UNIVERSIDADE DE SÃO PAULO

Escola de Artes, Ciências e Humanidades

Guilherme Jorge Aragão da Cruz

A Arquitetura Orientada a Serviços (SOA) Para Empresas: Estudo de Caso em um Sistema de Informação Voltado a um Processo de Negócio

São Paulo

Junho de 2015

Universidade de São Paulo

Escola de Artes, Ciências e Humanidades

Guilherme Jorge Aragão da Cruz

A Arquitetura Orientada a Serviços (SOA) Para Empresas: Estudo de Caso em um Sistema de Informação Voltado a um Processo de Negócio

Monografia apresentada à Escola de Artes, Ciências e Humanidades, da Universidade de São Paulo, como parte dos requisitos exigidos na disciplina ACH 2018 – Projeto Supervisionado ou de Graduação II, para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Flávio Luiz Coutinho

Modalidade: TCC Longo (1 ano) – individual

São Paulo

Junho de 2015

Universidade de São Paulo
Escola de Artes, Ciências e Humanidades

Guilherme Jorge Aragão da Cruz

A Arquitetura Orientada a Serviços (SOA) Para Empresas: Estudo de Caso em um Sistema de Informação Voltado a um Processo de Negócio

Monografia apresentada à Escola de Artes, Ciências e Humanidades, da Universidade de São Paulo, como parte dos requisitos exigidos na disciplina ACH 2018 – Projeto Supervisionado ou de Graduação II, para obtenção do título de Bacharel em Sistemas de Informação.

Modalidade: TCC Longo (1 ano) – individual

Data de Aprovação:

22/07/2015

Banca Examinadora:

_____

Marcelo Fantinato
EACH-USP

São Paulo

Junho de 2015

Agradecimentos

À minha amiga e mestranda Maria Andrade por todo suporte oferecido nos momentos finais deste projeto.

Aos meus amigos e familiares pela paciência, apoio e compreensão nos mais diversos momentos, bem como aos amigos de curso por toda ajuda e tempo oferecido.

Aos colegas de trabalho pelo apoio e acompanhamento de toda minha trajetória acadêmica ao longo destes anos.

Ao meu orientador, professor Flávio Coutinho, pela aceitação do tema proposto e também por todo apoio e ajuda nesta e em outras disciplinas.

Glossário

BPEL: Business Process Execution Language - a Linguagem de Execução de Processos de Negócio

CSV: Comma-separated values - arquivos com valores diversos separados por vírgulas.

DAO: *Data Access Object* - objeto de acesso a dados, padrão para persistência de dados.

ERP: *Enterprise Resource Planning* - sistema de informação que integra diversos processos e dados de uma organização.

ESB: *Enterprise Service Bus* - o barramento de serviços, que permite acoplar diferentes tecnologias.

.NET: dotNet - iniciativa da Microsoft para uma plataforma única para desenvolvimento de *software*.

OSIMM: *Open Group Service Integration Maturity* Model - o Modelo de Maturidade de Integração de Serviços do *The Open Group*.

PoC: *Proof of Concept* - modelo prático para provar conceitos teóricos.

POO: Programação Orientada a Objetos - paradigma para análise, projeto e desenvolvimento de sistemas de informação.

RF: Requisitos Funcionais - definição das funcionalidades de um determinado sistema de informação.

RNF: Requisitos Não funcionais - requisitos relacionados ao uso de um determinado sistema de informação.

ROI: Return on Investiment - a relação entre o dinheiro ganho como resultado de um eventual investimento.

SGBD: Sistema de Gerenciamento de Banco de Dados - conjunto responsáveis pelo gerenciamento de um banco de dados.

SOA: *Service-Oriented Architecture* - modelo de arquitetura de sistemas que prega que suas funcionalidades sejam expostas como serviços.

SOAP: *Simple Object Access Protocol* - padrão de protocolo para troca de informações.

UDDI: *Universal Description, Discovery and Integration* - serviço de diretório para registro e busca de serviços.

VPN: *Virtual Private Network* - conexão de rede privada através de uma infraestrutura pública.

W3C: *World Wide Web Consortium* - principal organização de padronização da *web*.

WS: *Web Services* - tecnologia que permite a comunicação entre aplicações de diferentes linguagens e sistema operacional.

WSDL: *Web Services Description Language* - linguagem baseada em XML para a descrição de *webservices*.

XML: *eXtensible Markup Language* - linguagem para a marcação de dados para necessidades especials.

Resumo

A Arquitetura Orientada a Serviços ou *Service-Oriented Architecture* (conhecida principalmente por sua sigla, “SOA”) tem se tornado a cada dia um assunto mais presente nas organizações, nas quais se exige cada vez mais o emprego de soluções de software de alta eficiência aliada à velocidade de desenvolvimento. Partindo deste contexto, o projeto tem por objetivo apresentar os principais conceitos relativos à arquitetura orientada a serviços, incluindo aspectos relativos ao seu nascimento, paradigmas, aplicações e também comparativos com outras abordagens. Também é objetivo deste trabalho apresentar um estudo de caso de uma real implementação de SOA em um sistema de informação voltado a um processo de negócio, que é o objeto de estudo do projeto.

Palavras chaves: Arquitetura orientada a serviços, serviços web, sistemas de informação, processos de negócio.

Abstract

The Service-Oriented Architecture (mainly known by its initials, “SOA”), has become, day after day, one of the most present subjects in organizations, in which increasingly requires the use of highly efficient software solutions, coupled with the development speed. From this context, the project aims to present the main concepts related to service-oriented architecture, including aspects of his birth, paradigms, applications and also comparison with other approaches. It is also an objective of this work, to present a case study of an actual implementation of SOA in an information system geared to a business process, which is the project’s object of study.

Keywords: Service-oriented architecture, web services, information systems, business processes.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

As últimas décadas tiveram um forte e importante papel com relação a novos desafios inerentes as organizações, nos quais processos de negócio passam por constantes evoluções e os sistemas que realizam a sua sustentação, precisam acompanhar este crescimento, criando assim, também um cenário de desafio tecnológico. Historicamente, arquiteturas de aplicações fortemente acopladas criam um paradoxo frente a este desafio, no qual a tecnologia infere diretamente nos indicadores de negócios, havendo assim, a necessidade de estruturar aplicações de uma forma voltada não somente às tecnologias, mas também frente à própria finalidade de negócio das organizações.

De acordo com a Workflow Management Coalition (1999), "um processo de negócio consiste em um conjunto de atividades relacionadas que visam atingir um único objetivo de negócio, no contexto de uma estrutura organizacional, definindo regras e relacionamentos". (p. 7) Ainda, segundo Camp (1996), um "processo de negócio", ou *bussines process* são atividades críticas para o sucesso do negócio, desta forma, o desenho funcional de um processo de negócio deve ser bem estruturado, dada sua grande importância para a eficiência e eficácia dos produtos e serviços das organizações.

Processos de negócio têm sofrido constantes transformações nas organizações, tornando-se cada vez mais um item com características de grande dinamismo e de evolução constante, sendo um dos principais pilares para o diferencial competitivo na atualidade. Segundo Cruz (2005), “negócio é a reunião de três elementos: pessoas, processos e Tecnologia da Informação, com a finalidade de atender as necessidades do cliente” (p. 46), desta forma há uma importância central nestes tópicos, que deve ser sempre levada em consideração pelas organizações.

Sistemas de informação que sustentam processos críticos de negócio são componentes fundamentais das organizações e, requisitos não funcionais como interoperabilidade, segurança de acesso, comportamento em relação ao tempo e modificabilidade (ISO, NBR. IEC 9126-1, 2003) têm sido cada vez mais cruciais para a sua garantia de sucesso.

Outro item de grande relevância nas corporações é a própria capacidade produtiva das áreas de negócio. Segundo Brasil (1993), “um fator adicional está vinculado às limitações

estruturais e circunstanciais internas à empresa" (p. 7), por muitas vezes é natural que ocorra um crescimento em um ritmo mais acelerado que a capacidade de contratação direta e até mesmo capacitação técnica de funcionários, sendo a terceirização uma alternativa a este problema. Ainda, segundo o autor, "definimos a terceirização como um processo de transferência, dentro da firma (empresa-origem), de funções que podem ser executadas por outras empresas (empresa-destino)". (p. 7).

Contextualizado o cenário em questão é possível avaliar que há uma grande necessidade para as organizações construírem - e mesmo evoluírem - sistemas de informação robustos e coesos, que possam manter as regras de negócio dentro do seu próprio domínio e que, ainda assim, consigam atender aos requisitos não funcionais já abordados, facilitando assim eventuais integrações com parceiros e colaboradores diretos.

É sob este escopo que nasceu a motivação de uso da *Service-Oriented Architecture* (SOA), que pode ser traduzida como arquitetura orientada a serviços, um estilo de arquitetura de *software* que prega, fundamentalmente, que as funcionalidades das aplicações devem ser disponibilizadas na forma de serviço, que, em grande parte dos cenários, estão conectados em uma solução conhecida como "barramento de serviços" (*enterprise service bus*), disponibilizando interfaces ou contratos acessíveis através de *webservices* ou uma outra forma de comunicação entre aplicações (BORIS, 2007).

2 Objetivos

2.1 Objetivo Geral

O trabalho teve como objetivo macro apresentar uma proposta de emprego da arquitetura orientada a serviços (SOA) aplicado a um sistema de informação, com base em um estudo de caso de uma seguradora de grande porte. Para a ilustração e melhor entendimento do estudo foi desenvolvido um protótipo funcional, que constitui-se em uma prova de conceito, baseado no produto final do objeto de estudo de caso, englobando um de seus requisitos funcionais de negócio.

2.2 Objetivos Específicos

Os objetivos específicos deste trabalho foram:

- Levantar e descrever o contexto atual do cenário da Arquitetura Orientada a Serviços;
- Traçar um comparativo SOA x demais abordagens;
- Descrever o cenário da empresa que motivou a adoção de uma aplicação baseada na arquitetura SOA;
- Apresentar a solução elaborada para o problema, envolvendo as fases de levantamento de requisitos, análise e construção;
- Explicar a solução técnica ao problema, através de uma prova de conceito.

3 Revisão Bibliográfica

Este capítulo tem como objetivo apresentar SOA e seus principais conceitos, seu posicionamento no cenário atual, abordando seus tipos de serviços e estágios de maturidade e relação com o estudo de caso deste trabalho, além de um breve comparativo com outras abordagens de arquitetura dentro da computação.

3.1 Visão Geral

Atualmente, SOA tem sido visto como um dos grandes modelos promissores para o futuro da computação distribuída. O fato de este padrão conseguir unir requisitos como os de interoperabilidade e possuir uma relação de baixo acoplamento entre seus componentes - chamados de serviços - é altamente relevante para os novos conceitos de desenvolvimento que o mercado exige. É importante contextualizar, dentro deste conceito, o que são serviços: serviços são unidades de uma aplicação, podendo conter desde operações mais simples, como por exemplo, a verificação de um dígito de CPF, como até mesmo um grande processo de negócio, como um cálculo de seguro para determinado automóvel. Serviços devem empregar um protocolo padrão e aberto para sua comunicação (como o *HiperText Transfer Protocol*, para aplicações SOA baseadas em *webservices*, por exemplo), permitindo um modelo de comunicação amplo e padronizado (SOUZA, 2006). Além disso, espera-se uma interface que forneça todos os detalhes da assinatura do serviço, tais como: parâmetros de entrada, saída e possíveis mensagens de retorno, sejam de sucesso ou falha, em outras palavras, a aplicação deve estar bem documentada para que sua finalidade de reúso seja aplicável a diferentes usuários.

Serviços devem atender uma funcionalidade completa ao seu cliente, recebendo requisições e retornando diretamente sua resposta, encapsulando assim, todo o processamento dos dados e/ou informações, sendo desta forma, totalmente transparentes em relação ao cliente.

De acordo com Hass (2003), fundamentalmente, a arquitetura de referência SOA agrega três principais papéis, que interagem entre si, sendo eles:

- **Provedor de Serviço:** entidade responsável por prover o acesso aos serviços, bem como efetuar todo seu gerenciamento, manutenção e publicação.

- **Consumidor do Serviço:** Entidade que consome os serviços disponibilizados pelo Provedor de Serviços. Caso não tenha conhecimento de requisitos específicos dos serviços, pode-se dispor de consultas no Registro de Serviços.

- **Agente de Serviços:** entidade mantenedora das informações sobre os serviços, tais como: nome, entradas, categoria, provedor, entre outras.

A Figura 1 representa o modelo de interação entre estas três entidades:

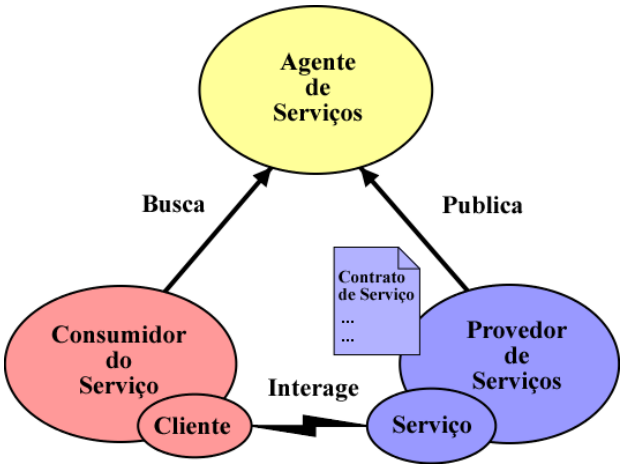

Figura 1 - Modelo de Interação de Entidades da Arquitetura SOA
Fonte: Adaptado de HASS, Hugo (2003)

A seguir serão abordadas principais características inerentes ao SOA, bem como, demais definições de termos que serão utilizados frequentemente ao longo do trabalho.

3.1.1 Principais Características

Em linhas gerais, a literatura especializada elenca diversas características inerentes a aplicações baseadas em SOA. Não há, contudo, um número exato desta quantidade. Para o presente trabalho, serão abordadas as características que foram fundamentais para a tomada de decisão de se adotar uma arquitetura orientada a serviços para o projeto da empresa, objeto do estudo de caso.

3.1.1.1 Interoperabilidade

A interoperabilidade consiste na capacidade que um sistema possui de se comunicar com outros sistemas - inclusive de tecnologias distintas, como uma integração Java *versus* .NET, por exemplo - da maneira mais transparente possível (JOSUTTIS, 2008).

Em aspectos relacionados a processos de negócios e integrações com sistemas já existentes, esta característica mostra-se essencial. A Figura 2 ilustra como ocorre a interoperabilidade entre sistemas para "casas inteligentes" através de webservices.

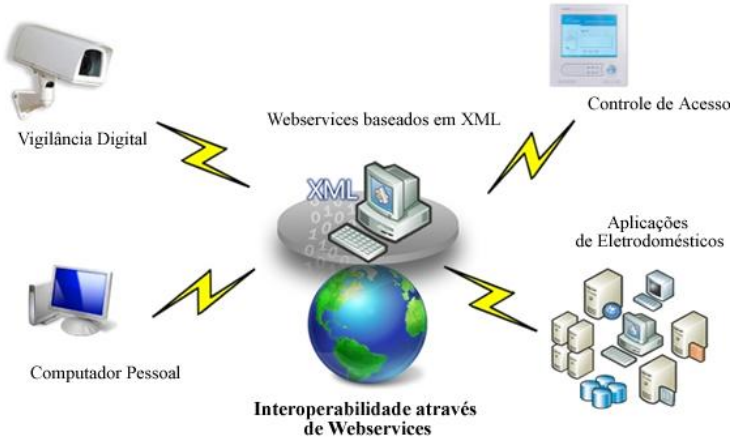

Figura 2 *- Interoperabilidade através de webservices*
Fonte: Adaptado de PERUMAL, Thinagaran (2008)

3.1.1.2 Barramento de Serviços

O Barramento de Serviço, comumente conhecido como ESB (*Enterprise Service Bus*), é o responsável por conectar todos os elementos de uma arquitetura SOA, atuando como um canal centralizador de comunicação. É também através do barramento que os serviços são compartilhados e expostos ao contexto corporativo (BLOOR, 2009).

O ESB possui a característica de suportar diversas tecnologias distintas, sejam elas referentes a linguagens de programação, sistemas operacionais ou mesmo protocolos de comunicação, o que, em linhas gerais, é a realidade das grandes organizações.

Apesar de não possuir como foco principal, o barramento também pode prover serviços de auditoria, segurança e controle de balanceamento entre requisições, tornando o ambiente mais estável e seguro. A figura 3 ilustra este modelo de arquitetura.

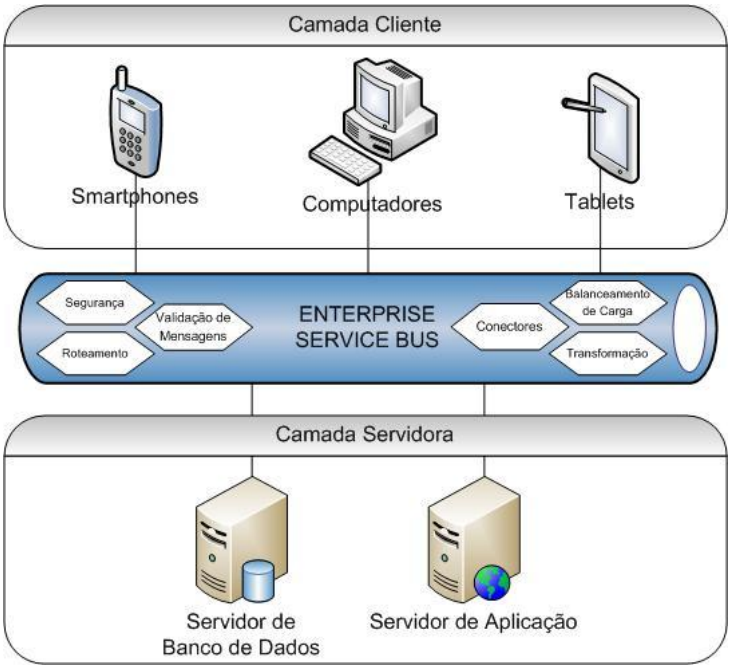

Figura 3 *- Arquitetura de um barramento de serviços*

Fonte: Elaboração própria

3.1.1.3 Fraco Acoplamento

Uma das principais características de aplicações baseadas na arquitetura SOA é o chamado baixo ou fraco acoplamento. Dentro do contexto deste trabalho, o fraco acoplamento possibilita a construção de serviços distintos e também minimizando as suas dependências. Em linhas gerais, por encapsular a lógica de implementação do serviço, um desenvolvimento de fraco acoplamento permite alta flexibilidade para mudanças em regras de negócio, deixando as mudanças transparente aos clientes do serviço (JOSUTTIS, 2008).

3.1.1.4 Contrato de Serviços

Um dos principais pilares que sustentam e também direcionam arquiteturas orientadas a serviços é o chamado de contrato de serviço. Também chamado simplesmente de “contrato”, tal característica descreve aos clientes que consumirão os serviços detalhes como a estrutura que as requisições devem possuir, bem como a estrutura de resposta do serviço.

Além dos detalhes inerentes a requisições e respostas do serviço, um contrato também define detalhes de itens que são obrigatórios para o consumo do serviço, tais como: tempo de resposta esperado, requisitos de segurança, como controle de acesso, criptografia e assinaturas digitais e também uma possível lista de exceções esperadas (ERL, 2009).

3.1.1.5 Reusabilidade

A reusabilidade de software é uma característica de grande importância para um sistema de alta qualidade, ou seja, um programa (ou partes de um programa) deve ser projetado e implantado de forma que possa ser utilizado em outras aplicações (PRESSMAN, 2011).

Serviços baseados em SOA são construídos de forma que possam ser reutilizados dentro de diferentes processos e também contextos de negócio que possuam sinergia. O conceito de reusabilidade em SOA está diretamente relacionado ao fato de seus módulos possuírem uma alta coesão e fraco acoplamento, o que contribui para o reúso.

3.1.2 Tecnologias

É importante frisar inicialmente que SOA por si só, não é uma tecnologia. São seus padrões (tecnologias) que lhe oferecem suporte que a viabiliza. Em grande parte dos casos, implementações SOA fazem uso de *webservices* e suas tecnologias de apoio, como o SOAP e WSDL. Estes padrões de tecnologia formam um SOA fundamentado na tecnologia XML, também conhecida como Primeira Geração de Arquitetura de *Webservices*.

Diante do exposto, serão abordadas a seguir as principais - e talvez fundamentais - tecnologias que SOA emprega.

3.1.2.1 *eXtensible Markup Language* (XML)

Basicamente, a base para as diversas tecnologias utilizadas em uma abordagem SOA, a Linguagem Extensível de Marcação Genérica, conhecida como XML (Extensible Modeling Language) permite que seu usuário defina sua própria marcação, sendo assim, uma meta-linguagem (uma linguagem que pode descrever outras linguagens). É um padrão recomendado pelo W3C (*World Wide Web Consortium*) para gerar linguagens de marcação para necessidades especiais.

A Figura 4 ilustra um XML para registrar um laudo de vistoria prévia, documento este que será utilizado em nossa prova de conceito.

```xml
<laudo>
   <idLaudo/>
   <laudostatusvistorias>
      <riscoAceitavel/>
   </laudostatusvistorias>
   <laudovistoriaclientes>
      <nome>GUILHERME CRUZ</nome>
      <numeroCPF>193224896</numeroCPF>
      <digitoCPF>00</digitoCPF>
      <possuiMultas>0</possuiMultas>
   </laudovistoriaclientes>
   <laudovistoriaveiculos>
      <idVeiculo>1</idVeiculo>
      <possuiAvarias>1</possuiAvarias>
   </laudovistoriaveiculos>
</laudo>
```

Figura 4 *- Exemplo de um Documento XML*

Fonte: Elaboração Própria

3.1.2.2 Simple Object Access Protocol (SOAP)

O Protocolo SOAP, ou em sua tradução direta, Protocolo Simples de Acesso a Objetos, é um protocolo baseado em XML para troca de informações dentro de uma plataforma descentralizada e distribuída.

Conforme a especificação do W3C (2000) o protocolo SOAP está segmentado em três elementos distintos:

- **SOAP Envelope:** Toda mensagem SOAP deve contê-lo. Trata-se da raiz do documento XML e contém possíveis *namespaces* (atributos que evitam o conflito de nomes nas marcações) e demais atributos que definem o estilo de codificação da mensagem.

- **SOAP *Header*:** Cabeçalho opcional que pode carregar atributos adicionais como um usuário e senha para autenticação no barramento de serviços.

- **SOAP *Body*:** Atributo obrigatório que contém a informação que deve, de fato, ser transportada ao destino final. Pode ainda conter o elemento especial *Fault*, que pode transportar possíveis mensagens de status ou mesmo erros.

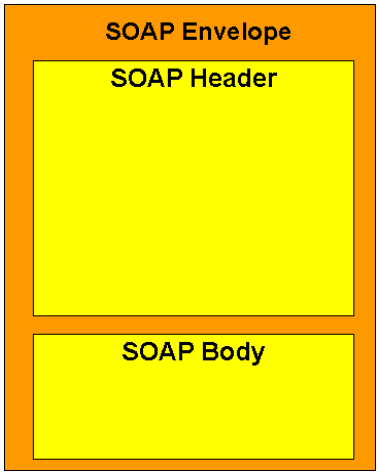

Figura 5 *- Estrutura do SOAP*

Fonte: Adaptado de WORLD WIDE WEB CONSORTIUM et al. (2000)

3.1.2.3 Web Services Description Language (WSDL)

A Linguagem Para Definição de Webservice ou WSDL é também uma especificação da W3C para a descrição e definição de *webservices*. É através dela que serviços externos podem ser descritos de forma independente de plataforma e linguagem de programação.

Tem como principal objetivo descrever todas as interfaces oferecidas por determinado serviço, bem como o seu *endpoint*, ou seja, sua localização na *web*. Segundo ERL (2009) sua estrutura básica é composta pelos seguintes elementos:

- **<definition>:** Elemento raiz do documento contém as definições de nome do serviço e eventuais *namespaces*.
- **<types>:** Define os tipos de dados utilizados pelo serviço.
- **<message>:** A mensagem propriamente dita de um serviço, para cada operação especificada no "*portType*" uma mensagem de requisição e outra de resposta pode ser incluída.
- **<portType>:** Contém as operações que um determinado serviço provê. Combina elementos especificados na "*message*" para formar uma requisição ou resposta.
- **<binding>:** Descreve os detalhes técnicos de como as mensagens serão transmitidas, no exemplo abaixo, através do Protocolo SOAP.

- **<service>:** O chamado *endpoint*, o local na *web* no qual o serviço é acessível aos clientes.

```
<wsdl:definitions name="GerenciarLaudoBusinessService" targetNamespace="http://provider.ejb.tcc.si.usp.br/" xmlns:ns1="http://sch
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://provider.ejb.tcc.si.usp.br/" xmlns:wsdl="http://schemas.xmls
   <wsdl:types>
      <xs:schema targetNamespace="http://provider.ejb.tcc.si.usp.br/" version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
   </wsdl:types>
   <wsdl:message name="salvar">
      <wsdl:part name="laudo" type="tns:laudoVistoriaCapa"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="salvarResponse">
      <wsdl:part name="return" type="tns:laudoVistoriaCapa"></wsdl:part>
   </wsdl:message>
   <wsdl:portType name="GerenciarLaudoBusiness">
      <wsdl:operation name="salvar">
         <wsdl:input message="tns:salvar" name="salvar"></wsdl:input>
         <wsdl:output message="tns:salvarResponse" name="salvarResponse"></wsdl:output>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="GerenciarLaudoBusinessServiceSoapBinding" type="tns:GerenciarLaudoBusiness">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="salvar">
         <soap:operation soapAction="" style="rpc"/>
         <wsdl:input name="salvar">
            <soap:body namespace="http://provider.ejb.tcc.si.usp.br/" use="literal"/>
         </wsdl:input>
         <wsdl:output name="salvarResponse">
            <soap:body namespace="http://provider.ejb.tcc.si.usp.br/" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="GerenciarLaudoBusinessService">
      <wsdl:port binding="tns:GerenciarLaudoBusinessServiceSoapBinding" name="GerenciarLaudoBusinessPort">
         <soap:address location="http://localhost:8080/app-service-1.0.0/GerenciarLaudoBusiness"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Figura 6 *- Documento WSDL*

Fonte: Elaboração Própria

3.1.2.4 Universal Description, Discovery and Integration (UDDI)

Segundo ERL (2009), a especificação UDDI possibilita a publicação - bem como a descoberta - de diretórios de serviços em uma arquitetura orientada a serviços. Dentro de um contexto organizacional, onde provedores de serviços precisam registrar e publicar diversos *webservices*, e clientes precisam localizá-los. Sendo assim, pode-se compreender que o UDDI é uma peça fundamental para a sustentação e governança destes serviços.

Geralmente, em termos de interface, um registro UDDI assemelha-se a uma página de busca na web, conforme ilustra a Figura 7.

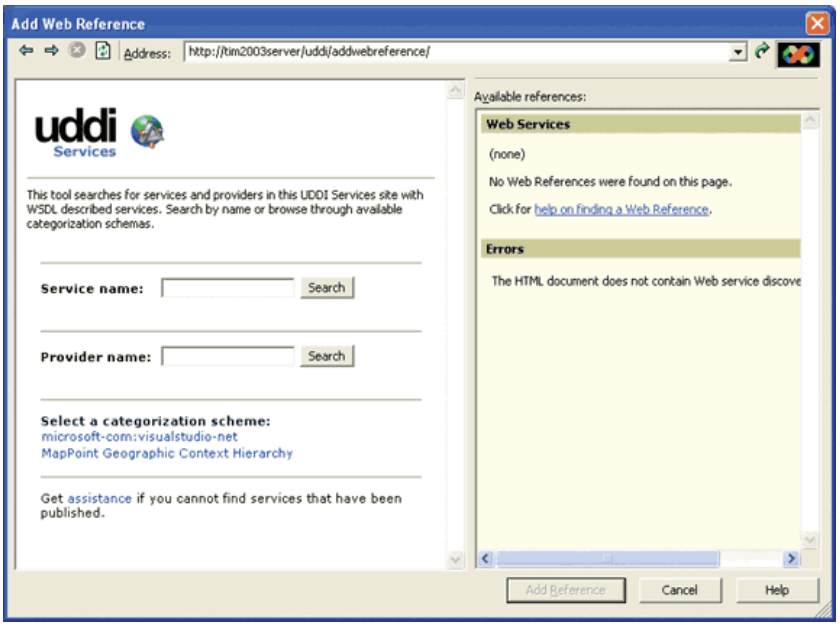

Figura 7 *- Interface de um serviço UDDI*

Fonte: MCCARTHY, Tim (2004)

3.1.2.5 Webservices

Webservices são sistemas designados para suportar a interoperabilidade entre máquinas em uma rede, sendo fortemente empregada para realizar integrações de sistemas e comunicação entre aplicações distintas. Utiliza-se de tecnologias padronizadas, tais como SOAP, WSDL e UDDI, o que torna seu desenvolvimento independente de plataforma ou linguagem de programação.

3.1.2.6 Relacionamento Entre Tecnologias da Arquitetura SOA

Introduzidos os principais conceitos tecnológicos relacionados a SOA podemos traçar um comparativo - com uma abordagem mais técnica - frente à Figura 1 (Modelo de Interação de Entidades da Arquitetura SOA):

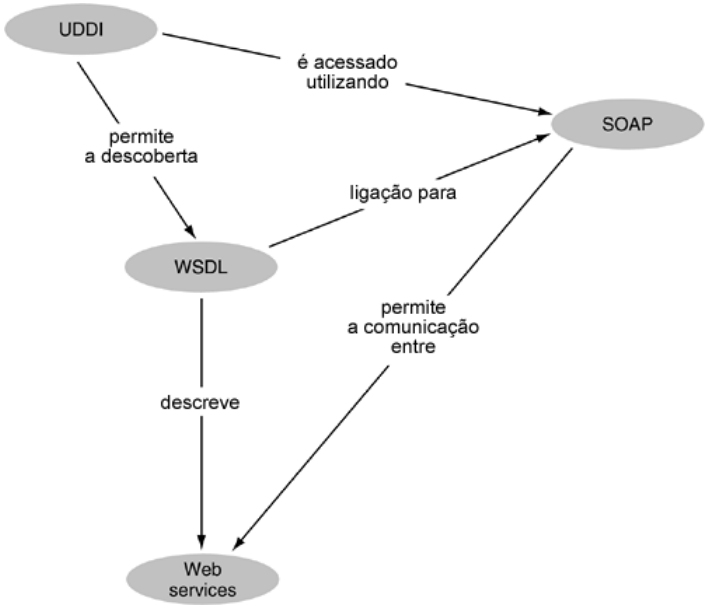

Figura 8 *- Relacionamento Entre Tecnologias da Arquitetura SOA*

Fonte: Adaptado de ERL, Thomas (2009)

3.2 O Cenário Atual

A Arquitetura Orientada a Serviços tem conquistado cada vez mais espaço nas organizações. A literatura especializada de certa forma, é rica, estruturada e contempla diversos pontos relevantes que auxiliam na tomada de decisão das organizações. Questões relacionadas a segmentação em níveis de maturidade bem definidos e estruturados, além de uma boa separação de quais tipos de serviços se encaixam para determinados cenários são fatores fundamentais para um uso efetivo e eficaz do modelo SOA. A seguir, será apresentado o modelo de maturidade SOA proposto pelo *The Open Group*, um consórcio formado por mais de 450 empresas de tecnologia - tais como IBM, HP e Oracle - que visa estabelecer padrões abertos para a infraestrutura em computação (THE OPEN GROUP, 2012).

3.2.1 Modelo de Maturidade SOA do *The Open Group*

De acordo com o *The Open Group* (2011), o modelo de maturidade SOA é denominado como "OSIMM" (*The Open Group Service Integration Maturity Model* ou Modelo de Maturidade de Integração de Serviços do *The Open Group*). Através dele pode-se estabelecer em qual nível de maturidade determinada organização insere-se em relação ao emprego de

SOA. Como pode-se observar na figura 9, o modelo consiste em uma matriz de relacionamento de sete dimensões (linhas) e sete níveis (colunas), estes detalhados a seguir.

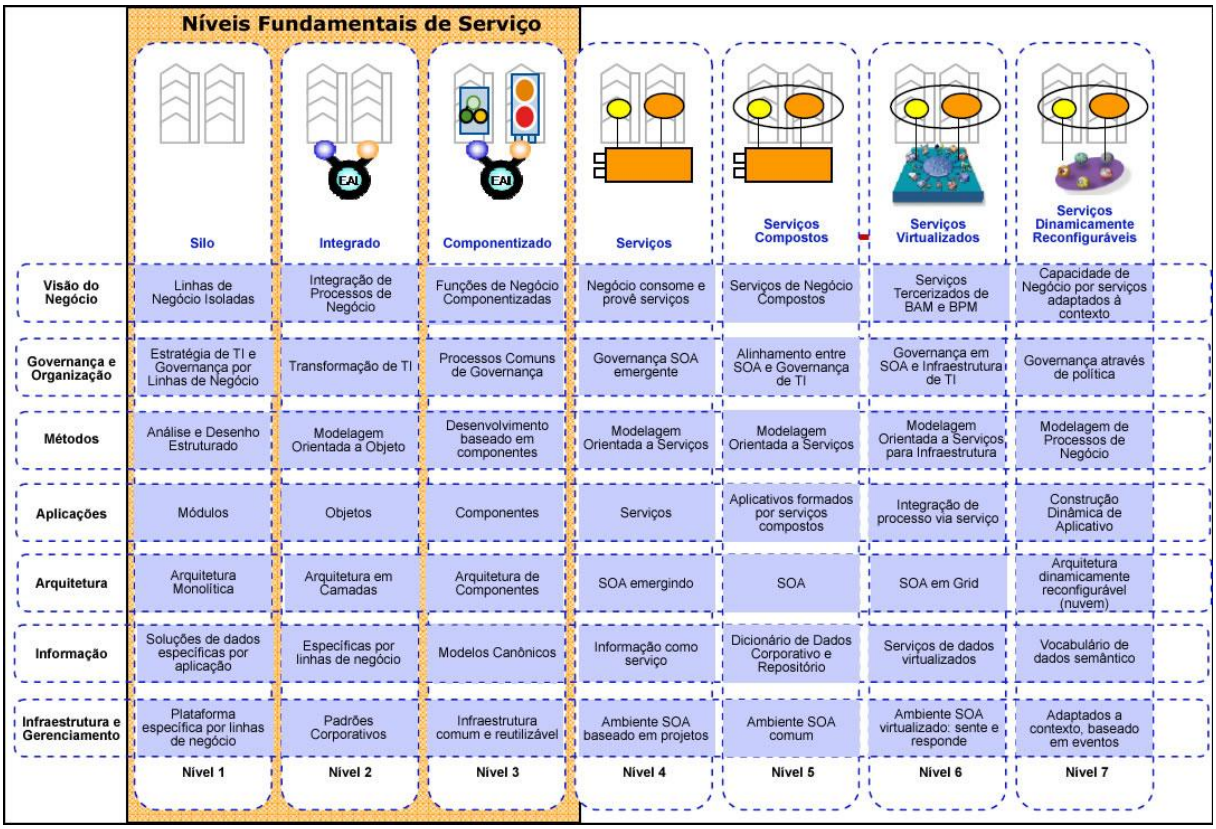

Figura 9 *- Matriz de Maturidade OSIMM*

Fonte: Adaptado de THE OPEN GROUP, 2011

3.2.2 Níveis de Maturidade SOA do *The Open Group*

Segundo o *The Open Group* (2011), os níveis de maturidade SOA são:

- **Nível 1 - Silo:** as aplicações da organização são desenvolvidas de forma separada e não são integradas;
- **Nível 2 - Integrado:** há uma visão com foco em integrações entre os silos;
- **Nível 3 - Componentizado:** as aplicações dos diferentes silos já passaram por um processo de análise e dividida em componentes, permitindo um desenvolvimento orientado.

- **Nível 4 - Serviços:** as aplicações já são desenvolvidas de forma orientada à serviços, possibilitando o baixo acoplamento e baseadas em padrões abertos, atingindo assim a interoperabilidade entre sistemas;

- **Nível 5 - Serviços Compostos:** é possível construir e modelar processos através de uma linguagem para esta finalidade, como por exemplo BPEL (*Business Process Execution Language* ou Linguagem de Execução de Processos de Negócio) e não somente através de desenvolvimento tradicional.

- **Nível 6 - Serviços Virtualizados:** os consumidores não acionam o serviço de forma direta, mas sim através de um serviço virtual que encapsula o serviço original, ficando a cargo da infraestrutura todo processo de conversão e acionamento.

- **Nível 7 - Serviços Dinamicamente Reconfiguráveis:** realizando um breve paralelo com os níveis anteriores, nos quais exige-se uma equipe de tecnologia composta por arquitetos, desenvolvedores e analistas, um nível de maturidade 7 permite que analistas de negócio ou de processos definam serviços através de ferramentas mais amigáveis e em tempo de execução.

Ainda como elemento importante em SOA, há a figura do *application frontend* que, apesar de não ser um serviço é um elemento ativo na matriz SOA das organizações. Na prática, este elemento consiste nas diversas aplicações que interagem diretamente com o usuário final. Dada a forte característica de interoperabilidade que SOA emprega, essa aplicação pode estar contida em um *smartphone*, computador *desktop* ou mesmo na própria *web* (KRAFZIG, BANKE, SLAMA, 2005).

3.3 Arquitetura Orientada a Serviços: Um Breve Comparativo

Neste capítulo será realizado um breve comparativo entre as abordagens da Arquitetura Orientada a Serviços (SOA) *versus* a Programação Orientada a Objetos (POO), termos estes que ainda geram muita disputa e confusão. É recomendável ao leitor ter conhecimento dos principais conceitos da orientação a objetos, visto que o objetivo deste estudo é realizar um comparativo com SOA, que já foi explorado nos capítulos anteriores. Espera-se ainda poder desmistificar a ideia de que exista uma possível rivalidade direta entre as mesmas.

3.3.1 A Programação Orientada a Objetos e a Arquitetura Orientada a Serviços

Para que seja possível traçar o paralelo entre POO x SOA é importante relembrar o que é a POO: a orientação a objetos, por si só, consiste em um modelo ou paradigma para as fases de análise, projeto e programação de software. Ainda, segundo Souza (2013) "É um paradigma que baseia-se na utilização de componentes individuais (objetos) que colaboram para construir sistemas mais complexos. A colaboração entre os objetos é feita através do envio de mensagens".

3.3.1.1 Objetivos

A programação orientada a objetos engloba uma série de vantagens que contribuem para uma boa produtividade no desenvolvimento de software. Podendo destacar, principalmente, o seu custo reduzido para manutenção, graças a características como o encapsulamento e a herença entre as suas classes. Compreendendo estas vantagens também é possível traçar um paralelo de objetivos em comum de ambas abordagens. ERL (2009) elenca os seguintes objetivos:

- Melhor atendimento aos requisitos de negócio;
- Maior robustez;
- Maior capacidade de extensão;
- Maior flexibilidade;
- Mais reúso e produtividade.

Traçado estes objetivos fica claro que grande parte da orientação a serviços tem como base princípios e modelos que nasceram com a orientação a objetos. A diferença ocorre principalmente na ênfase destes objetivos: enquanto, historicamente, a orientação a objetos foi sendo aplicada em segmentos distintos das organizações, a orientação a serviços também está preocupada com a governança e em como estruturar serviços que possuam benefícios estratégicos de longo prazo (ERL, Thomas, 2009). A figura 10 ilustra esta visão de escopo de SOA x POO.

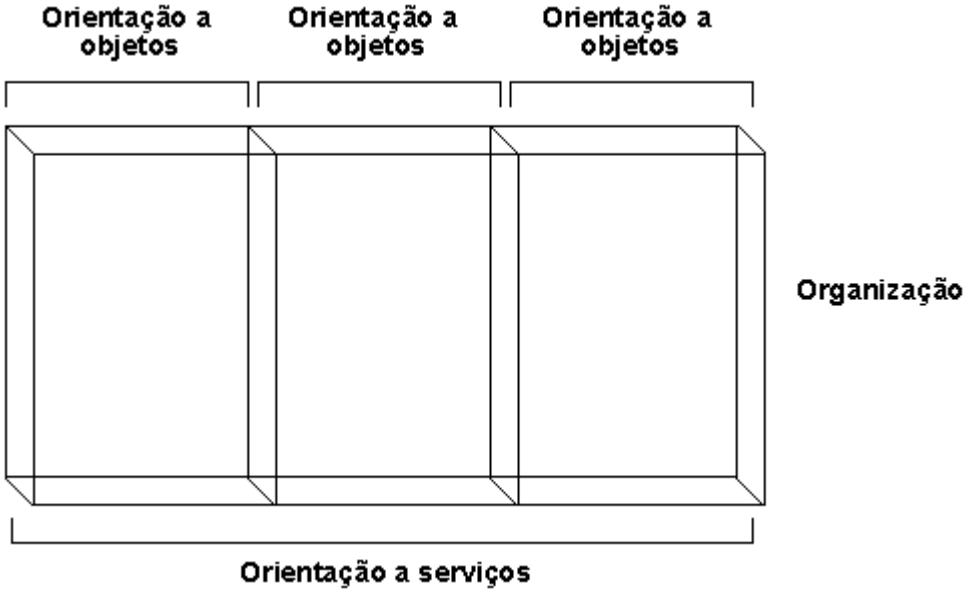

Figura 10 *- Escopo Organizacional SOA x POO*

Fonte: Adaptado de ERL, Thomas (2009)

3.3.1.2 Conceitos

Em termos conceituais, POO e SOA possuem semelhanças, porém não são idênticas. Esta seção irá apresentar, por meio da tabela 2 quais são estas semelhanças e diferenças conceituais e também discutir como aplicar tais conceitos olhando para a POO e SOA como tecnologias complementares, e não rivais.

Tabela 1 *- Comparativo dos conceitos de POO com SOA*

	POO	**SOA**
Classes	São a representatividade de objetos que possuam características comuns.	É de certa forma comparável ao contrato de serviço, porém não equivalente. Enquanto uma classe pode definir acessos públicos e privados um contrato sempre será público. Seguindo este conceito o contrato é similar a uma interface de OO.
Objetos	Um objeto pode armazenar diversos atributos e se relacionar enviando e recebendo mensagens a outros objetos.	Em tempo de execução, uma instância de classe é um objeto e, de certa forma, podemos definir que uma instância de serviço também é um objeto.
Métodos e Atributos	Utilizados para, respectivamente, associar comportamento e dados entre objetos, comportamento, em linhas gerais, representa a funcionalidade de uma classe.	Em OS comportamentos são capacidades abstratas. Capacidade é o equivalente de um método se um serviço é implementado como um componente e uma operação se um serviço é implantado como um *webservice* (WS).

Mensagens	Consiste na chamada de um objeto, invocando um de seus métodos e proporcionando um comportamento descrito por sua classe.	Em serviços implementados como WS as mensagens se manifestam como uma forma de comunicação que podem ser trocadas de forma síncrona ou mesmo assíncrona.
Encapsulamento	É a separação dos aspectos internos e externos dos objetos. Na OO o aspecto é diretamente ligado ao ocultamento de informações.	Tal princípio é válido dentro de SOA, que possui como interesse fundamental as características relacionadas a implementação de seus serviços
Herança	Permite que uma classe estenda outra classe, herdando seus comportamentos e variáveis possíveis.	SOA desencoraja o conceito de herança. Serviços devem possuir um fraco acoplamento e terem automonia individual.
Abstração	Em OO é um outro princípio que leva em consideração a ocultação de informação. A abstração pode criar uma classe simplificada, que oculta a complexidade da implementação e expõe apenas os métodos (também abstratos) e atributos mais necessários	Em termos conceituais a abstração em SOA é similar no sentido de que ambos pretendem simplificar informações públicas sobre a lógica e detalhes de implementação. Entretanto, conforme definido no conceito de herança, aqui também não há o conceito de classe abstrata.

Fonte: Adaptado de ERL, Thomas (2009)

Diante de uma breve análise da tabela 2 é possível observar que POO e SOA partilham, de fato, de princípios comuns; estes geralmente relacionados a aumento de robustez, flexibilidade, reúso e aumento na produtividade. Então é possível dizer que SOA pode substituir ou é melhor que POO? Josuttis (2008) afirma que tal discussão é, na realidade, um desperdício: “A comparação entre SOA e POO simplesmente não faz sentido, porque eles possuem finalidades diferentes”.

4 Metodologia

4.1 Metodologia de Pesquisa

A parte referente a pesquisa do trabalho consistiu em um estudo dos diversos trabalhos relacionados à SOA (incluindo aqui os principais livros referências da área, através de uma revisão bibliográfica). Este estudo teve como foco elencar e explicar as principais características de SOA além de realizar um breve comparativo com a Orientação a Objetos, focos estes apresentados no capítulo anterior.

4.2 Metodologia de Desenvolvimento da Aplicação

O papel da aplicação desenvolvida foi o de atuar como uma PoC (Prova de Conceito, sigla do inglês: *Proof of Concept*) deste trabalho. PoC "é um termo utilizado para denominar um modelo prático que possa provar o conceito (teórico) estabelecido por uma pesquisa ou artigo técnico. Em Tecnologia da Informação (TI), o termo pode ser relacionado ao desenvolvimento de um protótipo como ferramenta para provar a viabilidade de um projeto de rede de computadores." (PINHEIRO, 2010).

Também vale destacar que, para uma melhor ilustração da problemática inerente ao negócio da organização estudada, a prova de conceito contemplou um requisito funcional de negócio essencial ao projeto objeto de estudo deste trabalho (Registrar Laudo de Vistoria), além de uma modelagem sistêmica similar.

O desenvolvimento da aplicação deste trabalho baseou-se na estrutura do modelo em cascata, envolvendo assim as principais fases do ciclo de vida de uma aplicação: análise de requisitos, projeto, implementação e testes de forma linear (PRESSMAN, 2011), visto o seu objetivo de atuar somente como uma PoC.

Para a construção da PoC utilizou-se a plataforma de programação Java EE 6[1] (*Java Platform, Enterprise Edition*, ou em português Plataforma Java, Edição Empresarial), que permite um ágil mapeamento objeto-relacional além de um desenvolvimento flexível para *web services* (GONCALVES, 2010). O SGBD escolhido foi o *MySQL Community Server*

[1] https://docs.oracle.com/javaee/6/tutorial/doc/index.html

5.6[2]. Por fim, os testes da PoC foram realizados através da aplicação soapUI[3], que permite a realização de testes com diferentes abordagens (de testes unitários a testes de desempenho) sem a necessidade de se construir uma aplicação final do lado cliente (KANKANAMGE, 2012).

4.3 Metodologia do Estudo de Caso

Por fim, o trabalho também baseia-se em um estudo de caso particular de um projeto de tecnologia da informação, que teve como desafio redesenhar uma arquitetura antiga e com diversos problemas com um objetivo de atender ao menos no nível 4 de maturidade SOA. O próprio *The Open Group*, através de seu modelo OSIMM, define algumas etapas para a evoluir ou avaliar o nível de maturidade SOA nas organizações, as etapas adotadas neste estudo de caso estão demonstradas na figura 11.

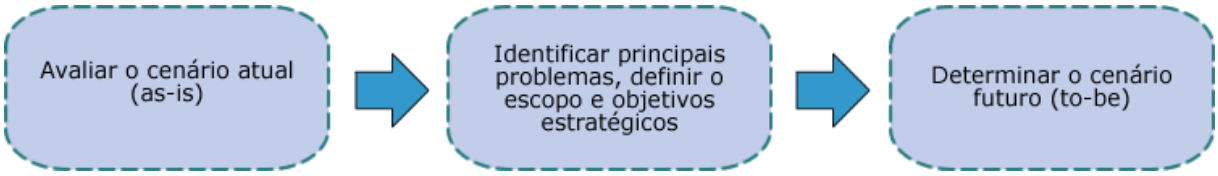

Figura 11 *- Etapas para a evolução/avaliação da maturidade SOA*

Fonte: Adaptado do THE OPEN GROUP (2011)

[2] http://dev.mysql.com/downloads/mysql/

[3] http://www.soapui.org/downloads/soapui/open-source.html

5 Resultados

Os resultados apresentados a seguir compreendem o desenvolvimento da Prova de Conceito e o estudo de caso referente a adoção de uma solução SOA para o processo de negócio de realização de vistoria em veículos em uma seguradora de grande porte, que nomearemos a partir deste ponto de “Seguradora Alfa”.

5.1 Prova de Conceito

5.1.1 Levantamento de Requisitos

A atividade de levantamento de requisitos é um dos pilares fundamentais no processo de desenvolvimento de *software*. Abaixo serão apresentados os requisitos funcionais (RF) e não funcionais (RFN) necessários para o desenvolvimento da PoC.

5.1.1.1 RF 1: Registrar Laudo de Vistoria

Requisito responsável pelo registro e gravação do laudo de vistoria prévia referente a determinado veículo e cliente. A aplicação deverá passar com sucesso pelos seguintes requisitos para efetuar a gravação do laudo: Corrigir Dados, Validar Dados e Consistir Dados.

5.1.1.2 RF 2: Corrigir Dados

Requisito responsável pela prévia correção de dados que serão recebidos de diferentes clientes do serviço. O objetivo principal é evitar a entrada de caracteres especiais, tais como, caracteres acentuados, símbolos especiais, entre outros.

5.1.1.3 RF 3: Validar Dados

Requisito responsável pela validação de dados que serão recebidos de diferentes clientes do serviço, o objetivo principal é garantir que dados sensíveis ao negócio, tais como, placa de veículo e CPFs estejam válidos em termos de preenchimento, como, por exemplo, uma placa de veículo deve possuir três letras e quatro números.

5.1.1.4 RF 4: Consistir Dados

Requisito responsável pela garantia de consistência de dados que serão persistidos em banco, tais como, um veículo deve possuir um código válido e ativo no cadastro de veículos para gravar um laudo de vistoria com sucesso.

5.1.1.5 RNF 1: Interoperabilidade

O sistema deve possuir um alto grau de flexibilidade para futuras integrações, garantindo um contrato de serviço que possa ser utilizado por diferentes tecnologias para desenvolvimento de software.

5.1.1.6 RNF 2: Modificabilidade

O sistema deve dispor as regras de negócio inerentes a organização de maneira centralizada além de possuir uma alta flexibilidade para eventuais alterações com o menor impacto possível aos clientes.

5.1.2 Desenho

5.1.2.1 Diagrama de Caso de Uso

O diagrama de caso de uso é a representação gráfica dos requisitos funcionais de uma aplicação, para a PoC o seguinte modelo foi proposto:

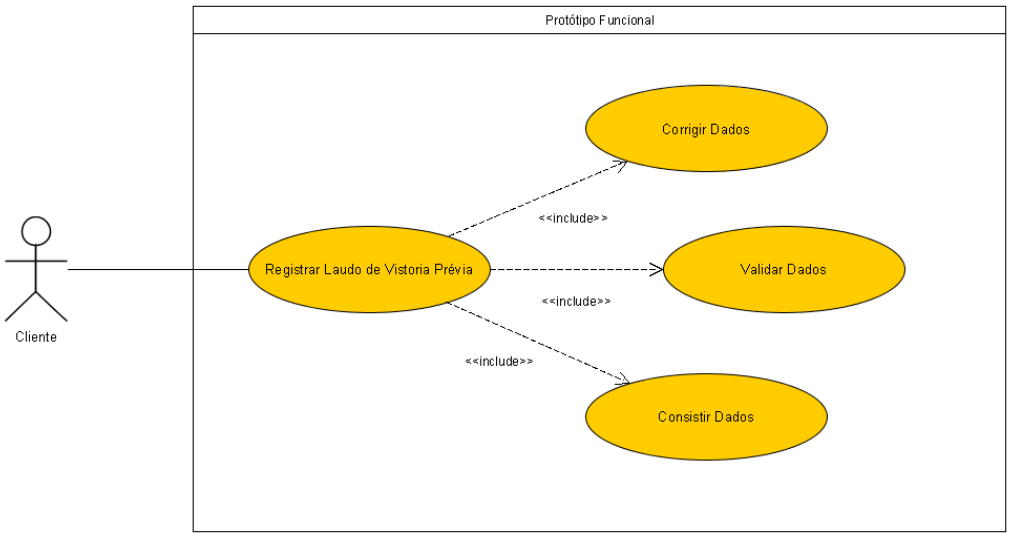

Figura 12 *- Prova de Conceito: Diagrama de Caso de Uso*

Fonte: Elaboração própria

5.1.2.2 Modelo Físico de Dados

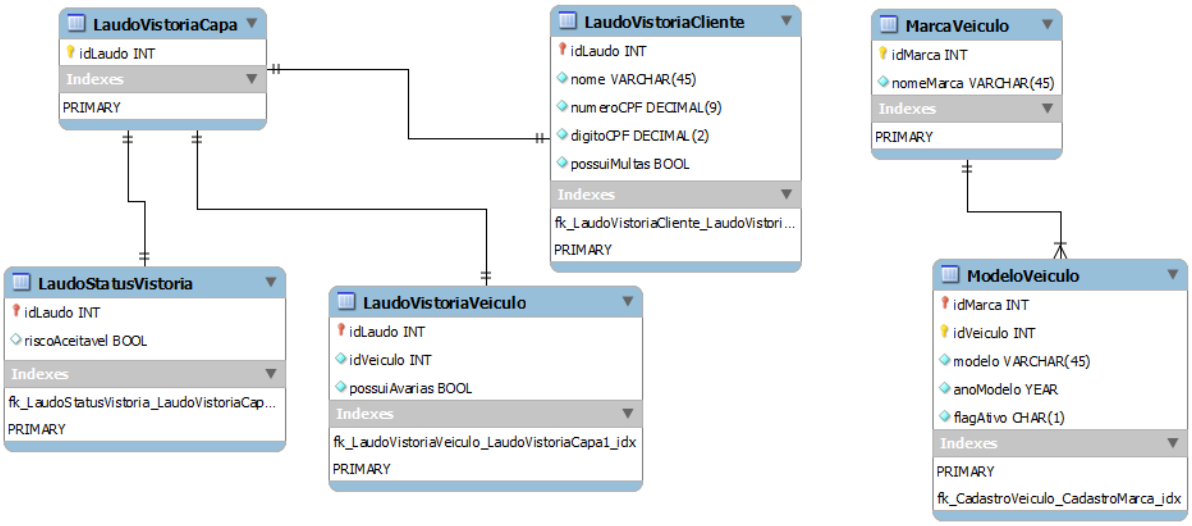

Figura 13 *- Prova de Conceito: Modelo Físico de Dados*

Fonte: Elaboração própria

5.1.3 Implementação

5.1.3.1 Diagrama de Classes de Projetos

Os principais pontos que motivaram a adoção de uma solução SOA ao problema estão mapeados como classes distintas dentro da aplicação e a PoC também é composta por todas elas. A figura 14 ilustra este relacionamento.

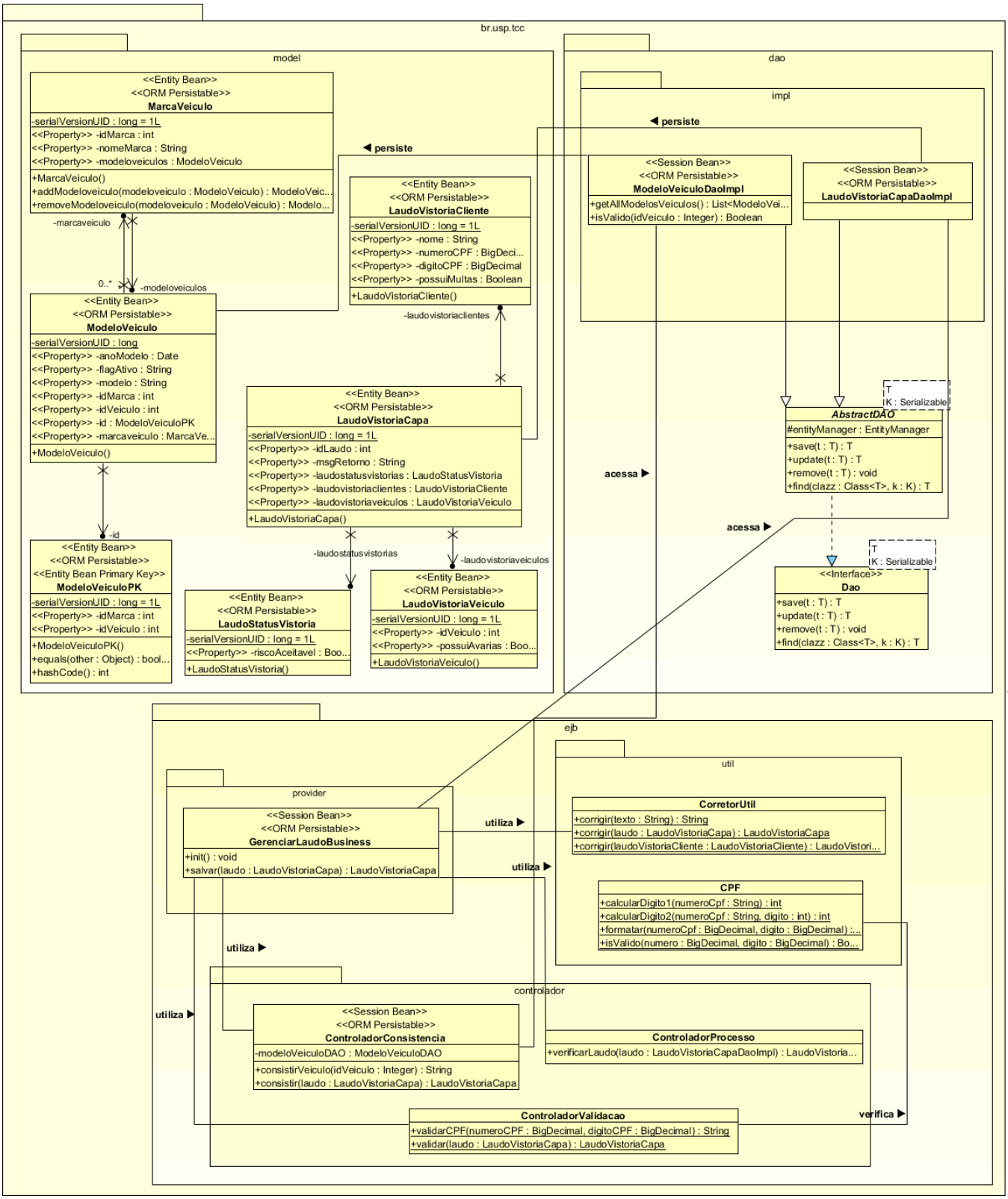

Figura 14 *- Prova de Conceito: Diagrama de Classes de Projeto*

Fonte: Elaboração própria

5.1.3.2 Fluxograma de Processos

Uma vez identificado e modelado os relacionamentos das classes do projeto é possível definir e ilustrar a ordem de execução de cada processo (Figura 15). A funcionalidade detalhada de cada processo também será apresentada a seguir.

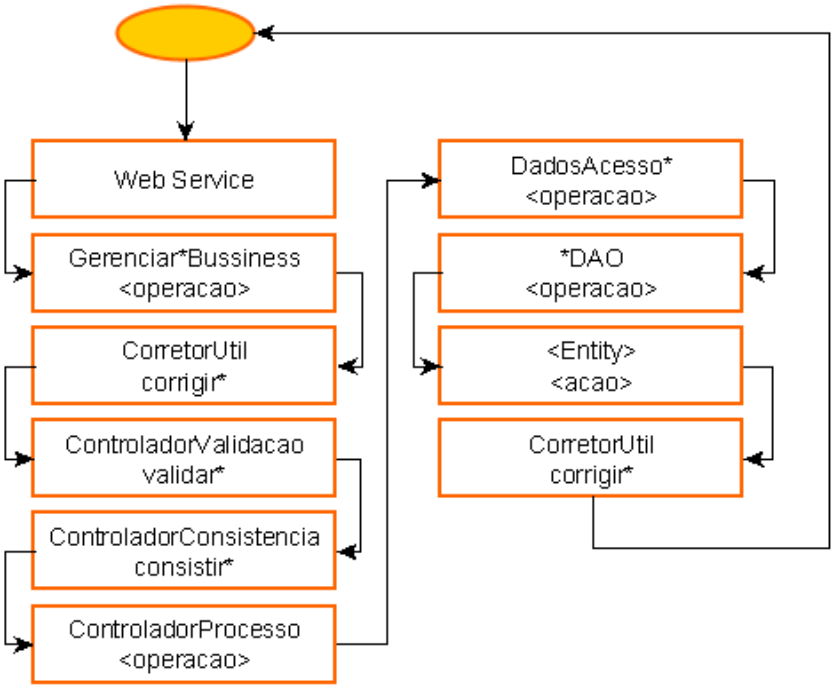

***Figura 15** - Prova de Conceito: Fluxograma de Processos*

Fonte: Elaboração própria

- **Gerenciar*Bussiness <operacao>:** entidade chave da aplicação, responsável por centralizar as chamadas e retornos das camadas de correção, validação e consistência, bem como, em caso de sucesso em todas estas operações, efetuar o retorno do objeto já persistido em banco.

- **CorretorUtil - corrigir*:** responsável pelo requisito "**Corrigir Dados**". Entende-se como "correção" a remoção de caracteres especiais, tais como sinais de acentuação ou mesmo símbolos que possam comprometer os sistemas e bancos de dados legados. Seu acionamento ocorre em dois momentos: na entrada do fluxo principal - corrigindo assim dados enviados pelo cliente - e também em seu término, garantindo que o cliente recebe como resposta o exato objeto que foi persistido no banco de dados.

- **CorretorValidacao - validar*:** responsável pelo requisito "**Validar Dados**". Entende-se como "validação" a verificação de dados sensíveis ao negócio e que possuam regras bem estruturadas, como o dígito verificador do CPF e a placa de um veículo.

- **Controlador Consistência - consistir*:** responsável pelo requisito "Consistir Dados". Entende-se como "consistência" a garantia de que o dado a ser gravado possua um registro válido em uma eventual tabela de cadastros, como um veículo, por exemplo.

- **Controlador Processo* <operacao>:** entidade responsável pelas regras de negócio inerentes a um determinado processo, controlando eventuais bloqueios ou mesmo permitir a atualização em caso de laudos já preenchidos.

- **Dados Acesso* <operacao>:** responsável pela centralização dos acessos em banco, controlando as chamadas aos objetos DAO e possíveis regras de negócio.

- ***DAO <operacao>:** entidade da camada de acesso a dados, para realizar a separação lógica de instruções de negócio e acesso a banco.

- **<Entity> <acao>:** entidade que representa um objeto manipulável em banco e possui as ações para gravar, excluir, atualizar ou selecionar dados.

5.1.4 Testes

Com o objetivo de testar o trabalho desenvolvido serão apresentados quatro testes, nos quais será possível observar o comportamento da integração entre os *webservices* desenvolvidos através da ilustração das mensagens de entrada e saída via protocolo SOAP.

5.1.4.1 Teste 1 - Corrigir Dados

Para campos enviados com caracteres especiais, tais como sinais de acentuação ou símbolos, o serviço deverá corrigi-los, mantendo assim a compatibilidade com sistemas legados.

```
<soapenv:Header/>
<soapenv:Body>
   <prov:salvar>
      <laudo>
         <idLaudo/>
         <laudostatusvistorias>
            <riscoAceitavel/>
         </laudostatusvistorias>
         <laudovistoriaclientes>
            <digitoCPF>00</digitoCPF>
            <nome>JOÃO CALÇADE VINÍCIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>0</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>1</idVeiculo>
            <possuiAvarias>1</possuiAvarias>
         </laudovistoriaveiculos>
      </laudo>
   </prov:salvar>
</soapenv:Body>
</soapenv:Envelope>
```

Figura 16 *- SOAP de entrada Teste 1 (Nome acentuado)*

Fonte: Elaboração própria

```
<soap:Body>
   <ns1:salvarResponse xmlns:ns1="http://provider.ejb.tcc.si.usp.br/">
      <return>
         <idLaudo>0</idLaudo>
         <laudostatusvistorias/>
         <laudovistoriaclientes>
            <digitoCPF>0</digitoCPF>
            <nome>JOAO CALCADE VINICIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>false</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>1</idVeiculo>
            <possuiAvarias>true</possuiAvarias>
         </laudovistoriaveiculos>
      </return>
   </ns1:salvarResponse>
</soap:Body>
</soap:Envelope>
```

Figura 17 *- SOAP de saída Teste 1 (Nome corrigido - sem acentuação)*

Fonte: Elaboração própria

5.1.4.2 Teste 2 - Validar Dados

Dados que possuam validações sensíveis, tais como um CPF inválido, deverão ser criticados pelo serviço, caso sejam inválidos e informar ao usuário.

```
<soapenv:Header/>
<soapenv:Body>
   <prov:salvar>
      <laudo>
         <idLaudo/>
         <laudostatusvistorias>
            <riscoAceitavel/>
         </laudostatusvistorias>
         <laudovistoriaclientes>
            <digitoCPF>20</digitoCPF>
            <nome>JOÃO CALÇADE VINÍCIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>0</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>1</idVeiculo>
            <possuiAvarias>1</possuiAvarias>
         </laudovistoriaveiculos>
      </laudo>
   </prov:salvar>
</soapenv:Body>
</soapenv:Envelope>
```

Figura 18 *- SOAP de entrada Teste 2 (CPF inválido)*

Fonte: Elaboração própria

```
<soap:Body>
   <ns1:salvarResponse xmlns:ns1="http://provider.ejb.tcc.si.usp.br/">
      <return>
         <idLaudo>0</idLaudo>
         <laudostatusvistorias/>
         <laudovistoriaclientes>
            <digitoCPF>20</digitoCPF>
            <nome>JOAO CALCADE VINICIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>false</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>1</idVeiculo>
            <possuiAvarias>true</possuiAvarias>
         </laudovistoriaveiculos>
         <msgRetorno>CPF INVALIDO!</msgRetorno>
      </return>
   </ns1:salvarResponse>
</soap:Body>
</soap:Envelope>
```

Figura 19 *- SOAP de saída Teste 2 (Informar dado inválido)*

Fonte: Elaboração própria

5.1.4.3 Teste 3 - Consistir Dados

Informações que possuam relação com o cadastro da organização deverão ser verificadas antes de serem persistidas em banco, como um código de veículo que deve ser válido e ativo, a verificação é realizada com base na tabela 4, para veículos que possuam a *flag* ativo igual a "A".

Tabela 2 *- Modelos de veículos cadastrados na base de consistência*

ID Marca	ID Veículo	Modelo	Ano Modelo	Flag Ativo
1	1	GOL	2010	A
1	2	FOX	2010	A
1	3	VOYAGE	2010	I
2	4	PALIO	2010	I
2	5	STILO	2010	I
2	6	PUNTO	2010	A

Fonte: Elaboração própria

```
<soapenv:Header/>
<soapenv:Body>
   <prov:salvar>
      <laudo>
         <idLaudo/>
         <laudostatusvistorias>
            <riscoAceitavel/>
         </laudostatusvistorias>
         <laudovistoriaclientes>
            <digitoCPF>00</digitoCPF>
            <nome>JOÃO CALÇADE VINÍCIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>0</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>99</idVeiculo>
            <possuiAvarias>1</possuiAvarias>
         </laudovistoriaveiculos>
      </laudo>
   </prov:salvar>
</soapenv:Body>
</soapenv:Envelope>
```

Figura 20 *- SOAP de entrada Teste 3 (Veículo não cadastrado)*

Fonte: Elaboração própria

```
<soap:Body>
   <ns1:salvarResponse xmlns:ns1="http://provider.ejb.tcc.si.usp.br/">
      <return>
         <idLaudo>0</idLaudo>
         <laudostatusvistorias/>
         <laudovistoriaclientes>
            <digitoCPF>20</digitoCPF>
            <nome>JOAO CALCADE VINICIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>false</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>99</idVeiculo>
            <possuiAvarias>true</possuiAvarias>
         </laudovistoriaveiculos>
         <msgRetorno>O VEICULO DE ID 99 NAO E VALIDO!</msgRetorno>
      </return>
   </ns1:salvarResponse>
</soap:Body>
</soap:Envelope>
```

Figura 21 *- SOAP de saída Teste 3 (Informar id não é válido)*

Fonte: Elaboração própria

5.1.4.4 Teste 4 - Registrar Laudo de Vistoria Prévia

Passado com sucesso nos testes anteriores o laudo deverá ser gravado em banco e ter um eventual status de aceitação gerado, conforme as definições do processo de negócio.

```
<soapenv:Header/>
<soapenv:Body>
   <prov:salvar>
      <laudo>
         <idLaudo/>
         <laudostatusvistorias>
            <riscoAceitavel/>
         </laudostatusvistorias>
         <laudovistoriaclientes>
            <digitoCPF>00</digitoCPF>
            <nome>JOÃO CALÇADE VINÍCIUS</nome>
            <numeroCPF>193224896</numeroCPF>
            <possuiMultas>0</possuiMultas>
         </laudovistoriaclientes>
         <laudovistoriaveiculos>
            <idVeiculo>1</idVeiculo>
            <possuiAvarias>0</possuiAvarias>
         </laudovistoriaveiculos>
      </laudo>
   </prov:salvar>
</soapenv:Body>
</soapenv:Envelope>
```

Figura 22 *- SOAP de entrada Teste 4 (Laudo preenchido corretamente)*

Fonte: Elaboração própria

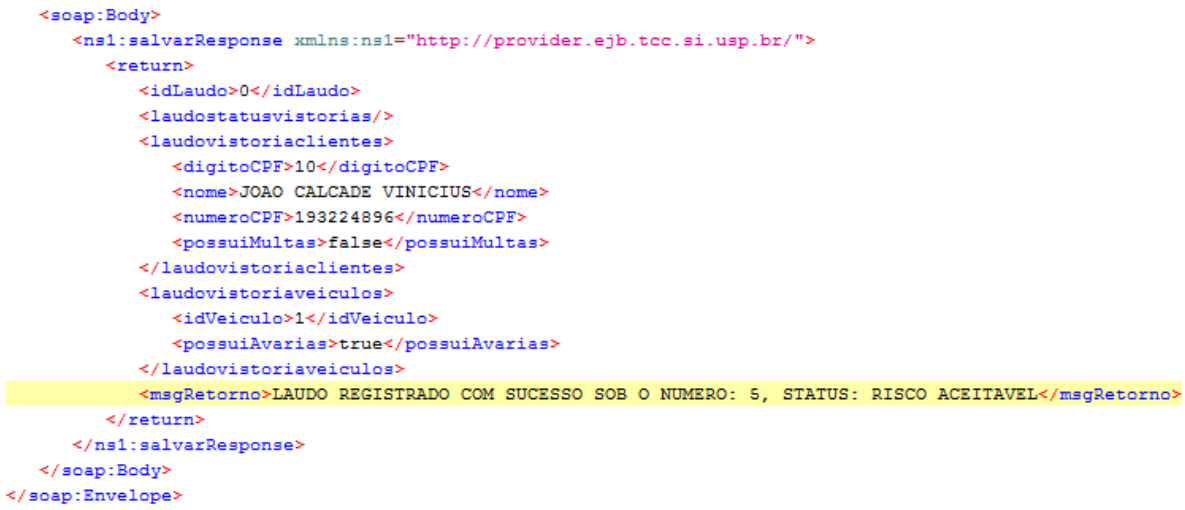

Figura 23 *- SOAP de saída Teste 4 (Informar número e status do laudo registrado)*

Fonte: Elaboração própria

5.2 Estudo de Caso

5.2.1 A Empresa Pesquisada

A Seguradora Alfa foi fundada há mais de 70 anos e emprega diretamente mais de 10 mil funcionários, conta com cerca de 100 escritórios comerciais no país, além de uma unidade internacional. Em 2014 apresentou um lucro líquido de mais de R$ 500 milhões.

Os trabalhos de pesquisa junto a Seguradora Alfa consistiram na realização de diversas reuniões presenciais com as áreas de Arquitetura de TI e Infraestrutura e a própria área cliente, potencial patrocinadora do projeto. Em conjunto com as áreas de TI foram definidos todos os padrões e boas práticas que seriam adotados para o uso de SOA no projeto. Já na parte voltada ao negócio foi apresentado um plano de negócio com o custo estimado do projeto *versus* o custo atual para manutenibilidade de infraestrutura e também os eventuais gastos com retrabalhos de laudos que poderiam ser resolvidos dentro da solução proposta (dados inconsistentes, por exemplo), apresentada estas variáveis foi possível calcular o ROI (sigla do inglês *Return on Investiment* ou Retorno Sobre o Investimento) estimado do projeto, conquistando desta forma o patrocínio do mesmo pela organização.

5.2.2 Visão da Arquitetura Atual

Atualmente o serviço para vistoria em veículos da Seguradora Alfa é realizado por três grupos de usuários distintos:

- **Vistoriadores:** funcionários diretos da Seguradora Alfa, fazem uso de uma aplicação acessada através de um *Pocket* PC, baseado em uma versão antiga do *Microsoft Windows CE*;

- **Empresas Parceiras:** grandes empresas que realizam, de forma terceirizada, serviços de vistoria para diversas seguradoras ou mesmo concessionárias de veículos. Utilizam-se de uma aplicação "*desktop*" baseada em *Windows 2000*;

- **Autônomos:** pessoas físicas que atuam de forma independente e prestam serviços para diversas seguradoras. O preenchimento do laudo é através de um simples arquivo de texto no formato CSV (da sigla *Comma-separated values*, ou seja, valores separados por vígula), não possuindo uma interface amigável ao usuário, que preenche o laudo de forma similar a uma planilha eletrônica.

A Figura 24 ilustra em detalhes o modelo atual desta arquitetura. Podemos observar que existem três aplicações distintas na camada cliente, com duas delas realizando a tarefa de validação de dados (no *Pocket* e no PC das empresas parceiras) enquanto a aplicação utilizada pelos autônomos só possui a função de validação de dados no lado servidor, através de uma aplicação *batch*, ou seja, a validação é realizada em grandes lotes e o usuário só terá a resposta após a conclusão de todo processamento que, no geral, ocorre somente no dia posterior ao envio do laudo pelo cliente. Além disto, as três aplicações compartilham da falta de um requisito extremamente sensível ao negócio: uma camada para verificar a consistência dos dados, que na sua falta permite que muitos laudos sejam cadastrados com informações relacionadas a veículos já inativos ou sem aceitação, gerando problemas para toda cadeia de processos de negócios da seguradora.

Cabe aqui também a análise das questões relacionadas à infraestrutura: devido a limitações técnicas da época, a aplicação atual só funciona em redes locais, tornando-se obrigatório o uso de uma conexão do tipo VPN (*Virtual Private Network*) que possibilita, de forma "virtual", o acesso ao ambiente interno da Seguradora Alfa de forma quase transparente ao usuário, porém

está passível de vulnerabilidades de segurança, devido a problemas conhecidos de sistemas baseados em *Windows* CE e 2000 que não são mais suportados pelo fornecedor.

Além dos problemas relacionados ao suporte do *software* em si há também uma maior deficiência no quesito *hardware*: com o mercado de *smartphones* cada vez mais em alta o aparelho utilizado atualmente pela Seguradora Alfa - um antigo *Pocket* PC de 2008 - tem ficado cada vez mais escasso no mercado.

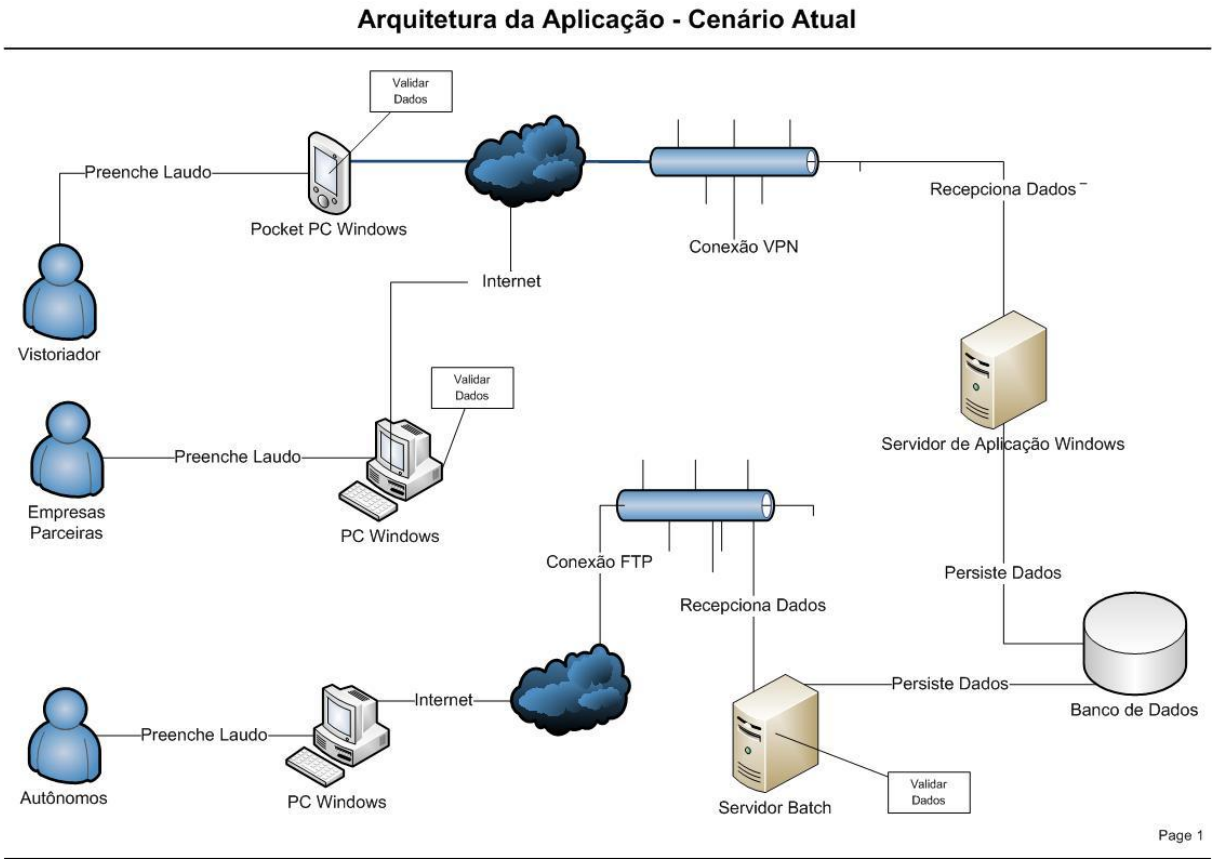

Figura 24 *- Arquitetura da Aplicação - Cenário Atual*
Fonte: Elaboração própria

Diante do cenário exposto pode-se elencar em tópicos os principais problemas enfrentados pela Seguradora Alfa diante de sua arquitetura atual de aplicação:

- Redundância e conflitos no próprio processo de negócio e na cadeia de processos dependentes devido as diferentes customizações existentes nas plataformas clientes;

- Custo elevado nas manutenções e desenvolvimentos devido a redundância de serviços;
- Vulnerabilidades de segurança devido ao uso de uma rede VPN em sistemas baseados em *Windows* CE e 2000;
- *Hardware* utilizado pelo cliente defasado em relação a grande gama de *smartphones* com sistemas operacionais mais robustos e atuais;
- Perda de oportunidades de negócio por não possuir uma plataforma de fácil integração para uso de potenciais parceiros;

5.2.3 Visão da Arquitetura Proposta

Expostos os principais problemas vivenciados pela empresa diante da arquitetura atual uma solução orientada a serviços mostrou-se como a decisão mais consistente para a construção do novo desenho de arquitetura. A tabela 4 realiza um paralelo entre as características de SOA estudadas e sua relação de apoio aos problemas elencados na seção anterior. A figura 25 ilustra o modelo de arquitetura proposto.

Tabela 3 *- Problemas x Características de SOA*

	Características de SOA				
Problemas	**Interoperabilidade**	**Barramento de Serviços**	**Fraco Acoplamento**	**Contrato de Serviços**	**Reusabilidade**
Redundância e conflitos no próprio processo de negócio e na cadeia de processos dependentes			**X**		**X**
Custo elevado nas manutenções e desenvolvimentos			**X**		**X**
Vulnerabilidades de segurança		**X**		**X**	
Hardware utilizado pelo cliente defasado	**X**				
Perda de oportunidades de negócio				**X**	

Fonte: Elaboração própria

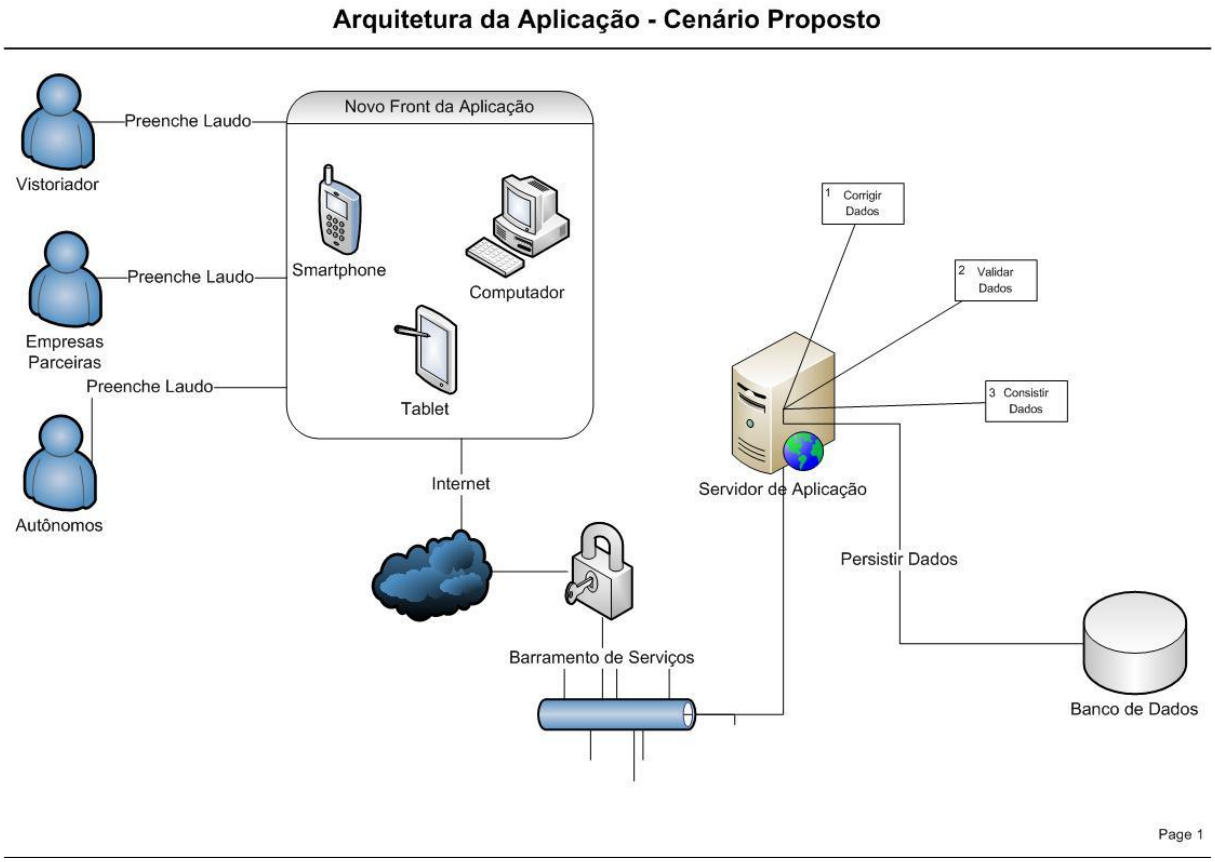

Figura 25 *- Arquitetura da Aplicação - Cenário Proposto*
Fonte: Elaboração própria

A nova proposta de arquitetura para a Seguradora Alfa foi radical: diante da análise do cenário atual era necessário construir uma solução de sistema de informação a partir do zero, eliminando a infraestrutura obsoleta na parte cliente e também no lado servidor.

O objetivo do projeto estudado na empresa era atingir ao menos o nível 4 de maturidade em SOA, desta forma um *application frontend* foi construído e apresentado como uma nova aplicação *web* aos clientes e o serviço central para o registro do laudo de vistoria incorporou as funcionalidades de correção, validação e consistência dos dados, permitindo uma resposta *online* aos clientes. Além do próprio processo de negócio também ter passado a consumir outros serviços da organização, como a consulta na base de veículos com sinistro.

O contrato do serviço a ser desenvolvido também foi oferecido para as empresas parceiras, para que pudessem estudar uma possível integração com a plataforma que elas já trabalhavam com outras empresas congêneres.

5.2.4 Resultados Específicos

Conforme exposto anteriormente, o projeto teve seu apoio pela organização, em especial, devido a potencial economia financeira que geraria em um futuro próximo. Nesta seção serão apresentados os resultados técnicos do projeto, deixando de lado a esfera de discussão e resultados baseados na questão monetária.

A construção de toda a arquitetura do cenário proposto, bem como o desenvolvimento da parte de serviço e do novo *application frontend* foram concluídos em 11 meses e após a finalização dos testes unitários, integrados e de aceite, que tiveram mais 3 meses de duração, definiu-se uma estratégia para a implantação do projeto em fases, possibilitando desta forma uma análise comparativa com os usuários que ainda faziam uso da plataforma antiga em relação aos usuários da nova plataforma. A tabela 3 realiza este comparativo com base em laudos registrados neste período de transição.

Tabela 4 *- Comparativo entre o uso da nova plataforma versus antiga*

	Nova Plataforma	**Antiga Plataforma**
Laudos com dados inválidos (*)	0	98
Laudos com dados inconsistentes (*)	0	123
Reenvio de laudos corrigidos (**)	0,5	18

(*) Para cada mil laudos

(**) Tempo gasto em horas

Destaca-se aqui também a forte adesão das empresas parceiras, que conseguiram integrar o serviço desenvolvido com as versões atuais de suas aplicações com poucas dificuldades, uma vez que o contrato do serviço estava bem definido. As empresas menores optaram por utilizar o *application frontend* que foi desenvolvido para o projeto.

6 Discussão

Como visto na seção de resultados, a adoção de SOA demonstrou-se como uma solução eficiente para o êxito do projeto estudo de caso. Os problemas decorrentes do sistema anterior foram praticamente sanados em 100%, graças à abordagem de correção, validação e consistência dos dados. Para a organização, SOA foi de fato, uma escolha que teve grande aderência ao processo de negócio para laudos de vistorias.

Laudos anteriormente cadastrados com dados inválidos, através dos usuários autônomos e laudos com textos ilegíveis - problema latente de todas as interfaces - foram resolvidos, trazendo também agilidade para toda a cadeia de processo de negócio da Seguradora Alfa, uma vez que as informações dos laudos são compartilhadas por diversas áreas.

O desenvolvimento do protótipo corroborou para a ideia inicial de que a abordagem de correção, validação e consistência dos dados poderia ser oferecida como um serviço único e centralizado. Através dos testes realizados pode-se notar que, através do atributo "*msgRetorno*" o serviço mantém uma constante comunicação com o usuário, informando eventuais erros e/ou inconsistências encontradas. Além disto, o fato do serviço estar baseado em uma comunicação do tipo síncrona trouxe um ganho em especial aos usuários do grupo de autônomos, que dependiam de um processamento batch realizado somente no final do dia.

Com a centralização de todos os requisitos, questões relacionadas à manutenção e evolução do sistema também tiveram um ganho significativo, eliminando duplicidades de códigos de diferentes linguagens que a organização se via obrigada a manter. Desta forma as aplicações passaram a se basear em serviços distintos e desacoplados e a informação passou a ser fornecida unicamente como serviço, tendo assim, aderência as dimensões de visão de negócio, métodos, aplicações, arquitetura e informação do nível 4 de maturidade SOA do OSIMM.

7 Conclusão

O trabalho apresentou, conforme proposto, o emprego da Arquitetura Orientada a Serviços para sistemas baseados em processos de negócio. Com o suporte da prova de conceito foi possível analisar e compreender melhor o ganho tido com o projeto na prática, bem como o seu próprio funcionamento.

Também através do desenvolvimento da prova de conceito e das pesquisas realizadas no levantamento bibliográfico foi possível constatar que a Arquitetura Orientada a Serviços e a Programação Orientada a Objetos são, de fato, conceitos complementares e que foram fundamentais para o êxito do projeto.

Desta forma pode-se concluir que o trabalho conseguiu atingir seus objetivos iniciais, fruto da realização de uma PoC em conjunto a um estudo de caso.

Por fim, vale ainda ressaltar que, diante do ganho obtido através dos resultados apresentados, uma abordagem SOA desta característica pode-se mostrar eficiente e também eficaz para diferentes processos de negócio que possuam tal aspecto, fator este inclusive em análise para projetos futuros da organização estudada. Desta forma, espera-se que trabalhos futuros possam fazer uso desta pesquisa/estudo de caso como apoio.

Referências Bibliográficas

CASTELA, Nuno; TRIBOLET, José. Representação As-Is em Engenharia Organizacional. **5a CAPSI, Lisboa, Portugal (Novembro 3-5, 2004),** 2004. APA.

BLOOR, Robin et al. **Service oriented architecture for dummies**. John Wiley & Sons, 2009.

BORIS, Lublinsky. Defining Soa As An Architectural Style, 2007. Disponível em <http://www.ibm.com/developerworks/library/ar-soastyle>

Acesso em 6 set. 2014.

BRASIL, Haroldo Guimarães. A empresa e a estratégia da terceirização. **Revista de Administração de Empresas**, v. 33, n. 2, p. 6-11, 1993.

CAMP, Robert C. Benchmarking dos processos de negócios. **Rio de Janeiro: Editora Qualitymark**, 1996.

CRUZ, T. Sistemas, métodos & processos: administrando organizações por meio de processos de negócio. São Paulo: Atlas, 2005.

ERL, Thomas. SOA Princípios de design de serviços. Pearson Prentice Hall, São Paulo, 2009.

GONCALVES, Antonio. **Beginning Java EE 6 with GlassFish 3**. Apress, 2010.

HASS, Hugo Designing the architecture for Web services, 2003. Disponível em <http://www.w3.org/2003/Talks/0521-hh-wsa/Overview.html>

Acesso em 22 out. 2014.

ISO, NBR. IEC 9126-1. **Engenharia de software - Qualidade de produto. Parte**, v. 1, 2003.

JOSUTTIS, Nicolai M. SOA na Prática, A Arte da Modelagem de Sistemas Distribuídos. **Rio de Janeiro, RJ: Jacaré**, 2008.

KANKANAMGE, Charitha. **Web services testing with soapUI**. Packt Publishing Ltd, 2012.

KRAFZIG, Dirk; BANKE, Karl; SLAMA, Dirk. **Enterprise SOA: Service-Oriented Architecture Best Practices**. Indianapolis: Prentice Hall, 2005.

MCCARTHY, Tim. Configuring UDDI Services, 2004. Disponível em <http://windowsitpro.com/windows-server/configuring-uddi-services>

Acesso em 28 mar. 2015

PERUMAL, Interoperability: The Next Big Thing for Smart Homes, 2008. Disponível em <http://hometoys.com/emagazine.php?url=/ezine/08.06/perumal/interoperability.htm#.UXGMmLWsiSo>

Acesso em 28 mar. 2015.

PINHEIRO, José Maurício Santos. Prova de Conceito (PoC) no Projeto de Redes de Computadores, 2010. Disponível em

<https://desmontacia.wordpress.com/2010/12/21/prova-de-conceito-poc-no-projeto-de-redes-de-computadores/>

Acesso em 05 mai. 2015.

PRESSMAN, Roger S. **Engenharia de software**. McGraw Hill Brasil, 2011.

SOUZA, Cleidson. Conceitos de Orientação a Objetos. 2013.

Workflow Management Coalition. "The Workflow Management Coalition Specification - Terminology & Glossary". Document Number WFMC-TC-1011, Feb 1999.

THE OPEN GROUP. About Us, 2012. Disponível em <http://www.opengroup.org/aboutus> Acesso em 21 jul. 2015.

THE OPEN GROUP. **OSIMM Version 2 Technical Standard:** The Model, 2011. Disponível em <http://www.opengroup.org/soa/source-book/osimmv2/model.htm> Acesso em 21 jul. 2015.

WORLD WIDE WEB CONSORTIUM et al. Simple object access protocol (SOAP) 1.1. 2000.