Marcos Roberto Eugênio

APLICAÇÃO DOS CONCEITOS DA ARQUITETURA ORIENTADA A SERVIÇOS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo

Área de concentração
Engenharia de Software

Orientadora:
Profa. Dra. Jussara Pimenta Matos

São Paulo
2007

FICHA CATALOGRÁFICA

Eugênio, Marcos Roberto
Aplicação dos Conceitos da Arquitetura Orientada a Serviços no Processo de Desenvolvimento de Software /
Marcos Roberto Eugênio. – São Paulo, 2007.
79p
Monografia (MBA- Engenharia de Software) Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, 2007.

PECE – Programa de Educação Continuada em Engenharia.
1. Arquitetura Orientada a Serviços. 2. Processo de Desenvolvimento de Software. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais

DEDICATÓRIA

Aos pais, Maria e Dourival
pelo apoio.
À esposa Giselma e filha Bruna pelo
carinho incondicional.

AGRADECIMENTOS

À Profa. Dra. Jussara Pimenta Matos, pelo apoio, orientação, dedicação e paciência que sem os quais não seria possível a execução deste trabalho.

À Profa. Dra. Selma Shin Shimizu Melnikoff, pela oportunidade de realizar este curso.

Aos professores do curso de MBA em Tecnologia de Software, pelo conhecimento transmitido.

Aos amigos Estiver Herrera Hipólito, Hugo de Lima Rego e Amilton Santana da Silva, por colaborarem com o meu aprendizado ao trabalharmos em grupo durante este curso.

Aos amigos de classe, pela oportunidade da troca de experiências.

RESUMO

O objetivo deste trabalho é avaliar as características da arquitetura orientada a serviços (*Service Oriented Architecture*-SOA) que influenciam o processo de desenvolvimento de software, mais especificamente nas disciplinas de Análise e Projeto, apresentando uma proposta de alteração de um processo de software existente. A Arquitetura Orientada a Serviços (Service Oriented Architecture – SOA) pode ser utilizada de forma a integrar diferentes tipos de sistemas de software, disponibilizando suas funções através de serviços. Para este tipo de desenvolvimento é importante considerar como a sua utilização afeta o processo de desenvolvimento de software.

ABSTRACT

The objective of this work is to evaluate characteristics of the services oriented architecture (SOA) that influence the process of software development, especially in the disciplines of Analysis and Design and an amendment of a existing process of software. The Service-Oriented Architecture (SOA) can be used to integrate different types of software systems, providing its functions through services. For this type of development it is important to consider how the use of this architecture affects the developing software process.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

CORBA - Common Object Request Broker
DBA – Data Base Administrator
DCOM - Distributed Component Object Model
DSOM - Distributed System Object Model
HTTP – Hyper Text Transfer Protocol
OO – Object Oriented
SOA – Service Oriented Architecture
SOAP – Simple Object Access Protocol
SOX - Sarbanes Oxley
UDDI - Universal Description Discovery and Integration
UP – Unified Process
WSDL - Web Services Description Language
XML - eXtensible Markup Language
MVC - Model View Controller
z/OS - z Series Operating System IBM
IBM - International Business Machine
OMG - Object Management Group
COBOL - Common Business-Oriented Language
MER - Modelo Entidade Relacionamento

SUMÁRIO

1. INTRODUÇÃO

1.1 OBJETIVO

O objetivo deste trabalho é avaliar características da arquitetura orientada a serviços (*Service Oriented Architecture*- SOA) que influenciam na tomada de decisão em relação a uma determinada solução em um sistema de software, durante o processo de desenvolvimento de software, mais especificamente nas disciplinas de Análise e Projeto.

Para abordar a arquitetura orientada a serviços (SOA), primeiramente, são apresentados conceitos relevantes no âmbito deste trabalho (ERL, 2005; KARACSONY; TERRY, 2007). Em uma segunda instância são apresentadas as tecnologias relacionadas aos conceitos (STOJANOVIC; DAHANAYAKE, 2005) e uma implementação que utiliza uma destas tecnologias (BARRY, 2003).

Em seguida, são apresentados os conceitos de Processo de Desenvolvimento de Software (PRESSMAN, 2002), o processo unificado (UP) (JACOBSON: BOOCH; RUMBAUGH, 1999), e o processo de desenvolvimento da *Empresa Exemplo* que será utilizado como estudo de caso. É discutida uma proposta de um processo de desenvolvimento de software já adaptado às disciplinas relacionadas ao SOA (GRÜNBAUER et al., 2004) e ao final é feita uma avaliação das mesmas.

A partir dessa avaliação, tanto das características de SOA, quanto das tecnologias aplicáveis, são propostas alterações no processo de desenvolvimento da *Empresa Exemplo*, para incluir as disciplinas referente à adoção da Arquitetura Orientada a Serviços nos futuros projetos.

1.2 JUSTIFICATIVA

A necessidade de utilização de informações originadas de sistemas de software heterogêneos é motivo de estudo por diferentes pesquisadores, isto levou ao surgimento do SOA que permite este tipo de integração com um grau satisfatório de

facilidade em sua aplicação, de uma forma geral, no desenvolvimento de software (BROWN; JOHNSTON; KELLY, 2002; DUAN et al., 2005).

Atualmente as empresas estão sob constante pressão para responder de forma rápida as mudanças de mercado. Para obter sucesso, estas devem mudar a maneira como se comportam, de forma a atender as demandas de mercado. Segundo Cherbakov et al., (2005), a componentização e a orientação a serviços são os elementos necessários que possibilitam a agregação de valor entre a rede de parceiros, tais como clientes e fornecedores, suportados por sistemas de informação e tecnologias.

As atividades para definição da arquitetura de um sistema de software são pouco discutidas no processo de desenvolvimento de software. Com a crescente complexidade no desenvolvimento e considerando a necessidade cada vez mais premente de estabelecer diversas parcerias, é preciso avaliar a maneira mais adequada de atender os diferentes interesses de uma empresa.

1.3 MOTIVAÇÃO

Para atender as mudanças exigidas pelo mercado, a Tecnologia da Informação deve estar alinhada com a visão estratégica da empresa, e a constante demanda de novos negócios e processos. Portanto, utilizar-se da Arquitetura Orientada a Serviços é um passo critico, é uma das maneiras de prover uma integração fim a fim na organização e serviços (BIEBERSTEIN; BOSE; WALKER; LYNCH, 2005).

A adoção de uma abordagem sistemática para processo de desenvolvimento de software, incluindo a adoção do SOA vai de encontro às necessidades do negócio da organização. Essa abordagem auxilia na melhoria da produtividade e da qualidade no desenvolvimento de software, além de permitir a redução dos custos de desenvolvimento de projetos e tempo de resposta ao mercado.

1.4 ESTRUTURA DO TRABALHO

Primeiramente foi realizada uma pesquisa referente ao conceito de Arquitetura Orientada a Serviços e das tecnologias que a implementam. Em seguida, é apresentado o Processo Unificado (UP), este processo foi selecionado devido à sua característica de permitir diferentes configurações, e o processo da *Empresa Exemplo*, em ambos, as disciplinas de Análise e de Projeto são investigadas. Os documentos utilizados para apresentar o desenvolvimento da metodologia da *Empresa Exemplo* não estão referenciados, pois são de propriedade da organização.

Para dar prosseguimento, seguiu-se a pesquisa bibliográfica para identificar artigos e bibliografias que discutiam a respeito de um processo de desenvolvimento de software voltado para SOA (ZIMMERMANN; KROGDAHL; GEE, 2004) (PAPAZOGLOU; YANG, 2002). Após a análise e seleção, um deles foi avaliado detalhadamente (GRÜENBAER et al., 2004). Para concluir, é apresentado um estudo de caso do processo de desenvolvimento da *Empresa Exemplo* utilizando SOA. A *Empresa Exemplo* é a companhia onde o autor deste trabalho atua.

O Capítulo 1 apresenta a Introdução, Objetivo, Justificativa, Motivação e a Estrutura do Trabalho.

O Capítulo 2 apresenta os conceitos sobre Arquitetura Orientada a Serviços (SOA), Tecnologias Orientada a Serviços e características de implementação de uma destas tecnologias.

O Capítulo 3 apresenta o Ciclo de vida do Processo Unificado e da *Empresa Exemplo* destacando-se as disciplinas de Análise e Projeto.

O Capítulo 4 apresenta uma proposta de um processo de desenvolvimento de software com SOA e uma avaliação deste processo.

O Capítulo 5 apresenta o estudo de caso da *Empresa Exemplo* utilizando a Arquitetura Orientada a Serviços em seu processo de desenvolvimento de software destacando-se as disciplinas de Análise e Projeto.

O Capítulo 6 apresenta a conclusão deste trabalho, bem como, sugestões para trabalhos futuros.

2. ARQUITETURA ORIENTADA A SERVIÇOS

2.1 CONSIDERAÇÕES INICIAIS

O objetivo deste capítulo é apresentar os fundamentos do SOA (ERL, 2005), utilizado como base para o desenvolvimento deste trabalho e os componentes deste tipo de arquitetura (KARACSONY; TERRY, 2007).

Para o desenvolvimento de uma aplicação que tem como base o SOA, devem ser consideradas tecnologias orientadas a serviço, para tanto essas também são apresentadas neste capítulo (STOJANOVIC; DAHANAYAKE, 2005). Além disso, é incluída uma implementação a partir da tecnologia de Web Services (BARRY, 2003; KOROTKIY; TOP, 2006), utilizada para exemplificar como aplicações baseadas em SOA podem ser implementadas, o que pode ter grande influencia no processo de desenvolvimento de software.

2.2 FUNDAMENTOS DA ARQUITETURA ORIENTADA A SERVIÇOS

2.2.1 A EVOLUÇÃO DA ARQUITETURA

A arquitetura de software é um dos elementos críticos no desenvolvimento de um sistema e na evolução de um software. Existem várias definições de arquitetura (JACOBSON; BOOCH; RUMBAUGH, 1999; HOFMEITER; NORD; SONI, 2000), mas pode-se considerar que a Arquitetura de Software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, constituído de elementos de software, das propriedades externas e visíveis desses elementos e de seus inter-relacionamentos (BASS; CLEMENTS; KAZMAN, 2003)

A evolução da arquitetura é a base para a discussão de SOA porque é preciso compreender onde se está e o que se pretende alcançar com a arquitetura. Karacsony e Terry (2007) em seu trabalho apresentam a evolução da arquitetura de software conforme as camadas mostradas na figura 2.1. As definições a seguir têm

como base este trabalho, onde serão apresentados os conceitos da arquitetura monolítica até chegar ao SOA para ter-se uma visão da evolução da arquitetura.

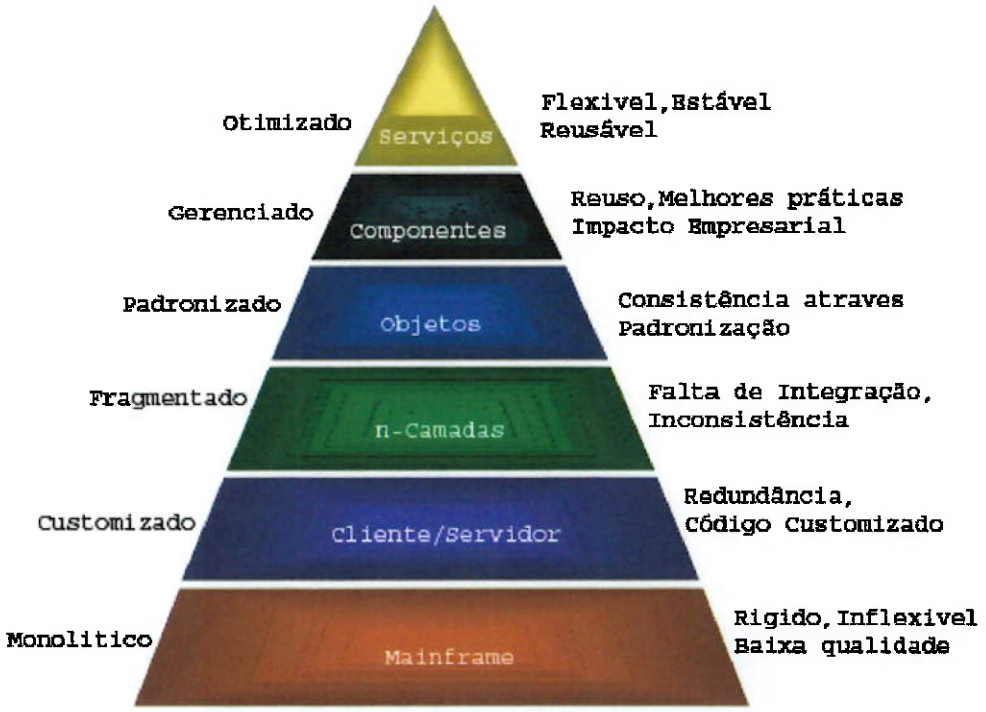

Figura 2.1 – Evolução da Arquitetura
(KARACSONY; TERRY, 2007)

Os principais conceitos apresentados nas camadas da figura 2.1 são descritos a seguir.

a) Monolítico

Em um ambiente Monolítico, os dados e os processos se misturam, os sistemas não se comunicam, não existe reuso, a localização física de arquivos, bancos de dados e programas não são alterados, o que torna critica qualquer mudança, devido à duplicidade de informação. O *Bug do milênio* (KARACSONY, TERRY, 2007) é um exemplo clássico disto, onde sistema monolítico não tem agilidade para atender a pequenas mudanças.

b) Customizado

Neste tipo de arquitetura, geralmente, a aplicação é processada no cliente e os dados no servidor, mudanças na base de dados têm grande impacto, pois os programas o acessam diretamente. Estas limitações muitas vezes induzem os

departamentos da empresa a criar cópias locais dos dados. Normalmente, as aplicações desenvolvidas são construídas em hardware especifico e software incompatível, criando assim um ambiente distribuído de sistemas, onde a interface e processamento ficam no cliente e os dados no servidor.

c) Fragmentado, Padronizado e Gerenciado

Fragmentado, Padronizado e Gerenciado são atributos da computação distribuída (KARACSONY, TERRY, 2007) o sistema é dividido logicamente em partes, e são distribuídas. Uma divisão típica seria os dados residirem em um servidor de dados, a aplicação ou a lógica de negócio do sistema em um outro servidor, a apresentação ou a interface pode estar em um servidor Web que será acessado nas estações de trabalho dos usuários.

Neste tipo de ambiente, dados, aplicação, e apresentação são separados em suas próprias camadas a exemplo disto pode-se citar o *Model-View-Controller* (MVC), conforme LARMAN (2004). Isto permite um maior gerenciamento e escalabilidade de recursos. Porém, com a computação distribuída surgiram os problemas de integração, padrões inconsistentes e diversas tecnologias. As dificuldades de integração que muitas empresas estão sofrendo são em grande parte resultado da mudança de arquitetura cliente-servidor para computação distribuída.

d) Otimizado

Otimizado ou Serviços difere de Objetos e Componentes (STOJANOVIC; DAHANAYAKE, 2005), pois segue padrões abertos. Ela provê transparência de tecnologia com o conceito de serviços produtor e consumidor. Como a tecnologia de interface as mensagens são independentes, ou seja, as mudanças podem ocorrer na implementação do serviço, porém o que deve ser considerado para o consumidor do serviço é que o formato da mensagem da interface não mude, esta separação proporciona flexibilidade e baixo custo de manutenção. As características chave do SOA são baixo acoplamento, separação de conhecimento, reuso, e padrões abertos (OMG, 2007).

Em ZDNet (2004) é afirmado que:

> *"Gartner prevê que em 2007 a maioria das empresas utilizará SOA como um guia na criação de aplicações missão critica e processos".*

A evolução da arquitetura citada no artigo de (KARACSONY, TERRY, 2007) vem de encontro a previsão do Gartner sobre a tendência de utilização do SOA.

2.2.2 PRINCÍPIOS DE PROJETO PARA SOA

De acordo com Karacsony e Terry (2007), na implementação do SOA vários princípios devem fazer parte do projeto para garantir uma arquitetura flexível, estável e ágil. Os principais princípios são: separação de atividades (serviços), o conceito de baixo acoplamento de sistemas, a utilização de padrões abertos e projetar para o re-uso. Os conceitos aplicáveis ao reuso são apresentados a seguir.

a) SEPARAÇÃO DE ATIVIDADES (SERVIÇOS)

Esta separação de atividades deve acontecer antes do sistema, ou seja, no processo de negócio, cada atividade deve ser identificada, assim como seu responsável, para que uma atividade possa ser executada isoladamente, evitando redundância de informações e trabalho, obtendo-se alguns benefícios. Com a separação de atividades fica clara a tarefa de cada funcionário, pode-se aperfeiçoar o processo de determinada área sem afetar a outra, identificar as falhas e corrigi-las.

b) BAIXO ACOPLAMENTO

A dificuldade de comunicação que existe entre sistemas que operam em uma variedade de plataformas com diferentes linguagens de programação. Portanto, para que os sistemas possam se comunicar, a área de TI das organizações investe muito dinheiro e tempo desenvolvendo interfaces.

Os padrões surgiram para tentar reduzir o acoplamento, ou seja, reduzir a dependência de plataforma e facilitar a interface, tais como, CORBA (CORBA, 1995), DCOM (MICROSOFT, 2007. Mas ainda assim, a dependência de plataforma

continuou, pois estes padrões não são totalmente abertos, ou seja, foram construídos direcionados para uma determinada plataforma ou tecnologia como, por exemplo, COBOL do z/OS.

O baixo acoplamento de SOA é o resultado de padrões não específicos de determinado fabricante da indústria do software; melhor do que desenvolver padrões, os grandes fabricantes de software concordaram em seguir os padrões abertos determinados pelos consórcios como, por exemplo, o *SOA consortium* (CONSORTIUM, 2007), independente de linguagem e plataforma.

c) PADRÕES ABERTOS

Padrões abertos (OMG, 2007) são aspectos importantes da arquitetura, porque permite flexibilidade no projeto de sistemas. Ao invés de se dedicar somente a um fabricante, hardware, aplicação, ou linguagem. Para as empresas é melhor a adoção de padrões abertos pela flexibilidade de troca de fornecedores de tecnologia.

d) PROJETAR PARA REUSO

Considerando os conceitos apresentados anteriormente, o reuso de componentes poderá obter sucesso. A ausência destes conceitos na construção de um sistema pode ser apontada como uma das falhas nos projetos de TI que visam o reuso. Os sistemas baseados na arquitetura SOA são complexos, requerem reuso, controle centralizado e o tratamento de erros.

2.2.3 CAMADAS DA ARQUITETURA ORIENTADA A SERVIÇOS

Para a implementação do SOA é necessário um ambiente que forneça suporte estes princípios. Podemos citar, como exemplo, a tecnologia Web Services que implementa os princípios de SOA (STOJANOVIC; DAHANAYAKE, 2005). Porém, é necessário que se coordene e propague estes serviços, isto pode ser realizado pela abstração das camadas de serviços (ERL, 2005). As definições a seguir têm como base o trabalho de ERL (2005), que propõe uma separação em diferentes camadas os tipos de serviços.

A camada de serviços de interface esta localizada entre as camadas de processo de negócio e da aplicação, e é onde residem os serviços de conectividade onde as características do SOA são mais relevantes. Durante a disciplina de análise da orientação a serviços deve-se definir:

- A lógica que deve representar os serviços;
- O relacionamento dos serviços e a lógica de aplicações existentes;
- A melhor maneira dos serviços representarem a lógica do processo de negócio;
- A melhor maneira de construir e posicionar os serviços para promover agilidade.

Os serviços são modelados de acordo com resposta a estímulos externos ao negócio.

Cada camada pode abstrair um aspecto específico da solução, dirigindo-se a um dos assuntos identificados. Ao invés de criar os serviços que acomodam o negócio, a aplicação, e as considerações da agilidade de uma vez, pode-se fazer isto separadamente. As três camadas identificadas no SOA são:

- Camada de serviços da aplicação;
- Camada de serviços do negócio;
- Camada de serviços de orquestração.

Como mostra a figura 2.3 através da abstração de camadas distintas, as principais características do SOA podem ser realizadas isoladamente, aumentando a agilidade.

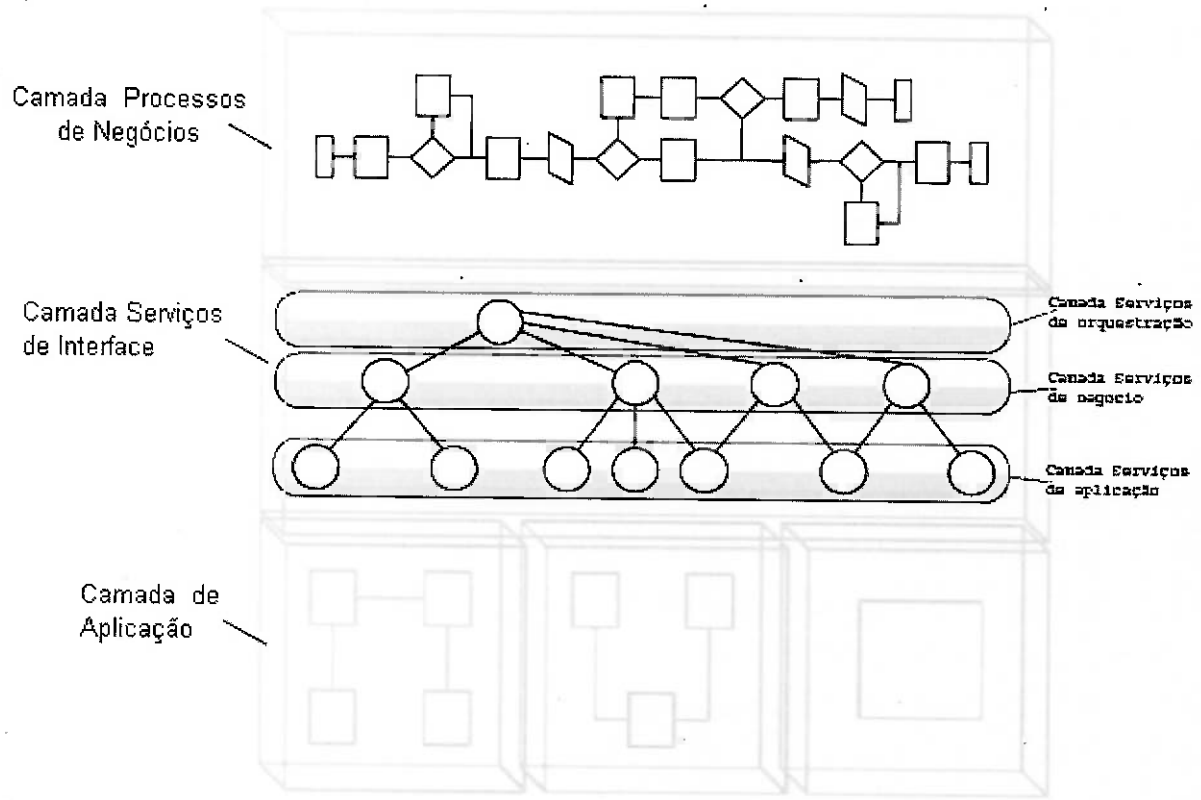

Figura 2.3 – As três camadas primárias de Serviços
(ERL, 2005)

2.3 COMPONENTES DE UMA ARQUITETURA ORIENTADA A SERVIÇOS

As definições a seguir têm como base o trabalho de Karacsony e Terry (2007), onde são apresentados os componentes de uma arquitetura SOA, seus papéis dentro desta arquitetura e como eles se relacionam.

A figura 2.4 mostra a colaboração entre o produtor, o repositório, e consumidor de serviços que são críticos na arquitetura orientada a serviços. A separação de papéis provê o baixo acoplamento, transparência e independência que permitem a flexibilidade do SOA.

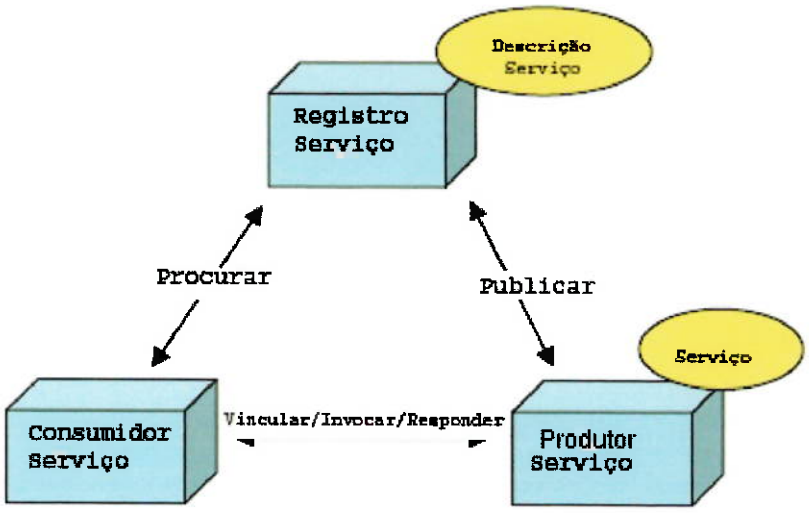

Figura 2.4 – Componentes do SOA
(KARACSONY;TERRY, 2007)

A seguir são descritos os principais itens de uma Arquitetura Orientada a Serviços.

a) PAPÉIS

- *Produtor de Serviços*: o produtor de serviços tem duas funções. Na primeira, divulga os serviços para o repositório de serviços para que os consumidores possam invocá-lo. Na segunda, recebe e executa requisições dos consumidores através de serviços pré-definidos via uma interface comum.
- *Repositório de Serviços*: no repositório de serviços é onde estão contidas informações dos serviços. O repositório aceita uma publicação de um produtor de serviços e é acessado por um consumidor que deseja encontrar um serviço. O repositório de serviço não sabe *como* um serviço será executado e nem *o que* ou *onde*. O repositório contém somente uma descrição dos serviços, a localização e como contatar estes serviços.
- *Consumidor de Serviços*: o consumidor de serviços pode ser uma aplicação, software, programa, ou outro serviço. O consumidor interage com o repositório de serviços para encontrar um serviço. Uma vez encontrado o serviço, o consumidor irá então conectar-se ao produtor de serviço e invocar um serviço de acordo com a interface, que define o formato da mensagem.

A figura 2.5 apresenta a interação dos componentes da Arquitetura Orientada a Serviços.

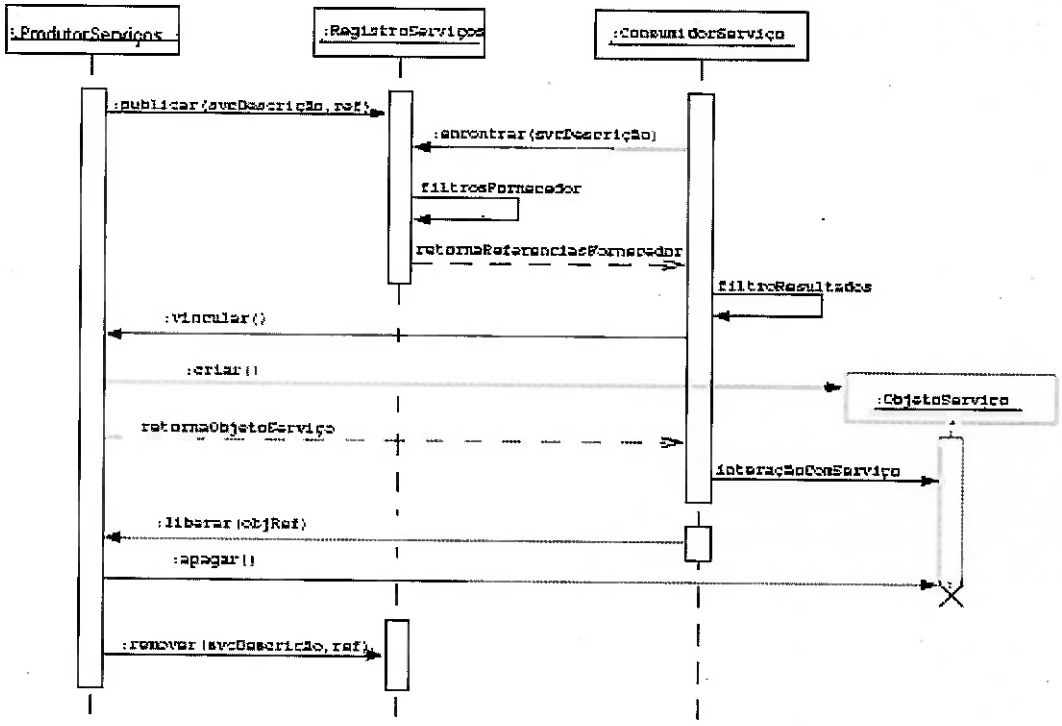

Figura 2.5 – Interação dos componentes da Arquitetura Orientada a Serviços (STOJANOVIC; DAHANAYAKE, 2005)

b) AÇÕES

- *Publicar*: é a ação de divulgar um serviço que pode ser encontrado e acessado por um consumidor. O que é publicado é a descrição que define o que um serviço faz e onde esta localizado.
- *Procurar*: um consumidor interage com um repositório para encontrar um serviço que atende uma necessidade especifica.
- *Conectar e Invocar*: uma vez encontrado um serviço, é conectado o produtor e então invocado o serviço. A maneira como o consumidor deve invocar um serviço esta descrita no repositório do serviço.
- *Resposta*: um pedido de serviço irá resultar em uma resposta do produtor para o consumidor, através de uma mensagem com um formato pré-definido que está no repositório de serviço, baseado em padrões, tais como XML.

c) ARTEFATOS

- *Serviço*: é o próprio serviço, com uma função de estado de negócio que executa uma ação.

- *Descrição do Serviço*: A descrição define o protocolo de interação entre consumidor e produtor. Especifica o formato de pedido e resposta de um serviço. Esta descrição pode ser especificada como um conjunto de pré-condições, pós-condições e/ou níveis de qualidade de serviço.

2.4 TECNOLOGIAS ORIENTADAS A SERVIÇO

As definições a seguir têm como base o trabalho de Stojanovic e Dahanayake (2005), são apresentadas algumas tecnologias SOA e em seguida essas tecnologias são comparadas.

Dos conceitos da orientação do serviço descritos previamente, é possível estabelecer uma lista das características que são úteis para categorizar as tecnologias orientadas a serviço, que são as plataformas que executam estes princípios.

- *Descrição do serviço:* a descrição do serviço propriamente dito.
- *Publicação do serviço:* as operações disponíveis no repositório de serviços, que os produtores e os consumidores de serviços podem publicar e revogar.
- *Procurar serviço:* as operações disponíveis para consumidor de serviço para encontrar e conectar um produtor de serviço bem como filtros de busca de serviços.
- *Políticas de criação de objetos de serviço:* políticas utilizadas pelos produtores de serviço quando estão criando objetos de serviço.
- *Notificação de serviços:* mecanismos de notificação suportados pela plataforma.
- *Liberação de serviços:* políticas suportadas para liberação de objetos de serviço.

A seguir são apresentadas as principais tecnologias da Arquitetura Orientada a Serviços.

a) CORBA Trader

CORBA Trader (STOJANOVIC; DAHANAYAKE, 2005) é um conjunto de serviços de *middleware* definidos na especificação CORBA (CORBA, 1995). O CORBA Trader se distingue de outras tecnologias orientadas a serviços pelo fato de suportar a criação de redes de comércio (conhecida como federação) que pode aumentar o

numéro de respostas que são retornadas para o consumidor, esta rede pode evoluir continuamente.

b) JAVABEANS Context

O conceito de JavaBeans context foi introduzido na seqüência da especificação do Modelo de componentes JavaBeans (SUN, 2007). É um grupo de instâncias de componentes JavaBeans sendo executados, que permite a estas instâncias obter serviços do repositório em tempo de execução.

As aplicações não são distribuídas e são montadas visualmente e são orientadas ao usuário, isto é, significa que suportam interação através de uma interface com usuário. No JavaBeans context, somente um produtor de serviço pode estar em um contexto (repositório de serviços) em determinado momento.

c) JINI

JINI (JINI, 2007) é uma plataforma de serviço distribuído definida pela Sun que compartilha vários conceitos do CORBA Trader. JINI é uma tecnologia Java que nivela as capacidades da plataforma Java para carregar dinamicamente um código da rede. A comunicação entre o consumidor e o objeto de serviço é sempre remota. O JINI suporta políticas de aluguel para a publicação e remoção de serviços, esses serviços são organizados em grupos. Outra característica é que o consumidor e produtor devem inicialmente estar em um repositório para iniciar a procura por serviços.

d) WEB SERVICES

Web Services surgiu da necessidade de interação entre aplicações heterogêneas residentes em diferentes corporações. Podem ser consideradas como heterogêneas também as diferenças de interação de modelos, protocolos de comunicação, e qualidade de serviços. A descrição dos serviços é feita através da linguagem chamada Web Service Description Language (WSDL), uma linguagem baseada em XML, o registro é feito através do Universal Description Discovery (UDDI).

A tabela 2.1 apresenta a comparação das principais características de cada tipo de tecnologia.

Tabela 2.1 – Sumário das características de tecnologias SOA (STOJANOVIC; DAHANAYAKE, 2005)

	CORBA Trader	JavaBeans Context	Jini	Web Services
Descrição Serviço	IDL interface + mandatorio e propriedades opcionais	Java interface ou classes	Java interface + atributos	WSDL descrição
Publicação Serviço	exportação recuperação	AddService revokeService	Register lease.cancel or lease expiration	save_service delete_service
Procura de Serviços e políticas de filtros para busca	Query linguagem restritiva, ordenação de resultados, politicas	GetService sem filtros	Lookup atributos de pedido devem estar na descrição do serviço	find_service
Política de criação de objetos de serviço	objetos compartilhados ou tudo (se proxy)	tudo	um objeto por vinculação ou objetos compartilhados	tudo
Notificações	não definido	serviço de saida e chegada	saida,chegada,alteração	saida,chegada,alteração
Liberação	explicita	explicita	implicita usando "gabage collection" e "leasing"	ambos
Distribuido	sim	não	sim	sim
Numero de registros	multiplos registros que colaboram e formam uma federação	um registro por contexto, mas pode ser agrupado hierarquicamente	multiplos registros sem colaboração	multiplos registros replicados
Outras caracteristicas		somente um fornecedor de serviço por serviço pode existir em um dado contexto	objetos de serviço são baixados para os clientes	contempla toda a interação de longo periodo

2.5 WEB SERVICES E SOA

Para se construir uma aplicação SOA, não é obrigatório utilizar Web Services, pois existem tecnologias equivalentes como as apresentadas na tabela 2.1, que implementam o paradigma da Arquitetura Orientada a Serviços, porém o Web Services é a tecnologia mais utilizada atualmente o que ajuda muito, tanto

considerando o aspecto de encontrar profissionais com experiência nesta tecnologia, quanto por ser uma tecnologia estável (ERL, 2005).

A tecnologia Web Services (BARRY, 2003) foi apresentada inicialmente como sendo a tecnologia de conexão do futuro. Essencialmente Web services utiliza XML para criar uma conexão robusta, o uso de XML retira a fragilidade do layout de registros fixos onde as conexões podem falhar se o formato apropriado não for utilizado. O XML permite também enviar mais dados, que podem ser utilizados ou não, com o Web services os dados enviados a mais não causam problemas com o serviço receptor como no padrão de registros fixos.

Os componentes básicos de Web Services são descritos a seguir.

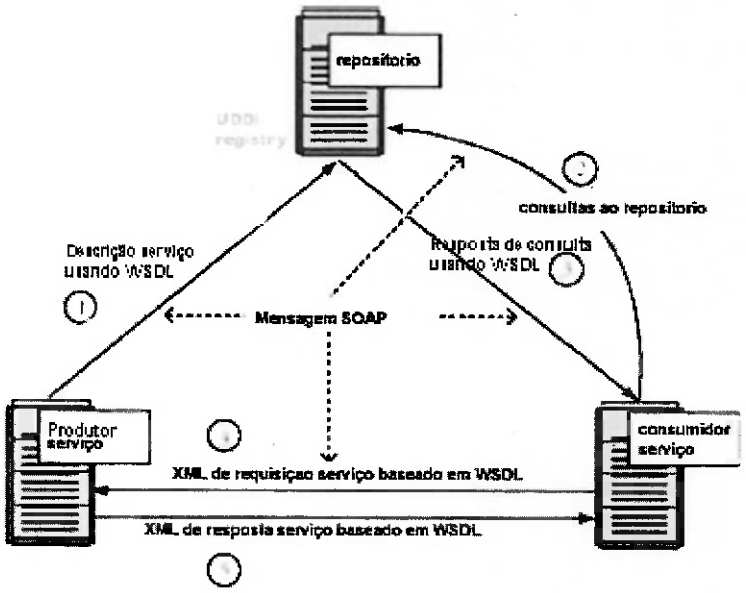

Figura 2.6 – Estrutura básica de Web Services
(BARRY, 2003)

a) WSDL

WSDL é a base de Web Services, a figura 2.6 mostra o uso do WSDL. Os passos, envolvendo um produtor e um consumidor de serviços, são:

- O produtor de serviço descreve o serviço utilizando WSDL, esta definição é publicada no repositório de serviços (directory). O repositório pode também utilizar o UDDI.

- O consumidor solicita uma ou mais consultas ao repositório para encontrar um serviço e descobrir como se comunicar com aquele serviço
- O repositório de serviços envia ao consumidor a informação necessária para que o mesmo possa utilizar os serviços do produtor.
- O consumidor usa o WSDL para enviar um pedido para o produtor.
- O produtor retorna a resposta esperada pelo consumidor.

b) UDDI

UDDI é utilizado para encontrar Web Services descritos com WSDL, a intenção é que o UDDI *registry* pode ser encontrado em vários caminhos para obter informações e os Web Services disponibilizados por várias empresas, mesmo sem o discovery, o UDDI registry é uma maneira para manter atualizado os web services que uma empresa utiliza.

c) SOAP

Todas as mensagens mostradas na *figura 2.6* são enviadas utilizando SOAP. SOAP essencialmente provê o meio para enviar as mensagens Web Services, geralmente utiliza HTTP, mas outros meios de conexão podem ser utilizados. HTTP é uma conexão familiar utilizada por toda internet, o que ajuda na adoção de Web Services.

d) XML com WSDL

WSDL utiliza XML para definir mensagens, tanto o produtor como o consumidor utiliza as tags do XML para enviar ou receber dados, ou seja, não importa a ordem dos dados, pois o que indica inicio e fim de um dado são as tags de inicio e fim, como apresentado na *figura 2.7*.

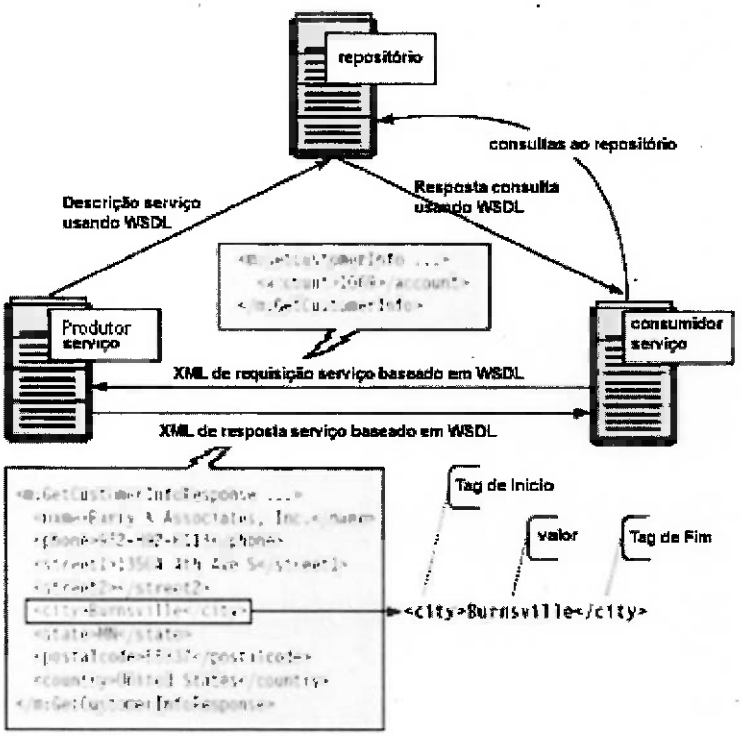

Figura 2.7 – Tags de mensagens
(BARRY, 2003)

2.6 CONSIDERAÇÕES FINAIS

Este capítulo destacou os conceitos e os princípios da Arquitetura Orientada a Serviços, aplicáveis a esse trabalho e, apresenta uma solução para facilitar sua implementação. Com a apresentação da evolução da arquitetura de software até chegar no SOA, fica evidente como foi a evolução da arquitetura, no inicio sistemas monolíticos depois se particionando os sistemas em módulos especializados até se chegar a arquitetura de serviços, onde cada parte ou componente é especializado e o conjunto destes componentes podem formar serviços de um determinado negócio.

Em seguida, é detalhada a composição do SOA que identifica quais são os componentes e artefatos essenciais para a construção de uma aplicação, além disso, também são apresentadas várias tecnologias que implementam o conceito de SOA e finalmente, foi detalhada uma destas tecnologias, o Web Services. O Web Services é uma tecnologia que melhor representa o SOA e apesar de muito recente já é bem difundida entre desenvolvedores e alguns fornecedores de software que

tem investido muito neste padrão (CLARK, 2002), o que é bom para o processo de desenvolvimento de software, fica mais fácil encontrar profissionais capacitados nesta tecnologia o que reduz riscos na disciplina de implementação, muitas linguagens implementam esta tecnologia , por exemplo, plataforma Java (JAVA, 2007), .Net (.NET, 2007).

3. O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

3.1 CONSIDERAÇÕES INICIAIS

O objetivo deste capítulo é apresentar, de maneira breve, os principais modelos de processo de software com base no trabalho de PRESSMAN (2002). E também é apresentada a metodologia da *Empresa Exemplo* em questão que será utilizada como estudo de caso. As disciplinas de Análise e Projeto são destacadas, pois a arquitetura de um sistema de software é definida nas fases iniciais de desenvolvimento, é uma das atividades críticas, pois a partir de sua definição outros artefatos são construídos. Essas duas disciplinas são apresentadas tendo como base o processo unificado (UP) (JACOBSON; BOOCH; RUMBAUGH, 1999) que também é aplicado na metodologia da *Empresa Exemplo*.

3.2 O PROCESSO DE SOFTWARE

Um processo de software pode ser caracterizado por uma Estrutura Comum de Processo, onde é definido um número de atividades dessa estrutura, que se aplicam à todos os projetos de sistema de software, independentemente de seu tamanho ou complexidade.

Um conjunto de tarefas, constituído por uma coleção de atividades específicas da área de engenharia de software, marcos de projeto, produtos do trabalho e pontos de garantia de qualidade, permite que as atividades da estrutura sejam adaptadas às características do projeto de software e às necessidades da equipe de software. As atividades, tais como, garantia da qualidade de software, gestão de configuração de software e medição, são independentes de qualquer atividade da estrutura e ocorrem ao longo de todo o processo.

Um modelo de processo para engenharia de software é escolhido com base na natureza do tipo de aplicação, nos métodos e nas ferramentas a serem usadas, nos controles e nos produtos intermediários e finais que são requeridos. O

desenvolvimento de software pode ser caracterizado como um ciclo de solução de problema, conforme apresentado na figura 3.1, onde:

- A situação atual representa o estado atual das coisas;
- A definição do problema identifica o problema específico a ser resolvido;
- O desenvolvimento técnico resolve o problema por intermédio da aplicação de alguma tecnologia e;
- A integração da solução entrega os resultados àqueles que solicitaram a solução inicialmente.

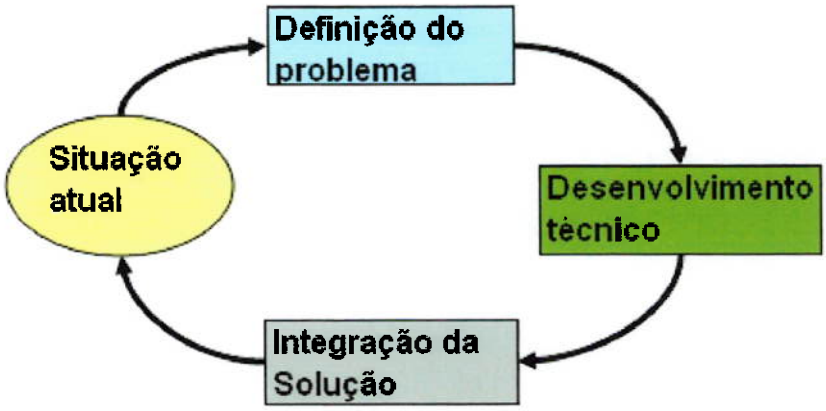

Figura 3.1 – Fases de um ciclo de solução de problema
(PRESSMAN, 2002)

Para se ter uma visão da evolução de processos de desenvolvimento de software são apresentados os modelos mais representativos de processos de desenvolvimento até se chegar ao processo unificado.

a) MODELO SEQUENCIAL LINEAR

Este modelo também é conhecido como Modelo Clássico ou Cascata, sugere uma abordagem sistemática seqüencial para o desenvolvimento de software, que se inicia no nível de sistema e progride através da análise, projeto, codificação, teste e manutenção. A figura 3.2 ilustra o modelo seqüencial linear para a engenharia de software.

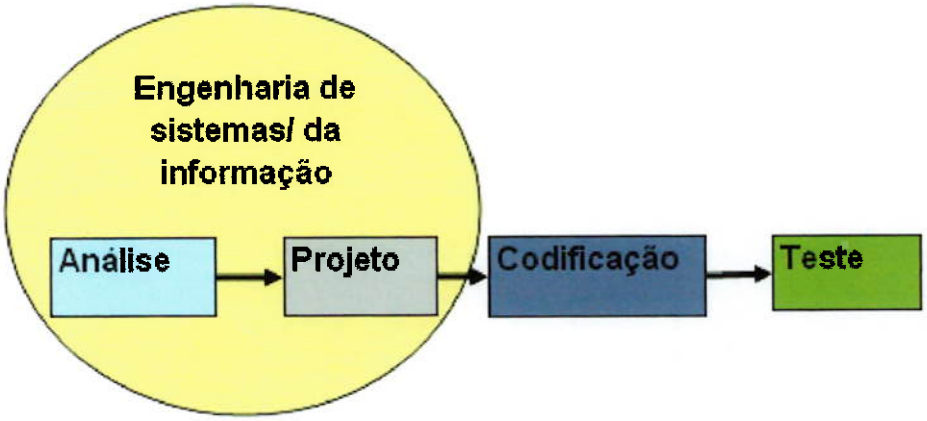

Figura 3.2 – O modelo seqüencial linear
(PRESSMAN, 2002)

O modelo seqüencial linear é o mais antigo e é um paradigma amplamente usado, porém, a crítica levou ao questionamento de sua eficácia, entre os problemas mais comuns estão:

- Projetos reais raramente seguem o fluxo seqüencial, modificações podem causar confusão à medida que a equipe de projeto prossegue;
- É difícil para o cliente estabelecer todos os requisitos explicitamente;
- O cliente precisa ter paciência, pois uma versão do executável não vai ficar disponível até o projeto terminar.

O ciclo de vida clássico ou cascata continua sendo um modelo amplamente usado. Apesar de ter pontos fracos, é significativamente melhor que uma abordagem aleatória para o desenvolvimento de software.

b) MODELO DE PROTOTIPAGEM

O paradigma de prototipagem figura 3.3 começa com a definição de requisitos, onde o desenvolvedor e o cliente definem os objetivos gerais do software, identificam as necessidades conhecidas e delineiam áreas que necessitam de mais definições. Esse modelo concentra-se na representação daqueles aspectos do software que ficarão visíveis ao cliente/usuário, o protótipo é avaliado pelo cliente/usuário e usado para refinar os requisitos do software que será desenvolvido. Interações ocorrem à medida que o protótipo é ajustado para satisfazer às necessidades do cliente,

enquanto, ao mesmo tempo, permitem ao desenvolvedor entender melhor o que precisa ser feito.

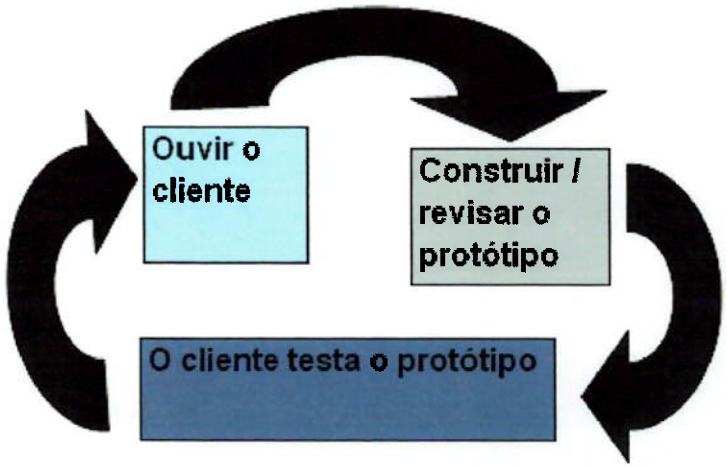

Figura 3.3 – O modelo de prototipagem
(PRESSMAN, 2002)

Os usuários conseguem ter uma visão real do sistema e os desenvolvedores podem construir algo imediatamente, porém, a prototipagem pode ser problemática pelas seguintes razões:

- O cliente aprova o que parece ser a versão executável do software, ignorando que o protótipo apenas consegue funcionar precariamente, não levando em conta a sua qualidade global ou manutenibilidade em longo prazo.
- O desenvolvedor freqüentemente faz concessões na implementação, a fim de obter rapidamente um protótipo executável. Um sistema operacional ou uma linguagem de programação inapropriada pode ser usado por estar disponível e serem conhecidos; um algoritmo ineficiente pode ser implementado para demonstrar uma possibilidade, a escolha muito aquém da ideal tornou-se agora parte integral do sistema.

Apesar dos problemas, a prototipagem pode ser de grande valia, o importante é o cliente e o desenvolvedor estarem de acordo que o protótipo seja construído para servir como um mecanismo para definição dos requisitos, mesmo que depois ele seja descartado.

c) MODELO INCREMENTAL

O modelo incremental combina elementos do modelo seqüencial linear com a filosofia interativa da prototipagem como apresentado na figura 3.4. O modelo incremental aplica seqüências lineares ao longo do desenvolvimento do software, onde ao final de cada uma é gerado um executável do software que será avaliado pelo usuário e revisado, um plano é desenvolvido para o próximo incremento, como resultado do uso e/ou avaliação. O primeiro incremento é frequentemente o núcleo do sistema, ou seja, satisfaz os requisitos essenciais. .

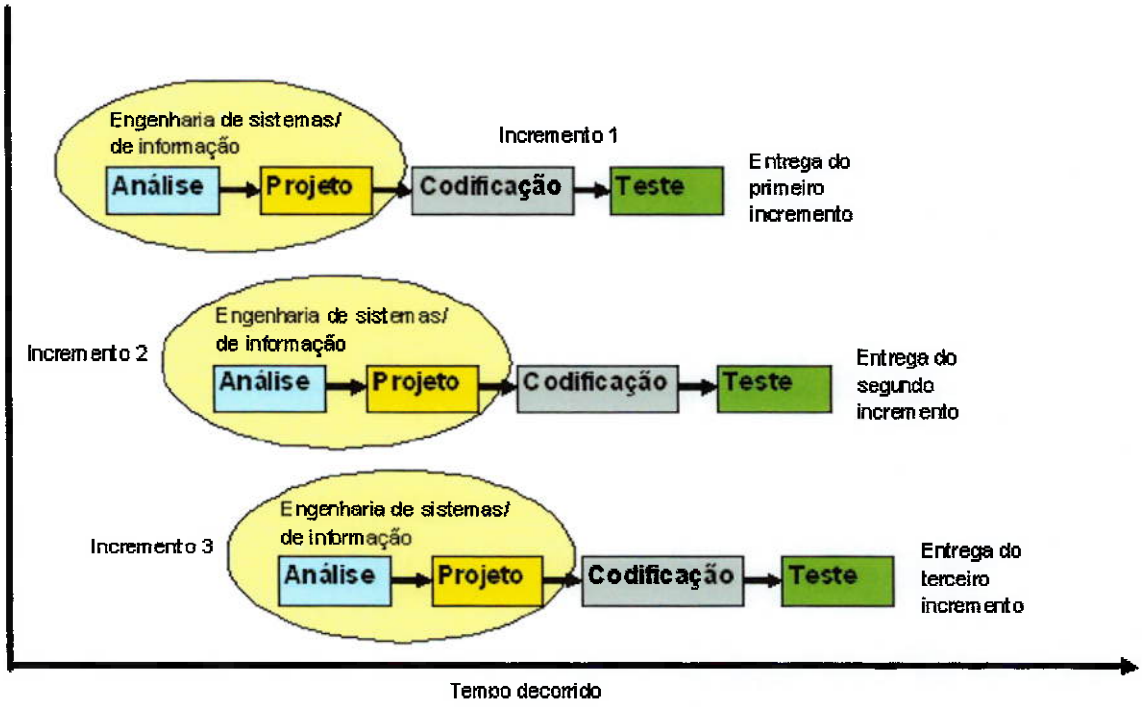

Figura 3.4 – O modelo incremental
(PRESSMAN, 2002)

d) MODELO ESPIRAL

O modelo espiral, originalmente proposto por Boehm (BOEHM, 1988), é um modelo evolucionário que combina a natureza iterativa da prototipagem com os aspectos controlados e sistemáticos do modelo seqüencial linear. O software é desenvolvido numa série de versões incrementais, onde as primeiras iterações podem resultar em um modelo de papel ou protótipo, e nas próximas iterações são produzidas versões cada vez mais completas do sistema. O modelo espiral é formado por certo número de atividades chamadas de regiões de tarefas, conforme mostra a figura 3.5:

- *Comunicação com o cliente:* tarefas necessárias para estabelecer efetiva comunicação entre o desenvolvedor e o cliente
- *Planejamento:* tarefas necessárias para definir recursos, prazos e outras informações relacionadas ao projeto.
- *Análise de risco:* tarefas necessárias para avaliar os riscos, tanto técnicos quanto gerenciais.
- *Engenharia:* tarefas necessárias para construir uma ou mais representações da aplicação
- *Construção e liberação:* tarefas necessárias para construir, testar, instalar e fornecer apoio ao usuário.
- *Avaliação pelo cliente:* tarefas necessárias para obter realimentação do cliente, com base na avaliação das representações do software criadas durante o estágio de engenharia e implementadas durante o estágio de instalação.

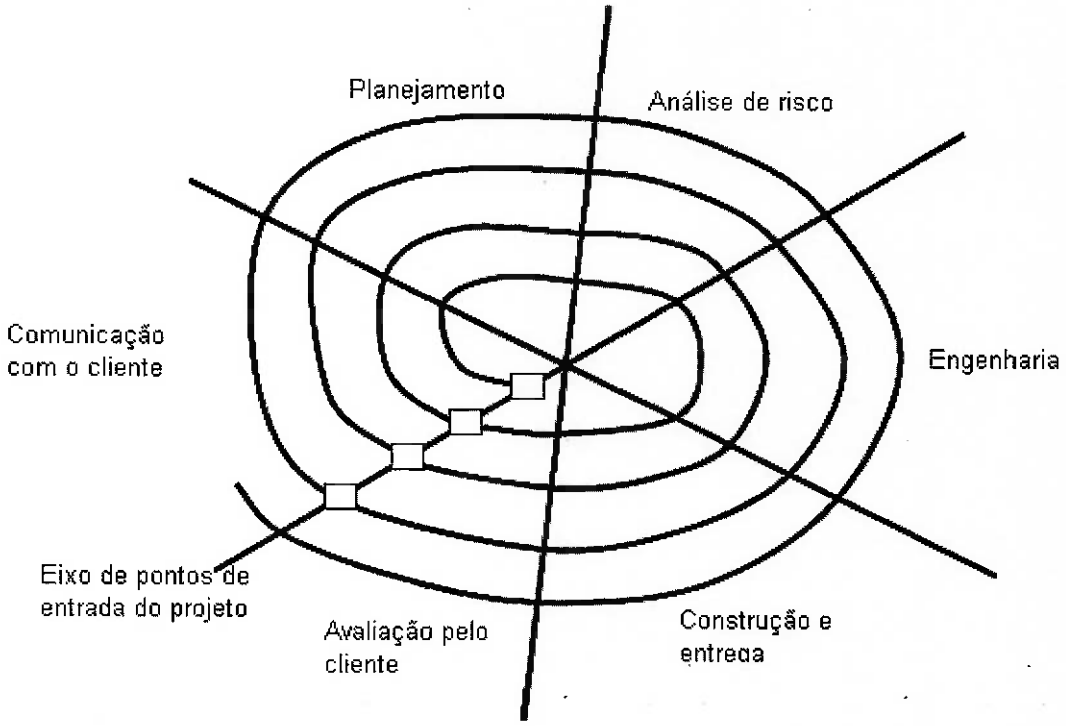

Figura 3.5 – O modelo espiral
(PRESSMAN, 2002)

No modelo espiral, o conjunto de tarefas de trabalho é adaptado às características do projeto a ser desenvolvido, para projetos pequenos, a quantidade e suas formalidades são pequenas, para projetos maiores, mais críticos, cada conjunto possui mais tarefas, que são definidas para alcançar um alto nível de formalidade. O modelo espiral exige competência considerável na avaliação de risco e depende

dessa competência para ter sucesso, se um risco importante não for descoberto e gerenciado, fatalmente ocorrerão problemas.

3.3 O PROCESSO UNIFICADO DE DESENVOLVIMENTO DE SOFTWARE

O ciclo de vida do processo unificado é composto de quatro fases: concepção, elaboração, construção e transição (JACOBSON; BOOCH; RUMBAUGH, 1999). Cada fase é subdividida em iterações, onde cada iteração tipicamente incorpora todas as atividades, como mostra a figura 3.6.

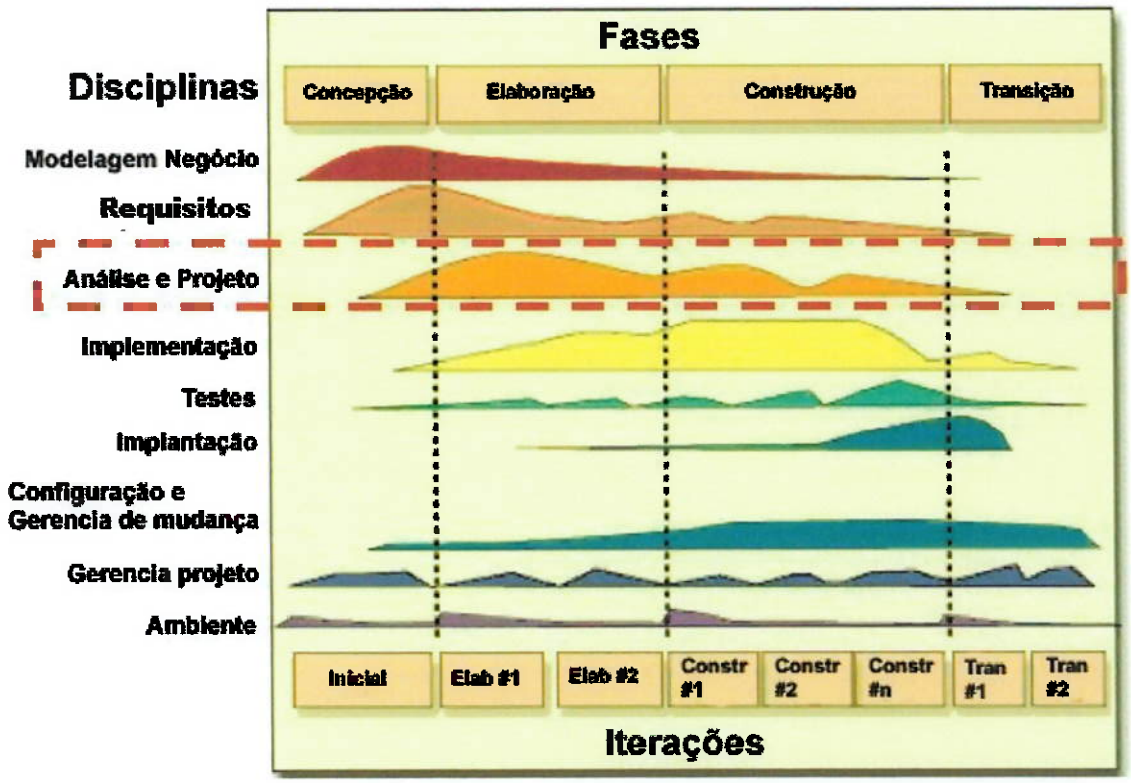

Figura 3.6 – O modelo do Processo Unificado
(JACOBSON; BOOCH; RUMBAUGH, 1999)

A seguir são apresentadas as principais características das disciplinas de Análise e Projeto desse processo.

3.3.1 ANÁLISE

Na análise os requisitos são analisados, refinados e estruturados, de forma a se obter uma compreensão mais precisa destes, o que irá ajudar a estruturar o sistema

como um todo e sua arquitetura. A linguagem utilizada na análise é baseada no modelo conceitual de objetos, chamado de Modelo de Análise este modelo permite refinar e estruturar os requisitos e mais importante, permite uma abstração para a solução de alguns problemas e também o adiamento de alguns requisitos para as disciplinas de projeto e implementação.

Nas primeiras iterações da fase de Elaboração a análise contribui para se obter uma arquitetura estável e facilitar um entendimento mais profundo dos requisitos, ou seja, a análise nesta fase inicia o detalhamento de cada requisito proporcionando delinear uma arquitetura para o sistema, nas iterações mais para o fim da elaboração e inicio da construção quando a arquitetura já é estável e os requisitos estão claros, o foco passa a ser as disciplinas de projeto e implementação, ou seja, uma vez finalizado o detalhamento dos requisitos de negócio cabe a disciplina de projeto e implementação aplicar a tecnologia delineada pela arquitetura definida.

a) ARTEFATOS

Com o produto gerado a partir do modelo de análise obtêm-se:

- *Pacotes de análise e serviço:* os pacotes de serviços identificam mudanças individuais nos serviços oferecidos pelo sistema e são base para o reuso.
- *Classes de Análise:* cada tipo de classe (controle, entidade, interface) serve para identificar mudanças no comportamento e informações destes tipos de classe.
- *Realização de Casos de Uso:* descreve como os casos de uso são refinados em termos de colaboração no modelo de análise, serve para localizar mudanças nos casos de uso.
- *Visão arquitetural do Modelo de Análise:* são as classes de análise, pacotes e realização de casos de uso, é a visão do ponto de vista arquitetural destes artefatos, através desta visão é possível identificar mudanças na arquitetura, quando necessário.

b) PAPÉIS

Os papéis considerados para desenvolver a disciplina de análise são:

- *Arquiteto:* é o responsável pela integridade do modelo de análise, assegurando que o todo esteja correto, consistente, e de fácil compreensão. A arquitetura descrita pelo arquiteto, deve atender aos requisitos descritos no modelo do caso de uso. O arquiteto também deve manter a compatibilidade dos novos requisitos com a arquitetura já existente no caso de um sistema construído.
- *Engenheiro de Caso de Uso:* é responsável pela integridade de um ou vários realização de casos de uso, assegurando que todos os requisitos do caso de uso sejam atingidos.
- *Engenheiro de Componentes:* é o responsável por definir e manter responsabilidades, atributos, relacionamentos e atributos especiais de uma ou várias classes de análise.

c) FLUXO DE TRABALHO

A figura 3.7 apresenta os papéis e suas responsabilidades na disciplina de Análise.

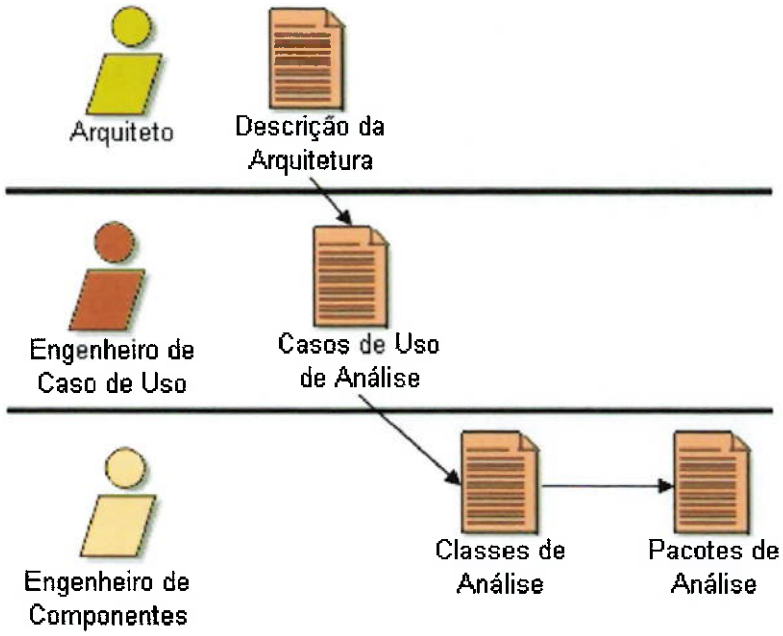

Figura 3.7 – Fluxo de Trabalho da Análise
(JACOBSON; BOOCH; RUMBAUGH, 1999)

O fluxo de trabalho se inicia com o arquiteto identificando os maiores pacotes de análise, então o engenheiro de caso de uso cria a realização de cada caso de uso, em termos de participação das classes de análise, pelo comportamento dos

requisitos em cada classe. Estes requisitos são especificados pelo engenheiro de componentes e integrados em cada classe.

3.3.2 PROJETO

No projeto, o sistema ganha uma forma que represente todos os requisitos, incluindo os requisitos não funcionais e outras restrições. Uma entrada essencial no projeto é o resultado da disciplina de análise. Os principais objetivos do projeto são:

- Entender os requisitos não funcionais e as restrições relacionados à linguagem de programação, reuso, sistema operacional, distribuição e concorrência, banco de dados, interface com usuário, gerenciamento de transações.
- Gerar saída para a próxima disciplina, a implementação.
- Decompor o sistema para que a implementação possa ser gerenciada em partes, possibilitando a divisão em equipes.
- Obter a maioria das interfaces entre subsistemas no inicio do ciclo de vida do desenvolvimento.
- Utilizar uma notação comum.
- Possibilitar uma abstração para a disciplina de implementação, fazendo com que a mesma seja um refinamento do projeto, mas sem mudar sua estrutura.

a) ARTEFATOS

Com o produto gerado do modelo de projeto tem-se:

- O projeto de subsistemas e serviços em duas camadas altas que devem ser demonstradas nos pacotes de análise.
- Projeto de classes, incluindo classes ativas, suas operações, atributos, relacionamentos e a implementação dos requisitos.
- Realização de casos de uso, como os casos de uso são projetados em termos de colaboração no modelo de projeto.
- A visão arquitetural, incluindo elementos significantes para a arquitetura tais como infra-estrutura de rede e hardware, subsistemas e suas interfaces, mecanismos genéricos para atingir requisitos não funcionais.

- Diagrama de implantação, descrevendo infra-estrutura de rede e hardware suas características e conexões, um mapa inicial das classes ativas sobre o hardware.

b) PAPÉIS

Os papéis considerados para desenvolver a disciplina de análise são:

- *Arquiteto:* é responsável pela integração dos modelos de projeto e implantação, garante que os modelos estejam corretos, consistentes e de fácil compreensão. Estes dois modelos devem realizar as funções descritas no modelo de casos de uso, requisitos suplementares, e modelo de análise.
- *Engenheiro de Casos de Uso:* é responsável pela integração de uma ou várias realizações de casos de uso, garante que o comportamento de todos os requisitos sejam atendidos por este modelo.
- *Engenheiro de Componentes:* define e mantém as operações, métodos, atributos, relacionamentos e a implementação de um ou vários requisitos no projeto de classes. Também mantém a integridade de um ou mais subsistemas e suas interfaces e dependências.

c) FLUXO DE TRABALHO

A figura 3.8 apresenta os papéis e suas responsabilidades na disciplina de Projeto.

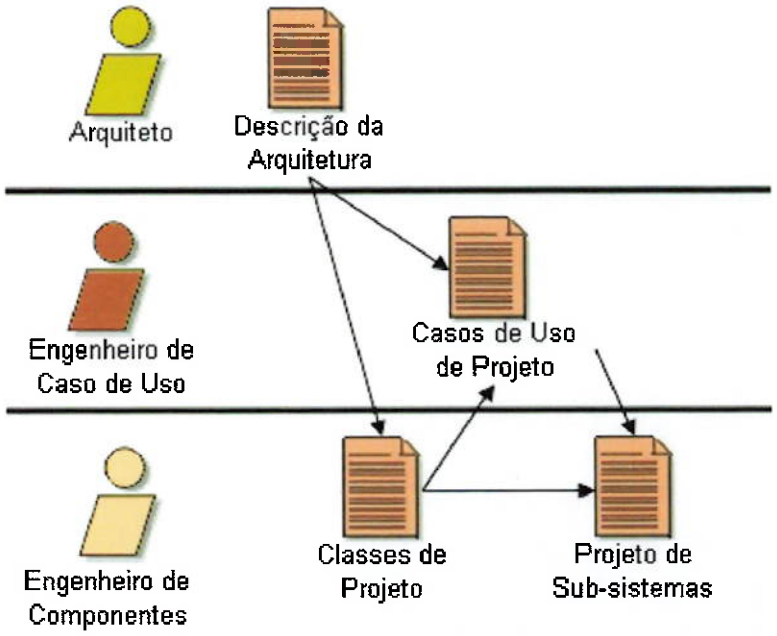

Figura 3.8 – Fluxo de Trabalho de Projeto
(JACOBSON; BOOCH; RUMBAUGH, 1999)

O fluxo de trabalho se inicia com o arquiteto através dos modelos de projeto e implantação, que define a infra-estrutura de rede e hardware, para a maioria dos subsistemas e suas interfaces. Então o engenheiro de casos de uso realiza cada caso de uso em termos de participação no projeto de classes e/ou subsistemas e suas interfaces. O engenheiro de componentes especifica os requisitos e integra em cada classe, criando as operações, atributos e relacionamentos de cada classe.

3.4 A METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE DA *EMPRESA EXEMPLO*

A metodologia de desenvolvimento de software da *Empresa Exemplo* é bem recente, sendo que sua primeira versão foi liberada em Janeiro de 2006. A necessidade de uma metodologia de desenvolvimento de software, assim como outras mudanças na área de Tecnologia de Informação, surgiu devido a uma exigência da matriz, onde todas as filiais pelo mundo deveriam ter aderência ao SOX (MOELLER, 2004) até o final de 2006.

A implantação do SOX na área de TI da filial brasileira teve um grande impacto, pois não havia um processo formal de desenvolvimento. Cada área desenvolvia sistemas à sua maneira, sem nenhuma padronização, o que dificultava a manutenção e o desenvolvimento, não havia segregação de ambientes, ou seja, os desenvolvedores tinham acesso ao ambiente de produção, onde também eram responsáveis pelo suporte aos usuários.

Com o SOX, a tarefas passaram a ser divididas o que trouxe um pouco de dificuldade para os usuários e desenvolvedores, mas em compensação, trouxe grandes benefícios para a empresa, uma vez que tudo passa a ser documentado e auditado, um dos grandes benefícios foi a criação da metodologia de desenvolvimento de software para o TI.

A metodologia de desenvolvimento de software da *Empresa Exemplo* tem como base o Processo Unificado (JACOBSON; BOOCH; RUMABUGH, 1999), segue tanto

as fases, isto é, Concepção, Elaboração, Construção e Transição, quanto as disciplinas, isto é, Requisitos, Análise e Projeto, Implementação, Testes e Implantação, com exceção da Modelagem de Negócio e Gerencia de Configuração que ainda não estão no escopo desta primeira versão da metodologia.

Artefatos resultantes das disciplinas do Processo Unificado foram adaptados para atender a área de Tecnologia da Informação. A metodologia foi construída por pessoas que contribuíram com sua experiência e conhecimento, de maneira à atender as necessidade da empresa em se obter um processo de desenvolvimento ágil, mas que preze a qualidade da solução de software gerada ao final do processo.

Assim como no UP, o processo é *Dirigido aos Casos de Uso,* significa que o processo de desenvolvimento segue um fluxo, realizado através de uma série de atividades que derivam dos casos de uso. Os casos de uso são especificados, projetados e, por fim, os casos de uso servem de fonte para a construção dos casos de teste e de outros artefatos ao longo do processo de desenvolvimento.

Na metodologia desenvolvida, foi criado um *framework* de arquitetura da *Empresa Exemplo* (FRAMEWORKX, 2007). A partir desse *framework*, todos os projetos têm como base esta arquitetura, que já prevê o uso de outros *frameworks* de mercado (FRAMEWORK, 2007) e a utilização de melhores práticas *patterns* (LARMAN, 2004), o que facilita e padroniza o desenvolvimento de software.

No Processo Unificado, o desenvolvimento de software é Iterativo e Incremental, é uma boa prática dividir o trabalho em partes ou em mini-projetos, onde cada mini-projeto é uma iteração que resulta em um incremento. As iterações se referem aos passos das disciplinas e os incrementos ao crescimento do produto. Na *Empresa Exemplo,* normalmente os projetos de software são desenvolvidos como o processo cascata, mas para alguns projetos longos já se utilizou o processo iterativo e incremental.

O Anexo I mostra os artefatos produzidos nos fluxos de trabalho de Análise e Projeto da *Empresa Exemplo.*

3.4.1 ANÁLISE

Nessa disciplina, primeiramente é criado o Modelo Entidade Relacionamento (MER) (MACHADO, F., 2006), com as tabelas participantes do sistema. Além disso, também é criado um dicionário de dados com os campos e índices por tabela. Para os diagramas de casos de uso é criado um modelo de domínio, sendo detalhada uma especificação técnica do caso de uso, onde é descrito um algoritmo considerando as tabelas do banco de dados que participam do caso de uso, incluindo os campos e restrições do algoritmo. Ao final pode-se ter um termo de aceite do usuário, para projetos que possuem uma validação ao final da Especificação e Análise de software.

a) ARTEFATOS

Como produtos gerados na disciplina de análise têm-se:

- *Especificação Técnica de Casos de Uso:* utilizando-se o MER e a descrição dos casos de uso. São construídos algoritmos que serão utilizados para a implementação dos casos de uso, nestes algoritmos são mencionadas as tabelas e campos do banco de dados a ser utilizado.
- *Modelo Entidade Relacionamento (MER):* é criado um modelo de dados relacional com todas as tabelas que participam do sistema, e seus relacionamentos.
- *Dicionário de Dados:* descreve todas as tabelas do MER com suas colunas, formatos, descrições e índices.
- *Diagrama de Domínio:* é criado com as classes de negócio identificadas nesta disciplina e seus relacionamentos.
- *Diagrama de Casos de Uso:* são criados os diagramas de todos os casos de uso identificados nesta disciplina, depois é criado um diagrama de caso de uso para o sistema todo que é representação de todo o sistema, este diagrama contém somente os principais casos de uso, seus relacionamentos e atores.

b) PAPÉIS

Os papéis considerados para desenvolver a disciplina de análise são:

- *Analista de Sistemas:* é responsável pela criação dos diagramas de casos de uso e de domínio.
- *DBA:* é responsável pela criação e integridade do modelo de dados, com base na descrição dos casos de uso, documento de visão e diagrama de domínio.
- *Projetista:* o projetista cria a especificação técnica dos casos de uso, com base no modelo de dados e descrição dos casos de uso.

c) FLUXO DE TRABALHO

A figura 3.9 apresenta os papéis e suas responsabilidades na disciplina de Análise.

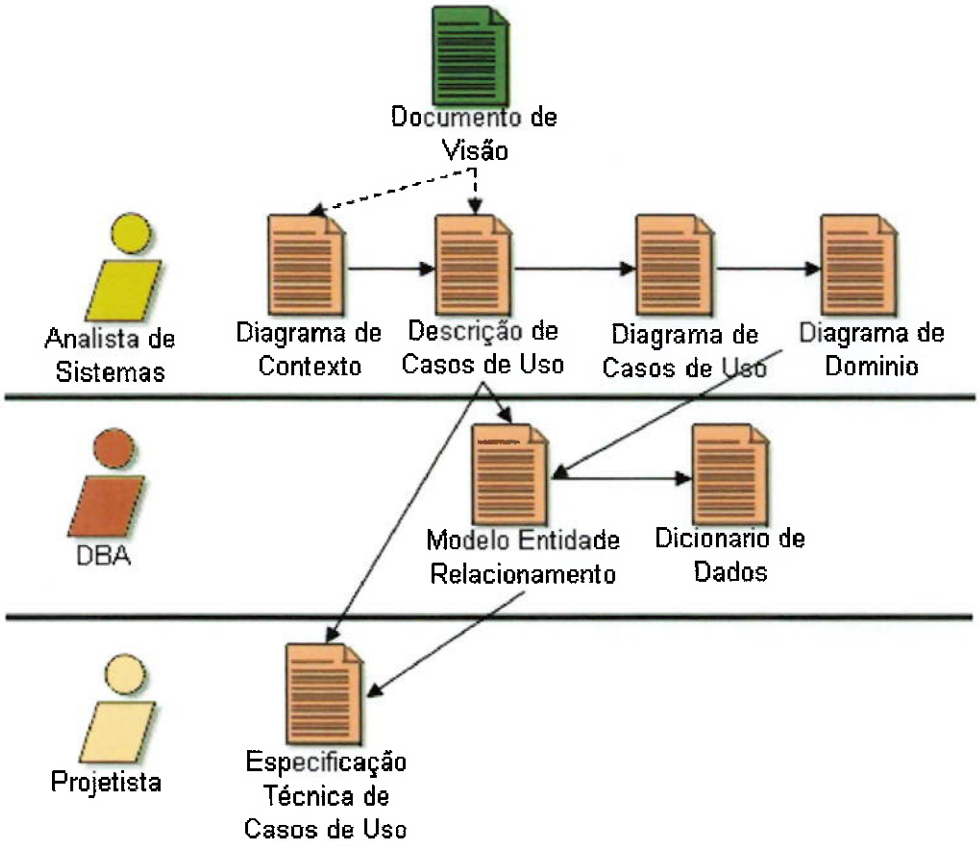

Figura 3.9 – Fluxo de Trabalho da disciplina de Análise da Metodologia

O fluxo de trabalho se inicia com o analista de sistemas identificando as classes de negócio e seus relacionamentos para então criar o diagrama de domínio, depois o analista cria diagramas para todos os casos de uso e um diagrama com todos os casos de uso que representa o sistema. O administrador de dados (DBA) cria o modelo e o dicionário de dados baseado na descrição dos casos de uso e no

diagrama de domínio, o projetista inicia a especificação técnica dos casos de uso utilizando para isto a descrição dos casos de uso e o modelo de dados.

3.4.2 PROJETO

A disciplina de projeto tem como um dos objetivos dar forma a todos os requisitos, incluindo não funcionais. Uma entrada essencial no projeto é o resultado da disciplina de análise, que é o modelo de análise. É nesta disciplina que é criado o documento de arquitetura, base para todos os diagramas.

A *Empresa Exemplo* adota uma arquitetura padrão chamada de *FRAMEWORKX* (FRAMEWORKX, 2007) que é aplicada a todos os projetos, e que tem flexibilidade para ser adaptada a sistemas que venham ser desenvolvidos. Além disso, também existem documentos que servem de referência para esta disciplina, tais como, *Documento de Arquitetura Software TDB*, *FRAMEWORKX*, *Documento de Papéis de Ferramentas*, *Design Guidelines*. O documento de descrição da arquitetura é baseado na metodologia *SunTone Architecture Methodology* (SUN, 2007).

Os principais objetivos da disciplina de projeto são:

- Entender e analisar os requisitos não funcionais e as restrições relacionados à linguagem de programação, reuso, sistema operacional, distribuição e concorrência, banco de dados, interface com usuário, gerenciamento de transações e outros.
- Gerar artefatos para a próxima disciplina a implementação.
- Decompor o sistema para que a implementação possa ser gerenciada em partes, possibilitando a divisão da implementação em equipes.
- Obter a maioria das interfaces entre subsistemas já no inicio do ciclo de vida do desenvolvimento.
- Utilizar uma notação comum para todos os envolvidos no desenvolvimento.
- Possibilitar uma abstração para a disciplina de implementação, fazendo com que a mesma seja um refinamento do projeto, mas sem mudar sua estrutura.

a) ARTEFATOS

Como produto gerado do modelo de projeto tem-se:

- Diagrama de Implantação, descrevendo infra-estrutura de rede e hardware suas características e conexões, um mapa inicial das classes ativas sobre o hardware.
- O Documento de Arquitetura, que é composto pela identificação dos principais requisitos do sistema, diagramas de implantação de alto nível e detalhado, diagrama de pacotes e níveis de camada, matriz de qualidade e níveis, FRAMEWORKX.
- Projeto de classes, incluindo classe ativas, suas operações, atributos, relacionamentos e a implementação dos requisitos.
- O diagrama de seqüência e de atividade não são obrigatórios, ficando a cargo do projetista a sua confecção para os casos de uso de maior complexidade.

b) PAPÉIS

Os papéis considerados para desenvolver a disciplina de projeto são:

- *Arquiteto:* é responsável pela integração dos modelos de projeto e implantação, garante que os modelos estejam corretos, consistentes e de fácil compreensão e que por sua vez estejam dentro dos padrões de projeto estabelecidos pela empresa, de acordo com o FRAMEWORKX. Estes modelos estão descritos no documento de arquitetura, estes dois modelos devem realizar funcionalidades descritas no modelo de casos de uso, requisitos suplementares, e modelo de análise.
- *Projetista e Analista de Sistemas:* define e mantém as operações, métodos, atributos, relacionamentos e a implementação de um ou vários requisitos no projeto de classes. Também mantém a integridade de um ou mais subsistemas e suas interfaces e dependências. O projetista também é responsável pela criação e manutenção dos diagramas de seqüência e atividade.

c) FLUXO DE TRABALHO

A figura 3.10 apresenta os papéis e suas responsabilidades na disciplina de Análise.

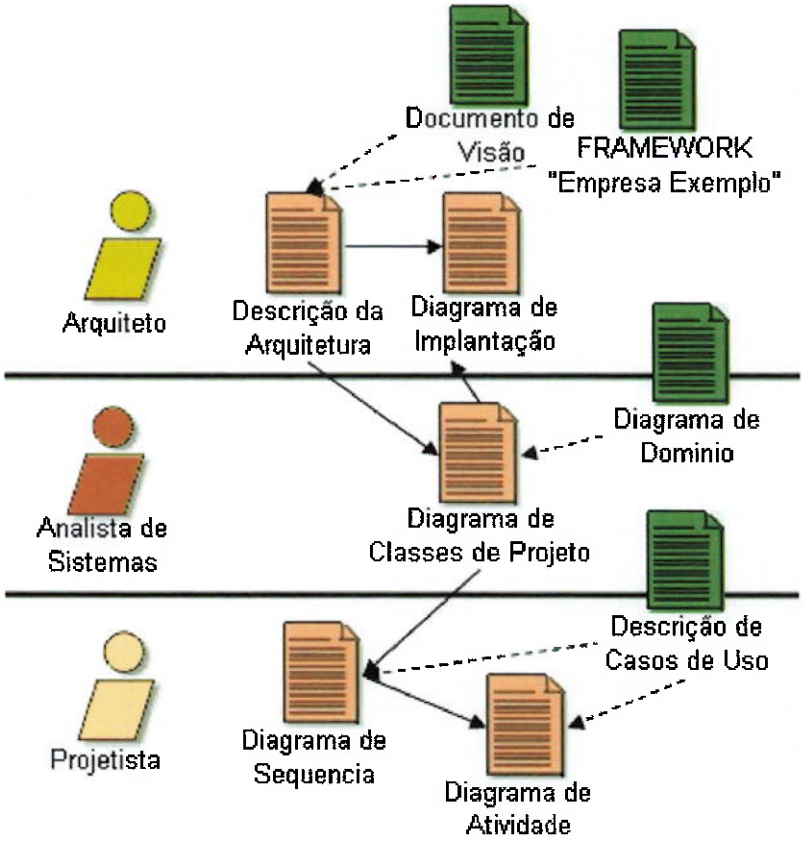

Figura 3.10 – Fluxo de Trabalho da disciplina de Projeto da Metodologia

O fluxo de trabalho se inicia com o arquiteto através dos modelos de projeto e implantação, que define a infra-estrutura de rede e hardware, para a maioria dos subsistemas e suas interfaces. A definição esta contida no documento de descrição da arquitetura, então o projetista e analista de sistemas realizam cada caso de uso, em termos de participação no projeto de classes e/ou subsistemas e suas interfaces. Especificam os requisitos e integram em cada classe, criando as operações, atributos e relacionamentos de cada classe. Se necessário, o projetista cria os diagramas de seqüência e atividade para os casos de uso mais complexos.

3.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou primeiramente a definição de processo de software e alguns modelos de processo, depois foi apresentado o UP focando-se as disciplinas de Análise e Projeto que são relevantes para este trabalho e por último foi apresentada a metodologia de desenvolvimento de software utilizada na *Empresa*

Exemplo que foi utilizada como estudo de caso, focando-se também as disciplinas de Análise e Projeto.

A metodologia utilizada pela *Empresa Exemplo* tem como base o UP, sendo que foram feitas adaptações às necessidades da empresa, pois nem todos os papéis são desempenhados na empresa. Alguns papéis foram redefinidos e, artefatos foram modificados e/ou excluídos. Também se pode observar que a metodologia apresentada não considera a arquitetura orientada a serviços.

4. AVALIAÇÃO DA ARQUITETURA ORIENTADA A SERVIÇOS

4.1 CONSIDERAÇÕES INICIAIS

O objetivo deste capítulo é apresentar uma avaliação do SOA no processo de desenvolvimento de software. Portanto, é analisada uma proposta de ciclo de desenvolvimento de software (GRÜNBAUER et al., 2004), mais especificamente as disciplinas de Análise e Projeto, destacando-se pontos que podem ser adaptados nestas disciplinas para se adequar ao SOA. A avaliação deste processo de desenvolvimento de software para SOA proposta por Grünbauer et al. (2004), é a base para a proposta abordada neste trabalho.

Primeiramente é apresentado o ciclo de desenvolvimento de um sistema de software, utilizando SOA. Em seguida, é feita uma análise do ciclo de desenvolvimento apresentado.

4.2 CICLO DE DESENVOLVIMENTO DE SOFTWARE COM SOA

A proposta de Grünbauer et al. (2004), conforme figura 4.1, apresenta um ciclo de desenvolvimento de software baseado em processos de software apresentados no capítulo anterior, tais como o modelo cascata, e o processo unificado. A proposta é integrar o conceito de serviços no processo de desenvolvimento de software.

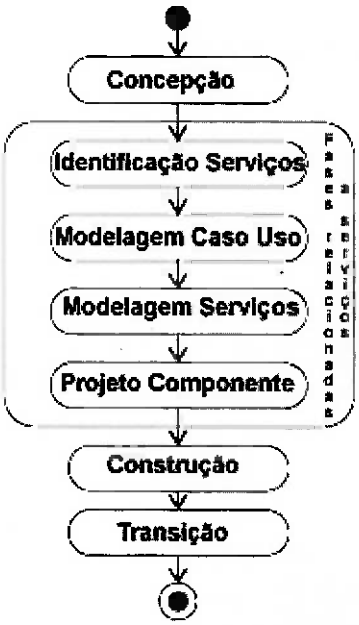

Figura 4.1 – Fases do Processo de Desenvolvimento Orientado a Serviços (GRÜNBAUER et al., 2004)

A figura 4.1 mostra as fases do processo de desenvolvimento de software orientado a serviços. Na fase de Concepção é estabelecida a missão do projeto e onde são elaborados os requisitos, é gerada uma lista de requisitos. Este é o ponto inicial do processo, porém nesta fase ainda não é aplicado o conceito de serviço. O produto final desta fase é o documento que descreve a missão do projeto e a lista de requisitos.

Em seguida a concepção tem-se a fase de Identificação dos Serviços, onde é feita a separação dos sistemas, ou seja, são identificados os serviços que tem afinidade ou que estão dentro de um mesmo contexto de negócio, formando-se os módulos dos sistemas. O produto desta fase são os diagramas de atividades que modelam o funcionamento dos serviços e o diagrama de arquitetura lógica de serviços.

Na fase de Modelagem de Casos de Uso são utilizados os serviços identificados na fase anterior, onde é definido o fluxo de eventos para cada função de serviço, são especificadas as entradas e saídas de dados, as pré-condições, os requisitos de segurança e as dependências. O produto desta fase é a descrição dos casos de uso, diagrama de seqüência para alguns serviços, descrição dos requisitos de segurança e uma arquitetura lógica de serviços.

Na fase de Modelagem de Serviços são utilizados os diagramas de seqüência e arquitetura lógica de serviços gerados na fase anterior, estes diagramas geram uma primeira versão para o modelo de análise, o comportamento dos serviços, entradas e saídas. O produto desta fase é o SSD (System Structure Diagram) que é um refinamento da arquitetura lógica dos serviços, mais a execução de cada cenário dos serviços e a validação dos requisitos de segurança.

Na fase de Projeto de Componentes é o fim das atividades de especificação de serviços, são mapeados os serviços em componentes do sistema, onde a arquitetura lógica é transformada em arquitetura do sistema. O produto desta fase é um conjunto de componentes e seus serviços e um modelo do comportamento destes componentes.

As fases Concepção, Construção e Transição, não sofreram alterações devido ao processo proposto para SOA não se aplicar a estas fases, sendo a Elaboração a fase que sofreu maior impacto pelo processo de desenvolvimento proposto.

4.2 DISCIPLINAS RELATIVAS A SERVIÇOS

O serviço, sob o ponto de vista da arquitetura SOA, é uma função de um sistema computacional que é disponibilizado para outro sistema na forma de um serviço. Um serviço deve funcionar de forma independente do estado de outros serviços e deve possuir uma interface bem definida.

As disciplinas descritas a seguir estão destacadas na figura 4.1, de acordo com as definições apresentadas em Grünbauer et al (2004).

4.2.1 IDENTIFICAÇÃO DOS SERVIÇOS

Nesta disciplina os requisitos são divididos e organizados por atores, onde podem ser representados por regras, sistemas ou serviços.

Como primeiro passo, os requisitos são transformados em fluxos de atividades de granularidade suficiente, de forma que cada atividade pode ser atribuída a um ator.

Como segundo passo os atores são atribuídos a cada fluxo de atividade, seja ele o ator que necessita da informação ou que envia a informação. No modelo, cada raia está atribuída ao seu ator, onde os nós representam atividades e as setas o relacionamento.

Na modelagem convencional existem dois tipos de atores, um abstrai uma pessoa real representando um papel, o segundo é representado por sistemas externos que interagem com o sistema a ser desenvolvido. Na modelagem orientada a serviços deve-se adicionar um terceiro ator que será um serviço.

O sistema não será modelado somente pelos atores, mas sim pelo conjunto de atores e serviços que compõe pequenas funcionalidades responsáveis por certo número de atividades. Em uma visão geral, os requisitos funcionais são dados por todas as funções de serviços associadas que correspondem ao ator serviço, de certo modo a construção é dirigida pela usabilidade e funções de serviço.

Na arquitetura orientada a serviços (SOA), os clientes tipicamente podem invocar serviços, que poderão atuar como clientes para outros serviços. No SOA usa-se o termo Ator para designar quer seja ele um cliente ou um serviço (GRÜNBAUER et al., 2004). A figura 4.2 mostra os tipos de atores, sendo: (a) pessoa real, pode ser um usuário que interage com o sistema; (b) sistema, é um outro modulo ou sistema externo; (c) serviço, são as funcionalidades (serviços) que o sistema provê.

Figura 4.2 – Tipos de atores (a) pessoa (b) sistema (c) serviço
(GRÜNBAUER et al., 2004)

A figura 4.3 mostra um exemplo de diagrama de atividade com os respectivos atores e suas atividades, o exemplo se refere a um sistema da indústria automobilística o sistema de *Diagnostico OnBoard.*

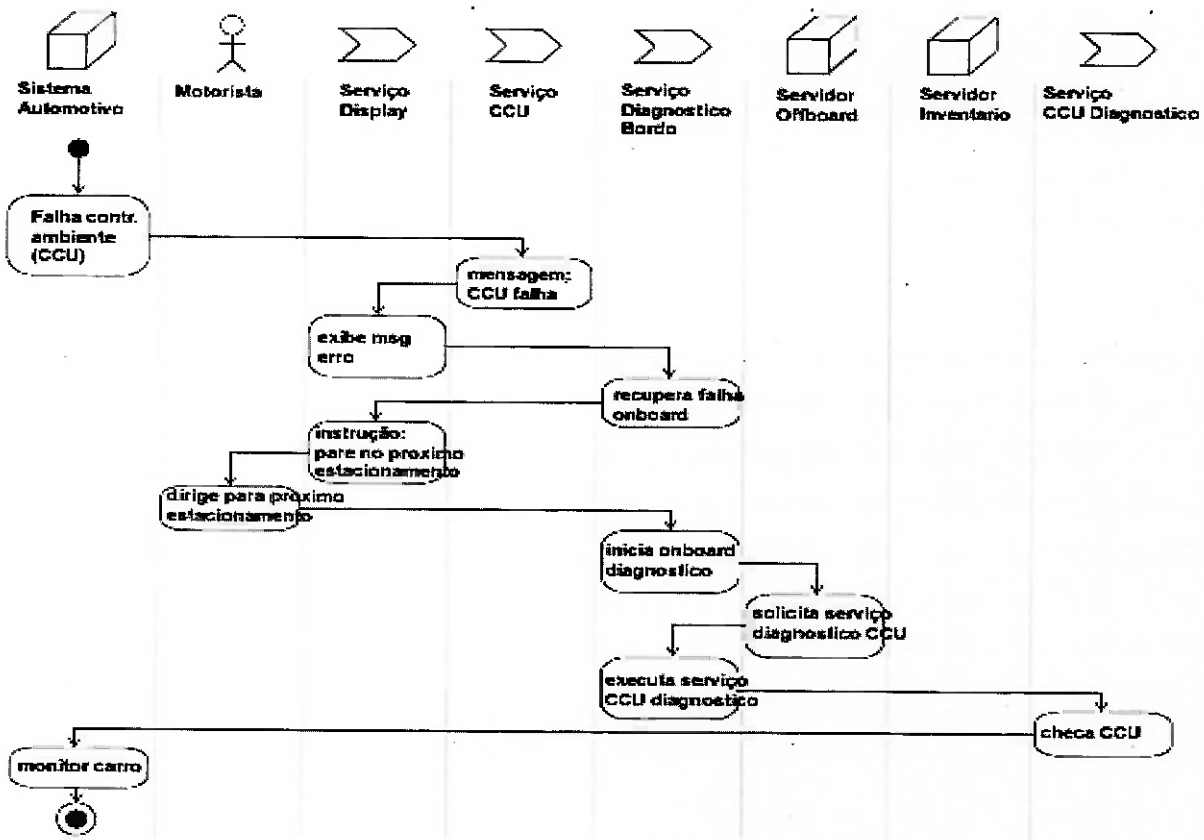

Figura 4.3 – Diagrama Atividade do requisito Onboard Diagnosis
(GRÜNBAUER et al., 2004)

Após a identificação dos serviços e seus relacionamentos como exemplificado na figura 4.3, pode-se chegar a uma arquitetura lógica de serviços, como mostra a figura 4.4. Esta figura apresenta uma visão das funções do sistema e seus relacionamentos, a arquitetura lógica de serviços substitui os diagramas de casos de uso, pois, estes não suportam relações refinadas. Este tipo de diagrama mostra os serviços e seus relacionamentos no diagrama de casos de uso é mostrado o relacionamento dos casos de uso.

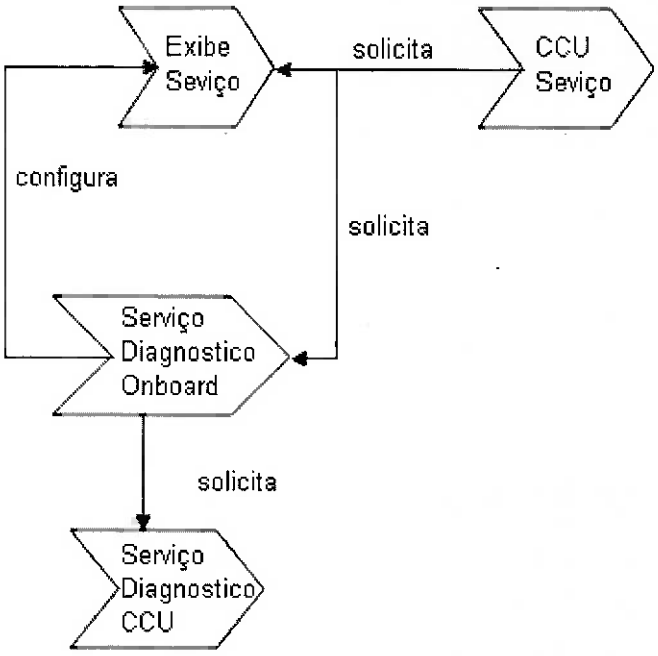

Figura 4.4 – Arquitetura Lógica de Serviços
(GRÜNBAUER et al., 2004a)

4.2.2 MODELAGEM DE CASOS DE USO

A modelagem orientada a objetos, tem como idéia básica obter a iteração do usuário com o sistema já nas primeiras disciplinas do desenvolvimento. Desta forma, proporciona um direcionamento de todo o desenvolvimento; pois através dos mesmos é possível identificar classes, subsistemas, interfaces e casos de teste. Porém, os casos de uso não cobrem aspectos da modelagem orientada a serviços, então as seguintes particularidades devem ser consideradas:

- Não existe a necessidade de um modelo comum de domínio, que é trabalhado na modelagem de negócio e refinado na modelagem de caso de uso, os serviços tratam somente o subconjunto de objetos do sistema, estes necessitam para armazenar informações dos serviços e como entrada e saída de serviços.
- Os atributos de segurança devem ser considerados já nesta disciplina, devem ser descritos cenários onde possam ser identificados possíveis ameaças de segurança. Desse modo, os atributos como, confidencialidade, autenticidade, integridade, disponibilidade, devem ser analisados.

- Uma seção como, serviços envolvidos, destaca os serviços necessários para atender os casos de uso. É utilizada para identificar os relacionamentos dos serviços do sistema e como eles se relacionam, o que pode ser utilizado para identificar problemas de interação já no começo do desenvolvimento.
- Um diagrama de seqüência para uma primeira análise é criado com base na descrição dos casos de uso.

Em resumo, os seguintes passos devem ser tomados na modelagem de casos de uso SOA, o modelo de objeto não é construído, uma vez que na modelagem orientada a serviços não se tem este modelo.

1. Elaborar descrição textual e estruturada dos casos de uso
2. Adicionar atributos de segurança na descrição dos casos de uso
3. Descrever nos casos de uso os serviços relacionados
4. Elaborar diagramas de seqüência dos casos de uso

A tabela 4.1 mostra a descrição do caso de uso do serviço *Onboard Diagnostic Service*, de acordo com o diagrama de atividade da figura 4.3.

Tabela 4.1 – Descrição parcial do caso de uso Onboard Diagnostic Service
(GRÜNBAUER et al., 2004)

Campo	Descrição
Caso Uso	Diagnostico On Board
Atores	Motorista, Sistema Automotivo, Servidor Offboard, Serviço CCU, Serviço Diagnostico Onboard, Serviço Exibir, Serviço Diagnostico CCU
Pré-condição	Motor ligado, Freio de Mão puxado
Fluxo Principal	1. Serviço Diagnostico Onboard checa causa do erro
	2. Informa motorista
	3. Baixa e carrega Serviço diagnostico CCU do servidor Offboard
	4. Inicia Serviço diagnostico CCU
Fluxo Alternativo	(a) Causa do erro não pode ser detectada
	(b) Erro pode ser corrigido pelo Serviço diagnostico CCU, download não necessario
	(c) Não foi possivel executar download
	(d) Finalizado pelo motorista, reinicio posterior
Entradas	Mensagem de falha na CCU, dados de teste do sistema, mensagem de confirmação pelo motorista depois do aviso, download do Serviço diagnostico CCU
Saidas	Mensagem de status para motorista, solicitação de diagnostico, solicitação download, sinal de inicio para o serviço diagnostico CCU
Diretivas de Segurança	Confidencialidade: sem perigo, Autehticidade: sem perigo, Integridade: para garantir comunicação interna, Sem Rejeição: sem perigo, Disponibilidade: possibilidade de conectar Internet, não critico
Serviços envolvidos	Serviço Exibir, Serviço CCU, Serviço diagnostico CCU

A figura 4.5 mostra o diagrama de seqüência correspondente ao caso de uso descrito na tabela 4.1. Diferente da representação do diagrama de seqüência apresentado em (BOOCH, RUMBAUGH, JACOBSON, 2006) , este mostra a iteração entre todos os atores da Orientação a Serviços, presentes no sistema em questão.

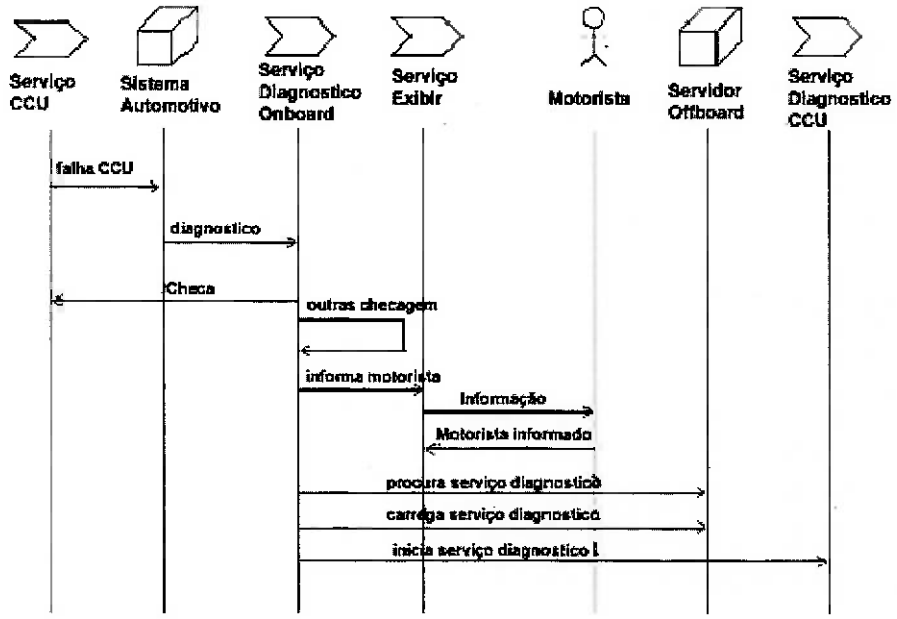

Figura 4.5 – Diagrama Seqüência Onboard Diagnostic Service
(GRÜNBAUER et al., 2004)

4.2.3 MODELAGEM DE SERVIÇOS

Na fase de modelagem de serviços é utilizado o diagrama de seqüência e a descrição dos casos de uso como base para uma análise mais detalhada. Além disso, também é utilizado o diagrama de arquitetura lógica de serviços, criado na fase de identificação de serviços. É gerado um modelo de análise que contém os três tipos de atores descritos anteriormente e a interação entre eles.

A figura 4.6 mostra a representação de um serviço, que consiste de interfaces de entrada e saída e seus tipos. O comportamento dos serviços é parcialmente descrito através das interfaces de entrada e saída, porém são especificados somente os serviços mais relevantes neste modelo. Os requisitos de segurança são concretizados em termos.

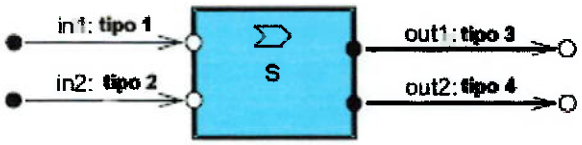

Figura 4.6 – Serviço S (GRÜNBAUER et al., 2004)

Este modelo prove uma base para verificação de consistência, do comportamento das propriedades de proteção e segurança, da geração de casos de teste, da geração de monitor de código e desenvolvimento de componentes. É possível identificar todos os serviços do sistema e seus relacionamentos através de suas entradas e saídas.

A figura 4.7, mostra o diagrama SSD (*System Structure Diagram*), da execução do cenário correspondente ao diagrama de seqüência. Este tipo de diagrama é similar ao diagrama de colaboração onde mostra a estrutura do sistema e suas interfaces.

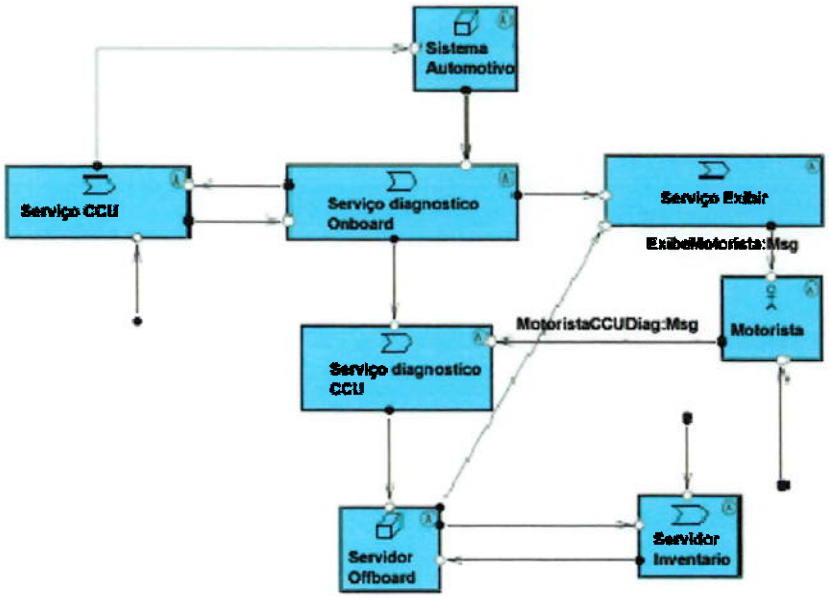

Figura 4.7 – SSD do Onboard Diagnostic (GRÜNBAUER et al., 2004)

4.2.4 PROJETO DE COMPONENTES

A arquitetura lógica de serviços provê uma visão funcional do sistema, enquanto que no projeto de componentes não existe nenhuma restrição estrutural que deva ser seguida. Os serviços podem ser mapeados para uma ou até mesmo uma variedade de arquiteturas de componentes.

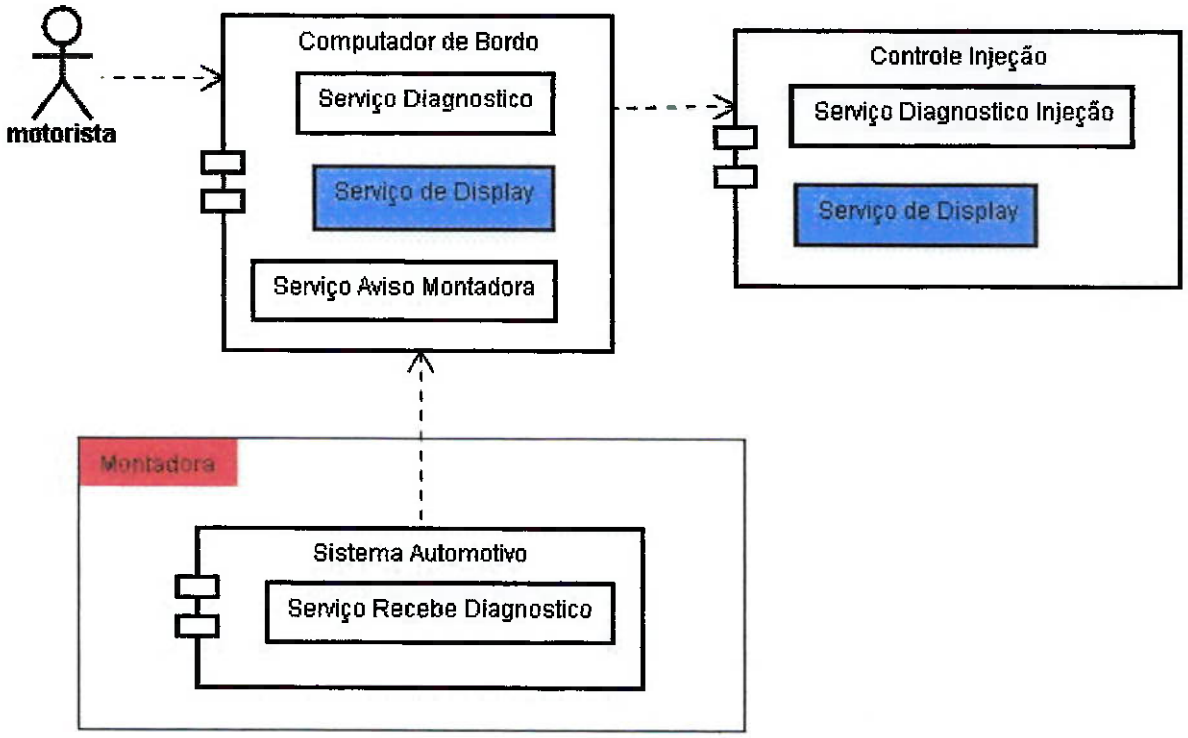

Figura 4.8 – Exemplo de Diagrama de Componentes

4.3 AVALIAÇÃO DA PROPOSTA

O processo proposto por (GRÜNBAUER et al., 2004) é bem similar ao UP, pois possui as mesmas fases. Segundo a proposta, na fase de Elaboração as disciplinas de Análise e Projeto são as que têm maior número de alterações, pois nelas são introduzidas algumas disciplinas que não são presentes no UP. Sendo essas, a Identificação dos Serviços, a Modelagem dos Serviços, além disso, são também adaptadas outras, tais como, Modelagem de Casos de Uso.

Na disciplina Identificação de Serviços é construído um diagrama de atividade antes mesmo da descrição dos casos de uso. Baseando-se na lista de requisitos, são identificados os atores para a construção dos diagramas de atividade, onde para cada requisito é construído um diagrama de atividade de alto nível, mas que seja possível identificar as atividades de cada ator. Nesta disciplina é introduzido um novo ator que representa um Serviço, que representa uma funcionalidade do sistema. Assim como no UP pode-se iniciar com os requisitos de maior risco na primeira iteração e assim, consecutivamente até o fim dos requisitos. O diagrama de

Arquitetura Lógica de Serviços substitui o diagrama de casos de uso, uma vez que este não permite representar um detalhamento maior dos relacionamentos dos serviços que são necessários na modelagem orientada a serviços.

Fazendo uma comparação, pode-se concluir que a Arquitetura lógica de serviços substitui o diagrama de domínio, pois se tem uma visão geral dos serviços e de seus relacionamentos e, o diagrama de Atividades substitui o diagrama de casos de uso. Nesta disciplina são gerados os seguintes artefatos: Diagrama de Atividade, Diagrama de Arquitetura Lógica de Serviços, Lista de Atores, Lista de Requisitos.

Na Modelagem de Casos de Uso, o que altera é a descrição dos casos de uso, onde devem ser identificados os dados e as mensagens de Entrada e Saída, atributos de Segurança e os Serviços envolvidos no caso de uso, tudo isto com base nos diagramas de atividade. Para formalizar a descrição dos casos de uso, são construídos diagramas de seqüência. Nesta disciplina são gerados os seguintes artefatos: Descrição de Casos de Uso, Diagrama de Seqüência.

Na Modelagem de Serviços, diferente do UP ao invés de se criar um diagrama de domínio com as principais classes identificadas, é criado o diagrama SSD *System Structure Diagram*, com base nos artefatos gerados nas disciplinas anteriores e levando-se em conta os atributos de segurança. Nesta disciplina são gerados os seguintes artefatos: SSD *System Structure Diagram*.

Uma vez que os serviços são constituídos e modelados, tendo como último modelo o SSD, que pode ser considerado como base para o modelo da arquitetura do sistema. Os próximos passos do processo de desenvolvimento de software se seguem como no UP, ou seja, o próximo passo é a Construção do que foi modelado na Elaboração. A arquitetura de sistema pode ser mapeada para componentes, ou seja, cada serviço pode compor um ou mais componentes e vice-versa, prosseguindo-se então para as próximas fases, Construção e Transição.

Seguindo o modelo iterativo e incremental, na próxima iteração novos requisitos são transformados em diagramas de atividades e novas funcionalidades são adicionadas a serviços existentes ou novos são criados, onde estes devem passar pelo mesmo

processo na Modelagem de casos de uso, na Modelagem de serviços, sendo mapeados para componentes. Então, prosseguindo para as fases de Construção e Transição, iniciando-se uma nova iteração, até que todos os requisitos sejam atendidos.

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma proposta de um processo de desenvolvimento de software utilizando SOA (GRÜNBAUER et al., 2004), destacando as disciplinas de Análise e Projeto, onde foram apresentadas todas as atividades envolvidas e seus artefatos.

Por fim, foi feita uma avaliação desta proposta comparando-se com o UP apresentado não capítulo 3 deste trabalho. Também foram destacados os artefatos gerados como resultado destas disciplinas.

Pela conclusão da avaliação, a fase de Elaboração é a mais impactada pelo SOA, onde são introduzidas e modificadas disciplinas tais como: Identificação de Serviços, Modelagem de Serviços, Modelagem de Casos de Uso, Projeto de Componentes. Com esta avaliação e os conceitos apresentados nos capítulos anteriores, serão a base para toda a proposta a ser apresentada no próximo capítulo.

5. ESTUDO DE CASO

5.1 CONSIDERAÇÕES INICIAIS

O objetivo deste capítulo é apresentar uma proposta de adaptação do processo de desenvolvimento de software utilizado pela *Empresa Exemplo* apresentado no capítulo 3, para um processo orientado ao desenvolvimento SOA, com base nos conceitos do capitulo 2 e na proposta apresentada no capítulo 4.

A proposta de adaptação contempla as disciplinas de análise e projeto do processo avaliado (GRÜNBAUER et al., 2004), integrado ao framework adotado pela empresa em questão.

5.2 CICLO DE DESENVOLVIMENTO DE SOFTWARE COM SOA PARA A *EMPRESA EXEMPLO*

5.2.1 A METODOLOGIA

A metodologia da *Empresa Exemplo* é *Dirigido aos Casos de Uso* e a proposta de GRÜNBAUER et al. (2004) é orientada a serviços. Porém, também utiliza casos de uso para descrever os requisitos do sistema, o que diferencia é a existência de uma disciplina antes da especificação de casos de uso, a identificação dos serviços, construída com base na lista de requisitos. Portanto, esta nova disciplina deve ser incluída na metodologia da *Empresa Exemplo* antes da modelagem dos casos de uso, como sugerido no capítulo 4 deste trabalho.

Tanto no UP e quanto na metodologia da *Empresa Exemplo,* o desenvolvimento é centrado na arquitetura. Na proposta de Grünbauer et al., (2004), pode-se notar que também é centrado na arquitetura, porém, uma arquitetura orientada a serviços. É possível destacar que logo no inicio do processo proposto é feita a identificação dos serviços, antes mesmo da modelagem dos casos de uso. Como mencionado no capítulo 3 deste trabalho a *Empresa Exemplo* possui um framework (FRAMEWORKX, 2007) de arquitetura que direciona todos os desenvolvimentos de

software da empresa, este framework não sofrerá uma revisão, pois o mesmo descreve as tecnologias de implementação que serão utilizadas na fase de construção para a criação do sistema modelado, os serviços modelados serão mapeados em componentes na camada de serviços, como mostra a figura 5.1.

Na *Empresa Exemplo* o processo é iterativo e incremental, o mesmo ocorre na proposta de GRÜNBAUER et al. (2004), pois é baseado na lista de requisitos, onde nas primeiras iterações são implementados os requisitos de maior risco e assim, consecutivamente, até finalizar todos os requisitos. Nas fases de Elaboração, Construção e Transição não devem ser impactadas pela proposta, uma vez que o processo proposto atinge basicamente a Análise e Projeto. No Anexo II é apresentada uma proposta do fluxo da metodologia da *Empresa Exemplo* já com a nova disciplina e novos artefatos propostos.

A figura 5.1 apresenta o diagrama da arquitetura definida e utilizada pela *Empresa Exemplo* em seus projetos, esta arquitetura é baseada no modelo MVC (LARMAN, 2004).

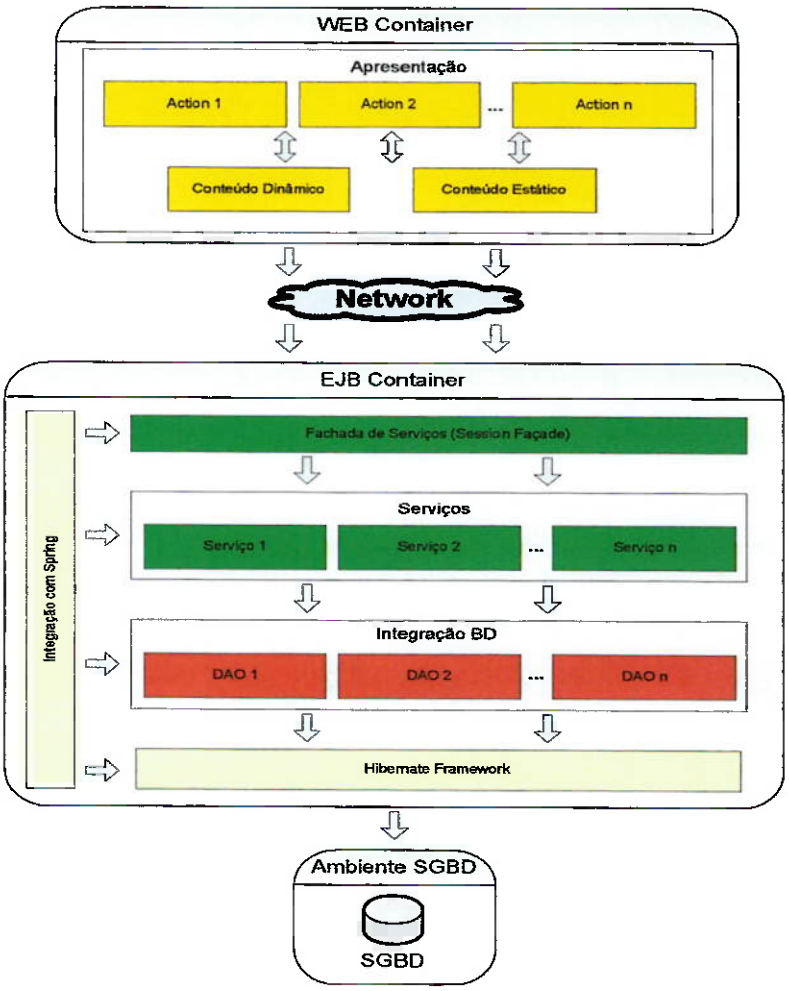

Figura 5.1 – Arquitetura do FRAMEWORKX

5.2.2 IDENTIFICAÇÃO DE SERVIÇOS

Esta disciplina foi introduzida na metodologia, assim como na proposta apresentada no capítulo 4 deste trabalho, antes das disciplinas de Análise e Projeto, conforme Anexo II. Com base nos requisitos são identificados os serviços e agrupados por atores para a criação dos diagramas de atividade e arquitetura lógica de serviços. Nesta fase cada serviço é representado como um ator.

a) ARTEFATOS

Como produto gerado da identificação de serviços tem-se:

- *Lista de Atores:* na orientação a serviços existem três tipos de atores, que são; um abstrai uma pessoa real representando um papel, o segundo é representado por sistemas externos e o terceiro representa um serviço.
- *Diagrama de Atividade:* com base nos requisitos, são criados diagramas de atividade, de forma que cada atividade possa ser atribuída a um ator.

- *Arquitetura Lógica de Serviços:* fornece uma visão das funções do sistema e seus relacionamentos, substitui o diagrama de domínio e casos de uso. Pode ser criado um diagrama por caso de uso ou um do sistema como um todo.

b) PAPÉIS

Os papéis considerados são:

- *Analista de Sistemas:* com base na lista de requisitos, é responsável pela lista de atores incluindo o terceiro tipo de ator que é um serviço.
- *Projetista:* com base na lista de requisitos e atores, o projetista cria os diagramas de atividade e arquitetura lógica de serviços.

c) FLUXO DE TRABALHO

A *figura 5.2* apresenta a relação entre os papéis e suas responsabilidades.

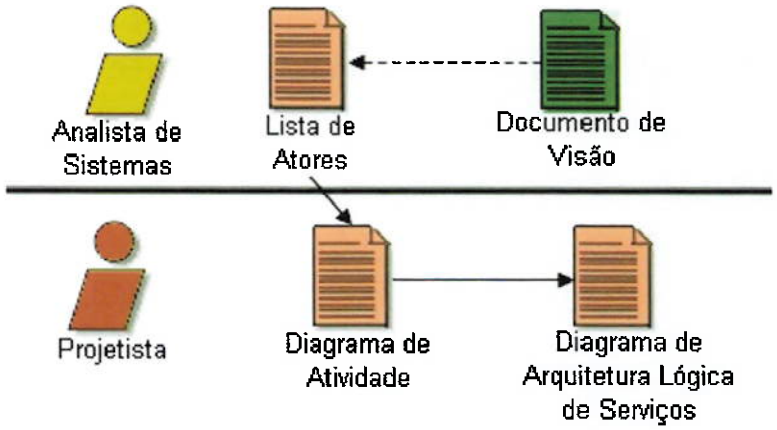

Figura 5.2 – Fluxo de Trabalho da disciplina Identificação de Serviços

O analista de sistemas divide e organiza os requisitos por atores, onde podem ser representados por regras, sistemas ou serviços, gerando assim a lista de atores. Com base na lista de requisitos e atores, o projetista transforma os requisitos em fluxos de atividade de alto nível, o projetista também cria um diagrama de arquitetura lógica de serviços onde é possível ter uma visão geral dos serviços e relacionamentos que compõe o sistema.

5.2.3 ANÁLISE

A disciplina de Análise sofreu algumas alterações, o *Diagrama de Casos de Uso* e *Diagrama de Domínio* foram eliminados e substituídos pela *Arquitetura Lógica de*

Serviços que se encontra na disciplina de Identificação de serviços, e foi incluída a *Especificação de Casos de Uso*. Na Especificação de Casos de Uso foram incluídos alguns atributos relativos a orientação a serviços. Outro artefato incluído na Análise é o *Diagrama de Seqüência* que antes se encontrava na fase de Projeto, os outros artefatos permaneceram inalterados.

a) ARTEFATOS

Como produto gerado do modelo de análise tem-se:

- *Especificação Técnica de Casos de Uso:* utilizando-se do MER e a descrição dos casos de uso. Em português estruturado com o nome de tabelas e campos, são construídos os algoritmos que serão utilizados para a implementação dos casos de uso.
- *Modelo Entidade Relacionamento (MER):* é criado o modelo entidade-relacionamento, com todas as tabelas que participam do sistema.
- *Dicionário de Dados:* descreve todas as tabelas do MER com suas colunas, formatos, descrições e índices.
- *Diagrama de Seqüência:* é criado com base na descrição de casos de uso feita no documento de Especificação de casos de uso, diferente do diagrama de seqüência do OO, este não contém as classes do sistema, e sim os atores de determinado caso de uso que podem ser sistemas, serviços ou pessoas.
- *Especificação de Casos de Uso:* ao invés de ser criado na fase de requisito, passa a ser criado na análise. Também foram inseridos atributos relativos a orientação a serviços.

b) PAPÉIS

Os papéis considerados são:

- *Analista de Sistemas:* é responsável pela criação da Especificação de casos de uso, tendo como base a lista de requisitos e o diagrama de atividade gerado na identificação de serviços.
- *DBA:* é responsável pela criação e integridade do modelo de dados, com base na descrição dos casos de uso, documento de visão.

- *Projetista:* o projetista cria a especificação técnica dos casos de uso, com base no modelo de dados e descrição dos casos de uso e agora também cria o diagrama de seqüência já na análise.

c) FLUXO DE TRABALHO

A figura 5.3 apresenta a relação entre os papéis e suas responsabilidades.

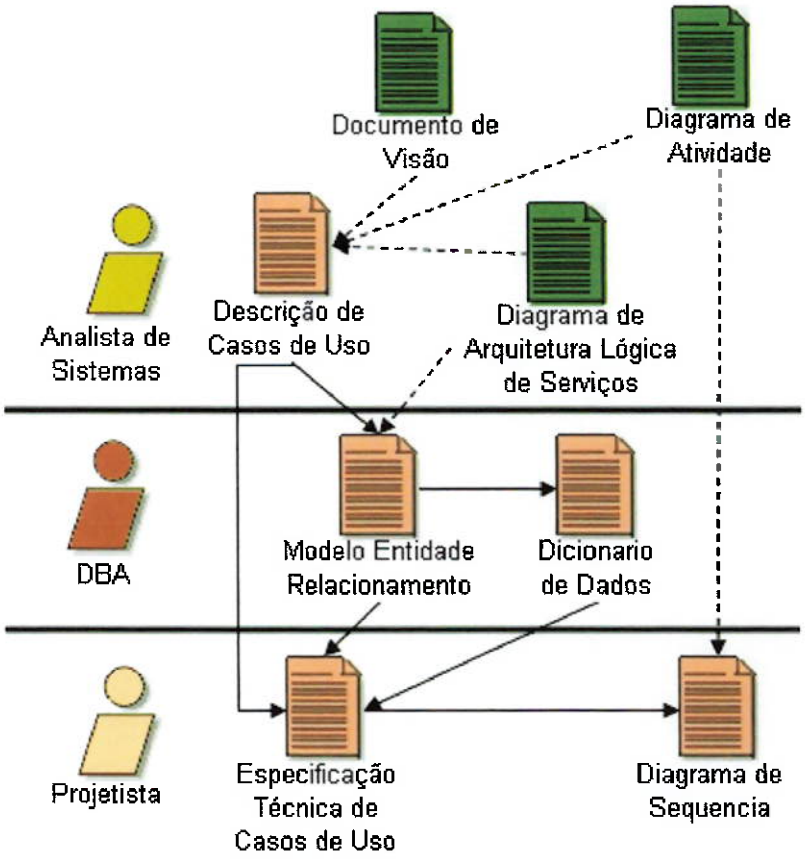

Figura 5.3 – Fluxo de Trabalho da disciplina Análise

O fluxo de trabalho se inicia com o analista de sistemas descrevendo os casos de uso com base nos requisitos, diagrama de atividade e diagrama de arquitetura lógica de serviços que é gerado na disciplina *Identificação de Serviços*. Também deve-se identificar os atributos de segurança e serviços envolvidos para cada caso de uso. Além disso, os diagramas de casos de uso e de domínio foram substituídos pelo *Arquitetura Lógica de serviços*, gerado na *Identificação de Serviços*. Então, o administrador de banco de dados (DBA) cria o modelo de dados e dicionário de dados baseado na descrição dos casos de uso. O projetista inicia a especificação

técnica dos casos de uso utilizando para isto a descrição dos casos de uso e o modelo de dados, também é responsável pela criação do diagrama de seqüência, agora na Análise.

5.2.4 PROJETO

A disciplina de Projeto sofreu algumas alterações; o *Diagrama de Atividade* passa a ser criado na disciplina de *Identificação de Serviços*, pois nesta disciplina ele é necessário para modelar o funcionamento dos serviços e o *Diagrama de Seqüência* passa a ser criado na disciplina de *Análise* para uma análise inicial dos casos de uso. O *Diagrama de Classes* foi eliminado do fluxo da metodologia, mas podendo ser adotado opcionalmente se necessário, nesta disciplina foi introduzido o *Diagrama de Estrutura do Sistema* que fornece uma visão geral dos serviços e seus relacionamentos.

Como citado na seção 3.4.2 deste trabalho, os padrões adotados pela *Empresa Exemplo* serão mantidos, pois se referem à tecnologia de implementação da aplicação a ser desenvolvida.

a) ARTEFATOS

Como produto gerado do modelo de projeto tem-se:

- Diagrama de Implantação, descrevendo infra-estrutura de rede e hardware suas características e conexões, um mapa inicial dos componentes ativos sobre o hardware.
- O Documento de Arquitetura,é composto pela identificação dos principais requisitos do sistema, diagramas de implantação em alto nível e detalhado, diagrama de pacotes e níveis de camada, matriz de qualidade e níveis, FRAMEWORKX.
- Diagrama de Estrutura do Sistema, com base na Arquitetura Lógica de Serviços e nos diagramas de seqüência gerados é criado este diagrama, onde se tem uma visão global de todos os serviços e seus relacionamentos.

b) PAPÉIS

Os papéis considerados são:

- *Arquiteto:* é responsável pela criação do diagrama de implantação e descrição da arquitetura do sistema, também garante que os modelos estejam corretos, consistentes e de fácil compreensão e que por sua vez estejam dentro dos padrões de projeto estabelecidos pela empresa de acordo com o FRAMEWORKX. Para descrever a arquitetura é utilizado como entrada o documento de visão contendo os requisitos do sistema e o diagrama de Estrutura do sistema gerado pelo projetista, a arquitetura descrita deve realizar as funcionalidades especificadas no modelo de casos de uso, requisitos suplementares, e modelo de análise.
- *Projetista:* é responsável pela criação do diagrama de Estrutura do Sistema que será utilizado pelo arquiteto na descrição da arquitetura, este diagrama proporciona uma visão geral do sistema em termos de serviços e seus relacionamentos. Para a criação deste diagrama o projetista utiliza o diagrama de arquitetura lógica de serviços e atividade.

c) FLUXO DE TRABALHO

A figura 5.4 apresenta a relação entre os papéis e suas responsabilidades.

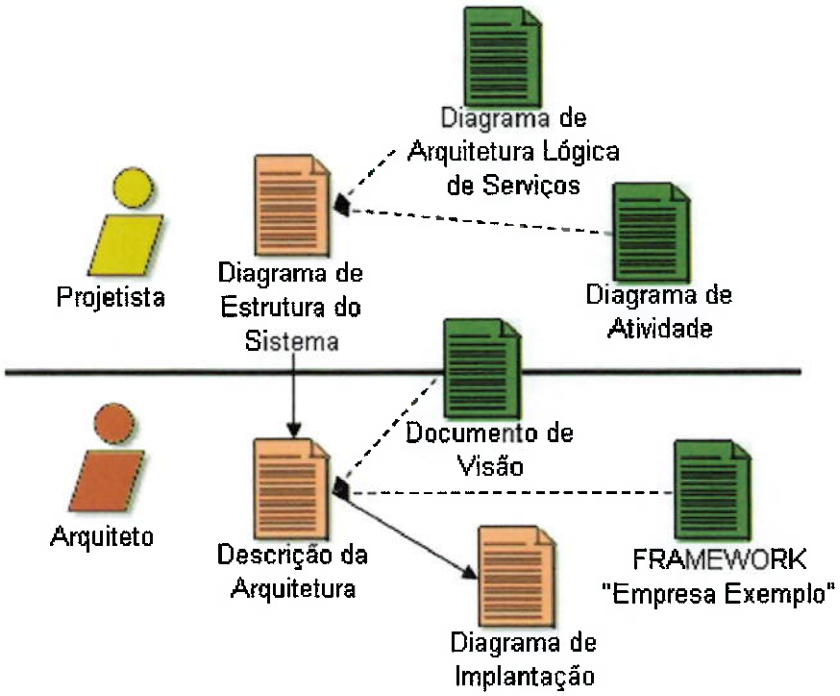

Figura 5.4 – Fluxo de Trabalho da disciplina de Projeto

O fluxo de trabalho se inicia com o projetista através da criação do diagrama de estrutura do sistema que fornece uma visão global de todos os serviços e seus relacionamentos, para isto o projetista utiliza como entrada os diagramas de atividade e arquitetura lógica de serviços. Em seguida, o arquiteto cria o diagrama de implantação, onde é definida a infra-estrutura de rede e hardware, para a maioria dos subsistemas e suas interfaces. Esta definição está contida no documento de descrição da arquitetura. O diagrama de implantação criado deve refletir a infra-estrutura necessária para a realização dos serviços apresentados no diagrama de estrutura do sistema.

5.3 CONSIDERAÇÕES FINAIS

Este capítulo apresenta uma proposta de adaptação da metodologia de desenvolvimento de software da *Empresa Exemplo* para uma nova versão da metodologia, porém incluindo desenvolvimento de software para SOA. Esta proposta de adaptação é realizada com base no trabalho de (GRÜNBAUER et al., 2004), destacando as disciplinas de Análise e Projeto, onde foram apresentadas todas as atividades envolvidas e seus artefatos.

As mudanças propostas são:

- a inclusão da disciplina de identificação de serviços, onde foi introduzido a lista de atores, o diagrama de arquitetura lógica de serviços e o diagrama de atividade,
- na disciplina de análise o que foi alterado é a descrição dos casos de uso que deve conter requisitos de segurança e serviços relacionados a este caso de uso,
- na disciplina de projeto o projetista cria o diagrama de estrutura do sistema que dá uma visão geral dos serviços do sistema e seus relacionamentos, com base neste documento o arquiteto cria o diagrama de implantação e a descrição da arquitetura do sistema, seguindo os padrões do FRAMEWORKX.

6. CONCLUSÃO

6.1 CONSIDERAÇÕES FINAIS

De acordo com a avaliação feita, o impacto do SOA no processo de desenvolvimento, é a inclusão de mais uma disciplina a Identificação de Serviços, que deve ser executada antes da disciplina de análise. Também, alguns artefatos das disciplinas de Análise e Projeto foram eliminados ou sofreram alterações e outros foram introduzidos, tais como, diagrama de seqüência e atividade são modificados para uma visão Orientada a Serviços.

6.2 CONTRIBUIÇÃO

O principal beneficio do SOA é que ele possibilita um alinhamento mais próximo da arquitetura de TI com os requisitos de negócio de uma organização e ao mesmo tempo facilita a automação de alto nível dos processos de negócio. A implementação do SOA permite entregas rápidas, incrementando os benefícios na integração dos projetos devido ao conceito de encapsulamento de componentes de aplicações existentes como serviços, expondo funcionalidades de sistemas de computação antigos como serviços.

6.3 TRABALHOS FUTUROS

A implementação das adaptações propostas neste trabalho no processo da *empresa exemplo* será o próximo passo. Esta implementação se dará de forma gradativa, primeiro deve-se introduzir o conceito do SOA para a equipe de desenvolvimento para que todos estejam com ele em mente, para então iniciar a utilização do SOA em projetos. Como primeiro passo pode-se começar a utilizar o SOA na integração entre sistemas utilizando-se de Web Services, ou seja, fazendo a integração através de serviços. Em paralelo pode-se iniciar a criação de uma versão da metodologia atual conforme o proposto neste trabalho, esta nova metodologia e seus artefatos deverão ser criados e validados através de um processo de um processo gradual

utilizando-se primeiro em projetos menores, até que a mesma esteja estável para ser aplicada a projetos maiores.

REFERÊNCIAS BIBLIOGRÁFICAS

BARRY, D. K., "Web Services and Service-Oriented Architecture: The Savvy Manager's Guide", Morgan Kaufmann Publishers, 2003, capítulo 3

BASS, L.; CLEMENTS, P.; KAZMAN, R. "Software Architecture in Practice", Addison Wesley, 2003, capítulo 2

BIEBERSTEIN, N.; BOSE, S.; WALKER, L.; LYNCH, A. "Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals", IBM Systems Journal, vol44 n.4, 2005.
Disponível em: http://www.research.ibm.com/journal/sj/444/bieberstein.html (acessado em 04/04/07)

BOEHM, B. W.. "A Spiral Model of Software Development and Enhancement",
IEEE, May, 1988
Disponivel em:
http://ieeexplore.ieee.org/search/freesearchresult.jsp?history=yes&queryText=%28%28a+spiral+model+of+software+development+and+enhancement%29%3Cin%3Emetadata%29
(acessado em 27/10/07)

BROWN, A; JOHNSTON, S.; KELLY, K., "Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications", Rational Software Corporation, 2002.
Disponível em: http://www-106.ibm.com/developerWorks/rational/library/4860.html
(acessado em 11/06/07)

CHERBAKOV, L.; GALAMBOS, G.; HARISHANKAR, R.; KALYANA, S.; RACKHAM, G. "Impact of service orientation at the business level", IBM Systems Journal vol44 n.4, 2005.
Disponível em: http://www.research.ibm.com/journal/sj/444/cherbakov.html (acessado em 04/04/07)

CLARK, M., "Web Services Business Strategies and Architectures", Expert Press Ltd, 2002, pg25 a pg36

CONSORTIUM, 2007
Disponivel em: http://www.soa-consortium.org/ (acessado em 29/10/07)

CORBA, OBJECT MANAGEMENT GROUP. "The common object request broker: Architecture and specification", 1995
Disponivel em: http://www.omg.org/technology/documents/formal/corba_2.htm (acessado em 29/10/07)

ERL, T., "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall, 2005, capítulo 9

FRAMEWORK, 2007
Disponivel em: (STRUTS:http://struts.apache.org/, HIBERNATE:http://www.hibernate.org/, SPRING:http://www.springframework.org/) (acessado em 06/10/07)

FRAMEWORKX "Framework Toios v1.4", 2007
(O documento referente a esta referencia não pôde ser anexado a este trabalho por motivo de ser um documento confidencial da empresa)

GRÜNBAUER, J; DEUBLER, M.; POPP, G.; WIMMEL, G.; SALZMANN C. "Towards a Model-Based and Incremental Development Process for Service-based Systems", IASTED Conf. on Software Engineering, : 183-188, 2004.
Disponível em: http://www4.in.tum.de/~gruenbau/papers/iasted_se04.pdf (acessado em 06/12/06)

GRÜNBAUER, J; DEUBLER, M.; POPP, G.; WIMMEL, G.; SALZMANN C. "Tool Supported Development of Service-Based Systems*", Software Engineering Conference, 2004a. 11th Asia-Pacific.
Disponível em: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1371910 (acessado em 07/08/06)

JACOBSON,I; BOOCH,G.; RUMBAUGH,J. "The Unified Software Development Process", Addison-Wesley, 1999, capítulo 8 e 9

BOOCH,G.; RUMBAUGH,J.; JACOBSON,I "UML Guia do Usuário", Editora Campus, 2006

JAVA, 2007
Disponivel em: http://www.java.com/pt_BR/about/ (acessado em 29/10/07)

JINI ARCHITECTURE SPECIFICATION, 2007
Disponivel em: http://www.sun.com/software/jini/specs/jini1.2html/jini-title.html (acessado em 29/10/07)

KARACSONY, K.; TERRY, E. "Implementing SOA PartII – Defining and Developing the Architecture", Qualitydm.com, 2007
Disponível em: http://qualitydm.com/SOAPartII.pdf (acessado em 11/06/07)

KOROTKIY, M.; TOP, J., "Onto-SOA: From Ontology-enabled SOA to Service enabled Ontologies", IEEE, February, 2006
Disponivel em: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1602257 (acessado em 11/06/07)

LARMAN, C., "Applying Uml And Patterns An Introduction To Object-Oriented Analysis (2004), 3Ed.chm", Addison Wesley, 2004, capítulo 13.7

MACHADO, F. N. R. "Tecnologia e Projeto de Data Warehouse", Erica, 2006, pg68 a pg73

MICROSOFT, 2007
http://msdn2.microsoft.com/en-us/library/ms809340.aspx

MOELLER, R.R. "Sarbanes-Oxley and the New Internal Auditing Rules", John Wiley & Sons, Inc., 2004, capítulo 2

OMG, 2007
Disponivel em: http://www.omg.org/ (acessado em 24/11/07)

PAPAZOGLOU, M.P.; YANG, J. "Design Methodology for Web Services and Business Process", CiteSeer.IST Scientific Literature Digital Library, 2002
Disponível em: http://citeseer.ist.psu.edu/581188.html (acessado em 26/06/07)

PRESSMAN, R. S. "Engenharia de Software", McGrawHill, 2002, capítulo 2

STOJANOVIC, Z.; DAHANAYAKE, A. "Service-Oriented Software System Engineering: Challenges and Practices", Idea Group Publishing, 2005 ACM
Disponível em: http://portal.acm.org/citation.cfm?id=1205070 (acessado em 26/06/07)

SUN MICROSYSTEMS, 2007
Disponivel em: http://www.sun.com/ (acessado em 29/10/07)

ZDNET "All roads lead to SOA", ZDNet, 2004
Disponível em: http://news.zdnet.com/2100-3513_22-5415942.html (acessado em 24/11/07)
ZIMMERMANN, O.; KROGDAHL, P.; GEE, C. "Elements of Service-Oriented Analysis and Design", IBM Developer Works, 2004
Disponível em: httphttp://www.ibm.com/developerworks/library/ws-soad1/ (acessado em 24/11/07)

.NET, 2007
Disponivel em: http://www.microsoft.com/brasil/msdn/framework/default.mspx (acessado em 29/10/07)