UNIVERSIDADE CATÓLICA DE BRASÍLIA

PRÓ-REITORIA DE GRADUAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

Bacharelado em Ciência da Computação

SIGED – Sistema de Gestão Educacional

Alunos: Brenda Regina Silva Bezerra
Felipe Carolino Machado
Renato Alves de Lima

Orientador: MSc. Milton Pombo da Paz

BRASÍLIA 2008

BRENDA BEZERRA
FELIPE CAROLINO
RENATO ALVES

SISTEMA DE GESTÃO EDUCACIONAL - SIGED

Monografia apresentada para obtenção do título de bacharel no Curso de Ciência da Computação pela Universidade Católica de Brasília, sob a orientação do professor Milton Pombo.

Brasília – DF
2008

Dedicamos a Deus, aos nossos mestres que nos apoiaram e passaram um vasto conhecimento durante esses anos, aos nossos familiares que sempre nos apoiaram e compreenderam nos momentos em que necessitamos e aos nossos amigos.

AGRADECIMENTOS

Primeiramente a Deus, pois sem ele nada seria possível.

Aos nossos pais (Odontina, Graça, Nivanda, Valdisson, Geni, Reinaldo) que fizeram esse sonho se tornar realidade.

Aos nossos professores que se dedicaram muito para nos transformarem em profissionais respeitáveis.

Aos nossos irmãos (Gustavo, Henrique, Elisa, Márcia) pelo carinho que sempre tiveram conosco.

Aos nossos familiares pelo incentivo.

As nossas paixões (Pollyanna, Daniela) que estiveram conosco em cada passo nos dando amor e incentivado na busca de nossos ideais.

Aos nossos amigos, que encheram essa etapa com muita alegria e descobertas.
Em especial a nossa amiga Brenda porque sem ela nada disso seria possível.

"A vida é mais simples do que a gente pensa; basta aceitar o impossível, dispensar o indispensável e suportar o intolerável."

(Kathleen Norris)

RESUMO

A Arquitetura Orientada a Serviços (SOA) permite que novas aplicações sejam criadas através da combinação dos serviços, integrando aplicativos de forma mais rápida, reduzindo custos e tempo de desenvolvimento, além de oferecer mais flexibilidade à área de Tecnologia de Informação (TI), que consegue responder com agilidade as necessidades da área de negócios da empresa.

Este projeto de pesquisa sobre SOA aplicado em um ambiente de gestão acadêmica de uma instituição de ensino fundamental.

A validação da pesquisa será por meio de um protótipo desenvolvido em computador que trate a gestão dos processos básicos em um ambiente educacional, com o objetivo de automatizar os processos necessários para a gestão escolar e financeira de uma instituição de ensino. A utilização da SOA visa garantir a reusabilidade e flexibilidade para acompanhar e apoiar o ambiente organizacional.

Esta solução esta baseada em um ambiente construído sobre a arquitetura SOA - oferecerá a Instituição possibilidade de construir e integrar sistemas.

A solução aqui apresentada fará parte de uma automatização completa da Escola, foco deste trabalho de pesquisa, na qual é composta por dois outros projetos: Sistema de Gestão Educacional – Redes e Sistema de Gestão Educacional – Data Warehouse.

Este estudo propõe analisar esta nova vertente de Engenharia de Software e o impacto que o desenvolvimento e popularização deste paradigma trarão ao desenvolvimento de software.

Espera-se que, ao final do texto, o leitor seja capaz de compreender exatamente o que é a arquitetura e o que ela propõe, assim como as suas vantagens e possíveis desvantagens.

ABSTRACT

The Service Oriented Architecture (SOA) allows that new applications be created through combination of services, which integrates applications faster than usual and also reduces its development costs and time. It also provides more flexibility to the Information Technology (IT) area enabling it to comply with the enterprise business divisions more quickly.

This research project about SOA has been applied to an academic management environment of an educational fundamental school.

The validation of the research project will be done by the development of a computer prototype that treats the management of the basic processes in an educational environment, which main objective is to authomatize all processes needed in order to carry out the school and financial management of the educational institution involved in the project. The utilization of SOA aims at guaranteeing reusability and flexibility, which will allow the accompaniment and support to the organizational environment.

This solution is based on an environment built under the SOA architecture – it will offer the institution the possibility to construct and integrate systems.

The solution presented here will be part of a complete authomatization of the school, which is the focus of this research work, and it is composed by two other projects: Educational Management System – Educational Management System Network – Data Warehouse.

This study aims at analyzing this new Software Engineering trend and the impact that the development and popularization of this paradigm will have on the software development.

It is expected that, after reading the text, the reader is able to understand exactly what the architecture is and what it proposes, as well as its advantages and possible disadvantages.

SUMÁRIO

1 INTRODUÇÃO 1

1.1 Motivação 1

1.2 Breve Histórico 2

1.3 Problemas Diagnosticados 3

1.3.1 Problemas Específicos 3

1.4 Surgimento da Necessidade da Pesquisa 3

1.5 Usuários Beneficiados da Pesquisa 4

2 OBJETIVOS DA PESQUISA 5

2.1 Objetivo Geral 5

2.2 Objetivos Específicos 5

3 PROPOSTA DA PESQUISA 5

3.1 Descrição da Pesquisa 5

3.2 Resultados Esperados 6

3.3 Restrições da Pesquisa Proposta 6

3.4 Relação Custo x Benefício 6

3.5 Áreas afetadas pela Pesquisa 7

4 METODOLOGIA DA PESQUISA 8

4.1 Recursos Utilizados 8

5 ARQUITETURA ORIENTADA A SERVIÇOS (SOA) 9

5.1 Definição 9

5.2 História de SOA 11

5.3 SOA é um paradigma 12

5.4 SOA x Sistemas Distribuídos 12

5.5 Diferentes Proprietários **13**

5.6 Heterogeneidade **14**

5.7 Conceitos de SOA **14**

5.8 Infra – estrutura **15**

6 ARQUITETURA SOA **15**

6.1 Benefícios de uma arquitetura SOA **15**
6.1.1 SOA no Contexto 17

6.2 As Camadas de uma Arquitetura SOA **18**
6.2.1 Modelos de Arquitetura Lógica 19
6.2.2 Modelos de Arquitetura Mistos [Josuttis,2008] 20
6.2.3 Escolhendo o Modelo de Arquitetura Correto [Josuttis,2008] 21

6.3 Papéis e Responsabilidades **22**

6.4 Validação Multicamadas **22**
6.4.1 Panoramas com Múltiplos Canais de *Front-ends* 24
6.4.2 *Front-ends* de Múltiplas Aplicações 24

6.5 Gerenciamento Futuro **26**

6.6 Processos **26**

6.7 Governança **27**

6.8 SOA X Sistemas Homogêneos **27**

6.9 SOA X Tecnologia Especifica **28**
6.9.1 Terminologia SOA 28

7 SERVIÇOS **28**

7.1 Serviços Representam Funcionalidade de Negócio **29**
7.1.1 *Interfaces* e Contratos 30

7.2 Classificação de Serviços **32**
7.2.1 Serviços Básicos [Josuttis,2008] 33
7.2.2 Serviços Básicos de Dados 33
7.2.3 Serviços Básicos de Lógica 35

7.3 SOA Fundamental **35**
7.3.1 Serviços Compostos 37
7.3.2 Serviços Compostos para Múltiplos *Back-ends* 37
7.3.3 Serviços Compostos para um *Back-end* 38

7.4 SOA Federativa **38**
7.4.1 Serviços de Processos 39

7.5 SOA Habilitada para Processos **40**

7.5.1 Estado de Serviço Versus Estado de *Back-end* ... 40
7.5.2 Lidar com Tipos Diferentes de Consumidores ... 41
7.5.3 Serviços de Leitura Versus Serviços de Escrita ... 41
7.5.4 Categorização de acordo com Erl ... 41
7.5.5 Serviços Técnicos e de Infra-estrutura ... 44

7.6 Gerenciamento de Serviços ... 45
7.6.1 Repositórios e Registros ... 45
7.6.2 Estabelecendo Repositórios e Registros ... 46

7.7 Ciclo de vida dos Serviços ... 47
7.7.1 Serviços Sob Desenvolvimento ... 47
7.7.2 Desenvolvimento Iterativo de Serviços ... 48
7.7.3 Identificação dos Serviços ... 49
7.7.4 Serviços em Produção ... 50
7.7.5 Modificação Serviços em Produção ... 50
7.7.6 Retirando um Serviço ... 52

7.8 SOA e Desempenho ... 53
7.8.1 Onde Desempenho Importa ... 53
7.8.2 Desempenho e Reusabilidade ... 55
7.8.3 Restrições de Chamada ... 55
7.8.4 Serviços Customizados ... 55
7.8.5 Reusabilidade na Prática ... 56
7.8.6 Desempenho e Compatibilidade com Versões Anteriores ... 56

7.9 SOA e a Tecnologia Web Services ... 58

7.10 Gerenciamento de Processos de Negócios ... 60

7.11 Terminologia de BPM ... 61

7.12 BPM E SOA ... 62

7.13 Ferramentas de Modelagem de Processos de Negócio ... 63

7.14 SOA E ORGANIZAÇÃO ... 63
7.14.1 Gestão ... 63

7.15 ROI (Return on Investment) ... 64
7.15.1 Determinar o ROI do SOA ... 65

7.16 Segurança ... 66
7.16.1 Requisitos de Segurança ... 66
7.16.2 Segurança com XML e *WEB SERVICES* ... 67
7.16.3 Alguns Conceitos de Segurança em SOA ... 67

7.17 Desafios SOA ... 68

8 ESTUDO DE CASO ... 69

8.1 Proposta de Solução ... 69

8.2 A Utilização do SIGED no TANGRAN ... 71

8.3 Requisitos mínimos que o sistema deve atender 72

8.4 Modelando a Solução 73
8.4.1 Diagrama de Classes de Negócio 73
8.4.2 Especificações de Caso de Uso 75

8.5 Diagramas de Caso de Uso 86
8.5.1 Diagrama de Caso de Uso – Efetuar Login 86
8.5.2 Diagrama de Caso de Uso – Cadastro 87
8.5.3 Diagrama de Caso de Uso – Cadastrar Responsável 87
8.5.4 Diagrama de Caso de Uso – Matricular Aluno 88
8.5.5 Diagrama de Caso de Uso – Cadastrar Disciplina 88

8.6 Documentos Gerados 89
8.6.1 Documentos administrativos 89
8.6.2 Documentos técnicos 89
8.6.3 Levantamento das necessidades 89
8.6.4 Resistência do usuário 90

9 TELAS SIGED 91

9.1 – Login Sistema 91

9.2 - Usuário inválido 91

9.3 – Tela Principal do Sistema – Menus do Sistema 92

9.4 - Menu Acadêmico 92

9.5 - Menu Administrativo 93

9.6 - Tela de Listagem de Alunos 93

9.7 - Tela de Inclusão de Alunos 94

9.8 - Tela de Listagem de Responsáveis 94

9.9 - Tela de Inclusão de Responsáveis 95

9.10 - Tela de listagem de Categorias de séries 95

9.11 - Tela de Inclusão de Categorias da série 96

9.12 - Tela de Listagem de Disciplinas 96

9.13 - Tela de Inclusão de Disciplinas 97

9.14 - Tela de Listagem de Turmas 97

9.15 - Tela de Inclusão de Turmas 98

9.16 - Tela de Listagem de Alunos 98

9.17 - Tela de Matrícula de Alunos 99

9.18 - Tela de Inclusão de Usuários do Sistema 99

9.19 - Tela de Listagem de Usuários do Sistema 100

CONCLUSÃO 101

LISTA DE FIGURAS

Figura 1– Integração – palavra chave 16

Figura 2– Ciclo de Vida SOA..... 17

Figura 3– Camada de serviços posicionada entre as camadas de processos de negócio e de aplicação [ERL, 2006]. 19

Figura 4– Camada Lógica 20

Figura 5– Camadas Básicas de serviços e estágios de expansão de SOA [Josuttis,2008] 33

Figura 6– As características dos tipos de serviços de acordo com [Allen,2006] ... 43

Figura 7- Fornecedor, consumidor e *broker* [Josuttis2008] 45

Figura 8– Um ciclo de vida típico de *software* se aplica á serviços. [Josuttis,2008] 48

Figura 9– O ciclo de vida do *software* com iterações [Josuttis,2008] 48

Figura 10– Começando o ciclo de vida com identificação do serviço [Josuttis,2008] 49

Figura 11– Ciclo de vida do serviço com correções de *bugs* e modificações [Josuttis,2008] 51

Figura 12– Ciclo de vida do serviço completo [Josuttis,2008]..... 52

Figura 13- Paradigma "procura-consolida-executa" [adaptado de ARSANJANI, 2004] 59

Figura 14- Arquitetura 71

Figura 15 - Diagrama de Classes – Visão sistêmica do negócio..... 73

Figura 16– Diagrama de Processo de Negócio..... 74

Figura 17– Serviços (Pacotes) 74

Figura 18 – Serviços (Matricular Aluno) 75

Figura 19 - Diagrama de Caso de Uso – Efetuar Login..... 86

Figura 20 - Diagrama de Caso de Uso – Cadastro 87

Figura 21 - Diagrama de Caso de Uso – Cadastrar Responsável..... 87

Figura 22 - Diagrama de Caso de Uso – Matricular Aluno 88

Figura 23 - Diagrama de Caso de Uso – Cadastrar Disciplina..... 88

LISTA DE TABELAS

Tabela 1 -Caso de Uso 01 – Efetuar Login 75

Tabela 2- Caso de Uso 02 – Matricular Aluno..... 76

Tabela 3 - Caso de Uso 03 – Cadastrar Responsável 77

Tabela 4 - Caso de Uso 04 – Cadastrar Funcionário 78

Tabela 5 - Caso de Uso 05 – Cadastrar Turma..... 80

Tabela 6 - Caso de Uso 06 – Cadastrar Disciplina..... 81

Tabela 7 - Caso de Uso 07 – Emitir Boletim 82

Tabela 8 - Caso de Uso 08 – Registrar Ata de Reunião 83

Tabela 9 - Caso de Uso 09 – Registrar Ocorrência..... 84

Tabela 10 - Caso de Uso 10 – Controlar Inventário 85

LISTA DE SIGLAS

SOA **-** *Service-Oriented Architecture*

SOA – Arquitetura Orientada a Serviços

ESB – *Enterprise Service Bus*

BPM – *Business Process Modeling*

MDSD – *Model Driven Software Development*

WFMS - *Workflow Management System*

BPMS - *Business Process Management System*

B2B - *Business-to-business*

BPEL - *Business Process Execution Language*

ESB - *Barramento de Serviços Empresariais*

SOAP - *Simple Object Access Protocol*

SLA - *Service Level Agreement*

XML - *eXtensible Markup Language*

WSDL- *Web Services Description Language*

UDDI - *Universal Description, Discovery, and Integration*

1 INTRODUÇÃO

Este projeto de pesquisa tem como objetivo descrever a Arquitetura Orientada a Serviços SOA. Existem dois conceitos fundamentais a serem compreendidos: Arquitetura Orientada (metodologia para desenvolvimento de *Software*) a Serviços (panorama de todos os ativos de *software* de uma empresa é a representação de todas as peças, que juntas formam uma construção).

Ao se definir uma estratégia para conquistar clientes e conseqüentemente aumentar o lucro e o capital das companhias, faz com que o uso de tecnologias da informação voltadas para a tomada de decisão seja cada vez mais empregado dentro das empresas de forma a auxiliar no alcance dos objetivos contemplados por sua estratégia. Essas tecnologias são vitais porque a partir da visão dos dados contidos nos repositórios transacionais, será possível a extração de informações importantes e históricas para análise estratégica do negócio em um ambiente analítico próprio.

1.1 Motivação

A palavra chave seria flexibilidade, sendo que o seu maior desafio é a manutenibilidade, para que com isso possa se houver uma nova abordagem, que aceite heterogeneidade e que conduza á descentralização, esse seria o grande foco da arquitetura SOA.

Com isso, é possível identificar vários assuntos sobre os quais se deseja tomar uma decisão, a partir dos questionamentos gerenciais dos usuários interessados na implementação deste sistema.

Trazendo este cenário para o mundo acadêmico, percebe-se a necessidade de uma análise estratégica dentro do ambiente escolar para o alcance de objetivos

de igual importância quando comparados ao ramo da Indústria, Telecomunicações, Logística, entre outras. Com base nesta necessidade, esta pesquisa tem como intuito responder a questões estratégicas referentes a um ambiente escolar específico.

1.2 Breve Histórico

Com o crescente uso da Internet, pode-se evidenciar uma necessidade crescente de comunicação e interação entre as partes envolvidas.

A utilização dos sistemas de computação tornou-se simples. Agora, houve a necessidade de tornar estes sistemas mais autônomos para facilitar o uso destes no dia-a-dia.

Basta ver que havendo a análise do que é a Rede (*Web*) atualmente, conclui-se que ela é uma fonte imensa de informação.

A orientação a serviços surgiu para suprir tal necessidade, unindo diversos conceitos e práticas sob um único paradigma. Tal paradigma é conhecido como Arquitetura Orientada a Serviços ou SOA (do acrônimo em inglês) e, em essencial, trata-se de um padrão criado com o objetivo de desenvolver sistemas modularizados, o que traz diversos benefícios ao produto final. Ao invés de serem desenvolvidas grandes aplicações como um único bloco, podem-se encapsular algumas funções importantes e disponibilizá-las na forma de serviços em uma rede.

Atualmente o foco da TI está em promover o alinhamento estratégico entre os objetivos e estratégias do negócio, baseado na adoção de modelos e práticas sugeridas no mercado, e assim conseguir implementar planos, padrões e procedimentos além de garantir à adequação estratégica e integração funcional entre os ambientes externo (mercado, política,fornecedores, etc.) e interno (infra-estrutura, estrutura administrativa, recursos financeiros,tecnológicos e humanos).

1.3 Problemas Diagnosticados

Todo o processo era realizado de forma manual o que gerava uma baixa produtividade e falta de informações confiáveis para as tomadas de decisões. Ausência de integração entre os departamentos da escola, falha de comunicação entre a escola e os pais e responsáveis.

1.3.1 Problemas Específicos

Há uma grande dificuldade no ambiente escolar de pequeno porte em controlar seus processos, o que acaba gerando desorganização, devido os seus processos serem realizados manualmente.

1.4 Surgimento da Necessidade da Pesquisa

Cada vez mais as organizações estão tornando-se dependentes da tecnologia para satisfazer as necessidades do seu negócio, isso resultou no aumento de atividades realizadas pelas empresas que tendem a procurar soluções correspondentes aos objetivos da organização, e que preencham as exigências e as expectativas do cliente.

A crescente competitividade do ambiente de negócios está desafiando os administradores de hoje. A emergência e o fortalecimento da economia global, a transformação da sociedade industrial na sociedade baseada na informação e no conhecimento.

Para atender às necessidades deste novo ambiente, a informação precisa ter como suporte uma adequada TI, a fim de disponibilizar as respostas rápidas e eficientes que a competitividade está constantemente exigindo (FREITAS, 1997).

Com a informação no centro de tudo, é fundamental saber usá-la de forma estratégica, uma vez que seus benefícios podem ser inúmeros, bem como permite a descentralização do poder na tomada de decisão, uma vez que proporciona a democratização da informação, elimina o excesso de trabalho pela desburocratização e diminuição do volume de papéis, assim como permite uma maior velocidade na tomada de decisões. Atualmente a tecnologia de informação

tem participado de toda atividade de negócios de uma empresa que oferece um produto ou serviço, desde a concepção, planejamento e produção até a comercialização, distribuição e suporte. Como resultado, a TI tem se tornado um componente crítico de planejamento estratégico corporativo e de vantagem competitiva.

1.5 Usuários Beneficiados da Pesquisa

Os principais usuários dos sistemas serão os gestores executivo, administrativo, financeiro, educacional e a coordenação pedagógica, além disso, deverá possuir uma capacidade de integração entre a escola e os responsáveis, bem como futuramente com a comunidade.

2 OBJETIVOS DA PESQUISA

2.1 Objetivo Geral

O objetivo é desenvolver uma pesquisa sobre SOA e validar em um estudo de caso para uma escola. E o desenvolvimento de um sistema com uma arquitetura orientada a serviços que possibilite a integração com os processos de negócio de uma instituição educacional, visando à flexibilidade de manutenção para acompanhar as constantes mudanças organizacionais.

2.2 Objetivos Específicos

- Validação da pesquisa por meio do protótipo;
- Automatização de processos;
- Utilização de serviços para o sistema de gestão educacional; e
- Estudo sobre reutilização de componentes.

3 PROPOSTA DA PESQUISA

Desenvolver uma pesquisa bibliográfica sobre SOA em busca de informações importantes de maneira que auxilie na construção de sistemas e de infra-estrutura que oferecerão interação entre à área de Negócio e à área de Tecnologia da Informação.

3.1 Descrição da Pesquisa

A pesquisa desenvolvida com um aparato bibliográfico diversificado reúne informações necessárias à adoção de práticas que visam à melhoria do negócio por meio da arquitetura SOA. A enfatização da pesquisa visa dar uma visão geral de como esse novo paradigma vem crescendo nas grandes organizações e quais os benefícios de usar essa arquitetura.

3.2 Resultados Esperados

Atuando como facilitador na busca e no registro das informações dos processos de tomada de decisão no ambiente acadêmico, espera-se ter como resultado o aumento da eficiência e eficácia dos gestores educacionais, com conseqüente aumento da produtividade e dos resultados financeiros. Igualmente é esperada á diminuição do tempo para alteração ou inclusão de novos serviços.

3.3 Restrições da Pesquisa Proposta

A pesquisa desenvolvida dá uma visão geral de todas as áreas abordadas na Arquitetura Orientada a Serviços - SOA, listando e as descrevendo basicamente apenas para servirem como fonte bibliográfica e como fonte complementar na pesquisa, não entrando em detalhes específicos de como funciona a arquitetura.

3.4 Relação Custo x Benefício

Implementar a Arquitetura Orientada a Serviços – SOA significa adotar um paradigma para as necessidades do negócio. As empresas precisam de uma abordagem de SOA porque ela oferece a possibilidade de utilizar o potencial de TI para acelerar e dinamizar o negócio, eliminando frustrações – isso, a partir da estruturação de soluções mais flexíveis e prazos mais curtos de desenvolvimento. Ela também justifica os investimentos de TI, ao economizar recursos, esforço e tempo em função da reutilização de componentes e da sua flexibilidade. Implantação de uma estratégia traz diversos benefícios para a empresa, entre eles:

- Mais agilidade e *time-to-market* na entrega de aplicações;

- Redução de custos de novos projetos liberando recursos para outras melhorias; Facilidade e flexibilidade de manutenção das aplicações existentes;
- Melhoria da qualidade e maturidade no desenvolvimento de aplicações;
- Otimização dos processos e padronização da arquitetura dos produtos desenvolvidos; Transformação dos negócios que geram oportunidades de receita através de novas formas de interação com o mercado;
- Resultados visíveis.

Mas para isso, a empresa deverá conscientizar sua organização, para que haja integração em todas as áreas. E o custo de implementação dessa arquitetura é muito alto, e para tal necessita que haja um grande investimento.

3.5 Áreas afetadas pela Pesquisa

Todo o serviço dentro da arquitetura SOA tem vínculos com as unidades de negócio.

4 METODOLOGIA DA PESQUISA

Esta pesquisa é classificada como bibliográfica, pois se destina a investigar possibilidades do emprego dos melhores métodos e praticas a serem recomendadas para o uso e adequação da arquitetura orientada a serviços SOA.

Foram utilizadas as seguintes tecnologias:

- Paradigma de programação: Orientação a Objeto
- Linguagem de modelagem: UML 2.0 (Unified Modeling Language)
- Processo de desenvolvimento: RUP (Rational Unified Process)
- Arquitetura de desenvolvimento: MDA (Model Driven Architecture)
- Linguagem de programação: Java, PHP
- Linguagem de modelagem de processos: BPEL

4.1 Recursos Utilizados

- Modelagem e desenvolvimento do software: IBM Rational Software Architect;
- Modelagem de processos: IBM Websphere Business Modeler;
- Integração de modelos: IBM Websphere Integration Developer;
- SGBD: mySQL; e
- Equipamentos: Notebook T42.

5 Arquitetura Orientada a Serviços (SOA)

5.1 Definição

O termo *Service-Oriented Architecture* foi inicialmente proposto por [SCHULTE e NATIS, 1996], então analistas do Gartner Group. Eles definiram SOA como “uma configuração de computação multicamadas que ajudam as organizações compartilhar lógica e dados através de múltiplas aplicações e modelos de uso”.

Imagine estar preparado para atender às demandas das áreas de negócios com mais agilidade e flexibilidade. Coloque como cenário um ambiente baseado em uma arquitetura de software modular, onde todos os aplicativos são acessados por uma única *interface web* e os sistemas utilizam dados uns dos outros e “conversam” indiscriminadamente. Isto é o pano de fundo de uma arquitetura orientada a serviços (SOA). Neste novo modelo arquitetural, não há mais uma *interface*, um banco de dados e um sistema de integração para cada *software*. Na *web*, em uma única tela, os usuários acessam todos os programas de forma transparente. Além de mais flexível e mais amigável, a *interface* única dá muito mais agilidade e transparência ao usuário. Os aplicativos, por sua vez, utilizam qualquer informação, independentemente de onde ela tenha sido gerada. Graças ao mapeamento lógico dos dados, é como se todos estivessem em uma única base, ajudando a reduzir a inconsistência.

A *Organization for the Advancement of Structured Information Standards* [OASIS,2006], define SOA da seguinte forma:

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

Há quase uma década, quando os portais corporativos saíram da teoria e se tornaram realidade na prática, em diversas companhias dos mais variados

setores e portes, o que mais se falava e se questionava era a integração de *interfaces*. Todas as aplicações corporativas se tornariam integradas em uma única *interface*, beneficiando os usuários. Paralelamente se questionava a arquitetura orientada a serviços, ou SOA. A pergunta é: de que maneira esses dois conceitos convergem? A grande promessa SOA é liberar as áreas de negócios das restrições da área de TI.

Na verdade esses dois conceitos se complementam na medida em que dizem respeito á camada de *interface* e a camada de integração. Pode-se dizer que enquanto os portais corporativos mostram os benefícios da *interface* única, personalização e gestão de conteúdo, o conceito de SOA mostra que utilizar serviços de maneira estruturada pode trazer ganhos expressivos para a implantação e integração de aplicações, além de permitir uma aderência cada vez maior aos requisitos de negócio.

A arquitetura SOA possui inúmeras vantagens, uma delas é a possibilidade de olhar as aplicações sob o enfoque de processos de negócio, adequando-as aos diversos contextos corporativos. Ou seja, passamos a entregar conteúdo de negócios de uma forma completamente diferente do paradigma tradicional. No modo antigo, era preciso entrar no sistema A e B para efetuar um determinado procedimento. Se o portal for aliado à tecnologia SOA, podemos efetuar todo o procedimento em uma mesma *interface*, sem se preocupar com o sistema que está sendo acessado.

Essa mudança de paradigma do antigo para o novo vem sendo o maior desafio dos CIO's (*Chief Information Officer*) que é o executivo encarregado de todo o processamento de informação da organização, que têm a chance de mudar os rumos de como sua empresa opera utilizando-se de novos conceitos. Além dos usuários, as equipes de TI também se beneficiam com essa novidade, elas otimizam todo o processo de desenvolvimento das aplicações, a partir do momento em que podem contar com uma "biblioteca" de serviços já desenvolvidos e uma única *interface* gráfica. Desta forma, também é possível observar um evidente ganho de produtividade e redução de custos.

SOA veio para resolver o problema da lacuna negócio/TI, para deixar ambos mais próximos, pois ela é uma abordagem que ajuda os sistemas a permanecerem escaláveis e flexíveis enquanto crescem, para conseguir tal abordagem, está baseada no seguinte tripé:

- **Serviços** – representam funcionalidades de negócio independentes, e podem ser implementados com qualquer tecnologia em qualquer plataforma;
- **Infra – estrutura específica**, chamada de barramento de serviços corporativos (ESB – *Enterprise Service Bus*), que permite combinar esses serviços de maneira fácil e flexível; e
- **Políticas e processos** que lidam com o fato de que os grandes sistemas distribuídos são heterogêneos, estão sob manutenção e possuem diferentes proprietários.

5.2 História de SOA

Alexander Pasik, um antigo analista da Gartner, definiu o termo SOA para uma turma de *middleware* que estava lecionando em 1994. Pasik foi levado a criar o termo SOA porque "cliente/servidor" havia perdido o seu significado clássico. Para evitar a confusão entre o significado antigo e o novo de cliente/servidor, Pasik adotou "*server orientation*" ao encorajar os desenvolvedores a modelarem aplicações corporativas SOA.

Sendo que os analistas da Gartner, Roy W.Schulte e Yefim V.Natis publicaram os primeiros relatórios sobre SOA, em 1996. Veja [Gartner,1996] e [Gartner, 2003] para detalhes. O momento verdadeiro para SOA foi criado pelas *Web Services* que, inicialmente dirigidos pela Microsoft, atingiu um público maior em 2000. Para citar [Gartner, 2003].

Embora os *Web Services* não necessariamente representem SOA, e nem toda SOA seja baseada em *Web Services*, a relação entre as duas direções tecnológicas é importante e elas são mutuamente influenciáveis: o momento dos

Web Services irá trazer SOA para os usuários finais, e as boas práticas da arquitetura SOA irão ajudar a tornar a iniciativa de *Web Services* bem-sucedida.

5.3 SOA é um paradigma

A arquitetura orientada a serviços (SOA) é um paradigma, uma estratégia que requer tempo e esforço, para a realização e a manutenção dos processos corporativos que se encontram em grandes sistemas distribuídos. Baseada em três conceitos técnicos principais: serviços, interoperabilidade por meio de um barramento corporativo de serviços e acoplamento fraco, que foram mencionados anteriormente.

SOA não é uma arquitetura concreta: é algo que conduz uma arquitetura concreta, é uma abordagem, uma maneira de pensar, um sistema de valores que leva a certas decisões concretas quando se projeta uma concreta arquitetura de serviços.

Seu principal objetivo é melhorar a flexibilidade, ou seja, ajudar no negócio.

Pensamos da seguinte forma: para entregar uma solução de qualidade no tempo certo, é necessário ter flexibilidade. Porém, a flexibilidade tem muito a ver com uma clara organização, papéis, processos, e assim por diante. Portanto, SOA tem que lidar com todos esses aspectos.

SOA é um paradigma de arquitetura para lidar com os processos corporativos distribuídos em um grande cenário de sistema heterogêneos existentes e novos que estão sob controle de diferentes proprietários. Para citar [Josuttis,2008].

5.4 SOA x Sistemas Distribuídos

SOA é um conceito para grandes sistemas distribuídos, porque não se pode entender SOA, sem entender as propriedades de grandes sistemas distribuídos, porque envolve mudar a estrutura de um sistema existente, e isso significa que lidará com plataformas antigas e questões de compatibilidade, e o SOA é uma

abordagem para manutenção dos cenários de grandes sistemas. Por isso, será feita uma explanação sobre sistemas distribuídos.

Uma das razões da heterogeneidade é que os grandes sistemas e os seus dados possuem um tempo de vida incrivelmente logo, durante esse tempo de vida, as novas funcionalidades que movem o negócio são desenvolvidas adicionando-se novos sistemas e processos.

Por natureza, os grandes sistemas são complexos, por essa razão encontrar o lugar certo para as modificações e determinar os efeitos pode ser difícil. Conforme [Spanyi,2003] declara.

Não há tal coisa como “conserto rápido...”. As organizações são sistemas de negócios complexos, dentro das quais á mudança em um componente quase certamente terá impacto em outro componente.

Os grandes sistemas distribuídos possuem também uma importante propriedade adicional: proprietários diferentes, e destaca-se por algumas características a imperfeição e uma certa quantidade de redundância, além de gargalos.

Com o SOA os dados são trocados entre sistemas diferentes e cada sistema opera com a sua cópia local (redundante) e com seus próprios métodos e procedimentos. Diferente de objetos distribuídos, esta abordagem desacopla os sistemas e permitem que escalem.

5.5 Diferentes Proprietários

O SOA inclui práticas e processos que são baseados no fato de que redes de sistemas distribuídos não são controladas por um único proprietário. Equipes diferentes, departamentos diferentes ou até empresas diferentes podem gerenciar os sistemas diferentes. Portanto, diferentes plataformas, agendas, prioridades, orçamentos, e assim por diante, devem ser levados em conta. Esse conceito é fundamental para o entendimento de SOA e dos grandes sistemas distribuídos em geral.

5.6 Heterogeneidade

A outra diferença muito importante entre os pequenos e os grandes sistemas é a falta de harmonização. Os grandes sistemas distribuídos com proprietários diferentes são heterogêneos. A abordagem SOA aceita a heterogeneidade, ela lida com os grandes sistemas distribuídos reconhecendo e suportando essa propriedade.

No entanto, a maioria dos grandes sistemas distribuídos é heterogênea e, quanto mais se integra, mais heterogeneidades se obtêm. SOA é uma abordagem para lidar com essa situação. Ela não se preocupa se heterogeneidade é boa; o seu objetivo é somente lidar com ela de maneira apropriada onde ela existir.

5.7 Conceitos de SOA

- **Serviços** - na essência, um serviço é uma representação da TI de alguma funcionalidade de negócio. Sendo o objetivo do SOA estruturar grandes sistemas distribuídos baseados em abstração das regras e das funções de negócio;
- **Interoperabilidade** – é a base da qual se começa implementar as funcionalidades de negócio (serviços) e que é espalhada pelos múltiplos sistemas distribuídos; e
- **Acoplamento fraco** – precisa-se de tolerância ás falhas, então os objetivos principais são: flexibilidade, escabilidade e tolerância ás falhas. Seu conceito é de minimizar as dependências. Além disso, o acoplamento fraco leva á escabilidade.

Todos os grandes sistemas funcionam apenas se os negócios comuns puderem ser feitos de forma mais descentralizada possível. Uma maneira de introduzir o acoplamento fraco é evitar introduzir mais centralização do que o necessário, não se pode deixar salientar que no caso SOA será necessário alguma centralização, para que se tenha uma base comum.

5.8 Infra – estrutura

A infra-estrutura é a parte técnica de SOA que possibilita a alta interoperabilidade. A infra-estrutura de um cenário SOA é chamada de barramento corporativo de serviços (ESB).

O principal recurso de ESB é possibilitar que faça a chamada de serviços entre os sistemas heterogêneos. As suas responsabilidades incluem transformação de dados, roteamento (inteligente), lidar com segurança e confiabilidade, gerenciamento de serviços, monitoramento e *logging*.

6 Arquitetura SOA

A arquitetura é necessária para restringir todas as possibilidades de SOA de tal maneira que leve a um sistema funcional e de fácil manutenção.

6.1 Benefícios de uma arquitetura SOA

Os benefícios para interessados em implementar um SOA para gerenciamento da informação são significantes. Segundo (Selvage,2005), os seguintes benefícios são alcançados:

- Permite a reutilização dos ativos de TI: Este é o primeiro e o mais importante benefício de implementar SOA. Pode-se construir um serviço de negócio como uma agregação de componentes existentes. Para utilizar este novo serviço é exigido somente saber seu nome e *interface*. A implementação de um serviço específico, bem como sua arquitetura, bem como as complexidades do fluxo de dados que compõe este serviço, são transparentes para quem o chama. Isto permite às organizações aproveitar investimentos atuais, construir serviços de um conglomerado de componentes construídos em diferentes plataformas, que rodam em diferentes sistemas operacionais e desenvolvidos em diferentes linguagens

de programação. Sistemas legados podem ser encapsulados e acessados utilizados as *interfaces web services*;

- Diminui o tempo de desenvolvimento e reduz os custos de desenvolvimento e manutenção: como as demandas do negócio crescem e novos requisitos são introduzidos á todo momento, o custo de desenvolver e criar novos serviços através do SOA *framework* e biblioteca de serviços, o custo é bastante reduzido. A curva de aprendizagem de uma equipe de desenvolvimento é também reduzida, pois já podem estar familiarizados com os componentes existentes;
- Diminuição do risco: reusar componentes existentes reduz o risco de introduzir novas funcionalidades dentro de processos de melhoria ou criar novos serviços do negócio. O risco de manutenção e gerenciamento da infra-estrutura dos serviços de suporte também será reduzido; e
- Proteção do investimento: protege o investimento do cliente no gerenciamento da informação que é altamente volátil num mercado onde as fusões e aquisições são freqüentes.

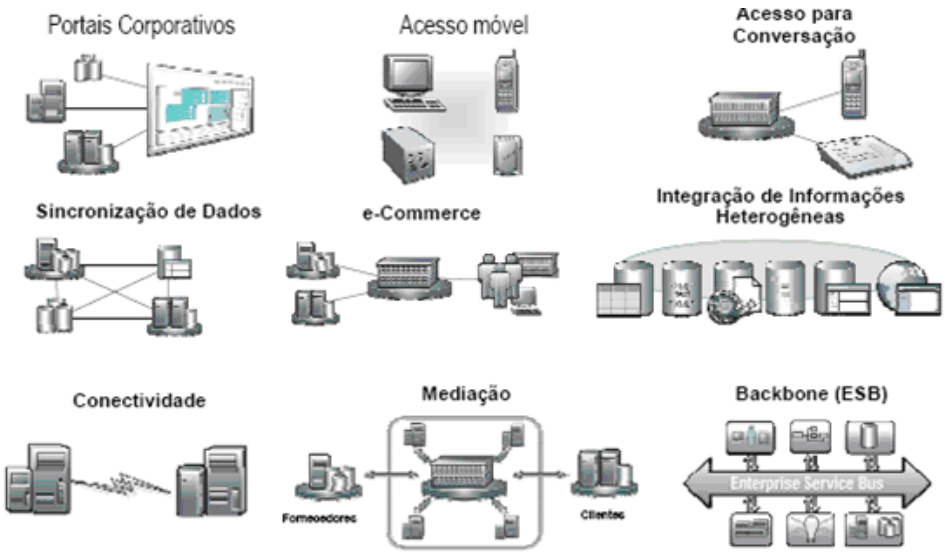


Figura 1– Integração – palavra chave
Fonte: [Mendes,2007]

- SOA é uma arquitetura conceitual de negócios onde as funcionalidades de negócio, ou lógicas de aplicação, é tornada disponível aos seus usuários como um conjunto de serviços reusáveis sobre uma rede de TI; e
- Serviços em SOA são módulos de negócios ou funcionalidades de aplicação com *interfaces* bem definidas e normalmente invocadas com mensagens.

Os seguintes elementos formam uma arquitetura SOA:

- Visão Conceitual SOA;
- Serviços;
- Tecnologia habilitadora;
- Políticas e Governança SOA;
- Métricas; e
- Modelo organizacional e comportamental.

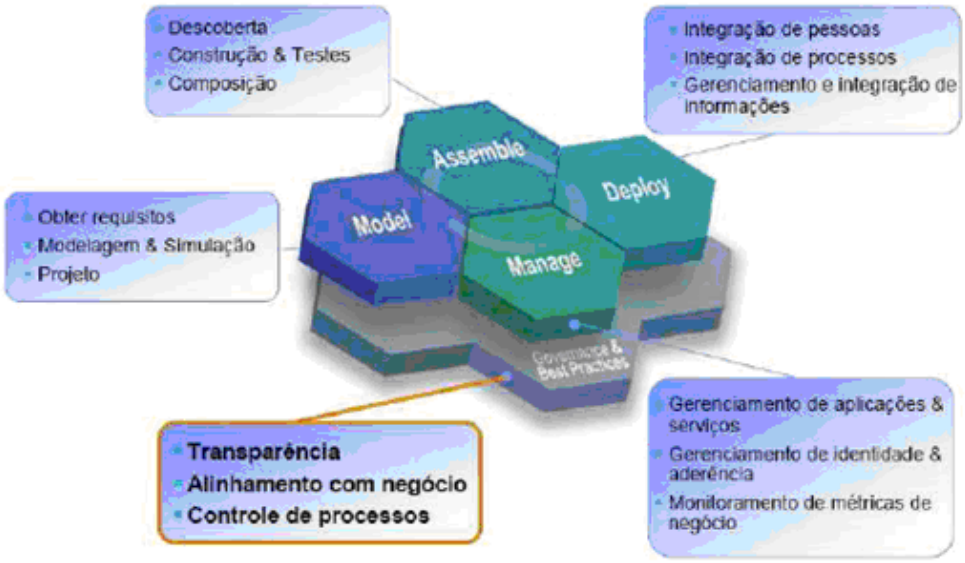


Figura 2– Ciclo de Vida SOA
Fonte: [Mendes, 2007]

6.1.1 SOA no Contexto

Na prática SOA é apenas um aspecto da arquitetura global de um cenário de sistema. SOA tem a ver com como os sistemas se comunicam para executar processos distribuídos. Esses processos distribuídos são iniciados por *front-ends* ou processos em lotes e guardam os seus dados em *back-ends*, e a maneira como SOA organiza a comunicação tem um impacto tanto em *front-ends* como em *back-ends*.

Os diagramas de um ambiente de sistema baseado em SOA podem ser muito diferentes dependendo das intenções do ilustrador e do ponto de vista.

6.2 As Camadas de uma Arquitetura SOA

Apresenta um alto nível de abstração da camada de serviços posicionada entre as camadas dos processos de negócio e de aplicação, que demonstra a conectividade aberta entre as camadas de serviço. No nível físico, serviços são desenvolvidos em ambientes proprietários, onde eles são individualmente responsáveis pelo encapsulamento de uma aplicação lógica específica [ERL,2005].

A Figura 3 mostra como serviços individuais dentro da camada lógica de serviços, representam aplicações lógicas originadas de diferentes plataformas.

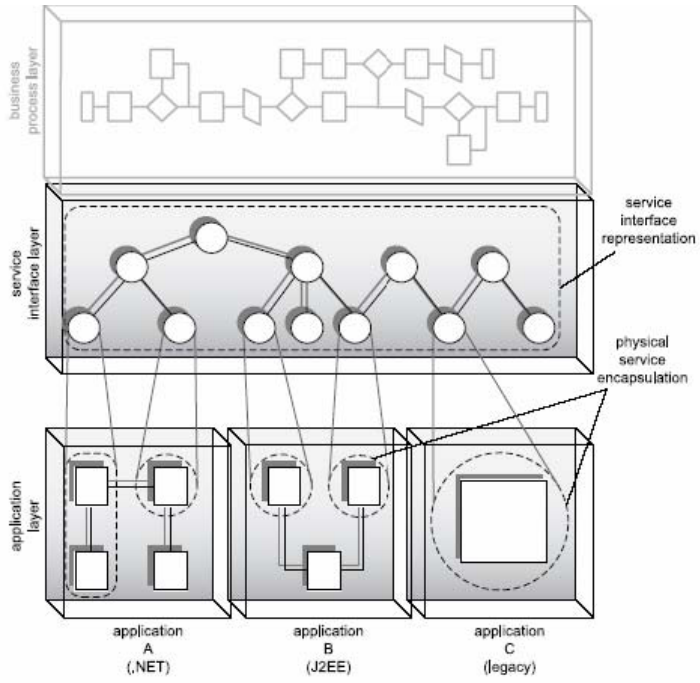


Figura 3– Camada de serviços posicionada entre as camadas de processos de negócio e de aplicação [ERL, 2006].

6.2.1 Modelos de Arquitetura Lógica

Dentro de um cenário poderá haver domínios diferentes, que desempenham papéis diferentes e têm responsabilidades específicas. Um domínio é normalmente algo que consegue identificar dentro da sua estrutura organizacional.

Numa situação ideal, com o alinhamento domínio/TI perfeito, as fronteiras de sistemas correspondem ás fronteiras de negócio. Sendo que de acordo com o modelo de domínio, poderão existir domínios “interdepartamentais” responsáveis pelos processos globais.

Se não tem departamentos interdomínio, precisará de alguma maneira de designar os serviços interdomínio aos domínios específicos, porque alguém tem que ser responsável por tais serviços. Neste caso, designar um serviço a um domínio deve ser parte da solução de uma modelagem. Também poderá haver hierarquias de domínios, tornando as regras que governam as responsabilidades mais complicadas.

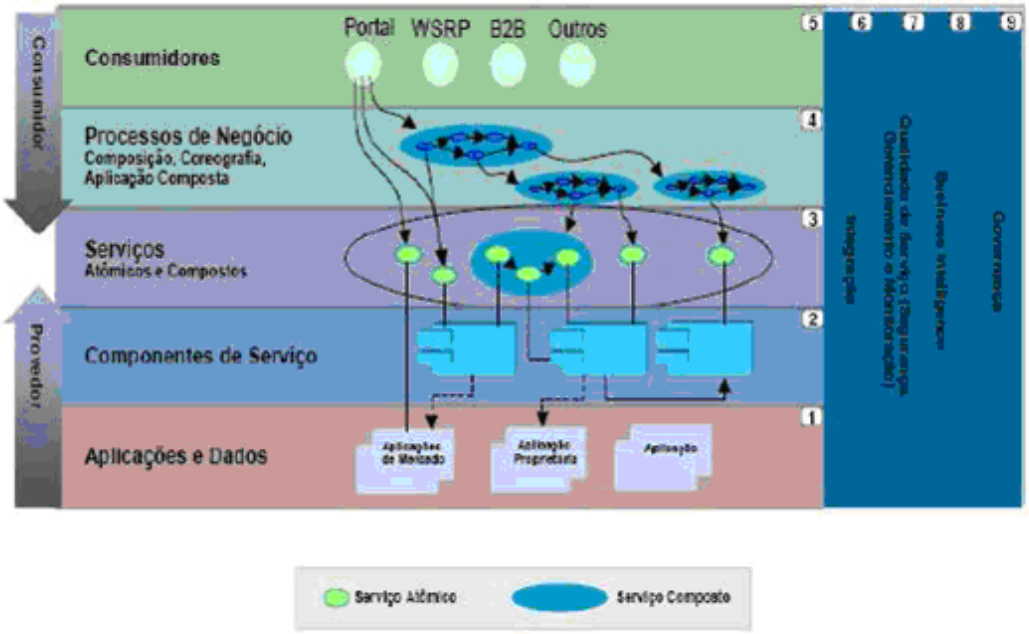


Figura 4– Camada Lógica
Fonte: [Mendes,2007]

6.2.2 Modelos de Arquitetura Mistos [Josuttis,2008]

- Todas as chamadas de serviços são mostradas agora como sendo roteadas através de um ESB (que tem a conseqüência que não pode ver quais serviços compostos e de processos chamam tais serviços básicos);

- Os serviços de processos podem ser implementados (e gerenciados) por um mecanismo ou uma ferramenta BPM; e

- Todos os serviços básicos de lógica que provêem regras de negócio são implementados em um mecanismo de regras.

6.2.3 Escolhendo o Modelo de Arquitetura Correto [Josuttis,2008]

- Por um lado, um domínio deveria ser responsável por toda a sua lógica de negócio. Por esta razão, ele deveria controlar não apenas os serviços básicos de leitura e escrita, mas também os serviços básicos de lógica e os serviços de processos, se isso fizer sentido do ponto de vista de negócio; e
- Existe uma ferramenta e um mecanismo BPM comuns e/ou um mecanismo de regras comum gerenciados por uma equipe que ajuda a obter efeitos sinérgicos para ferramentas e permite que haja uma visão comum para todos os processos e regras de negócio.

Os fabricantes das ferramentas tendem a preferir o modelo técnico, porque isso permite a eles embutir juntas as coisas que você pode comprar. ESB, ferramentas BPM e mecanismos de regras podem até ser considerados um conjunto comum, na qual combinada com o repositório que gerencia todos os serviços, esta abordagem oferece uma maneira homogênea para gerenciar todo o seu negócio. Apenas os detalhes de implementação de serviços básicos estariam fora do escopo central. Usando ferramentas boas facilmente poderia obter uma visão de todos os seus serviços, regras de negócio e processos de negócio tanto em tempo de modelagem como em tempo de execução. Sendo que SOA e grandes sistemas são projetados para permitir heterogeneidade.

Mas deve-se, ter cuidado para não se cair na armadilha de requerer a homogeneidade mesmo quando o preço é alto demais (problemas podem surgir quando tentar chegar a um acordo comum, juntar diferentes requisitos, e assim por diante). Os aspectos técnicos são apenas detalhes de implementação, sendo o ponto de vista de negócio o mais importante dos dois. Deve-se levantar em conta, que existe uma diferença entre igualdade e centralização, sendo que se igualdade levar á centralização, o sistema não escalará mais.

6.3 Papéis e Responsabilidades

Agora se dará foco a *front-ends* que são *interfaces* de usuários. Os serviços em SOA são *interfaces* para cenários B2B (*Business-to-business*) tornou-se a linguagem para negócios. Esses cenários são organizados de tal forma que um sistema ou componente se comunica de tempos em tempos com outro sistema para ler ou escrever alguns dados. Entre essas chamadas de serviços, o consumidor do serviço utiliza a sua lógica de negócio para executar a sua tarefa. [Josuttis, 2008].

Isso significa que um *front-end* que age com um consumidor do serviço para alguma extensão controla o comportamento global. Isso é diferente de *front-end* que quase não possui nenhuma lógica de negócio apenas apresenta dados.

Em SOA, um serviço serve o consumidor, ele não o controla. Isso se aplica até para serviços de processos. Por esta razão, os serviços de processos normalmente não permitem interação humana. Na verdade, BPEL (*Business Process Execution Language*) oferece uma plataforma robusta e escalável para automação de processos de negócios, na qual é uma linguagem em lotes que pode definir um serviço composto por outros serviços. O serviço pode ser de execução longa, mas ele não pode interagir com usuário.

Na prática, pode até ser melhor ter algumas regras de negócio estáveis implementadas redundantemente no *front-end* em vez de chamar um serviço para cada validação. Depois que os dados de entrada são coletados, envia a requisição de alteração para *back-end*(s) chamando direta ou indiretamente um ou mais serviços de escrita.

6.4 Validação Multicamadas

Pré-validação das entradas de usuário, a princípio, poderá usar as seguintes políticas básicas de validação/pré-validação [Josuttis,2008]:

- **Validar no *back-end*** – toda a validação pode ser feita apenas em *back-end*(s). Neste caso, *front-ends* não pré-validam nada. Com esta abordagem, se o usuário preenche muitos dados e o *back-ends* os rejeita porque o problema foi causado pelo primeiro dos muitos campos de entrada, o usuário final pode ficar muito frustrado. Detectar o problema antes teria salvado muito tempo e esforço;

- **Validar no *front-end*** – toda a validação pode ser feita somente em *front-end*(s). Neste caso, os *back-ends* assumem que os consumidores dos serviços não enviarão dados errados ou inconsistentes. É claro que está é uma abordagem arriscada, porque os *front-end*(s) podem cometer erros ou perder alguns detalhes; isso por sua vez pode levar a um comportamento indefinido nos *back-ends*, tornando-os inconsistentes e/ou fazer com que quebrem;

- **Redundância** – tanto *front-end(s)* como *back-end(s)* podem validar independentemente. O *front-end* tentará capturar possíveis problemas na frente, e o *back-end* irá assegurar que a entrada que ele recebe seja válida. A validação independente, no entanto, pode levar á inconsistências; e

- **Pontos centrais de decisão** – validação pode ser tratada por um ponto de decisão comum, que é um serviço que conhece as regras de negócio (que se aplicam aos dados específicos do usuário). Neste cenário os *front-end(s)* e *back-end(s)* são pontos de esforço que reforçam as decisões fornecidas por um fornecedor de serviços central, que tecnicamente pode ser um mecanismo de regras. O problema com esta abordagem é que ela pode levar á muitas chamadas de serviço.

Na técnica, a validação normalmente envolve uma mistura destas abordagens. *Back-ends* deveriam sempre verificar todas as entradas para validade. No geral, também seria necessário fazer com que os *front-ends* pré-

validem as entradas para os serviços. O objetivo da pré-validação deveria ser assegurar que os serviços de processos irão normalmente suceder.

Validar nos *front-ends* e nos *back-ends* levanta a questão se devem ser programados independentemente ou se devem usar serviços comuns (tais como serviços de regras). No entanto, terá que definir serviços para cada tipo de verificação que executa, e existe um problema de desempenho em chamar esses serviços repetidamente durante o tempo de execução. Na prática, deve haver um lugar central para regras de negócio que mudam com freqüência.

6.4.1 Panoramas com Múltiplos Canais de *Front-ends*

É comum que sistemas distribuídos requeiram suporte para uma abordagem de múltiplos canais. Um cliente pode iniciar um processo de negócio (como pedido) usando um portal *online*, por exemplo, e continuá-lo através de um *call center*. De forma semelhante, os processos iniciados por uma aplicação de *call center* podem ser processados por *back office* e monitorados pelo cliente usando o portal. Para suportar isso, em vez de gerenciar processo de cada canal independentemente, terá que ser capaz de alternar entre canais de *front-end* diferentes. Isso mantém o processo consistente. Isso significa que os mecanismos de processos devem prover habilidade de localizar e identificar os processos de execução. [Josuttis,2008].

Uma solução melhor provavelmente seria que o mecanismo de processos recebesse a tarefa de não apenas processar os serviços, mas também de fazê-los localizáveis enriquecendo-os com todos os critérios de busca úteis.

6.4.2 *Front-ends* de Múltiplas Aplicações

Outro cenário é que diferentes *front-ends* que executam no mesmo sistema ou *desktop* precisam dos mesmos dados fornecidos por um serviço.

Considere um *desktop* onde tenha múltiplos clientes com *interfaces* gráficas de usuário lidando com o mesmo cliente. Os *front-end* fornecem diferentes

processos de negócio de outros *front-ends*. Agora digamos que precise dos mesmos dados do cliente em outro *front-end*.

Conceitualmente, o segundo *front-end* precisaria chamar o mesmo serviço para obter os dados do cliente. No entanto, pelo fato de os dados já estarem no *desktop*, isso seria um desperdício de recursos. A alternativa é possibilitar comunicação direta ou indireta entre os dois *front-ends*: [Josuttis,2008].

- Troca de dados direta significa que os dois *front-ends* estabelecem e gerenciam uma camada de comunicação para troca de dados. É claro que isto aumentaria o acoplamento entre os clientes, o que poderia ou não ser apropriado; e
- Troca de dados indireta significa que existe alguma mágica que sabe que os dados do cliente foram carregados por um front-end e podem ser usados por outro *front-end*.

Soluções diferentes afetam diferentes áreas da programação: enquanto a primeira abordagem não tem nada a ver com SOA (ela é uma comunicação especial entre dois *front-ends* ou clientes), a segunda abordagem pode ser e é suportada pela sua infra-estrutura (ESB - *Barramento de Serviços Empresariais*), que é uma tecnologia para gerenciar os serviços de qualquer lugar em plataformas seguras e confiáveis.

Em ambos os cenários normalmente requer em algum tipo de comunicação local, porque tem que existir alguma maneira de trocar a informação indicando que um *front-end* está processando os mesmo dados que outro *front-end*. Isto pode ser considerado uma otimização, que deverá normalmente começar com os dois *front-ends* chamando o mesmo serviço para carregar os dados do cliente e ver se esta abordagem causa algum problema [Josuttis,2008].

6.5 Gerenciamento Futuro

Clientes e processos de negócio como um todo podem definir modificações que devem acontecer depois, em um determinado momento. Então, vem á pergunta como lidar com as modificações futuras? A princípio, terá as seguintes opções: [Josuttis,2008].

- Organizar o gerenciamento futuro de tal forma que os *back-ends* possam lidar com as tarefas que devem ser processadas no futuro; e
- Tornar o gerenciamento futuro parte do seu gerenciamento de processos.

Com a primeira abordagem, precisará dos serviços que permitam modificações dos estados de *back-ends* e que recebam datas específicas como parâmetros. Além disso, os *back-ends* terão que ser capazes de gerenciar as tarefas para o futuro. Este é um requisito freqüentemente difícil de satisfazer. Por essa razão, normalmente é melhor fazer com que o gerenciamento futuro seja parte do seu gerenciamento de processos [Josuttis,2008].

Mecanismo de processos será responsável por: [Josuttis,2008].

- Executar os processos diretamente;
- Colocar os processos para um processamento posterior dentro do sistema de gerenciamento de tarefas futuras; e
- Processar essas tarefas quando o sistema de gerenciamento de tarefas futuras indicarem que elas devem ser executadas.

Tal separação entre uma ferramenta ou sistema que executa as tarefas atuais e outra ferramenta que gerencia as tarefas futuras pode ajudar a separar os interesses. Mas, é claro, gerenciar as tarefas futuras e atuais concomitantemente pode também ser possível e ter algumas vantagens [Josuttis,2008].

6.6 Processos

Algo que torna os grandes sistemas complicados é que muitas pessoas e equipes diferentes estão envolvidas neles.

Esses processos incluem: Modelagem de processos corporativos (BPM – *business process modeling*), no qual sua tarefa é quebrar os processos corporativos em atividades e tarefas menores, o que neste caso são os serviços.

Desenvolvimento de software orientado a modelos (MDSD – *Model-driven software development*) é o processo de geração de código para lidar com os serviços.

6.7 Governança

O meta processo de todos os processos, e a estratégia SOA como um todo é a governança. Geralmente há uma equipe central, que poderá ser chamada de centro de competência SOA que lida com a infra-estrutura, a arquitetura e os processos.

6.8 SOA X Sistemas Homogêneos

Não se pode deixar de destacar que SOA é a solução ideal para circunstâncias muito especiais: sistemas distribuídos heterogêneos com diferentes proprietários, porém a um preço a ser pago com a heterogeneidade e com os diferentes proprietários, e por fornecer flexibilidade, escabilidade e tolerância ás falhas.

Por esta razão, caso não se possua o tipo de sistema com as características acima mencionadas, SOA não se faz necessário, se tudo estiver sob controle (por exemplo, sistemas homogêneos e/ ou nenhum proprietário diferente), SOA pode se tornar desnecessariamente cara.

SOA é apenas um meio para um fim, que é o de encontrar uma solução apropriada para as suas necessidades particulares.

6.9 SOA X Tecnologia Especifica

SOA não é a mesma coisa que *Web Service*. SOA é um paradigma; *Web Services* são uma maneira possível para fornecer a infra-estrutura com o uso de uma estratégia de implementação específica.

Isso quer dizer que *Web Services*, pode ajudar a prover infra-estrutura, mas ainda terá que construir a arquitetura e criar todos os processos complicados que são necessários para uma SOA bem-sucedida. È possível implementar SOA com outras tecnologias.

SOA como conceito não está amarrado a nenhuma tecnologia específica.

6.9.1 Terminologia SOA

- **Fornecedor** – é um sistema que implementa um serviço (uma funcionalidade de negócio) de tal forma que os outros sistemas possam chamá-lo; e
- **Consumidor** – é um sistema que chama um serviço (usa um serviço fornecido).

7 SERVIÇOS

Os Serviços – são porções de *software* construídas de tal modo que possam ser facilmente vinculadas a outros componentes de *Software*, sendo que no centro deste conceito de serviços está a idéia de que é possível definir partes dos códigos em diversas áreas da organização, com isso algumas tarefas passam a ser automatizadas. Para alcançar essa meta, os desenvolvedores criam um envoltório complexo em torno do código empacotado, essa caixinha é uma *interface*, na qual, descreve o que a porção faz e como vincular a ele.

Existem muitas maneiras diferentes de conectar serviços, como *links* de programação customizados ou *software* de integração de fornecedores. O que diferencia SOA de outras arquiteturas é a sua possibilidade de abrangência para

criar aplicações dentro de uma empresa – pensamos como num projeto arquitetônico - mas, neste caso, a arquitetura demanda que todos os programas sejam criados com uma metodologia de desenvolvimento de *software* específica, conhecida como programação orientada a serviço, assim sendo SOA é uma estratégia de TI.

Sendo uma estratégia arquitetural, SOA envolve muito mais do que o mero desenvolvimento de *software*. A criação de uma arquitetura baseada em um portfólio de serviços demanda que os *CIOs* (responsável pela tecnologia da informação numa empresa) elaborem um "*case*" convincente para uma arquitetura corporativa, uma metodologia de desenvolvimento centralizada e uma equipe centralizada de gerentes de projeto, arquitetos e desenvolvedores.

O ponto de fusão seria a governança, para que os serviços sejam reutilizados na empresa, tem de haver uma metodologia de desenvolvimento de *software* única e centralizada de modo que áreas diferentes não criem o mesmo serviço de maneiras diferentes ou usem conectores incompatíveis. Tem que haver um repositório centralizado para que os desenvolvedores saibam onde procurar serviços - e a TI saiba por quem eles estão sendo utilizados. Os serviços têm de ser bem documentados para que os desenvolvedores saibam para que eles servem, como integrá-los e as regras para usá-los. Para se obter sucesso as empresas que avançaram no caminho para SOA criou um grupo de arquitetura centralizado para escolher processos que serão capacitados para serviço e consultar áreas diferentes da empresa para criar os serviços específicos.

7.1 Serviços Representam Funcionalidade de Negócio

SOA é focada em processos corporativos. Esses processos são executados em diferentes passos (também chamados de atividades ou tarefas) em sistemas diferentes. O principal objetivo de um serviço é representar um passo "natural" da funcionalidade de negócio. Em resumo seria as pessoas de negócio deveriam ser capazes de entender o que um serviço faz.

7.1.1 *Interfaces* e Contratos

Tecnicamente falando, um serviço é uma *interface* para (múltiplas) mensagens que retornam informação e/ ou alteram o estado de uma entidade associada (*back-end*). Sendo que uma *interface* orientada a negócio é diferente de uma *interface* orientada a tecnologia. Para citar [Josuttis, 2008].

Um tipo de *interface* é uma assinatura, que descreve parâmetros de entrada, parâmetros de saída e possíveis exceções.

Sendo o contrato uma especificação completa de um serviço entre um fornecedor específico e um consumidor específico. Do ponto de vista do consumidor, isso define “tudo o que precisa saber quando usar esse serviço”, de tal forma que (preferencialmente) não fique nenhuma dúvida, e na prática, normalmente começa a descrever um serviço com uma *interface* bem definida. Para citar [Josuttis, 2008].

Todas as definições de SOA concordam que é um objetivo da modelagem que os serviços sejam independentes (autônomos, autárquicos), pode-se citar como exemplo: os serviços podem compartilhar ao menos alguns tipos de dados fundamentais (tais como *string*), sendo o objetivo principal minimizar as dependências de tal forma que SOA seja apropriada para sistemas distribuídos com diferentes proprietários.

Os serviços são abstrações que escondem os detalhes da implementação dos consumidores, por está razão, normalmente é melhor ter uma chamada de serviço transferindo todos os dados necessários entre o fornecedor e os seus consumidores, em vez de ter múltiplas chamadas de serviço processando a mesma quantidade de dados.

Definição de Serviço segundo o modelo de referência de SOA da OASIS. Para citar veja [OasisSOARM,2006].

O substantivo “serviço” é definido em dicionários como “a execução do trabalho (uma função) por um para outro". No entanto, o serviço, da forma que o termo é geralmente entendido, também combina as seguintes idéias relacionadas:

- A capacidade de executar o trabalho para outros;
- A especificação de um trabalho oferecido para outros; e

- A oferta para executar um trabalho para outros.

Em SOA os serviços são o mecanismo através do qual as necessidades e as capacidades são reunidas. Processar muitos dados leva tempo e se o consumidor não precisa desses dados, esse tempo é desperdiçado. Para citar [Josuttis,08].

Deve-se levar em conta, que para se chamar um serviço, deve-se saber que o serviço existe. Ás vezes os serviços definidos (preferencialmente) como sem estado, as questões são onde e durante quanto tempo o estado existe, e se isso é um estado técnico ou de negócio.

Se, ocorre uma chamada de serviço que modifica alguma coisa e não obtém nenhuma resposta confirmativa, terá um problema. E para resolver o mesmo, a idem potência, tem a habilidade de refazer uma operação caso não tenha certeza que a mesma foi completada.

Evitar a redundância é um objetivo geral do desenvolvimento de *software*, ou seja, idealmente cada funcionalidade deve ser implementada apenas uma vez. Um caso comum de negócio para SOA é que ela leva a uma reusabilidade melhor porque todos os sistemas que precisam de certa funcionalidade apenas chamam o mesmo serviço.

Os serviços podem usar/chamar outros serviços, ou seja, as funcionalidades de negócio maiores podem ser quebradas em passos menores, que por sua vez também são serviços.

As definições de SOA permitem também os chamados “serviços técnicos” que não necessariamente representam funcionalidades de negócio independentes, caso tenha uma infra-estrutura SOA para os serviços corporativos, use essa infra-estrutura para trocar dados técnicos. Para citar [Josuttis,08].

A alta interoperabilidade é um requisito fundamental de SOA, pelo fato de os sistemas serem chamados a partir de qualquer outro sistema, eles são inerentemente interoperáveis.

7.2 Classificação de Serviços

Dentro de um mesmo contexto ou cenário, os serviços podem diferir. A variedade das funcionalidades de negócio é enorme, e os serviços servem a propósitos muito diferentes e desempenham papéis muito diferentes. Alguns serviços lêem dados; outros os escrevem. Alguns serviços têm efeitos simples e rápidos; outros executam processos complicados. Os serviços podem ter acesso a nenhum, um ou múltiplos *back-ends*. Assim como acontece com objetos na programação orientada a objetos, existe uma variedade imensa de serviços. [Josuttis, 2008].

Classificação de serviços está relacionada a três categorias básicas: [Josuttis, 2008]:

- Serviços Básicos;
- Serviços Compostos; e
- Serviços de Processos.

E com base nestas três categorias pode-se definir três camadas de serviços diferentes e estágios de expansão:

- O **primeiro** estágio de expansão é o chamado SOA fundamental, que tem apenas uma camada de serviços básicos;

- O **segundo** estágio de expansão é o chamado SOA federativa que, além dos serviços básicos, tem uma camada de serviços compostos (esta camada pode ser chamada de camada de orquestração ou composição); e

- O **terceiro** estágio de expansão é o chamado SOA habilitada para processos, que tem uma terceira camada para serviços de processos.

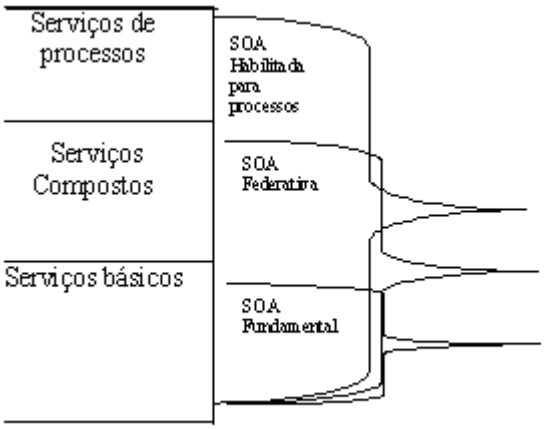


Figura 5– Camadas Básicas de serviços e estágios de expansão de SOA [Josuttis,2008]

7.2.1 Serviços Básicos [Josuttis,2008]

O primeiro estágio de expansão provê apenas os serviços básicos, que são os serviços onde cada um deles provê uma funcionalidade de negócio básica e que não faz sentido dividir em múltiplos serviços. Normalmente, estes serviços fornecem a primeira camada fundamental de negócio para um *back-end* específico ou domínio do problema. Tipicamente, estes serviços são de execução rápida e conceitualmente sem estado. O papel destes serviços é encapsular um *back-end* ou um domínio do problema de tal forma que os consumidores (e serviços de níveis mais altos).

Existem dois tipos de serviços básicos:

- Serviços básicos de dados; e
- Serviços básicos de lógica.

7.2.2 Serviços Básicos de Dados

Os serviços básicos de dados lêem ou escrevem dados de ou para um sistema de *back-end*. Tipicamente cada um desses serviços representa uma operação de negócio fundamental do *back-end*.

Os serviços básicos encapsulam os aspectos específicos da plataforma e os detalhes da implementação do mundo externo para os consumidores possam requisitar um serviço sem saber como o mesmo é implementado.

Estes serviços devem prover algumas funcionalidades mínimas de negócio, não deve haver nenhuma *interface* técnica, os serviços básicos de dados tipicamente ou lêem e retornam ou escrevem blocos de dados que normalmente pertencem uns aos outros, de acordo com as necessidades do negócio, ou seja, as pessoas de negócio devem ser capazes de entender o que estes serviços fazem. [Josuttis,2008].

Define "orientações para alcançar um nível ótimo de granularidade para o nível mais baixo de serviços" [Allen,2006].

- Deve ser possível descrever os serviços em termos de função, informação, objetivos e regras, mas não em termos de grupos ou outros serviços;
- O conjunto de funções de um serviço deve operar como uma unidade de família que oferece capacidade de negócio;
- Um papel único deve tomar a responsabilidade pelo serviço; e
- O serviço deve ser auto-suficiente o máximo possível.

Sendo assim importante que estes serviços encapsulem ou abstraiam de tal forma que alguns problemas sejam resolvidos. Neste caso, os serviços devem encapsular os detalhes técnicos e aliviar o perigo das inconsistências no *back-end*, e é um objetivo importante que estes serviços não sejam capazes de criar inconsistências no *back-end* em que operam, na verdade, os serviços básicos devem ter as chamadas propriedades **ACID**, o que significa que eles devem ser: [Josuttis,2008]

- ***Atômicos*** – isto significa que uma chamada de serviço tem que ser bem-sucedida ou não ter efeito algum, assegurar esta propriedade é tarefa do *back-end* que provê o serviço;
- ***Consistentes*** – isso significa que depois da chamada do serviço, o *back-end* é deixado em um estado válido e consistente;

- ***Isolados*** – isso significa que um serviço que está sendo processado pelo *back-end* não é influenciado por outras chamadas de serviço que executam no mesmo *back-end*, e ao mesmo tempo, ou seja, um serviço de leitura deve retornar dados que sejam consistentes; e
- ***Duráveis*** – isso significa que depois que acontece a chamada do serviço, é garantido que o efeito seja persistentes, ou seja, nenhuma falha do sistema é capaz de desfazer o resultado da chamada do serviço por acidente.

7.2.3 Serviços Básicos de Lógica

Os serviços básicos de lógica representam as regras fundamentais de negócio, estes serviços normalmente processam alguns dados de entrada e retornam os resultados correspondentes, sendo que os serviços básicos de lógica são uma minoria.

No entanto, esses serviços são também freqüentemente encapsuladores dos sistemas de *back-end* que provêem essa funcionalidade. A fronteira entre os serviços básicos de lógica e os serviços básico de dados nem sempre é clara. Sendo que não acarreta problema porque no SOA todos os serviços básicos, sejam eles serviços de dados ou de lógica, são normalmente tratados de forma semelhante, se existe uma diferença significante nos serviços básicos, ela está normalmente entre os serviços de leitura e escrita. [Josuttis,2008].

7.3 SOA Fundamental

Serviços básicos é o primeiro estágio de expansão de SOA, chamado de SOA fundamental.

Ao citar *back-end* o seu significado é amplo, podendo ser qualquer coisa responsável por um grupo específico de dados e funcionalidade, sendo seu ponto importante o fato de desempenhar um papel claro e mais ou menos único.

Para ser, mas técnico um *back-end* pode ser um banco de dados, um computador, um mainframe, um sistema SAP, um mecanismo de regras, um grupo de servidores J2EE, uma conexão remota com outra empresa, e assim por diante.

Pode ser constituído de um ou de múltiplos sistemas. É tarefa dos adaptadores dos serviços básicos esconderem os detalhes técnicos e fornecer uma API de serviço comum para que o *back-end* possa ser acessado por meio de ESB. [Josuttis,2008].

Um *back-end* é normalmente algo que pode ser identificado em suas estruturas organizacionais. É um sistema (ou grupo de sistemas) que, do ponto de vista de negócio, desempenha um papel específico e é mantido por um grupo específico de pessoas.

Os *back-ends* dos sistemas de TI contêm e gerenciam os dados e as funcionalidades de negócio centrais de uma forma muito condensada (sem recursos adicionais como *interfaces* de usuários ergonômicas e similares), e pelo fato desses dados representarem à essência da empresa, erros e usos inadequados podem ter conseqüências importantes.

A consideração mais importante para os *back-ends* é que eles não podem se tornar inconsistentes ou corrompidos. Os serviços básicos, portanto, devem assegurar que os sistemas externos possam acessar esses *back-ends* sem serem capazes de corrompê-los, ou seja, eles devem esconder os detalhes técnicos, um serviço básico deve representar uma funcionalidade de negócio auto-suficiente. Esta funcionalidade deve ser atômica, de forma que seja ou bem-sucedida ou não tenha efeito algum. Em todos os casos, o *back-end* permanece consistente.

O serviço básico nunca deve acessar mais de um *back-end*, ele deve encapsular esse *back-end* único (o qual internamente pode ser composto de múltiplos sistemas) de tal forma que sistemas externos possam fazer algo útil com os dados no *back-end*.

Nos grandes sistemas sempre há múltiplos *back-ends* com dados redundantes, e não é tarefa dos serviços básicos assegurarem a consistência por meio de *back-ends*, isso é responsabilidade dos serviços mais altos ou (se esses ainda não existem) dos *front-ends* que chamam os serviços básicos.

A vantagem desta abordagem é que fica claro qual equipe é responsável pelos serviços básicos (a equipe também responsável pelo *back-end*), essa equipe pode se concentrar em preencher a necessidade para prover os acessos externos aos seus *back-ends* sem o perigo dele se tornar corrompido.

Os sistemas que utilizam os serviços básicos podem ser muito diferentes. Eles podem ser simples clientes ou programas em lotes, ou clientes multicamadas. Do ponto de vista deles, chamam os serviços básicos de arquivo e recuperação de dados ou provêem as transformações de dados básicas. Isto é, os serviços básicos são a sua camada mais baixa. Podem existir camadas adicionais para processamento de negócio adicional e apresentação. Essas camadas podem até ser separadas fisicamente como, por exemplo, em um *software* de portal onde um servidor fornece os conteúdos para um navegador de *Internet*. [Josuttis,2008].

SOA não requer que as *interfaces* desses serviços básicos sejam harmonizadas, os nomes e os atributos dos serviços básicos semelhantes podem diferir, e é tarefa do consumidor dos serviços básicos lidarem com essas diferenças. No entanto, não faz sentido que cada *front-end* tenha que lidar constantemente com o fato de que existem múltiplos *back-ends* que lidam com os dados dos clientes. Seria preferível que um *front-end* fosse capaz de fazer as mudanças globais por meio de *back-ends*. É onde os serviços compostos entram em jogo. [Josuttis,2008].

7.3.1 Serviços Compostos

O segundo estágio de expansão adiciona os serviços compostos. Estes representam á primeira categoria de serviços que são compostos por outros serviços (básicos e / ou outros serviços compostos) [Josuttis,2008].

Na terminologia de SOA, compor novos serviços partindo dos serviços já existentes é chamado de orquestração [Josuttis,2008].

Estes serviços operam em um nível mais alto que os serviços básicos, mas eles ainda são considerados de execução rápida e conceitualmente sem estado. Para usar um termo de *workflow*, um serviço composto representa um *micro flow*, que é um conjunto de atividades rápidas (serviços neste caso) dentro de um processo de negócio. [Josuttis,2008].

7.3.2 Serviços Compostos para Múltiplos *Back-ends*

Os serviços compostos são tipicamente os serviços que acessam múltiplos *back-ends* e, por isso são compostos de múltiplos serviços básicos. Um exemplo típico é um serviço que atualiza dados redundantes em múltiplos *back-ends*, por exemplo, os grandes sistemas podem conter os mesmos dados dos clientes em *back-ends* diferentes (em um sistema local a uma empresa os dados redundantes dos clientes podem ser encontrados em sistemas de CRM e de faturamento, enquanto que em um sistema espalhando por várias empresas os mesmos dados dos clientes podem ser encontrados em cada sistema de CRM dessas empresas). Outro exemplo seria um serviço que transfere dinheiro de um *back-end* para outro, neste caso, o serviço composto chamaria um serviço básico que retira o dinheiro de um *back-end* e outro serviço básico que deposita o dinheiro em outro *back-end* [Josuttis2008].

7.3.3 Serviços Compostos para um *Back-end*

Os serviços compostos podem também mapear ou adaptar os serviços existentes de alguma forma. Tais serviços também podem ser citados como "serviços adaptadores". Estes serviços podem ser úteis para fornecer uma *interface* diferente para um serviço que, por exemplo, tem um nome diferente. As aplicações típicas seriam para prover compatibilidade com versões anteriores ou para mapear serviços para uma *interface* requerida [Josuttis,2008].

7.4 SOA Federativa

Quando existem os serviços básicos e os compostos, isso leva ao estágio de expansão I chamado de SOA federativa [KrafzigBankeSlama,2004] o chamam de SOA em rede. Este fase introduz uma camada adicional para os compostos.

A compensação neste cenário significa que possa chamar ambos os serviços independentemente de outro, e se apenas um serviço básico for bem-sucedido enquanto outra falha, você "compensa" fazendo algo apropriado, o que é uma reação apropriada depende das circunstâncias. Por exemplo, chamar um

terceiro serviço para desfazer os efeitos da chamada de serviço bem-sucedida (por exemplo, o primeiro *back-end* pode prover um serviço adicional que seja capaz de desfazer a retirada do dinheiro e que pode ser chamado pelo serviço composto se a chamada de serviço no segundo *back-end* falhar). Outra reação seria colocar esse caso em um *desktop* de erros onde a inconsistência pode ser resolvida manualmente (esta abordagem também pode ser usada em caso de *rollback* falhar) [Josuttis,2008].

Embora a compensação torne o processo de negócio mais complicado, ela tem diversos benefícios [Josuttis,2008]:

- Os *back-ends* não precisam fornecer o suporte técnico para um contexto de transação comum;
- Implementar os serviços básicos se torna mais fácil; e
- Manter a segurança da transação nos processos de negócio de longa duração é quase impossível, ou menos muito cara em termos de recursos.

A camada de serviços compostos não esconde completamente a camada de serviços básicos, ou seja, do ponto de vista de negócio, não importa se um serviço é um serviço básico ou composto. O único ponto importante é se ele preenche a funcionalidade necessária de negócio.

7.4.1 Serviços de Processos

O terceiro estágio de expansão adiciona os serviços de processos, os quais representam os *workflows* ou processos de negócio de longo prazo. Do ponto de vista de negócio, um serviço de processo representa um macro *flow*, que é uma execução demorada de atividades (serviços) ininterrupta (pela intervenção humana).

Diferente dos serviços básicos e compostos, um serviço de processo normalmente tem um estado que se mantém estável durante múltiplas chamadas.

Pelo fato dos serviços de processos serem tipicamente *staterful*, você tem que prover certos recursos em sua SOA para manter esse estado por meio de múltiplas sessões e ainda prover uma escalabilidade linear para que múltiplas chamadas de serviço possam ser tratadas com ela, por isso poderá prover a

habilidade de mudar os clientes e os canais de *front-end* durante os processos [Josuttis,2008].

Outro aspecto a ser analisado como os serviços de processos é a tolerância às falhas – ou seja, se e como guardar o estado do processo quando um sistema com o estado atual falhar. Esses requisitos extras são a principal razão para se ter uma categoria separada para esse tipo de serviços. A outra razão é que do ponto de vista de negócio, os serviços de processos de execução longa normalmente têm qualidades diferentes dos serviços compostos de execução rápida [Josuttis,2008].

7.5 SOA Habilitada para Processos

Serviços de processos leva a um estágio de expansão chamado de SOA habilitada para processos [KrafzigBankeSlama,2004]. Conforme mencionado anteriormente, a camada adicional para os serviços de processos possibilita que gerencie os processos que podem ser iniciados e controlados por diferentes *front-ends* e interrompidos por intervenção humana [Josuttis,2008].

7.5.1 Estado de Serviço Versus Estado de *Back-end*

- Se os dados do processo se tornam “dados jurídicos”, talvez seja mais apropriado gerenciar o estado em *back-end* comum; e
- Se os dados do processo são apenas alguns dados de persistência que não possuem nenhuma relevância essencial para o seu negócio, os serviços *stateful* são suficientes [Josuttis,2008].

Em outras palavras, caso tenha que ser capaz de dizer algo sobre o estado de um processo depois, os dados do processo devem ser persistidos em um *back-end*. Isso é uma questão de modelagem (influenciada pelos requisitos de negócio). O importante é entender a diferença entre o serviço de manutenção no serviço ou no *back-end*. Isso é uma questão importante que deva considerar sempre que introduzir os processos de negócio e os sistemas de *workflow*.

7.5.2 Lidar com Tipos Diferentes de Consumidores

Uma maneira de categorizar os serviços é diferenciá-los de acordo com o seu público alvo. A razão para estas distinções é que estes serviços normalmente têm requisitos diferentes em termos de segurança e estabilidade (para que as *interfaces* permaneçam inalteradas). Ter diferentes tipos de consumidores pode levar á diferentes ferramentas, processos e ciclos de vida dos serviços. Sendo assim viável lidar com essas diferenças, usando diferentes categorias de serviços para elas.

7.5.3 Serviços de Leitura Versus Serviços de Escrita

Outra distinção a ser realizada é a de serviços de leitura (serviços que não modificam o estado do *back-end* correspondente) e os serviços de escrita (serviços que o fazem).

Por definição, os serviços de leitura têm atributos especiais e normalmente não requerem os contextos de transação para *back-ends* diferentes ou *rollbacks* em casos de executarem parcialmente. Os serviços de escrita, por outro lado, freqüentemente requerem uma quantidade adicional de segurança para que as modificações possam ser rastreadas. Por esta razão, todos os serviços de escrita podem precisar de dados técnicos adicionais que identifiquem o chamador [Josuttis,2008].

Os serviços de leitura são freqüentemente menos estáveis do que os serviços de escrita. A razão é que as extensões dos serviços de leitura, não afetam os processos de negócio complicados, enquanto que modificar os atributos dos serviços de escrita pode ter impacto em muitos lugares. Pelo mesmo motivo, os serviços de escrita são mais freqüentemente usados nos serviços compostos ou orquestrados. Os serviços de leitura são usados em primeiro lugar pelo *front-ends* para exibir dados.

7.5.4 Categorização de acordo com Erl

Segundo [Erl,2005], Tomaz Erl introduz uma categorização dos serviços cujos termos são parecidos com uma classificação possível de categoria das classes na modelagem orientada a objetos. Como uma base para a definição das diferentes camadas SOA, ele introduz as seguintes categorias:

- Serviços de aplicação são serviços que “contém a lógica derivada de uma solução ou plataforma de tecnologia”; e
- Serviços de negócio são serviços que “contém a lógica de negócio”. Podem ser divididos:
  - Serviços de negócio centrados em tarefas
  - Serviços de negócio centrados em entidades

Serviços de processos são serviços que “representam um processo de negócio que é implementado por uma plataforma de orquestração e descrito por uma definição de processo”.

Serviços de negócio correspondem aos serviços básicos e serviços de processos incluem os serviços compostos. Um serviço sempre deve apresentar alguma funcionalidade de negócio.

[Erl,2005] Também introduz algumas categorias de serviços adicionais:

- **Serviços de integração** – estes são serviços de aplicação que “agem como um ponto final para um ambiente de solução para os propósitos de integração das aplicações";

- **Serviços *wrapper* e *proxy*** – estes são serviços de integração que “encapsulam e expõem” ou espelham “a lógica existente em um sistema legado”;

- **Serviços utilitários** – estes são serviços de aplicação ou de negócio que “fornecem lógica reutilizável”; e

- **Serviços controladores** – estes são serviços que agem “como serviço gerador para os membros da composição do serviço”; em outras palavras, eles coordenam outros serviços para “contribuírem para a execução de uma tarefa de negócio global”.

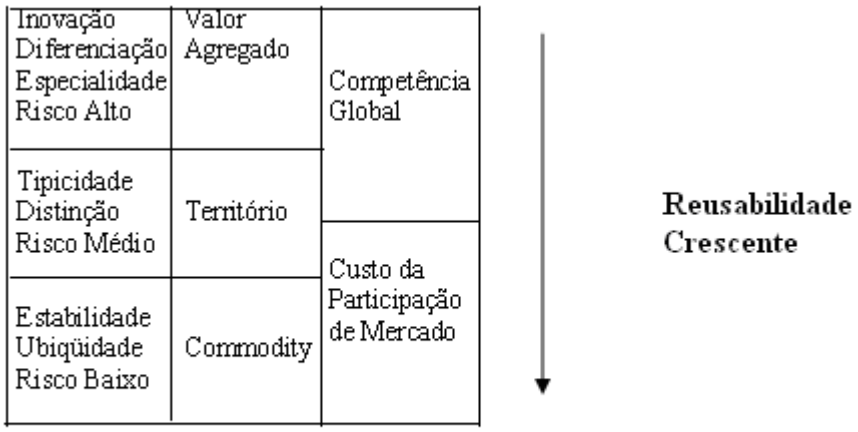


Figura 6– As características dos tipos de serviços de acordo com [Allen,2006]

Os três tipos principais são os seguintes:

- **Serviços de *commodity*** são serviços gerais que cada participante do mercado corporativo tem que possuir. Eles são serviços universais e estáveis, estabelecidos o suficiente para serem usados a baixo risco e com o máximo de reusabilidade em mente. Estes serviços são candidatos para funcionalidades terceirizadas e produtos padrões;

- **Serviços de território** representam à funcionalidade que é essencial e central para um negócio específico. Eles são comuns, porém voláteis. Os exemplos incluem os serviços que fornecem regras de negócio; e

- **Serviços de valor agregado** são serviços que refletem o valor especial que uma empresa traz ao mercado. Eles diferenciam a empresa dos concorrentes e normalmente são altamente inovadores.

7.5.5 Serviços Técnicos e de Infra-estrutura

Na maioria dos ambientes SOA existem alguns serviços comuns que não são oferecidos para representar funcionalidade de negócio. Eles são freqüentemente chamados de serviços técnicos ou de infra-estrutura. Tais serviços tipicamente utilizam a infra-estrutura de serviço para fornecer alguma funcionalidade "adicional". Os exemplos típicos são os serviços que fornecem suporte técnico ou de infra-estrutura. Podem ser [Josuttis,2008]:

- Consultar as informações sobre *deploy;*
- Monitorar as estatísticas de execução;
- Imprimir, auditar ou rastrear;
- Habilitar ou desabilitar componentes e sistemas; e
- Verificar *interfaces.*

Do ponto de vista de SOA, eles não são serviços porque serviços devem prover algum tipo de funcionalidade de negócio.

Enquanto chamar esse tipo de serviços de "serviços", deverá sempre marcá-los como algo especial. Deve ficar claro que eles servem para um propósito particular, para que as equipes de negócio não fiquem com a impressão de que serviços não necessariamente têm que apresentar funcionalidade de negócio. Se possível, talvez seja melhor evitar explicitamente chamá-los de serviços [Josuttis,2008].

Deve-se ter cuidado ao introduzir a impressão de que os serviços não precisam representar funcionalidade de negócio auto-suficiente, porque é importante manter uma camada de abstração clara que esconde os detalhes técnicos.

7.6 Gerenciamento de Serviços

***Brokers* de Serviços** – ao introduzir-se *Web Services* existem três papéis fundamentais: o de um fornecedor de serviço, o de um consumidor de serviço (requisitante) e o de um *broker* de serviço. Ele interage conforme a figura 7.

Esse triângulo reconhece que existe uma necessidade de uma maneira de juntar os fornecedores e os consumidores. Por esta razão, assim que um fornecedor começa a fornecer um serviço, ele registra o serviço em um lugar bem conhecido (ou seja, com um *broker* de serviço). Então, quando um consumidor precisa do serviço, ele pode usar o *broker* para encontrar os fornecedores desse serviço.

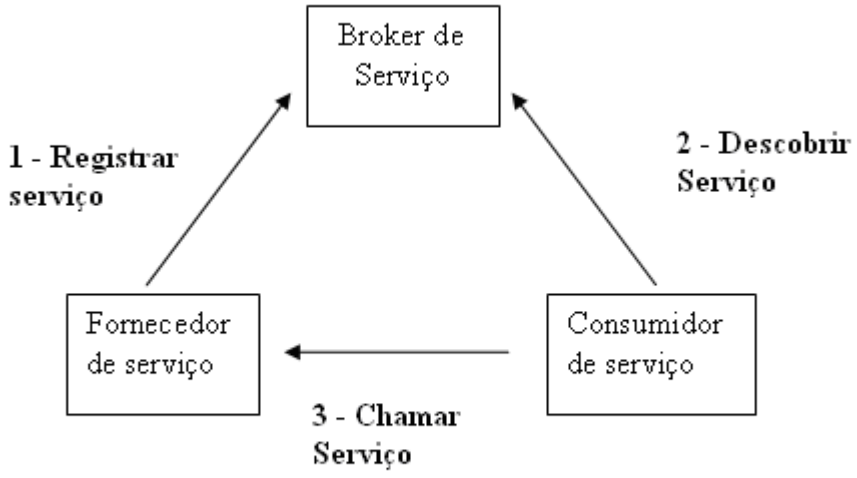


Figura 7- Fornecedor, consumidor e *broker* [Josuttis2008]

O *broker* é um repositório de serviços, em outras palavras, a necessidade de um *broker* que ajuda os consumidores a encontrarem os serviços certos surge quando o ambiente SOA atinge um determinado tamanho.

7.6.1 Repositórios e Registros

Os termos mais comuns para *brokers*, hoje em dia, são repositórios e registros.

- Repositórios gerenciam serviços e seus artefatos de um ponto de vista de negócio. Ou seja, eles gerenciam *interfaces*, contratos, SLAs, dependências, entre outros, para ajudar a identificar, modelar e desenvolver serviços. Um repositório deveria conter toda a informação sobre o comportamento e a *interface* de um serviço. Essa informação deveria ser independente dos detalhes e aspectos da infra-estrutura; e
- Registros gerenciam serviços do ponto de vista técnico. Isto é, eles gerenciam as *interfaces*, mas não todos os detalhes do contrato: eles gerenciam todos os detalhes técnicos necessários para usar um serviço em tempo de execução (isto é, informação de *deploy*) e podem ser usados para encaminhar as chamadas de serviço para diferentes sistemas que fornecem os serviços correspondentes. . [Josuttis,2008].

Tanto repositório como registro podem existir em um ambiente SOA. Sendo que repositórios e registros podem ser usados para propósitos diferentes. Por esta razão, deve-se deixar claro qual papel e responsabilidade que um repositório e/ou registro tem no seu cenário SOA concreto. Papéis e responsabilidades específicas levam á diferentes tipos de solução e a diferentes processos para lidar com essas entidades.

Um requisito geral é que esses modelos sejam transformados no formato de entrada do registro central, o qual provê as *interfaces* para os consumidores e fornecedores e efetua *deploy* da informação do serviço dentro do ESB.

7.6.2 Estabelecendo Repositórios e Registros

Repositórios e registros normalmente evoluem ao longo do tempo, á medida que a sua SOA cresce, pode-se pensar sobre maneiras gerais de projetar processos e modelar serviços. A sinergia será melhor, quando todos usarem as mesmas ferramentas, mas o esforço organizacional pode ser alto demais, e lidar

com soluções que funcionam pela metade facilmente levar á frustração nas equipes de negócio.

Ter um repositório no começo de um projeto SOA não significa que não faz sentido ter uma idéia de quais aspectos a atributos devem ser definidos para serviços.

7.7 Ciclo de vida dos Serviços

Serviços são pedaços de *software* como qualquer outra software. Por isso se aplica o ciclo de vida usual para o desenvolvimento de *software*. Algumas delas são baseadas no fato de que falamos tanto sobre o *software* sob desenvolvimento como sobre o *software* sob manutenção. Além disso, os ciclos de vida dos serviços são apenas partes dos processos de desenvolvimento mais gerais e distribuídos. O ciclo de vida do serviço se aplica á artefatos diferentes, tais como especificações dos serviços, implementações de serviços e código de teste. [Josuttis,2008].

7.7.1 Serviços Sob Desenvolvimento

Um serviço é uma representação da TI de uma funcionalidade de negócio. Por isso, os serviços são pedaços de *software*. Embora a linguagem e o formato desse software possam diferir (por exemplo, dependendo da plataforma de implementação e/ou da classificação do serviço), o seu ciclo de vida é o usual de qualquer software sob desenvolvimento, que basicamente consiste das fases de modelagem, implementação, integração e colocação em produção.

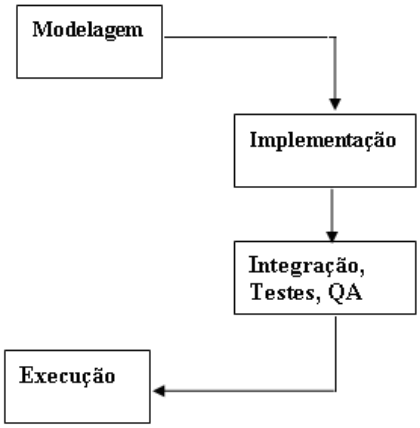


Figura 8– Um ciclo de vida típico de *software* se aplica á serviços. [Josuttis,2008]

7.7.2 Desenvolvimento Iterativo de Serviços

Uma abordagem do tipo cascata não funciona. No entanto há um aspecto especial dos serviços a ser considerado: um serviço é parte de um processo de negócio mais geral. Por isso, qualquer modificação na modelagem ou implementação de um serviço pode ter impacto em outros sistemas. A fase de modelagem normalmente produz a especificação da *interface* do serviço.

Leva-se em conta que as *interfaces* dos serviços sob desenvolvimento podem precisar de mudanças, e introduz os processos que tornam as possíveis modificações durante a modelagem de um serviço (se necessário).

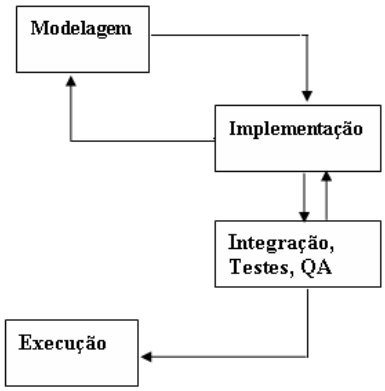


Figura 9– O ciclo de vida do *software* com iterações [Josuttis,2008]

Quando melhor a modelagem, menos mudanças são normalmente requeridas.

7.7.3 Identificação dos Serviços

Normalmente, um serviço em si não é um fim. Deve existir algo que leva á sua necessidade e modelagem, os serviços são partes de um processo de negócio generalizado. Por isso, o cenário típico que leva á criação de um novo serviço é a modelagem de um (novo) processo de negócio, também conhecido como modelagem de processos de negócio.

Novos serviços também podem ser introduzidos para atender os novos requisitos dos serviços existentes. Se uma modificação não é compatível com as versões anteriores, um novo serviço (ou nova versão do serviço) é necessário. Uma nova versão do serviço é tipicamente considerada, tecnicamente falando, um novo serviço. [Josuttis,2008].

Normalmente a fase antes da modelagem é chamada de análise, mas isso na verdade não se aplica. Segundo [Josuttis,2008] chama o passo do ciclo de vida antes da modelagem “identificação do serviço” ou “descoberta do serviço”, o que pode ser parte da modelagem da solução.

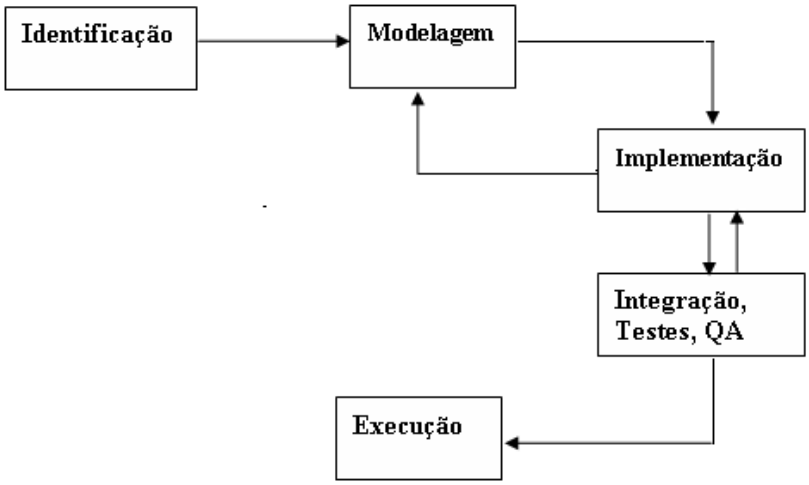


Figura 10– Começando o ciclo de vida com identificação do serviço [Josuttis,2008]

Agora este diagrama inclui todos os passos para um serviço sob desenvolvimento. Após identificar que um (novo) serviço é necessário, sua *interface* e o seu contrato são modelados, implementasse esse serviço e executasse as tarefas usuais para verificar a sua qualidade (pode ser chamado esta fase de integração, testes, controle de qualidade). Não se deve esquecer que o processo é iterativo, sendo que testes e integrações podem levar a implementações modificadas, e a modelagem do serviço em si pode precisar de alteração.

7.7.4 Serviços em Produção

Após o serviço ser desenvolvido, normalmente ele é usado, ou seja, o software é colocado em um sistema em execução, executando funcionalidade de negócio mais ou menos de missão crítica para companhia, empresa ou mundo.

Este é o momento em que o *software* que implementa um serviço passa de *software* sob desenvolvimento para *software* sob manutenção. Em geral, SOA é um conceito que combina *software* novo sob desenvolvimento com o *software* existente sob manutenção. No entanto, para os sistemas sob manutenção regras diferentes se aplicam. Por exemplo, as modificações se tornam muito mais críticas, especialmente se os sistemas possuem proprietários diferentes. Por esta razão, temos que ver também como modificar e retirar os serviços em produção. [Jusuttis,2008].

7.7.5 Modificação Serviços em Produção

Assim que um serviço esteja em produção, ele pode ser usado em cenários de missão crítica e processos de negócio. Isso significa que toda vez que modificar tal serviço (ou *interface* ou apenas sua implementação), terá sido modificado processos de negócio existentes que estão em uso. E se isso resultar em processos de negócio quebrados poderá acarretar problemas.

Na realidade, a melhor prática para serviços em produção é que sejam estáveis. Toda vez que precisar mudar o comportamento de um serviço em produção deve fazê-lo introduzindo um novo serviço ou uma versão nova do serviço, que é independente da versão anterior do serviço, também seria uma boa abordagem que o novo serviço (versão) não tenha nenhum impacto no comportamento existente. No entanto, existe uma exceção comum para esta regra: correções de *bugs*. Em outras palavras, a diferença entre uma correção de *bug* e uma modificação é que normalmente irá querer aplicar a correção para o comportamento existente assim que possível. [Jusuttis,2008]

Quando os serviços estão em estado de produção (a fase de execução) poderá corrigir *bugs*, mas isso jamais deve ter impacto na modelagem. Se a correção de *bug* tem impacto na modelagem, então se torna uma modificação, o que deve resultar em um novo serviço (versão) que tem o seu próprio ciclo de vida.

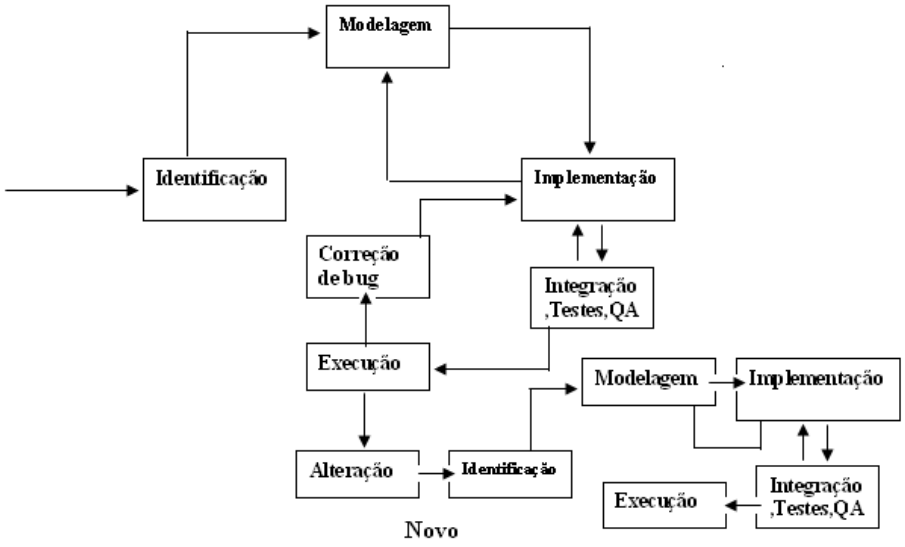


Figura 11– Ciclo de vida do serviço com correções de *bugs* e modificações [Josuttis,2008]

Pode-se modificar um serviço em execução sem trazer um novo serviço (versão) em produção, contanto que seja do interesse de todos os sistemas consumidores executarem a modificação. No entanto, deve decidir essas coisas com base em um conceito geral para que todos os participantes do serviço sejam cientes das regras, processos e ciclos de vida.

7.7.6 Retirando um Serviço

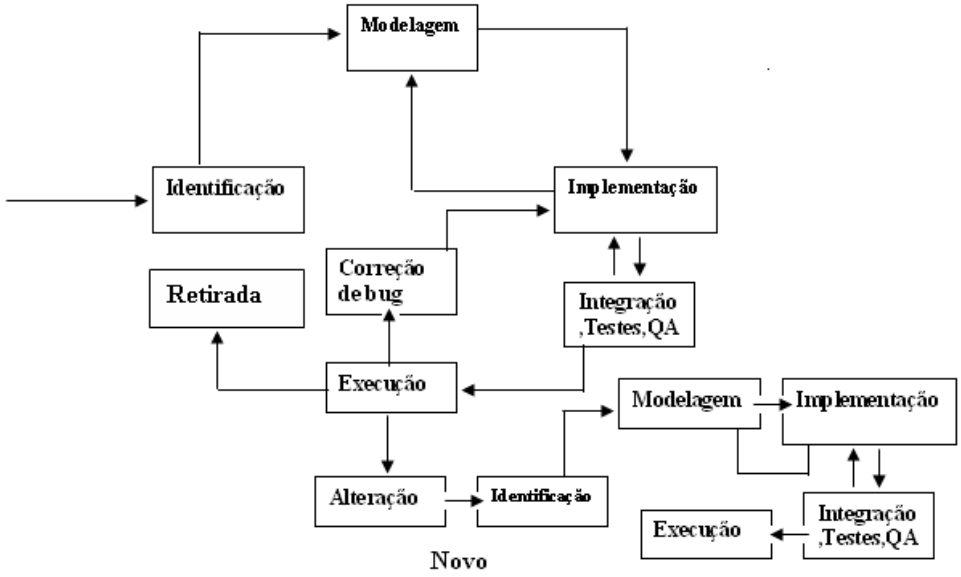


Figura 12– Ciclo de vida do serviço completo [Josuttis,2008]

A retirada é o passo do ciclo de vida do serviço. Para que os sistemas sejam de fácil manutenção, normalmente é importante evitar o inchaço de número de serviços e versões de serviços.

O fato de os sistemas distribuídos terem proprietários diferentes complica as coisas mais adiante, porque a retirada do serviço tem que ser do interesse do fornecedor do serviço quando do(s) consumidor (es) do serviço. Não faz sentido nenhum remover um serviço sem a colaboração do(s) consumidor (es).

Então, qual a maneira correta de retirar um serviço em um sistema distribuído? Segundo [Jusuttis,2008], deve-se seguir um conjunto de procedimentos:

1. Primeiro marca o serviço como obsoleto (com a sugestão de qual serviço utilizar no lugar).
2. Então monitora se (e por quais sistemas) o serviço ainda está sendo usado. Obviamente isso exige código de monitoração, que normalmente deve ser parte do seu barramento de serviços corporativos.
3. Se o serviço continua a ser usado, você deve contatar o(s) consumidor(es) relevante(s) e discutir uma solução. Como resultado, deverá ser capaz de produzir uma agenda para a retirada final do serviço.

Deve-se tentar criar um entendimento comum do benefício de uma atualização para todo o sistema ou empresa.

7.8 SOA e Desempenho

Os sistemas de TI têm dois aspectos que repetidamente quebram planos, conceitos e modelagens: desempenho e segurança.

7.8.1 Onde Desempenho Importa

Há diferentes lugares onde o desempenho entra em jogo no contexto de serviços e infra-estrutura de SOA. Se o desempenho se torna crítico, isso é normalmente por causa do tempo de execução de um serviço. Vamos explorar isso introduzindo uma chamada de serviço síncrona bem-sucedida que inclui uma mensagem de requisição do consumidor para o fornecedor e uma mensagem de resposta do fornecedor de volta para o consumidor.

Sendo que primeiramente tem que serializar a chamada de serviço no formato do protocolo de barramento de serviços corporativos. Então, enviar a requisição para o fornecedor através do ESB. O fornecedor por sua vez tem que

desserializar a requisição para que possa processá-la. Para a resposta os passos são os mesmos, sendo que na ordem inversa. Segundo [Jusuttis,2008], os seguintes passos podem ser críticos:

- Enviar a requisição e resposta através de ESB;
- Desserializar a requisição ou resposta; e
- Processar a requisição.

Sendo que SOA é um conceito para cenários de sistemas heterogêneos, e precisa de um protocolo intermediário que o consumidor e o fornecedor possam usar para trocar dados. Hoje em dia esse formato é normalmente baseado em XML (por exemplo, SOAP - *Simple Object Access Protocol,* que é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, para uma infra-estrutura de *Web Services*).

O uso de um protocolo intermediário também pode afetar o desempenho da desserialização. Isto pode (mas normalmente não deveria) ser crítico, porque pode levar tempo para transformar uma seqüência de *bytes* em dados complexos.

Na prática, a principal razão pelos serviços lentos é que a sua implementação leva tempo. Ou seja, o fornecedor de serviço precisa de tempo para processar a requisição e prover a resposta. Neste caso, tem que haver o tempo usual de resposta de uma chamada de serviço e considerar o tanto que esse tempo de resposta pode escalar bem. Por esta razão, um contrato de serviço deve incluir um acordo no nível de serviço (SLA - *service level agreement),* que é um contrato entre um fornecedor de serviços de TI e um cliente especificado, em geral em termos mensuráveis, quais serviços o fornecedor vai prestar, qual o tempo médio de resposta quanto ao número de chamadas a serem respondidas em um determinado período de tempo.

As redes e os protocolos não confiáveis também podem ter um impacto no desempenho de um sistema. Uma boa prática para colecionar as estatísticas sobre os tempos de execução de diferentes requisições de serviço é monitorar os números ao longo do tempo, isso pode ajudar a identificar os problemas mais imediatos e as tendências futuras.

7.8.2 Desempenho e Reusabilidade

Freqüentemente é recomendado que os serviços sejam de granularidade grossa. No entanto, além do problema de definir quando uma granularidade se torna grossa, esta recomendação é altamente influenciada pela sua ligação com desempenho, segundo [Jusuttis,2008]:

- Quando o *overhead* de muitas chamadas de serviço de granularidade fina se torna muita grande, se comparado a uma chamada de serviço de granularidade grossa, as questões de desempenho podem levar a serviços de granularidade mais grossa; e
- Por outro lado, quando o preço para processar mais dados fica muito alto, as questões de desempenho podem levar á serviços de granularidade mais refinadas.

7.8.3 Restrições de Chamada

Um problema com o desempenho é que ele é difícil de prever. O serviço ou uma operação de serviço não deve ser tão genérico a ponto de retornar estruturas diferentes. Ou, do ponto de vista de negócio, um serviço deve executar uma funcionalidade de negócio concreta. Normalmente, o limite para granularidade grossa é quando o serviço se torna genérico (sem ter um requisito de negócio para *interfaces* genéricas, como em caso quando as estruturas mudam freqüentemente).

7.8.4 Serviços Customizados

É uma inovação, mas é importante para SOA por outro motivo: ele demonstra que podem existir serviços que são projetados individualmente para um consumidor específico. Esta divisão de dados específica não faz sentido em nenhuma outra circunstância. Na verdade, em uma situação como esta, o

melhor é trazer junto o arquiteto do sistema consumidor e o arquiteto do sistema fornecedor para discutirem a modelagem exata do serviço. Em síntese, este é um serviço customizado, que demonstra novamente que nem todos os serviços se encaixam na categoria de serviços reutilizáveis. [Jusuttis,2008].

7.8.5 Reusabilidade na Prática

Enquanto que na teoria talvez precise de apenas um serviço que retorna todos os dados de cliente, na prática é mais provável que acabe precisando de vários serviços fornecendo diferentes (ou até customizadas) visualizações dos dados do cliente.

Um fato interessante é que o número médio de consumidores por serviços pode ser abaixo de um. Não deveria cada serviço ter ao menos um consumidor? Na prática este nem sempre é o caso, por três motivos principais, segundo [Jusuttis,2008]:

- Em grandes sistemas, muitos *softwares* implementados nunca são colocados em produção;
- Se os serviços fornecidos de acordo com a abordagem de gerenciamento de portfólio, sempre existe apenas uma suposição que os serviços serão consumidos; e
- Os serviços podem não estar mais em uso porque os consumidores mudaram para outros serviços.

7.8.6 Desempenho e Compatibilidade com Versões Anteriores

As preocupações com desempenho também podem ter impacto na compatibilidade com versões anteriores de um sistema modificado.

Quase todas as grandes corporações que têm milhares ou milhões de clientes categorizam esses clientes em diferentes grupos. Os melhores clientes obtêm certos privilégios: pode haver programas especiais de bônus para ligá-los á empresa ou eles podem ser encaminhados aos funcionários

mais experientes do *call center* porque seus problemas tendem a ser mais complicados ou eles requerem serviço melhor.

Segundo [Jusuttis,2008] existem algumas lições importantes para aprender sobre SOA e grandes sistemas distribuídos:

- Primeiro, em grandes e complicados sistemas é impossível saber ou antecipar tudo. O envolvimento de diferentes pessoas, departamentos e sistemas podem ocultar coisas óbvias;
- Segundo, nenhuma solução pode se provar como sendo uma bala mágica. Neste caso, a abordagem de processar os valores dos clientes de noite não seria na verdade melhor, porque processar milhões de valores dos clientes cada noite iria requerer muito mais recursos do que processar dezenas de milhares de valores dos clientes (isto é, apenas os valores daqueles clientes que fizeram a ligação) cada dia em tempo de execução; e
- Terceiro, mesmo que reconheça o problema e decida que uma solução diferente (como processar os valores dos clientes toda noite) seria uma abordagem melhor, você pode ter problemas ao realizá-la. Novamente, proprietários diferentes de domínios diferentes são a razão: as pessoas ou o departamento responsável por manter o banco de dados, incluindo fornecer os trabalhos em lotes para ele, podem não ser as mesmas pessoas ou departamento responsável por fornecer serviços de cliente. Repriorizar os recursos em departamentos diferentes para consertar essa modelagem imediatamente pode se mostrar difícil (mesmo que, a princípio a colaboração funcione).

Do ponto de vista do processo de negócio, a maneira como os valores dos clientes são processados é um detalhe técnico, mas esse detalhe pode quebrar todo o processo de negócio.

7.9 SOA e a Tecnologia Web Services

SOA 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 utilizam de *Web services*. Entretanto, uma implementação de SOA pode se utilizar de qualquer tecnologia padronizada baseada em *web*.

A base do padrão *Web services* relevantes ao SOA incluem:

- **XML (*eXtensible Markup Language*)** – é uma linguagem de marcação para descrever dados em cargas de mensagens em um formato de documento;
- **SOAP (*Simple Object Access Protocol*)** - é um protocolo baseado em XML utilizado para trocas de informações num ambiente distribuído. A utilização de XML é uma forma eficiente para trocas de informações mais não é suficiente para trocas de dados via *Web*. Ainda precisamos de um protocolo em comum para envio de documentos XML pela *Web* de tal forma que o receptor entenda o que esta recebendo e o que deve ser respondido. É neste ponto que entra o SOAP;
- **WSDL (*Web Services Description Language*)** - é um documento XML que descreve um conjunto de mensagens SOAP e a forma como essas mensagens são trocadas. Como o WSDL é XML, ele é legível e editável, mas na maioria dos casos, ele é gerado e consumido pelo *software*. Para enxergar o valor do WSDL, imagine que você quer invocar um método SOAP que é fornecido por um de seus parceiros de negócios; e
- **UDDI (*Universal Description, Discovery, and Integration*)** - UDDI é uma das formas de localizar uma Web *Service* num registro, como se fosse um catálogo de páginas amarelas, de tal forma que um programa em busca de um determinado serviço possa facilmente localizar e entender o que ele faz.

Segundo [ARSANJANI,2004] a Arquitetura Orientada a Serviços pode ser bem representada a partir do seguinte modelo envolvendo três partes principais : o fornecedor do serviço, o usuário e o registro do serviço , também chamado de "*find-bind-execute paradigm* " o que significa paradigma de "procurar-consolidar-executar" conforme melhor demonstrado na Figura 13.

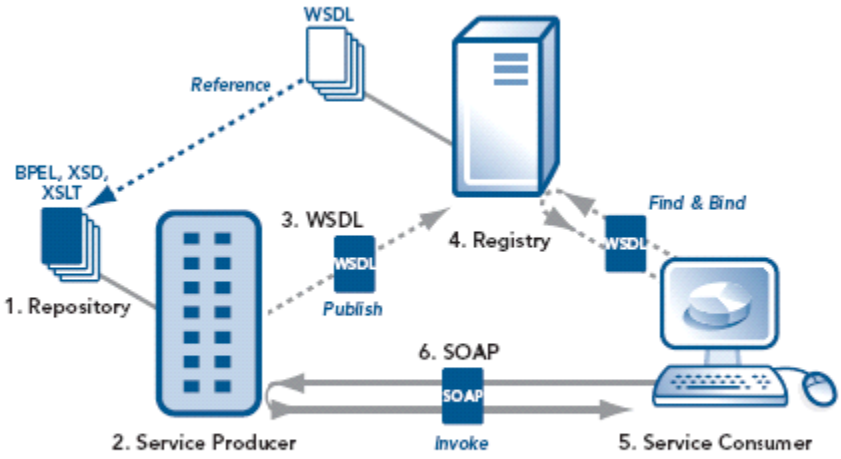


Figura 13- Paradigma "procura-consolida-executa" [adaptado de ARSANJANI, 2004]

Tecnicamente falando, o processo preconiza que os provedores de serviços registrem informações em um registro central, com suas características, indicadores, e aspectos relevantes às tomadas de decisões. O registro é utilizado pelo cliente para determinar as características dos serviços necessários, e se o mesmo estiver disponível no registro central, como por exemplo, por um catálogo de serviços, o cliente poderá utilizá-lo, sendo este oficializado através de um contrato de serviço que enderece este serviço. A comunicação entre fornecedor e consumidor do serviço segundo o paradigma “procura-consolida-executa” é realizado através de um Contrato de Serviço.

7.10 Gerenciamento de Processos de Negócios

Durante as últimas décadas a Tecnologia da Informação (TI) tem buscado mudanças rápidas para a melhoria dos processos de negócios. A Arquitetura Orientada a Serviços (SOA) permite criar uma arquitetura que aproveita os investimentos já existentes de infra-estrutura e *software*.

SOA visa maximizar a reutilização de serviços existentes e a integração com novos serviços para aumentar a eficiência e produtividade de uma empresa. Entre os principais benefícios da SOA está à total interoperabilidade entre todos os padrões do mercado, isso significa que agora os usuários não precisarão mais se preocupar com a falta de integração entre tecnologias.

Os serviços existentes combinados de diversas formas diferentes reduzem o tempo para atender as novas oportunidades ou alteração das regras, resultando em agilidade e reaproveitamento de serviços.

Segundo Gartner, confirmando o grau de satisfação de quem já está utilizando o SOA 69% das empresas que já adotaram este modelo de arquitetura e disseram que vão aumentar seus investimentos em projetos SOA. A pesquisa foi mais além e descobriu também que 83% das empresas que adotam SOA o usam para resolver problemas de integrações internas, mas que uma parcela significativa (46% das grandes corporações e 27% das médias) usa SOA para transformar seus negócios.

7.11 Terminologia de BPM

“Gerenciamento de processos de negócios pode significar duas coisas diferentes". Normalmente significa gerenciamento de processos de negócios (*Bussiness Process Management*), que é termo geral referente a todas as atividades de gerenciamento e melhoramento nos processos de negócios.

“BPM pode significar também modelagem de processos de negócios (*Bussiness Process Modeling*), que é conjunto de práticas ou tarefas que as empresas podem executar para descrever aspectos do processo de negócios...” [BLOOMBERG, 2006].

De acordo com as definições acima citadas essas práticas aplicadas ajudam a maximizar os resultados e a desempenho dos processos. E assim fazer que as organizações tenham melhores resultados financeiros, vantagem competitiva, redução de custos, otimização de recursos, aumento da satisfação dos clientes, pois os produtos e os serviços têm um nível melhor de qualidade.

Outro conceito importante é o de Sistema de Gerenciamento de *Workflow* (WfMS - *Workflow Management System*). O WfMS supri a automação procedural de um processo de negócio gerenciando a seqüência de atividades de trabalho e chamando os recursos humanos e/ou eletrônicos apropriados associados com os vários passos das atividades.

E *Web Services Business Process Execution Language* (BPEL) que é uma linguagem baseada em XML para a especificação formal de processos de negócio e protocolos de interação. BPEL agrega-se ao modelo de interação de *Web Services*, permitindo o suporte a transações de negócio. Ela resulta de uma iniciativa conjunta de IBM, BEA e Microsoft para desenvolver uma linguagem de processos suportada universalmente.

7.12 BPM E SOA

Existe hoje no mercado muita confusão em relação aos conceitos e termos utilizados para definir as novas possibilidades que se abrem para automação dos negócios: *Business Process Management*, *Service Oriented Architecture*. BPM e Modelagem de processos de negócio oferecem abordagens para identificar os serviços que são partes dos processos de negócios distribuídos. Trata-se de uma abordagem que combinam técnicas *top-down* e *botton-up* de desmembramento de processos em pedaços menores (ou seja, atividades, as quais são representadas como serviços) enquanto é considerado o ambiente de sistemas e os sistemas existentes.

"Sob o ponto de vista tecnológico, SOA, ou arquitetura orientada a serviços, possibilita o desacoplamento e a independência dos vários elementos de *software*, estruturais e funcionais, que uma empresa vai adquirir ou construir para suportar seu negócio, permitindo flexibilidade e agilidade muito interessantes tanto em relação a modificações nos elementos existentes quanto à troca de um ou mais deles. BPM, por sua vez, possibilita a otimização na utilização dos serviços disponibilizados via SOA, gerenciando a relação entre eles sob uma ótica de processos de negócio. Com BPM se ganha também muito em agilidade e flexibilidade, pois os BPMS (*Business Process Management System*) possibilitam a rápida alteração dos processos automatizados, bem como a automação de novos processos, contribuindo para que uma empresa alcance seus objetivos estratégicos." [BENDER, 2006].

Enfim, BPM e SOA combinados é uma opção muito interessante para que a tecnologia esteja realmente a serviço do negócio e para que os processos automatizados possam acompanhar de maneira efetiva a incrível agilidade que os negócios demandam atualmente.

7.13 Ferramentas de Modelagem de Processos de Negócio

Diferentes camadas e relacionamentos de um processo de negócio são chamados de modelo de processo de negócio. Com SOA, estes modelos evoluem para possibilitar a modelagem de processos em ferramentas diferentes. Algumas ferramentas até permitem que execute os processos modelados com mecanismos de processos.

Existem múltiplos padrões para modelagem e execução de processos de negócios, alguns ainda possuem notações padronizadas. Esses padrões permitem usar ferramentas e mecanismos diferentes. Os principais padrões de notação são BPMN, UML e EPC. Os padrões de processos de negócio mais importantes que mecanismos podem executar são BPEL E XPDL (*XML Process Definition Language*).

Atualmente uma ferramenta bastante utilizada na execução de processos de negócio é BPEL. Considerado um montador de processos de negócio que possibilita definir processos de serviços compostos ou de processos usando as tecnologias de *Web Services*. É importante lembrar que BPEL não atende os requisitos necessários para uma boa implementação de BPM. BPEL é apenas uma linguagem de marcação baseada no WSDL (Web Services Description Language) que forma uma Web *Service* compostos de várias outras Web *Services*.

7.14 SOA E ORGANIZAÇÃO

7.14.1 Gestão

Adotar SOA é mais complexo do que parece. Empresas que decidem utilizar arquiteturas orientadas a serviços precisam estar preparadas para enfrentar mudanças que vão além da tecnologia e afetam o comportamento de suas equipes de TI. Muitas corporações que estão deixando de lado as tecnologias cliente-servidor para adotar arquiteturas orientadas a serviços têm

enfrentado desafios técnicos e culturais que obrigam seus grupos de desenvolvimento de TI a fazer uma revisão geral em seus processos.

Enquanto cada grupo funcional freqüentemente faz um ótimo trabalho de gerenciamento do processo de trabalho dentro do seu território, eles são menos especialistas em coordenar o trabalho além das fronteiras do departamento - a essência da *performance* empresarial global da perspectiva do cliente. Em outras palavras, “os processos de negócio corporativos” são aqueles que atingem mais do que um departamento, grupo ou território interno da empresa e que determina a eficácia de uma empresa como um todo. [SPANY,2003].

7.15 ROI (Return on Investment)

[ASSAF,2003] define que Retorno sobre o Investimento (ROI) é a relação de dinheiro ganho ou perdido num investimento de acordo com a quantia de dinheiro investido. A quantia de dinheiro ganho ou perdido pode ser referido ao lucro, lucro/perda, ganho/perda, ou ganho/perda líquida. O dinheiro investido pode ser referido ao ativo, capital, principal, ou a base de custo do investimento.

ROI normalmente é dado como um valor percentual. Por exemplo, um investimento de $1.000 investimento que ganha $50 em lucro obviamente gera mais dinheiro que um investimento de $100 que ganha $20 em lucro.

- $50/$1,000 = 5% de ROI
- $20/$100 = 20% de ROI

Em termos matemáticos, o ROI é definido como segue:

$$ROI = \frac{V_f - V_i}{V_i} = \frac{V_f}{V_i} - 1$$

onde: Vi é o investimento inicial e Vf é o valor final.

O ROI tem as seguintes características:

- ROI > zero quando o investimento é lucrativo
- ROI < zero quando o investimento tem perda

7.15.1 Determinar o ROI do SOA

Enquanto o mundo é limite e determina a construção de uma arquitetura SOA dentro de sua empresa - talvez mesmo entre empresas - poucos tentam determinar seu Retorno em Investimento (ROI) para justificar esta implementação. Atualmente, arquitetos e pessoas de negócio devem trabalhar conjuntamente para determinar se as mudanças propostas são adequadas para um negócio. De fato, a aplicação de um SOA tem graus diferentes de ROI, dependendo do domínio de problema. O custo de implementação de um SOA diretamente deve estar relacionado aos benefícios ao negócio.

Implementamos SOA por duas razões importantes. Primeiro é a capacidade de economizar no desenvolvimento pela reutilização de serviços. Estes serviços podem ter sido construídos dentro ou fora da companhia, e quanto mais serviços forem reutilizáveis de sistema para sistema, maior o ROI do SOA. O segundo é a capacidade de mudar a infra-estrutura de TI mais rápido para se adaptar às necessidades variáveis do negócio. Isto, naturalmente, fornece uma vantagem estratégica enorme e assim permite para o negócio possa ter melhores possibilidades de sobrevivência á longo prazo. Enquanto determinar o ROI em agilidade é mais difícil de converter em termos financeiros, nós sabemos que o benefício está presente e é muito importante.

Um caso de sucesso utilizando SOA foi observado na empresa de seguros suíça “Helvetia Patria Group”, usuária veterana de SOA, obteve 201% de retorno do investimento desde que implantou SOA há seis anos. Representantes da Helvetia disseram que o projeto reduziu em 59% os custos de TI das unidades baseadas em *internet*, apesar dos desafios técnicos e culturais de SOA, o retorno pode ser substancial.

A Helvetia superou dificuldades de obter a adesão dos desenvolvedores ao utilizar um programa de gerenciamento de mudança da Hewlett-Packard. As ferramentas e os serviços da HP ajudaram os desenvolvedores a integrar 15 sistemas em uma plataforma SOA centralizada.

"Nosso modo de trabalhar hoje é realmente muito diferente porque, antes, não havia qualquer contato entre as diversas subsidiárias – elas tinham seus

próprios processos e suas próprias ferramentas de desenvolvimento. As conseqüências e o impacto foram extraordinariamente grandes. Uma implementação SOA é uma jornada, e você tem que investir muito para atingir um novo nível de agilidade" segundo Didier Beck, diretor do *eBusiness Center* da Helvetia.[COPUTERWORLD, 2006].

Os novos processos de desenvolvimento da Helvetia incluíram a centralização do gerenciamento de mudança e o uso de cronogramas de liberação de *software*. Além disso, a empresa agora proporciona a todos os novos desenvolvedores de seis a doze meses de treinamento em seu *eBusiness Center*, onde gerencia centralmente o SOA.

7.16 Segurança

7.16.1 Requisitos de Segurança

Segurança é parte fundamental de qualquer sistema, tem impacto em todos os serviços, deve ser construída no inicio e melhorada evolutivamente.

Os requisitos e técnicas de segurança utilizado por SOA são os basicamente os mesmo utilizados por sistemas distribuídos.

- **Autenticação**: Descobrir quem chama o serviço;
- **Autorização**: Verificar se uma identidade tem permissão de chamar o serviço e acessar o resultado;
- **Confidencialidade**: Assegurar que somente o solicitante do serviço possa ver os dados mesmo durante a transferência entre fornecedor e consumidor;
- **Integridade**: Dados não sejam manipulados;
- **Disponibilidade**: Inoperância inchamento do sistema (negação de Serviço);
- **Contabilização**: Monitorar as chamadas de serviço para gerenciamento, planejamento, financeiro e outros propósitos; e

- **Auditoria**: Manter e monitorar informações referentes à segurança buscando detectar ou analisar as falhas de seguranças e ataques.

Para autenticação e autorização, utiliza o ID de usuário e senha. Para o ID usuários atribuem-se papéis tais como: privilégios e permissões. Confidencialidade e integridade, os conceitos comuns como criptografia e assinaturas digitais são usados.

7.16.2 Segurança com XML e *WEB SERVICES*

Existem vários padrões específicos de segurança para XML e *Web Services*. Esses padrões não trazem inovações tecnológicas ou procedimentos de segurança, mas definem como aplicar o que já existe durante a troca de dados em arquivos XML ou *Web Services*. Assinatura XML, Criptografia XML, Gerenciamento de Chaves XML, *WS Security*, *WS Security Policy, WS Trust, WS Security Conversation, WS Federation* são exemplos.

Para a segurança na Comunicação é utilizado o **WS-Security**, trata de um protocolo de comunicação desenvolvido pela IBM em conjunto com Microsoft e *VeriSign*, que aplica segurança aos serviços, possui especificações para reforçar a integridade e a confidencialidade na troca de mensagens entre os serviços. A Confidencialidade da informação é mantida pelo XML- *Encryption,* que utiliza do esquema de chaves pública para criptografar as mensagens.

7.16.3 Alguns Conceitos de Segurança em SOA

- Introdução e alteração em conceitos de segurança à medida que o ambiente evolui;
- SOA força alta interoperabilidade, isso diminui a segurança padrão;
- SOA lida com conceitos de segurança heterogêneos de sistemas existentes;
- Processos distribuídos transferem dados utilizando de múltiplos serviços, então as soluções ponto a ponto não são suficientes para segurança fim a fim;

- Implementação de abordagem de segurança estratégica que cubra estrutura, arquitetura e aplicações. (segurança como serviço);
- XML e *Web Services* podem introduzir problema de segurança especiais, como o perigo de bombas XML e injeções *XPath*; e
- Ter segurança em mente.

7.17 Desafios SOA

Atualmente a Tecnologia da Informação tem sofrido várias modificações, impulsionada pela necessidade de melhoria no relacionamento que existe entre as áreas que suportam tecnologicamente os negócios e os negócios propriamente ditos. Seus gestores estão se dando conta da necessidade e da conveniência do enfoque nos serviços, como um fim, sendo vitais para os negócios em um futuro que já se tornava atualidade. SOA trouxe à tona a necessidade de fortalecer o enfoque no cliente e tornar a gestão de serviços como uma atividade produtiva, que gere valor à empresa.

Como tal, essa atividade é fortemente dependente das pessoas e a caminhada rumo a SOA é árdua, exigindo um forte investimento na evolução organizacional, na conscientização da necessidade de evoluir e mudar, no estabelecimento de um gerenciamento de pessoas eficaz, orientada a conhecer a potencialidade, objetivos e desejos das pessoas em detrimento dos objetivos da organização, orientando à gerência de serviços em acordo com o desempenho individual. Tornar os desafios da organização aderentes ás pessoas é um dos maiores desafios da Gestão da mudança organizacional. Dessa forma SOA possibilita a criação de novos aplicativos com maior coerência, rapidez e diminuição nos custos, tudo isso com excelente aproveitamento do legado.

8 Estudo de Caso

8.1 Proposta de Solução

O Centro Educacional Tangram é uma instituição de ensino infantil, localizada em Brasília. Hoje a instituição possui aproximadamente cem alunos e deseja ampliar a sua estrutura para a educação fundamental.

O principal objetivo do Centro Educacional Tangram é promover uma educação de qualidade, buscando sempre a transparência com os pais e responsáveis, porém a instituição enfrenta barreiras por não contar com um sistema para automatizar os processos. Atualmente, todo o processo é manual o que gera uma baixa produtividade e falta de informações confiáveis para as tomadas de decisões. O enfoque dessa pesquisa será avaliar a arquitetura orientada a serviços para o desenvolvimento de uma solução que tem como objetivo promover o aumento da eficiência nos processos de negócio. O SIGED (Sistema de Gestão Educacional) facilitará a integração entre os departamentos da escola, além de ser um canal de comunicação entre a escola, pais e futuramente com a comunidade.

Para isso o foco desse projeto, será desenvolver uma pesquisa cuja proposta é a integração com os processos de negócio, visando à flexibilidade de manutenção para acompanhar as constantes mudanças organizacionais.

A ferramenta deverá funcionar como um gerenciador de processos, sendo que os papéis deverão ser substituídos por formulários eletrônicos o que resultando na diminuição do tempo de resposta, armazenamento dos dados, redução de custo e conseqüentemente aumentando a eficiência e eficácia dos gestores educacionais, visando garantir a melhoria dos resultados financeiros.

O processo de cadastro e efetivação de matrícula que agora é realizado manualmente será automatizado para que se possa facilitar tanto o processo de matrícula como de renovação, evitando assim desperdício de tempo e falhas na confirmação dos dados. Antes de efetivar a matrícula será realizada uma pesquisa no sistema para verificar a disponibilidade de vagas em uma determinada turma/turno, senão houver possibilidade de efetivar a matrícula será feito o cadastro reserva do aluno.

Outra preocupação dos gestores educacionais e criar uma massa de dados para identificar as necessidades de investimentos, para isso o sistema contará com um inventário compartilhado, para que demonstre as necessidades da escola.

Além disso, haverá maior facilidade na hora de organizar as turmas, pois antes da alocação do professor o sistema identificará em qual turno o mesmo estará disponível para ministrar uma determinada disciplina, e de acordo com a quantidade de alunos identificar a necessidade de abertura de novas turmas ou contratação de novos docentes.

Outras funcionalidades que os gestores educacionais desejam automatizar são: registro de anamnese, emissão de diário de classe, registro de conteúdo programático, elaboração do livro de ata. Todas com os respectivos controles de segurança de acesso por nível de usuário.

Porém, o principal diferencial do sistema deverá ser a facilidade para manter o canal de comunicação entre os pais e a escola. Por meio do SIGED será possível o registro do comportamento do aluno, ausências, notas ou qualquer comunicação dos pais para escola.

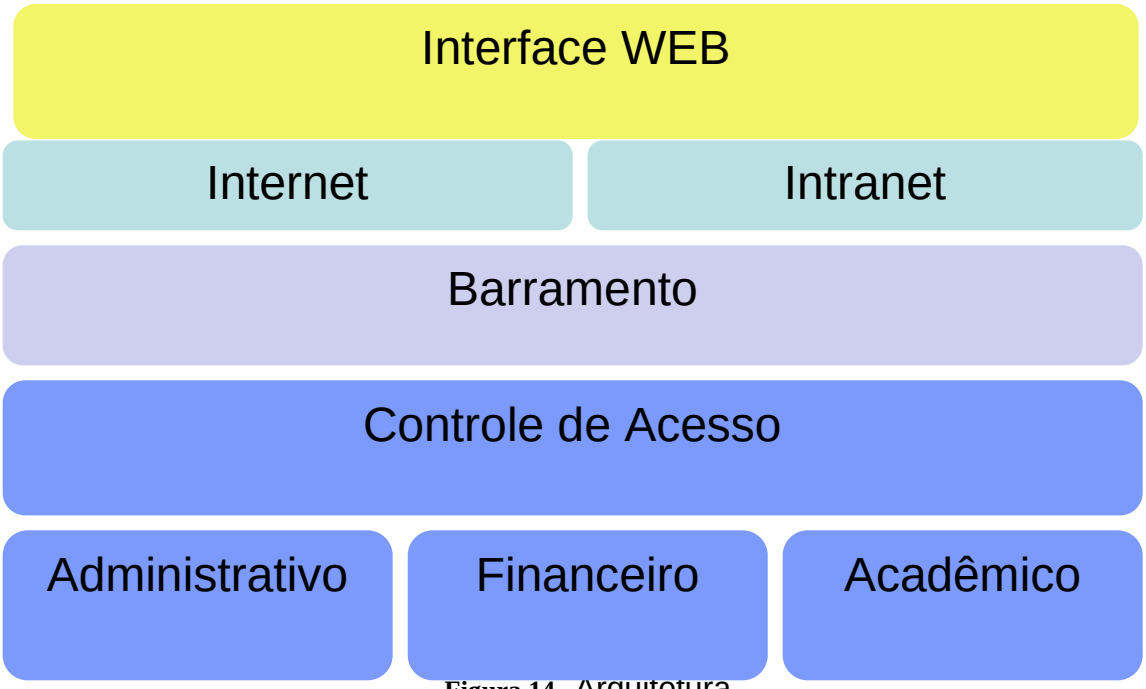


Figura 14- Arquitetura

8.2 A Utilização do SIGED no TANGRAN

O TANGRAN possui seus processos de controle acadêmico que são fundamentais para o funcionamento da Instituição. Seu controle era realizado de maneira manual o que gerava baixa produtividade.

A instituição, que durante sua existência educou diversas crianças no ensino infantil e fundamental, acumulou uma grande quantidade de documentos referentes aos registros dos alunos, dos funcionários, livros de ata, controle de inventário, que acabou gerando um problema que agora necessita ser administrado, a grande quantidade de documentação que necessita ser automatizada.

Estes registros são vitais para o TANGRAN, que tem agora a responsabilidade de zelar por estas informações que fazem parte de um acervo extremamente valioso, bem como à documentação histórica dos seus alunos, que também se encontra armazenada de forma perigosa, deixando a Instituição em uma situação de vulnerabilidade diante de desastres, problemas de vazamento nas instalações hidráulicas, uso inadequado e exposição a insetos e pragas.

O SIGED vem para auxiliar o TANGRAN a controlar seus processos, para que assim possa garantir a melhoria do negócio e seu gerenciamento, trazendo assim maior transparência e agilidade.

8.3 Requisitos mínimos que o sistema deve atender

Matricular Aluno: O sistema deverá permitir ao atendente que realize a matrícula de um novo aluno. Ao efetuar o *logon* no sistema, que será concedido utilizando uma senha individual, o sistema apresentará um Menu de opções, entre elas efetuar Matrícula. Ao finalizar a matrícula, o sistema irá gerar o registro do aluno, contrato de prestação de serviços, e a lista de material.

Cadastrar Funcionário: O sistema deverá permitir ao funcionário que realize ao cadastro de um novo funcionário. Ao efetuar o *logon* no sistema, que será concedido utilizando uma senha individual, o sistema apresentará um Menu de opções, entre elas cadastrar funcionário, ao preencher os dados solicitados, o sistema irá gerar o número de matrícula e a senha individual do novo funcionário.

Processar Disciplinas: O funcionário deverá ser capaz de processar disciplinas no sistema. Ao efetuar o logon no sistema, que será concedido utilizando uma senha individual, o sistema apresentará um Menu de opções, entre elas Processar Disciplinas, na qual o mesmo passará por as etapas disponíveis, e ao finalizar as mesmas será capaz de processar nova disciplina.

Registrar Ocorrência: O professor deverá ser capaz de registrar ocorrência do aluno. Ao efetuar o logon no sistema, que será concedido utilizando uma senha individual, o sistema apresentará um Menu de opções, entre elas registrar ocorrência, o professor irá registrar a ocorrência do aluno, de acordo com o comportamento do aluno, e o sistema enviará o email para o pais ou responsável do aluno.

8.4 Modelando a Solução

8.4.1 Diagrama de Classes de Negócio

Para facilitar o entendimento e entender bem o funcionamento de um sistema acadêmico, é extremamente necessário o conhecimento sobre as estâncias que compõem o modelo. O modelo a seguir representa as classes que deverão ser implementadas.

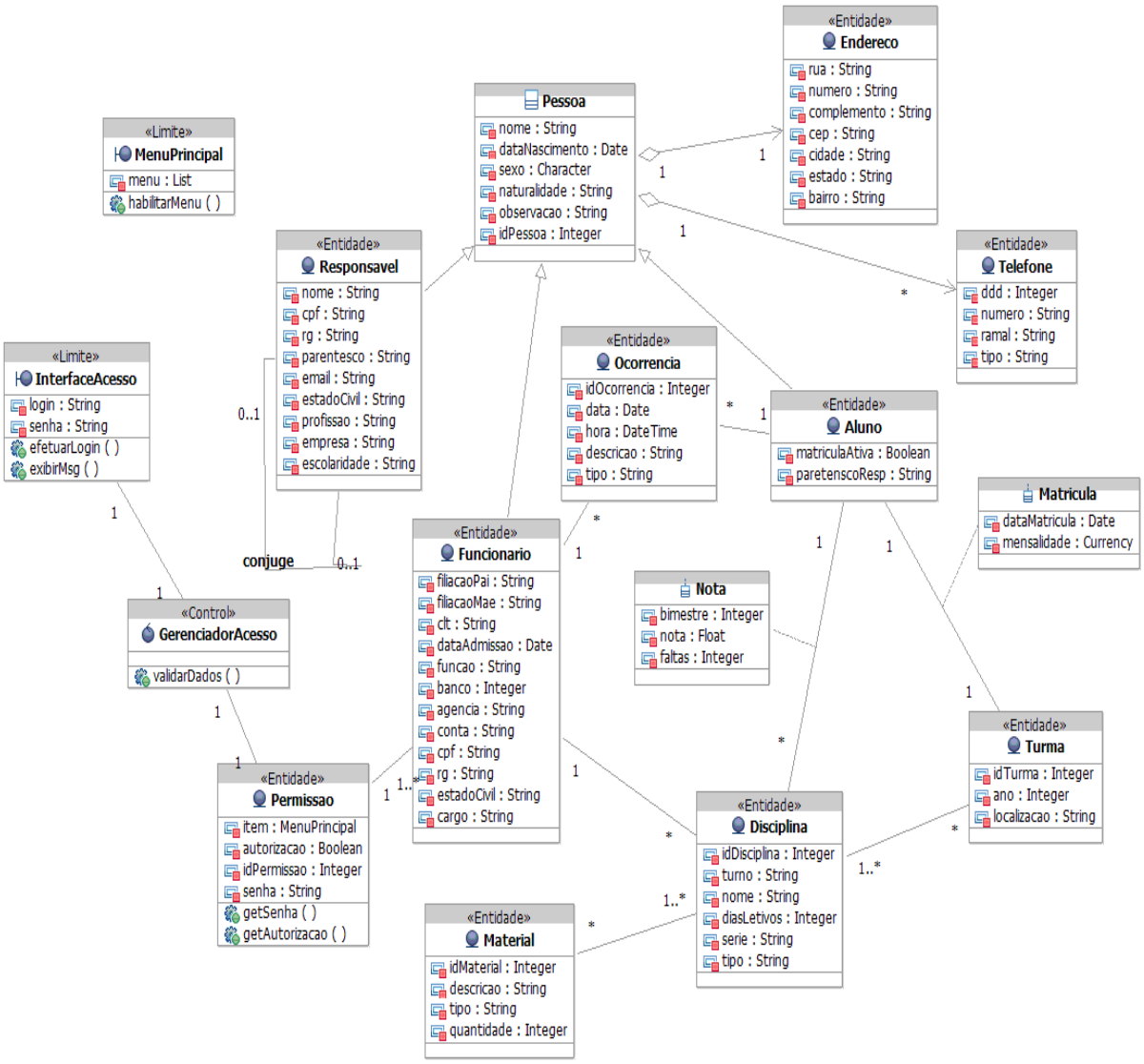


Figura 15 - Diagrama de Classes – Visão sistêmica do negócio

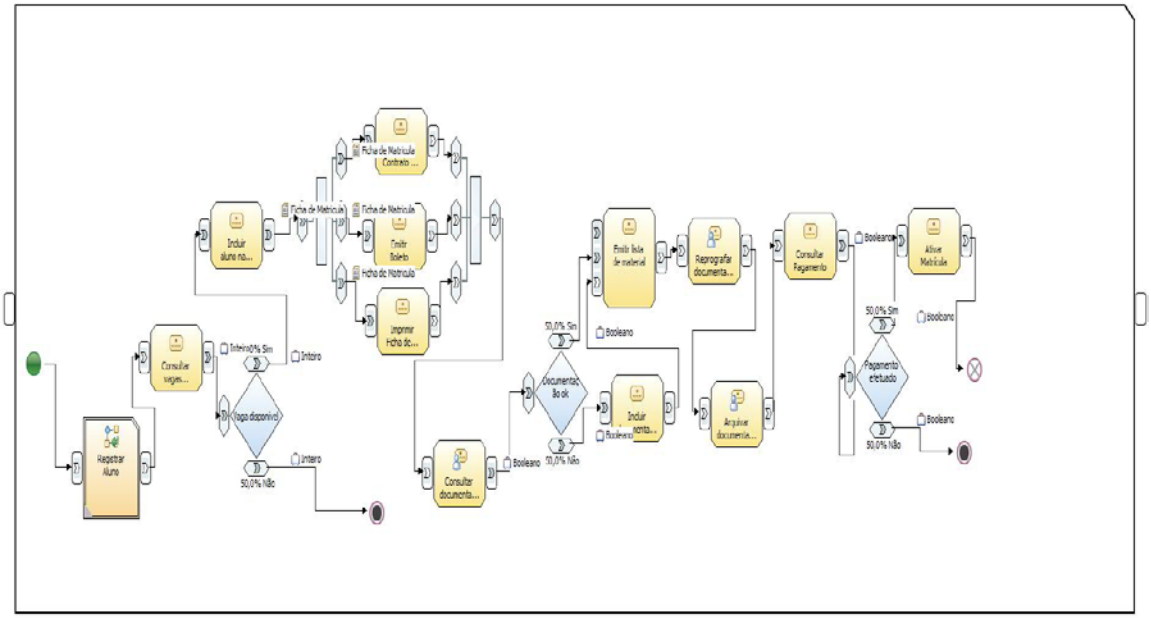


Figura 16– Diagrama de Processo de Negócio

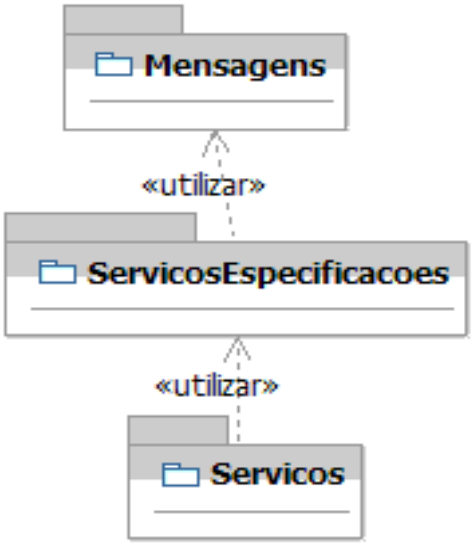
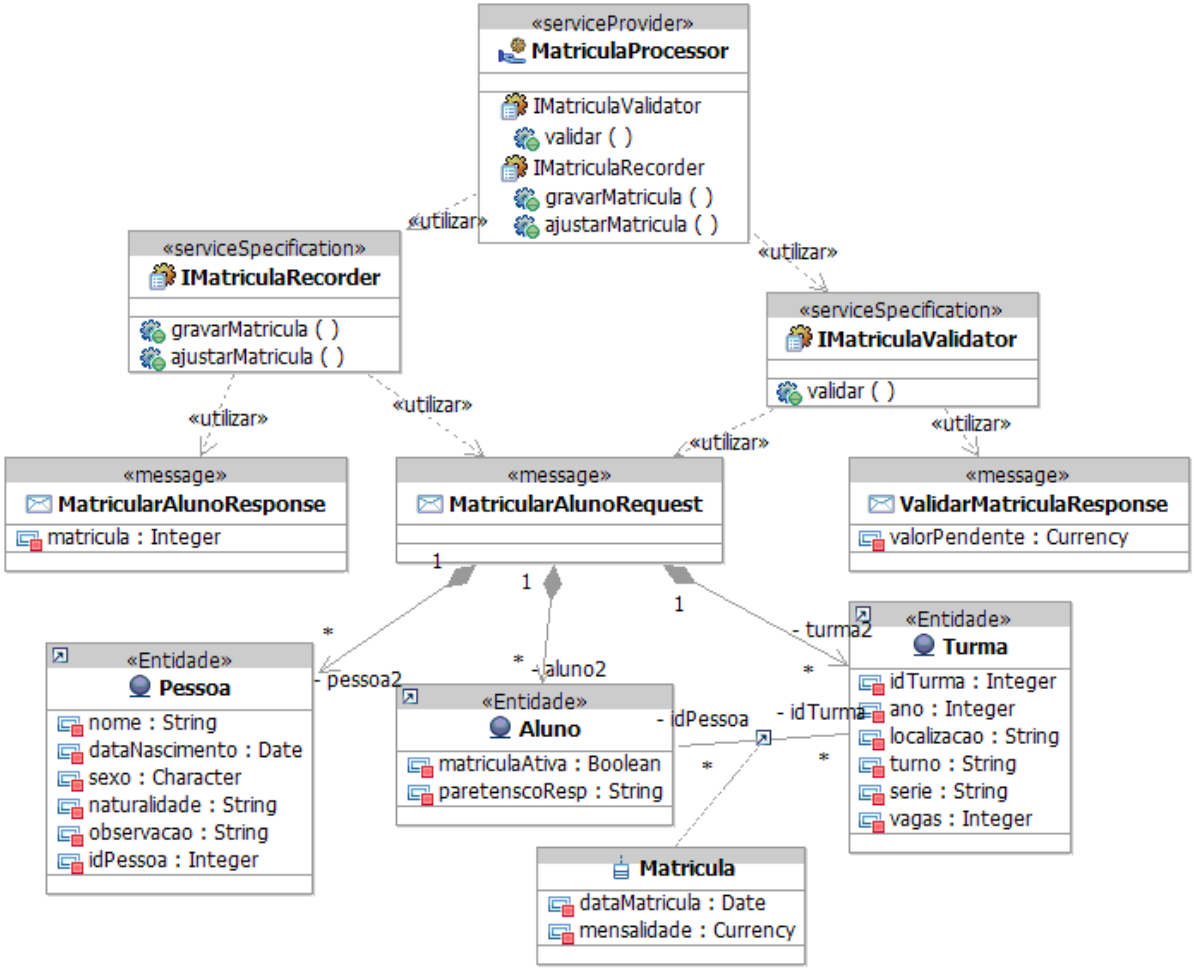


Figura 17– Serviços (Pacotes)



8.4.2 Especificações de Caso de Uso

Tabela 1 -Caso de Uso 01 – Efetuar Login

Número	Requisito Funcional

RF - 01	Efetuar Login	Essencial (x)	Oculto ( )
Descrição			
O funcionário solicita a visualização do menu de opções do sistema.			
Requisitos Não Funcionais			

Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente funcionário com seus dados validados poderá, efetuar login no sistema.	Controle de Acesso	( )	(x)
RNF- 02:	Se o acesso for negado ao funcionário, o mesmo deverá solicitar permissões.	Controle de Processos	(x)	( )
RNF–03:	Acesso ao sistema somente por estação protegida por firewall.	Segurança	( )	(x)

Atores
Funcionário, Atendente ,Diretor e Coordenador.
Pré-Condições
• O funcionário visualizar o menu de opções do sistema.
Fluxo Principal
P1. O funcionário abre o menu de opções do sistema; P2. O sistema apresenta na tela a lista de opções; P3. O funcionário escolhe a opção efetuar login; P5. O funcionário digita seu login; P6. O funcionário digita sua senha; P7. O sistema valida os dados; P8. O sistema retorna para P2;
Fluxo Alternativo
A1. Se ocorrer erro do funcionário ao digitar seus dados de acesso, o sistema retornará ao menu principal.
Pós-Condições
Após validação dos dados o funcionário está logado no sistema.

Tabela 2- Caso de Uso 02 – Matricular Aluno

Número	**Requisito Funcional**		
RF - 02	Matricular Aluno	Essencial (x)	Oculto ( )

Descrição				
O atendente executa o *logon* no sistema e faz a opção por matricular o aluno no sistema.				
Requisitos Não Funcionais				
Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente atendente com logon cadastrado no sistema poderá matricular aluno.	Controle de Acesso	( )	(x)
RNF- 02:	O atendente deverá verificar se o responsável já está cadastrado no sistema.	Controle de Processos	(x)	( )
RNF- 03:	Somente será efetuada a matrícula se houver cadastro de responsável no sistema	Controle de Processos	( )	(x)
RNF- 04:	O aluno só terá sua matrícula ativa após o pagamento da primeira mensalidade.	Controle de Processos	( )	(x)
RNF - 05	Acesso ao sistema somente por estação protegida por firewall.	Segurança	( )	(x)
Atores				
Atendente, Responsável e o Aluno.				
Pré-Condições				
• O atendente tem que estar logado no sistema. • As turmas devem estar disponíveis no sistema				
Fluxo Principal				
P1. No menu o atendente escolhe a opção Registrar Aluno; P2. O atendente registra os dados do aluno; P3. O atendente registra os dados da anamnese; P4. O sistema retorna para P2; P5. O atendente escolhe a opção Efetuar Matrícula; P6. O atendente resgata o registro do aluno; P7. O atendente solicita a lista de turmas disponíveis; P8. O sistema apresenta a lista de turmas disponíveis; P9. O atendente inclui o aluno na turma disponível de acordo com os pré-requisitos da série; P10. O sistema retorna para P2; P11. O atendente escolhe a opção Ficha de Matrícula; P12. O atendente resgata os dados da matrícula; P13. O atendente solicita a impressão do contrato de prestação de serviço; P14. O atendente imprime o boleto para pagamento; P15. O sistema imprime lista de material de acordo com a série; P16. O atendente ativa a matrícula; P17. O sistema retorna ao P2.				
Fluxo Alternativo				
A1. O aluno só poderá ser matriculado mediante a disponibilidade de vagas na turma; A2. Se o responsável esquecer alguma documentação necessária para que a matrícula do aluno seja efetuada, o mesmo poderá trazer posteriormente, mas constará observação no sistema de que faltam documentos; A3. Somente após a confirmação do pagamento da mensalidade, o aluno estará com sua matrícula ativa.				
Pós-Condições				
Os dados cadastrais do aluno serão cadastrados no sistema.				

Tabela 3 - Caso de Uso 03 – Cadastrar Responsável

Número	**Requisito Funcional**		
RF - 03	Cadastrar Responsável	Essencial (x)	Oculto ( )

Descrição				
O atendente executa o *logon* no sistema e faz a opção por cadastrar responsável no sistema.				
Requisitos Não Funcionais				
Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente atendente com logon cadastrado no sistema poderá cadastrar responsável.	Controle de Acesso	( )	(x)
RNF- 02:	O atendente deverá verificar se o responsável já está cadastrado no sistema.	Controle de Processos	(x)	( )
RNF- 03:	Somente será efetuada a matrícula se houver cadastro de responsável no sistema	Controle de Processos	( )	(x)
RNF–04:	Acesso ao sistema somente por estação protegida por firewall.	Segurança	( )	(x)
Atores				
Atendente, Responsável				
Pré-Condições				
• O atendente tem que estar logado no sistema.				
Fluxo Principal				
P1. No menu o atendente escolhe a opção Cadastrar Responsável; P2. O atendente registra os dados do responsável; P3. O atendente verifica se os dados estão cadastrados corretamente; P4. O sistema mostra os dados cadastrados; P5. O sistema retorna ao P2.				
Fluxo Alternativo				
A1. Se o responsável esquecer alguma documentação necessária para que seu cadastro seja efetuado, o mesmo poderá trazer posteriormente, mas constará observação no sistema de que faltam documentos; A2. Se o responsável não tiver email pessoal, deverá constar observação no sistema, para que o mesmo possa cadastrá-lo posteriormente;				
Pós-Condições				
Os dados cadastrais do responsável serão cadastrados no sistema.				

Tabela 4 - Caso de Uso 04 – Cadastrar Funcionário

Número	**Requisito Funcional**		
RF - 04	Cadastrar Funcionário	Essencial (x)	Oculto ( )

Descrição				
O funcionário executa o *logon* no sistema e faz a opção por cadastrar funcionário.				
Requisitos Não Funcionais				
Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente funcionário com logon cadastrado no sistema poderá cadastrar outro funcionário.	Controle de Acesso	( )	(x)
RNF- 02:	O atendente está sob posse dos documentos exigidos para admissão do novo funcionário, e arquivar os documentos numa pasta.	Controle de Processos	( )	(x)
RNF -03:	Acesso ao sistema somente por estação com firewall.	Disponibilidade	(x)	( )
Atendente e Funcionário				
Pré-Condições				
• O atendente tem que estar logado no sistema.				
Fluxo Principal				
P1. No menu o atendente escolhe a opção cadastrar funcionário; P2. O atendente inclui os dados do funcionário; P3. O sistema gera número de matrícula para o funcionário; P4. O sistema retorna P4; P5. O atendente seleciona a opção controle de acesso; P6. O atendente cadastra os acessos permitidos ao funcionário; P7. O sistema gera senha para o funcionário; P8. O atendente solicita a emissão do contrato de trabalho; P9. O sistema imprime contrato de trabalho; P10. O sistema retorna ao P2.				
Fluxo Alternativo				
A1. Na falta da documentação exigida para admissão do novo funcionário, o mesmo poderá trazer posteriormente, mas constará observação no sistema de que faltam documentos.				
Pós-Condições				
Após confirmação da contratação os dados do funcionário serão guardados no sistema.				

Tabela 5 - Caso de Uso 05 – Cadastrar Turma

Número	**Requisito Funcional**		
RF - 05	Cadastrar Turma	Essencial (x)	Oculto ( )
Descrição			
O coordenador executa o *logon* no sistema e faz a opção por cadastrar turma no sistema.			
Requisitos Não Funcionais			

Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente coordenador com logon cadastrado no sistema poderá cadastrar turmas.	Controle de Acesso	( )	(x)
RNF- 02:	As turmas serão identificadas por códigos compostos com a série e turno.	Controle de Processos	( )	(x)
RNF - 03	Acesso apenas por estação protegida por firewall	Segurança	( )	(x)
RNF -04	O direito de acesso será fornecido apenas pelo Diretor	Disponibilidade	( )	(X)

Atores

Coodenador

Pré-Condições

- O coordenador tem que estar logado no sistema.
- As disciplinas devem estar incluídas no sistema

Fluxo Principal

P1. No menu o coordenador escolhe a opção Cadastrar Turma;
P2. O coordenador solicita a lista com disciplinas disponíveis;
P3. O sistema retorna a lista de disciplinas disponíveis;
P4. O coordenador incluirá o código da turma, série e a localização;
P5. O coordenador incluirá as disciplinas que fazem parte da turma;
P6. O coordenador salvará o registro efetuado;
P7. O sistema retorna ao P2.

Fluxo Alternativo

A1. Caso não haja docentes disponíveis o sistema encerrará o processo;
A2. O sistema retorna uma mensagem de ambigüidade caso a turma já exista no sistema.

Pós-Condições

Após as turmas serem processadas será cadastrada no sistema.

Tabela 6 - Caso de Uso 06 – Cadastrar Disciplina

Número	**Requisito Funcional**		
RF - 06	Cadastrar Disciplina	Essencial (x)	Oculto ( )
Descrição			
O coordenador executa o *logon* no sistema e faz a opção por cadastrar disciplina no sistema.			
Requisitos Não Funcionais			

Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente coordenador com logon cadastrado no sistema poderá processar disciplinas.	Controle de Acesso	( )	(x)
RNF- 02:	As disciplinas que não compõem a grade obrigatória deverão ser cadastradas como disciplinas extra-curriculares	Controle de Processos	( )	(x)
RNF- 03:	As disciplinas devem ser aprovadas pela Secretaria de Educação do DF	Controle de Processos	( )	(x)
RNF -04:	Acesso ao sistema somente por estação com firewall.	Segurança	( )	(x)

Atores

Coordenador

Pré-Condições

- O coordenador tem que estar logado no sistema.
- Os professores deverão estar cadastrados no sistema.

Fluxo Principal

P1. No menu o coordenador escolhe a opção Cadastrar Disciplina;
P2. O coordenador insere as disciplinas que compõem uma turma para cada série e turno;
P3. O coordenador identifica o tipo de disciplina de acordo com os requisitos do governo;
P4. O coordenador solicitará a lista de professores disponíveis;
P5. O sistema retornará a lista de professores;
P6. O coordenador incluirá os professores disponíveis na turma;
P7. O sistema retorna ao P2;
P8. O coordenador escolhe a opção Lista de Material;
P9. O sistema retorna a tela para a inclusão da lista de material;
P10. O coordenador cadastrará a lista de material de acordo com a disciplina e série a ser cadastrada;
P11. O sistema retorna ao P2.

Fluxo Alternativo

A1. Se o funcionário não tiver com a lista de material disponível, deverá providenciar para que a mesma seja cadastrada no sistema;
A2. Caso haja necessidade de cadastro de disciplina extra-curricular, a mesma deverá ser solicitada com antecedência.

Pós-Condições

Após as disciplinas serem processadas serão cadastradas no sistema.

Tabela 7 - Caso de Uso 07 – Emitir Boletim

Número	**Requisito Funcional**		
RF - 07	Emitir Boletim	Essencial (x)	Oculto ( )
Descrição			
O funcionário executa o *logon* no sistema e faz a opção por emitir boletim.			
Requisitos Não Funcionais			

Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente funcionário com logon cadastrado no sistema poderá emitir boletim.	Controle de Acesso	( )	(x)
RNF- 02:	O funcionário deverá está com posse das notas e faltas dos alunos para que sejam cadastradas no sistema.	Controle de Processos	( )	(x)
RNF -03:	Acesso ao sistema somente por estação com firewall.	Disponibilidade	(x)	( )

Atores

Funcionário, Professores e Alunos.

Pré-Condições

- O funcionário tem que estar logado no sistema.

Fluxo Principal

P1. No menu o funcionário escolhe a opção Emitir Boletim;
P2. O funcionário irá checar as turmas disponíveis;
P3. O sistema mostrará as turmas disponíveis;
P4. O funcionário irá checar as disciplinas disponíveis;
P5. O sistema mostrará as disciplinas disponíveis;
P6. O funcionário irá escolher a turma e a disciplina para que sejam cadastradas as notas e faltas;
P7. O sistema disponibilizará a turma e a disciplina solicitada;
P8. O funcionário irá escolher o bimestre no qual as notas e faltas serão cadastradas;
P9. O sistema disponibilizará a turma e a disciplina de acordo com o bimestre;
P10. O funcionário cadastrará as notas e faltas disponibilizadas pelos professores;
P11. O funcionário solicitará a impressão dos boletins da turma;
P12. O sistema irá imprimir os boletins;
P13. O sistema retorna ao P2.

Fluxo Alternativo

A1. Se houver disciplinas com notas e faltas a serem cadastradas, o funcionário deverá procurar o professor para solicitar as mesmas;

Pós-Condições

Após serem processadas as notas do aluno deverão ser guardadas no sistema.

Tabela 8 - Caso de Uso 08 – Registrar Ata de Reunião

Número	**Requisito Funcional**		
RF - 08	Registrar Ata de Reunião	Essencial (x)	Oculto ( )
Descrição			
O funcionário executa o *logon* no sistema e faz a opção por registrar ata de reunião.			
Requisitos Não Funcionais			

Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente funcionário com logon cadastrado no sistema poderá registrar ata de reunião.	Controle de Acesso	( )	(x)
RNF- 02:	O funcionário deverá está com posse dos dados da reunião a serem registrados no livro de ata.	Controle de Processos	( )	(x)
RNF -03:	Acesso ao sistema somente por estação com firewall.	Disponibilidade	(x)	( )

Atores
Funcionário e Responsável
Pré-Condições
• O funcionário tem que estar logado no sistema.
Fluxo Principal
P1. No menu o funcionário escolhe a opção registrar ata de reunião; P2. O funcionário irá descrever a ata de reunião no sistema; P3. O sistema irá salvar as informações cadastradas; P4. O funcionário irá cadastrar no sistema os participantes da reunião; P5. O sistema irá salvar os nomes dos participantes da reunião; P6. O sistema imprime a ata de reunião; P7. O sistema retorna ao P2.
Fluxo Alternativo
A1. Na falta da ata o funcionário deverá procurar o responsável pela reunião.
Pós-Condições
Após ser processado o livro de ata as informações deverão ser guardadas no sistema.

Tabela 9 - Caso de Uso 09 – Registrar Ocorrência

Número	Requisito Funcional		
RF - 09	Registrar Ocorrência	Essencial (x)	Oculto ( )
Descrição			
O professor executa o *logon* no sistema e faz a opção por registrar ocorrência.			
Requisitos Não Funcionais			

Nome	Condição/Restrição	Categoria	Desejável	Permanente
RNF–01:	Somente professor com logon cadastrado no sistema poderá registrar ocorrência.	Controle de Acesso	( )	(x)
RNF- 02:	O professor está sob posse do nome do aluno para que seja registrada a ocorrência.	Controle de Processos	( )	(x)
RNF-03:	Os pais ou responsáveis ao matricular o aluno deverão cadastrar seu email pessoal, para que seja enviada a ocorrência sobre o comportamento de seu filho.	Controle de Processos	( )	(x)
RNF-04:	Os pais ou responsáveis poderão responder ao email enviado pelo professor.	Controle de Processos	( )	(x)
RNF -05:	Acesso ao sistema somente por estação com firewall.	Disponibilidade	(x)	( )

Atores

Professor, Aluno e Responsável

Pré-Condições

- O professor tem que estar logado no sistema.

Fluxo Principal

P1. No menu o professor escolhe a opção registrar ocorrência;
P2. O professor escolhe a disciplina, série, turma e turno;
P3. O sistema mostrará as informações solicitadas;
P4. O professor escolhe o nome do aluno para que seja registrada sua ocorrência;
P5. O professor cadastrará a ocorrência do aluno;
P6. O sistema salva a ocorrência registrada;
P7. O sistema envia email para os pais ou responsável do aluno;
P8. O sistema retorna ao P2.

Fluxo Alternativo

A1. Na falta do email dos pais ou responsável cadastrado no sistema, o professor deverá solicitar ao atendente que entre em contato com os mesmos;
A2. Na falta de opções de categorias para registrar ocorrência do aluno, haverá opção outros, na qual o professor poderá descriminar outra conduta.

Pós-Condições

Após ser registrado o comportamento deverá ser enviado email para os pais ou responsável.

Tabela 10 - Caso de Uso 10 – Controlar Inventário

Número	**Requisito Funcional**		
RF - 10	Controlar Inventário	Essencial (x)	Oculto ( )
Descrição			
O funcionário executa o *logon* no sistema e faz a opção por controlar inventário.			
Requisitos Não Funcionais			
Nome	**Condição/Restrição**	**Categoria**	**Desejável**	**Permanente**
RNF–01:	Somente funcionário com logon cadastrado no sistema poderá controlar inventário.	Controle de Acesso	( )	(x)
RNF- 02:	O funcionário está sob posse do material a ser cadastrado no sistema.	Controle de Processos	( )	(x)
RNF -03:	Acesso ao sistema somente por estação com firewall.	Disponibilidade	(x)	( )
Atores			
Funcionário			
Pré-Condições			
• O funcionário tem que estar logado no sistema.			
Fluxo Principal			
P1. No menu o funcionário escolhe a opção controlar inventário; P2. O funcionário cadastra o material; P3. O sistema cadastrará o material; P4. O funcionário imprime a lista de materiais disponíveis; P5. O sistema retorna ao P2.			
Fluxo Alternativo			
A1. Na falta de material disponível o funcionário deverá solicitar a compra do mesmo;			
Pós-Condições			
Após ser registrado o material no inventário os dados serão cadastrados no sistema.			

8.5 Diagramas de Caso de Uso

8.5.1 Diagrama de Caso de Uso – Efetuar Login

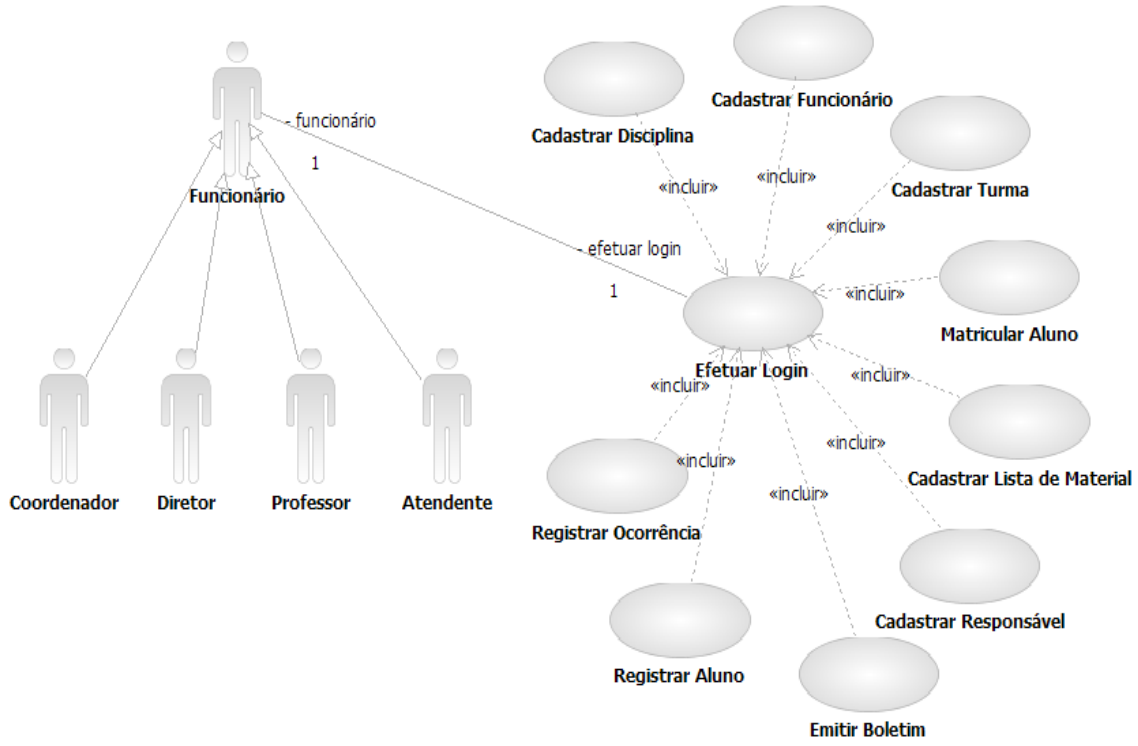


Figura 19 - Diagrama de Caso de Uso – Efetuar Login

8.5.2 Diagrama de Caso de Uso – Cadastro

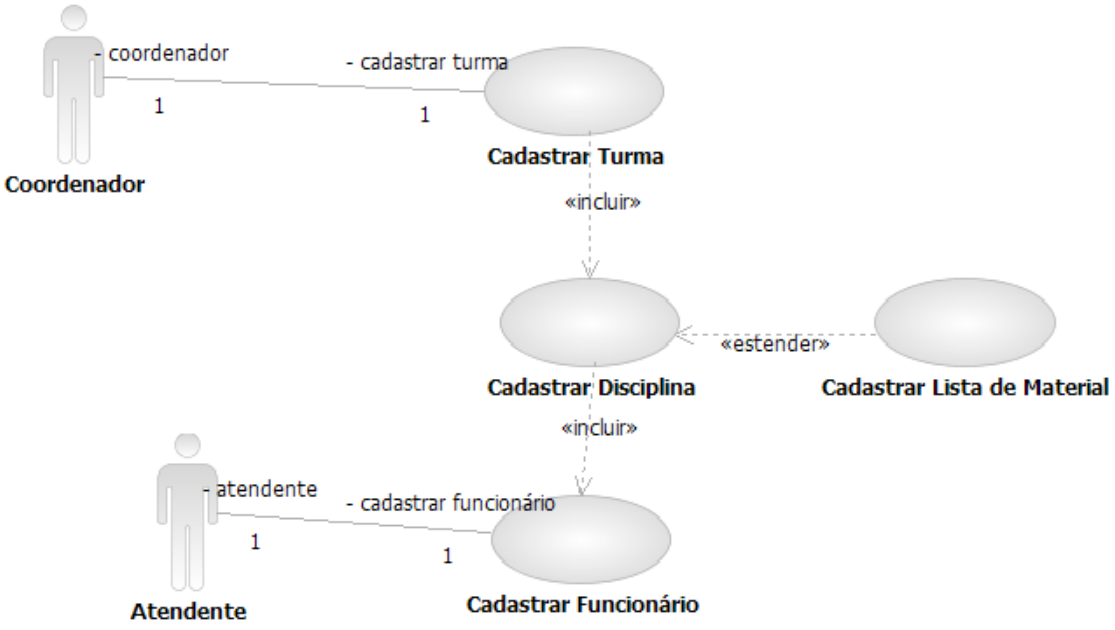


Figura 20 - Diagrama de Caso de Uso – Cadastro

8.5.3 Diagrama de Caso de Uso – Cadastrar Responsável

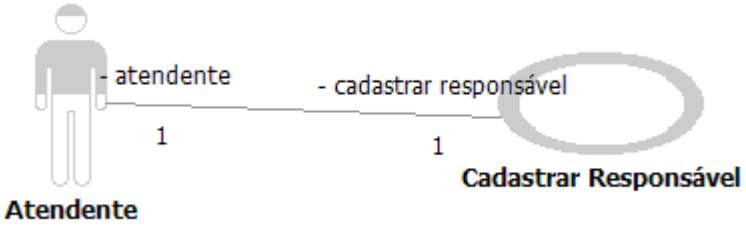


Figura 21 - Diagrama de Caso de Uso – Cadastrar Responsável

8.5.4 Diagrama de Caso de Uso – Matricular Aluno

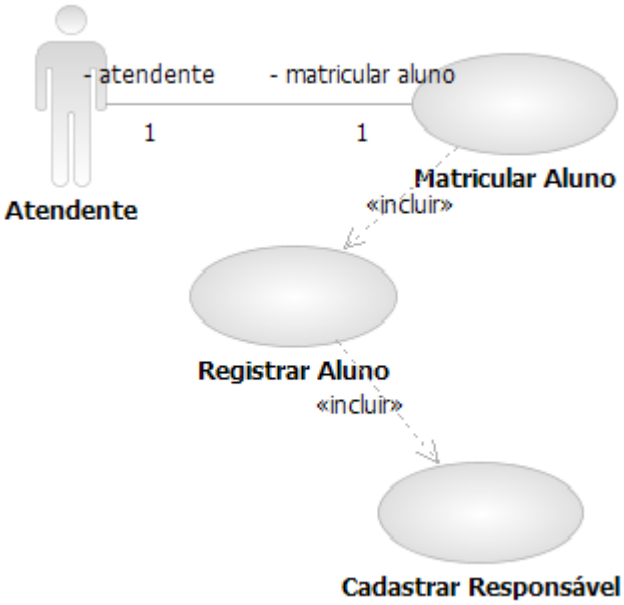


Figura 22 - Diagrama de Caso de Uso – Matricular Aluno

8.5.5 Diagrama de Caso de Uso – Cadastrar Disciplina

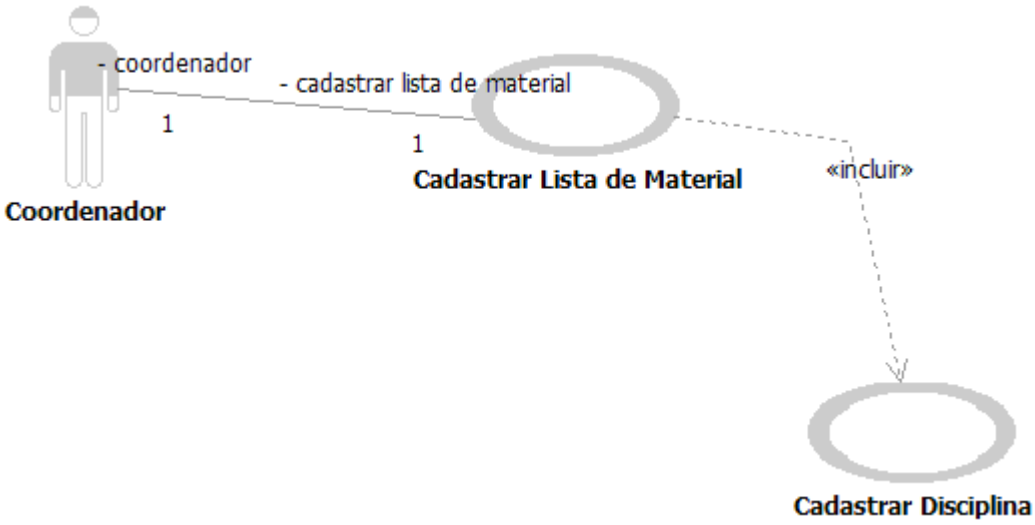


Figura 23 - Diagrama de Caso de Uso – Cadastrar Disciplina

8.6 Documentos Gerados

No Tangram existem diversos tipos de documentos, não somente os processos provenientes da matricula dos alunos, que merecem ser controlados pela escola. Procurou-se gerenciar todos os processos que são necessários para o negócio.

8.6.1 Documentos administrativos

São considerados administrativos aqueles documentos inalteráveis, cuja função é a regulamentação da Escola, são os documentos voltados à área burocrática e que por isto mesmo devem ser de fácil localização e guardados com a máxima segurança possível. São eles:

- Ficha de Matrícula;
- Cadastro de Anamnese;
- Contrato de prestação de serviços; e
- Registro de Funcionário.

8.6.2 Documentos técnicos

Os documentos técnicos são utilizados como auxílio na execução das tarefas rotineiras, cujo conteúdo pode sofrer alterações esporadicamente.

- Controlar Inventário;
- Emitir Boletim; e
- Registrar Ocorrência.

8.6.3 Levantamento das necessidades

A necessidade de melhorar o gerenciamento escolar levou Tangram a buscar no mercado de tecnologia, uma solução em termos de gestão escolar que

atendesse as necessidades básicas e que oferecesse algumas vantagens em relação aos controles que estavam sendo usados até então.

Alguns benefícios que o SIGED pode oferecer:

- Redução do consumo de papel;
- Padronização de software de gerenciamento;
- Rapidez na recuperação de uma informação; e
- Controle automatizado do inventário;

8.6.4 Resistência do usuário

Sabe-se que toda tecnologia está sujeita e enfrentar resistências que devem ser superadas ao longo da implantação da ferramenta, isto porque, logo os funcionários perceberão que na verdade a ferramenta está sendo implantada com a intenção de melhorar o processo e não de controlar o rendimento do funcionário.

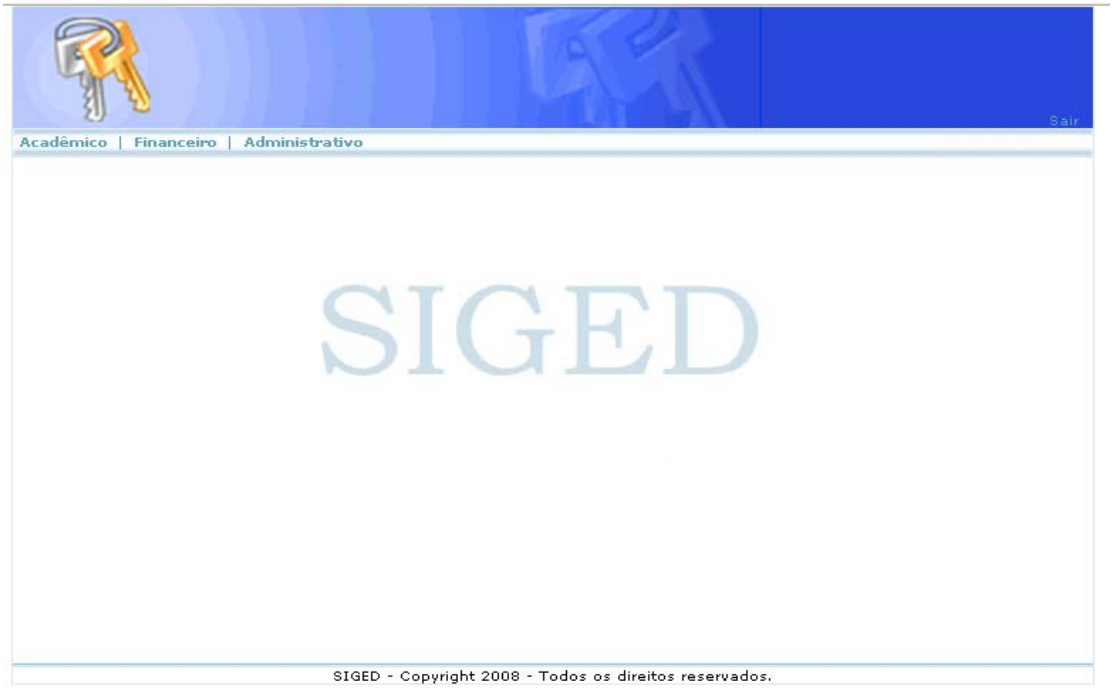
O estudo realizado entre os funcionários não apontou um número de resistência que fosse capaz de comprometer o funcionamento ou mesmo a aquisição do SIGED, portanto, pode-se afirmar que quanto a isto, o Tangram não terá nenhum problema.

9 TELAS SIGED

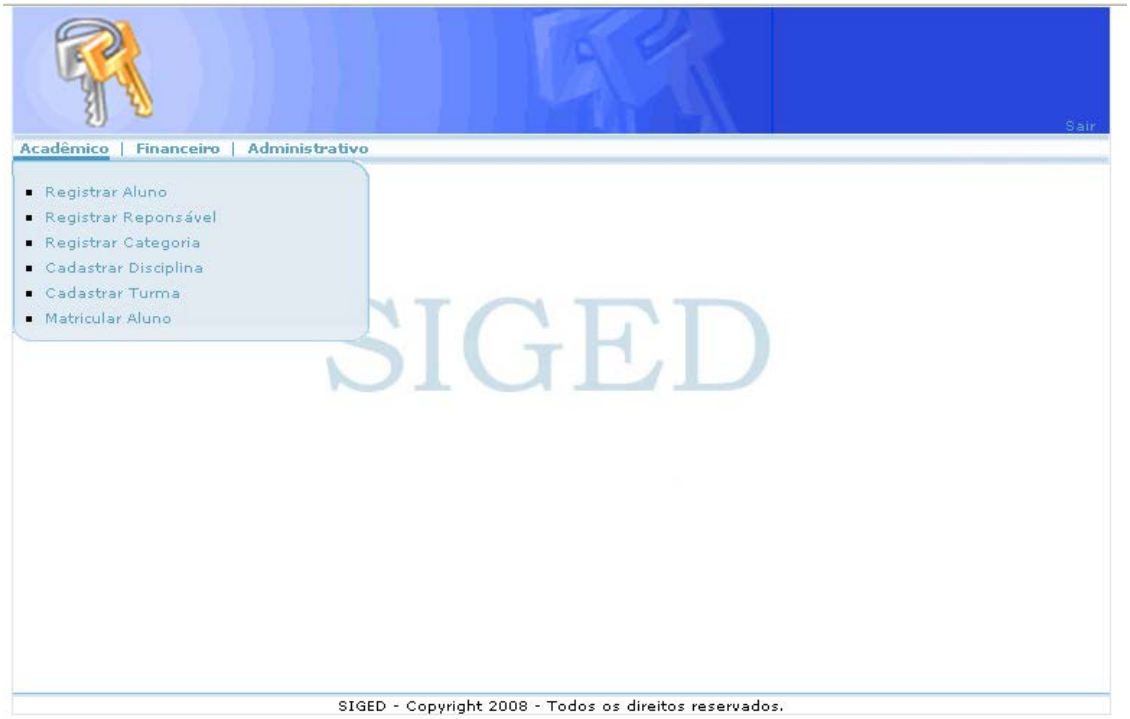
9.1 – Login Sistema

9.2 - Usuário inválido

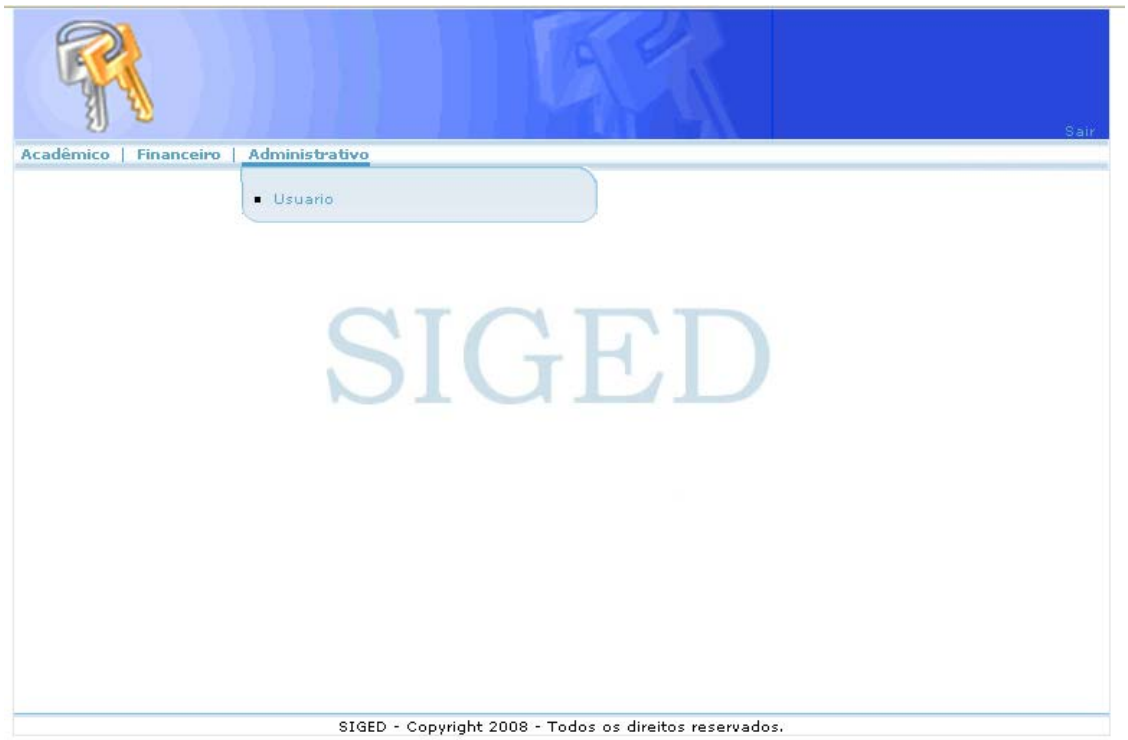
9.3 – Tela Principal do Sistema – Menus do Sistema



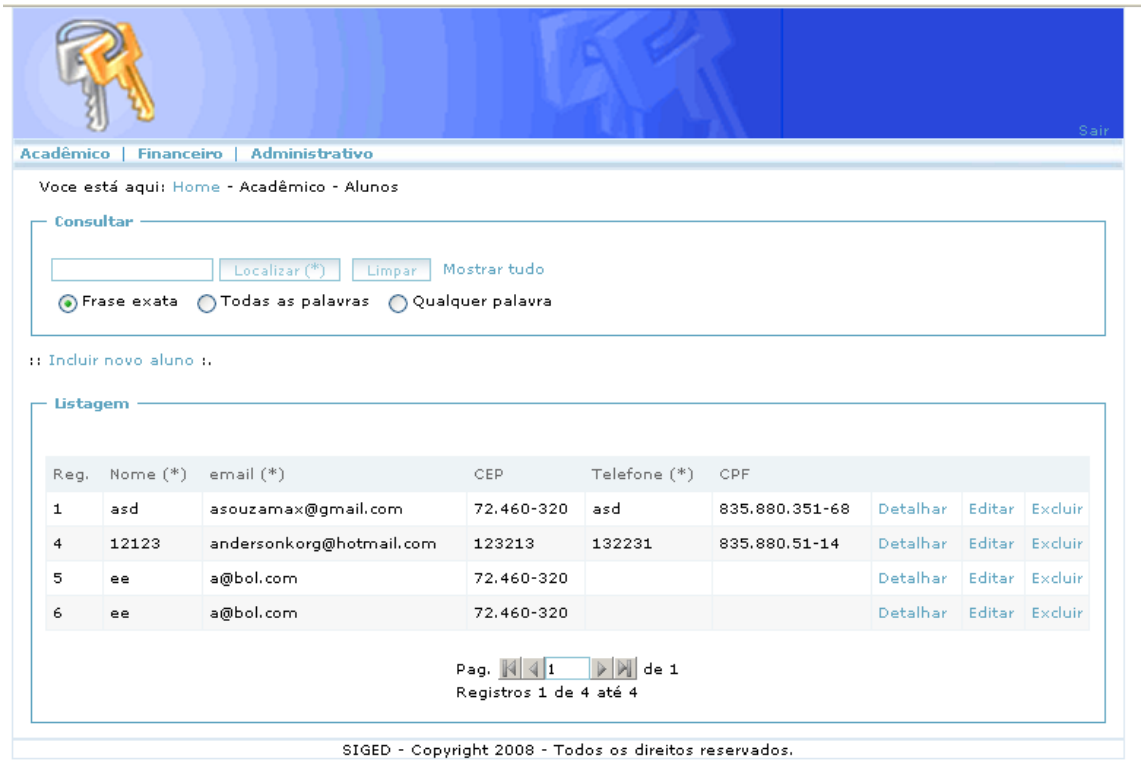
9.4 - Menu Acadêmico



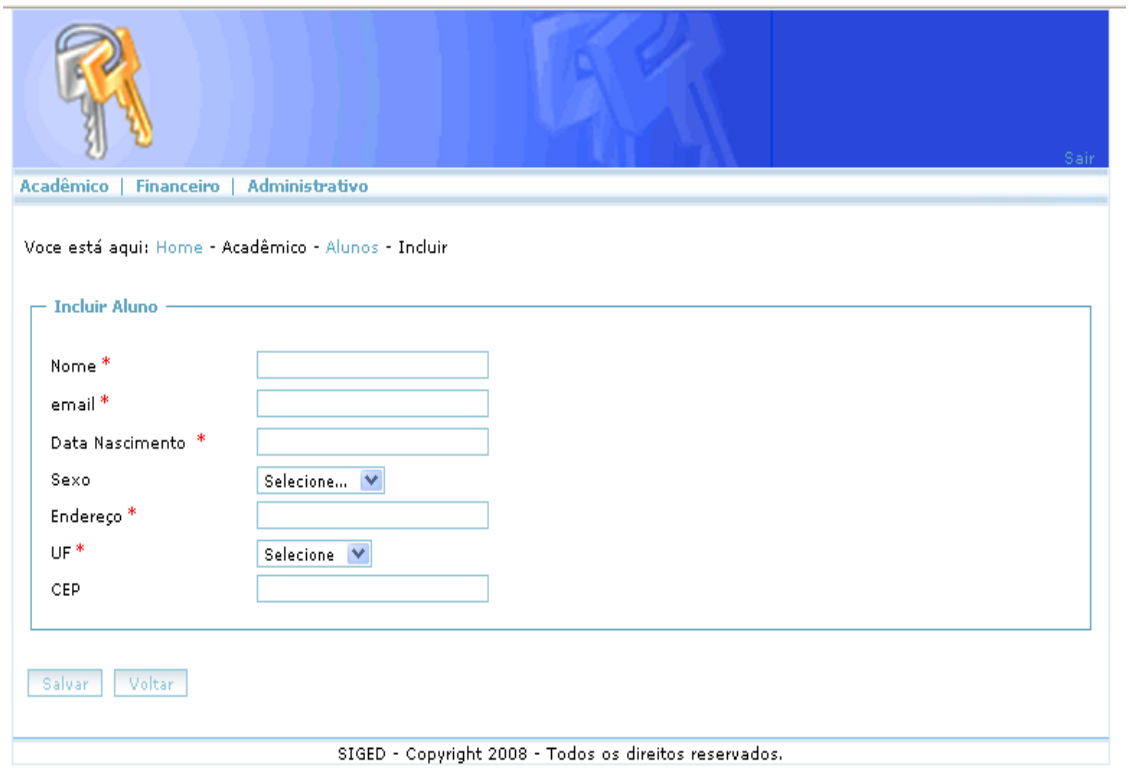
9.5 - Menu Administrativo



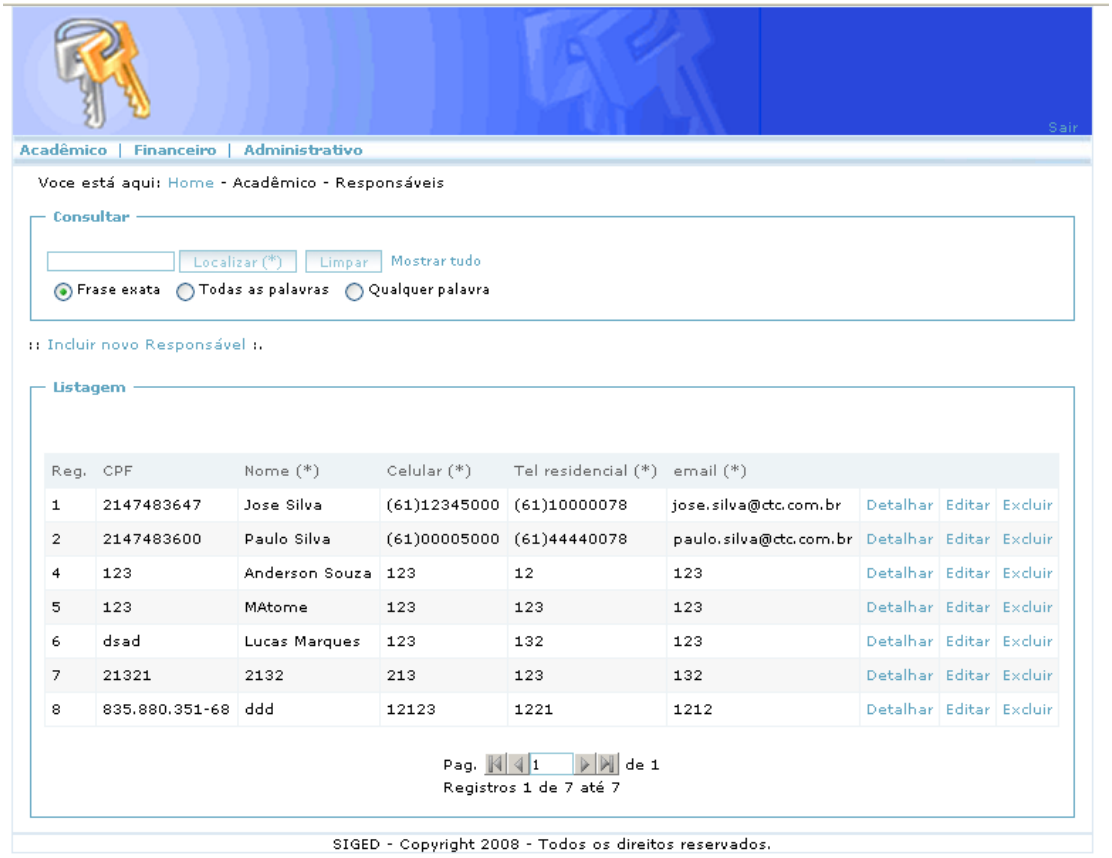
9.6 - Tela de Listagem de Alunos



9.7 - Tela de Inclusão de Alunos



9.8 - Tela de Listagem de Responsáveis



9.9 - Tela de Inclusão de Responsáveis

Voce está aqui: Home - Acadêmico - Responsáveis - Incluir

Incluir Professor

CPF *	83588035168
Nome *	Anderson de Souza Alves
Endereço *	Q 27 lote 13
UF *	DF
CEP	72460300
Celular	
Data Nascimento *	28/04/1979
Tel residencial	
Sexo	Masculino
Cidade	
email *	asouzamax@gmail.com
Observação *	teste
Profissão	
Parentesco	

Responsável por :

Alunos: Selecione Incluir

Aluno
asd
ee

Salvar Voltar

9.10 - Tela de listagem de Categorias de séries

Sair

Acadêmico | Financeiro | Administrativo

Voce está aqui: Home - Acadêmico - Categorias

Consultar

Localizar (*) Limpar Mostrar tudo

Frase exata Todas as palavras Qualquer palavra

:: Incluir nova categoria :.

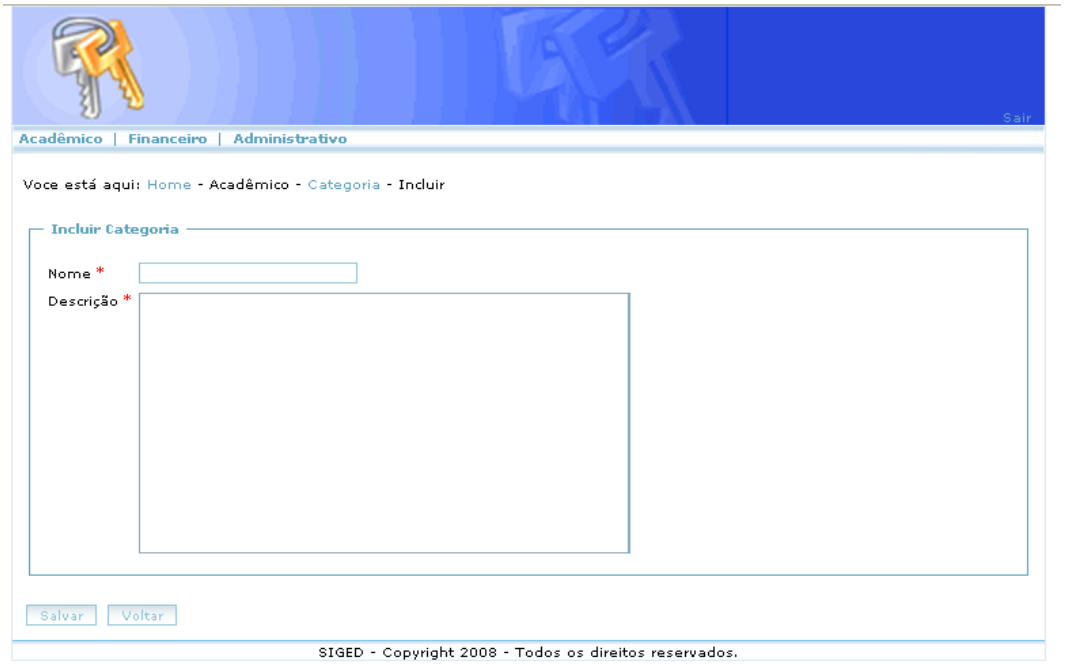
Listagem

Reg.	Nome (*)	Descrição (*)			
1	1ª Série		Detalhar	Editar	Excluir
2	2ª Série		Detalhar	Editar	Excluir
3	aa	aa	Detalhar	Editar	Excluir

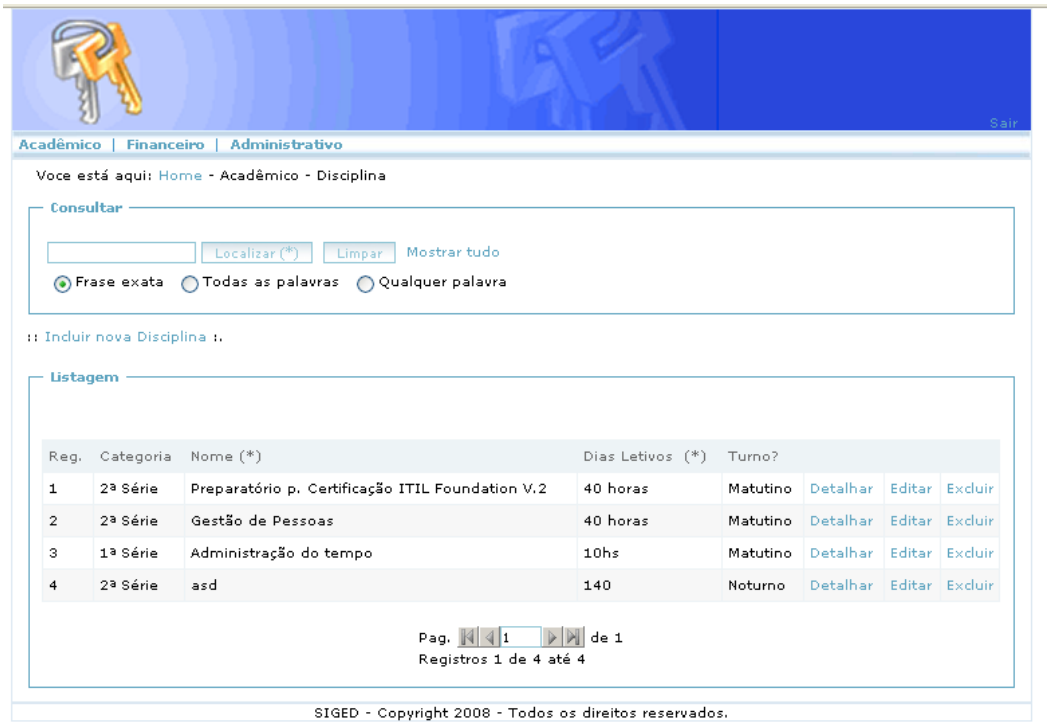
Pag. 1 de 1
Registros 1 de 3 até 3

SIGED - Copyright 2008 - Todos os direitos reservados.

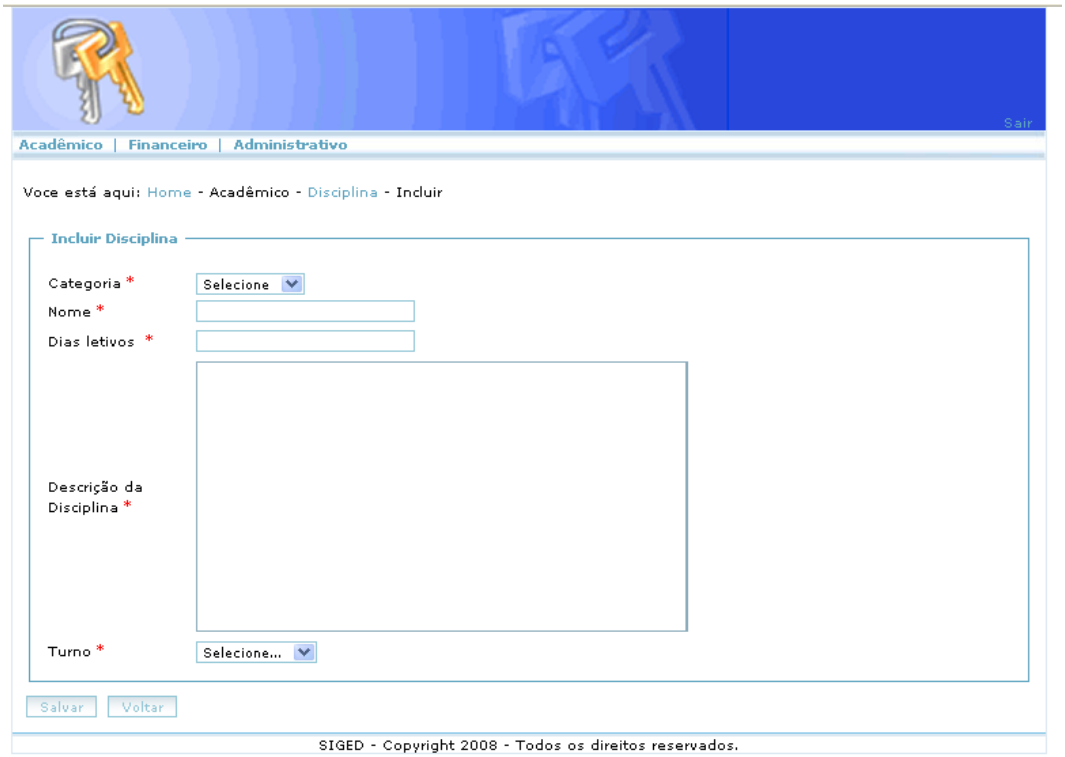
9.11 - Tela de Inclusão de Categorias da série



9.12 - Tela de Listagem de Disciplinas

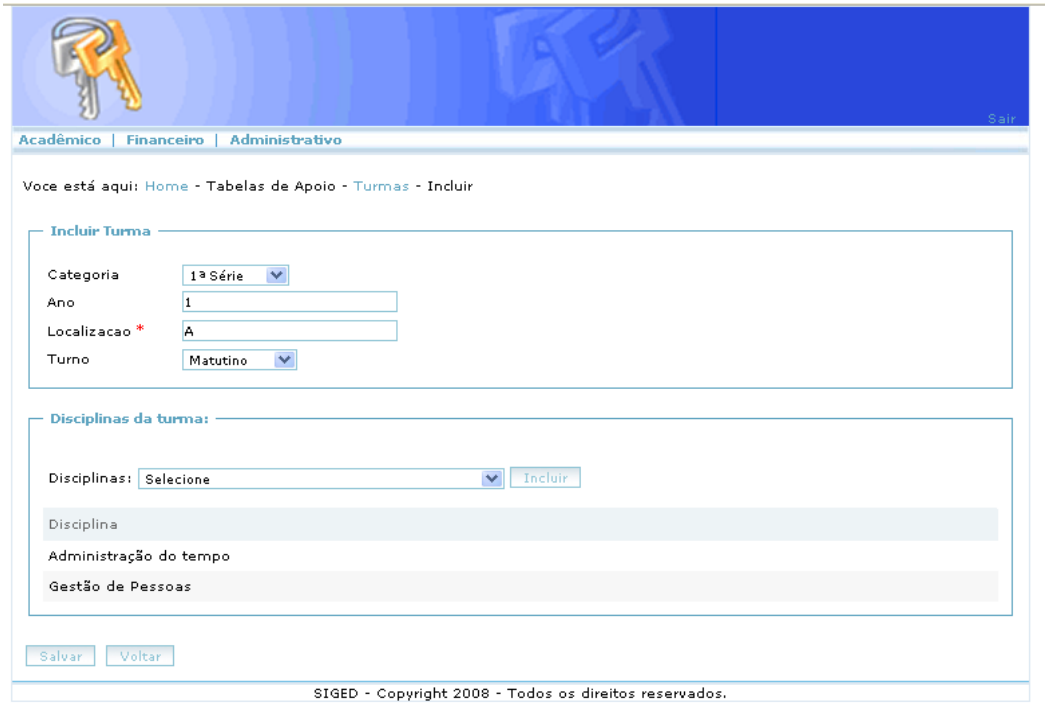


9.13 - Tela de Inclusão de Disciplinas



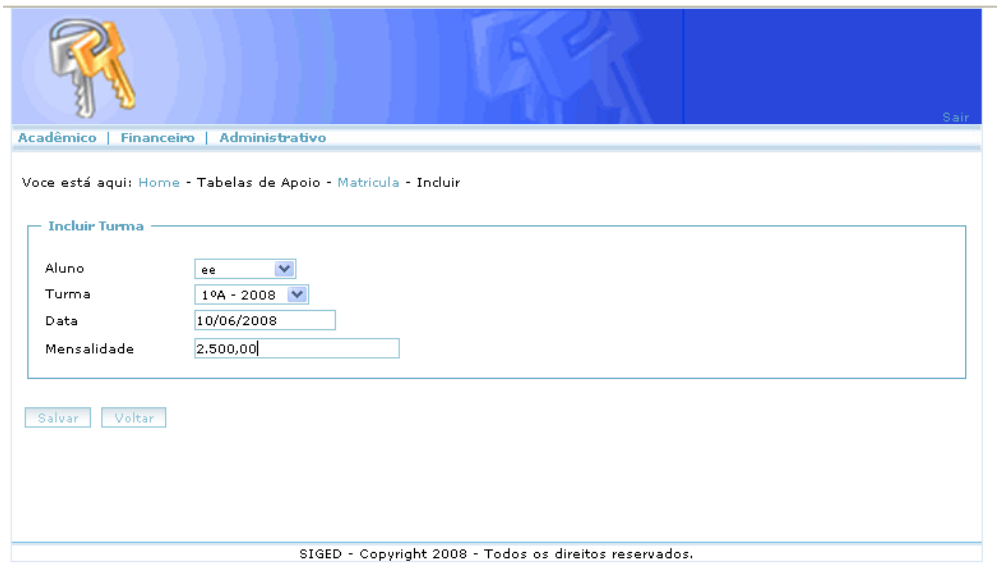
9.14 - Tela de Listagem de Turmas

9.15 - Tela de Inclusão de Turmas

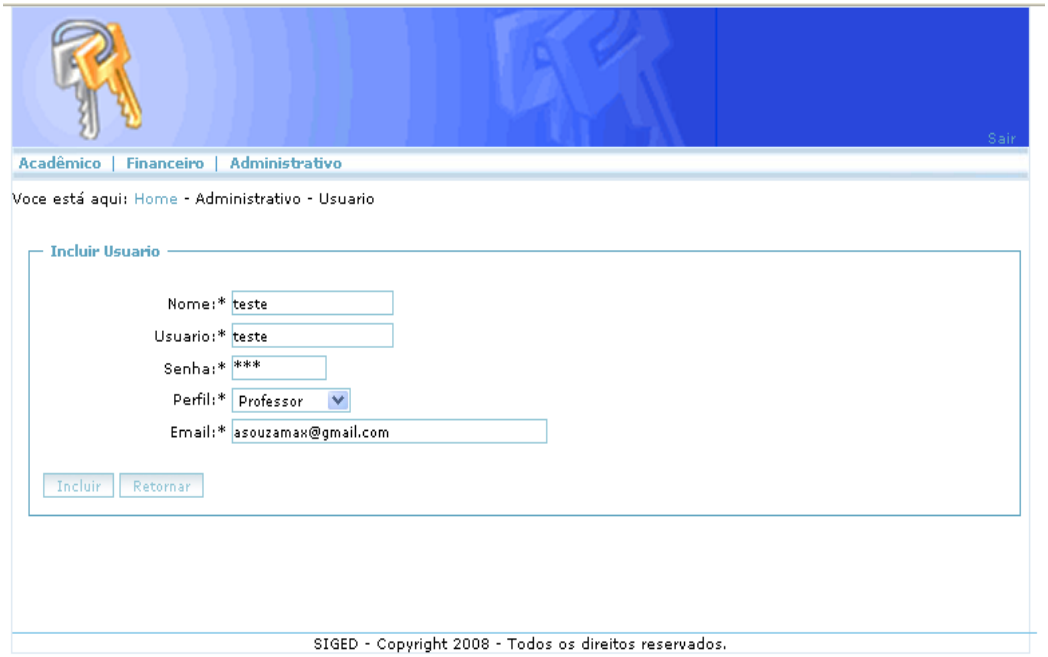


9.16 - Tela de Listagem de Alunos

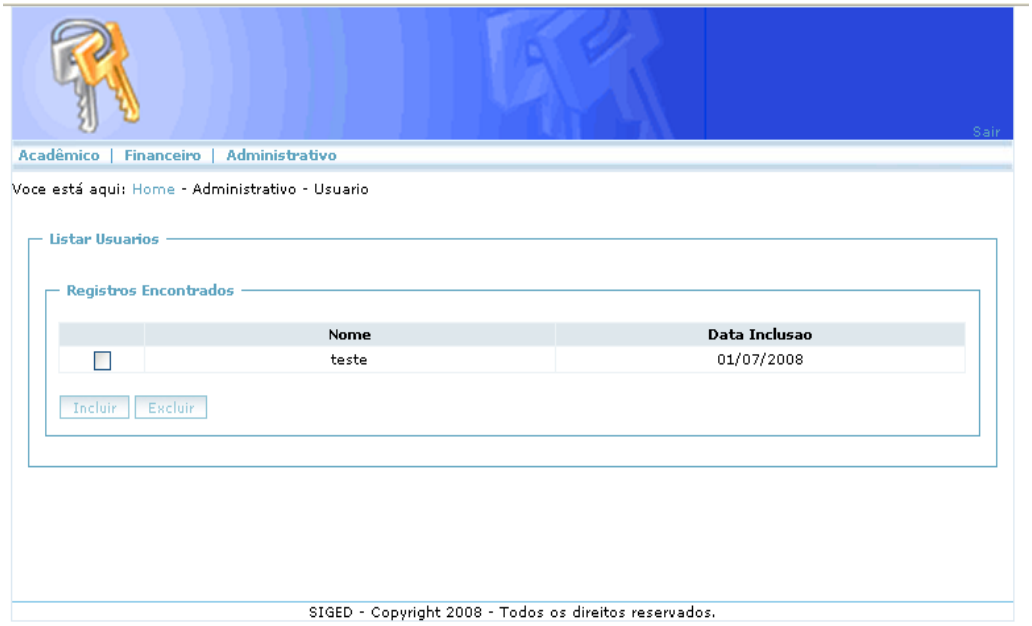
9.17 - Tela de Matrícula de Alunos



9.18 - Tela de Inclusão de Usuários do Sistema



9.19- Tela de Listagem de Usuários do Sistema



CONCLUSÃO

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

O advento da arquitetura orientada a serviços (SOA) não é apenas outra moda passageira, um dentre tantos conjuntos de produtos e serviços: acredita-se que ela terá o mesmo impacto de longo alcance que a Internet teve quando surgiu como empreendimento comercial no início dos anos 90. Olhando para o passado, muitas pessoas não tinham a menor idéia de como esta tecnologia seria usada realmente no mundo comercial.

Assim como a Internet, SOA terá conseqüências imprevistas para o futuro da computação. Estima-se que a arquitetura orientada a serviços vai forçar uma mudança drástica no equilíbrio de poder no mundo do software, tanto do lado do fornecedor quanto do cliente. O novo modelo pode representar uma mudança radical no modo como o software é criado, reutilizado, recombinado, gerenciado e vendido. Da perspectiva do cliente, SOA põe o poder nas mãos do usuário de negócio.

Referência Bibliográfica

1 – [Josuttis,2008] Nicolai M. Josuttis . SOA na Pratica**,** Alta *Books* , Fevereiro de 2008

2 – [Schulte e Natis,1996] SCHULTE, W. R. ; NATIS, Y. V. N.. *Service-Oriented Architecture*. Gartner, Abril de 1996.

3 – [Ikeda, 2006], Ikeda Blog. Mudança de Paradigma: Arquitetura SOA. 2006, Disponível em: http://ikedarevolution.typepad.com/ikeda/2006/11/mudana_de_parad.html. Acesso em 18 de março de 2008.

4 – [Mendes,2007], Marco Mendes. Arquitetura Orientada por Serviços. 2007, Disponível em: http://www.marcomendes.com/ArquivosBlog/Introdu%E7%E3oSOA.pdf. Acesso em 02 de maio de 2008.

5 – [Junior,2007], Antonio Junior. Roteiro para a definição de uma arquitetura SOA utilizando BPM. 2007, Disponível em: http://www.pece.org.br/cursos/TI/monografias/MBA-MONO-AntonioCarlosJr.pdf. Acesso em 02 de maio de 2008.

6 -[Bianchi,2007], Claudio Bianchi. Método para análise, especificação, desenvolvimento e governança de uma SOA. 2007, Disponível em: http://www.pece.org.br/cursos/TI/monografias/MBA-MONO-ClaudioBianchi.pdf. Acesso em: 02 de maio de 2008.

7 – [Koch,2006], Christopher Koch. ABC da SOA.2006, Disponível em: http://cio.uol.com.br/tecnologia/2006/07/17/idgnoticia.2006-07-17.3732358054/. Acesso em: 15 de março 2008.