FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DIOGO MARCOS DOS SANTOS

ARQUITETURA ORIENTADA A SERVIÇOS: DA TEORIA À PRÁTICA

MARÍLIA
2008

DIOGO MARCOS DOS SANTOS

ARQUITETURA ORIENTADA A SERVIÇOS: DA TEORIA À PRÁTICA

Monografia apresentada ao Curso de Ciência da Computação do Centro Universitário de Marília, mantido pela Fundação de Ensino "Eurípides Soares da Rocha", como requisito para obtenção do Título Bacharel em Ciência da Computação.

Orientador (a):
Profª. Dra. Maria Istela Cagnin Machado

MARÍLIA
2008

SANTOS, Diogo Marcos dos

Arquitetura Orientada a Serviços: da Teoria à Prática / Diogo Marcos dos Santos; orientador (a): Profª. Dra. Maria Istela Cagnin Machado. Marília, SP: [s.n.], 2008.

49 f.

Trabalho de Conclusão de Curso Bacharelado em Ciência da Computação - Centro Universitário Eurípides de Marília – Fundação de Ensino Eurípides Soares da Rocha.

1. Arquitetura de Software 2. WebServicess 3. SOA

CDD: 005.1

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Diogo Marcos dos Santos

ARQUITETURA ORIENTADA A SERVIÇOS: DA TEORIA À PRÁTICA

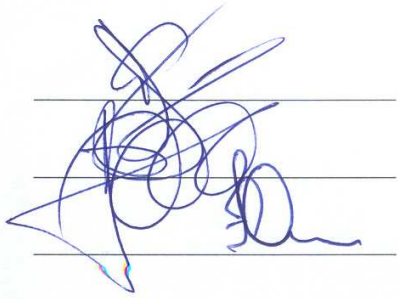
Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 6,0 (seis)

Orientador: Elvis Fusco

1°. Examinador: Jose Eduardo Santarem Segundo

2°. Examinador: Fabio Lucio Meira



Marília, 19 de novembro de 2008.

Á Deus por ter concedido a oportunidade de realizar meus sonhos;

À minha mãe Geni, meus irmãos Luciane e Daniel, pelo apoio, dedicação e carinho nessa jornada.

AGRADECIMENTOS

Agradeço o carinho de todos os colegas da Fundação de Ensino "Eurípides Soares da Rocha", que foram magníficos para o sucesso desse trabalho.

Agradeço em especial:

Aos amigos Claudinei Antonio da Silva e Wesley Portugal de Matos, pelo companheirismo e apoio para o sucesso dessa jornada.

À profª Drª Maria Istela Cagnin Machado, pela belíssima orientação que foi imprescindível para conclusão de trabalho.

Bem aventurado o homem que não anda segundo o conselho dos ímpios, nem se detém no caminho dos pecadores, nem se assenta na roda dos escarnecedores.

Antes tem o seu prazer no Senhor, e na sua lei medita de dia e de noite.

Será como árvore plantada junto a ribeiros de águas, a qual dá seu fruto na estação própria e suas folhas não caem. Tudo o que fizer prosperará. Os ímpios não são assim, mas são como a moinha que o vento espalha.

Porque os ímpios não subsistirão no juízo, nem os pecadores na congregação dos justos.

Pois o Senhor conhece o caminho dos justos, mas o caminho dos ímpios perecerá.

Salmos 1:1-6.

SANTOS, Diogo Marcos dos. **Arquitetura Orientada a Serviços: da Teoria à Prática**. 39 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMO

Engenharia de Software é essencial e deve ser considerada no desenvolvimento de qualquer tipo de software. No entanto, existem diversas metodologias e padrões de desenvolvimento de software, sempre descrevendo vantagens em sua utilização, porém é difícil escolher quais devem ser utilizados no desenvolvimento de uma aplicação. Neste trabalho será apresentada a arquitetura SOA, ou seja, Arquitetura Orientada a Serviços, que ultimamente tem sido um tema abordado por várias empresas de tecnologia para fornecer às áreas de TI uma melhor gestão no desenvolvimento das aplicações das empresas. Para facilitar o entendimento da arquitetura, será apresentado neste trabalho o desenvolvimento de uma aplicação utilizando SOA, a fim de destacar como os conceitos de tal arquitetura são aplicados na prática.

Palavras-chave: SOA. Arquitetura de Software. WebServices. Serviços. Interoperabilidade.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

SOA - *Service Oriented Architecture*

XML - Extensible Markup Language

SOAP - *Simple Object Access Protocol*

UDDI - *Universal Description Discovery Integration*

ESB - Enterprise Service Bus

WSDL - *Web Services Description Language*

HTTP - HyperText Transfer Protocol

PTC - Parametric Technology Corporation

JDBC - Java Database Connectivity

JMS - Java Message Service

SUMÁRIO

INTRODUÇÃO

Motivação

Conforme Pressman (1995), o termo Engenharia de Software foi usado pela primeira vez em 1969 por Friedrich Ludwig Bauer na *NATO Conference on Software Engineering* (Conferência sobre Engenharia de Software da OTAN), e foi definido como:

> O estabelecimento e uso de sólidos princípios de engenharia para obter software economicamente confiável e que trabalhe de forma eficiente em máquinas reais.

Atualmente a *Engenharia de Software* é um ponto relevante para o desenvolvimento de softwares, e pode ser definida como uma área do conhecimento voltada para a especificação, desenvolvimento e manutenção de sistemas de software. De acordo com Pressman (1995), engenharia de software é definida como um rebento da engenharia de sistemas de hardware que abrange um conjunto de três elementos fundamentais: métodos, ferramentas e procedimentos.

- Métodos: proporcionam detalhes de como fazer e construir software.
- Ferramentas: estabelece um suporte ao desenvolvimento de software utilizando ferramentas CASE (*Computer-Aided Engineer Software*), que automatizam os métodos utilizados.
- Procedimentos: é o elo entre os métodos e ferramentas que possibilitam o desenvolvimento racional e oportuno de software.

Na década de 90 surgiu o termo SOA (*Service Oriented Architecture* – Arquitetura Orientada a Serviços), como o propósito de flexibilizar as soluções de TI permitindo automatizar processos comuns (JOSUTTIS, 2008).

Diversos estudos apontavam que arquiteturas orientadas a serviços, em alguns anos, teriam grande influência sobre o desenvolvimento de novos sistemas. Segundo prognósticos do Instituto Gartner, com 70% de probabilidade, até 2008 SOA seria a prática de Engenharia de Software predominante (McCoy, 2003).

Objetivos

O presente trabalho tem como objetivo apresentar as características essenciais de SOA e sua aplicação no desenvolvimento de software. O objetivo é identificar as vantagens e

os resultados obtidos durante o processo de desenvolvimento de um software utilizando tal técnica.

Organização da Monografia

Esta monografia está organizada da seguinte forma:

- No Capítulo 1 é apresentada uma Visão Geral Sobre SOA (Arquitetura Orientada a Serviços).
- No Capítulo 2 são apresentados padrões que compõem um ambiente SOA.
- No Capítulo 3 são apresentadas algumas tecnologias para criar um ambiente SOA.
- No Capítulo 4 é apresentado um estudo de caso para um cenário SOA a fim de apresentar os conceitos e tecnologias discutidos nos capítulos 1 a 3.
- No Capítulo 5 é apresentada uma discussão sobre os resultados obtidos com a utilização de SOA.

1. ARQUITETURA ORIENTADA A SERVIÇO (SOA)

1.1. Considerações Inicias

Neste capítulo serão apresentados os conceitos e termos tratados por SOA. Na Seção 1.2 é fornecida uma visão geral sobre SOA, apresentando principalmente sua definição. Na seção 1.3 é descrita a característica de serviços, um dos conceitos principais de SOA.

1.2. Visão Geral sobre SOA

Muitos autores definem SOA de diferentes maneiras, porém todas estabelecem pontos em comum: compartilhar serviços; tornar a aplicação independente da plataforma, independente de linguagem de programação e possibilitar flexibilidade e agilidade no desenvolvimento de uma aplicação para gerenciar um negócio (ERL, 2008).

SOA não é uma aplicação que pode ser comprada, mas SOA pode ser referida como uma filosofia, um paradigma, um conceito para pensar em como tomar decisões para projetar uma arquitetura de software (JOSUTTIS, 2008).

Thomas ERL (2005) define SOA como, arquitetura aberta, extensível e composta que promove a Orientação a Serviços e que compreende serviços autônomos, capazes de qualificar os serviços de diferentes fabricantes para que possam ser reutilizáveis.

De acordo com JOSUTTIS (2008), SOA é um paradigma para realização e manutenção de processos corporativos e baseia-se em três conceitos: serviços, interoperabilidade e acoplamento fraco. Esses conceitos são apresentados respectivamente nas Seções 1.3, 1.4 e 1.5.

1.3. Serviços

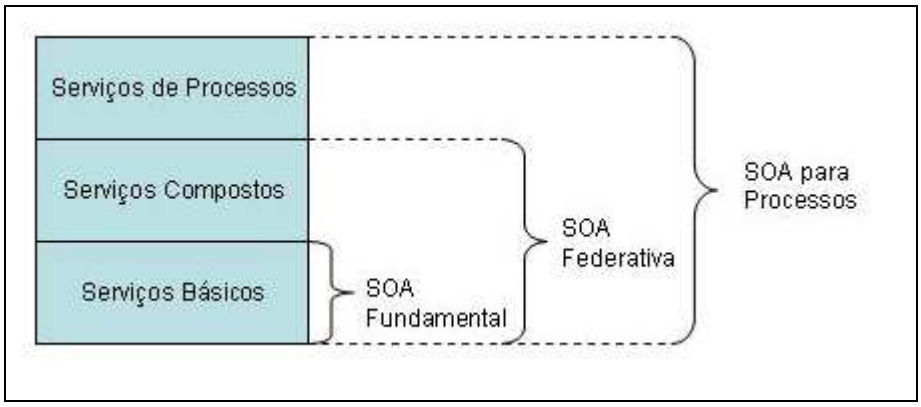
Da mesma forma que se têm várias definições para SOA, se tem para serviços. Serviço é um substantivo que indica uma ação ou função executada por algo ou alguém. No conceito de SOA adota-se a seguinte definição para serviço: uma representação de alguma funcionalidade do negócio, independente e sem estado, que aceita uma ou mais requisições e retorna uma ou mais respostas por meio de uma interface padronizada e bem definida (JOSUTTIS, 2008).

Um serviço em SOA é uma função de aplicação disponibilizada como um componente reutilizável no desenvolvimento de um software. Através de protocolos de

comunicação bem definidos, serviços podem ser invocados por aplicações distintas garantindo interoperabilidade e transparência (COLAN, 2004).

Josuttis (2008) classifica serviços em três tipos: Básicos, Compostos e de Processos, que estão explicados nas subseções a seguir. Conforme ilustrado na Figura 1, de acordo com os três tipos de serviços são definidas as camadas de expansão.

Figura 1 – Camadas de Expansão



Fonte: Josuttis 2008

Maiores detalhes sobre cada camada e tipo de serviço são apresentados nas subseções a seguir.

1.3.1. Serviços Básicos – SOA Fundamental

Os serviços básicos são aqueles em que cada serviço provê uma funcionalidade básica da aplicação e que não faz sentido que sejam divididos em múltiplos serviços. Geralmente são serviços de rápida execução e sem estado, isto é, depois de invocado o serviço todas variáveis e objetos locais são descartados. Esses serviços devem ter as seguintes características, serem atômicos, consistentes, isolados e duráveis. Como exemplos de serviços básicos têm-se: criação de um novo cliente, retorno do endereço do cliente, criar um novo portfólio.

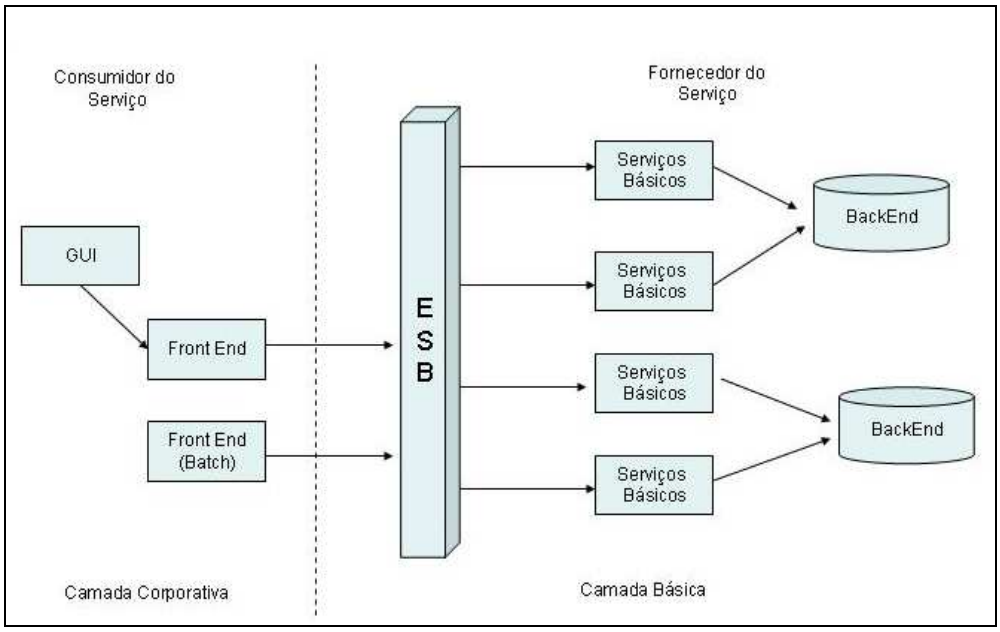
As seguintes orientações são definidas para alcançar um ótimo nível de granularidade para os serviços básicos (ALLEN, 2006):

- Deve ser possível descrever os serviços em termos de funções, informando objetivos e regras;
- Conjunto de funções de um serviço deve operar como uma unidade de família que oferece capacidade de negócio;
- Um papel único de tomar a responsabilidade pelo serviço, e;

- O serviço deve ser auto-suficiente o máximo possível.

Na Figura 2 é ilustrada, em detalhes, a camada de expansão SOA Fundamental oriunda dos serviços básicos, que consiste em consumir um serviço disponibilizado pela infra-estrutura de ESB (*Enterprise Service Bus*) e que cada serviço e responsável por uma atividade dentro de um BackEnd, isto é, uma aplicação final.

Figura 2 – SOA Fundamental



Fonte: Josuttis 2008

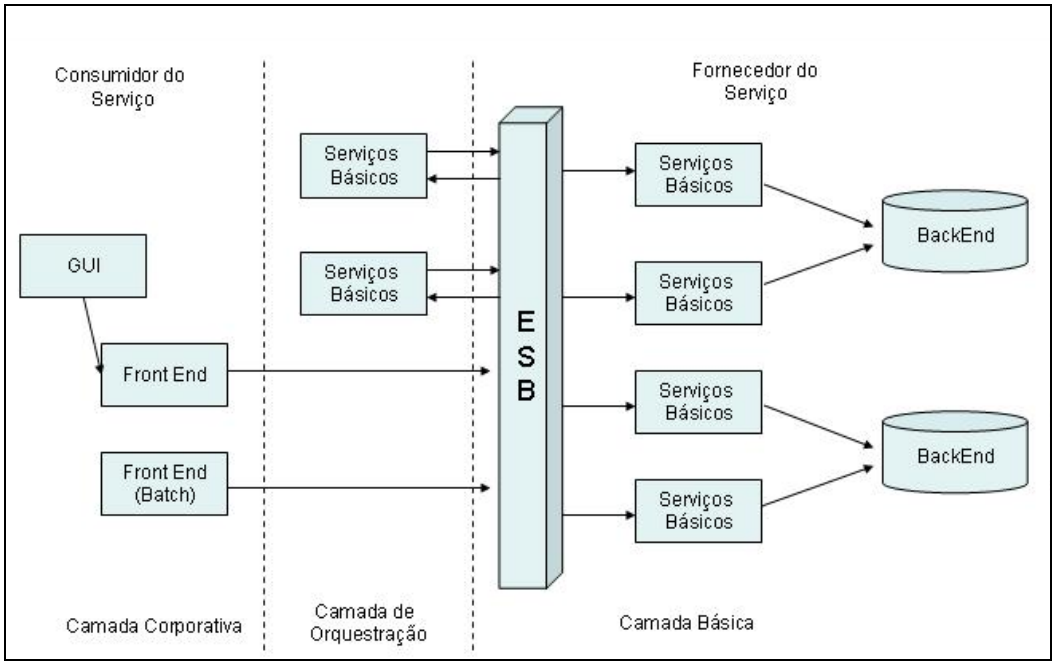
1.3.2. Serviços Compostos – SOA Federativa

Tem os mesmos conceitos dos serviços básicos, porém podem ser compostos por outros serviços básicos já existentes. Na terminologia SOA, compor novos serviços partindo de serviços existentes é chamado de *Orquestração*. Como em uma orquestra, são combinados vários instrumentos diferentes para executar tarefas que são mais complexas de se executar com um único instrumento. Esses serviços também podem ser chamados de *Serviços Orquestrados.* (JOSUTTIS, 2008). Uma transferência entre contas bancárias de um cliente é um exemplo de serviço composto. Tal serviço, na camada de expansão é composto por um serviço básico que debita o valor de uma conta, e outro serviço básico que credita o valor na conta destino.

Na Figura 3 é ilustrada, em detalhes, a camada de expansão oriunda dos serviços compostos. Conforme exemplo da transferência bancaria citado acima, o *FrontEnd* forneceria

o serviço de transferência e o ESB executaria dois serviços básico, de débito e crédito respectivamente para a conclusão do serviço de transferência solicitado.

Figura 3 – SOA Federativa



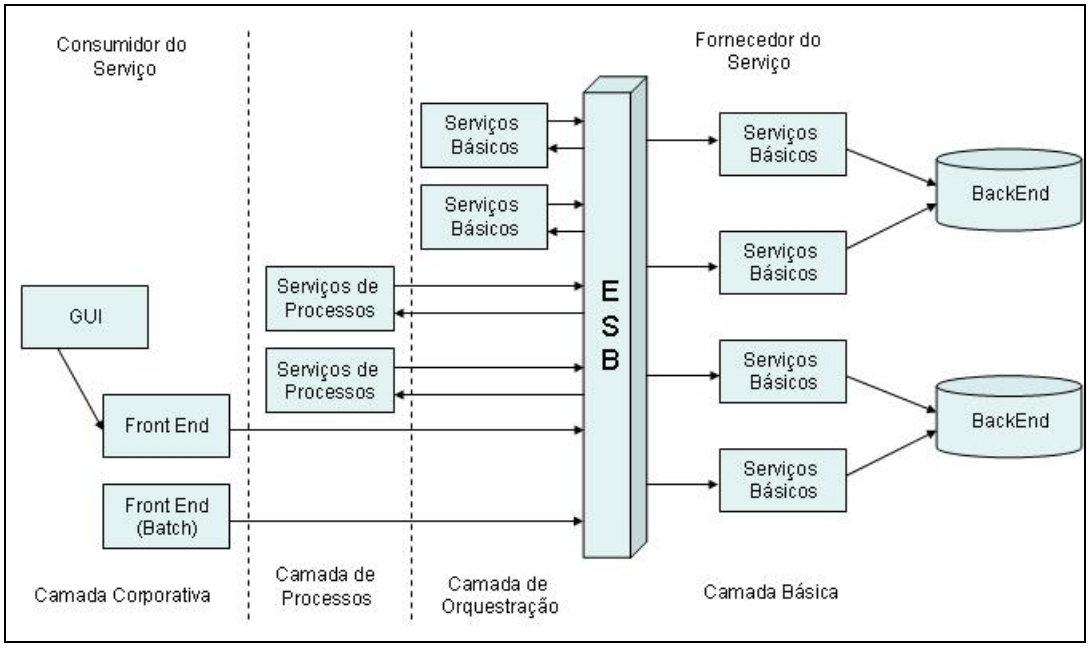
Fonte: Josuttis 2008

1.3.3. Serviços de Processos – SOA para Processos

Um serviço de processo representa um *workflow* ou um processo de negócio de longo prazo. Do ponto de vista de negócio, este tipo de processo representa um fluxo de atividades *(serviços)* que não podem ser interrompidos. Diferentemente de serviços básicos e compostos, estes serviços possuem um estado que permanece estável durante múltiplas chamadas do serviço. Um exemplo de serviços de processo citado por Josuttis (2008) seria a aquisição de uma apólice de seguro, que em seu processo de criação solicitaria a aprovação de dados do cliente, e que essa solicitação de dados fosse executada por um serviço básico que fornece essa atividade.

Na Figura 4 é ilustrada, em detalhes, a camada de expansão oriunda dos serviços de processos, onde uma solicitação do usuário utilizando um *FrontEnd* pode se estender por vários serviços das camadas de expansão seguindo um fluxo determinado.

Figura 4 – SOA para Processos



Fonte: Josuttis 2008

1.4. Interoperabilidade

Com o advento de SOA, o objetivo é que os serviços possam ser consumidos entre sistemas diferentes, e que um sistema possa se conectar a outro facilmente. Exemplo, utilizando a tecnologia de *WebServices* podemos consumir os serviços disponibilizados em aplicações escritas em diferentes linguagens (JOSUTTIS, 2008).

Essa interoperabilidade é oferecida por um Barramento de Serviços Corporativos (ESB – *Enterprise Service Bus*) (COLAN, 2004).

Um aspecto geral de um *ESB,* é que essa estrutura seja responsável por prover a troca de informações entre os serviços requisitados pelos usuários, fornecendo a transformação de dados, monitoramento e gerenciamento dos serviços disponíveis. Enfim, um ESB é a infra-estrutura de um cenário SOA que possibilita a interoperabilidade dos serviços (JOSUTTIS, 2008).

Segundo Colan (2004), a IBM vê SOA como chave principal para interoperabilidade e flexibilidade requerida na demanda comercial. SOA suporta integração *End-toEnd,* ou seja, integrar aplicações de diferentes domínios e implementação entre empresas e parceiros comerciais.

Muitas integrações comerciais têm se beneficiado do padrão que habilita a interoperabilidade eficiente entre sistemas heterogêneos. Essas integrações são denominadas, coletivamente, como *WebServicess* (W3C, 2006).

O início de *WebServicess* foi marcado com a introdução do protocolo SOAP (Simple Object Access Protocol - W3C, 2007), o qual define o uso de XML (Extensible Markup Language - W3C, 2006) como padrão de interação entre sistemas (COLAN, 2004).

Salienta-se que um *WebServices* pode ser considerado como um ESB, pois conceitualmente ele define tudo o que é necessário para prover interoperabilidade entre sistemas, sem a necessidade de adquirir algum software ou hardware específico (JOSUTTIS, 2008).

1.4.1. Acoplamento Fraco

Como mencionado na Seção 1.4, interoperabilidade e flexibilidade são uns dos conceitos chave para implantar SOA. Nesse contexto, acoplamento fraco significa estar apto para implementar mudanças no sistema rapidamente, ou seja, de forma ágil.

Atualmente, tempo de efetuar mudanças em um sistema e atender uma nova regra de negócio pode ser longo, quando essas mudanças requerem modificações em várias partes do sistema (JOSUTTIS, 2008).

Em um cenário SOA, acoplamento fraco implica ter o mínimo de dependências na infra-estrutura modelada.

1.5. SOA – Cenário e Oportunidade

Segundo McCoy (2003), O Instituto Gartner previu que de 2003 à 2008 SOA se tornaria um padrão de desenvolvimento de software. Nesse aspecto não podemos confirmar essa previsão, mas podemos citar várias empresas que tem investido no tema.

Takahashi (2008) em seu artigo sobre cenário SOA menciona que:

> A SAP está 100% comprometida com a arquitetura orientada a serviços, conforme afirmou o Diretor-Executivo Henning Kagermann, em Abril de 2008. Sobre as principais tendências que direcionam a adoção de SOA estão as fusões e aquisições de empresas e a inovação como fator de diferenciação. A maior contribuição que a SOA traz é a possibilidade dos processos de negócios serem formados por composição de serviços corporativos, a fim de que mudanças de negócios sejam rapidamente atendidas de maneira rápida e eficiente.

D'Auria (2008) comenta que:

> A adoção de SOA extrapola o conceito de ser válida apenas em grandes empresas e mostra que sua adoção sempre traz benefícios significativos.

> Tudo isso é possível graças ao avanço da tecnologia e a adoção de padrões abertos (por exemplo, XML, HTTP, WSDL, SOAP, UDDI – Capítulo 2), nos quais os produtos são baseados, garantindo assim, confiabilidade e funcionalidade aliadas a um baixo custo.

1.6. Considerações Finais

Este capítulo esclarece os principais aspectos em torno de SOA (Arquitetura Orientada a Serviços), expondo principalmente os conceitos do que vem a ser SOA. As principais características de SOA são prover funcionalidades como serviços que interagem entre si com mínima dependência.

2. COMPOSIÇÃO DE UM AMBIENTE SOA

2.1. Considerações Iniciais

Nesse Capítulo são apresentados padrões que compõem um ambiente para implantação de SOA. Na seção 2.2 está descrito a importância do uso de disciplinas e ferramentas para a definição de um software.

2.2. Padrões de Apoio ao Desenvolvimento de Software

Conforme Zimmermann (2004, p.01) afirma em seu artigo, existem ferramentas e disciplinas, como Orientação a Objetos, *Enterprise Arquitecture* (EA) e Processos de Modelagem de Negócio BPM (*Business Process Modeling*), que nos fornece práticas de alta qualidade para auxílio na identificação e definição de abstrações dentro de uma arquitetura. De qualquer forma, essas práticas se tornam um pouco ineficiente quando aplicadas de formas independentes uma das outras.

Utilizando-se desses conceitos já amplamente utilizado no desenvolvimento de software, Zimmermann (2004, p.01) propõe um modelo de desenvolvimento, o *Service-Oriented Analysis and Design* (SOAD), que mescla elementos da Orientação a Objetos, BPM e EA para o desenvolvimento de aplicações baseadas em SOA.

Nesse contexto de SOA, várias empresas têm investindo nesse novo conceito de desenvolvimento, empregando a utilização de *WebServicess* como tecnologia para implantação de SOA. *WebServicess* se refere a um conjunto de padrões que fornecem interoperabilidade (JOSUTTIS, 2008): XML (*Extensible Markup Language*), HTTP (*HiperText tranfer Protocol*), WSDL (*Web Services Description Language*), SOAP (*Simple Object Access Protocol*) e UDDI (*Universal Description Discovery Integration*). Esses padrões, apresentados nas próximas subseções, têm sido organizados por diversas instituições de padronização, como Consortium W3C (*World Wide Web Consortium*), OASIS (*Organization for the Advancement of Structured Information Standards*) e WS-I (*WebServicess Interoperability Organization*).

2.2.1. Extensible Markup Language (XML)

O padrão *XML* (W3C, 2006) é um subconjunto de *SGML (Standard Generalized Markup Language,* ISO 879:1986), descreve uma classe de objetos de dados e especifica

parcialmente o comportamento dos programas de computador que irá processá-la, ou seja, quais aplicações utilizaram os dados transportados pela estrutura do arquivo XML.

Documentos XML são constituídos de unidades chamadas entidades que estruturam os dados a serem analisados. Essas entidades são *markups,* que codificam as descrições do documento e armazena sua estrutura lógica e o seu *layout* (W3C, 2006).

Na Figura 5 é ilustrado um exemplo de estrutura de um arquivo XML.

Figura 5 – Exemplo XML

```
<?xml version="1.0"?>
<aviso>
<para>Janice data="01/04/2000"</para>
<de>Jefferson</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Amanha voce tem prova de matematica</corpo>
</aviso>
```

Fonte: Marcoratti 2000

A primeira linha do documento (Figura 5) possui uma declaração XML que deve sempre ser incluída, pois define a versão XML do documento. Neste caso está sendo especificada a versão 1.0 da XML. O *Browser* de Internet deve possuir um analisador XML que atende o padrão 1.0. A linha <aviso> define o primeiro elemento do documento – o elemento raiz ou nó raiz. As demais linhas, exceto a última, definem quatro elementos filhos da raiz *(para, de, cabecalho e corpo*). A última linha </aviso>, define o fim do elemento raiz.

2.2.2. HiperText Transfer Protocol (HTTP)

HTTP (IETF, 2007) é um protocolo de baixo nível da camada de aplicação no Padrão RM-OSI /ISO (ISO-7498), usado pela Internet. É possível usá-lo para enviar *WebServices* pela rede, utilizando a tecnologia de Internet.

2.2.3. Web Services Description Language 2.0 (WSDL 2.0)

WSDL 2.0 (W3C, 2007) é um padrão mantido pela W3C e fornece um modelo no formato *XML* para descrever um *WebServices. WSDL 2.0* permite separar a descrição da funcionalidade oferecida pelo serviço dos detalhes concretos de como o serviço é oferecido (W3C, 2006).

Um arquivo *WSDL 2.0* possui três seções: o quê, como e onde. Na primeira seção especificam-se as mensagens de entrada e saída, representando o que o serviço faz. Na segunda seção definem-se como as mensagens devem se empacotadas na mensagem *SOAP* (Seção 2.2.4) e como devem ser transportadas. Além disso, define que informação deve estar no cabeçalho do *SOAP*. Na terceira e última seção descreve-se a implementação específica de um serviço Web e onde poderá ser encontrado (ROSENBERG, 2004).

2.2.4. Simple Object Access Protocol (SOAP)

O protocolo *SOAP* (W3C, 2007) foi criado para transportar mensagens XML de um computador para outro, via vários protocolos padrões de transporte. *HTTP* é o mais comum destes protocolos, obviamente porque é o predominante na Web.

SOAP fornece uma definição de informações estruturadas, baseadas em XML, a qual pode ser utilizada para trocar informações entre sistemas. Essa definição é chamada de Envelope SOAP e é composto por duas partes, *Header* (Cabeçalho) e *Body* (Corpo) (W3C, 2006), conforme é mostrado na Figura 6.

- *Header*: é um mecanismo que fornece uma maneira de transmitir a informação da mensagem *SOAP,* entretanto a informação contida no *Header* não é o dado importante da mensagem. Somente controla a informação contida na mensagem.
- *Body*: é o elemento obrigatório na mensagem *SOAP,* isto é, é onde estão as principais informações transportadas pela mensagem SOAP.

Figura 6 – Exemplo envelope SOAP

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-
              envelope">
   <env:Header>
     <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
       <n:priority>1</n:priority>
      <n:expires>2004-06-22T14:00:00-5:00</n:expires>
     </n:alertcontrol>
   </env:Header>
   <env:Body>
     <m:alert xmlns:m="http://example.org/alert">
       <m:msg>Pick up Bobby at school at 2PM</m:msg>
     </m:alert>
   </env:Body>
</env:Envelope>
```

Fonte: ROSENBERG, 2004

2.2.5. WS-Security

A principal maneira de atingir a arquitetura orientada a serviços, atualmente, é através de *WebServicess*. Com o objetivo de viabilizar a comunicação entre serviços, foi criado o protocolo *SOAP* que contém uma estrutura para as mensagens XML trocadas. Essa padronização deu origem a outros padrões, no qual se destaca o *WS-Security* (VARCHAVSKY, 2008).

O padrão *WS-Security* (OASIS, 2004), propõe um conjunto de definições para complementar o protocolo *SOAP* de forma a atingir a integridade e confiabilidade da mensagem. Apache WSS4J (Apache Foudantion) é uma implementação do padrão WS-Security. É uma API Java que pode ser utilizada para atribuir e verificar mensagens *SOAP* com o padrão de segurança *WS-Security* (Apache Foundation, 2008).

Em março de 2004, uma nova especificação do padrão WS-Security foi publicada, como "Web Services Security: SOAP Message Security 1.0", pelo grupo OASIS (*Organization for the Advancement of Structured Information Standards*) que descreve avanços na mensagem SOAP, ao especificar um perfil no uso da Assinatura XML (XML *Signature*) e da Criptografia XML (XML *Encryption*) para garantir integridade e confidencialidade para mensagens SOAP (OASIS, 2005).

3. TECNOLOGIAS PARA UM AMBIENTE SOA

3.1. Considerações Iniciais

Com o surgimento do paradigma *SOA*, muitas tecnologias têm surgido com o propósito de auxiliar o desenvolvimento de aplicações utilizando os conceitos de orientação a serviços. Neste contexto, as subseções a seguir apresentam algumas dessas tecnologias: *framework Axis* (Apache Foundation, 2008), *SAP NetWeaver* (SAP, 2008) e o *ESB Mule* (MuleSource, 2008).

3.2. FrameWork Axis

Conforme especificação da Apache Software Foundation (2008), Axis é essencialmente um *SOAP Engine*, ou seja, um *framework* para construção de processos *SOAP* cliente-servidor. O *framewor* Axis é escrito em Java e inclui as seguintes características:

- Simples servidor *stand-alone*: é uma aplicação que executa em background, sem interação com o usuário.
- *Plug-in* para *Container Servlet* assim como *Apache TomCat:* da mesma forma que o TomCat, o *framework* Axis processa as requisições do usuário enviadas por meio do protocolo HTTP e retorna as respostas para o usuário.
- Suporte para *WSDL*: durante o processo da requisição do usuário, o *framework* Axis gera o arquivo WSDL que contém as descrições do serviço a ser disponibilizado.

O *framework* Axis implementa os padrões XML, SOAP, WSDL para disponibilizar os serviços.

3.3. SAP NetWeaver

O SAP *NetWeaver* é a base tecnológica da arquitetura empresarial orientada a serviços (SOA empresarial) da SAP. SOA empresarial é um modelo para soluções corporativas baseadas em serviços e adaptável às empresas que oferecem níveis de adaptabilidade, flexibilidade e abertura para integrar processos necessárias para reduzir o custo total de desenvolvimento.

O SAP *NetWeaver* usa padrões da Internet, como HTTP, XML e serviços Web, permitindo abertura e interoperabilidade com Microsoft .NET e ambientes Java2 Platform Enterprise Edition (J2EE), como a IBM WebSphere (SAP, 2008).

A infra-estrutura para utilização do SAP NetWeaver é:

- MaxDB: sistema gerenciador de banco de dados desenvolvido pela SAP
- SAP NetWeaver Developer Studio: ambiente de desenvolvimento embasado no eclipse (versão Europa 3.3)

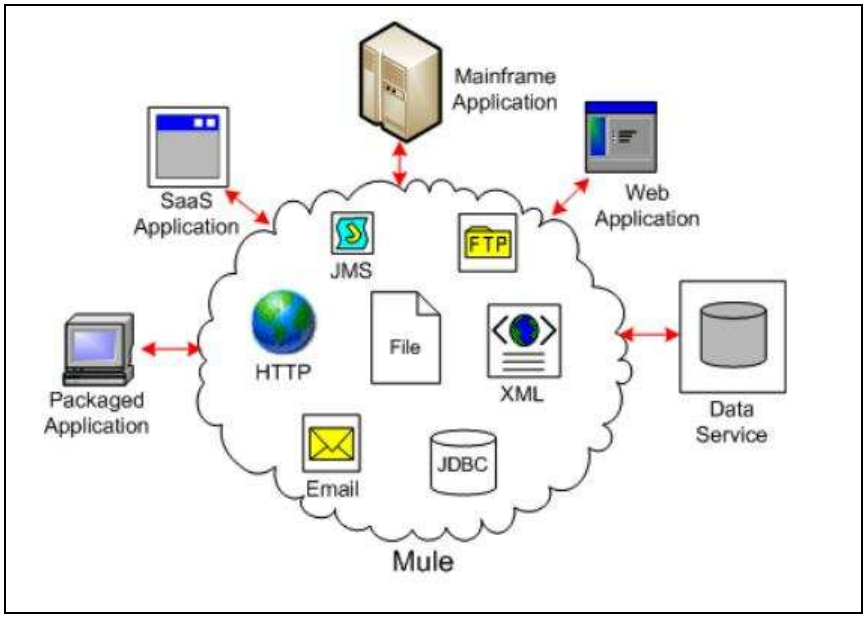
3.4. Enterprise Service Bus Mule

Mule é um *framework* baseado na JMS (*Java Message Service*). Java Message Service é um padrão de mensagem para aplicações baseadas na plataforma Java, que permite a criação, envio, recebimento e leitura de mensagens em um sistema fracamente acoplado. (SUN, 2008).

Mule permite conectar aplicações facilmente e de maneira rápida, habilitando a troca de dados entre essas aplicações, utiliza o paradigma SOA (Arquitetura Orientada a Serviço), habilitando uma fácil integração entre sistemas, independente das tecnologias usadas nas aplicações. Mule utiliza os seguintes padrões de desenvolvimento para fornecer essa integração, JMS (*Java Message Service*), *WebServices,* JDBC (*Java Database Connectivity*), HTTP, XML (MuleSource, 2008).

Mule provê uma variedade de funcionalidades para permitir a interoperabilidade entre os serviços. Na Figura 7 apresenta-se uma visão geral do ESB Mule, que utilizando alguns dos padrões apresentados no capitulo 2 demonstra a integração de vários sistemas.

Figura 7 – Visão geral ESB Mule



Fonte: MuleSource 2008

3.5. Considerações Finais

O objetivo dessa seção foi apresentar algumas tecnologias disponíveis como, *framework Axis,* a plataforma SAP *NetWeaver* e o ESB Mule para implantação de um ambiente SOA .

4. ESTUDO DE CASO

4.1. Considerações Iniciais

Neste capítulo apresenta-se a especificação do estudo de caso proposto por meio de alguns diagramas da UML (*Unified Modeling Language*), ou seja, diagramas de casos de uso, de classes e de componentes. Na Seção 4.2 descreve-se o estudo de caso conduzido. Na Seção 4.3 apresenta-se uma visão geral sobre cada diagrama da UML utilizado e ilustram-se os diagramas elaborados durante o estudo de caso. Na Seção 4.4 apresenta-se o desenvolvimento de uma arquitetura SOA, visando a implementação do estudo de caso.

4.2. Descrição do Estudo de Caso

Com o objetivo de exemplificar a utilização dos conceitos de *SOA* (Arquitetura Orientada a Serviços), o estudo de caso conduzido neste trabalho tem como objetivo analisar a flexibilidade sugerida por SOA em uma regra de negócio utilizada em um ambiente de Engenharia de Produto. A regra de negócio a ser analisada é a criação de Materiais de Engenharia, sendo que, esse material pode ser criado por um Engenheiro de Produto ou por um Engenheiro de Processo, utilizando aplicações diferentes.

Material de Engenharia é um termo utilizado para descrever um produto a ser desenvolvido, desde o projeto à produção. Sendo representado por um Modelo 3D e tendo seus dados cadastrais armazenados em um PDM (*Product Data Management*).

Para atender o processo de desenvolvimento de um material desde sua concepção 3D até sua inclusão em um PDM será utilizado o conceito de WebServices que tem se tornado um padrão na implantação de SOA (COLAN, 2004), ou seja, será criado um Modelo 3D com o auxílio de uma ferramenta CAD (*Computer-Aided Design*), o ProEngineer (PTC, 2008), e será criado um Material no PDM para representar os dados cadastrais desse modelo.

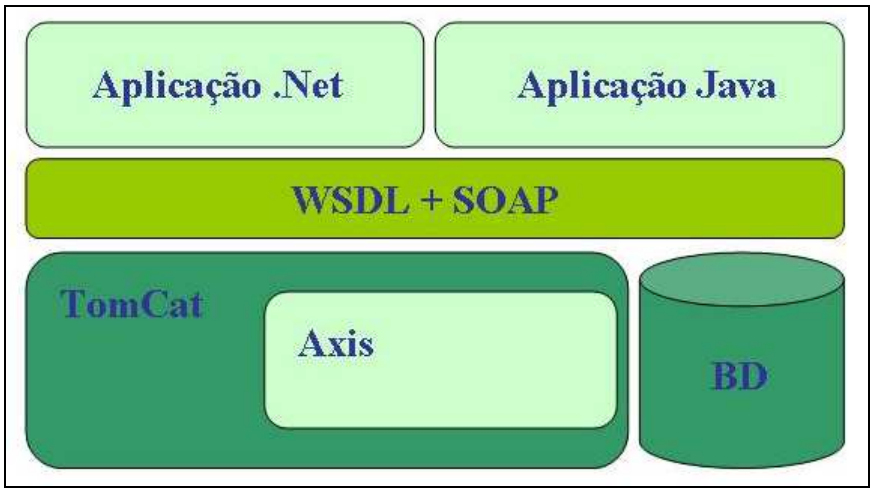
ProEngineer é uma ferramenta CAD (*Computer-Aided Design*) para modelagem 3D, sendo uma das ferramentas mais utilizadas para essa finalidade na área da mecânica (PTC, 2008).

O objetivo da aplicação desenvolvida no estudo de caso utilizando o desenvolvimento baseado em SOA é analisar os conceitos de criação de serviços e interoperabilidade

A arquitetura proposta para a aplicação desenvolvida no estudo de caso está apresentada na Figura 8:

- *WebServices* executando no *framework Axis* que está para prover o serviço de criação do material de engenharia;
- Uma aplicação em *dotNet* para consumir o *WebServices,* que será utilizada pelo engenheiro de processo;
- Uma aplicação em Java para consumir o *WebServices*, que será utilizada pelo engenheiro de produto;
- Uma tabela solitária criada no MYSQL (MySql AB, 2008) para armazenar os dados cadastrais do item no SGBD para representar o modelo 3D.

Figura 8 – Arquitetura de Estudo de Caso



4.3.Modelos Conceituais

Com o surgimento das linguagens Orientadas a Objetos, também surgiram os métodos de análise para projetos de *software* na década de 90. Os métodos mais utilizados eram Booch de Grady Booch, OOSE (*Object Oriented Software Engineer*) de Ivar Jacobson e OMT (*Object Modeling Thechinique*). Os autores desses métodos se uniram para unificar tais métodos, mesclando o que cada método tinha de melhor, assim, deram origem a UML (*Unified Modeling Language*) (BOOCH; JACOBSON; RUMBAUCH, 2005).

A UML é uma linguagem padrão para estruturar o desenvolvimento de projeto de software. É utilizada para elaborar, especificar e documentar sistemas orientados a objeto (BOOCH; JACOBSON; RUMBAUCH, 2005). Atualmente a UML é mantida pela OMG (*Object Management Group*) (OMG, 2005).

Os diagramas da UML utilizados nesse estudo de caso são:

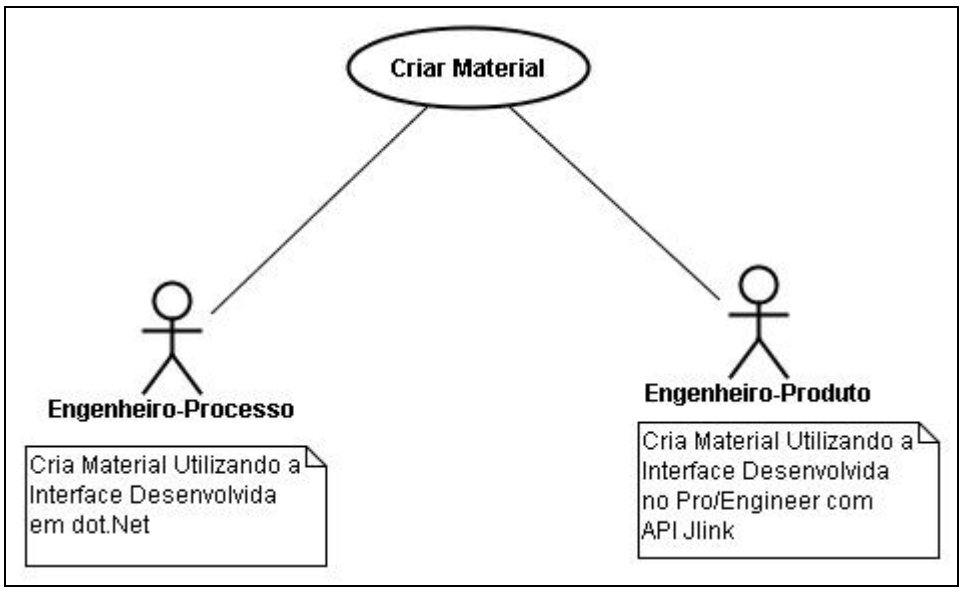
- **Diagrama de Casos de Uso**: Modelam um sistema do ponto de vista dos usuários, ou seja, exemplificam como cada ator irá interagir com os processos do sistema
- **Diagrama de Classes**: classe é uma descrição de um conjunto de objetos com os mesmos atributos, relacionamentos, operações e semântica. Um diagrama de classes é a modelagem de um conjunto de classe que compõe um sistema.
- **Diagrama de Componentes**: são utilizados para modelar os aspectos físicos de um sistema, por exemplo, aspectos de um banco de dados, de uma API ou de um código fonte de um aplicativo.

4.3.1. Diagrama de Caso de Uso

Na Figura 9 ilustra-se o diagrama de casos de uso do estudo de caso, apresentando o principal serviço que deve ser criado por um Engenheiro de Produto ou por um Engenheiro de Processo e consumido em duas diferentes aplicações: uma implementada em dotNet e outra em Java (API Jlink). Esta última é utilizada como *plug-in* na ferramenta Pro-Engineer.

J-Link é uma API escrita em linguagem Java que fornece funções para interações com o sotfware ProEngineer. Com essa API é possível interagir por meio de eventos e funcionalidades fornecendo uma funcionalidade extra chamada de *Auxiliary Application* (PTC, 2008).

Figura 9 – Diagrama do Caso de Uso

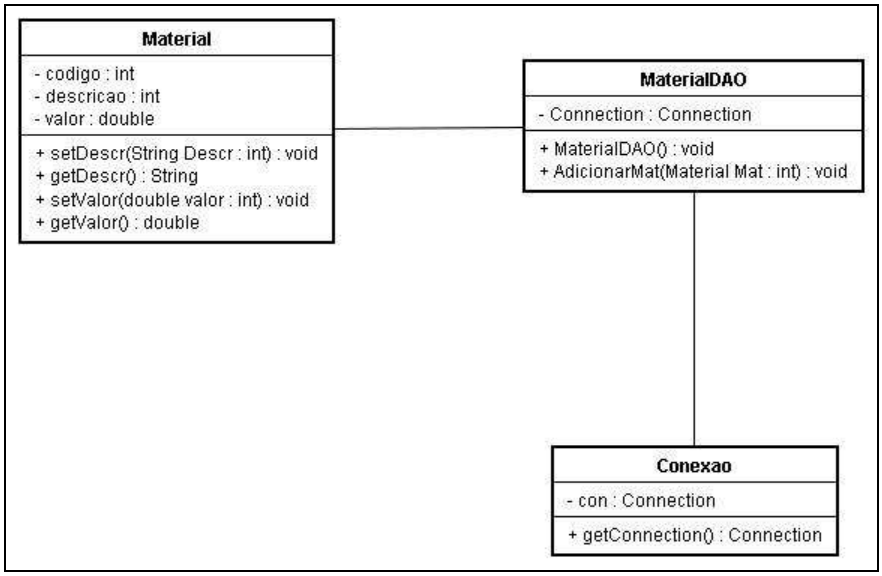


Para descrição do Caso de Uso, ver apêndice A.

4.3.2. Diagrama de Classes

A aplicação desenvolvida é composta por três classes: `Material`, `MaterialDAO` e `Conexão`, conforme apresentado na Figura 10. A primeira classe representa o material a ser criado, a segunda classe é responsável por persistir os dados do material na base de dados e a terceira classe realiza a conexão com o sistema gerenciador de banco de dados MYSQL.

Figura 10 – Diagrama de Classes



Na Figura 11 é apresentado o script para a criação da tabela Material, resultante do mapeamento do diagrama de classes para banco de dados relacional, ou seja, apenas a classe Material cujos objetos devem ser persistidos foi mapeada como uma tabela no banco de dados.

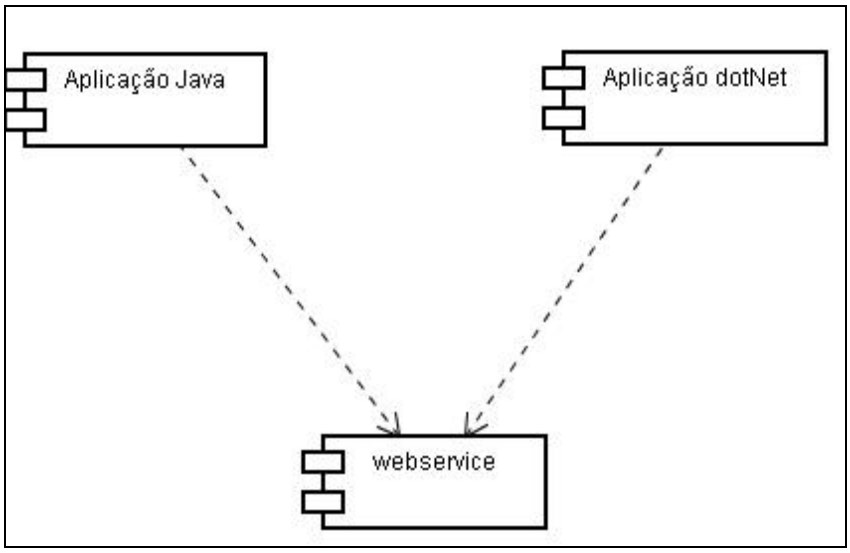
Figura 11 – Script para criar tabela Material

```
CREATE TABLE cadastro.Material (
  ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Descricao VARCHAR(45) NOT NULL,
  Valor FLOAT NOT NULL,
  PRIMARY KEY (ID)
)
```

4.3.3. Diagrama de Componentes

Na Figura 12 é apresentado o diagrama de componentes referente às aplicações desenvolvidas, escritas em linguagens de programação distintas, ou seja, em Java e em dotNet que farão uso de um *WebServices* para garantir a interoperabilidade no sistema, conforme relacionamentos de dependência explicitados na figura.

Figura 12 – Diagrama de Componentes



4.4. Implementação

A implementação de um *WebServices* pode ser feita em diversas linguagens. No estudo de caso deste trabalho, o *WebServices* foi implementado em Java e foi disponibilizado por meio do *framework Axis,* que gera WSDL necessário para que o *WebServices* possa ser disponibilizado e consumido por outras aplicações.

4.4.1. Criando e publicando um WebServices em Java

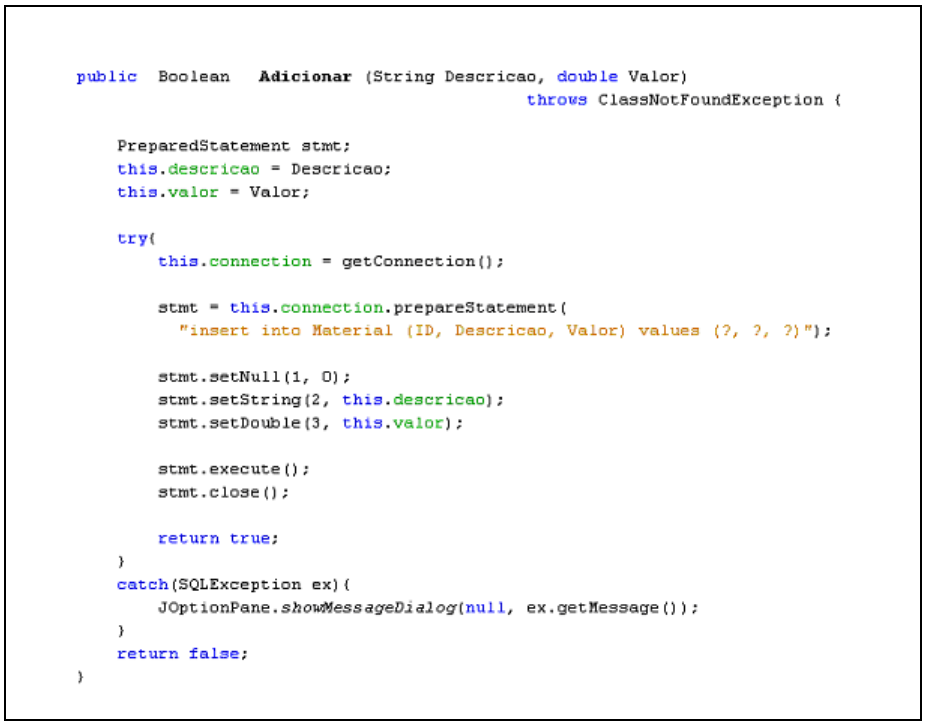
Na Figura 13 tem-se a classe de conexão com o Banco de dados. O método *getConnection* carrega a classe *JdbcOdbcDriver* e retorna um objeto de conexão *Connection,* conectando-se ao banco com usuário “root” e senha “admin”.

Figura 13 – Classe de Conexão

```
package MODEL;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class Conexao {


    public static Connection getConnection() throws SQLException,
                                                    ClassNotFoundException {
      try {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         return DriverManager.getConnection("jdbc:odbc:myodbc","root","admin");

      } catch (SQLException e) {

          throw new SQLException(e.getMessage());
      }

    }

}
```

Na Figura 14 apresenta-se o código do método de inserção da classe MaterialDao. O método recebe os valores `Descricao` e `Valor`, estabelece a conexão com o banco de dados e utiliza o objeto da classe `PreparedStatement` para realizar a inserção.

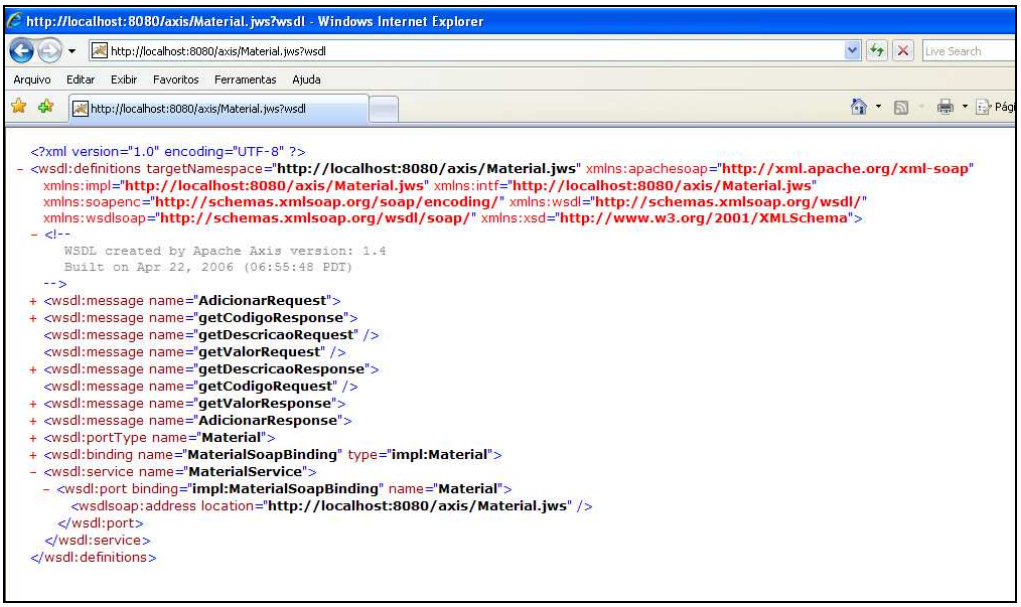
Figura 14 – Método de inserção da classe MaterialDao.



```
public  Boolean   Adicionar (String Descricao, double Valor)
                                            throws ClassNotFoundException {

    PreparedStatement stmt;
    this.descricao = Descricao;
    this.valor = Valor;

    try{
        this.connection = getConnection();

        stmt = this.connection.prepareStatement(
          "insert into Material (ID, Descricao, Valor) values (?, ?, ?)");

        stmt.setNull(1, 0);
        stmt.setString(2, this.descricao);
        stmt.setDouble(3, this.valor);

        stmt.execute();
        stmt.close();

        return true;
    }
    catch(SQLException ex){
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
    return false;
}
```

Na Figura 15 exibe-se o arquivo WSDL publicado pelo *framework Axis,* indicando o local onde o serviço está disponível, no exemplo, está disponível em "http://localhost:8080/axis/Material.jws". O campo *<wsdl:message name>* exibe quais métodos o serviço disponibiliza e o campo *<wsdl:service name>* exibe o nome do serviço.

Figura 15 - WSDL publicado através do *framework Axis*

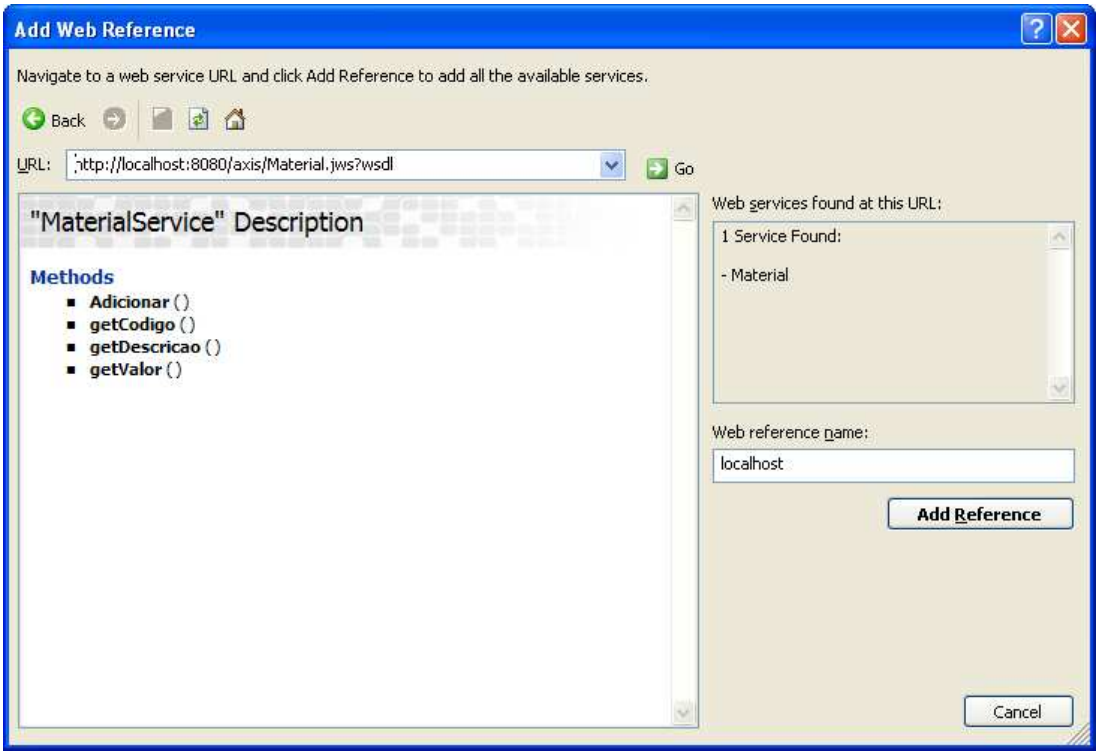


4.4.2. Consumindo o WebServices em dotNet

Depois de criado e disponibilizado o *WebServices*, é possível utilizá-lo em outra aplicação, mesmo sem conhecer a sua implementação. Para isso, é necessário utilizar somente o arquivo *WSDL* gerado para consumir o *WebServices.*

Utilizando o Visual Studio, foi criado um projeto em `vb.net` para consumir o WebServices. A opção *Add Web Reference* possibilita localizar um arquivo WSDL publicado por um *framework*, nesse exemplo o Axis. Na Figura 16 apresenta-se o exemplo da opção *Add Web Reference* localizando o WebServices na url "http://localhost:8080/axis/Material.jws" e os métodos disponibilizados. Nesse estudo de caso, será consumido o método `Adicionar`.

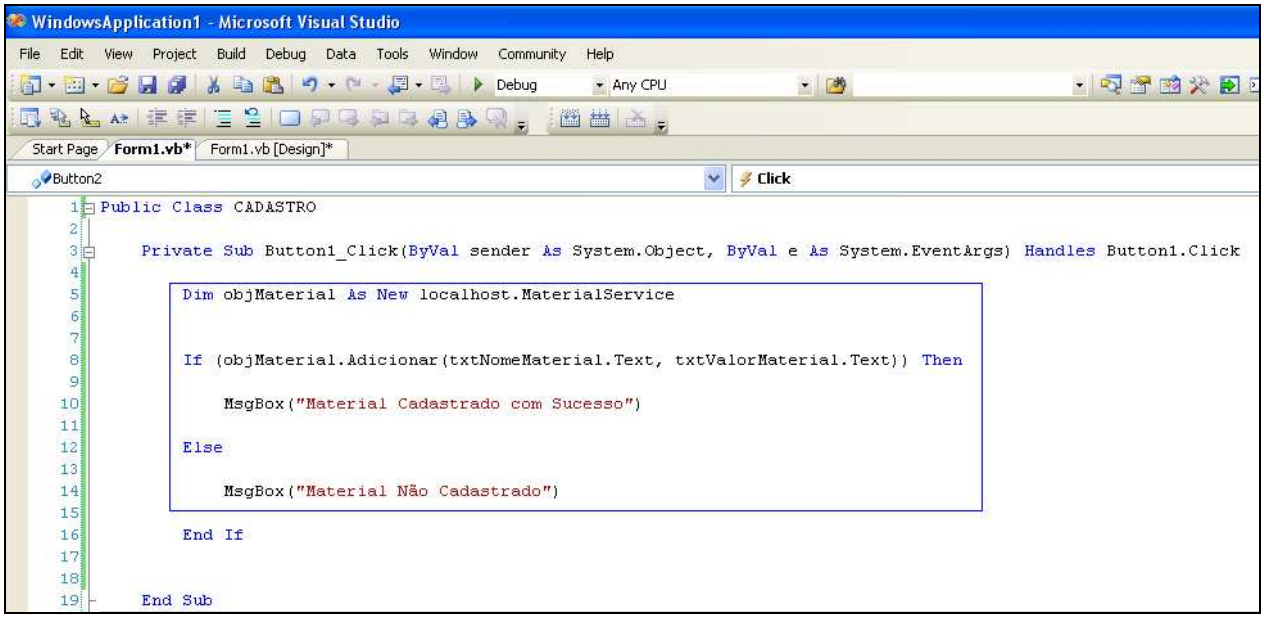
Figura 16 – WSDL Localizado utilizando dotNet



Na aplicação em *dotNet* foi codificado o método `Button_Click()` no formulário Cadastro (Figura 17), referenciando o método **Adicionar** disponibilizado pelo *WebServices* para criar um material no banco de dados.

A declaração "`Dim objMaterial as New localhost.MaterialService`" cria um objeto para referenciar o WebServices. A declaração "`If objMaterial.Adicionar(...) Then`" executa o método disponibilizado pelo WebServices, exibindo a mensagem de sucesso.

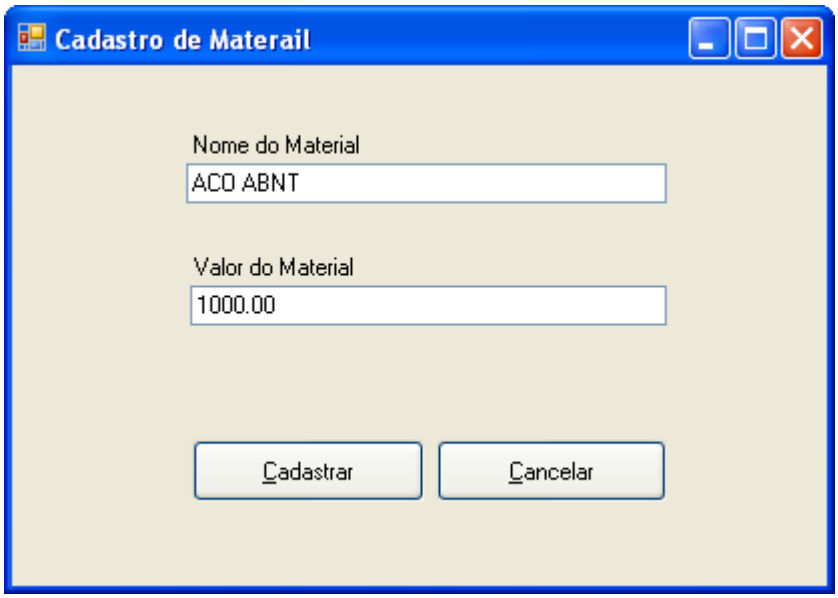
Figura 17 – Código de Conexão com o WebServices



4.4.3. Utilizando a Aplicação em dotNet

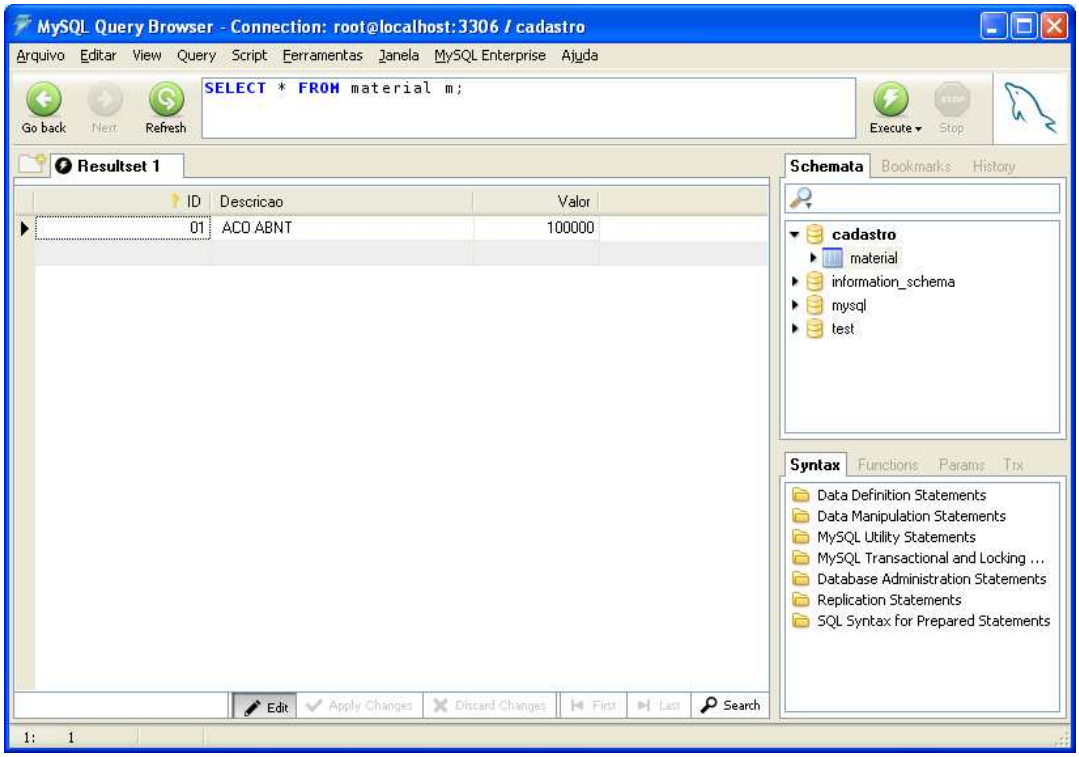
Como descrito na Seção 4.2, uma das aplicações deve ser executada em dotNet, mostrando a criação de um material pelo engenheiro de processo, conforme apresentado na Figura 18.

Figura 18 – Tela de Cadastro de Material



Na Figura 19 é exibida a consulta SQL para localizar na tabela Material o item criado utilizando a aplicação dotNet.

Figura 19 – Consulta na Tabela Material



4.4.4. Consumindo o WebServices no ProEngineer

Para utilizar o WebServices no Proengineer foi construída uma aplicação utilizando a API Jlink disponibilizada no arquivo `pfc.jar`. Na Figura 20 é exibida a classe `TesteMaterial` que utiliza a API `pfc.jar` para implementar uma novo evento que será disponibilizado no Proengineer.

O método `start` será executado sempre que o evento for disparado e um novo objeto da classe `Cadastro` será inicializado, o qual se encarrega de efetuar o cadastro do material consumindo o WebServices.

Figura 20 – Classe TesteMaterial

```
import com.ptc.pfc.pfcGlobal.*;
import com.ptc.pfc.pfcSession.*;
import com.ptc.cipjava.*;
import javax.swing.*;

public class TesteMaterial {

    public static void start()
    {
       try{
            Session session;
            session = pfcGlobal.GetProESession();
            session.ChangeDirectory("C:\\workspace");

            new Cadastro().setVisible(true);

      }catch (jxthrowable x){
           JOptionPane.showMessageDialog(null, x);
       }
    }

    public static void stop(){

    }
}
```

Na Figura 21 tem-sse o método `btncadstrarActionPerformed` da classe `Cadastro` que implementa a interface `ActionEvent`. Esse método localiza o WebServices utilizando a String "*http://localhost:8080/axis/Material.jws*". Para criar o objeto `call` é necessário importar os pacotes `org.apache.axis.client.Call` e `org.apache.axis.client.Service` disponibilizados pelo *framework* Axis.

Através da chamada `call.setOperationName("Adicionar")`, o WebServices disponibilizado é executado.

Figura 21 – Método de Cadastrar Material.

```
private void btncadastrarActionPerformed(java.awt.event.ActionEvent evt) {

    String local = "http://localhost:8080/axis/Material.jws";
    String Descr = txtDescricao.getText();
    double valor = Double.valueOf(txtvalor.getText());


    try {
        // Criando e configurando o servico
        Call call = (Call) new Service().createCall();

        // Configurando o endereco.
        call.setTargetEndpointAddress(local);

        // Marcando o metodo a ser chamado.
        call.setOperationName("Adicionar");

        Object[] param = new Object[]{new String(Descr), new Double(valor)};

        if ((Boolean)call.invoke(param))
                JOptionPane.showMessageDialog(null, "Cadastrado com Sucesso");
        else
                JOptionPane.showMessageDialog(null, "Nao Cadastrado");

    }catch(Exception ex){
        JOptionPane.showMessageDialog(null, ex.getMessage() + "AQUI");
    }

}
```

No apêndice B encontra-se as seqüências de execução da aplicação JLink para consumir o WebServices disponibilizado.

4.4.5. Avaliação do Estudo de Caso

O principal objetivo do estudo de caso conduzido é utilizar os conceitos de SOA para desenvolver uma aplicação, exemplificando com um exemplo prático os conceitos envolvidos, conforme apresentado nos capítulo anteriores desta monografia. No contexto SOA, o que mais tem se destacado é a utilização de *WebServices* para fornecer serviços, e nesse contexto, o uso de *WebServices* foi eficaz, pois a sua utilização garantiu que a criação do material no banco de dados fosse efetuada.

A interoperabilidade foi alcançada, pois o serviço foi desenvolvido em Java, disponibilizado pelo *framework* Axis e foi consumido pela aplicação em *dotNet* sem nenhuma perda de informação quanto aos dados enviados pelas mensagens SOAP.

Da mesma forma que se garante a independência de plataforma utilizando-se *WebServices,* também é possível alcançar o acoplamento fraco, pois qualquer mudança pode ser efetuada de maneira rápida sem comprometer o restante das aplicações. Esse estudo de

caso explicou de maneira simples os conceitos de SOA, utilizando a tecnologia de WebServices. O serviço implementado é classificado como um serviço básico da SOA fundamental.

A partir do estudo de caso realizado, observou-se ganho quanto ao tempo de desenvolvimento, pois não foi necessário codificar vários sistemas que efetuassem a mesma operação, sendo necessário somente disponibilizar um serviço utilizando o *framework Axis*, podendo ser consumido por sistemas distintos.

5. CONCLUSÃO

SOA (Arquitetura Orientada a Serviços) tem sido considerada como um conceito para desenvolver software, conforme afirma o Instituto Gartner (McCoy, 2003).

Nesse trabalho foram descritos os conceitos principais de SOA, por exemplo, disponibilizar processo de software que possam ser acessados como serviços, ou seja, disponibilizar serviços que possam ser acessados de forma independente. Nesse contexto, SOA torna-se fácil de ser implantada com a utilização de ferramentas de auxílio como o SAP *NetWeaver,* por exemplo. A proposta de SOA também se aplica para resolver vários aspectos como, diminuir custos de desenvolvimento, diminuir tempo nas modificações, melhorar a adaptação de sistemas legados e a utilização de ferramentas BPM (*Business Process Modeling)* para análise de negócio para composição do cenário SOA.

Para realização desse trabalho houve dificuldade na revisão bibliográfica, principalmente relacionadas a publicações nacionais, o que dificultou uma melhor definição sobre SOA, visto que, muitas definições foram encontradas na literatura.

5.1. Contribuições

Esse trabalho serviu como iniciativa para o estudo de um conceito de desenvolvimento, SOA, que tem sido abordado por grandes empresas de tecnologias como, SUN, Oracle, Microsoft e SAP. Tal tecnologia mostrou-se eficaz no estudo de caso proposto em que a utilização das tecnologias disponíveis tornou o trabalho simples de ser executado.

A proposta do trabalho de exemplificar SOA da teoria à prática foi concluída, exemplificando os principais conceitos e tecnologias. Entretanto, para uma melhor análise, características como segurança, linguagens de modelagem de processos e projetos *opensource* poderiam ser abordados em trabalhos futuros como descrito na Seção 5.3.

5.2. Limitações

Conforme mencionado na seção anterior, a falta de publicações nacionais foi um fator limitante. Os testes foram efetuados em um único computador, assim não foi possível mensurar o tempo de acesso e velocidade caso os testes fossem executados em um ambiente cliente servidor.

5.3. Trabalhos Futuros

Para dar continuidade ao trabalho realizado nesta monografia, a seguir são elencados sugestões de trabalhos futuros:

- Um estudo de caso mais detalhados referentes aos padrões de segurança XML *Signature* e XML *Encryption*;
- Análise de uma linguagem de modelagem de processos, BPM, exemplo, BPEL. Pois essas linguagens fornecem uma visão de como as empresas definem seus processos.

REFERÊNCIAS

ALLEN, Paul. Service Orientation: Winning Strategies and Best Practices. Cambridge, UK. Cambridge University Press, 2006.

ALVES, Artur. Arquitetura Orientada a Serviços. JavaPT06 5ª Edição. Sun Microsystems Portugal. Disponível em: <http://pt.sun.com/sunnews/events/2006/javapt> Acesso em 25/092008.

Apache WSS4J. Disponível em <http://ws.apache.org/wss4j/> Acesso em 20/10/2008.

BOOCH, Grady; JACOBSON, Ivar; RUMBAUCH, James. UML Guia do Usuário. Rio de Janeiro, Elsevier, 2005.

CARDOSO, Caíque. UML NA PRÁTICA: Do Problema ao Sistema. Rio de Janeiro, Ciência Moderna, 2003.

COLAN, Mark. Service-Oriented Architecture expands the vision of WebServicess, Part 1. Disponível em: < http://www.ibm.com/developerworks/library/ws-soaintro.html> Acesso em: 18/01/2008.

DETALHES TÉCNICOS DO SAP NetWeaver: Arquitetura empresarial orientada a serviços Disponível em <http://www.sap.com/brazil/platform/netweaver/technicaldetails/esa.epx> Acesso em: 21/10/2008.

ENTERPRISE SOA DEVELOPMENT - Handbook End-to-End Guide for Enterprise SOA Development, SAP AG. 2008

ERL, Thomas. Editorial. Disponível em: <http://www.savoirfaire.com.br/default.asp> Acesso em: 14/03/2008.

ERL, Thomas. Service-Oriented Architecture: Concepts, Thecnology, and Design: UpperSaddle River, NJ, Prentice Hall, 2005.

Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation 16 August 2006. Disponível em: <http://www.w3.org/TR/2006/REC-xml-20060816/> Acesso em 11/10/2008.

GEE, Clive. Elements of Service-Oriented Architecture analysis and Design. Disponível em: <http://www-128.ibm.com/developerworks/library/ws-soad1/> Acesso em: 18/01/2008.

GONÇALVES, Edson. Desenvolvendo Aplicações Corporativas com NetBeans IDE 5.5: Rio de Janeiro, Ciência Moderna, 2007.

Hypertext Transfer Protocol HTTP/1.1. Disponivel em < http://www.w3.org/Protocols/HTTP/1.1/rfc2616bis/draft-lafon-rfc2616bis-03.html> Acesso em: 20/10/2008.

JONES, Esteve. Enterprise SOA adoption Strategies, Using SOA to deliver IT to the business: USA, C4Media Inc. 2006.

JOSUTTIS, Nicolai M. SOA na Prática, A Arte de modelar Sistemas Distribuídos: Rio de Janeiro, Alta Books, 2008.

Macoratti, José Carlos. Introdução a XML. Disponível em: <http://www.macoratti.net/xml.htm> Acesso em: 20/10/2008.

McCOY, David W. Service-Oriented Architecture: Mainstream Straight Ahead. Disponível em: <http://www.gartner.com/pages/story.php.id.3586.s.8.jsp> Acesso em 11/04/2008.

Mule Documentation and Resource Center. Disponível em <http://mulesource.org/pages/viewpage.action?pageId=13729892>. Acesso em 20/10/2008.

NETO, Agotinho Campos. WebServicess em Java com Axis - Teoria e Prática. Disponível em: <http://www.guj.com.br> Acesso em: 11/09/2008.

OMG Object Management Group. Disponível em <http://www.rational.com> Acesso em: 20/10/2008.

Parametric Technology Corporation - ProEnginneer. Disponível em <http://www.ptc.com/> Acesso em: 20/04/2008.

PRESSMAN, Roger S. Engenharia de Software. São Paulo, Makron Books, 1995.

ROSENBERG, Jothy, REMY, David. Securing Web Services with WS-Security. USA. Sams Publishing, 2004.

SCHAICH, Benny. REAL WORLD COMPOSITES - WORKING WITH ENTERPRISE SERVICES, SAP AG. 2008.

SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation 27 April 2007. Disponível em: <http://www.w3.org/TR/2007/REC-soap12-part0-20070427/> Acesso em: 25/09/2008.

TAKAHASHI, Kentaro; ACQUAVIVA, Claudio; D'AURIA, Paschoal. Artigo: Cenário e Oportunidade - MundoJava nº 26. Rio de Janeiro, Editora Mundo, 2008.

TINDALL, Paul. Desenvolvendo Aplicações Corporativas. Rio de Janeiro, Ciência Moderna, 2000.

VARCHAVSKY, Márcio. Segurança em Web Service com WSS4J. MundoJava nº 28. Rio de Janeiro, Editora Mundo, 2008.

Web Services – Axis. Axis User's Guide. Disponível em: <http://ws.apache.org/axis/java/user-guide.html#WhatIsAxis> Acesso em 21/10/2008

WebServicess Description Language (WSDL) Version 2.0 Part 1: Core Language W3C Recommendation 26 June 2007. Disponível em: <http://www.w3.org/TR/2007/REC-wsdl20-20070626/> Acesso em: 25/09/2008.

JDBC - Java Database Connectivity (JDBC). Disponível em <http://java.sun.com/javase/technologies/database/> Acesso em: 04/11/2008.

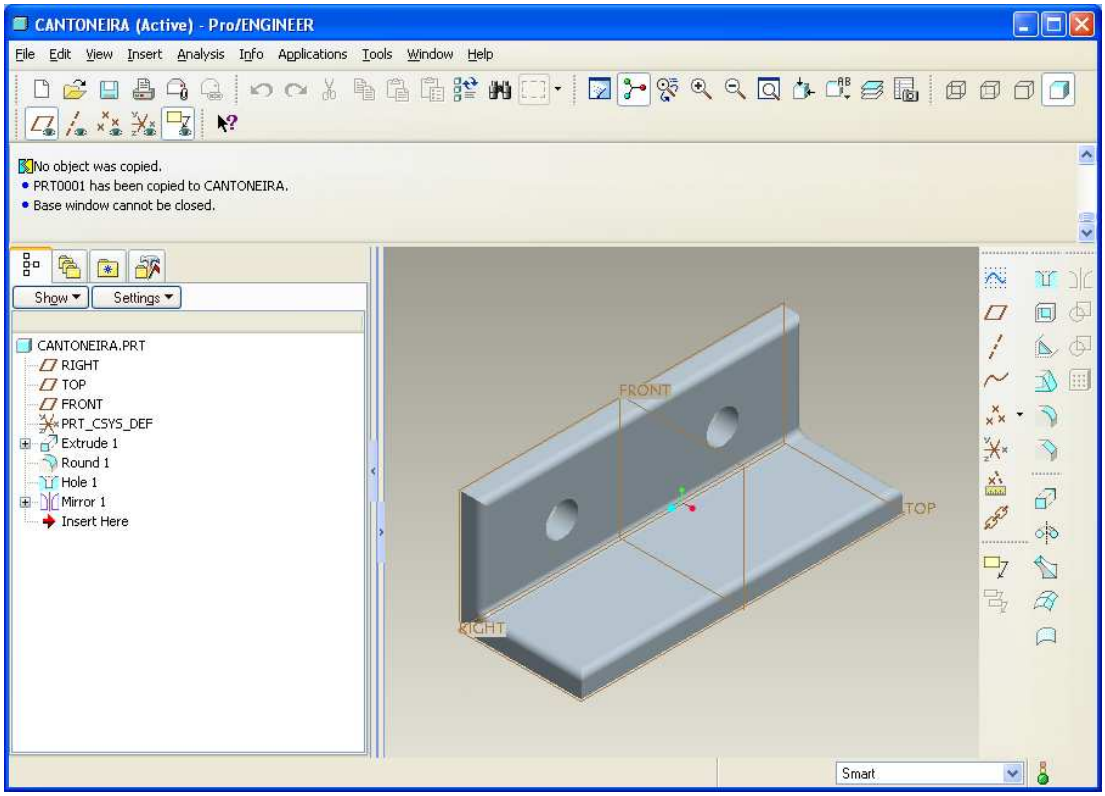
Java Message Service (JMS). Disponível em <http://java.sun.com/products/jms/> Acesso em: 04/11/2008.

APÊNDICE A – Descrição do Caso de Uso.

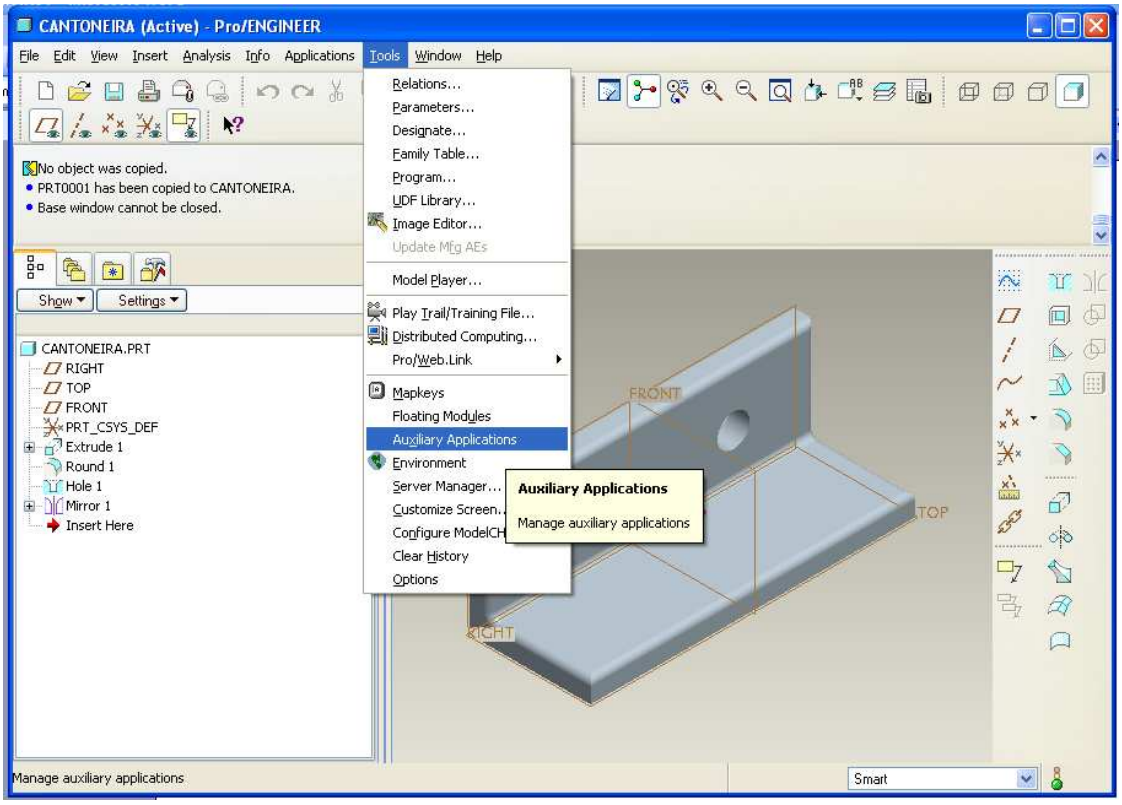
Caso de Uso Criar Material **Atores:** Engenheiro de Processo e Engenheiro de Produto. **Descrição:** Este caso de uso tem como criar um Material de Engenharia.
Curso Normal 1. Usuário acessa aplicação Java ou dotNet para Cadastrar um novo material no Sistema 2. Usuário informa descrição e valor do material 3. Sistema cria um material e emite a mensagem "Material cadastrado com sucesso"
Curso Alternativo 1 1 Usuário cancela o cadastro do material

APÊNDICE B – Execução do WebServices no Proengineer.

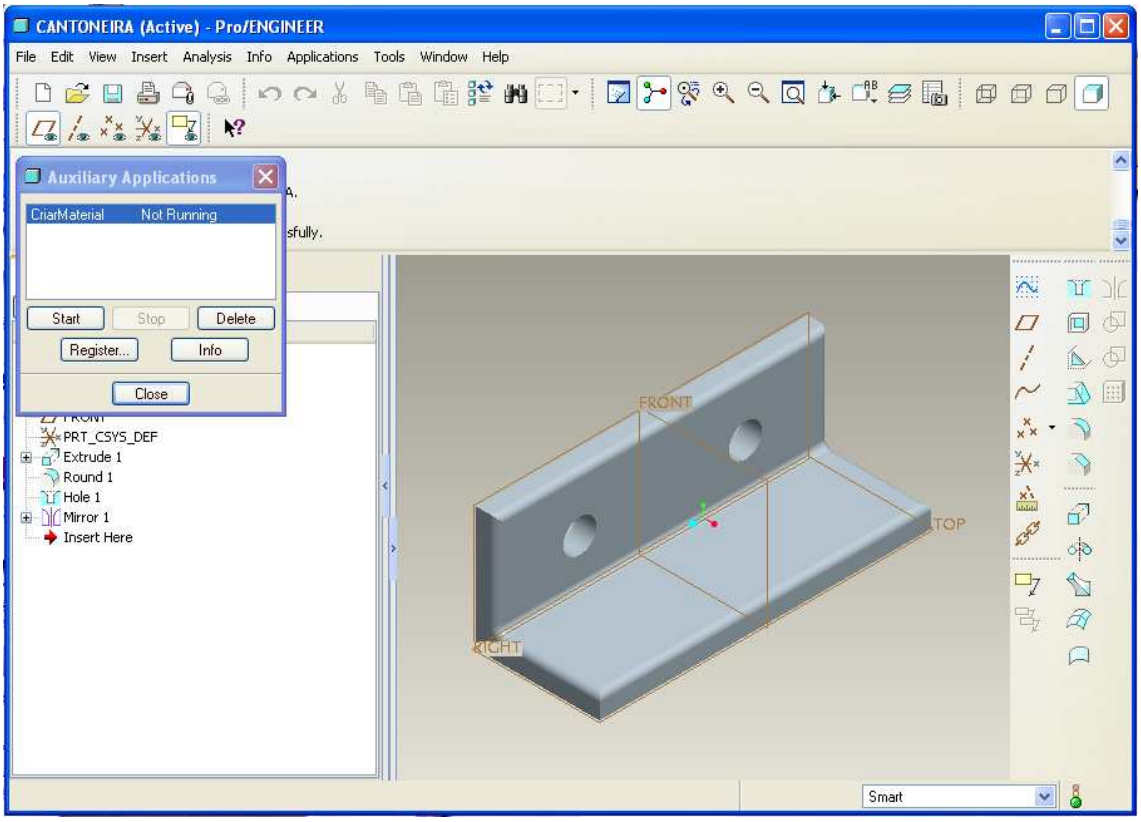
Criando um Modelo 3D.



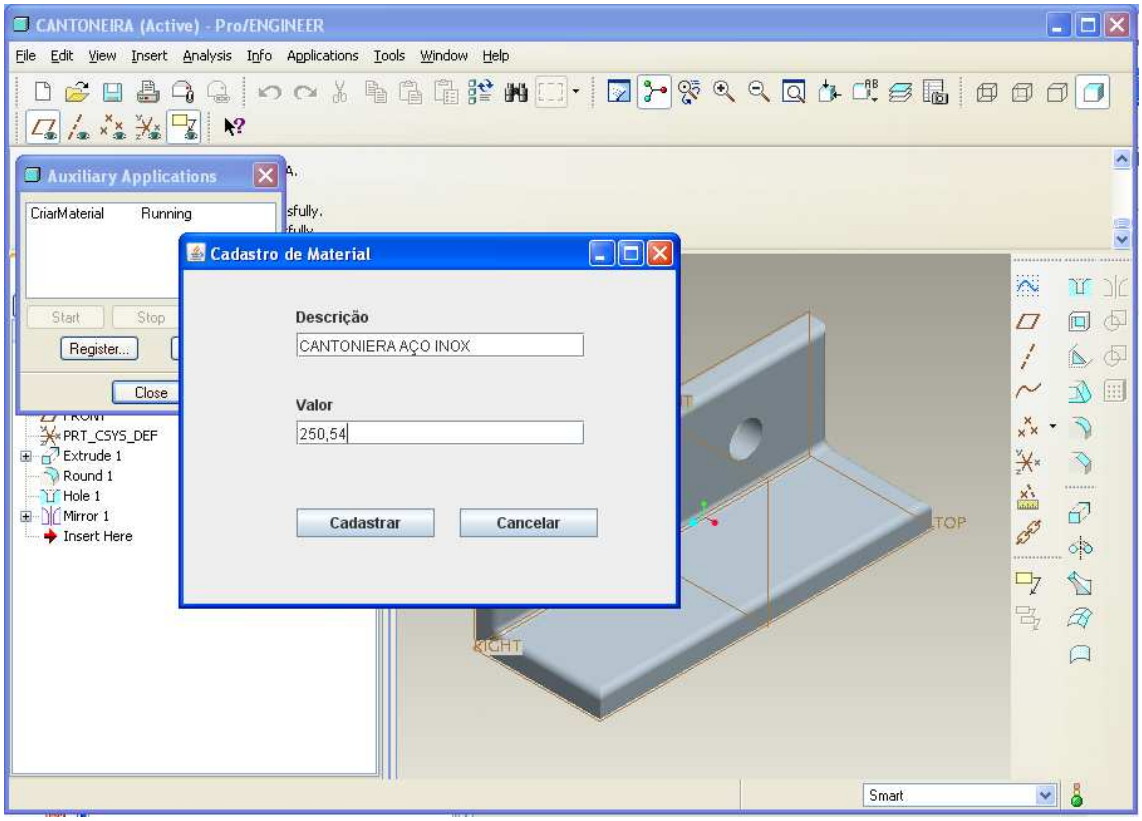
Habilitando a aplicação auxiliar.



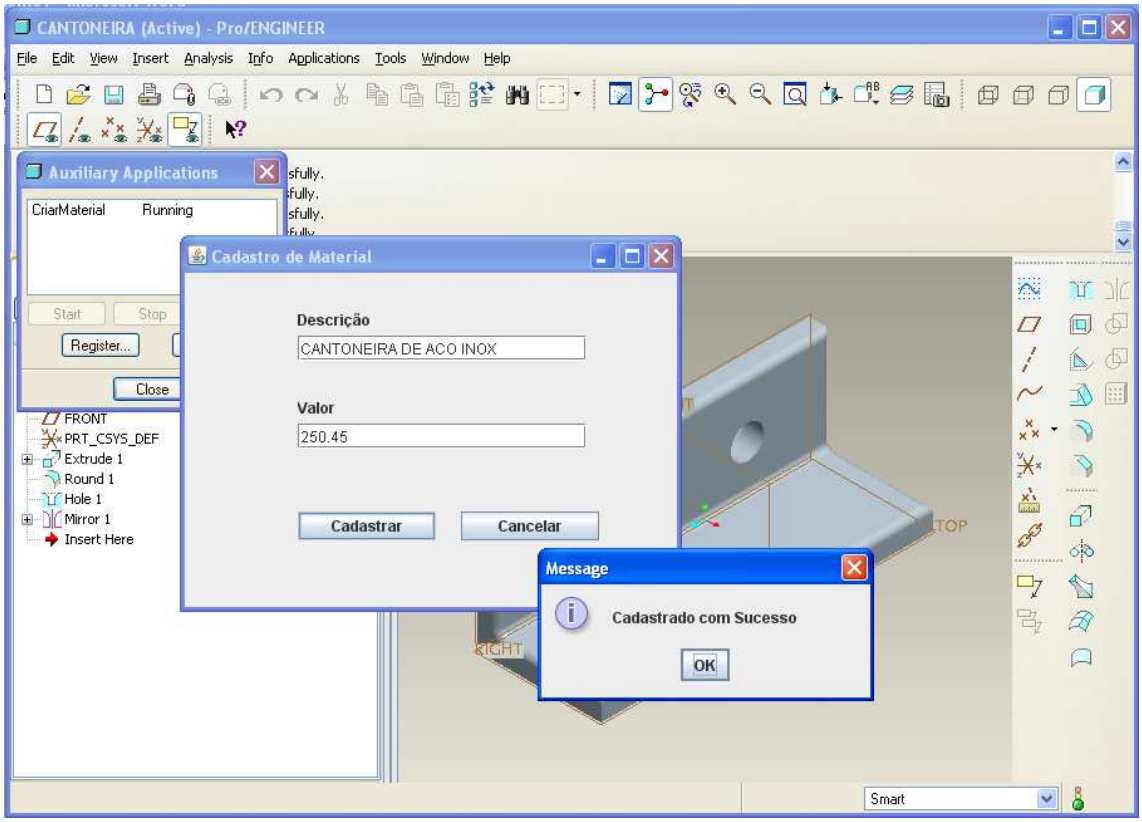
Executando a aplicação



Criando um material



Material criado com sucesso



Verificando no banco de dados o item criado no Proengineer