UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UMA ABORDAGEM PARA ADOÇÃO DOS CONCEITOS DE ARQUITETURA ORIENTADA A SERVIÇOS (SOA) NOS SISTEMAS DE SUPORTE A OPERAÇÃO (OSS)

Emerson Vieira dos Santos

ORIENTADOR: FLÁVIO ELIAS GOMES DE DEUS

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM – 075/2011

BRASÍLIA / DF: 12/2010

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UMA ABORDAGEM PARA ADOÇÃO DOS CONCEITOS DE ARQUITETURA ORIENTADA A SERVIÇOS (SOA) NOS SISTEMAS DE SUPORTE A OPERAÇÃO (OSS)

EMERSON VIEIRA DOS SANTOS

DISSERTAÇÃO DE MESTRADO PROFISSIONALIZANTE SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:

FLÁVIO ELIAS GOMES DE DEUS, Dr., ENE/UNB
(ORIENTADOR)

LUIS FERNANDO RAMOS MOLINARO, Dr., ENE/UNB
(EXAMINADOR INTERNO)

RICARDO STACIARINI PUTTINI, Dr., ENE/UNB
(EXAMINADOR INTERNO)

BRASÍLIA, 03 DE JANEIRO DE 2011.

FICHA CATALOGRÁFICA

SANTOS, EMERSON VIEIRA DOS
UMA ABORDAGEM PARA ADOÇÃO DOS CONCEITOS DE ARQUITETURA ORIENTADA A SERVIÇOS (SOA) NOS SISTEMAS DE SUPORTE A OPERAÇÃO (OSS) [Distrito Federal] 2010. xiv, 71p., 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2010).

Dissertação de Mestrado – Universidade de Brasília, Faculdade de Tecnologia. Departamento de Engenharia Elétrica.

1. OSS 2. SOA
3. TELECOMUNICAÇÕES

I. ENE/FT/UnB. II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

SANTOS, EMERSON VIEIRA DOS (2010). UMA ABORDAGEM PARA ADOÇÃO DOS CONCEITOS DE ARQUITETURA ORIENTADA A SERVIÇOS (SOA) NOS SISTEMAS DE SUPORTE A OPERAÇÃO (OSS). Dissertação de Mestrado, Publicação PPGENE.DM – 075/2011, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 71p.

CESSÃO DE DIREITOS

Emerson Vieira dos Santos
UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica
70910-900
Brasília – DF. - Brasil

"Não é o mais forte da espécie que sobrevive, nem o mais inteligente, mas os mais sensíveis às mudanças"

(Charles Darwin)

AGRADECIMENTOS

A minha família, principalmente minha esposa e meus filhos, que me deram o suporte e inspiração necessários para a conclusão deste trabalho.

Ao meu orientador Prof. Dr. Flávio Elias Gomes de Deus, pelo constante apoio, incentivo, dedicação e amizade, essenciais para o desenvolvimento deste trabalho e para o meu desenvolvimento como pesquisador. Ao Prof. Dr. Luis Fernando Molinaro e ao Prof. Dr. Ricardo Staciarini Puttini pela contribuição essencial para o desenvolvimento do trabalho.

A todos, os meus sinceros agradecimentos.

RESUMO
UMA ABORDAGEM PARA ADOÇÃO DOS CONCEITOS DE ARQUITETURA ORIENTADA A SERVIÇOS (SOA) NOS SISTEMAS DE SUPORTE A OPERAÇÃO (OSS)

Autor: Emerson Vieira dos Santos

Orientador: Flávio Elias Gomes de Deus

Programa de Pós-graduação em Engenharia Elétrica

Brasília, dezembro de 2010

O trabalho descrito nesta dissertação tem como objetivo apresentar uma abordagem para adoção dos conceitos de Arquitetura Orientada a Serviços (SOA) nos Sistemas de Suporte a Operação (OSS) de uma empresa de telecomunicações.

ABSTRACT
AN APPROACH TO ADOPTION OF THE CONCEPTS OF SERVICE ORIENTED ARCHITECTURE (SOA) IN THE OPERATION SUPPORT SYSTEMS (OSS)

Autor: Emerson Vieira dos Santos

Orientador: Flávio Elias Gomes de Deus

Programa de Pós-graduação em Engenharia Elétrica

Brasília, December 2010

The work described in this thesis aims to present an approach for adopting the concepts of Service Oriented Architecture (SOA) in Operation Support Systems (OSS) for a telecommunications company.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

3GPP	3rd Generation Partnership Project
AMATPS	Automatic Message Accounting Tele-Processing System
API	Applications Programming Interface
ASP	Active Server Pages
B2B	Business To Business
B2C	Business to Client
BAM	Business Activity Monitoring
BML	Business Management Level
BPEL	Business Process Execution Language
BPM	Business Processing Modeling
BPMN	Business Process Modeling Notation
BSS	Business Support Systems
CAPEX	Capital Expenditure
CCXML	Call Control XML
CEP	Complex Event Processing
CMMI	Capability Maturity Model Integration
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CRM	Customer relationship Management
CSOBS	Centralized Service Order Bureau System
EADAS	Engineering and Administrative Data Acquisition System
EAI	Enterprise Application Integration
ebXML	Electronic Business using XML
EDI	Electronic Data Interchange
EJB	Enterprise Java Beans
EML	Element Management Level
ESB	Enterprise Service Bus
eTOM	enhanced Telecom Operation Map
FCAPS	Acrônimo para Fault (Falha, Configuration (Configuração), Accounting (Contabilidade), Performance e Secutiry (Segurança).
HSS	Home Subscribe Server
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IDL	Interface Data Language
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
ISO	International Organization for Standardization
ISV	Independent Software Vendor
ITU-T	International Telecommunication Union, seção de padronização
JSP	Java Server Pages
JSR	Java Specification Request
LAN	Local Area Network
MIME	Multipurpose Internet Mail Extensions
MTNM	Multi-Technology Network Management
MTOSI	Multi-Technology Operations System Interface
NCC	Loose class cohesion
NDC	Number of Direct Connections
NFS	Network File System

NGOSS	New Generation Operations Systems and Software
NML	Network Management Level
OASIS	Organization for the Advancement of Structured Information Standards
OMA	Open Mobile Alliance
OPEX	Operational Expenditure
OSS	Operational Support Systems
PDCA	Plan, Do, Check, Action
QoE	Quality of Experience
REST	Representational State Transfer
RMAS	Remote Memory Administration System
ROI	Return On Investment
RPC	Remote Procedure Call
SAML	Security Assertion Markup Language
SCCS	Switching Control Center System
SDP	Service Delivery Platform
SEI	Software Engineering Institute
SES	Service Evaluation System
SID	Shared Information/Data model
SIP	Session Initiation Protocol
SML	Service Management Level
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SS7	Sistema de Sinalização 7
TCC	Tight class cohesion
TCP/IP	Transfer Control Protocol/Internet Protocol
TIRKS	Trunks Integrated Record Keeping System
TMF	TeleManagement Forum
TNA	Technology Neutral Architecture
UDDI	Universal Description Discovery and Integration
UML	Unified Markup Language
VXML	Voice XML
W3C	World Wide Web Consortium
WS	Web Service
WS-CDL	Web Services Choreography Description Language
WSDL	Web Service Description Language
XML	eXtensible Markup Language
YAWL	Yet Another Workflow Language

1. INTRODUÇÃO

Antes da década de 1970 todas as atividades de operação das empresas de telecomunicações eram executadas através de processos administrativos manuais. Não demorou em se perceber que muitas destas atividades poderiam ser automatizadas pelo uso da computação.

Nos cinco anos que se seguiram, as empresas de telefonia criaram um número de sistemas de computadores (aplicações de software) para automação de grande parte destas atividades. Estas ações levaram ao desenvolvimento do Sistema Operacional UNIX[1] e da linguagem C[2] pelos laboratórios da Bell[3].

Vários sistemas foram desenvolvidos nesta época, dentre eles o EADAS (*Engineering and Administrative Data Acquisition System*), CSOBS (*Centralized Service Order Bureau System*), RMAS (*Remote Memory Administration System*), SCCS (*Switching Control Center System*), TIRKS (*Trunks Integrated Record Keeping System*), AMATPS (*Automatic Message Accounting Tele-Processing System*) e o SES (*Service Evaluation System*). Estes sistemas, chamados Sistemas de Suporte a Operação (O*perations Support Systems* - OSS[4]), não comunicavam entre si e a grande maioria das operações necessitava de intervenção manual. Por exemplo, a ordem de serviço criada no sistema de atendimento ao cliente para a ativação de uma linha de assinante não era enviada automaticamente para o sistema de configuração da central telefônica, necessitando a re-digitação de todos os parâmetros pelo técnico.

---

[1] **Unix** é um sistema operacional portável, multitarefa e multiusuário originalmente criado por Ken Thompson, que trabalhava nos Laboratórios Bell da AT&T. A marca **UNIX** é uma propriedade do The Open Group, um consórcio formado por empresas de informática.

[2] **C** é uma linguagem de programação compilada de propósito geral, estruturada, imperativa, procedural , de alto nível, padronizada pela ISO, criada em 1972, por Dennis Ritchie, no AT&T Bell Labs, para desenvolver o sistema operacional Unix (que foi originalmente escrito em Assembly).

[3] **Bell Telephone Laboratories** ou **Bell Labs** era originalmente o braço de pesquisa e de desenvolvimento AT&T dos Estados Unidos, desenvolvendo uma série de tecnologias consideradas revolucionárias desde comutadores telefônicos, cabos de telefone, transistores, LEDs, lasers, a linguagem de programação C e o sistema operativo Unix. http:// *www.**bell**-labs.com/*

[4] **Operations Support Systems** (Sistemas de Suporte a Operação ou OSS) são sistemas informatizados utilizados pelos prestadores de serviços de telecomunicações. O OSS termo mais frequentemente descreve os sistemas de rede que suportam a própria rede de telecomunicações, os processos de manutenção de inventário de rede, fornecimento de serviços, configuração de componentes de rede e gestão de falhas.

1.1. MOTIVAÇÃO E JUSTIFICATIVA

A clara ineficiência destes processos somada ao aumento da concorrência no setor além da grande complexidade e demora na implantação dos novos serviços levou e leva até hoje as empresas de telecomunicações a procurarem alternativas de integração entre os diversos sistemas que compõem o OSS[4] a um custo cada vez menor.

Nesta busca pela integração, simplificação e agilidade na entrega dos serviços as empresas passaram das interfaces ponto-a-ponto, pela abordagem utilizando Integração Corporativa de Aplicações (*Enterprise Application Integration ou EAI*) e agora utilizam ESB[5], acrônimo para *Enterprise Service Bus* (Barramento Corporativo de Serviços, em uma tradução livre), onde será apresentado mais adiante, é um componente fundamental na Arquitetura Orientada Serviços, ou SOA (*Service Oriented Architecture*). A arquitetura orientada a serviços tem como objetivo dar a agilidade e simplicidade necessárias para disponibilização dos serviços de uma empresa de telecomunicações reduzindo o tempo de projeto e concepção de um serviço até a entrega deste para os clientes ou *Time to Market*[6].

Fica evidente que os antigos processos das diversas áreas, mas principalmente dos processos referentes aos OSS, já não conseguem mais atender a esta celeridade e eficiência mandatórias no mercado competitivo de hoje. Segundo o levantamento da *Frost & Sullivan* (2010), com a profusão de dispositivos móveis com acesso à internet, abrem-se diversas oportunidades de negócios para o mercado de telecomunicações, mas há alguns obstáculos no caminho que devem ser superados. Segundo o mesmo levantamento, um dos principais obstáculos é a estrutura atual dos OSS e por isso as empresas estão planejando investir nos próximos anos 47%, em média, de seus orçamentos na melhoria dos OSS.

Os OSS de uma empresa de telecomunicações têm cada vez mais importância estratégia, pois estes sistemas automatizam os processos de operação de áreas fundamentais para configuração, fornecimento, medição, gerência de falhas e

[5] **ESB** (*Enterprise* Service Bus ou Barramento de Serviços Corporativo) consiste em construir uma arquitetura de software que fornece serviços fundamentais para arquiteturas complexas orientada a eventos (*event-driven*) e baseada em padrões de mensagens.
[6] ***Time to Market*** é o tempo de projeto e concepção de um produto ou serviço até a disposição deste para o consumidor final.

desempenho e tarifação dos serviços, diminuindo assim os custos e ainda melhorando a Qualidade de Experiência ou QoE[7] (*Quality of Experience*) do usuário.

Uma das ferramentas que tem se mostrado eficiente no mapeamento, definição e melhoria de processos é a Gestão de Processos de Negócio ou BPM[8] (*Business Process Modeling*) que é o conjunto de conceitos e técnicas que auxiliam na criação de modelos dos processos de negócio. Como será visto adiante este trabalho propõe a utilização desta ferramenta no mapeamento dos processos utilizados no SOA.

Como será mostrado mais adiante no capítulo 3, SOA permite minimizar os problemas de integração entre as aplicações, além de proporcionar maior agilidade na criação de novos serviços e agregar maior valor aos negócios. Portanto migrar os OSS para o SOA é de fundamental importância estratégica para que as empresas de telecomunicações possam atingir os objetivos esperados.

Percebe-se que as empresas de telecomunicações estão convencidas que a migração para SOA é uma conseqüência quase que inevitável, e que a adoção do SOA, principalmente nos OSS, terá um impacto significativo, no sentido positivo, na entrega dos serviços, porém ainda não sabem exatamente como devem proceder a implantação, pois não há nenhum guia ou modelo consagrado como referência para tal.

Segundo Siqueira(2004), embora haja consenso sobre a importância da melhoria permanente dos processos, muitas empresas têm tido dificuldades na definição das estratégias mais adequadas para obter melhorias relevantes e sustentadas. Muitas iniciativas perdem força ao longo do tempo por falta de objetividade e de resultados que justifiquem os investimentos realizados.

Neste trabalho, será proposta uma abordagem que dividirá em cinco fases o processo de adoção dos conceitos de SOA em OSS.

---

[7] **Qualidade de Experiência** (*Quality of Experience* - QoE), também conhecido como "Qualidade de Experiência do Usuário", no contexto de telecomunicações, é um parâmetro que representa o desempenho global de uma rede sob o ponto de vista dos usuários, e que coloca uma perspectiva pessoal de satisfação dos serviços oferecidos pelo provedor.

[8] Gestão de Processos de Negócio ou **BPM** (*Business Process Modeling*) é uma abordagem de gestão focada em alinhar todos os aspectos de uma organização com os desejos e necessidades dos clientes. (Ryan K. L Ko, 2009)

1.1.1. Gestão de serviços e convergência

Tradicionalmente, o foco das operações de Telecom era voltado para a eficiência interna. Devido principalmente à privatização do setor de Telecomunicações, o foco no cliente passou a ser a fundamental. É fato que a qualidade percebida pelo cliente (QoE) é uma das principais preocupações das empresas prestadoras de serviço de telecomunicações.

A busca constante pela melhoria da qualidade técnica dos serviços tem levado as operadoras a maximizar sua eficiência operacional através da automação das atividades que envolvem a entrega do serviço aos clientes. Em um cenário onde a convergência das redes e serviços é a palavra de ordem da vez, fica claro que a gestão eficiente dos serviços torna-se crucial para o sucesso do negócio.

A evolução das redes de computadores propiciou perspectivas no emprego de serviços baseados em redes convergentes (Cunha, 2009). Estas redes disponibilizam novos processos integrados para dados, voz e multimídia, com inovações tecnológicas abordando elementos, processos e interligações de redes (Oliveira, 2008).

Nas décadas posteriores, ocorreram evoluções sem precedentes, nas tecnologias de protocolos de redes de dados, centrais de comutação e nos sistemas de transmissão que culminaram, nos dias de hoje, na convergência tecnológica para transmissão de dados, voz e vídeo (Tanenbaum, 1997).

Os *backbones* das Empresas de Telecomunicações tornaram-se a fundação tecnológica que permitiu a combinação de aplicações de vídeo, voz e missão crítica, culminando com a convergência de tecnologias, serviços e conhecimento/especialização dos profissionais do setor (Duarte, 2009).

Um dos elementos diferenciadores das operadoras está situado na inovação decorrente da convergência digital, entre as tecnologias de redes de comunicação, de processamento e de apresentação das informações. Apesar de a convergência digital existir tecnologicamente desde a década de 60 com o advento do protocolo TCP/IP[9] (RFC[10] 793 e RFC 791), é atualmente que se observa o surgimento de novos modelos

[9] **TCP/IP** – Protocolo de Controle de Transmissão / Protocolo Internet ou *Transmission Control Protocol / Internet Protoco*l, é o conjunto de protocolos de comunicação usado para conectar os computadores na Internet.

[10] **RFC** ou *Request For Comment*s (Requisição para Comentários, em uma tradução livre) são documentos originalmente criados para registrar anotações não oficiais durante o desenvolvimento da ARPANET (Rede que deu origem a Internet), e que desde então se tornaram a fonte de informação padrão no que diz respeito à descrição de métodos, comportamento, pesquisa ou inovação aplicável a Internet e/ou sistemas relacionados.

de negócios e de produtos e serviços convergentes, baseados em novos atributos de valor fornecidos ao consumidor final. Nesta conjuntura de elevada competitividade e incerteza, julga-se relevante que as iniciativas estratégicas relativas à inovação sejam adequadamente geridas, especificamente aquelas referentes à convergência digital, para que se tornem elementos diferenciadores e sustentadores da competitividade empresarial (Frontini, 2008).

A Figura 1.1 mostra um exemplo de solução tecnológica adotada por uma empresa de Telecomunicações em que a convergência da conectividade das redes e serviços utiliza a arquitetura TCP/IP[9].

Neste cenário cada vez mais complexo é fundamental que exista um alinhamento entre o portfólio de serviços ofertados com a estratégia de negócios da empresa e esta estratégia deve levar em conta o dinamismo exigido pelo mercado na disponibilização de serviços de tecnologia, pois se sabe que a redução do *time to market* é um dos maiores desafios das organizações. As soluções utilizadas atualmente já estão mostrando sinais de fadiga assim como os profissionais de tecnologia envolvidos nestes processos. O que se percebe nas empresas de telecomunicações são ações reativas e não proativas ao mercado, pois o grande volume de trabalho despedido na manutenção da estrutura atual não deixa margem para planejamento e melhoria de processos.

Percebe-se que esta convergência, somada as novas tecnologias que estão surgindo, está demandando por novos paradigmas que consigam suportar a complexidade exigida na disponibilização dos novos serviços.

1.2. OBJETIVOS

Neste trabalho serão apresentadas as ferramentas, tecnologias necessárias para a adoção gradativa de uma Arquitetura Orientada a Serviços (SOA) voltada para os Sistemas de Suporte a Operação (OSS) de uma empresa de telecomunicações, aproveitando a estrutura e sistemas atuais, preservando assim o investimento e causando o menor impacto para os clientes.

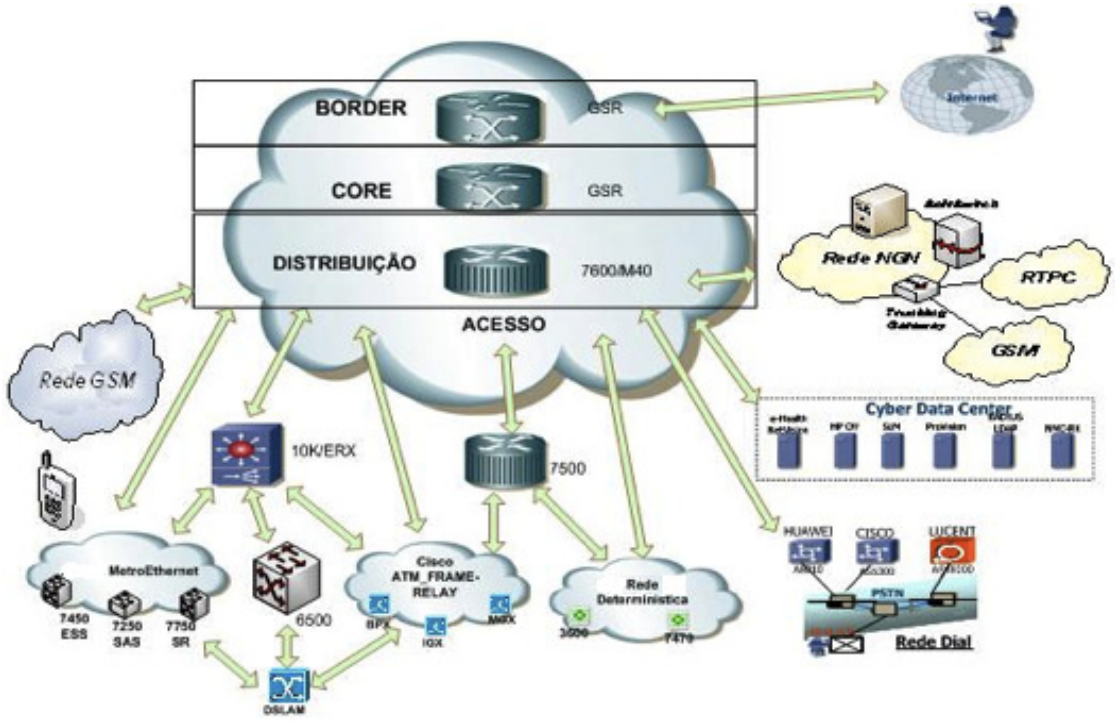

Figura 1.1 - Topologia de convergência tecnológica geralmente adotada pelas empresas de Telecom. Fonte: (Duarte, 2009)

Apesar do foco principal do trabalho ser voltado para Sistemas de Suporte a Operação utilizados por empresas de telecomunicações, nada impede que seja utilizado, com algumas adaptações, em OSS de empresas de outros segmentos.

1.3. METODOLOGIA E ORGANIZAÇÃO DA DISSERTAÇÃO

Este trabalho procurou utilizar as ferramentas, metodologias e tecnologias já consagradas, tais como BPM, PDCA[11], XML[12] e outras, para a transição dos OSS para SOA ocorra em cinco fases distintas, de forma gradativa e com menor impacto possível.

Será mostrada no capítulo 2 uma visão geral do OSS, bem como o histórico e evolução das arquiteturas. No capítulo 3 os conceitos, idéias, vantagens e benefícios do

[11] **PDCA** (do inglês, *Plan, Do, Check, Act*) é um processo de quatro passos iterativos de resolução de problemas, tipicamente usado na melhoria de processos de negócios.

[12] A Linguagem de Marcação Extensível ou **XML** (*eXtensible Markup Language*) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais. Esta linguagem permite descrever e entregar dados estruturados de qualquer aplicação.

SOA serão colocados além do conceito de serviço juntamente com os princípios fundamentais e as principais tecnologias utilizadas na implementação de SOA. No capítulo 4, será explanada a entrega de serviços de telecomunicações, o conceito de Plataforma de Entrega de Serviços ou *Service Delivery Platform*(SDP) bem como a definição de seus elementos básicos. Neste mesmo capítulo, será apresentado o Modelo de Maturidade de Capacidade e Integração ou *Capability Maturity Model Integration* (CMMI) como modelo que servirá como referência para avaliação da maturidade na implantação de SOA. Na última seção deste capítulo será apresentado um guia, que irá propor a migração das funcionalidades dos OSS para SOA em cinco fases. O capítulo cinco apresenta as conclusões e sugestões de trabalhos futuros.

2. SISTEMAS DE SUPORTE A OPERAÇÃO (OPERATIONS SUPPORT SYSTEMS - OSS)

Neste capítulo será mostrado que os Sistemas de Suporte a Operação (OSS) tem um papel fundamental para o funcionamento dos serviços, por isso constituem uma parte nevrálgica em uma empresa de telecomunicações. Não é surpresa que os OSS têm sido alvos prioritários dos investimentos. Segundo Fonseca (2009), as operadoras gastam entre 40% e 70% do orçamento de Tecnologia da Informação (TI) para efetuarem a integração das aplicações que transitam pela rede.

2.1. CONCEITOS FUNDAMENTAIS

Os Sistemas de Suporte à Operação – OSS (*Operational Support Systems*) foram desenvolvidos para automatizar as operações que envolvem desde o aprovisionamento dos elementos de rede até a bilhetagem necessária para o cálculo da fatura dos clientes. Esta automação é fundamental para garantirmos a qualidade de entrega do serviço, pois geralmente as empresas de Telecom possuem milhões de clientes e operacionalizar estas atividades manualmente seria economicamente inviável.

Além disso, uma arquitetura OSS deve ser preferencialmente, independente de fornecedor, tecnologia e aplicações, e com uma capacidade de integração que garantirá às empresas de Telecom que não fiquem sujeitos a implantações de atualizações onerosas. Geralmente, mais da metade dos custos de uma operadora advém das operações, é um afinado ecossistema (Light Reading, 2006). No passado, os OSS gerenciavam a rede de infra-estrutura. No novo cenário, os OSS devem gerenciar os clientes, serviços e os relacionamentos entre eles além da rede de infraestrutura (Misra, 2004).

Portanto, os objetivos principais dos OSS são aumentar a produtividade, reduzir os custos operacionais da empresa e entregar os serviços com qualidade. As operadoras são cada vez mais pressionadas para novas estruturas de redes e suportar novos serviços.

Podem-se categorizar as principais áreas componentes do OSS conforme a Figura 2.1.

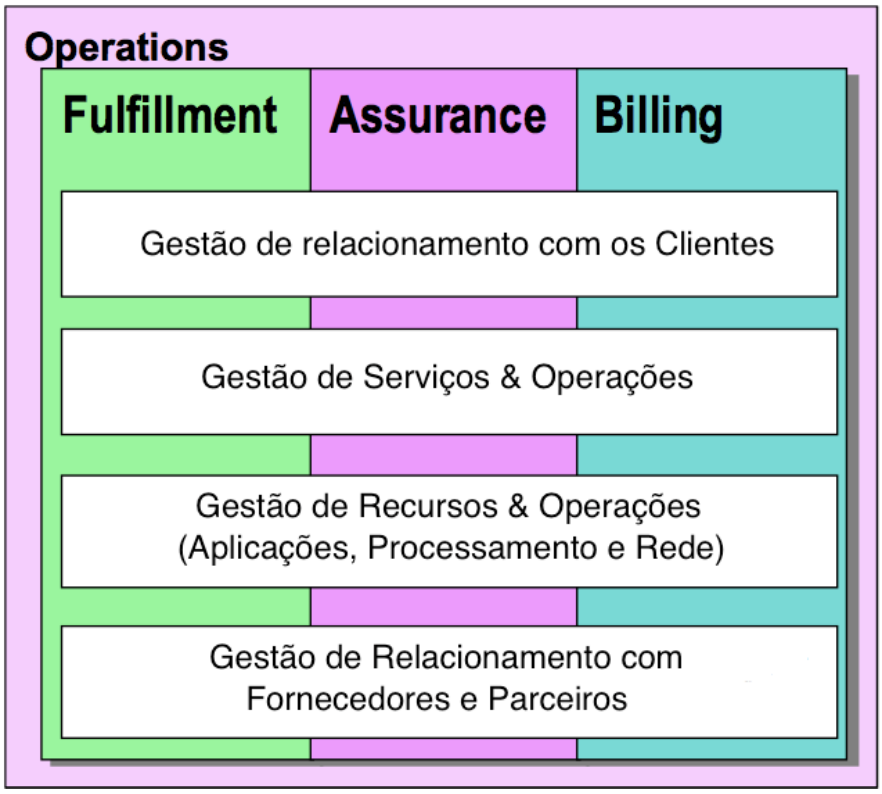

Figura 2.1:Principais áreas atendidas pelos OSS. Fonte: eTOM/NGOSS - TMF)

Na figura vemos as áreas fundamentais de uma arquitetura típica de um OSS. Mantivemos os nomes das componentes principais em inglês devido a estes termos terem a sua consagração global nesta língua:

- *Order* ***Fulfillment*** – Em uma tradução livre: atendimento fim a fim das ordens de serviço. Incluem as seguintes subáreas: Gestão de Ordem de Serviços, Aprovisionamento de Serviços e Gestão de Inventário;
- *Service* ***Assurance*** – Garantia de Serviço: é a área que cuia com que os serviços serão entregues aos clientes com qualidade esperada pelos contratos de SLA[13] (*Service Level Agreement* ou Acordo de Nível de Serviço). As seguintes áreas fazem parte do *Service Assurance*: Gestão de Falhas e Problemas; Gestão de Rede e Desempenho.

[13] Um Acordo de Nível de Serviço (freqüentemente abreviado como **SLA**) é parte de um contrato de serviço onde o nível de serviço é definido formalmente. Na prática, o termo SLA é por vezes utilizado para se referir ao prazo de entrega contratados (do serviço) ou desempenho.

- ***Billing*** – É o processamento e compilação de informações que permitem a cobrança pelo uso da rede, transações de recursos e taxas de acesso dos sistemas e serviços. Os sistemas de Mediação coletam os dados de uso da rede a partir de elementos da rede e converte em informações que podem ser faturáveis.

2.2. HISTÓRICO E EVOLUÇÃO

A importância estratégica dos OSS ficou clara para as empresas de Telecom que tem investido cada vez mais nestes sistemas, sempre buscando o máximo possível de automação com o mínimo possível de intervenções humanas, o que nitidamente proporciona uma eficiência maior na prestação dos serviços traduzindo-se em maior competitividade para o negócio.

Dada esta importância, é natural que os OSS tenham passado por evoluções significativas ao longo do tempo. No passado, os OSS gerenciavam a rede de infra-estrutura. Neste novo cenário, os OSS devem gerenciar os clientes, serviços e os relacionamentos entre eles e a rede de infra-estrutura (Misra, 2004). Esta nova visão de OSS vai ao encontro de certos aspectos da plataforma de entrega de serviços (SDP – *Service Delivery Platform*), entretanto existe certa lógica de como os OSS e a plataforma SDP trabalharão juntos (Light Reading, 2006). Na seção 4.1.2 o conceito do SDP será melhor detalhado.

É esperado dos OSS que eles consigam responder rapidamente às mudanças que ocorrem no ambiente de negócio, que consigam diminuir o tempo para se colocar no mercado novos serviços (*time to market*), que simplifiquem a arquitetura através da utilização de uma única interface de integração e que estejam prontos para as redes da próxima geração com uma arquitetura a "prova de futuro". Tudo isso economizando OPEX[14] e CAPEX[15] através da uma infra-estrutura comum. Dadas estas necessidades, pode-se concluir que integração com a gestão dos processos de negócio é a chave para esta transformação.

[14] **OPEX** é a sigla da expressão inglesa *Operational Expenditure* (em português, Despesas Operacionais) e que designa o montante de dinheiro utilizado para manter em operação os bens de capital de uma determinada empresa, nomeadamente os equipamentos, softwares e instalações, ou seja, manter em funcionamento o negócio.

[15] **CAPEX** é a sigla da expressão inglesa *Capital Expenditure* (em português, despesas de capital ou investimento em bens de capital) e que designa o montante de dinheiro despendido na aquisição (ou introdução de melhorias) de bens de capital de uma determinada empresa.

2.3. ARQUITETURAS

Nos anos 90 o ITU-T[16] definiu uma arquitetura de OSS no seu modelo TMN [17](*Telecommunications Management Network*) que estabeleceu quatro camadas: *Business Management Layer* (BML) , *Service Management Level*(SML), *Network Management Level* (NML) e *Element Management Level* (EML).

O *Business Management layer* – BML (ou Camada de Gerência do Negócio) inclui as funções relacionadas com os aspectos do negócio, analisa tendências e questões de qualidade, por exemplo, ou para fornecer uma base para o faturamento e outros relatórios financeiros.

O *Service Management layer* – SML (ou Camada de Gerência de Serviço) cuida dos serviços na rede: definição, administração e cobrança dos serviços;

O *Network Management layer* – NML (Camada de Gerência de Rede) distribui os recursos de rede, executa tarefas de configuração, controle e supervisão da rede.

Já o *Element Management layer* – EML (Camada Gerência de Elementos) cuida dos elementos individuais de rede, incluindo o gerenciamento de alarmes, gestão da informação, backup, loggind e manutenção de hardware e software.

Mais tarde a ITU-T adotou o modelo de framework proposto pela ISO[18] chamado FCAPS que é um acrônimo para *Fault* (Falha), *Configuration* (Configuração), *Accounting* (Contabilidade), *Performance (*Desempenho*)* e *Secutiry* (Segurança).

Um grande problema de rede e gestão de serviços é a capacidade de gerir e controlar os elementos de rede de acesso e de redes *core* (ou rede principal). Historicamente, muitos esforços foram despendidos nos fóruns de normalização (ITU-T, 3GPP[19]), a fim de definir o protocolo padrão para gerenciamento de rede, mas sem sucesso e resultados práticos. Em contrapartida o protocolo SNMP (*Simple Network*

---

[16] O **ITU-T** é a seção de Padronização da área de Telecomunicações do **ITU** - União Internacional de Telecomunicações (*International Telecommunication Union*), que é sediado em Genebra, Suíça. Antes de 1992, o ITU-T era conhecido como **CCITT** (Comitê Consultivo Internacional de Telefonia e Telegrafia).

[17] O termo **TMN** foi introduzido pelo **ITU-T** (anteriormente CCITT) como uma abreviação para *Telecommunications Management Network*" ou Rede de Gerenciamento de Telecomunicações.

[18] **ISO** (*International Organization for Standardization* ou Organização Internacional para Padronização) é uma organização não governamental que congrega grêmios de padronização/normalização em 163 países, um membro por país, com um Secretariado Central em Genebra, na Suíça, que coordena o sistema.

[19] O **3GPP** (*3rd Generation Partnership Project*, ou Projeto de Parceria de Terceira Geração) é uma colaboração entre grupos de associações de telecomunicações, para aplicar globalmente a especificação do sistema de telefonia móvel de terceira geração (3G), no âmbito do projeto *International Mobile Telecommunications*-2000 da União Internacional das Telecomunicações (ITU[16]).

Management Protocol ou Protocolo Simples de Gerenciamento de Rede) do IETF[20] (*Internet Engineering Task Force ou* Força Tarefa para Engenharia da Internet) tornou-se o padrão de fato para a Internet e gestão de telecomunicações, no nível de comunicação EML-NML.

O TeleManagement Fórum(TMF) é uma organização internacional de prestadores de serviços de comunicações e fornecedores da indústria de comunicações. Enquanto os OSS são geralmente dominados por tecnologias exclusivas e personalizadas, o TMF é considerado como a fonte mais autorizada para os padrões e *frameworks* de OSS. O TMF tem sido ativo em prover um quadro e um fórum de discussão para os avanços dos OSS e BSS[21].

O último estágio dos trabalhos em arquitetura OSS veio com o programa NGOSS[22] da TMF, que foi criada em 2000. Isto estabeleceu um conjunto de princípios que uma integração que os OSS devem adotar, juntamente com um conjunto de modelos que fornecem abordagens padronizadas. Os modelos incluem um modelo de informação (*Shared Information/Data model* - SID), um modelo de processo (*enhanced Telecom Operation Map* eTOM), um modelo de aplicação (*Telecom Applications Map* - TAM), uma arquitetura (*Technology Neutral Architecture* - TNA) e um modelo de ciclo de vida. A TMF descreve NGOSS como uma arquitetura que é de baixo acoplamento (*loosely coupled*), distribuída e baseada em componentes juntamente com o funcionamento de componentes de aplicação em que um provedor de serviços de comunicações pode ser executado.

Os componentes interagem através de um veículo de comunicação comum (utilizando uma infra-estrutura de troca de informações, por exemplo, EAI[23], ou *Web Services*[24]). O comportamento pode ser controlado através do uso de gerenciamento de

---

[20] **IETF** é a sigla para ***Internet Engineering Task Force*** e se refere a uma instituição que desenvolve e promove as normas de Internet. O site do **IETF** é http://www.ietf.org e nele encontram-se as especificações de diversos protocolos associados à implementação e operação da Internet. Os documentos de especificação são chamados de RFC´s (*Request for Comments*).

[21] O **BBS** (*Business Support Systems* ou Sistemas de Suporte de Negócios) são os componentes que uma operadora de telefonia ou de telecomunicações usa para executar suas operações de negócio voltadas para o cliente.

[22] **NGOSS** (*New Generation Operations Systems e Software* ou Nova Geração de Software e Sistemas de Operações) é o programa do TeleManagement Fórum que visa fornecer meios para ajudar a comunicação dos provedores de serviços na gestão do seu negócio.

[23] **EAI** é a sigla para *Enterprise Application Integration* (Integração de Aplicações Corporativas) é definida como o uso de princípios de arquitetura de software e sistemas para integrar um conjunto de aplicações da empresa.

[24] Os ***Web Services*** (Serviços Web) são normalmente interfaces de programação de aplicativos (API**Erro! Indicador não definido.**) ou APIs Web que são acessadas normalmente via HTTP (*Hypertext Transfer*

processos e/ou gestão da política de orquestração das funcionalidades fornecidas pelos serviços oferecidos pelos componentes.

O foco inicial do trabalho do TMF no NGOSS foi na construção de modelos de referência para apoiar a visão das partes interessadas no processo de negócio, informação e interação entre aplicativos, correndo em paralelo atividades que foram apoiadas uma visão das partes interessadas na implementação de especificações de interface para fornecer acesso aos OSS (principalmente MTNM[25] - *Multi-Technology Network Management*). O trabalho MTNM evoluiu para um conjunto de *Web Service* chamado MTOSI[26] (*Multi-Technology Operations System Interface*). Mais recentemente, o OSS através da Java Initiative (OSS/J[27]) se juntou ao TMF para fornecer APIs (*Application Programming Interface*) BSS/OSS baseado em NGOSS APIs.

O programa NGOSS contempla tanto uma arquitetura recomendada para os Sistemas de Suporte à Operação quanto a Metodologia para implantá-la (Faro, 2007). Esta evolução da arquitetura dos OSS voltada para o negócio vem de encontro com os conceitos e objetivos da Arquitetura Orientada a Serviços (SOA) e fica evidente que a adoção destas arquiteturas é fundamental para que as empresas consigam atender as mudanças na velocidade que o mercado exige.

No próximo capítulo serão apresentados estes conceitos, vantagens e princípios fundamentais para adoção desta arquitetura.

Protocol ou Protocolo de Transferência Hipertexto) e executados em um sistema remoto que hospeda os serviços solicitados. Na seção 3.4 será melhor detalhado.

[25] O modelo **MTNM** (*Multi-Technology Network Management* ou Gerenciamento de Redes de Multi-tecnologia) proporciona uma solução única e comum para a gestão de multi-tecnologia de redes (incluindo SONET, SDH, DWDM, ATM, Ethernet e outras) e as novas tecnologias e funcionalidades de redes emergentes da ITU e SG15 OIF, incluindo Plano de Controle (*Control Plane*) e Ethernet.

[26] **MTOSI** (*Multi-Technology Operations System Interface ou Interface Multi-tecnologia para Sistemas de Operações, em uma tradução livre)* é um padrão para implementação de interfaces entre OSS criado pelo Fórum Telemanagement (TM Forum).

[27] O **OSS/J** (OSS através da iniciativa Java) é um programa técnico do TM Forum, cujo principal objetivo é desenvolver um conjunto padrão de APIs para suportar os OSS e BSS.

3. ARQUITETURA ORIENTADA A SERVIÇOS (*SERVICE ORIENTED ARCHITECTURE* – SOA)

Segundo (Erl, 2007), não há nenhuma definição reconhecida de SOA, o que existe é uma base de conceitos e princípios que existe dentro das empresas, plataformas e iniciativas que vem influenciando e que continua moldando o conceito de SOA. Dentre os conceitos, podemos destacar:

- Agilidade: Capacidade de resposta frente às mudanças de negócio;
- Reutilização: Aumento da eficiência através do reuso de serviços de negócio comuns, compartilhados entre diversos processos e sistemas;
- Política de Implementação Consistente, Progressiva e Flexível;
- Governança: Previne a abordagem desorganizada e inconsistente para a implementação de serviços através de uso de políticas bem definidas e gerenciadas;
- Redução de Custos: Sistemas baseados em padrões, modulares e simples de serem modificados e atualizados, levando a redução de custos de manutenção.

Ainda segundo (Erl, 2007) "A sociedade é completamente orientada a serviços". De acordo com (Sampaio, 2006) "SOA é um novo paradigma de desenvolvimento de aplicações cujo objetivo é criar módulos funcionais chamados de serviços, com baixo acoplamento e permitindo a reutilização de código".

SOA traz a promessa de uma TI (Tecnologia da Informação) que responda mais rapidamente às tendências do mercado com a melhoria dos serviços ao cliente e redução dos custos de desenvolvimento de aplicativos. Estes benefícios por si só já são suficientes para justificar a adoção de SOA, principalmente em áreas estratégicas onde os resultados podem ter um impacto sem precedentes na organização.

3.1. CONCEITOS

O termo "*Service Oriented Architecture*" (SOA) ou Arquitetura Orientada a Serviços expressa um conceito onde aplicativos ou rotinas são disponibilizados como serviços em uma rede de computadores (Internet ou Intranets) de forma independente e se comunicando através de padrões abertos. A maior parte das implementações de SOA

se utiliza de Web Services (ou Serviços Web) e tecnologias SOAP[28], REST[29] e WSDL[30]. Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em web. A Figura 3.1 apresenta, de forma simplificada, uma visão geral de uma arquitetura orientada a serviços.

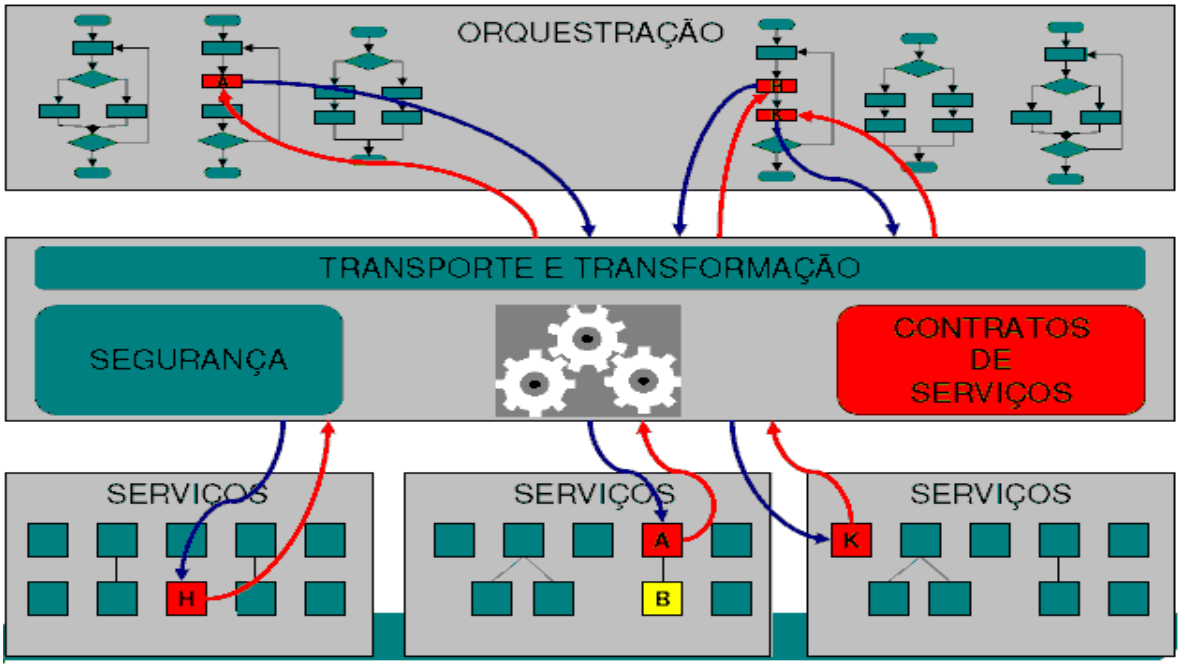

Figura 3.1: – Esquema de uma Arquitetura Orientada a Serviços. Fonte: (Gartner Group, 2006)

O problema é que as organizações não podem simplesmente comprar SOA; elas devem entendê-la e vivê-la. SOA é um paradigma, é uma maneira de pensar.

Constata-se que há um grande movimento por parte das empresas no sentido de se prepararem para a adoção do SOA. A AT&T[31], por exemplo, é um caso de sucesso

[28] **SOAP** (*Simple Object Access Protocol* ou Protocolo de Acesso a Objetos Simples) é um protocolo, criado pela W3C, relativamente leve destinado à troca de informações estruturadas em um ambiente descentralizado e distribuído.
[29] **REST** (*Representational State Transfer* ou Transferência de Representação de Estado) é um estilo de arquitetura de software para sistemas distribuídos como a World Wide Web.
[30] **WSDL** (*Web Service Description Language* ou Linguagem Descritiva de Serviços Web), definido pela W3C, é basicamente um arquivo em formato XML que descreve serviços de rede como um conjunto de parâmetros e mensagens operacionais que contenham qualquer informação orientada a documento ou a procedimentos (procedures).
[31] **AT&T** (*American Telephone and Telegraph*) é uma das principais companhias americanas de telecomunicações. A companhia que se tornou AT&T começou em 1875, em um acordo entre o inventor

na implementação de SOA em empresas de telecomunicações. Os números apresentados pela AT&T mostram vários benefícios alcançados: Redução média de 50% no custo com integrações de sistemas; A reutilização de serviços economizou de 50% a 85% o custo na construção de interfaces customizadas.

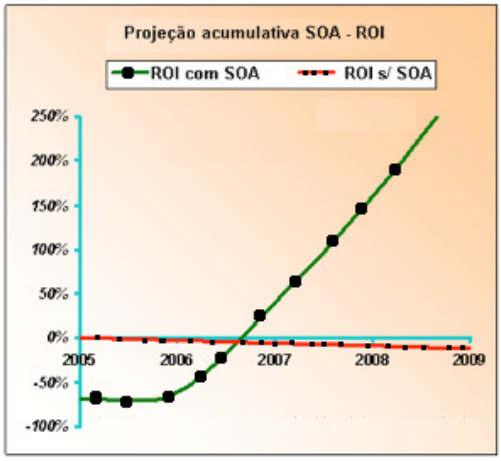

Figura 3.2 - Retorno do Investimento (ROI), Fonte: (Erickson, 2007)

A **Figura 3.2** mostra que a economia tende a se acumular à medida que o tempo passa. Esta projeção levou em consideração as seguintes premissas:

- Taxa de reuso dos serviços existentes de aproximadamente três vezes por serviço durante um período de 10 anos;
- Taxa de crescimento da adoção do SOA de 25% dos projetos em 2006 até 90% dos projetos em 2009;
- Média de gastos para criação de serviços SOA pela primeira vez é de 10% sobre os custos correntes;
- Gastos anuais com base no ano de 2005.

3.1.1. O conceito de serviços

As organizações implementaram o conceito de serviço no nível empresarial bem antes de utilizarem-no nos softwares. Muitas das funções de negócio em uma empresa

---

Alexander Graham Bell e dois homens, Gardiner Hubbard e Thomas Sanders, que concordaram em financiar o seu trabalho.

moderna são organizadas na forma de serviço – por exemplo, companhias usam serviços de limpeza, serviços de fotocópias, serviços de transportadoras e correios, agências de viagens e muitos outros serviços.

As mesmas características que tornam o conceito de serviço útil na organização das unidades de negócio são utilizadas no projeto de aplicações SOA. No entanto, utilizar serviços de desenvolvimento de software exige um grau de rigor e precisão que nem sempre são necessários em serviços de nível empresarial. Será mostrado mais adiante que este rigor é traduzido na forma de oitos princípios que são as diretrizes na criação de arquiteturas orientadas a serviços. Antes, porém, é importante que haja o entendimento do conceito de granularidade de serviço.

3.1.2. Granularidade de serviços

A granularidade consiste na mensuração do tamanho, escala e nível de detalhes dos serviços. Onde serviços "gordos" ou “grosseiros” são *coarse-grained* (granularidade grossa) e serviços “magros” ou "leves" são *fine-grained* (granularidade fina) (Barros, 2005).

A quantidade global de funcionalidades encapsuladas por um serviço determina a granularidade dos serviços. A granularidade de um serviço é determinada pelo seu contexto funcional que é muitas vezes derivado de um dos três modelos de serviço comum. Por exemplo, um serviço baseado em um modelo de serviço de entidade terá um quadro funcional associado a uma ou mais entidades de negócio relacionadas. Funcionalidade associada com a entidade de negócio escolhida pertence às fronteiras funcionais do serviço. Quanto maior a quantidade de funcionalidades relacionadas, mais “grossa” será a granularidade de serviços. Por outro lado, os serviços mais voltados para a funcionalidade “alvo” tendem a ter uma granularidade mais “fina” (Erl, 2007).

O nível de granularidade de serviço é definido pelo quadro funcional do serviço, não pela quantidade real de funcionalidade que reside dentro do serviço fisicamente implementado. A granularidade de serviço representa um dos quatro tipos de granularidade de *design*: granularidade de serviço, de capacidade, dados e granularidade de restrição (*constraint*). Cada um destes tipos de granularidade é afetado de forma diferente através da aplicação de princípios de projeto orientado a serviços.

3.2. OS OITO PRINCÍPIOS DO SOA

O conceito de SOA pode ser mais bem compreendido através de oito princípios básicos que são necessários para que se possam ter soluções realmente orientadas a serviços. Estes princípios darão as diretrizes fundamentais para migração dos OSS para SOA, pois o enfoque será voltado para a avaliação de sistemas existentes.

3.2.1. Contratos de serviços padronizados

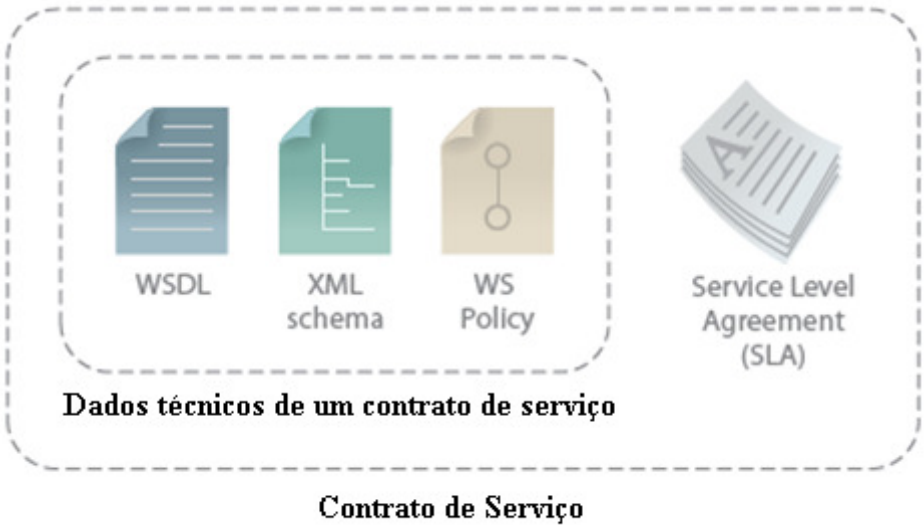

Figura 3.3 - Contrato de Serviço. Fonte: (Erl, 2007)

Em SOA, os serviços são definidos formalmente através de contratos. O contrato formal é um acordo que define o entendimento e as expectativas de um solicitante de serviço a partir do seu provedor do serviço. Tal entendimento possibilita a comunicação bem-sucedida e interação entre os solicitadores e provedores. Os ativos sob consideração para a avaliação deste princípio são os contratos de descrições/interface entre os sistemas existentes. Para avaliar este princípio, é preciso entender primeiro o que os componentes de um contrato formal são e, então, avaliar quantos destes são cobertos por contratos de descrições/interfaces empregadas nos sistemas existentes. Um contrato de serviço (Figura 3.3) normalmente contém o seguinte (Schmelzer, 2007):

- Os aspectos funcionais: Esta parte do contrato tipicamente menciona o conjunto de operações que são oferecidos por um serviço e suas localizações. Os

insumos necessários e as saídas produzidas por essas operações também são mencionados;

• Os aspectos não-funcionais: Esta parte trata do contrato sobre os aspectos não-funcionais de um serviço como segurança, confiabilidade, garantia de desempenho (SLA), etc. Isso dá uma compreensão do que o serviço não faz, mas a maneira que ele se comporta sob o ponto de vista do negócio;

• Políticas: Esta parte do contrato menciona as regras de engajamento entre consumidores e prestadores de serviços, conhecidos como políticas, que determinam quem pode acessar um provedor, os protocolos de comunicação necessários para os participantes ou de quaisquer outras regras que se aplicam.

Como já mencionado, é preciso ver se todos os componentes acima mencionados são cobertos por contratos existentes. Quanto maior o número de componentes abrangidos, mais próximo os contratos existentes estarão na concretização deste princípio.

3.2.2. Autonomia de serviço

O princípio de autonomia de serviço diz que os serviços devem ser coesos. Originalmente, o princípio de autonomia de serviço é proposto no contexto de paradigma estruturado, que é definido como o grau da relação entre os elementos dentro de um módulo (Stevens et al., 1999). Serviços com autonomia maior sugerem que todos os elementos internos formam uma unidade, minimizando assim a necessidade de dependências de elementos de outros serviços. Os ativos sob consideração para a avaliação deste princípio são as unidades de desenvolvimento como funções, módulos (módulo é um grupo de funções), classes e componentes. Como os ativos existentes normalmente seriam de grão fino e serviços são ásperos, a coesão de um grupo de ativos em que um serviço será construído, deve ser medida em alguns casos.

As métricas de coesão medem como os métodos de uma classe são relacionados uns aos outros. Uma classe coesa executa uma função. Uma classe não coesa executa duas ou mais funções independentes. Uma classe não coesa pode precisar ser reestruturada em duas ou mais classes de menor dimensão.

Bieman et al.(1995), propôs uma forma de medirmos esta coesão através de métodos baseados em duas medidas TCC(*Tight class cohesion* – Classe de autonomia forte) (Eq 3.2) e LCC (*Loose class cohesion* – Classe de autonomia fraca) (Eq 3.3). Os métodos podem ser diretamente relacionados pela mesma variável de instância que

chamaremos de NDC (*Number of Direct Connections* – Número de Conexões Diretas). Elas podem ser indiretamente ligadas se acessarem a variável através de alguns métodos intermediários, chamaremos esta variável de NIC (*Number of Indirect Connections* – Número de conexões indiretas). O número total de conexões possíveis (NP) (Eq 3.1) dado N métodos é:

$$NP = \frac{N * (N - 1)}{2} \qquad \text{(Eq. 3.1)}$$

$$TCC = \frac{NDC}{NP} \qquad \text{(Eq. 3.2)}$$

$$LCC = \frac{(NIC + NDC)}{NP} \qquad \text{(Eq. 3.3)}$$

TCC e LCC são valores com domínio entre 0 e 1. Valores altos indicam altos níveis de coesão.

Quais são os valores bons e ruins? Segundo Bieman e Kang(1995), TCC <0,5 e LCC <0,5 são consideradas classes não coesas. LCC = 0,8 é considerada "bastante coesa". TCC = LCC = 1 é uma classe de coesão máxima: todos os métodos estão ligados.

LCC indica o nível geral de conectividade. Depende do número de métodos e como eles se agrupam, a saber:

- Quando LCC = 1, todos os métodos da classe estão ligados, direta ou indiretamente. Este é o caso de coesão;
- Quando LCC < 1, existem dois ou mais grupos de métodos desconexos. A classe não é coesa. Pode-se querer revisar essas classes. Os métodos podem estar desconectados porque acessam as variáveis que não estão no nível da classe ou porque acessam variáveis totalmente diferentes;
- Quando LCC = 0, todos os métodos são totalmente independentes. Este é o caso de classe não coesa.

TCC nos diz qual a “densidade de ligação”, por assim dizer (enquanto LCC só é afetado se os métodos estão conectados em tudo):

- TCC = LCC = 1 é o máximo da classe coesa, onde todos os métodos estão diretamente ligados uns aos outros;
- Quando o TCC = LCC < 1, todas as conexões existentes são diretas (embora nem todos os métodos estão conectados);
- Quando TCC < LCC, a densidade de "ligação" é menor do que poderia ser na teoria. Nem todos os métodos estão diretamente ligados uns aos outros. Por exemplo, A & B estão ligados através da variável x e B & C através da variável y. A & C não compartilham uma variável, mas elas estão indiretamente conectadas através de B;
- Quando o TCC = 0 (e LCC = 0), a classe é totalmente não coesa e todos os métodos são totalmente independentes.

É importante notar que este é apenas um dos métodos disponíveis para medição de autonomia entre ativos. Chidamber e Kemerer (1994) propuseram uma métrica para a medição da autonomia de uma classe no contexto do paradigma orientado a objetos. Ott e Thuss (1993) apresentam uma metodologia para compreender a autonomia de módulos (que é um conjunto de funções).

3.2.3. Serviços com baixo acoplamento

O princípio de serviço com baixo acoplamento diz que os serviços devem ser fracamente acoplados, ou seja, o grau de dependência entre dois serviços deve ser mínimo. Para avaliar este princípio nos ativos já existentes, pode-se utilizar o método mencionado no princípio Autonomia de Serviço. Por exemplo, os valores mais elevados de coesão baseada na distância métrica (Simon et al., 1999) sugere autonomia pobre ao longo de um grupo de ativos e mostra que os ativos individuais são fracamente acoplados. Do mesmo modo, métricas com valores baixos para “utilização de serviços solicitados” e para “utilização de serviços prestados” (van der Hoek et. al., 2003) indicam a ligação indevida de um serviço com o outro.

Além da avaliação do nível de ativos, também é preciso compreender o apoio de infra-estrutura para avaliar o princípio de baixo acoplamento. A lista a seguir mostra os

diferentes aspectos no nível da infra-estrutura que são úteis na compreensão do princípio de baixo acoplamento:

- Estilo de comunicação síncrona versus assíncrona: Tipicamente, se o estilo de comunicação é síncrono, o acoplamento será mais forte em oposição ao estilo de comunicação assíncrona, que é tratado como acoplamento fraco. Mesmo no estilo de comunicação assíncrona, utilizando RPC assíncrono, o acoplamento é forte, pois há a necessidade de um *server* (neste caso é um serviço rodando em um servidor) ativo para tratamento das solicitações. Por outro lado, um *middleware* orientado a mensagens suporta solicitações sem a necessidade de um *server* ativo, pois as solicitações podem ser armazenadas em buffers e processadas quando o *server* estiver ativo;
- Semântica por interface versus semântica por *payload*[32]**:** a semântica por interface resulta em acoplamento forte, pois há uma forte dependência dos clientes com o tipo de informação do servidor. No caso da semântica por *payload*, essa dependência pode ser eliminada, pois há uma fraca dependência no tipo de informação do servidor;
- Serviços com estado (*statefullness*) versus serviços sem estado (*Statelessness*): serviços com estado tornam os clientes fortemente acoplados a um determinado servidor e, portanto, serviços sem estado são preferidos para o baixo acoplamento. Embora este seja listado como aspecto de infra-estrutura (serviços sem estado necessitam de apoio da infra-estrutura para armazenar o estado em algum outro lugar), a forma como os ativos individuais são projetados também influencia significativamente os serviços sem estado;
- Ligação (*biding*) dinâmica versus estática**:** o modo de ligação estático resulta em acoplamento forte, pois o cliente tem que saber de antemão os dados para conexão com o servidor antes de se conectar a ele. Por outro lado, ligação dinâmica, onde o cliente pode localizar o servidor em tempo de execução usando os mecanismos de registro (buscando os serviços desejados em registros utilizando nomes, propriedades, etc) resulta em acoplamento baixo.

[32] **Payload** (ou carga útil) em protocolos de comunicação refere-se aos dados reais que são transmitidos.

3.2.4. Abstração de serviço

Este princípio diz que os serviços devem abstrair toda a sua complexidade. Existem duas dimensões que dizem respeito à avaliação das abstrações existentes com a abstração do nível de serviço. Um deles é o aspecto de granularidade e o outro aspecto é o quão oculta está à complexidade.

Serviços abstratos são geralmente de granularidade grossa. A motivação para estas abstrações de granularidade grossa é a necessidade de alinhamento com a lógica de negócio. Assim, em um nível de granularidade, os sistemas baseados em componentes (a maior unidade de desenvolvimento será um componente) são mais adequados para adoção de SOA, pois os componentes são tipicamente considerados como abstrações de granularidade grossa com alinhamento relativamente próximo à lógica de negócio quando comparado com sistemas onde os altos níveis de abstração são procedimentos, módulos ou classes (Reddy et al., 2008)

No que diz respeito a esconder a complexidade, é necessário compreender o nível de detalhes que estão ocultos usando a abstração de serviço. Em última análise, é a tecnologia subjacente que facilita nesta ocultação. Por exemplo, ao implementar SOA utilizando *Web Services* (Alonsom et al., 2004), pode-se ocultar os detalhes sobre os recursos de hardware, sistemas operacionais e linguagens de programação. Além disso, os detalhes de implementação da plataforma subjacente podem ser escondidos. O cliente não precisa se preocupar com estes detalhes sobre o serviço a ser utilizado. Tudo o que é necessário para o cliente utilizar um serviço. É o WSDL (*Web Services Description Language*) que descreve o serviço e lista as operações que este realiza. O cliente simplesmente acessa a descrição WSDL e utilizando SOAP (*Simple Object Access Protocol*), por exemplo, chama as operações de serviço necessárias. Com suporte do UDDI (*Universal Description, Discovery and Integration*) até mesmo a localização dos serviços é transparente para os clientes.

Para avaliar o princípio de abstração entre os sistemas existentes, é preciso ver quantos dos detalhes mencionados acima podem ser suportados e ocultados pelas tecnologias existentes. Pode-se argumentar que, além do aspecto da ocultação da complexidade, existem outros aspectos da abstração, como a reutilização, coesão, etc. devem ser considerados para a avaliação. Todos estes aspectos são abordados nos demais princípios.

3.2.5. Capacidade de reuso de serviço

Este princípio diz que os serviços devem ser reutilizáveis, ou seja, cada serviço deve ser posicionado como um ativo de TI que traz valor repetível por ser reutilizado várias vezes (acessado por vários clientes) e em vários contextos (diferentes composições de serviços). Os ativos sob consideração para avaliar este princípio são as unidades de desenvolvimento, como classes, componentes e módulos.

Todas as métricas que dão uma compreensão da coesão são também úteis para a mensuração do nível de reutilização de um ativo. Um ativo com a natureza coesiva (com chances de minimizar as dependências externas) é facilmente reutilizável em relação a um ativo com a natureza menos coesa. Assim, há que se considerar a métrica apresentada no princípio Autonomia de Serviço para a medição da reutilização.

3.2.6. Visibilidade de serviços

Segundo este princípio, os serviços devem ser facilmente detectáveis. Este princípio ajuda na promoção da reutilização dos serviços (serviço que é detectável pode ser relativamente reutilizável) e também para impedir a criação acidental de serviços com lógica redundante. O registro do serviço fornece um mecanismo de descoberta, permitindo as solicitações de consultas e acesso aos serviços já cadastrados. Assim, a presença de registro no nível da infraestrutura melhora a descoberta dos serviços.

Variantes desse recurso como a busca dos serviços que utilizam consultas (*querys)* ao invés de nomes melhora ainda mais a descoberta. Por exemplo, uma consulta como "encontre a plataforma de mensagens do assinante John Doe Junior" é um mecanismo de descoberta de serviços mais fácil do que achar o nome de uma plataforma em especial.

3.2.7. Capacidade de composição de serviços

Este princípio prega que os serviços devem ser facilmente combináveis com outros para criar serviços de níveis diferentes de granularidade de uma maneira hierárquica. Cada nível da hierarquia é representado por uma composição de serviços que, por sua vez é composta pelos serviços compostos que representam os níveis mais baixos da hierarquia. Esses serviços compostos aumentam a simplicidade de uso para os clientes, ou seja, para o consumidor do serviço, a execução do serviço composto é completamente transparente. As métricas de reutilização (seção 3.2.2) são bastante úteis

na avaliação do princípio da capacidade de composição de serviço. Além das já mencionados, as seguintes características também afetam o nível de agregação (Buchhirano e Gnesi, 2006; Canfora et al., 2006):

- Papéis distintos: Um serviço pode desempenhar papéis diferentes, dependendo das etapas do processo. Por isso, é preciso verificar se os papéis do ativo (ativo aqui é uma unidade de desenvolvimento, como componente) sob diferentes cenários são claramente identificáveis ou não;
- Estilo de interação *Request-Response*: se o atual sistema usa um estilo de interação de solicitação-resposta dos serviços, em oposição a um estilo de interação, como resposta à consulta (estilo de interação baseado em formulários), será necessário algum esforço para mudar o estilo de interação;
- Suporte para processos de execução longa: os processos de negócio que resultam de composição de serviços podem ser geralmente de longa duração e, consequentemente, a capacidade de gerenciar transações e compensações é de grande importância para uma composição bem sucedida para tais processos. Deve verificar se existe suporte para a composição desses processos de longa duração nos sistemas existentes. Como pode ser visto, esta é uma avaliação de infra-estrutura ao invés de uma avaliação individual de ativos relacionados;
- Tratamento de exceções: Os serviços têm de lidar com as exceções se os serviços que eles invocam (como parte da composição), não respondem. A presença de tratamento de exceções é, portanto, uma característica importante a ser verificada no nível dos ativos;
- Disponibilidade de descrições semânticas: se os ativos existentes têm suas descrições semânticas (as capacidades e os comportamentos), disponíveis ou não. Tais descrições semânticas tornam mais fáceis a composição automatizada.

3.2.8. Independência de estado de serviço

Este princípio mostra que os serviços devem manter pouco ou nenhum estado. Serviços com estado são indesejáveis porque haveria um impacto na escalabilidade em

dois aspectos. Em primeiro lugar, haveria um limite para o número de clientes que pode ser conectado a um serviço particular, devido ao armazenamento de informações do estado sobre os clientes. Em segundo lugar, é difícil distribuir os pedidos em diferentes servidores, devido à afinidade do cliente com um serviço particular.

Para atingir este objetivo são necessários tanto o apoio de infraestrutura quanto a consciência que o serviço individual a ser projetado não deve ter estado. A manutenção do estado através de *tokens*[33] persistentes nas mensagens em uma sessão ou nível de atividade e à prestação de serviços contratados para gerenciar o estado de todas as atividades em execução são alguns mecanismos (Schmelzer, 2007) que se tem que verificar, nos sistemas existentes, para compreender o suporte de infra-estrutura necessário para implementação dos serviços sem estado. Da mesma forma, minimizar as informações de estado necessárias para cada sessão, liberando recursos de sessão rapidamente, evitando o acesso às variáveis de sessão a partir da lógica de negócios, projeto de operações idempotentes[34], sempre que possível, são alguns dos aspectos do projeto (Meier et al., 2004) para verificar se tem um nível de desenvolvimento individual por unidade (módulo, classe ou componente). Especialmente, o desenho das operações idempotentes tem uma influência significativa sobre as operações sem estado a medida que as operações idempotentes geralmente eliminam a necessidade de armazenar o estado. Por exemplo, a Figura 3.4 apresenta um pseudocódigo onde a operação moverAbsolutodoInicio(int metros) é uma operação idempotente e moverRelativamente(int metros) não é uma operação idempotente. Como se pode observar este último exige o estado anterior para mover-se relativamente em que a operação não é idempotente. Muitos sistemas existentes, como o NFS (*Network File System*) (Sun Microsystems, 1994), utilizam esta técnica para a construção serviços sem estado.

[33] Um **token** de sessão é um identificador único que é gerado e enviado a partir de um servidor para um cliente para identificar a interação de sessão atual.
[34] **Idempotentes** são operações que resultam em efeitos iguais, mesmo se chamadas várias vezes.

```
class ServicoA {
  private:
      int posicaoAtual = 0;
      // função interna que realiza a operação de mover
      void mover(int metros) {
           Carro.Mover(metros);
       }
  public:
      void moverAbsolutodoInicio(int metros)  {
           mover(metros);
      }
      void moverRelativamente(int metros) {
           mover(posicaoAtual - metros);
           posicaoAtual = posicaoAtual + metros;
       }
}
```

Figura 3.4 – Pseudo-código que exemplifica serviços com e sem estado.

3.3. A EVOLUÇÃO DO SOA

A partir dos sistemas monolíticos do início dos anos 1960 a indústria viu o desenvolvimento (Figura 3.5) das arquiteturas de TI passando para sistemas estruturados, depois cliente/servidor[35], *3-tier*[36] (três camada), *n-tier*[37] (n-camadas), sistemas/objetos distribuídos, componentes e finalmente as arquiteturas orientadas a serviços da idade moderna. (Jennings et al., 2008).

[35] O modelo **cliente/servidor** em computação é uma estrutura de aplicativos distribuídos em que as tarefas ou cargas de trabalho disponibilizadas por fornecedores de um recurso ou serviço, chamados de servidores, são invocadas via rede, pelos e os solicitantes de serviços, chamados de clientes.
[36] A arquitetura de três camadas ou **3-tier** é uma extensão do modelo cliente/servidor com a adição de uma terceira camada de persistência que tipicamente é um Banco de Dados.
[37] **N-Tier** (ou multi-camadas) É uma arquitetura cliente-servidor no qual a apresentação, o processamento de aplicações e o gerenciamento de dados são processos separados logicamente.

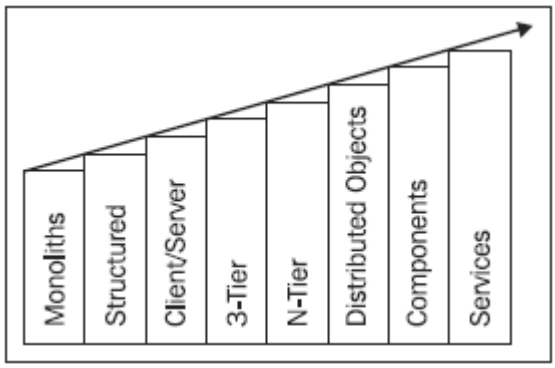

Figura 3.5 - Evolução das arquiteturas de TI. Fonte: (Jennings et. al., 2008)

As raízes da orientação a serviços podem ser encontrados em três áreas diferentes: os paradigmas de programação, a tecnologia distribuída e a computação empresarial (ou voltada para o negócio). O desenvolvimento de paradigmas programação de diferentes linguagens não só tem contribuído para a plataforma de implementação para os diferentes elementos de uma SOA, mas também tem influenciado as técnicas de interface utilizada em SOA, bem como os padrões de interação que são empregados entre os provedores e consumidores de serviços. Muitos dos conceitos originalmente encontradas em linguagens de programação também têm feito o seu caminho para a tecnologia de distribuição que é atualmente utilizado para oferecer acesso remoto a serviços prestados por diferentes aplicações em diferentes plataformas tecnológicas. Por último e talvez mais importante, a evolução da computação empresarial resultou em um grande número de proprietários, bem como pacotes de aplicações como ERP, SCM e CRM, que hoje estão a fornecer o conteúdo (os dados e lógica de negócio) que traz um enterprise SOA para a vida. Devido à proximidade de serviços para a funcionalidade do negócio concreto, a orientação de serviço tem o potencial para se tornar o primeiro paradigma que realmente traz tecnologia e negócios em um nível onde as pessoas de ambos os lados podem igualmente compreender e falar sobre os conceitos subjacentes.

Nos anos 1980, as aplicações em sua maioria eram verticais, construídas para atender as necessidades dos clientes em um segmento de mercado também vertical. Por exemplo, uma indústria automobilística (Figura 3.6) nunca necessitou interagir com os seus fornecedores por meios eletrônicos. O mesmo aconteceu no caso da maioria das

outras indústrias. Muito raramente, houve uma necessidade de se comunicar com outras empresas. (Jennings et al., 2008) .

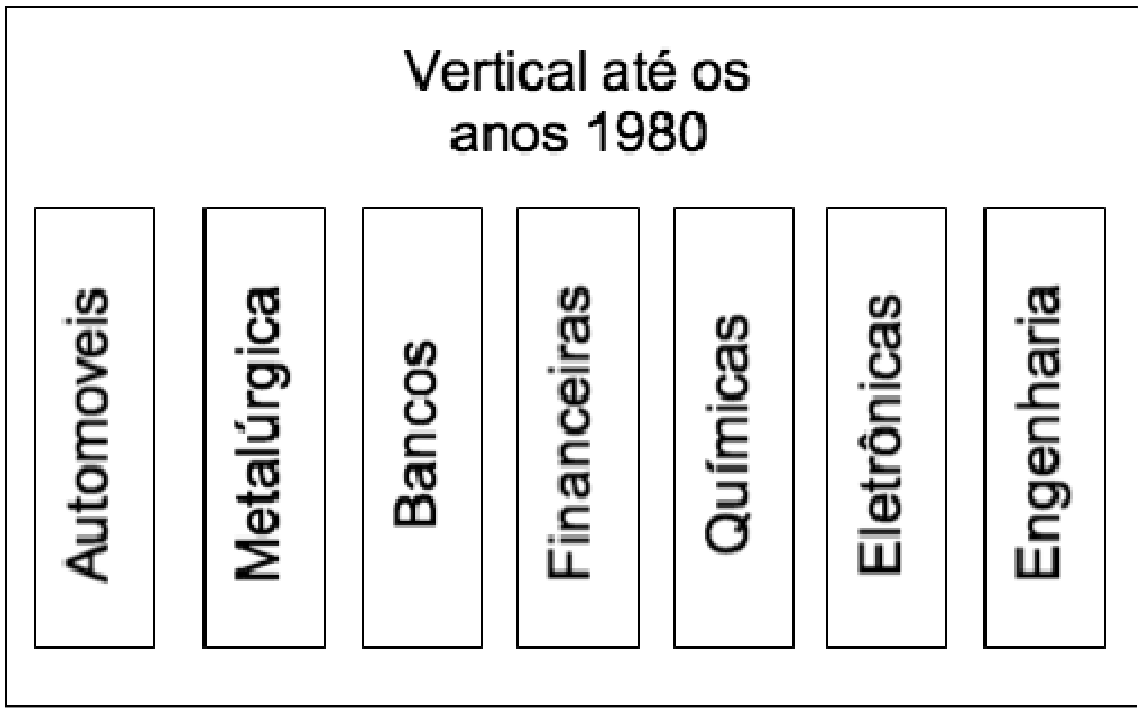

Figura 3.6 - Aplicações/mercados verticais Fonte: (Jennings et al., 2008)

No final dos anos 1980 e início dos anos 1990, percebeu-se a necessidade das aplicações de negócios crescerem horizontalmente para cooperarem com os parceiros de negócios. A indústria viu a evolução do B2B[38] (*Business-to-Business*) através de colaborações de componentes agora se espalhando por vários segmentos verticais da indústria. Esses componentes foram distribuídos dando origem a uma cadeia de suprimentos estendida, fornecendo aos clientes e parceiros de negócios acesso aos serviços (Figura 3.7).

No mundo de hoje, a maneira que as empresas operam mudou consideravelmente. As empresas não querem apenas a interação com seus parceiros, mas elas permitem que seus clientes e funcionários acessem seus serviços da empresa por meios eletrônicos.

[38] *Business to Business* (**B2B**) é o termo dado a operações realizadas entre duas empresas. Estas operações podem ser compra e venda, troca de informações, de produtos e serviços

Hoje, fala-se de B2C[39] (*Business-to-Consumer*), pelo qual os clientes têm acesso direto aos serviços oferecidos pelas empresas. Expor a lógica de negócios para uma base de usuários não confiáveis apresenta seus próprios desafios em termos de segurança e integridade. Além disso, esses serviços devem ser fáceis de usar e devem ocultar a complexidade dos processos de negócios internos do cliente final.

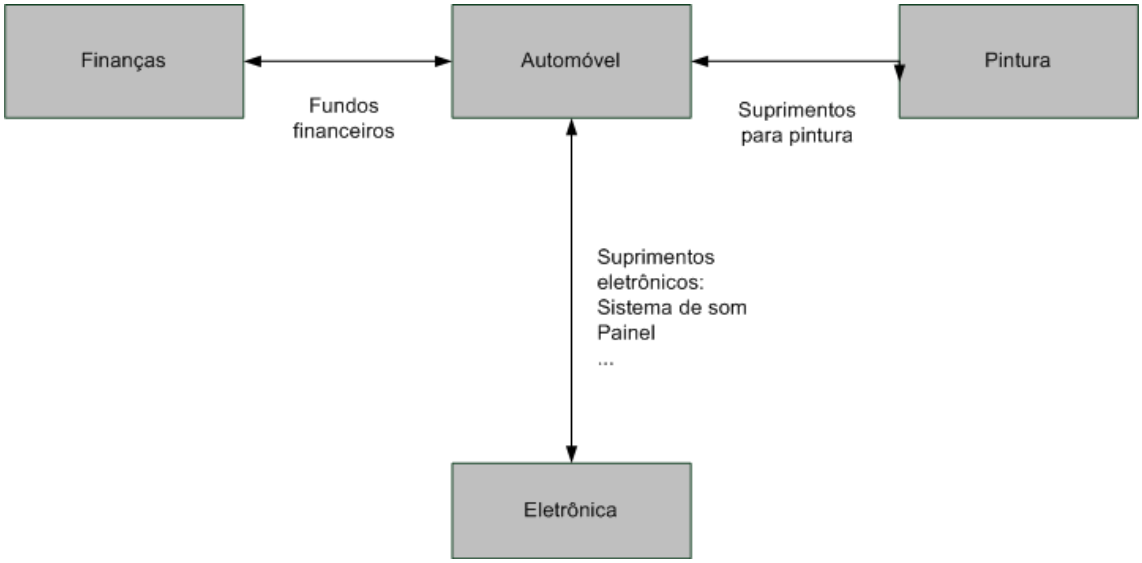

Figura 3.7 – Adaptado de Início da integração B2B(*Business-to-Business*). Fonte: (Jennings et al., 2008)

A arquitetura orientada a serviços tenta satisfazer as necessidades dos negócios de hoje. Eles são de baixo acoplamento, localização transparente e independente de protocolo. SOA esconde a tecnologia subjacente às arquiteturas do consumidor do serviço. A implementação do serviço pode estar em um Java EE[40] (anteriormente J2EE) ou .NET[41] (*dot Net*), ou pode ainda ser uma aplicação legada em um mainframe da

[39] **B2C** é uma sigla para *Business to Consumer*. O termo se refere a um tipo de negócio/site/tecnologia que incide sobre as transações entre empresas e consumidores, ao contrário de operações **B2B**[38].

[40] *Java Platform, Enterprise Edition* ou **Java EE** é uma plataforma amplamente utilizada para a programação do servidor na linguagem de programação Java. A plataforma Java (*Enterprise Edition*) é diferente do *Java Standard Edition Platform* (Java SE) na medida em que acrescenta bibliotecas que permitem implementar funcionalidades tolerante a falhas, distribuídos, multi-camada, amplamente baseado em componentes modulares rodando em um servidor de aplicativos. http://www.oracle.com/technetwork/java/index.html

[41] **Microsoft .NET** (comumente conhecido por .NET Framework - em inglês: dotNet) é uma iniciativa da empresa Microsoft, que visa uma plataforma única para desenvolvimento e execução de sistemas e

IBM[42]. O consumidor do serviço não precisa conhecer a plataforma na qual o serviço está sendo executada, a implementação do serviço é totalmente transparente para o consumidor.

3.4. AS PRINCIPAIS TECNOLOGIAS UTILIZADAS NA IMPLANTAÇÃO DE SOA

O conceito de SOA pode ser implantado em uma empresa sem necessariamente utilizar recursos computacionais, pois, como dito anteriormente, SOA é na verdade um conjunto de conceitos e princípios e não uma coleção de tecnologias. No entanto, os recursos computacionais disponíveis hoje em dia, fornecem uma ajuda significativa na tarefa de implantação destes conceitos e princípios.

Com a popularização e capilaridade da Internet, os serviços passaram a ser disponibilizados na internet, aproveitando tecnologias e protocolos já consagrados, como o HTTP, para aproveitar a infraestrutura e base de *softwares* já instalados.

A Figura 3.8 mostra as relações existentes entre as tecnologias e padrões mais comumente utilizadas para implantação de SOA.

aplicações. Todo e qualquer código gerado para **.NET**, pode ser executado em qualquer dispositivo que possua um framework de tal plataforma.

[42] *International Business Machines* (**IBM**) é uma multinacional de informática, tecnologia e consultoria de TI com sede em Armonk, Nova York, Estados Unidos.

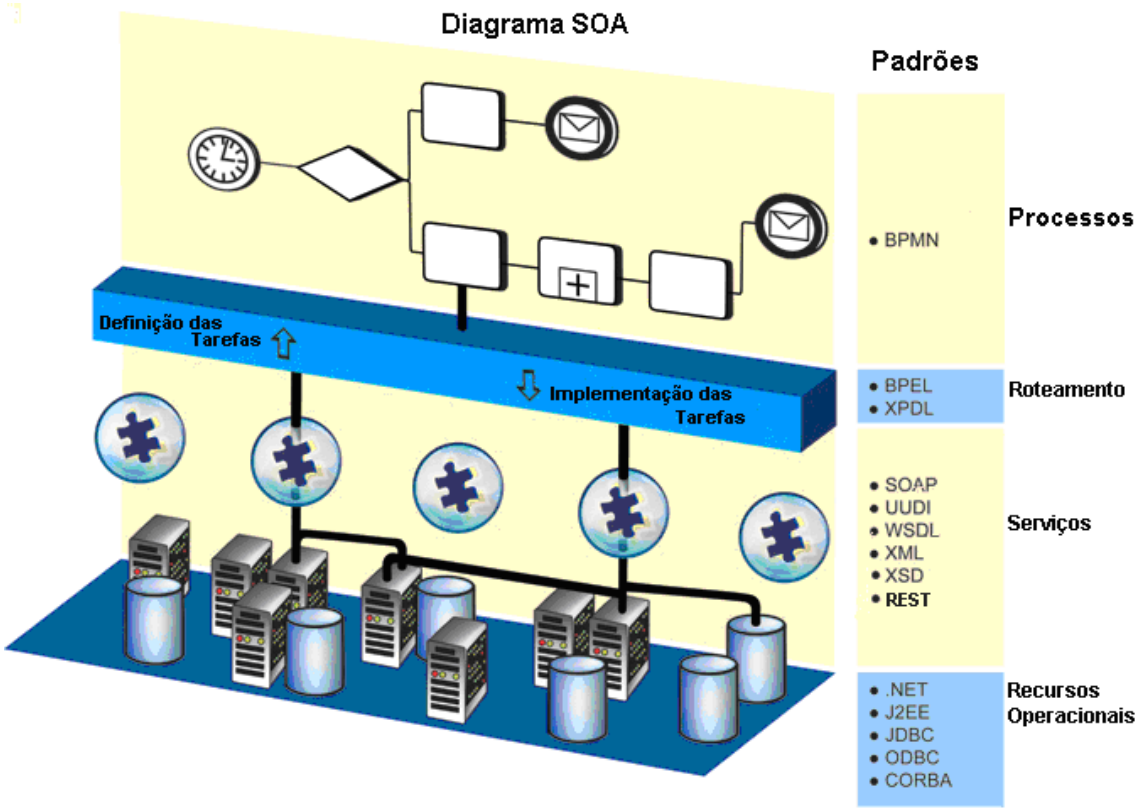

Figura 3.8 – Adaptado do *SOA Diagram – Interfacing People Processes* (2007)

As duas arquiteturas mais utilizadas atualmente são o SOAP e o REST que serão apresentados nas próximas seções.

3.4.1. Protocolo de Acesso Simples a Objetos (*Simple Object Access Protocol* – SOAP)

Embora originalmente concebido como uma tecnologia para cobrir o fosso deixado por diferentes plataformas de comunicação baseadas em RPC[43] (*Remote Procedure Call*), SOAP evoluiu para o formato de mensagem e protocolo para uso com XML *Web Services*. A especificação SOAP define um formato de mensagens padrão, que consiste em um documento XML capaz de hospedar RPC e os dados centrados em documentos (Figura 3.10). A Figura 3.9**Erro! Fonte de referência não encontrada.** apresenta um exemplo de esqueleto de envelope SOAP.

[43] **RPC** (*Remote Procedure Call* ou Chamada Remota de Procedimentos) É uma comunicação inter-processos que permite que um programa de computador invoque a execução de uma sub-rotina ou procedimento em outro espaço de endereçamento (geralmente em outro computador em uma rede compartilhada), sem que o programador explicitamente codifique os detalhes para essa interação remota.

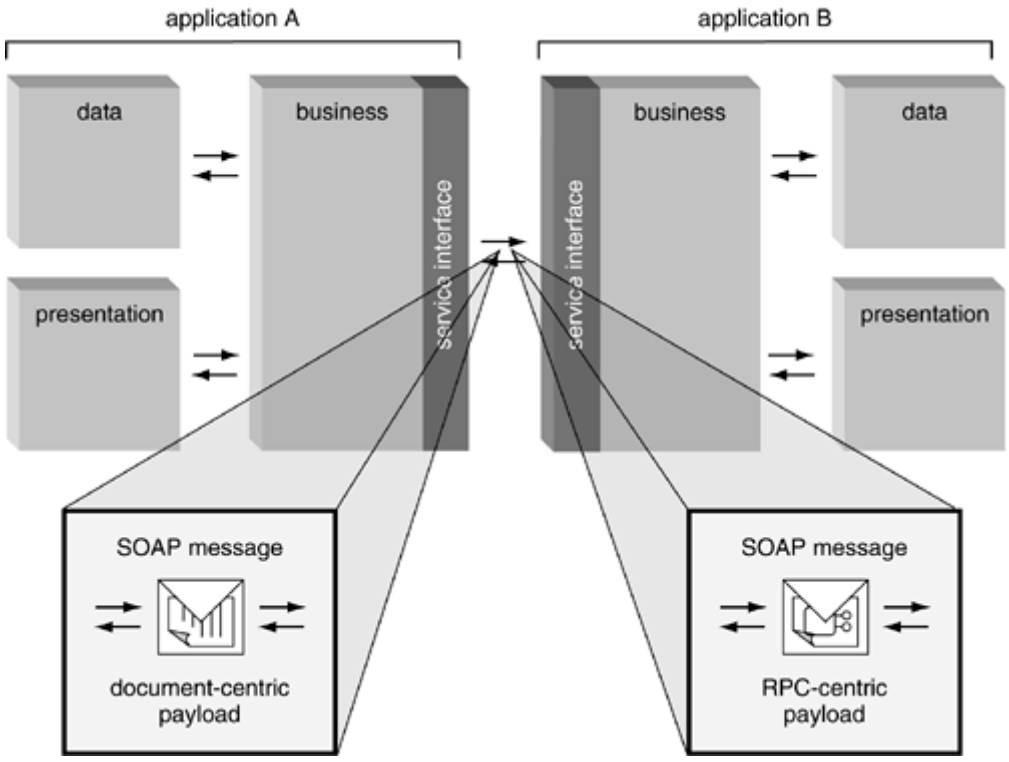

Figura 3.9 – SOAP – Estabelecimento de dois formatos: RPC e payload (dados). Fonte: (Erl, 2004)

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/
soap-envelope">
  <env:Header>
    ...
  </env:Header>
  <env:Body>
    ...
  </env:Body>
</env:Envelope>
```

Figura 3.10 – Exemplo de um esqueleto de um envelope SOAP

3.4.1.1. Descrição, Descoberta e Integração Universal (*Universal Description, Discovery and Integration* – UDDI)

Um componente fundamental da arquitetura SOAP é o mecanismo de descrições de serviços *Web* para serem descobertos por solicitadores potenciais. Para definirmos

esta parte de um *framework* de serviços *Web*, é necessário um diretório central para hospedar as descrições dos serviços. Tal diretório pode tornar-se parte integrante de uma organização ou uma comunidade da Internet, tanto que é considerada uma extensão da infraestrutura.

Esta é a razão pela qual o UDDI (*Universal Descrition, Discovery and Integration* – Descrição, Descoberta e Integração Universal), aprovado como padrão pela OASIS[44] (*Organization for the Advancement of Structured Information Standards*), tornou-se cada vez mais importante. As implementações podem variar em função da destinação dos serviços. Podem-se criar diretórios globais públicos, para integração entre empresas, por exemplo, assim como repositórios privados que são hospedados dentro de uma organização.

As informações capturadas no contexto UDDI são classificadas em três categorias principais e fazem uma analogia a uma lista telefônica (Sobral, 2006):

- Páginas Brancas (*White Pages*) – Essas incluem informações gerais sobre uma empresa específica, como por exemplo, nome de um negócio, descrição do negócio, informação de contato, endereço, números de telefone, fax, ou mesmo incluem identificadores de negócios (business identifiers), no formato de classificações D-U-N-S (*Data Universal Numbering System*), que são números de nove dígitos atribuídos a negócios. UDDI versão 2.0 oferece suporte para identificadores específicos de indústrias, tal como o sistema do *Standard Industrial Classification* (SIC), o qual atribui identificadores numéricos únicos a indústrias.

- Páginas Amarelas (*Yellow Pages*) – Essas incluem dados de classificação geral para qualquer empresa ou serviço oferecido. Por exemplo, esses dados podem incluir a indústria, o produto, ou códigos geográficos baseados sobre taxionomias padronizadas.

- Páginas Verdes (*Green Pages*) – Esta categoria contém informação técnica sobre um serviço na Web (*Web service*). Geralmente, essa informação inclui um

[44] **OASIS** (*Organization for the Advancement of Structured Information Standards ou* Organização para o Avanço de Padrões em Informação Estruturada) é um consórcio global, sem fins lucrativos, que conduz o desenvolvimento, convergência e adoção de padrões para o e-business e *web services*. http://www.oasis-open.org

apontador (ponteiro) para uma especificação externa e um endereço para invocar o serviço. UDDI não é restrito a descobrir serviços baseados em SOAP. Ao contrário, pode ser usado também, para descrever qualquer serviço, desde uma única página Web ou endereços de email, até serviços CORBA ou Java RMI[45], por exemplo.

3.4.1.2. Linguagem de descrição de serviços web *(Web Services Description Language - WSDL*)

WSDL é um formato XML para descrever serviços de rede como um conjunto de parâmetros operacionais sobre as mensagens que contenham qualquer informação orientada para documento ou orientada para processo. As operações e mensagens são descritas abstratamente e então ligadas a um protocolo de rede concreto e a um formato de mensagem para definição da ponta final de conexão (chamados de *endpoints*). Os *endpoints* concretos são combinados com os *endpoints* abstratos. WSDL é extensível para permitir descrição dos parâmetros e suas mensagens, independentemente dos formatos de mensagens ou protocolos de rede que são utilizados para comunicar-se, no entanto, somente as ligações contidas neste documento descrevem como usar o WSDL em conjunto com SOAP, HTTP GET / POST, e MIME. (W3C[46] Note 15, 2001)

Simplificadamente pode-se dizer que o arquivo WSDL é um documento XML que descreve um conjunto de mensagens SOAP e a forma como essas mensagens são trocadas. Em outras palavras, o WSDL é para o SOAP o que o IDL[47] (*Interface Definition Language*) é para o CORBA[48] (*Common Object Request Broker*

[45] Java **RMI** – *Java Remote Method Invocation* (Invocação de Métodos Remotos Java) é um recurso disponibilizado desde a versão 1.1 do Java que possibilita ao programador invocar métodos remotos de outras Java Virtual Machines (Maquinas virtuais Java), que podem ou não existir em diferentes servidores.

[46] **W3C** (*World Wide Web Consortium*) é uma comunidade internacional que desenvolve padrões para garantir o crescimento a longo prazo da Web.

[47] **IDL** (*Interface Definition Language* ou Linguagem de Definição de Interface), é uma linguagem de especificação utilizada para descrever a interface de um componente de software. IDLs descreve uma interface em forma de linguagem neutra, permitindo a comunicação entre os componentes de software que não compartilham a mesma linguagem, por exemplo.

[48] **CORBA** (*Common Object Request Broker Architecture* ou Arquitetura *de R*equisição Comum *a O*bjetos) é um padrão definido pelo *Object Management Group* (OMG – http://www.**omg**.org) que permite que componentes de software escritos em múltiplas linguagens e funcionando em vários computadores possam trabalhar em conjunto (ou seja, suporta várias plataformas).

Architecture) ou COM[49] (*Component Object Model*). Como o WSDL é XML, ele é legível e editável, mas na maioria dos casos, ele é gerado e consumido pelo software.

Os *Web Services* precisam ser definidos de uma forma consistente de modo que possam ser descobertos e ter interfaces por outros serviços e aplicações. O WSDL (*Web Services Description Language*) é uma especificação do W3C (*World Wide Web Consortium*) proporcionando o idioma principal para a descrição de definições de serviço Web. A mostra um exemplo de especificação de mensagens em WSDL.

```
<message name="recebeNumero">
<part name="numero" type="xsd:integer"/>
</message>
<message name="enviaResposta">
<part name="fatorial" type="xsd:float"/>
</message>
<portType name="fatorial_PortType">
<operation name="fatorial">
<input message="tns:recebeNumero"/>
<output message="tns:enviaResposta"/>
/operation>
</portType>
```

Figura 3.11 – Exemplo de especificação de mensagens em WSDL

[49] **COM** (*Component Object Model* ou Modelo Componente de Objeto) é uma plataforma da Microsoft (http://www.microsoft.com) para componentes de software lançada em 1993. Ela é usada para permitir a comunicação entre processos e a criação dinâmica de objetos em qualquer linguagem de programação que suporte a tecnologia.

3.4.2. Transferência de Estado Representacional (Representational State Transfer – REST)

A origem do termo REST vem da tese de Roy Fielding que descreve o conceito de Transferência de Estado Representacional. REST é um estilo arquitetônico que pode ser resumido em quatro verbos (GET, POST, PUT e DELETE do HTTP 1.1) e os substantivos, que são os recursos disponíveis na rede (referenciada na URI[50]). A Figura 3.12 apresenta estes verbos e seus equivalentes operacionais.

HTTP	Equivalência no CRUD
GET	read
POST	create, update, delete
PUT	create, update
DELETE	delete

Figura 3.12 – Verbos utilizados pelo REST e seus equivalentes operacionais

Um serviço para obter os detalhes de um usuário chamado "Silva", por exemplo, seria tratado com um HTTP GET para http://exemplo.com/usuarios/silva. Para excluir o usuário poderia utilizar-se HTTP DELETE, e para criar um novo poderia se utilizar um POST. A necessidade de se referenciar a outros recursos poderia ser obtida utilizando hiperlinks e separando as solicitações HTTP das respostas.

Os principais objetivos do REST incluem:

- Escalabilidade de interações entre os componentes;
- Generalidade de interfaces;
- Implantação dos componentes independente;
- Componentes intermediários para reduzir a latência, reforçar a segurança e encapsular sistemas legados.

[50] *Uniform Resource Identifier* (**URI**) é uma seqüência de caracteres utilizada para identificar um nome ou um recurso na Internet. Essa identificação permite a interação com representações do recurso através de uma rede (normalmente a *World Wide Web*), usando protocolos específicos.

3.4.2.1. Elementos Arquiteturais do REST

O estilo REST é uma abstração dos elementos arquitetônicos dentro de um sistema hipermédia distribuído. REST ignora os detalhes da implementação do componente e sintaxe do protocolo a fim de orientar sobre os papéis dos componentes, as restrições sobre a sua interação com outros componentes, e sua interpretação de elementos significativos de dados. Ela abrange as restrições fundamentais sobre os componentes, conectores e dados que definem a base da arquitetura da Web e, portanto, a essência do seu comportamento como um aplicativo baseado em rede (Fielding, 2000).

3.4.2.2. Elementos de dados

Os elementos de dados do REST estão resumidos na Tabela 3.1.

Tabela 3.1 – Elementos de Dados do REST

Elemento de dados	Exemplos Web
Recurso	O alvo conceitual pretendido de uma referência de hipertexto
Identificador de recurso	URL, URN
Representação	Documento HTML, imagem JPEG
Metadados de Representação	Tipo de mídia, hora da última alteração
Metadados de Recursos	*Link* com a fonte
Dados de controle	Controle de *cache*, Se-modificado-desde

3.4.2.2.1. Recursos e Identificadores de recursos

A chave para abstração de informação em REST é o recurso. Qualquer informação que possa ser nomeada pode ser um recurso: um documento ou uma imagem, um serviço de clima (por exemplo, "tempo de hoje em São Paulo"), uma coleção de outros recursos, um objeto não virtual (por exemplo, uma pessoa), e assim por diante. Em outras palavras, qualquer conceito que possa ser alvo de referência de um autor de hipertexto deve caber dentro da definição de um recurso. Um recurso é um mapeamento conceitual para um conjunto de entidades, e não a entidade que corresponde ao mapeamento em qualquer ponto específico no tempo (Fielding, 2000).

REST utiliza um identificador de recurso para identificar um recurso em especial envolvido em uma interação entre os componentes. Conectores REST fornecem uma interface genérica para acessar e manipular o valor ajustado de um recurso, independentemente de como a função de pertinência é definido ou o tipo de software que está a tratar do pedido. A autoridade de nomeação que atribui o identificador de recurso, tornando possível a referência ao recurso, é responsável por manter a validade semântica do mapeamento ao longo do tempo (ou seja, garantir que a função membro não se mude).

3.4.2.2.2. Representações de dados

Os componentes REST executam ações em um recurso usando uma representação para capturar o estado atual ou a que se destina a transferência de recursos e que representação entre os componentes. A representação é uma sequência de *bytes*, além de representação de metadados para descrever esses bytes. Outros comumente usados, mas nomes menos precisos para a representação são: documentos, arquivos e entidade da mensagem HTTP, instância ou variantes.

Uma representação de dados consiste de dados e metadados que descrevem os dados, e, ocasionalmente, os metadados para descrever os metadados (geralmente com a finalidade de verificar a integridade da mensagem). Os metadados estão sob a forma de pares nome-valor, onde o nome corresponde a uma norma que define a estrutura do valor e da semântica. As mensagens de resposta podem incluir uma representação de metadados e os metadados de recursos: informações sobre o recurso que não é específico para a representação fornecida (Fielding, 2000).

3.4.2.3. Conectores

REST utiliza diversos tipos de conectores para encapsular as atividades de acesso aos recursos e transferência de representações de recursos. Os conectores apresentam uma interface abstrata para a comunicação de componentes, aumentando a simplicidade, oferecendo uma clara separação das preocupações e escondendo a implementação básica de recursos e mecanismos de comunicação. A generalidade da interface também permite a substituição: se o acesso dos usuários apenas para o sistema é através de uma interface abstrata, a aplicação pode ser substituído sem afetar os

usuários. Uma vez que um conector gerencia a comunicação de rede para um componente, a informação pode ser compartilhada entre múltiplas interações, a fim de melhorar a eficiência e a capacidade de resposta.

3.4.2.4. Componentes

Componentes do REST, apresentados na Tabela 3.2, são categorizados pelo papel que exercem no âmbito geral de uma aplicação.

Tabela 3.2 – Componentes do REST

Componente	Exemplos da Web
Servidor de origem	Servidor HTTP(Apache, IIS)
Gateway	Squid, CGI, Proxy reverso
Proxy	CERN Proxy, Gauntlet
Agente do usuário	Navegador (Firefox, Safári, Internet Explorer, Chrome)

O agente do usuário utiliza um conector do tipo cliente para iniciar uma requisição e, em seguida, se torna o destino final de uma resposta. O tipo mais comum do de um agente do usuário é o navegador Web, que provê acesso a informações (recursos) e processa as respostas (representações) de forma que a ser visualizada pelo usuário de acordo com a necessidade (Melgarejo, 2007).

Um servidor de origem utiliza um conector do tipo servidor e trata das requisições de um cliente. É o servidor, também, a fonte de todas as representações dos seus recursos disponíveis aos clientes. Cada servidor deve prover seus serviços através de uma interface genérica para a sua hierarquia de recursos, detalhes de implementações destes recursos ficam escondidos por esta interface. Outra característica chave de um servidor de origem é ele ser o destino final de qualquer requisição feita por um cliente e que visa modificar (Fielding, 2000).

Os outros dois componentes, gateway e proxy, podem atuar ao mesmo tempo, como cliente e servidor para a exercer a funcionalidade de encaminhamento de requisições e respostas. Mais especificamente, o proxy é um componente intermediário explicitamente escolhido pelo cliente para exercer encapsulamento de outros serviços, deslocamento de dados, aumento de performance ou algum tipo de controle de acesso

de segurança. Já o gateway é transparente ao usuário e utilizado pelo servidor de origem dos recursos, funciona como um proxy reverso, provê os mesmos serviços que um proxy porém, a diferença entre os dois é que o proxy é utilizado (ou não) explicitamente pelo usuário, enquanto que o gateway é imposto pelo servidor.

3.5. ORQUESTRAÇÃO – UTILIZANDO SERVIÇOS PARA COMPOR PROCESSOS DE NEGÓCIO

A orquestração fornece recursos associados com a composição e execução coordenada de serviços. A tecnologia de orquestração é baseada também em middleware 51 e pode se tornar uma parte altamente centralizada da arquitetura que rege a concepção de definições de processos de negócios, bem como a execução da lógica de processos de negócios.

Segundo (Erl e Loesgen, 2010), espera-se dos ambientes de orquestração modernos suporte para composição de serviços com lógicas sofisticadas e complexas que podem resultar em atividades de execução extensa.

Na Figura 3.13, o Serviço A é um serviço composto que tem uma interface (IServiceA) exposta que está disponível para o Agente Consumidor e a sua implementação se baseia em dois outros serviços. O Agente Consumidor não sabe que os serviços atômicos B e C são utilizados pelo serviço A, ou se eles são executados em paralelo ou serialmente, ou se a execução do mesmo ocorreu como sucesso ou não. O Agente Consumidor apenas se interessa com o resultado da execução do serviço A. As interfaces dos serviços B e C (IServiceB e IServiceC) não precisam necessariamente ficar escondidas do Agente Consumidor somente pelo fato destes serviços serem usados como partes da composição do Serviço A. Neste exemplo, não há nenhuma razão prática que impeça que o Agente Consumidor não interaja com o Serviço B ou C em um algum outro cenário.

51 **Middleware** é o neologismo criado para designar camadas de software que não constituem diretamente aplicações, mas que facilitam o uso de ambientes ricos em tecnologia da informação.

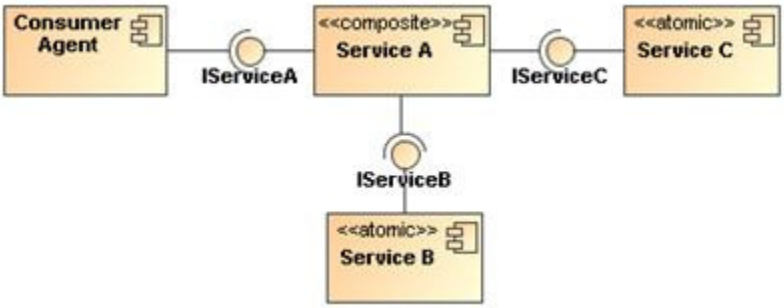

Figura 3.13 – Modelo de composição de serviço. Fonte: (Estefan et al., 2009)

Fundamentalmente a orquestração é composta pela coexistência das aplicações de abstração de processos, repositório de estados, centralização de processos e serviços de compensação de transação, e pode ser posteriormente estendida com as aplicações de serviço de transações atômicas, centralização de regras (*rules*), e modelos de transformação de dados (Erl, Losegan, 2006).

A orquestração deve fornecer um mecanismo dinâmico, flexível e adaptável para atender às necessidades de mudança de domínio. Isto é alcançado através da separação da lógica do processo dos serviços empregados. A natureza de baixo acoplamento da orquestração é fundamental, já que não é um requisito que todos os serviços estejam prontos para execução ao mesmo tempo a fim de executar as orquestrações. Isto também é essencial para as operações de longa execução. Além disso, à medida que os serviços mudam ao longo do tempo, geralmente não há necessidade de se alterar a camada de orquestração para acomodar estas mudanças se forem devidamente arquitetados.

Assim, pode-se considerar a orquestração realmente como outra camada completa, acima de abordagens mais tradicionais de integração de aplicativos, incluindo integração de informações e orientação a serviços. A orquestração encapsula esses pontos de integração, vinculando-os para formar os processos de nível superior e os serviços compostos.

Na prática, uma orquestração é tipicamente implementada usando a abordagem de scripts para compor os processos de negócio orientados a serviço. Normalmente envolve o uso de scripts escritos em linguagens baseadas em padrões, tais como

BPEL[52], WS-CDL[53] (*Web Services Choreography Description Language*) ou YAWL[54] (*Yet Another Workflow Language*).

Na Figura 3.14 apresentamos um exemplo de orquestração representado pelo um diagrama de atividade UML[55] (*Unified Markup Language*) para modelar um processo de negócio orientado a serviço. O modelo permite visualizarmos os principais elementos dos processos de negócio, tais como o conjunto de tarefas relacionadas a ser executo, a ligação entre as tarefas através do fluxo lógico, os dados que são passados entre as tarefas e qualquer outra regra relevante que governa a transição entre as tarefas. Uma tarefa (*task*) é uma unidade de trabalho que um indivíduo, sistema ou organização realiza e que pode ser realizado em uma ou mais etapas ou subtarefas.

[52] **BEPL** (*Business Process Execution Language* ou Linguagem de Execução de Processos de Negócio) é um dialeto executável do XML que facilita a modelagem das interações entre os serviços Web.
[53] **WS-CDL** (*Web Services Choreography Description Language* ou Linguagem descritiva de coreografia de Serviços Web) é uma recomendação da W3C(http://www.w3.org) e utiliza linguagem utiliza XML.
[54] **YAWL** (*Yet Another Workflow Language* ou Mais uma linguagem de workflow) é um sistema de fluxo de trabalho (workflow) e é baseado em uma linguagem concisa e poderosa de modelagem. YAWL manipula dados complexos, transformações, e permite a integração com os recursos de organização e Web Services.
[55] **UML** (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma linguagem de modelagem padrão de uso geral no domínio da engenharia de software. O padrão é gerido, e foi criado pelo OMG (*Object Management Group*).

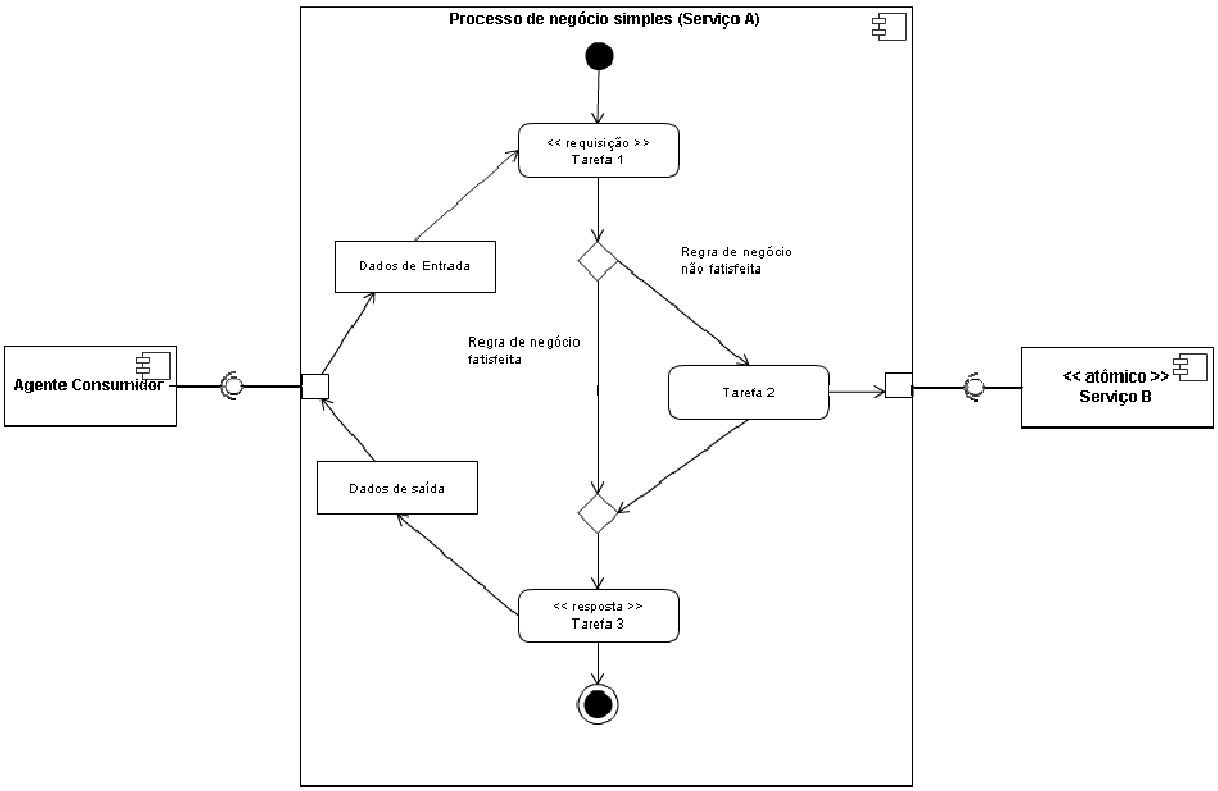

Figura 3.14 – Exemplo abstrato de orquestração de um processo de negócio orientado a serviço. Fonte: (Estefan et al., 2009)

3.6. NOTAÇÃO DE MODELAGEM DE PROCESSOS DE NEGÓCIO (*BUSINESS PROCESS MODELING NOTATION* – BPMN)

BPMN (*Business Process Modeling Notation*, Notação de Modelagem de Processos de Negócio) é uma notação gráfica que define as etapas em um processo de negócio, através de um conjunto padrão de elementos de diagramas (OMG, 2005).

Estes elementos possibilitam o desenvolvimento fácil de diagramas simples que irão parecer familiares à maioria dos analistas de negócio. Estes elementos foram escolhidos para serem distinguíveis uns dos outros e utilizam formas que são familiares à maioria dos modeladores. Por exemplo, as atividades são representadas por retângulos e as decisões são losângulos.

Deve ser enfatizado que um dos objetivos no desenvolvimento do BPMN foi criar um mecanismo simples para a criação de modelos de processos de negócios, enquanto ao mesmo tempo fosse capaz de lidar com a complexidade inerente a estes

processos. A abordagem para lidar com essas duas exigências conflitantes foi organizar os aspectos gráficos da notação em categorias específicas (White, 2008).

As quatro categorias básicas dos elementos são (Figura 3.15):

- Objetos de fluxo (*flow objects*);
- Objetos de conexão (*conecting objects*);
- Raias (*swinlanes*);
- Artefatos (*artifacts*).

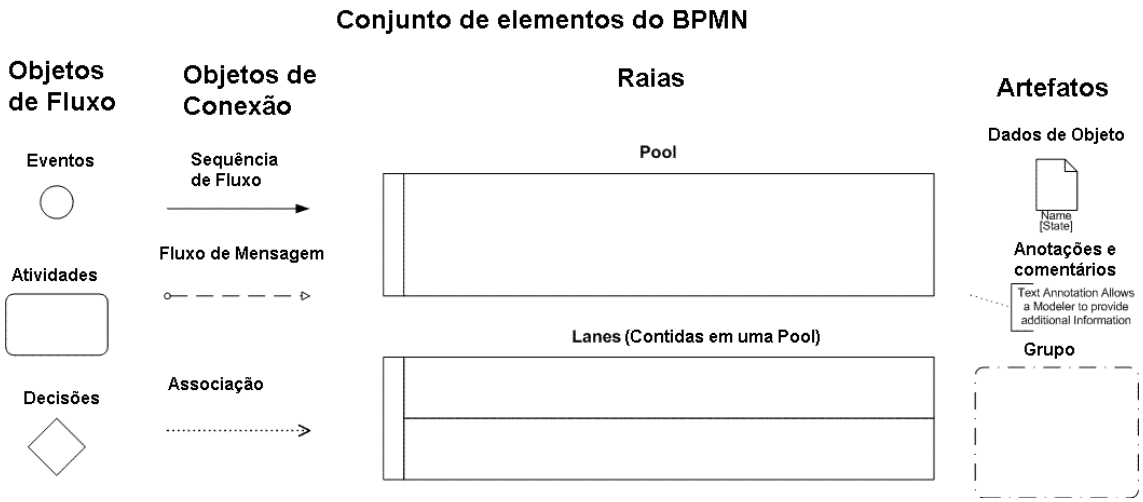

Figura 3.15 – Adaptado de Conjunto de elementos principais do BPMN. Fonte: OMG, 2005

3.7. LINGUAGEM DE EXECUÇÃO DE PROCESSOS DE NEGÓCIO (*BUSINESS PROCESS EXECUTION LANGUAGE* – BPEL)

BPEL (*Business Process Execution Language*) se tornou uma das mais importantes tecnologias de SOA e permite uma composição fácil e flexível de serviços em processos de negócios. BPEL é particularmente importante porque introduz um novo conceito de programação de sistemas. Este conceito permite-nos desenvolver processos rapidamente por definir a ordem em que os serviços serão chamados. Desta forma, sistemas, aplicações e informações se tornam mais flexíveis e podem se adaptar melhor às mudanças nos processos empresariais (Juric, 2006).

Para uma abordagem SOA eficiente na automação dos processos de negócio, são necessárias:

- Forma padronizada para expor e acessar a funcionalidade de aplicações como serviços;
- Infraestrutura de barramento de integração e gestão de serviços, incluindo a interceptação de mensagens, roteamento, transformação, etc.;
- Linguagem especializada para a composição das funcionalidades das aplicações expostas nos processos de negócio.

O primeiro requisito é cumprido através da mais recente arquitetura distribuída: *Web Services*. O segundo requisito é cumprido através do ESB (*Enterprise Service Bus, Barramento Corporativo de Serviços*), que fornece suporte para a gestão centralizada, declarativa, e bem coordenada dos serviços e suas comunicações. O terceiro requisito, a composição de serviços em processos, é preenchido por BPEL, a linguagem especializada comumente aceita para a definição de processos de negócios e de execução.

Um processo de negócios visto por BPEL é uma coleção de invocações de serviços (Figura 3.16) coordenados e atividades relacionadas que produzem um resultado, dentro de uma única organização ou em várias. Por exemplo, um processo de negócio para planejar viagens de negócios irá invocar vários serviços. Em um cenário simplista, o processo de negócio vai exigir-nos especificar o nome do empregado, destino, datas e detalhes de viagem. Em seguida, o processo irá invocar um *Web Service* para verificar o status do empregado. Com base no *status* do funcionário irá selecionar a classe de viagem adequada. Em seguida, ele irá invocar os *Web Services* de diversas companhias aéreas para verificar o preço de passagem área e comprar aquele com o menor preço.

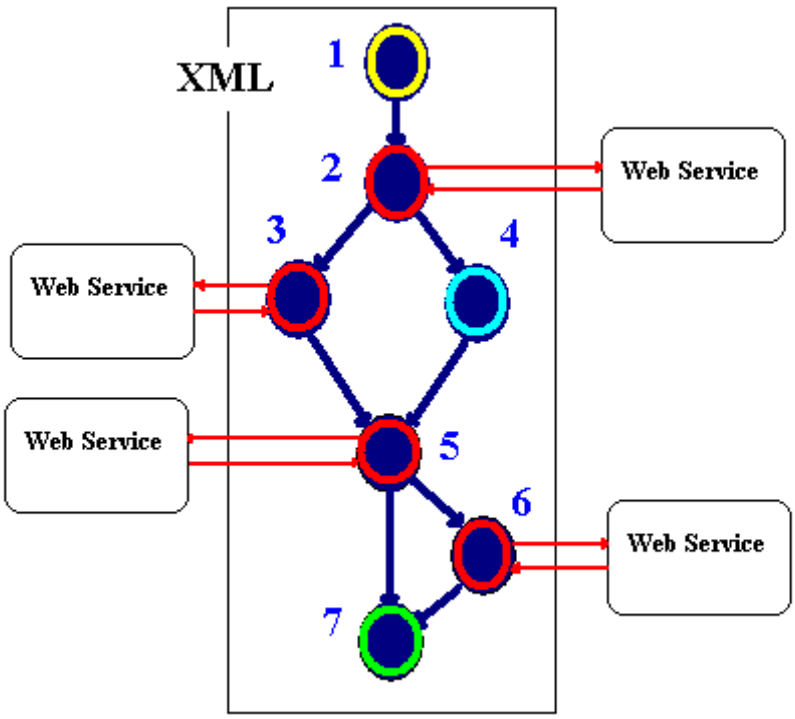

Figura 3.16 – Exemplo de BPEL

É importante notar que estas tecnologias relacionadas ao SOA são recentes e estão em evolução. O BPEL, por exemplo, possui alguns problemas de desempenho e conseqüente necessidade de máquinas super dimensionadas. Percebe-se que as empresas desenvolvedoras estão a cada dia melhorando o desempenho dos motores (*engines)* que suportam esta linguagem.

3.8. O BARRAMENTO DE SERVIÇOS CORPORATIVO (*ENTERPRISE SERVICE BUS* - ESB)

Um *Enterprise Service Bus* (ESB) é um sistema *middleware* distribuído para a integração de ativos de TI das empresas através de uma abordagem orientada a serviços. Um ESB é uma infraestrutura distribuída usada para integração corporativa. É constituída por um conjunto de contêineres de serviço, que integram os vários tipos de ativos de TI. Os contêineres são interligados com um barramento de mensagens confiável. Os contêineres de serviços adaptam os ativos de TI para um modelo de serviços padrão, com base na troca de mensagens XMLs padronizados. O ESB fornece serviços de transformação e roteamento de mensagens, bem como a capacidade de administrar centralmente o sistema distribuído. Nas seções anteriores foi dada uma

visão do SOA e seus principais elementos e tecnologias envolvidas. O entendimento destas tecnologias e conceitos é fundamental para a compreensão da entrega de serviços de telecomunicações com SOA que será apresentado no próximo capítulo.

4. ENTREGA DE SERVIÇOS DE TELECOM COM SOA

No capítulo 2 foi mostrada a importância dos OSS para uma empresa de telecomunicações através do histórico e evolução. O capítulo 3 mostrou o conceito de SOA, serviços e tecnologias e explanou os oito princípios através de uma ótica de reaproveitamento dos ativos existentes na organização. Neste capítulo utilizaremos as idéias, conceitos e tecnologias apresentados nos capítulos anteriores como base para entendimento da entrega de serviços de Telecom utilizando SOA.

Na próxima seção será apresentado o conceito de Plataforma de Entrega de Serviços e em seguia o CMMI que será utilizado como modelo de referência para avaliação da maturidade na transição para o SOA na seção 4.2.

4.1. PLATAFORMA DE ENTREGA DE SERVIÇOS (*SERVICE DELIVERY PLATFORM* - SDP)

O conceito de SDP é muito simples. Ela expõe a capacidade dos sistemas com interface unificada, reduz os custos de integração e permite que a organização possa implementar novos requisitos rapidamente, ou seja, o SDP permite que serviços e conteúdos criados e desenvolvidos por terceiros possam ser ofertados aos clientes de uma operadora de forma padronizada e organizada e com um mínimo de esforços para adequação à rede específica da operadora sejam necessários por parte da TI, e sem que a operadora precise entrar a fundo nos processos de autorização e aprovisionamento ligados a cada serviço (Terry, 2009).

Além disso, o SDP fornece serviços comuns, como faturamento, cobrança que elimina a duplicação da implementação. Este conceito é exatamente o combinado com SOA. Assim podemos chamar SDP como a "versão Telco específicas de SOA".

O SDP é a plataforma que padroniza a “conexão” de serviços de terceiros de forma transparente aos sistemas e redes de uma operadora, permitindo a orquestração de serviços e o uso de diferentes conteúdos para a oferta de serviços complexos e personalizados (Vuono, 2010).

4.1.1. Arquitetura SDP – Componentes-chave

A Figura 4.1 mostra os blocos principais de uma arquitetura de referência SDP (*Service Enablement Layer, Service Components e Service Enablement Layer*).

Mantiveram-se os nomes das componentes principais em inglês devido a estes termos terem a sua consagração global nesta língua.

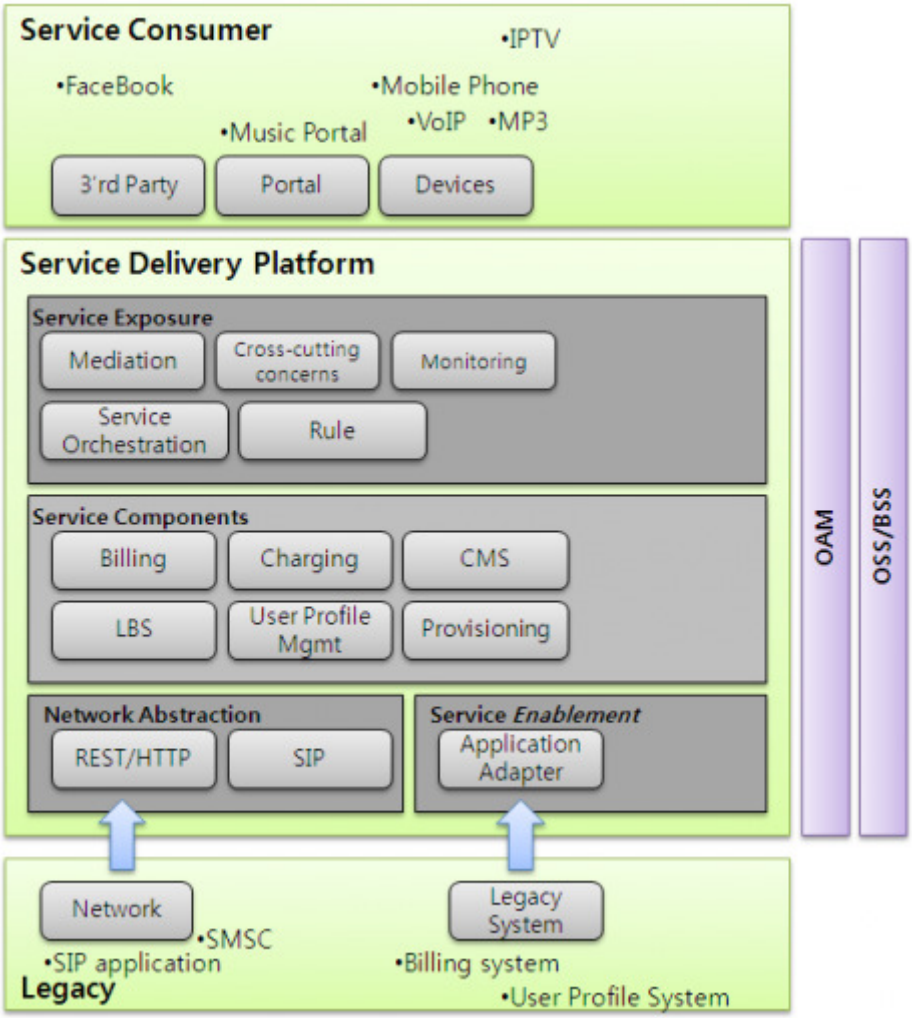

Figura 4.1 – Arquitetura de um SDP – Fonte: Terry, 2009.

O Plano de Entrega de Serviços é composto de três camadas:

SERVICE ENABLEMENT LAYER – são serviços não diretamente solicitados pelo usuário final, mas que têm uma importância vital na composição do serviço sendo ofertado. Eles permitem às aplicações concentrar-se em prover valor ao usuário final, aliviando-as do custoso processo de implementação e gerenciamento de características de baixo nível, horizontais e reutilizáveis por diversas aplicações/serviços (Vuono, 2009). Aplicações baseadas em rede, como o SMS e VoIP, tem protocolo próprio. A

camada *Service Enable* abstrai estas aplicações e expõe as capacidades com um protocolo de mensagem unificado. Ao fazer isso o usuário pode facilmente utilizar as aplicações de rede sem ter que utilizar as tecnologias específicas de telecomunicações (Terry, 2009).

SERVICE COMPONENTS - é o sistema de software que fornece capacidades ou funcionalidades. Assim como os serviços SOA. Em SDP existem os serviços comuns para o sistema de telecomunicações. Por exemplo: Gestão de Conteúdos, faturamento, *billing,* gestão do perfil de usuários, serviços baseados em localização.

SERVICE EXPOSURE – *os c*omponentes de serviço são expostos por esta camada. Esta camada fornece infraestrutura de serviço. Por exemplo, ela realiza a orquestração dos serviços existentes para criar um novo serviço.

4.1.2. Interação de serviços (SDP e SOA)

O SDP deve seguir os princípios da Arquitetura Orientada a Serviços (SOA) para integração de sistemas internos e externos (OSS / BSS), bem como gestão de serviços e orquestração. Esta exigência aplica-se principalmente para soluções abrangentes SDP. Hoje, porém, não é recomendável que uma empresa de Telecom implemente uma plataforma SDP ou qualquer outro serviço se não for aderente ao SOA. O SDP proporciona um modelo de referência de SOA de Telecom, com uma taxonomia definida para o domínio das telecomunicações.

Segundo Vuono(2009):

> *"O SDP baseia-se em SOA e nos seus princípios aplicados ao mundo das telecomunicações. A situação atual da SOA ainda não está totalmente clara, e "SOA" é, sem duvida, um dos acrônimos mais reconhecidos atualmente. Isso se deve ao fato de que a Arquitetura Orientada para Serviços tem potencial para habilitar um grau muito maior de agilidade de negócios com base em um modelo de entrega voltado para serviços.*
>
> *O enfoque multi-camada permite que novos serviços possam ser criados e testados utilizando ambientes padronizados e pré-aprovados de desenvolvimento. Qualquer interação de serviço utilizada no SDP garante que*

as regras apropriadas de negócio / acesso sejam aplicadas a todos os ISVs (Independent Software Vendor) ou provedores de serviço terceiros, assegurando uso autenticado e controlado da rede e para prevenir que desenvolvedores terceiros não-confiáveis sobrecarreguem elementos centrais da rede.

Este modelo de interação de arquitetura orientada a serviços (SOA) oferece grande flexibilidade e inovação, produzindo novos serviços especificamente desenhados de acordo com preferências únicas do usuário final, e alinhado com a estratégia de negócio da operadora de rede. Tal alinhamento permite o foco em categorias específicas de assinantes com pacotes compostos de serviços desenvolvidos de acordo com seus negócios ou mercado. Estes sofisticados serviços para usuários finais alavancam mais enablers de serviços tais como messaging, presença, ou localização. A Figura 4.2 mostra a evolução da convergência das arquiteturas utilizadas pelas empresas de telecomunicações."

A única diferença entre o SDP e SOA é que SDP requer uma camada de abstração de rede (*Network Abstraction Layer*). Em SOA geralmente tem-se a camada de ativação do serviço (*Service Enablement layer*). Esta camada converte o protocolo legado (SAP, MQ, CORBA, etc.) para a plataforma SOA unificada (Web Services, etc.) Geralmente é implementada com adaptadores (JCA[56], por exemplo) ou outras camadas de habilitação serviços. O SDP tem a mesma camada de abstração do SOA. Se em SOA geralmente integra-se com as aplicações legadas, o SDP integra-se com aplicações de rede legadas (SIP, SMSC[57], Paray-X, etc.) Assim, em SDP há uma camada de abstração de rede com adaptadores de rede.

[56] A J2EE *Connector Architecture* (**JCA**) fornece uma solução de tecnologia Java para o problema da conectividade entre os servidores de aplicação e muitos sistemas atuais de informação empresarial .

[57] **SMSC** (Short Message Service Center ou Centro de Serviços de Mensagens Curtas) é um elemento de rede da telefonia móvel que entrega o serviços de mensagens curtas (SMS)..

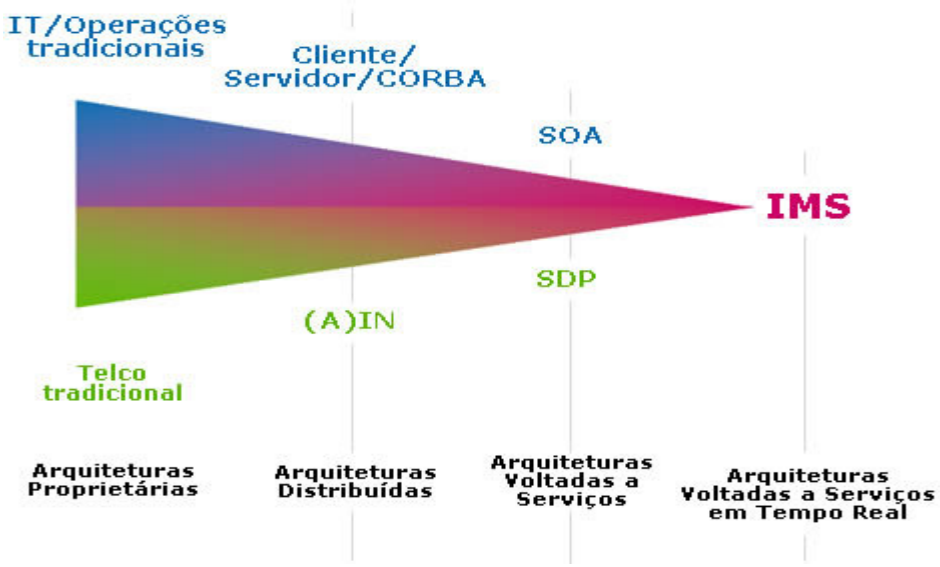

Figura 4.2 – Evolução das arquiteturas das operadoras de telecomunicações. Fonte: (Vuono,2009)

Na próxima seção, será apresentado um processo, em fases, para realização da transição da arquitetura atual dos sistemas de suporte a operação para uma arquitetura orientada a serviços.

4.2. PROCESSO DE TRANSIÇÃO PARA SOA

Podemos identificar os princípios fundamentais do SOA como princípios que irão nortear a avaliação da adequação dos recursos existentes para a orientação a serviços (Erl, 2005).

Segundo Harrison e Shiron (1999), para que seja possível verificar a eficácia de uma organização, é necessária a adoção de Modelo de Referência para viabilizar a comparação da situação desejada com as informações coletadas na organização.

Assim, a proposta deste trabalho é criar uma abordagem para aplicação dos conceitos de SOA nos OSS através de um processo dividido em cinco fases, que propõe a cada fase, assim como em modelos de avaliação maturidade como o CMMI, metas, práticas e atributos específicos a cada uma. A partir da referência de um estudo realizado por Wambecq, et. al. (2006), neste trabalho será proposto um modelo em

cinco fases e que para cada fase, será atribuído um nível de maturidade que ajudará na medicão do amadurecimento da gestão dos serviços nas áreas que compõem os OSS, nesta nova arquitetura orientada a serviços. Vale salientar que a organização pode optar pela priorização de uma das áreas do OSS (*Order Fulfillment, Service Assurance e Billing*) ao contrário da implantação em todas. Por exemplo, uma operadora pode decidir priorizar o *Billing* por constatar perda de receita devido a falhas de processamento e/ou ineficiência nesta área. A experiência mostra que as organizações funcionam melhor quando concentram seus esforços de melhoria de processos em um número controlado de áreas de processos que exigem um esforço cada vez mais sofisticado à medida que a organização melhora (CMMI-SE/SW, 2006).

Nível 1 - Inicial: No nível de maturidade 1 os serviços iniciais representam a aprendizagem e fase iniciais do projeto de adoção do SOA. A quase totalidade dos serviços de Telecom está estruturada verticalmente: além do serviço e da lógica de negócios, cada serviço tem suas próprias funções de gerenciamento, execução e interfaces com os ambientes OSS e BSS. Neste nível, os projetos são tipicamente realizados para responderem a uma necessidade específica ao mesmo tempo em que são experimentadas tecnologias específicas.

O ambiente evoluirá para uma arquitetura de baixo acoplamento em que blocos de serviços reutilizáveis (chamados de facilitadores ou *service enablers* abordados na seção 4.1.1) são montados em aplicações. Esta transição irá provavelmente acontecer em várias etapas para que se possam resguardar os investimentos feitos anteriormente.

As principais metas e práticas específicas do nível 1 estão na tabela Tabela 4.1 e os principais atributos na Tabela 4.2.

Tabela 4.1 – Principais metas e práticas específicas do nível 1

Nível de Maturidade	Principais metas específicas	Principais práticas específicas
1. Serviços Iniciais	1. Aprender SOA e tecnologias envolvidas através de pesquisa e desenvolvimento com projetos piloto; 2. Aplicar as tecnologias SOA na resolução de necessidades da organização relativas aos OSS; 3. Definir o ROI (*Return On Investiment*) para projetos SOA e aplicar em projetos iniciais.	1. Priorizar uma das áreas do OSS para definição do projeto piloto e definir os serviços reutilizáveis (facilitadores ou *service enablers*) 2. Integrar o SOA em metodologias de projetos de desenvolvimento; 3. Quantificar custos, tempos e benefícios de negócio dos projetos pilotos.

Tabela 4.2 – Principais atributos do nível 1

Nível de Maturidade	Benefícios principais de Negócio	Âmbito	Fatores Críticos de Sucesso da Tecnologia	Fatores Críticos de Sucesso de Pessoas e Organização	Padrões Relevantes Selecionados
1. Serviços Iniciais	Nova funcionalidade	Pesquisa e desenvolvimento, projetos piloto, portal, site, número pequeno de serviços	Padrões, integração de sistemas legados	Os desenvolvedores adquirem competências; Promoção da Gestão de desenvolvimento.	WSDL, XML, XSLT, SOAP, JEE

Nível 2 - Gerenciado: Neste nível de maturidade, os requisitos, processos, produtos de trabalho e serviços são gerenciados. A situação dos produtos de trabalho e a entrega dos serviços são visíveis para o gerenciamento em pontos definidos (por exemplo, nos marcos principais e no momento em que as principais tarefas são completadas). As áreas dos OSS devem ser elevar de um nível informal para um nível disciplinado estabelecendo um real gerenciamento de projetos voltados para SOA.

As aplicações dos OSS existentes irão expor algumas de suas funções em uma camada de integração (um barramento de integração, por exemplo). Este ambiente de integração permite que o provedor empacote o serviço: isto garante que o usuário final

irá experimentar um comportamento consistente através de diferentes aplicações. Por exemplo, a lógica de integração pode agregar funções específicas da aplicação de auto-gestão em um portal único e consistente para todos os serviços incluídos no pacote. Cada aplicativo do pacote requer informações do usuário (possivelmente em diferentes formatos) para estar disponível no seu ambiente de aplicação: esta informação é abastecida através de um processo na camada de orquestração.

O provedor dos serviços estabelece acordos com os clientes, bem como desenvolve e geri clientes e exigências contratuais. Configuração e processos de gestão e garantia de qualidade do produto são institucionalizadas, e o prestador do serviço também desenvolve a capacidade de medir e analisar o desempenho do processo.

Novas aplicações podem ser criadas como base em aplicativos existentes e terceiros os quais a interação entre eles é coordenada através de uma orquestração. É necessária uma camada específica para integração de funções de aplicações específicas de OSS/BSS. Os serviços devem ser arquitetados e os padrões especificados para a gestão técnica da implementação do SOA (tipicamente sob a liderança da área de Arquitetura de TI).

Na definição do ambiente de integração deve conter ferramentas e componentes para integrar rapidamente os serviços de valor agregado com sistemas internos, aprovisionamento e outros Sistemas de Suporte a Operação (OSS). Estas ferramentas e componentes incluem uma série de adaptadores que expõem as aplicações existentes de OSS como *Web Services*. Este ambiente deve incluir também serviços de metadados[58], identificação para integração, um mecanismo de orquestração e *templates* para configuração da lógica de negócio.

Na Tabela 4.3 apresentamos as principais metas e práticas específicas do nível 2 e os principais atributos na Tabela 4.4.

Tabela 4.3 – Quadro com as principais metas e práticas específicas do nível 2

Nível de Maturidade	Metas específicas	Práticas específicas
2. Serviços de Arquitetura	1. Criar uma cultura de SOA	1. Especificar padrões de

[58] **Metadados** são dados que descrevem completamente os dados (bases) que representam, permitindo ao usuário decidir sobre a utilização desses dados da melhor forma possível. São dados que permitem informar as pessoas sobre a existência de um conjunto de dados ligados às suas necessidades específicas. (ALMEIDA, 1998)

	nas áreas dos OSS; 2. A Arquitetura deve ser focada para SOA; 3. Antecipar a utilização das informações SOA para otimização do negócio.	tecnologia para SOA; 2. Integrar o SOA em processos de desenvolvimento nas áreas dos OSS; 3. Providenciar formação SOA no âmbito dos OSS; 4. Utilizar abordagem de integração incremental.

Tabela 4.4 – Principais atributos do nível 2

Nível de Maturidade	**Benefícios principais de Negócio**	**Âmbito**	**Fatores Críticos de Sucesso da Tecnologia**	**Fatores Críticos de Sucesso de Pessoas e Organização**	**Padrões Relevantes Selecionados**
2. Serviços de Arquitetura	Redução de custos e controle de TI nas áreas dos OSS	Múltiplas aplicações de OSS integradas	Suporte a sistemas heterogêneos e distrubuídos, facilidade de entrega, integração de base de dados, versionamento, segurança interna, gestão de desempenho	A equipe de Arquitetura deve liderar o processo. Criar um centro de compotência SOA; A alta direção deve patrocinar.	UDDI, SAML,WS-Security, WS-Addressing, WS-ReliableMessaging, WS-Policy, XQuery, BPEL, BPMN

Nível 3 - Definido: Neste nível os processos são bem caracterizados e compreendidos, e são descritos em normas, procedimentos, ferramentas e métodos. O conjunto padronizado de processos das áreas dos OSS, que é a base para nível de maturidade 3, é estabelecido e melhorado ao longo do tempo. Estes padrões de processos são utilizados para estabelecer a coerência em todas as áreas dos OSS. Os projetos estabelecem seus processos definidos adaptando o conjunto de processos padrão das áreas dos OSS de acordo com as instruções de adaptação (adaptado de CMMI-SE/SW, 2006).

A infra-estrutura é evoluída para aumentar a flexibilidade para criar, entregar e cobrar pelos serviços fixos e móveis. A exposição não deve ser limitada a aplicações somente, os componentes de Telecom reutilizáveis (facilitadores) também devem ser expostos. As aplicações são construídas a partir da aplicação da lógica de serviço

específica destes componentes facilitadores. Por ter a capacidade de reutilização de serviços comuns, conforme visto no princípio da reutilização, e interfaces bem definidas e padronizadas, os principais benefícios de negócio neste nível incluem a redução dos custos associados ao desenvolvimento e implementação, através da utilização dos componentes e infraestrutura padrões do SOA.

Estes componentes reutilizáveis preferencialmente devem aderir aos padrões abertos da *3G Partnership Project* (3GPP), da *World Wide Web Consortium* (W3C), da *Open Móbile Alliance* (OMA/OneAPI), da *Open Systems Alliance* (OSA/Parlay) e da *Liberty Alliance*. A lista a seguir fornece uma visão geral dos serviços que estão sendo padronizados nas diferentes organizações de normatização: Presença (detectar antecipadamente quem está disponível), Localização, Mensagens, Gerenciamento de Mobilidade (terminal e usuário), Gerenciamento de Sessão, Gestão de Regras e Políticas, Contabilidade (bilhetagem), Cobrança e Faturamento. Estes serviços podem ser comparados aos blocos de Lego® para composição de serviços avançados de Telecom e a utilização destes padrões facilita a integração com soluções de fornecedores diversos.

Os componentes de serviços podem ser hospedados em uma ou mais *Service Delivery Platforms* (SDP). Estes componentes (facilidadores) são expostos como web services (descritos em WSDL). Com a orquestração, as capacidades de diferentes SDP´s podem ser combinadas em novos serviços. Esta abordagem técnica permite o reuso de funções e capacidades, reduzindo os esforços no desenvolvimento e melhorando o *time-to-market* para os novos serviços. As interações inter-SDP são direcionadas para uma camada externa, reduzindo a dependência e aumentando a flexibilidade do serviço em um ambiente com diferentes fornecedores.

O nível três sugere ainda uma parceria entre organizações de negócio e de tecnologia, de forma a assegurar que a utilização do SOA proporcione uma clara capacidade de resposta por parte do negócio. Neste trabalho a idéia é reduzir o escopo para as áreas relacionadas aos OSS. O elemento chave do valor do SOA consiste na ligação entre processos de negócio e processos digitais. O nível três de maturidade do SOA nos OSS é definido por dois caminhos complementares: Serviços de Negócio, focalizados na melhoria dos processos de negócio internos, e Serviços Colaborativos, focalizados na melhoria dos processos colaborativos com parceiros externos. Esta melhoria consiste, principalmente, na padronização dos processos e no maior grau de

detalhamento dos mesmos. A Tabela 4.5 apresenta as principais práticas e metas específicas do nível 3. Na Tabela 4.6 temos os atributos principais.

Tabela 4.5 – Principais práticas e metas específicas do nível 3

Nível de Maturidade	Metas específicas	Práticas específicas
3.a. Serviços de Negócio	1. Criar parcerias entre organizações tecnológicas e de negócio para a gestão SOA; 2. Suportar e padronizar todos os processos de negócio via SOA relacionados aos OSS; 3. Provar o retorno associado à reutilização dos serviços e capacidade de resposta à mudança.	1. Especificar políticas para utilização de SOA na criação ou modificação de processos de negócio das áreas de OSS; 2. Tirar proveito dos eventos orientados e a mediação funcional das tecnologias SOA, especialmente no que diz respeito a recomendações à melhoria e expansão dos processos de negócio.
3.b. Serviços Colaborativos	1. Criar parcerias entre organizações tecnológicas e de negócio para a gestão SOA; 2. Estender os processos de negócio SOA para organizações externas que interagem com os OSS; 3. Provar o retorno da utilização dos serviços para a colaboração;	1. Especificar políticas para a utilização do SOA em colaboração com parceiros; 2. Implementar segurança no nível da Organização.

Tabela 4.6 – Principais atributos do nível 3

Nível de Maturidade	Benefícios principais de Negócio	Âmbito	Fatores Críticos de Sucesso da Tecnologia	Fatores Críticos de Sucesso de Pessoas e Organização	Padrões Relevantes Selecionados
3.a. Serviços de Negócio	Capacidade de resposta do negócio – mudar os processos de negócio rapidamente e efetivamente	Processos de negócio ao longo de unidades de negócio relacionados aos OSS	Reutilização, disponibilidade, facilidade de alteração, regras de processos de negócio, aplicações compostas	Parceria com TIs, parcerias no nível de organizações, gestão do ciclo de vida do SOA, compromentimento da gestão, competência de desenho de "orientação a eventos"	WS-BPEL (ou BPEL4WS), YAWL, padrões abertos da 3GPP, W3C, OMA/OneAPI e OSA/Parlay . SDP
3.b. Serviços Colaborativos	Capacidade de resposta do negócio – colaboração	Serviços disponíveis a parceiros externos que	Disponibilização de serviços externos, tradução de protocolos no	Patrocínio do gestor da unidade de negócio	ebXML, WS-Trust, RosettaNet

	com parceiros	interajam com os OSS	nível organizacional		

Nível 4 – Quantitativamente Gerenciado: Neste nível, a camada de orquestração assume não só o papel de integração de execução e exposição de gestão, mas o mais importante, desempenha um papel essencial na definição e implementação da execução lógica de serviço e gestão do mesmo. Enquanto que o nível três do modelo de maturidade se focaliza na execução dos processos internos e/ou externos do negócio, o nível quatro focaliza-se na medição e na apresentação desses processos ao nível do negócio, de modo a fornecer *feedback* contínuo sobre o desempenho e sobre o impacto de negócio dos processos implementados no nível três. Este nível inclui a atividade de monitoração para permitir que os utilizadores do negócio transformem a forma da resposta aos eventos do negócio. Os serviços e processos relacionados aos OSS devem ser monitorados através de coleta de informações que irão avaliar o desempenho dos mesmos. Qualidade e desempenho dos processos são compreendidos em termos estatísticos e são geridos ao longo da vida dos processos. Para estes processos, as medidas detalhadas do desempenho do processo são coletadas e analisadas estatisticamente. Causas especiais de variação do processo são identificados e, se for o caso, as fontes de causas especiais são corrigidos para evitar ocorrências futuras. No nível 4, o desempenho dos processos é controlado utilizando-se estatística e outras técnicas quantitativas, e é previsível quantitativamente (CMMI-SE/SW, 2006).

A Tabela 4.7 e a Tabela 4.8 apresentam as principais metas/práticas e os principais atributos, respectivamente, do nível 4.

Tabela 4.7 – Principais metas e práticas específicas do nível 4

Nível de Maturidade	Metas Específicas	Práticas Específicas
4. Serviços de Negócio Medidos	1. Instituir a transformação dos processos de negócio de reativo para tempo real; 2. Definir e ir ao encontro do negócio métricas orientadas para o desempenho.	1. Coletar e analisar dos processos de negócio, em tempo real, métricas orientadas para o desempenho. 2. Implementação contínua da avaliação e reengenharia dos processos de negócio.

Tabela 4.8 – Principais atributos do nível 4

Nível de Maturidade	Benefícios principais de Negócio	Âmbito	Fatores Críticos de Sucesso da Tecnologia	Fatores Críticos de Sucesso de Pessoas e Organização	Padrões Relevantes Selecionados
4. Serviços de Negócio Medidos	Transformação do negócio de reativo para tempo real; Coletar métricas de desempenho de negócio	Unidades de negócios relacionadas aos OSS	Monitoração da atividade do negócio, processamento de eventos complexos. Processamento de fluxos de eventos. Alertas e painéis de controle (Dashboards) conduzidos por eventos.	Continuidade da avaliação e resposta de negócio. Patrocínio da diretoria financeira.	BAM (*Business Activity Monitoring*) CEP (*Complex Event Processing*) BEP (*Business Event Processing*)

Nível 5 - Otimizado: Neste nível de maturidade, o objetivo é a otimização dos serviços de negócio. Pode-se utilizar, por exemplo, o ciclo PDCA para a melhoria contínua na composição e entrega dos serviços. Neste nível também é recomendável o acréscimo de respostas automáticas às medidas e às exposições da fase anterior. Desta forma, o sistema de informação do SOA torna-se o "sistema nervoso" das áreas dos OSS e tomam-se decisões de acordo com a ocorrência de eventos no nível do negócio e em conformidade com as regras de otimização dos objetivos estratégicos do negócio. A Tabela 4.9 mostra as principais práticas e metas e a Tabela 4.10 os principais atributos do nível 5.

Tabela 4.9 – Principais metas e práticas específicas do nível 5

Nível de Maturidade	Metas específicas	Práticas específicas
5 Serviços de Negócio Otimizados	1. Providenciar à organização forte liderança para o negócio e à gestão SOA; 2. Provar o retorno do SOA suportado pela melhoria contínua.	1. Implementar processos de negócio auto corretivos.

Tabela 4.10 – Principais atributos do nível 5

Nível de Maturidade	Benefícios principais de Negócio	Âmbito	Fatores Críticos de Sucesso da Tecnologia	Fatores Críticos de Sucesso de Pessoas e Organização	Padrões Relevantes Selecionados
5. Serviços de Negócio Otimizados	Otimização do negócio: reação e resposta automáticas.	Unidades de negócios relacionadas aos OSS	Eventos automaticamente orientados para a otimização.	Cultura da melhoria contínua. Patrocínio da diretoria financeira	

A utilização do modelo proposto permite que se possa sair de uma arquitetura como a apresentada na figura 4.4 para uma arquitetura como na figura 4.5. A primeira mostra uma situação típica onde as aplicações geralmente estão acopladas através de interfaces construídas para atender necessidades específicas de cada aplicação. Mesmo quando há um barramento de integração o que tem se observado é que estas interfaces continuam sendo específicas, impossibilitando a reutilização devido à falta de padronização e contratos bem definidos.

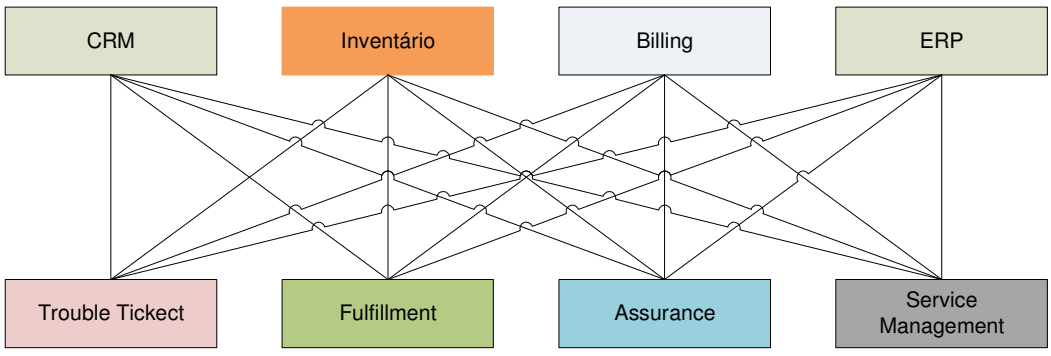

Figura 4.3 – Arquitetura típica de OSS em empresas de Telecom.

A Figura 4.4 mostra, de forma simplificada, um exemplo de OSS com uma Arquitetura Orientada a Serviços esperado após a conclusão das metas e objetivos do

modelo proposto. Através da padronização das interfaces e das técnicas mencionadas para disponibilização dos facilitadores, pode-se identificar e disponibilizar componentes de serviços (C1, C2, C3,..., Cn), nas principais áreas dos OSS (*Trouble Tickect, Fullfillment, Assurance e Service Management*), para a camada de orquestração através do barramento de integração (ESB). As funções dos OSS passam a ser executadas nesta camada, o que possibilita uma flexibilidade maior na criação de serviços. A Figura 4.5 mostra as fases necessárias para atingirmos esta arquitetura e que foram detalhadas anteriormente.

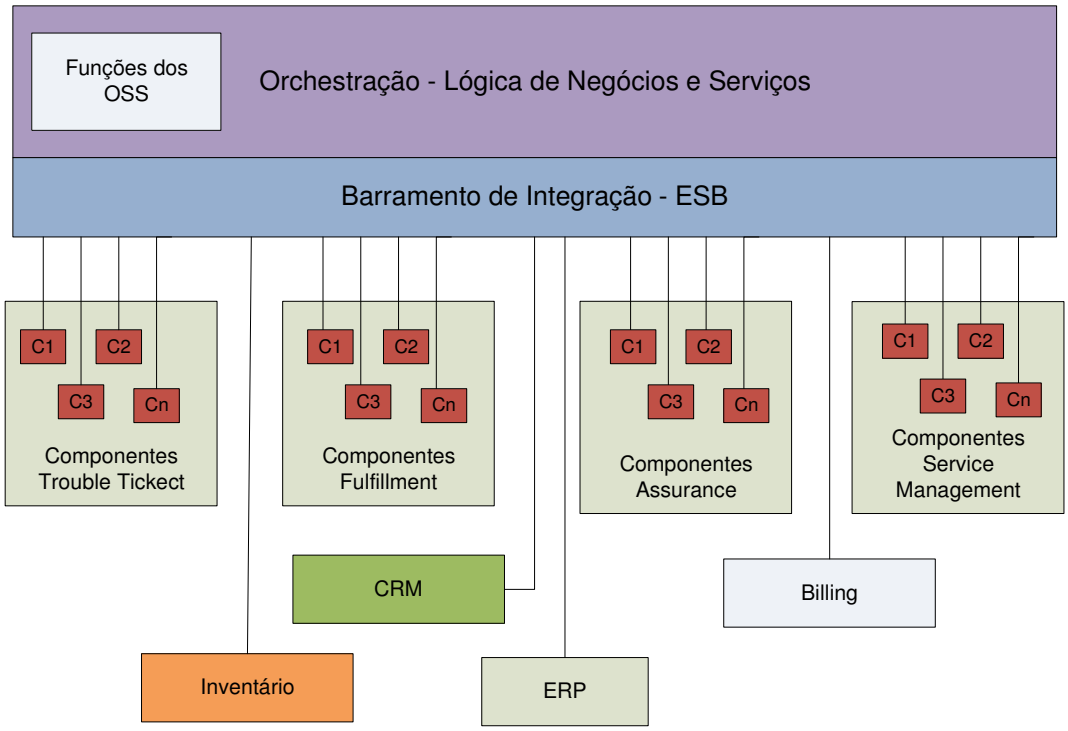

Figura 4.4 – Modelo simplificado de OSS em uma Arquitetura Orientada a Serviços.

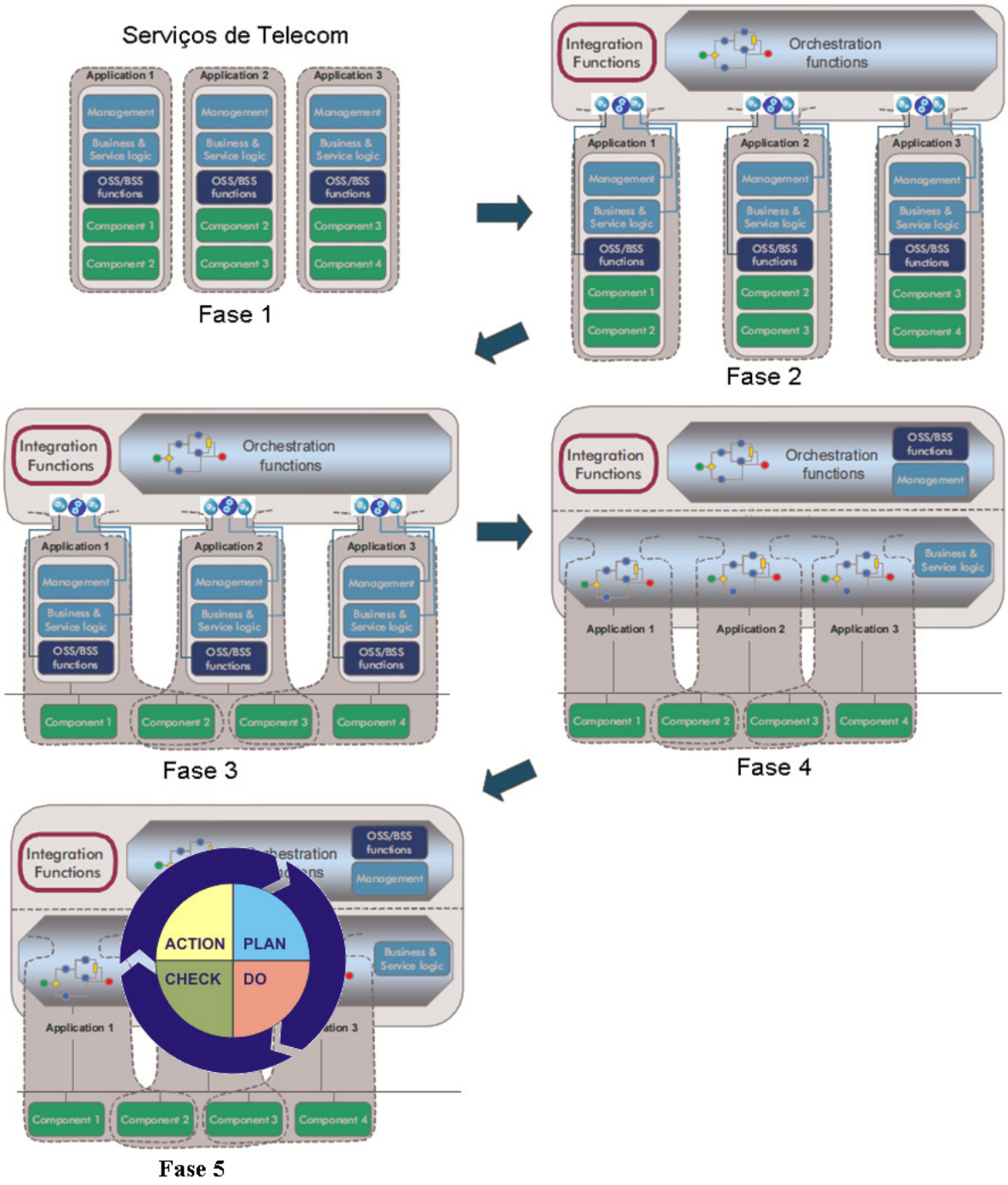

Figura 4.5 – Transição para o ambiente SOA. Fonte: Adaptado de Wambecq et al., 2009.

5. CONCLUSÕES

Este trabalho propôs um guia para avaliação da maturidade no processo de transição de uma arquitetura tradicional dos sistemas de telecomunicações para uma Arquitetura Orientada a Serviços, com foco nos Sistemas de Suporte a Operação tendo em vista a realidade atual das empresas e alto investimento já realizado ao longo dos anos nos sistemas já desenvolvidos. O processo procurou levar em conta estes fatores no sentido de se aproveitar parte destes sistemas através da identificação e uso de funções candidatas à composição de serviços.

Cada vez mais o que se constata nas empresas de telecomunicações é que os OSS não estão mais conseguindo atender a grande demanda por novos serviços. Vários fatores motivadores, dentre eles a convergência dos serviços e redes para a Internet (protocolo IP), a Web 2.0, o amplo acesso a dispositivos com acesso a internet e a demanda cada vez maior dos usuários induz as operadoras ao desenvolvimento de serviços cada vez mais complexos. A arquitetura dos OSS atualmente utilizada pelas empresas de Telecom não consegue mais fornecer o *time-to-market* e eficiência operacional necessários no desenvolvimento e entrega destes serviços. Para garantir competitividade no mercado, há a necessidade de uma evolução para uma Arquitetura Orientada a Serviços que possibilite que a empresa de Telecom possa acompanhar a demanda cada vez mais crescente pela criação de serviços avançados de comunicação e entretenimento através da disponibilização de serviços que podem ser intercambiáveis e combinados.

A avaliação da maturidade é muito útil para que uma determinada área possa localizar onde está e como está para em seguida realizar um plano para que ela possa chegar a algum ponto melhor do que o atual, na busca da excelência. Ter um guia como referência para avaliação da maturidade na migração dos OSS para SOA pode ainda ajudar a minimizar os riscos de impactos nos sistemas, na organização como um todo, e consequentemente os clientes irão experimentar uma transição mais transparente com menos interrupções nos serviços.

A introdução de SOA pode prover a focalização na tecnologia e na organização do negócio, de forma a atingir os objetivos comuns da organização. A utilização de um modelo aplicado ao SOA fornece objetivos e guias de orientação acerca de como SOA pode ter um impacto mais positivo na organização.

Outra estratégia que as empresas de Telecom vêm adotando é aumentar o ARPU[59] através da disponibilização de novos serviços que agregarão maior valor aos já existentes. A utilização do SOA tem se mostrado como uma prática comprovadamente eficiente para este aumento. Segundo Nolle (2008), para uma operadora de Telecom, a melhor maneira de criar serviços de valor agregado, é preferível que incluam recursos que explorem os conhecimentos obtidos a partir de outros serviços oferecidos. Os custos administrativos e de aquisição do cliente são pagos em uma base por cliente, e se o cliente tiver sido obtido em qualquer lugar (ou tornou-se um cliente através de qualquer serviço ou canal disponível para o operador de rede) é fundamental para alavancar a relação completa. Isto envolve muito mais do que simplesmente vender vários serviços agrupados ou "combos", isto envolve a criação de serviços de simbiose cruzada, ou seja, serviços que se complementam para a criação de outros serviços. A simbiose tanto pode reduzir os custos operacionais e aumentar a receita média por usuário (ARPU), criando uma situação de dupla vitória.

A Arquitetura Orientada a Serviços (SOA) facilita a criação destes serviços, tornando o estado e os elementos de um serviço de fácil acesso para o outro. Um serviço de voz baseado em SOA pode facilmente obter o estado da parte chamada de TV do sistema de IPTV, por exemplo. IP Multimedia Subsystem (IMS) fornece um meio de controlar ambas as chamadas fixas e móveis usando o Wi-Fi em casa ou gateways *femtocell*[60]. Ele também oferece um método seguro de controlar como os usuários acessam essas aplicações, mesmo longe de casa.

Segundo um estudo realizado pela IBM (2006), os benefícios para as empresas, como elas próprios reconhecem, são substanciais. Com base numa análise de 35 implementações de SOA reais em 11 indústrias de todo o mundo, obteve-se uma imagem muito clara dos tipos de empresas estão obtendo benefícios de SOA (Figura 5.1).

Pode-se inferir que a utilização de um modelo que permita avaliar a maturidade na adoção de SOA em OSS trará benefícios, norteando as ações em busca de metas claramente definidas pelo modelo de referência proposto.

[59] **ARPU** - Sigla que vem do inglês "*Average Revenue Per User*". Em português significa receita média por usuário e é um termo muito utilizado para medir o desempenho de operadoras de telecomunicações.
[60] **Femtocell** em telecomunicações é uma pequena estação base celular, tipicamente projetada para uso em casa ou pequenas empresas.

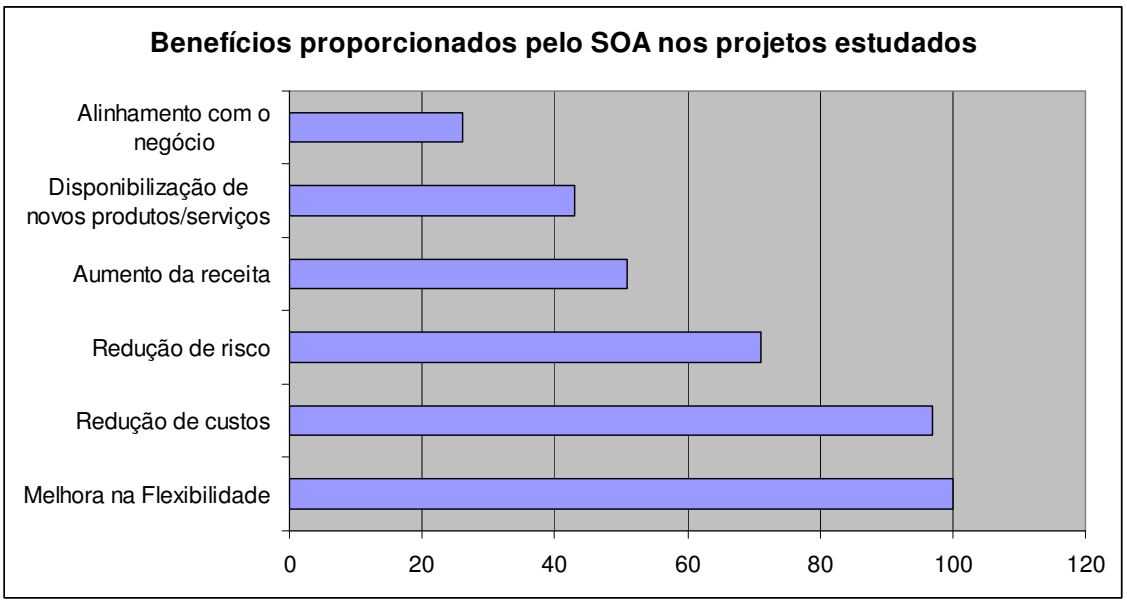

Figura 5.1 – Benefícios proporcionados pelo SOA nos projetos estudados. Fonte: IBM Global *Business Services Analysis of 35 SOA implementations,* 2006.

5.1. TRABALHOS FUTUROS

A aplicação em uma empresa de telecomunicações da abordagem proposta neste trabalho poderá contribuir para o aperfeiçoamento das técnicas sugeridas e para a aplicação em outras empresas.

Além disso, conforme já visto, as empresas de Telecom visam a convergência para a rede IMS (*IP Multimedia Subsystem*). SOA e IMS tem um conjunto de missões altamente complementares, e muitas operadoras já percebem que ambos são essenciais no planejamento de seus serviços. A questão principal pode ser a prioridade de implantação dos serviços e de metas iniciais, e estes se relacionam com o modelo de negócios que impulsiona a implantação dos serviços. Sugere-se que sejam realizados estudos no sentido do desenvolvimento dos OSS para o ambiente SOA já pensando nesta convergência para o IMS, pois há muito pouco trabalho prático sobre este tema.

Outra sugestão de trabalho é sobre o fato de apesar dos OSS terem uma importância indiscutível em uma empresa de telecomunicações, outras áreas importantes merecem o foco das atenções quando o assunto é SOA. Os sistemas de suporte ao negócio (BSS – *Business Support Systems*), por exemplo, complementariam a arquitetura orientada a serviços uma vez que o conceito de SOA pode e deve ser abrangido por toda a organização.

Percebe-se também que os OSS estão extrapolando o mundo das telecomunicações e indo para outras áreas como a de empresas produtoras de energias renováveis, por exemplo. Portanto, mesmo com algumas adaptações, há uma possibilidade de grande abrangência para utilização deste trabalho, e estudos neste sentido certamente irão contribuir.

REFERÊNCIAS BIBLIOGRÁFICAS

Bieman, J. M., & Kang, B.-K.. *Cohesion and reuse in an object-oriented system. In SSR'95: Proceedings of the 1995 Symposium on Software reusability* (pp. 259–262). New York, NY, USA: ACM Press, 1995.

Buchhirano, A., & Gnesi, S. *A survey on services composition languages and models. International workshop on web service modeling and testing (WS-MaTe)* 2006.

Chidamber, S. R., & Kemerer, C. F. *A metrics suite for object oriented design. IEEE Transactions on Software Engineering,* 20(6), 476–493, 1994.

Cunha, Mônica X. C., Souza Júnior, Marcilio e Dornelas, Jairo Simião – In *O uso da arquitetura SOA como estratégia de integração de sistemas de informação em uma instituição pública de ensino, 2009.*

Duarte, Emerson de Jesus. *Telecomunicações: a evolução tecnológica e a empregabilidade no setor, 2009.*

Erickson, Rich. *AT&T SOA Experience*. Disponível em www.infoworld.com/event/soa/resources/11_07_905_Erickson_ATT%20final.ppt. Acessado em 10/09/2007.

Erl, Thomas . In *SOA Principles – An introduction to the Service-Orientation Paradigm*. Disponível em http://www.soaprinciples.com/. Acessado em 13/09/2007.
Erl, Thomas, Loesgen, Brian. *SOA with .NET & Windows Azure: Realizing Service-Orientation with the Microsoft Platform*, 2010.

Erl*, Thomas. In SOA Methodology – Mainstream processes for Service-Oriented Analysys & Design.* Disponível em www.soamethodology.com/. Acessado em 08/09/2006.

Estefan, Jeff A., Laskey, Ken, McCabe, Francis G., Thjornton, Danny. *Reference Architecture Foundation for Service Oriented Architecture*. OASIS, 2009.
FARO, Luiz Felipe de Almeida Tovar. *Técnicas para garantia de qualidade fim-a-fim em serviços de telecomunicações digitais*, 2007.

Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

Fonseca, João Carlos Fonseca, Convergência – In *Prestadoras de telecomunicações pressionadas para novas estruturas de redes e novos serviços, 2009*

Frontini, Maria Alice Braga. *Convergência digital e a telefonia móvel – implicações à gestão estratégica e à inovação*, 2008. FROST & SULLIVAN. Convergência Digital (http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=22990&sid=17), acesso em 23.6.2010.

Harrison, Michael I., Shiron, Arie. *Organizational Diagnosis and Assessment*. Ed. Sage Publications, Inc. Thousand Oaks, California, 1999.

Jennings, Frank, Matjaz Juric, Poornachandra Sarang, Ramesh Loganathan. *SOA Approach to Integration,* 2008.

Juric, Matjaz B.. *BPEL: Service composition for SOA*. http://www.JavaWorld.com, acessado em 07/10/2006.

Light Reading, *"Light Reading´s Services Software Insider* I", v. 2, n. 4, August 2006.

Meier, J. D., Vasireddy, S., Babbar, A., & Mackman, A.. *Improving .NET application performance and scalability (patterns & practices)*. Microsoft Corporation, 2004.

Melgarejo, Luiz Fernando Bier. *Arquitetura REST: um estudo de sua implementação* MISRA, K. . *OSS for telecom networks: an introduction to network management*, 2004.

Oliveira, Andrey Guedes. *Indicadores de Desempenho e Dimensionamento de Recursos humanos nos Centros de Operações de Redes*, PUC 2008.

OMG. Conjunto de elementos principais do BPMN, 2005. http://www.omg.org
Ott, L. M., & Thuss, J. J. *Slice based metrics for estimating cohesion.* No IEEE-CS International Metrics Symposium (pp. 71–81), 1993.

Reddy, V. Kumar, Alpana Dubey, Sala Lakshmanan, Srihari Sukumaran e Rajendra Sisodia. *Evaluating legacy assets in the context of migration to SOA,* 2008.

Sampaio, Cleuton. *Soa e Web Services em Java.* - Rio de Janeiro: Brasport, 2006.

Schmelzer,R. *Should services be stateful?,* 2007.
http://searchwebservices.techtarget.com.

Schmelzer, R. *What belongs in a service contract?* 2007. http://www.zapthink.com/.

SIQUEIRA, Jairo. *O Modelo de Maturidade de Processos: como maximizar o retorno dos investimentos em melhoria da qualidade e produtividade*, 2004.

Stevens, W. P., Myers, G. J., & Constantine, L. L. *Structured design. IBM Systems Journal*, 38(2/3),231–256, 1999.

SUN microsystems. *NFS: Network file system version 3 protocol specification*. Technical report, 1994.

Tanenbaum, Andrew S.. *Redes de computadores.* Rio de Janeiro: Campus 1997.

Terry, Cho. *SDP(Service Delivery Platform) Intro-from SOA perspective,* 2009.

van der Hoek, A., Dincel, E., & Medvidovic, N. . *Using service utilization metrics to assess the structure of product line architectures*. In IEEE METRICS (pp. 298–308), 2003.

Vuono, Evandro. *Service Delivery Platform*. *A sua operadora ainda vai ter um... e logo*. http://www.teleco.com.br/hp/hp_artigos006.asp - acessado em 02/03/2010.

Wambecq, Albert; Thierry Pollet, Gerard Maas, Johan Marien. *Telecom Services Delivery in a SOA*, 2006.

White, Stephen. *Introduction to BPMN* , 2004.