análise para serem atualizados dentro do contexto de SOA.

No caso de modelagem de processos no desenvolvimento de uma arquitetura orientada a serviços, as especificações produzidas com BPEL devem focar na modelagem dos fluxos que os serviços distribuídos representam, observando as devidas orquestrações entre os serviços. Uma das vantagens do BPEL é trabalhar com linguagens extensíveis (por exemplo, XML), dando menos preocupações para os analistas que especificarem os serviços.

O diagrama 3.7 apresenta um exemplo de diagrama BPEL, representando um processo de negócio em execução:

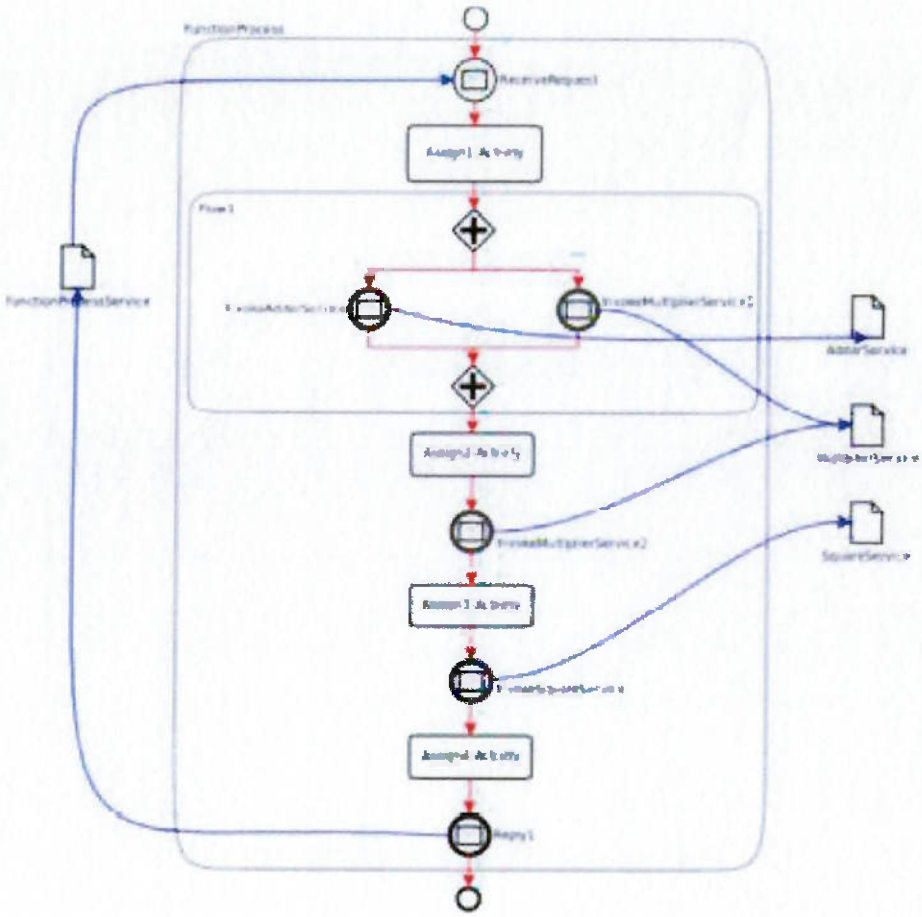

Figura 3.7: Exemplo de Diagrama BPEL, ilustrando coreografia de serviços (retirado de *Writing a simple WS-BPEL process for WSO2 BPS and Apache ODE*. Disponível em <http://wso2.org/library/articles/writing-simple-ws-bpel-process-wso2-bps-apache-ode>)

3.6. Desenvolvimento Orientado a Serviços

Esta fase envolve atividades de integração com sistemas existentes na corporação, tornando a escolha da plataforma de desenvolvimento efetiva e englobando também a realização de testes, sendo que os serviços, devido ao conceito de reuso, necessitam ser testados rigorosamente antes de sua implantação em ambiente de produção.

Na fase de desenvolvimento orientado a serviços, as etapas a serem seguidas são desenvolver os serviços especificados, integrar as aplicações através de um *Enterprise Service Bus* e realizar testes integrados, conforme a figura 3.8.

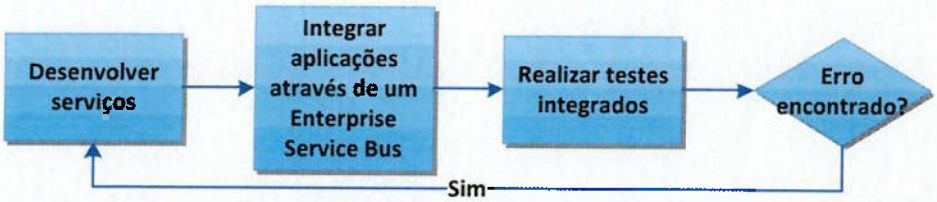

Figura 3.8: Etapas da fase de desenvolvimento orientado a serviços

3.6.1. Etapa 1: Desenvolver serviços

Nesta fase, o desenvolvedor usa a ferramenta mais indicada de desenvolvimento para construir os serviços a partir das especificações conforme a escolha de tecnologia utilizada feita na seção 3.4.4.. Algumas ferramentas de fornecedores de tecnologia automatizam a criação de *web services*, aplicando as especificações e coreografias UML na construção da arquitetura. Um exemplo disso é a criação de EJBs (*Enterprise Java Beans*) no caso de Java. Há ferramentas que registram os serviços desenvolvidos em registros UDDIs e criam a descrição do WSDL do serviço procurando e garantindo o reuso de serviços já publicados nos depositórios de serviços. Para as ferramentas que não registram os serviços automaticamente, é possível utilizar outra ferramenta responsável somente pela publicação dos serviços ou disponibilizá-los manualmente pelo sistema operacional do servidor que residirá o serviço construído.

Alguns pontos muito importantes deverão ser levados em consideração no desenvolvimento dos serviços:

- As entradas e saídas são baseadas em linguagens extensíveis (por exemplo, XML). Neste ponto, a utilização de APIs internas responsáveis pela extração e geração dos dados da mensagem é válida, desde que os dados não sejam propagados por outros componentes de *software* externos ao serviço;
- O serviço deve ser considerado como indivisível (também chamado de atômico). Um dos grandes desafios é garantir que códigos não sejam redundantes. Um serviço pode acessar outros serviços, desde que

acesso seja feito através da SOA. Em outras palavras, a reutilização dos serviços deverá ser explorada neste ponto;

- A estrutura da aplicação deve ser de tal forma que não impacte o funcionamento de outros serviços, garantindo a abstração da aplicação em camadas;
- Por se tratar de um serviço, é interessante criar também projetos de testes unitários responsáveis por testar o serviço sem utilizar o barramento, mas sim simulando possíveis situações que estiverem contidas nas especificações;
- Remover todas as referências a variáveis que possam armazenar estados da aplicação (um exemplo técnico seria uma variável estática dentro do padrão *singleton*).

3.6.2. Etapa 2: Integrar aplicações através de um *Enterprise Service Bus*

Nesta fase, é realizada a integração de aplicativos corporativos, utilizando a especificação da orquestração dos serviços desenvolvidos como parte dos processos de negócio já transformados e descritos na linguagem BPEL (seção 3.2.3.3.). Os serviços deverão ter sido desenvolvidos com base nas propriedades da arquitetura orientada a serviços para garantir a comunicação através do barramento. Neste ponto do processo, como o objetivo é a integração dos sistemas, as propriedades de *discoverability* e de acesso via XML são as mais importantes.

Como foi visto na seção 2.1.1., o ESB (*Enterprise Service Bus*, ou Barramento Corporativo de Serviços) é uma camada lógica de mensagens compartilhadas para interoperabilidade entre aplicações e serviços na empresa e entre empresas e faz transformação inteligente de mensagens e re-roteamento (como EAI). Para integrar as aplicações, o serviço (que deverá estar catalogado com um nome único dentro da arquitetura) deve ser publicado no ESB fazendo com que as informações trafegadas consigam ser roteadas para o serviço correspondente. Uma vez disponibilizado, o serviço poderá ser acessado e testado.

3.6.3. Etapa 3: Realizar testes integrados

Os testes integrados resolvem questões como integração entre sistemas distintos e comunicação entre as camadas do sistema implementado, que normalmente são feitos manualmente e de forma menos performática. É muito importante que esses testes sejam feitos desde o topo até a base desenvolvida por se tratar de um desenvolvimento de uma arquitetura corporativa, ou seja, desde as interfaces de integração da arquitetura orientada a serviços desenvolvida até a execução dos serviços e transmissão de dados para o solicitante. Para realizar os testes, é necessário levar em consideração os tipos de solicitantes de um serviço que podem acessá-lo, se as descrições dos serviços estão de acordo com a semântica dos mesmos e os tipos de condições de erro que possam fazer com que um serviço seja potencialmente exposto, impactando o funcionamento de outros serviços dependentes dele.

Neste ponto do processo, a infraestrutura necessária e o ambiente de testes para a realização dos testes integrados já devem estar disponibilizadas (por exemplo, acesso de clientes remotos e utilização de servidores próprios). As condições de teste mapeadas na fase de análise orientada a serviço deverão ser executadas.

Os testes executados podem apresentar sucesso (caso o resultado obtido seja o mesmo que o esperado), falha (caso o resultado obtido não seja o mesmo que o esperado) ou cancelado (caso o cenário de teste aborde um cenário que está fora do escopo do projeto). Para os cenários que apresentaram falhas, é necessário que haja outros ciclos de teste, cada um contemplando as manutenções exigidas no ciclo anterior. É importante que todos os cenários de teste sejam executados a cada ciclo de teste (processo chamado de "Teste de Regressão") para garantir que as manutenções não tenham impactado componentes ou funcionalidades que estavam construídos corretamente.

4. RESULTADOS

4.1. Resultados obtidos

O planejamento de uma SOA faz com que a empresa repense os objetivos corporativos de TI após a implantação da arquitetura orientada a serviços, focando na busca dos benefícios propostos e minimizando possíveis riscos existentes. A metodologia proposta surge para reduzir o retrabalho das atividades executadas durante projetos de implantação de arquiteturas orientadas a serviços com o auxílio de etapas definidas e acelerar o tempo de desenvolvimento de uma arquitetura orientada a serviços, agregando as tarefas importantes relacionadas aos processos de negócio e a tecnologia utilizada. Uma vez implantada, é possível estender a arquitetura para o desenvolvimento de novos serviços no atendimento de novas demandas e necessidades com mais agilidade por se tratar de uma tecnologia flexível e robusta.

A especificação da tecnologia e a escolha da plataforma de desenvolvimento são fatores que devem ser muito bem ponderados no início do projeto, levando também em conta as necessidades atuais e futuras do negócio. Também é importante saber que iniciar a modelagem dos serviços sem considerar os requisitos funcionais ou sem a definição dos requisitos não funcionais da arquitetura pode levar um projeto SOA ao fracasso. Outro ponto chave importante para o sucesso de um projeto SOA é o envolvimento de representantes de todas as áreas interessadas, sendo que elas devam ter objetivos em comum. Quando muitas áreas são envolvidas dentro de um mesmo projeto, pode ocorrer falha de comunicação entre os envolvidos, criando *gaps* que, possivelmente, podem levar o projeto ao fracasso.

A metodologia apresentada descreve claramente as fases e quais as tarefas a serem executadas para que os objetivos de cada tarefa sejam alcançados com êxito. O seguimento da metodologia faz com que o projeto seja mais organizado, possibilitando maior controle e maior foco nas áreas em que podem ser apresentados atrasos, falta de planejamento, qualidade abaixo das expectativas e retrabalho.

4.2. Resumo das Fases e Etapas

A figura 4.1 apresenta as fases e as etapas abordadas nesta monografia.

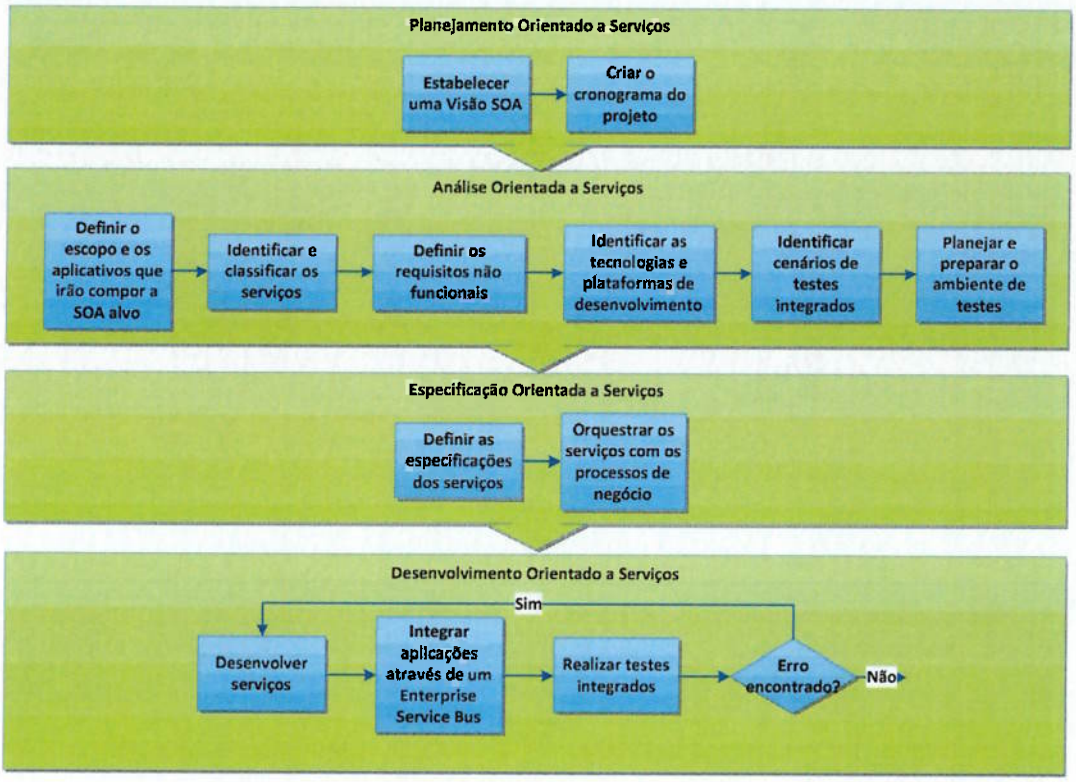

Figura 4.1: Fases e etapas da metodologia proposta

Para uma visão mais geral de cada uma das fases, a tabela 4.1 apresenta as etapas e um resumo de suas atividades:

Fase	Etapa	Tarefas	Artefatos de saída
Planejamento	Estabelecer Visão SOA	• Verificar quais os limites ou fronteiras que a SOA atingirá na organização, bem como quais benefícios (tangíveis e intangíveis) serão obtidos com sua implementação; • Listar Fatores Críticos de Sucesso para o projeto de desenvolvimento SOA; • Realizar a análise de impacto na área de TI da corporação.	Visão SOA e FCS

Fase	Etapa	Tarefas	Artefatos de saída
Planejamento	Criar o cronograma do projeto	• Planejar focado no investimento; • Criar o cronograma do projeto.	Cronograma do projeto.
Análise	Definir o escopo e os aplicativos que irão compor a SOA alvo	• Definir o escopo.	Escopo da análise
Análise	Identificar e classificar os serviços	• Decompor (detalhar) os processos de negócio; • Descartar passos dos processos não apropriados para encapsulação do serviço; • Criar os casos de uso para os candidatos a serviço; • Aplicar os princípios SOA aos serviços candidatos; • Classificar os serviços.	Classificação dos serviços
Análise	Definir os requisitos não funcionais	• Definir requisitos de desempenho e outros atributos de qualidade.	Requisitos não funcionais
Análise	Identificar cenários de testes integrados	• Levantar os cenários de testes integrados, mesmo que em versões não finais.	Cenários de testes integrados
Análise	Identificar as tecnologias e plataformas de desenvolvimento	• Identificar a tecnologia e plataformas de desenvolvimento para a fase de desenvolvimento da arquitetura.	Plataforma de desenvolvimento
Especificação	Definir as especificações dos serviços	• A modelagem dos processos de negócio em UML é utilizada para especificar os serviços, apresentando as entidades (com operações e atributos) e o sequenciamento.	Especificações (entidades e sequenciamento)

Fase	Etapa	Tarefas	Artefatos de saída
Especificação	Orquestrar os serviços com os processos de negócio	• A orquestração dos serviços com os processos de negócio se traduz em definir quais processos de negócio estão encapsulados em quais serviços e especificar o que pode dar errado na execução e como o sistema irá responder para condições de finalização não esperadas. Esta atividade pode ser realizada através da linguagem BPEL.	Orquestração dos serviços
Desenvolvimento	Desenvolver serviços	• Implementar as especificações e construir os serviços a partir da tecnologia adotada na fase de especificação.	Serviços desenvolvidos
Desenvolvimento	Integrar aplicações através de um Enterprise Service Bus	• Aplicação da especificação da coreografia dos serviços desenvolvidos como parte dos processos de negócio já transformados e descritos na linguagem BPEL.	Serviços integrados entre os diversos sistemas / plataformas
Desenvolvimento	Realizar testes integrados	• Além dos testes dos requisitos funcionais, especial atenção deve ser dada aos testes dos requisitos não funcionais, principalmente desempenho.	Documentação dos testes integrados

Tabela 4.1: Resumo das etapas e atividades

5. CONCLUSÕES

5.1. Contribuições do Estudo Realizado

Existe muito material que se encontra na internet sobre ferramentas e soluções para desenvolvimento e implementação de arquiteturas orientadas a serviços, mas não há um material centralizado com o "passo-a-passo" para as fases de planejamento, análise, especificação e desenvolvimento de uma SOA. A metodologia proposta nesta monografia vem preencher esta lacuna, focando mais nos aspectos essenciais que levarão ao sucesso na tentativa da implementação de uma arquitetura SOA do que uma visão de mercado para um determinado produto.

5.2. Trabalhos futuros

5.2.1. Extensão da metodologia

O trabalho de monografia abordou as fases de Análise, Especificação e Desenvolvimento de uma Arquitetura Orientada a Serviços, porém, de acordo com Erl (2005), existem outras fases do projeto que são abordadas após a implantação do sistema, tais como Implantação, Gerenciamento e Suporte, Otimização, Modelagem de Processos e Identificação de novos Requisitos. Essas fases poderão ser abordadas em trabalhos futuros.

5.2.2. SoaML

De acordo com o OMG (2001), SoaML (*Service Oriented Architecture Modeling Language*) é um projeto de especificação *open-source* baseado em UML (*Unified Modeling Language*) direcionado para modelagem e desenho de serviços dentro de uma arquitetura orientada a serviços.

O SoaML poderá ser abordado em futuros trabalhos envolvendo especificação e desenvolvimento de serviços baseados em SOA, pois atualmente esta especificação se encontra em versão Beta, ou seja, ainda se encontra em fase de desenvolvimento e testes.

5.3. Considerações Finais

Enquanto que a implementação de uma arquitetura orientada a serviços é um investimento de longo prazo a qual necessita de foco sustentado com relação à transformação da forma como a TI é entregue, ela deve corresponder imediatamente às iniciativas do negócio. Consequentemente, os benefícios de uma arquitetura orientada a serviços somente serão percebidos se houver um equilíbrio entre os objetivos de longo prazo e necessidades de curto prazo do negócio. Pode-se manter este equilíbrio por meio do estabelecimento de um conjunto de práticas organizacionais, financeiras, operacionais, de análise, especificação e de entrega desde o início de uma iniciativa SOA.

Por se tratar de uma tecnologia recente, as corporações devem se preparar de forma adequada para implementar SOA, já que é considerado um investimento a longo prazo. Uma vez implantada com sucesso, a arquitetura orientada a serviços abre novas oportunidades dentro das corporações, tendo em vista que a integração entre novos sistemas pode ser de mais ágil e eficaz, uma vez criado o canal comum de comunicação no nível da arquitetura.

Devido o aumento da complexidade dos sistemas desenvolvidos com portes cada vez maiores, é necessária uma metodologia para focar nas atividades mais importantes em cada etapa do processo como um todo. No caso do desenvolvimento de uma arquitetura orientada a serviços não é diferente. Por se tratar da integração de uma corporação como um todo, ligando todas as unidades corporativas e envolvendo *stakeholders* de diversas áreas, o grau de complexidade que a arquitetura irá demandar é muito grande por se tratar de setores e sistemas distintos que devem se comunicar entre si.

A reorganização dos processos de negócio é tão importante (ou até mais) do que a implementação da tecnologia, o que dá à fase de análise uma maior importância. Do lado da tecnologia, o ESB possui responsabilidade chave dentro da arquitetura que seja agregadora de serviços corporativos, os quais sejam efetivamente reutilizados.

Com o SOA, as empresas devem cada vez mais pensar que um sistema desenvolvido deve disponibilizar um serviço que possa ser consumido por

outros sistemas, sendo eles internos ou externos à corporação, antecipando possíveis integrações.

A necessidade de atender as demandas com cada vez mais agilidade é um dos grandes benefícios que a arquitetura orientada a serviços pode prover para as organizações, pois faz com que as unidades organizacionais se comuniquem como um todo, trazendo confiabilidade às informações trafegadas para um sistema interessado. Isso faz com que os requisitos não funcionais sejam tão importantes quanto os funcionais dentro do projeto SOA.

É importante esclarecer que a arquitetura orientada a serviços não é sempre a melhor solução. Caso uma corporação não possua sistemas críticos que precisem se comunicar, a SOA pode ser um investimento caro e desnecessário.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ARSANJANI, A. ***Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA***. IBM developerWorks, Novembro 2008.

BOTTO, R. **Arquitetura Corporativa de Tecnologia da Informação.** Rio de Janeiro: Brasport, 2008.

Business Performance Management Institute: ***SOA Watch: Is Your Enterprise Architecture Healthy?***. Disponível em: <http://www.bpminstitute.org/articles/article/article/is-your-enterprise-architecture-healthy/news-browse/13.html>. Acessado em 13 de janeiro de 2011.

ERL, T. ***Service-Oriented Architecture: Concepts, Technology, and Design***. Upper Saddle River: Prentice Hall PTR, 2005.

ERL, T. ***SOA Principles – An Introduction to the Service-Orientation Paradigm***. Disponível em: <http://www.soaprinciples.com/>. Acessado em: 26 de julho de 2010.

ERL, T. ***What is SOA – An Introduction to the Service-Orientation Paradigm***. Disponível em: <http://www.whatissoa.com/>. Acessado em: 10 de março de 2011.

GARTNER . ***Predicts 2010: SOA Advances.*** Disponível em: <http://www.gartner.com>. Acessado em: 10 de abril 2010.

GEAO: ***Global Enterprise Architecture Organization.*** Disponível em: <http://www.geao.org>. Acessado em: 21 de abril de 2010.

IBM: **Understanding SOA Security – Design and Implementation**. IBM Red books. 2007.

IBM: ***Service Oriented Architecture – SOA***. Disponível em <http://www-01.ibm.com/software/solutions/soa/>. Acessado em 4 de outubro de 2010.

iMasters: **A onda SOA**. Disponível em <http://imasters.com.br/artigo/8140/gerencia/a_onda_soa/>. Acessado em 4 de março de 2011.

Linha de Código: **Metodologia SOA: qual a necessidade?**. Disponível em <http://www.linhadecodigo.com.br/artigo/2528/Metodologia-SOA-qual-a-necessidade.aspx>. Acessado em 4 de março de 2011.

Microsoft: ***SOA & Business Process***. Disponível em <http://www.microsoft.com/soa/products/>. Acessado em 2 de novembro de 2010.

OASIS: **OASIS – *Advancing open standards for information society***. Disponível em <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.doc>. Acessado em 10 de novembro de 2010.

OMG: ***Class Diagrams***. Disponível em <http://www.omg.org/news/meetings/workshops/presentations/eai_2001/tutorial_monday/tockey_tutorial/4-Class_Models.pdf>. Acessado em 25 de janeiro de 2011.

OMG: ***Interaction Diagrams***. Disponível em <http://www.omg.org/news/meetings/workshops/presentations/eai_2001/tutorial_monday/tockey_tutorial/5-Interaction_Models.pdf>. Acessado em 25 de janeiro de 2011.

OMG: ***Use Case Diagrams***. Disponível em <http://www.omg.org/news/meetings/workshops/presentations/eai_2001/tutorial_monday/tockey_tutorial/3-Use_Cases.pdf>. Acessado em 25 de janeiro de 2011.

Oracle: ***Oracle SOA Suite***. Disponível em <http://www.oracle.com/us/technologies/soa/soa-suite-066466.html>. Acessado em 02 de novembro de 2010.

PMI: ***Project Management Institute***. Disponível em <http://www.pmi.org/>. Acessado em 13 de janeiro de 2011.

ROCKART, John F. ***Chief Executives Define Their Own Data Needs***, *Harvard Business Review*. 1979.

RUP: ***Best practices for software development teams***. Disponível em <http://www.ibm.com/developerworks/rational/library/253.html>. Acessado em 2 de janeiro de 2011.

RUP: **IBM *Rational Unified Process***. Disponível em: <http://www-01.ibm.com/software/rational/>. Acessado em 27 de julho de 2010.

RUP: ***TMap and the Rational Unified Process***. Disponível em: <http://www.ibm.com/developerworks/rational/library/dec04/koomen/>. Acessado em 27 de janeiro de 2011.

SCHULTE, W. R.; NATIS, Y. V. N.. ***Service-Oriented Architecture***. Gartner, Abril de 1996.

The SOA Magazine: ***Modern SOA Methodology and SOA Adoption Using Agile Practices***. Disponível em <http://www.soamag.com/I42/0810-2.php>. Acessado em 4 de março de 2011.

SoaML: **Object Management Group (OMG)**. Disponível em <http://www.omg.org/spec/SoaML>. Acessado em 21 de novembro de 2010.

SOARES, A.. **Introdução, Identificação e Análise em Engenharia de Requisitos**. 2005.

Sun: ***Assessing Your SOA Readiness***. Disponível em <http://www.sun.com/software/whitepapers/webservices/soa_ready.pdf>. Acessado em 13 de dezembro de 2010.

W3C: ***World Wide Web Consortium*** **(W3C)**. Disponível em <http://www.w3.org/TR/ws-arch/>. Acessado em 9 de novembro de 2010.

WEILL, P. e ROSS, J. W.. **Governança de TI: Tecnologia da Informação**. M. Books, 2008.

WSO2: ***Oxygen Tank – The developer portal for SOA***. Disponível em <http://wso2.org/>. Acessado em 25 de outubro de 2010.

ZACHMAN FRAMEWORK ASSOCIATES: ***Practitioners of Frameworks Elaborations***. Disponível em <http://www.zachmanframeworkassociates.com>. Acessado em 2 de outubro de 2010.

7. ANEXOS

Anexo A – Guia de soluções dos principais fornecedores SOA

Os principais fornecedores de software de negócios do mundo têm estratégia SOA. Os fornecedores e suas respectivas soluções SOA que mais se destacam são os descritos na tabela 7.1:

Fornecedor	Soluções SOA
IBM	A plataforma com as soluções e softwares para desenvolvimento e implementação da arquitetura SOA é basicamente composta das ferramentas WebSphere/J2EE e Rational e uma das mais completas do mercado. De acordo com a IBM (2010), a plataforma de desenvolvimento, implantação e governança SOA da IBM utiliza as seguintes soluções e ferramentas: • **IBM *WebSphere Business Integration Modeler***: Permite a modelagem, análise e simulação de processos de negócio, podendo transformar estes modelos em linguagem UML e BPEL; • **IBM *Rational Software Architect***: Ferramenta integrada para especificação e construção de aplicações orientadas a serviços; • **IBM *WebSphere Integration Developer***: Utilizada para compor aplicações pelo refinamento dos modelos de processo de negócio; • **IBM *Rational Application Developer***: ambiente integrado de desenvolvimento para automatizar tarefas construídas no padrão *Web Services*; • **IBM *WebSphere Process Server***: Integra as diversas infraestruturas da corporação, além de aproveitar os um grande conjunto de aplicações e adaptadores existentes no mercado; • **IBM *WebSphere Message Broker***: Disponibiliza um avançado ESB fornecendo conectividade e transformação de dados para aplicações baseadas em padrões ou não e serviços para SOA; • **IBM *WebSphere Partner Gateway***: Integra processos externos e comunidades parceiras com os processos e infraestruturas internos, permite um robusto e flexível ambiente B2B; • **IBM *WebSphere Portal***: Contém um leque grande de tecnologias de portal que ajudam a desenvolver e manter portais B2C (*business to consumer*), B2B (*business to business*) e B2E (*business to employee*).
Microsoft	De acordo com a Microsoft, a Plataforma de Aplicação da

Fornecedor	Soluções SOA
	Microsoft oferece um portfólio abrangente de recursos de SOA e Processos de Negócios pré-integrados - garantindo um rápido ROI - e interoperáveis por design para trabalhar com seus investimentos existentes. Com estas características nativas, a Plataforma de Aplicação Microsoft permite que você melhore a cada dia o uso da arquitetura orientada a serviços, de acordo com uma estratégia de investimentos adequada para a sua organização e seus recursos atuais de TI. Ainda de acordo com a Microsoft, os produtos para desenvolvimento SOA são: • **.NET *Framework***: O .NET *Framework* é o modelo de programação de código gerenciado para o Microsoft Windows. Desenvolvedores usam o .NET *Framework* para construir serviços e aplicativos para clientes Windows, Windows Servers ou dispositivos controlados pelo Windows; ou para construir interconexões entre componentes distintos em um arquitetura orientada a serviços; • ***BizTalk Server***: O *BizTalk Server* R2 oferece conectividade, serviços de mensagens e de processos de negócios à infraestrutura orientada a serviços da organização. Através de suporte amplo para conectividade com plataformas, aplicativos, dispositivos e pessoas, o BizTalk Server preserva investimentos existentes e minimiza o custo e riscos de integrar novos itens de tecnologia a suas iniciativas de SOA; • ***Visual Studio Team System***: O Microsoft Visual Studio Team System é uma solução de ALM (*Application Lifecycle Management* – Gerenciamento do Ciclo de Vida de Aplicativos) integrada composta de ferramentas, processos e orientação para ajudar todos na equipe a melhorar suas habilidades e trabalhar mais eficientemente juntos; • ***System Center***: O Microsoft *System Center* desempenha um papel central na visão da Microsoft para ajudar organizações de TI a se beneficiar de sistemas auto-gerenciados e dinâmicos. As soluções System Center são ajustadas para simplificar o gerenciamento dos sistemas e aplicativos que sua empresa já tem implementado, inclusive o Microsoft *SQL Server*, Microsoft *Exchange Server*, Microsoft *BizTalk Server*, *Internet Information Services* e o Microsoft .NET *Framework*; • ***SharePoint***: O Microsoft *Office SharePoint Server* é uma suíte integrada de recursos de servidor que pode ajudar a melhorar a eficiência organizacional ao fornecer gerenciamento de conteúdo e busca corporativa abrangentes, acelerar processos de negócios compartilhados e facilitar o compartilhamento de informações através de limites para melhor percepção comercial; • ***Oslo***: Com o *Oslo*, a Microsoft simplifica significativamente o esforço necessário para projetar, construir, implantar e gerenciar aplicativos distribuídos. A tecnologia para fornecer esses recursos será distribuída através do *BizTalk Server*, *System Center*, *Visual Studio*, *BizTalk Services* e do .NET

Fornecedor	Soluções SOA
	Framework; • ***Dynamics***: O Microsoft *Dynamics* é uma linha de soluções de gerenciamento comercial integradas e adaptáveis que automatizam e dinamizam processos financeiros, de relacionamento com clientes e de cadeia de fornecedores de uma forma que ajuda você a produzir sucesso comercial.
Oracle	Com as aquisições das empresas BEA Systems e Sun Microsystems, a Oracle se mostra como uma grande fornecedora de produtos para desenvolvimento SOA. A Oracle possui duas plataformas de desenvolvimento SOA: • ***Oracle WebLogic Platform***: Oracle Weblogic consiste na plataforma Java EE que inclui os seguintes componentes: o ***Oracle WebLogic Server Web Services***: Servidor para hospedar os *web services*; o ***Oracle Coherence***: Responsável pelo *caching* de dados utilizados por servidores múltiplos, minimizando problemas de desempenho da arquitetura; o ***JMS Messaging Standard***: Padrão de mensageria Java; o ***Enterprise Grid Messaging***: Algoritmo de mensageria para servidores múltiplos; o ***JRockit***: Responsável pelo monitoramento de desempenho da arquitetura, capaz de identificar "gargalos"; o ***Tuxedo***: Plataforma usada para distribuição transacional em um ambiente distribuído; o ***Oracle TopLink***: Provê um algoritmo de persistência de objetos Java em XML e banco de dados relacionais. • ***SOA Suite***: Suite da Oracle para desenvolvimento SOA que contém os seguintes produtos: o ***Oracle BPEL Process Manager***: Orquestra aplicações e *Web Services* em processos de negócio; o ***Oracle Business-to-Business Integration***: Responsável pela integração de serviços corporativos baseados em tecnologias diferentes; o ***Oracle Business Rules***: Responsável por fazer as aplicações e processos mais flexíveis por fazer com que analistas configam definir e modificar lógica de negócio se, programação definindo e alterando regras de negócio via ferramenta; o ***Oracle Business Activity Monitoring (BAM)***:Solução capaz de construir relatórios (também conhecidos como *dashboards*) com dados em tempo real para monitoramento de serviços e processos de negócio; o ***Oracle Enterprise Service Bus***: Plataforma de integração SOA baseada em padrões para atender um alto volume de demanda de dados. É responsável por conectar e gerenciar interações entre serviços heterogêneos, aplicações legadas a até mesmo outros ESBs; o ***Complex Event Processing (CEP)***: Provê ambiente de

Fornecedor	Soluções SOA
	desenvolvimento de aplicações a serem integradas à arquitetura com a ajuda da ferramenta *Oracle JDeveloper*; ○ ***Oracle JDeveloper***: IDE para desenvolvimento SOA, Java e REA (*Rich Enterprise Application*).

Tabela 7.1: Principais fornecedores e suas respectivas soluções SOA

Anexo B – Breve comparativo entre as plataformas abertas .NET e Java

De acordo com o *Gartner Institute*, as principais, mais utilizadas e conhecidas plataformas de desenvolvimento existentes atualmente no mercado são Java da Sun/Oracle e .NET da Microsoft. As plataformas .NET e J2EE possuem o foco para o mercado de aplicações corporativas e para a tecnologia *web services*.

Realizando uma comparação entre essas duas tecnologias, é necessário levar em as seguintes informações em consideração:

- Buoncompagno[13] afirma que os custos de ferramentas Microsoft são mais altos que os suportem especificamente Java, o que pode ser amenizado com parcerias, dependendo da corporação;
- Enquanto o J2EE é centrado na linguagem Java e é multiplataforma, o .NET *Framework* possui diversas linguagens (VB.NET , C# , J# ,COBOL, entre outras) e é centrado na plataforma Windows.

O .NET *Framework* é um produto da estratégia Microsoft baseado na evolução da IDE Visual Studio e é um esforço quase que isolado da Microsoft para atingir o mercado de *web services*. Por trás do padrão J2EE (que é basicamente uma série de padrões), a Sun procurou reunir as maiores empresas de software a fim de adaptar a interface J2EE, como BEA Systems, IBM e Oracle.

O Java está disponível para qualquer plataforma (por exemplo: Win32, Win64, Unix e Mainframe, desde que haja uma JVM) o que facilita muito a portabilidade das aplicações J2EE, diferentemente da plataforma .NET

[13] Alfredo Buoncompagno é coordenador de equipe e desenvolvimento sênior (*Sun Certified Java Programmer* e *Microsoft Certified Professional*)

Framework, que requer um esforço da Microsoft em oferecer meios para que as aplicações .NET rodem em outras plataformas que não o Windows.