ANDRÉ EDUARDO FAUSTINO

UTILIZANDO SOA NA CONSTRUÇÃO E EVOLUÇÃO DE SISTEMAS DE SEGURANÇA DE INFORMAÇÃO VOLTADOS AO CONTROLE DE ACESSO FÍSICO

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para conclusão do curso de MBA em Tecnologia de Software.

São Paulo
2015

ANDRÉ EDUARDO FAUSTINO

UTILIZANDO SOA NA CONSTRUÇÃO E EVOLUÇÃO DE SISTEMAS DE SEGURANÇA DE INFORMAÇÃO VOLTADOS AO CONTROLE DE ACESSO FÍSICO

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Profa. Dra. Maria Alice Grigas Varella Ferreira

São Paulo
2015

DEDICATÓRIA

Aos meus pais, Maria e Ademar, que nunca mediram esforços para me oferecer uma vida digna e sempre apoiada pelos estudos.

AGRADECIMENTOS

À Jesus Cristo, que através da sua infinita graça, iluminou todos os meus dias durante a minha trajetória neste MBA.

À minha orientadora Profa. Dra. Maria Alice Grigas Varella Ferreira, pela orientação e dedicação durante toda a fase de elaboração desta monografia.

Aos professores do MBA em Tecnologia de Software, pela competência e seriedade na trasmissão de conhecimento, que, com toda certeza, me fizeram evoluir profissionalmente.

À minha mãe Maria Perpétua Nogueira pela compreensão e apoio durante os momentos bons e difíceis desta jornada.

À professora de Inglês da Poli, Cristina Borba, que gentimente realizou a revisão do Resumo e *Abstract*, contribuindo de forma significativa, com a qualidade deste trabalho.

À Universidade de São Paulo – USP que a décadas, vem oferendo cursos com extrema excelência, em diversos níveis de especialização e áreas do conhecimento.

Nenhum homem terá uma chance para desfrutar um triunfo permanente se não começar por olhar-se num espelho para descobrir a causa real de todos os seus erros.

(Napoleon Hill)

RESUMO

A Segurança de Informação, seja para controlar o acesso lógico aos dados em sistemas informatizados, seja para controlar o acesso físico de funcionários ou pessoas comuns aos diferentes locais de uma empresa, vem se tornando um investimento primordial para todas elas. Para evitar-se fraudes e acesso indevido a dados confidenciais e, muitas vezes, críticos para a estratégia empresarial ou furtos de bens patrimoniais - em locais de acesso físico -, as corporações se valem da evolução tecnológica, empregando os mais diferentes equipamentos como senhas, equipamentos biométricos, câmeras, certificados dentre outros. O objetivo deste trabalho é fazer uma avaliação do uso de Arquitetura Orientada a Serviços (SOA) no desenvolvimento e manutenção/evolução de sistemas de controle de acesso de ambientes corporativos. Quanto aos aspectos arquiteturais, o foco se dá no estudo sobre a Orquestração de Web Services, apresentando ao final um estudo de caso sobre como aplicar tais conceitos ao controle de acesso de físico de usuários.

Palavras-chave: Controle de Acesso Físico e Lógico, Arquitetura Orientada a Serviços (SOA), MAPOS – Método de Análise e Projeto Orientado a Serviços.

ABSTRACT

Information Security, be it to control logical access to data in computer systems, be it to control physical access of employees or persons using the different locations of a company, has become a major investment for all. To avoid fraud and unauthorized access to confidential data, often critical to business strategy or assets of thefts - in physical access sites - corporations make use of technological developments, employing the most different equipment such as passwords, biometric equipment, cameras, certificates among others. The purpose of this work is to evaluate the use of Service Oriented Architecture (SOA) development and maintenance / evolution of access control systems of enterprise environments. As for the architectural aspects, the focus lies on Web Services Orchestration. A case study illustrates how to apply these concepts to control users physical access.

Keywords: Physical and Logical Access, Service-Oriented Architecture (SOA), Control, Service-Oriented Analysis and Design Method.

LISTA DE ILUSTRAÇÕES

Pág.

LISTA DE TABELAS

Pág.

LISTA DE ABREVIATURAS E SIGLAS

BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
CORBA	Common Object Request Broker Achitecture
DCOM	Distributed Componect Object Model
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IDE	Integrated Development Environment
OASIS	Organization for the Advancement of Structured Information Standards
SGBD	Sistema de Gerenciamento de Banco de Dados
SI	Sistema de Informação
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmition Control Protocol
UDDI	Universal Description, Discovery, and Integration
WSCI	Web Services Choreography Interface
WSDL	Web Services Description Language
WS-BPEL	Business Process Execution Language for Web Services
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSD	XML Schema Definition Language
XSLT	Extensible Stylesheet Language Transformation
MAPOS	Método de Análise e Projeto Orientado a Serviços

SUMÁRIO

Pág.

1. INTRODUÇÃO

Neste capítulo serão abordados os objetivos, as motivações que levaram a escolha do tema em questão, as justificativas e como serão estruturados os assuntos que serão apresentados.

1.1 Motivações

A Segurança de Informação, seja para controlar o acesso lógico aos dados em sistemas informatizados, seja para controlar o acesso físico de funcionários ou pessoas comuns aos diferentes locais de uma empresa, vem se tornando um investimento primordial para todas elas. Para evitar-se fraudes e acesso indevido a dados confidenciais e, muitas vezes, críticos para a estratégia empresarial ou furtos de bens patrimoniais - em locais de acesso físico -, as corporações se valem da evolução tecnológica, empregando os mais diferentes equipamentos como senhas, equipamentos biométricos, câmeras, certificados dentre outros.

Diante deste cenário, as empresas necessitam comprar ou desenvolver sistemas informatizados que sejam capazes de realizar o controle de acesso aos sistemas computacionais utilizados pelos funcionários, ou realizar o controle de acesso físico aos ambientes restritos da mesma por parte de qualquer pessoa.

Neste sistema, há importantes fluxos de trabalho (*workflows*), que precisam ser controlados para a utilização dos mesmos: autenticação de usuário, seja em uma catraca que permita que o mesmo acesse um ambiente dentro da empresa (acesso físico), ou num computador, para obter acesso a um sistema informatizado, que o usuário precisa utilizar (acesso lógico).

Desta forma, inúmeros recursos tecnológicos, como: redes de computadores, Sistemas de Gerenciamento de Banco de Dados, câmeras, monitores dentre outros, precisam ser controlados pelos sistemas de controle de acesso através de serviços

distribuídos na arquitetura de computadores dentro da empresa. Neste contexto, pode-se destacar a utilização da Arquitetura Orientada a Serviços (SOA).

SOA (*Service-Oriented Architecture*) é um estilo arquitetural que oferece às empresas uma maneira de projetar, desenvolver, distribuir e gerenciar sistemas empresariais, onde as necessidades tecnológicas da organização são analisadas sob o ponto de vista de serviços, os quais são projetados e desenvolvidos para atingir os objetivos de negócio em sua plenitude (BHALLAMUDI; TILLEY, 2008).

Segundo Alwadain *et al.* (2010, p. 2), "SOA é um dos estilos arquiteturais mais difundidos nos dias atuais. Ele considera cada negócio ou sistema como um fornecedor de serviços oferecendo um ou mais serviços.". Cada serviço encapsula uma ou mais atividades necessárias para o funcionamento de um sistema de informação, expondo-os através de interfaces (contratos), permitindo que seja consumido por outros softwares.

Através da adoção de SOA, é possível que sistemas automatizados, mesmo não projetados para favorecer a integração entre si, sejam representados sob um único ponto de vista em relação ao ambiente em uma empresa, canalisando as funcionalidades para obter maior eficiência e usabilidade de tais recursos (SWEENEY, 2010).

SOA, basicamente é constituído pelos serviços que são unidades de lógica que representam uma ou mais atividades necessárias para o processamento de dados em um sistema de informação. Tais atividades podem conter parte ou o total dos passos necessários para atingir um determinado objetivo que se pretende em um SI (Sistema de Informação) (ERL, 2005).

Neste trabalho, será estudado como a utilização da Orientação a Serviços pode contribuir para e construção de sistemas informatizados com fluxos de trabalhos complexos, como é o caso de um sistema de controle de acesso físico num ambiente corporativo.

O registro e recuperação das informações armazenadas em um Sistema de Gerenciamentos de Banco de Dados (SGBD) em um Sistema de Controle de Acesso, deve ocorrer em tempo real, exigência que deve ser cumprida a risco por sistemas com esta finalidade, pois o tempo de resposta deve ser, praticamente, entendido como instantâneo. A fim de cumprir essas exigências, a utilização de uma Arquiteturua Orientada a Serviços é vista como uma forma de cumprir tais requisitos no contexto do controle de acesso, torna motivadora a pesquisa que será realizada nesta monografia.

1.2 Objetivo

O objetivo deste trabalho é fazer uma avaliação do uso de Arquitetura Orientada a Serviços no desenvolvimento e manutenção/evolução de sistemas de controle de acesso de ambientes corporativos. Com a evolução tecnológica, tanto do software, quanto do hardware, inúmeros sistemas de software e equipamentos eletrônicos têm sido criados, para que, em conjunto, forneçam uma solução integrada, que permita o controle de acesso, seja físico (acesso aos locais dentro da empresa), ou lógico (acesso aos sistemas de informação).

Quanto aos aspectos arquiteturais, o foco se dá no estudo sobre a Orquestração de *Web Services* através da utilização da linguagem WS-BPEL[1] (Web Service Business Process Execution Language), apresentando ao final um estudo de caso sobre como aplicar tais conceitos para a construção de um Orquestrador de *Web Services*, voltado ao controle de acesso de físico de usuários.

1.3 Justificativas

A utilização de sistemas de software em conjunto com uma série de equipamentos digitais, como câmeras, filmadoras, leitores biométricos entre outros, tem se tornado

---

[1] **WS-BPEL: linguagem para Orquestração de *Web Services* associados a um Processo de Negócios (OASIS, 2007).**

cada vez mais frequente pelas empresas, a fim de garantir o controle de identificação de usuários e segurança patrimonial nas mesmas.

Desta forma, um conjunto de sistemas de software (*Web Services*, sistemas embarcados, Aplicativos Web, API's), são desenvolvidos para que, entre si, realizem uma série de tarefas que permitem o controle de acesso e segurança na empresa.

Neste contexto, é necessário a centralização e cordenação destes inúmeros sistemas informatizados que interagem entre si e tal centralização pode ocorrer através da orquestração destes sistemas, a fim de atingir as metas que devem ser alcançadas e que vão, desde autorizar o acesso a um determinado recurso de sistema, até registrar as imagens do monitoramento feito pelas câmeras nas bases de dados.

Através da utilização de SOA e a Orquestração de Web Services, é possível centralizar a execução de todos os Web Services associados às atividades de um fluxo de trabalho, utilizando um linguagem específica para tal propósito, como o WS-BPEL.

Outra característica interessante de SOA é que os serviços podem ser vistos de forma mais próxima aos negócios da organização, permitindo que seu desenvolvimento seja mais facilmente efetuado e, posteriormente, mantido ao longo de seu ciclo de vida (FUGITA; HIRAMA, 2012).

Uma das características mais atraentes desse estilo arquitetural é, segundo Verjus e Pourraz (2007), a habilidade subjacente de tal arquitetura ser expansível, e porque a ideia subjacente de SOA é de que os *Web Services* estão fracamente acoplados e de que SOA pode ser adaptável ao seu ambiente. O baixo acoplamento é obtido através do encapsulamento e troca de mensagens. A neutralidade tecnológica é obtida através de mecanismos padronizados e linguagens adequadas que permitem aos serviços exportarem informação suficiente, de forma que os clientes em potencial possam descobri-los e se conectarem a eles (Papazoglou, 2003 *apud* Verjus e Pourraz, 2007). Desta perpectiva, defendem esses autores, o foco dos novos projetos deve estar no projeto evolutivo e na arquitetura focada em qualidade.

Os *Web Services* devem ser partilhados entre grupos de desenvolvedores e múltiplas organizações. Este partilhamento permite a redução de custos de desenvolvimento; por outro lado, muitas empresas estão desenvolvendo tais serviços para uso de outras empresas, comerciais e governamentais. A possibilidade de uso compartilhado é uma forma importante de reuso de software. Exemplos de tais serviços podem ser encontrados na busca de endereços através do CEP, oferecido pelos Correios e de pendências financeiras na SERASA. Cada vez mais provedores estão oferecendo serviços dessa natureza. No que diz respeito à serviços de identificação pessoal, inúmeras tecnologias estão disponibilizadas pelos diversos provedores: uso de impressões digitais, identificação por voz, por íris etc. A tendência é que esses métodos de identificação fiquem a cada dia mais diversificados. Por exemplo, os celulares mais modernos em 2015 já utilizam digitais como senha.

Por outro lado, essa tecnologia está bastante consolidada através de organizações como OASIS[1] - Organization for the Advancement of Structured Information Standards - e W3C[2] - World Wide Web Consortium. Também, métodos para desenvolvimento de aplicações SOA e ferramentas adequadas estão sendo disponibilizadas às equipes de desenvolvimento.

Assim, esses recursos estão se tornando essenciais para atingir e cumprir com as metas de sistemas de segurança, atendendo o controle de acesso físico e lógico de um sistema computacional ou de uma determinada área física dentro da empresa.

1.4 Metodologia

A seguinte sequência de atividades foi aplicada para o desenvolvimento dessa monografia:

- Estudar e avaliar a literatura relacionada aos sistemas de controle de acesso para conhecer suas características e problemas, concentrando estudos,

[1] **https://www.oasis-open.org/org**
[2] **http://www.w3.org/**

principalmente, em trabalhos que abordassem a discussão de problemas práticos;

- Definir o estilo de arquitetura a ser utilizado; frente ao estado atual de desenvolvimento tecnológico optou-se pela Arquitetura Orientada a Serviços;

- Estudar a tecnologia em que se baseia SOA. Isso envolveu estudos relacionados com OASIS e W3C e seus padrões, bem como artigos relacionados ao tema;

- Aplicar os conhecimentos adquiridos num estudo de caso. Para isso adotou-se um caso real, vivenciado pelo autor, em que uma empresa decidiu automatizar o sistema de portaria, com vista a posteriores atualizações em direção a tecnologias mais modernas;

- Escolher um modelo de processo para representar a lógica de negócios que será suportada pela solução. A escolha recaiu no MAPOS - Método de Análise e Projeto Orientado a Serviços (FUGITA, 2009);

- Desenvolver o estudo de caso.

1.5 Estrutura do Trabalho

O Capítulo 1, INTRODUÇÃO, apresenta as motivações, o objetivo, as justificativas e a estrutura do trabalho.

O Capítulo 2, REVISÃO BIBLIOGRÁFICA, apresenta uma justificativa sobre a escolha de cada uma das fontes bibliográficas e em quais partes do estudo apresentado nesta monografia, tais fontes de estudo serão necessárias para complementação dos cápitulos.

No Capítulo 3, FUNDAMENTAÇÃO TEÓRICA, é realizada a conceituação de cada um dos assuntos que fornecem base para a construção desta monografia, como:

SOA, WS-BPEL, *Web Services* e Segurança da Informação, destacando os Sistemas de Controle de Acesso.

O Capítulo 4, ESTUDO DE CASO, apresenta um processo para construção do orquestrador de serviços voltado ao controle de acesso às dependências de uma empresa, através da utilização da linguagem WS-BPEL, apresentando um detalhamento dos principais processos de negócio relacionados ao estudo de caso.

O Capítulo 5, ANÁLISE DOS RESULTADOS, descreve os resultados obtidos com a aplicação dos conceitos através do estudo de caso apresentado no capitulo anterior, destacando os pontos forte e fracos observados durante o desenvolvimento do projeto.

O Capítulo 6, CONSIDERAÇÕES FINAIS, descreve uma sintese do estudo realizado, dos beneficios obtidos através da aplicação dos conceitos através do estudo de caso e sobre estudos futuros relacionados ao estudo em questão.

REFERÊNCIAS relaciona as fontes bibliográficas que serviram como base para fundamentação teórica e prática abordadas na monografia.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo são abordadas cada uma das fontes bibliográficas escolhidas para a pesquisa realizada neste trabalho, e a ordem em que o estudo foi conduzido, a partir da fundamentação teórica que serviu de base para a pesquisa proposta. É claro que muitas das fontes foram utilizadas em paralelo, tirando-se o melhor de cada uma delas.

Para a conceituação sobre Sistemas de Controle de Acesso, foram utilizados os livros: Fundamentos de Segurança de Sistemas de Informação, de Kim e Solomon (2014), Access Control in Data Management Systems, de Elena Ferrari (2010) e Access Control Systems, de Messaoud Benantar (2006). Todos os principais conceitos sobre segurança de informação aplicada ao controle de acessos, bem como os diagramas esquemáticos que exemplificam o controle de acesso, foram obtidos a partir das referências citadas. Tais definições são necessárias para análise e projetos dos processos de negócio associados ao controle de acesso físico e lógico.

O estudo sobre SOA começou com uma abordagem sobre essa arquitetura, para a qual foram selecionados os livros: Engenharia de Software, de Ian Sommerville (2008), SOA: Concepts, Tecnology, and Design, do Thomas Erl (2005) e o livro Patterns: SOA Design Using WebSphere Message Broker and WebSphere ESB, de Credle *et al.* (2006).

Sommerville (2011), no capítulo Arquitetura Orientada a Serviços, trata sobre os principais fundamentos deste estilo arquitetural, sobre a utilização de *Web Services* e definição de fluxos de trabalho para utilização de composição de serviços, que é um dos temas abordados neste trabalho. Para esta pesquisa, alguns desses fundamentos foram utilizados para comentar sobre a definição de SOA, e como os principais padrões de *Web Services* (SOAP, UDDI, WSDL entre outros), estão ligados a tal estilo arquitetural e porque são importantes para implementação de projetos SOA.

Erl (2005) traz no seu livro um estudo completo sobre SOA e como adotar tal abordagem em projetos que utilizem essa arquitetura. Nesta monografia, foram utilizados os conceitos que tratam sobre as possibilidades de representação que serviços podem assumir e sobre os princípios que norteiam um projeto SOA.

Credle *et al.* (2007) trazem algumas definições sobre a arquitetura de um projeto SOA, a qual foi adaptada neste trabalho para exemplificar o funcionamento de SOA e como os principais componentes e tecnologias de *Web Services* se interrelacionam neste estilo arquitetural.

Para o estudo sobre as Tecnologias de *Web Services*, foram selecionados como principais fontes de pesquisas, os conteúdos publicados na comunidade W3C, a qual é responsável pela padranização e revisões dos principais padrões abordados neste estudo - XML, XSD Schema, SOAP e WSDL (W3C, 2000, 2001, 2004, 2004a).

Além da documentação fornecida pela W3C, foram selecionadas outras obras, como Web Service: Principle and Technology do autor Michael P. Papazoglou (2011), que aborda detalhadamente cada um dos aspectos de *Web Services*. Nesta monografia, tal trabalho foi usado para o estudo sobre o padrão UDDI, por trazer um estudo completo sobre o assunto.

Endrei *et al.* (2004) também abordam, no livro Patterns: Service-Oriented Architecture and Web Services, alguns dos principais conceitos sobre os *Web Services*.

Este trabalho traz um estudo sobre Composição de *Web Services*, para o qual foram selecionados os livros SOA: Concepts, Tecnology, and Design de Thomas Erl (2005) e Engenharia de Software, de Sommerville (2011), o artigo sobre Coreografia e Orquestração de *Web Services*, de Chris Peltz (2003), o artigo Styles of Service Composition – Analysis and Comparison Methods, de Halili *et al.* (2013) e Web Service: Principle and Technology, de Michael P. Papazoglou (2011).

A Orquestração de *Web Services*, que é um dos temas principais desta monografia, é uma das abordagens que existem para compor *Web Services*. Peltz (2003) e Halili *et al.* (2013), trazem em seus artigos, uma definição completa sobre tal abordagem.

Peltz (2003) traz, também, alguns dos principais fundamentos sobre Orquestração e Coreografia de *Web Services*, bem como os padrões e tecnologia associados aos mesmos. Tais conceitos, foram utilizados nesta monografia para exemplificar ambas as abordagens de Composição de *Web Services*.

Para conceituação sobre WS-BPEL foram selecionados os autores Erl (2005) e OASIS (2007), pois são fontes que falam sobre o assunto e trazem os detalhes sobre tal linguagem para orquestração de processos de negócio executável.

Fugita (2009) fala sobre o Método de Análise Orientada a Serviços (MAPOS), cujas fases serão utilizadas no projeto voltado à construção de um Sistema de Controle de Acesso. Fugita e Hirama (2012) completa tal estudo.

3. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, estão apresentados os fundamentos teóricos que serão abordados em cada um dos assuntos que fazem parte da monografia, sendo apresentados na ordem que se segue.

Na seção 3.1, Arquitetura Orientada a Serviços, são abordados todos os conceitos, as vantagens e as tecnologias relacionadas a tal estilo arquitetural e o porquê de estarem sendo adotadas para a construção e integração de sistemas de informação por muitas empresas no mundo todo.

Na seção 3.2, Tecnologias de *Web Services*, são abordados os principais conceitos sobre os padrões, tecnologia e importância da utilização dos mesmos para construção de sistemas computacionais, qual a relação entre os *Web Services* e a arquitetura SOA e por qual motivo vêm sendo amplamente utilizados pelas empresas.

Na seção 3.3, Composição e Orquestração de *Web Services* com WS-BPEL, estão apresentados os conceitos sobre o que é Composição de *Web Services* e as diferenças entre os modelos de Orquestração e Coreografia.

Na seção 3.4 são apresentados fundamentos da linguagem WS-BPEL e quais as etapas necessárias para construir um Orquestrador de *Web Services* utilizando tal especificação.

Na seção 3.5, Sistemas de Controle de Acesso, são apresentados os principais conceitos e fundamentos referentes a Controle de Acesso Físico, destacando os sistemas e equipamentos envolvidos para tal controle de acesso. Em seguida, detalha-se Controle de Acesso Lógico, sua importância, como é realizado, e porque é necessário controlar o acesso aos sistemas.

3.1 Arquitetura Orientada a Serviços (SOA)

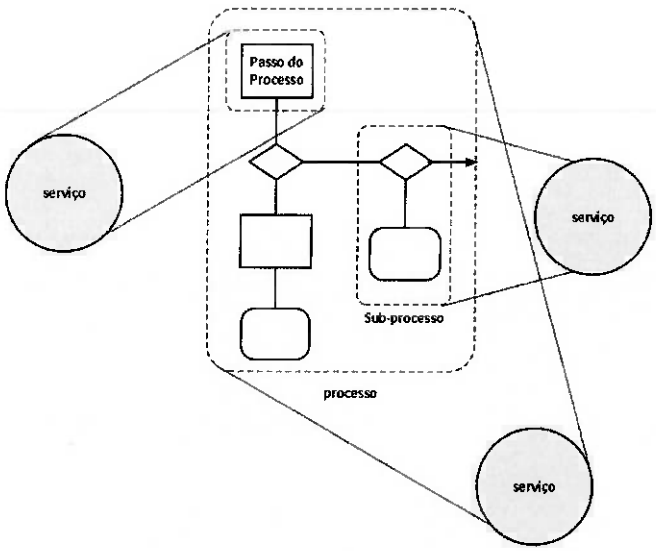
Nesta seção, serão apresentados alguns dos principais conceitos sobre Arquitetura Orientada a Serviços, quais tecnologias podem ser utilizadas para implementá-la e como pode ser utilizada no projeto de sistemas corporativos.

3.1.1 Definição

Arquiteturas Orientadas a Serviços ou SOA, é um um estilo arquitetural voltado à construção de sistemas corporativos, nos quais os componentes são projetados como serviços autônomos, que representam, num alto nível de abstração, os processos de negócio da corporação, permitindo a integração entre sistemas automatizados, seja localmente (em um mesmo domínio), ou localizados em diferentes pontos geográficos (SOMMERVILLE, 2011).

Em SOA, o principal elemento é o serviço, que representa um ou mais processos, encapsulando as unidades funcionais de uma aplicação através do fornecimento de uma interface bem definida e independente de implementação. Segundo Erl (2005), um serviço pode encapsular o passo de um processo, um subprocesso ou todo conjunto de processos, como é mostrado na Figura 1.

Figura 1: Possibilidades de representação que um serviço pode assumir.



Fonte: Erl (2005).

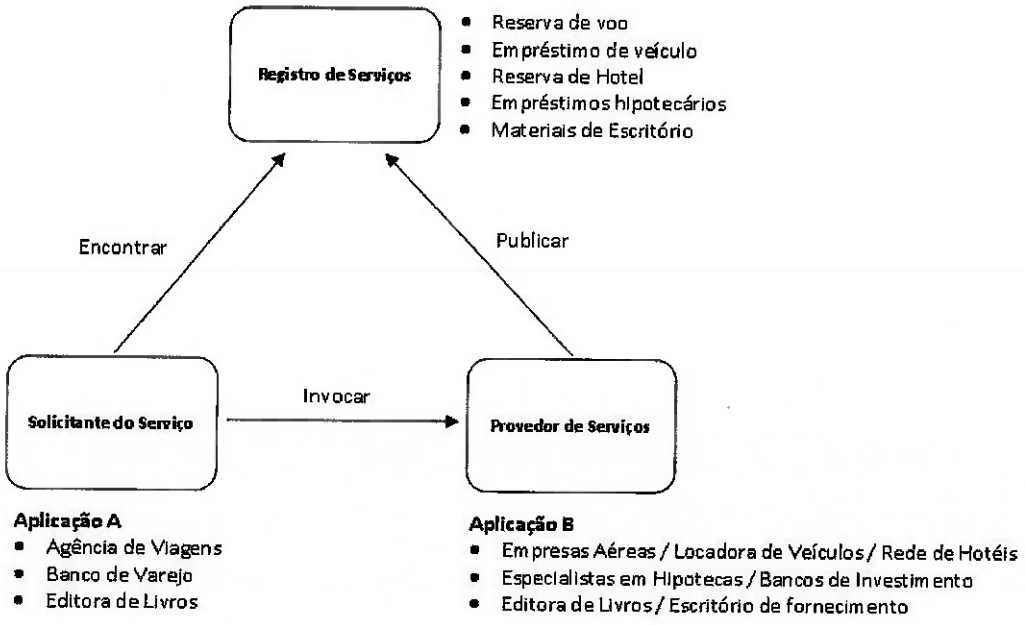
Na figura 1, tem-se um primeiro Serviço que encapsula as atividades de um único passo de um processo, um segundo Serviço que encapsula as atividades de um subprocesso, e um terceiro Serviço que representa o processo principal, o qual engloba o primeiro e o segundo Serviço.

Os serviços devem ser projetados a fim de fornecer funcionalidades independentes de tecnologia ou plataforma, cumprindo com um dos princípios fundamentais em SOA, a interoperabilidade. Através deste princípio, é possível que sistemas legados ou construído em plataformas diferentes (Linux, Windows, Mac etc.), possam trocar informações, permitindo a integração dos mesmos de uma forma mais flexível.

3.1.2 Componentes de uma Arquitetura SOA

SOA consiste em três principais componentes: o Provedor de Serviços, o Registro de Serviços e o Consumidor ou Solicitante de Serviços. Em conjunto, permitem que os serviços sejam registrados, publicados e acessados pelas empresas que estejam utilizando este estilo de arquitetura. A Figura 2, mostra como interagem esses componentes, destacando-se exemplos de uso típico ao lado de cada um.

Figura 2: Elementos que compõem SOA.



Fonte: Do Autor.

O Provedor de Serviços define os serviços e especifica as interfaces para esses serviços, publicando-os no Registro de Serviços. O Registro de Serviços permite que as Interfaces ou Contratos sejam disponibilizadas para um Consumidor de Serviços. O Consumidor de Serviços (Cliente ou Solicitante) encontra a especificação dos serviços e, em seguida, os invocam (CREDLE *et al.*, 2007).

Como exemplificado na Figura 2, Provedores de Serviços de empresas de vários segmentos, por meio da Aplicação B, publicam seus serviços em um Registro de Serviços, para que os mesmos fiquem disponíveis para os seus parceiros comerciais - Consumidores ou Solicitantes do Serviço - que descobrem tais serviços e os acessam através da Aplicação A.

Desta forma, protocolos de comunicação baseados em XML, SOAP, WSDL dentre outros, foram projetados para cobrir todos os requisitos exigidos para implementação de aplicações baseadas em SOA. A tecnologia de *Web Services* fornece um grupo de padrões que permitem cumprir tais exigências, como mostra o Quadro 1.

Quadro 1: Aspectos SOA e os padrões de *Web Services* relacionados.

Aspectos SOA	**Padrões de Web Services**
Tecnologias XML	XML, XSD, XSLT,...
Suporte	WS-Security, WS-Addressing,...
Processo de Negócio	WS-BPEL
Serviço de Mensagem	SOAP
Definição de Serviço	WSDL, UDDI
Transporte	HTTP, SMTP, TCP, ...

Fonte: Sommerviller (2011).

Dentre os aspectos e padrões citados no Quadro 1, pode-se destacar o Registro de Serviços através do protocolo UDDI, o Transporte de Mensagens através de HTTP, a Descrição de Serviço através de WSDL e XML, Protocolos de Comunicação através de SOAP e Processos de Negócio através do WS-BPEL. "Os principais padrões SOA são suportados por uma gama de padrões de suporte que concentram nos aspectos mais especializados de SOA" (SOMMERVILLE, 2011, p. 511).

3.1.3 Modelos de Referência SOA

As duas organizações responsáveis pela padranização de SOA, são a OASIS (*Organization for the Advanced of Structered Information*), organização sem fins lucrativos que trabalha no desenvolvimento de padrões abertos de Tecnologia de Informação (FUGITA, 2009) e o *Open Group*, consórcio industrial para definir software aberto de infraestrutura de computação, que sejam neutros em relação a tecnologias e fornecedores (YANAY, 2010).

De acordo com Yanay (2010), o *Open Group* elaborou um projeto específico para definição de SOA, segundo o qual, SOA é um estilo arquitetural que fornece suporte para orientação a serviço. Segundo a mesma, um serviço é uma representação lógica de uma atividade de negócio que se repete e tem um resultado específico.

OASIS elaborou um modelo de referência de SOA (OASIS SOA REFERENCE MODEL TC, 2006) com a intenção de definir os conceitos relacionados em uma arquitetura SOA e os relacionamentos entre eles, com o propósito de estabelecer uma base conceitual que servirá como guia para a criação de padrões e arquiteturas de referência e implementações (FUGITA, 2009).

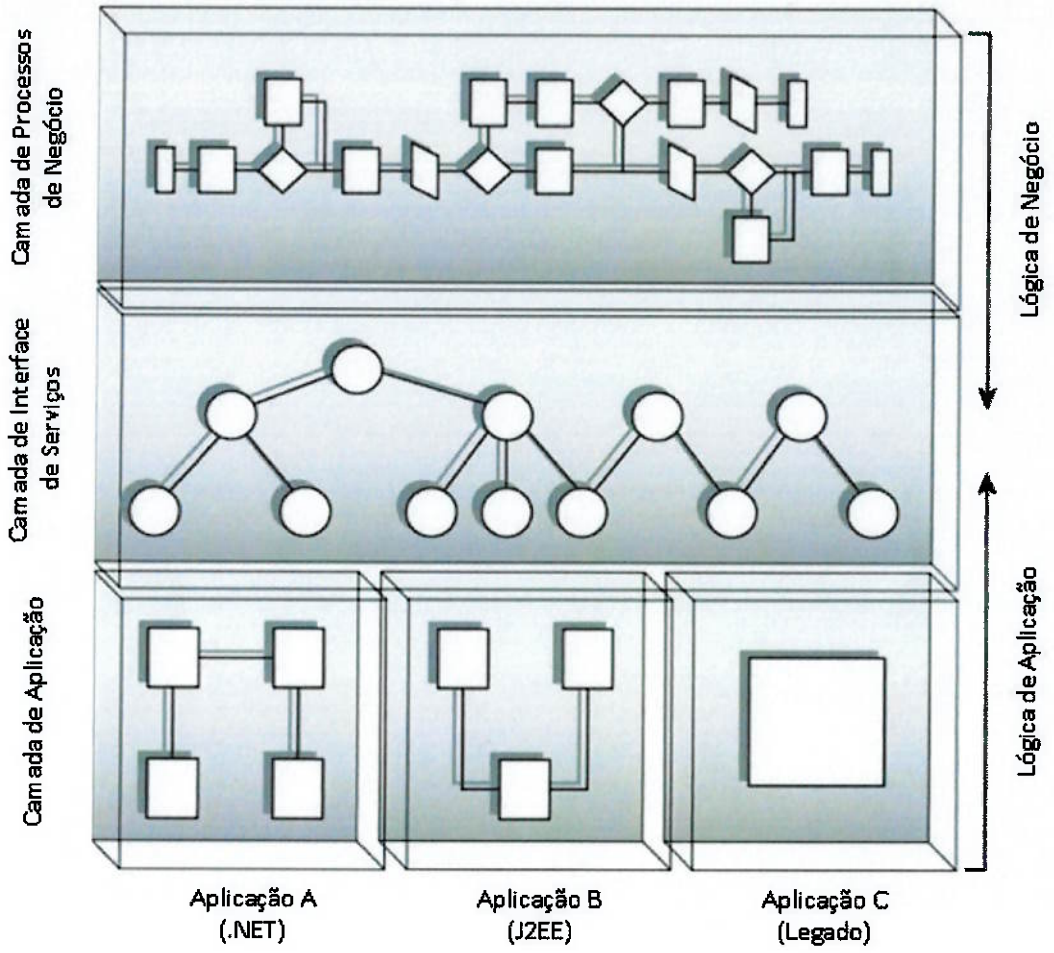
3.1.4 Camadas de Serviço SOA

Em uma arquitetura de software corporativa, as duas principais camadas que representam a Lógica de Negócio e a Lógica de Aplicação, são: a Camada de Processo de Negócio, na qual residem todos os elementos originados a partir dos requisitos das áreas de negócio da corporação e Camada de Aplicação, na qual ficam as aplicações de software que executam as operações, de acordo com os fluxos de processo (ERL, 2005).

A partir dos conceitos introduzidos por SOA, através da utilização de Serviços, uma nova camada chamada Camada de Interface de Serviços foi introduzida entre a Camada de Processos de Negócio e a Camada de Aplicação, produzindo desacoplamento entre as mesmas, criando uma abstração entre os processos de

negócio e as aplicações de software (FUGITA, 2009). Na Figura 3, tem-se um exemplo destas três camadas.

Figura 3: Camada de Interface de Seviços entre as Camadas de Processo de Negócio e de Aplicação.



Fonte: Erl (2005).

Na figura 3, têm-se as camadas de arquitetura SOA, na qual a Camada de Interface de Serviços está posicionada entre a Camada de Processo de Negócio e a Camada de Aplicação, gerando uma abstração entre as mesmas.

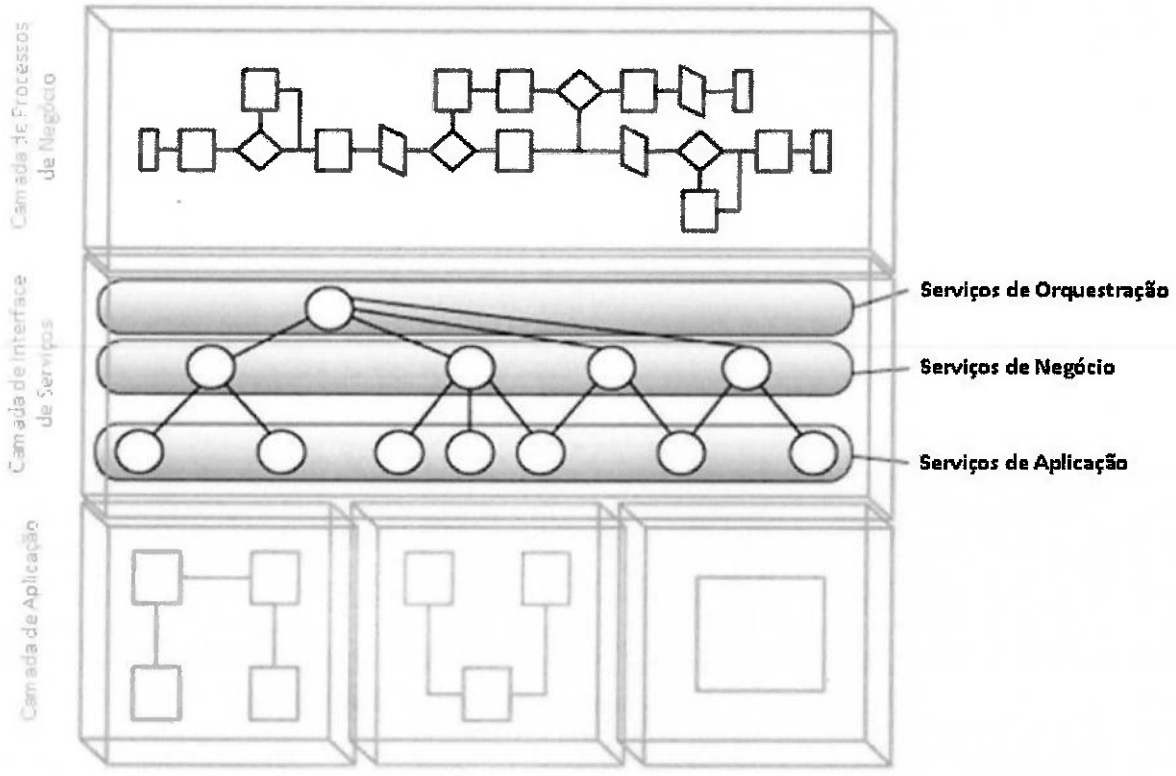
Para melhor representar o tipo e a granularidade dos Serviços, a Camada de Interface de Serviços pode ser dividida em mais três camadas de abstração, sendo elas: a Camada de Serviços de Aplicação, a Camada de Serviços de Negócio e a Camada de Orquestração de Serviços (FUGITA, 2009).

A Camada de Serviços de Aplicação contém os Serviços que oferecem funcionalidades específicas de tecnologia, provendo funções reusáveis e de processamento de dados contidos em sistemas legados e novas aplicações de software construídas (FUGITA, 2009).

A Camada de Serviços de Negócio, contém os Serviços baseados no Modelo de Serviços de Negócio, também chamados de Serviços de Negócio. O objetivo destes Serviços é promover o total alinhamento entre os objetivos de negócio da corporação e os domínios de tecnologia, sendo fundamentais na arquitetura SOA (ERL, 2005).

A Camada de Serviços de Orquestração contém os Serviços de Orquestração, cuja execução está diretamente ligada a um fluxo de trabalho (*workflow*) em um processo de negócio. Os mesmos realizam a composição dos Serviços contidos na Camada de Serviços de Negócio, sendo controlados, geralmente, através de um motor de software (*software engine*) baseado numa linguagem para orquestração como o WS-BPEL. Na figura 4, tem-se o exemplo destas três camadas de serviços.

Figura 4: Camada de Interface de Serviços divida em três Camadas distintas.



Fonte: Erl (2005).

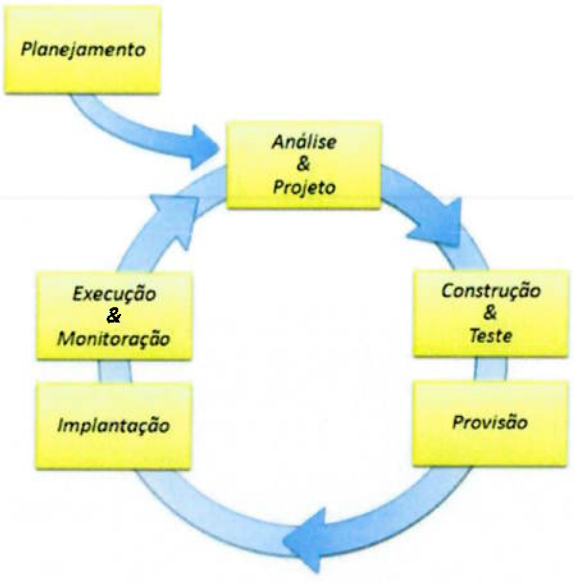
Na figura 4, tem-se a divisão da Camada de Interface de Serviços dividida em Camada de Serviços de Orquestração, Camada de Serviços de Negócio e Camada de Serviços de Aplicação.

3.1.5 Cíclo de Vida SOA

Segundo Papazoglou e Van Den Heuvel (2006) *apud* Fugita (2009), para que seja realizada a especificação, construção e refinamento de processos de negócio a partir de serviços, é necessário que seja utilizado um processo de desenvolvimento baseado em Serviços. Os mesmos descrevem o processo de Ciclo de Vida de Desenvolvimento de *Web Services*, sendo o mesmo baseado parcialmente no RUP (*Rational Unified Process*), como mostra a Figura 5.

Semelhante ao RUP, o método de Análise e Projeto faz parte de um processo cíclico, sendo executado em iterações, nas quais ocorre um processo contínuo de descoberta, criação e implementação. A cada iteração são incorporadas mais funcionalidades aos artefatos da solução, de modo previsível e repetível (FUGITA, 2009). Na Figura 5, se mostra a representação deste ciclo.

Figura 5: Ciclo de Vida de Desenvolvimento de *Web Services*.



Fonte: Papazoglou e Van Den Heuvel (2006) *apud* Fugita (2009).

Descrevem-se a seguir as etapas do ciclo de vida de SOA, conforme Fugita (2009).

- **Planejamento:** fase onde é feita a preparação do projeto da solução, sendo determinada a viabilidade, a natureza e o escopo do projeto;

- **Análise:** fase na qual é realizada a especificação dos requisitos de negócio da aplicação de software, para a qual, são definidos os modelos de processo *as-is*[1] e *to-be*[2];

- **Projeto:** fase na qual são especificadas as interfaces dos Serviços identificados na fase de Análise;

- **Construção:** fase na qual é realizado o desenvolvimento dos serviços e a descrição de suas interfaces e implementações;

- **Teste:** fase na qual são realizados testes funcionais, de desempenho, de interface e composição, com o objetivo de validar se os requisitos de negócio foram satisfeitos;

- **Provisão:** fase onde são tratados os aspectos organizacionais dos serviços, tanto dentro da organização, como nas relações entre mais de uma empresa.

- **Implantação:** fase na qual os novos processos são publicados e expostos para as organizações, aplicações e outros processos, sendo a interface e a implementação dos Serviços publicadas nos repositórios;

- **Execução:** fase na qual os Serviços e Processos estão em operação;

- **Monitoração:** fase na qual é realizado a medição do nível de Serviço dos processos, com objetivo de garantir a qualidade na execução, a qual é medida através de Métricas e *Key Performance Indicators* (KPI).

---

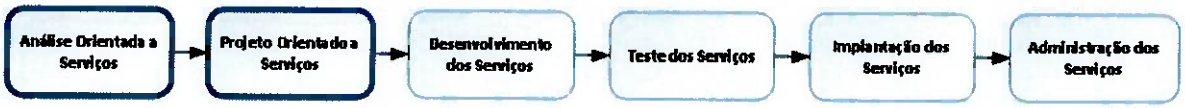
[1] **Processo *as-is* - como o processo de negócios é hoje.**
[2] **Processo *to-be* - como o processo de negócios deverá ser no futuro.**

3.1.6 Construção de Aplicações SOA segundo ERL

Para a construção de sistemas de software baseados na Arquitetura Orientada a Serviços (SOA), há uma série de etapas que precisam ser percorridas para a construção e distribuição dos serviços.

Erl (2005) propõe que o ciclo de vida de processo para construção e distribuição desses serviços, seja composto por seis etapas, como mostra a figura 6: Análise Orientada a Serviços, Projeto Orientado a Serviços, Desenvolvimento dos Serviços, Teste dos Serviços, Implantação dos Serviços e Administração dos Serviços. Nas etapas de Análise e Projeto Orientados a Serviços aplicam-se os princípios da arquitetura SOA.

Figura 6: Processo para construção e distruibuição de projetos baseados em SOA, destacando-se as fases de análise e projeto.



Fonte: Erl (2005).

Descrevem-se a seguir as etapas do ciclo de vida de SOA, conforme Erl (2005).

Etapa 1 - Análise Orientada a Serviços: neste estágio inicial são definidos os serviços que comporão a solução, provenientes do modelo de negócios; esses serviços são analisados e verificam-se quais deles já existem no portifólio da empresa ou domínio da aplicação, ou ainda, se podem ser reaproveitados de recursos existentes. Determinam-se também as camadas de serviços que serão necessárias para conter tais serviços.

Etapa 2 - Projeto Orientado a Serviços: define-se como será a arquitetura do projeto de software, a partir da escolha de quais padrões de *Web Services* utilizar, quais princípios de orientação a serviços incorporar ao mesmo, bem como convenções derivadas da indústria. É uma etapa em que muitas decisões devem ser tomadas cuidadosamente, definindo os contornos dos serviços que serão utilizados. Algumas camadas da Camada de Serviços são projetadas nessa etapa, dentre as quais se inclui a de Orquestração.

Etapa 3 - Desenvolvimento dos Serviços: escolhe-se qual plataforma de desenvolvimento e linguagem de programação serão utilizadas para projetar os serviços e processos de negócios orquestrados.

Etapa 4 - Teste dos Serviços: os serviços projetados devem passar por rigorosos testes até ser publicados em um ambiente de produção.

Etapa 5 - Implantação dos Serviços: realiza-se a instalação e configuração dos componentes distribuídos, interfaces de serviços e quaisquer produtos de *middleware*[1] nos servidores de produção. Essa etapa está muito associada à plataforma tecnológica para a qual os serviços foram desenvolvidos.

Etapa 6 - Administração dos Serviços: realiza-se o gerenciamento dos serviços publicados através de ferramentas destinadas para tal finalidade.

Para o estudo realizado nesta monografia, somente as fases de Análise e Projeto serão discutidas em maior detalhes. Estas fases apresentam ciclos internos, com outras etapas que devem ser percorridas. Na figura 6, essas duas etapas se encontram em destaque.

3.1.7 Análise e Projeto Orientado a Serviços Utilizado o MAPOS

MAPOS é um método de Análise e Projeto Orientado a Serviços, baseado na análise crítica de métodos existentes, procurando disciplinar o procedimento de identificação, análise e especificação de serviços alinhados aos requisitos de negócio (FUGITA, 2009).

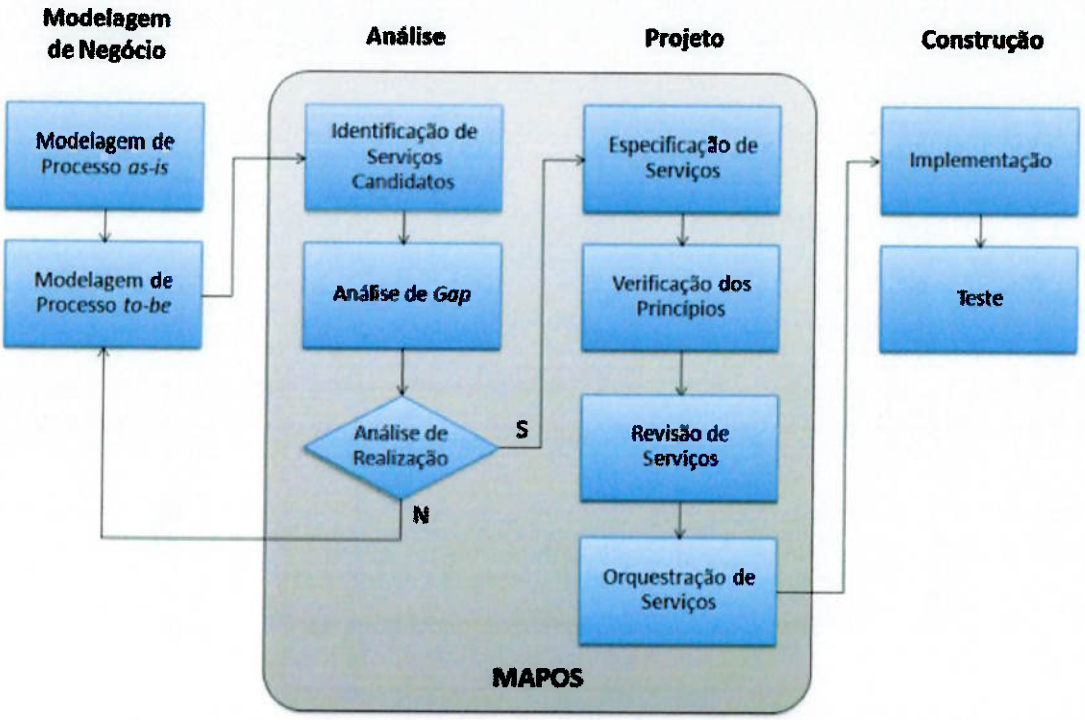
Segundo Fugita (2009), tal método busca oferecer maior flexibilidade em sua aplicação através de iterações durante a fase de Análise e Projeto. O ciclo de vida proposto por este método começa com a Modelagem de Negócio, passa pelas fases

---

[1] ***Middleware*: "sistema distribuído de serviços, com interfaces e protocolos padronizados, que busca solucionar problemas de heterogeneidade e interoperabilidade entre produtos de diferentes fabricantes" (FERREIRA, 2001, p. 13).**

de Análise e Projeto, passando pelas atividades de Implementação e Testes. Na figura 7, se tem o exemplo do ciclo de vida proposto pelo MAPOS.

Figura 7: Ciclo de Vida do Método de Análise e Projeto Orienteado a Serviços.



Fonte: Fugita (2009).

Descrevem-se a seguir as etapas do ciclo de vida MAPOS, conforme Fugita (2009).

- **Modelagem de Processo *as-is* e *to-be*:** fase onde é determinado o estado atual dos processos de negócio *as-is*, que serve de base para a construção dos modelos de processos de negócio *to-be*, na qual serão propostas modificações ao processo atual com o intuito de promover melhorias de desempenho ou reduzir custos (FUGITA, 2009);

- **Atividades de Análise:** fase na qual são identificados os Serviços Candidatos a partir dos modelos de processos de negócios fornecidos; é formada pelas três tarefas que se descreve a seguir.

  - **Identificação de Serviços Candidatos:** fase na qual são identificados os Serviços Candidatos a partir do modelo de processo *to-be* ou do modelo de

casos de uso de negócio, que servirão também como base para esse trabalho;

- **Análise de *Gap*:** fase na qual é feita uma comparação entre os Serviços Candidatos e os recursos já existentes na infraestrutura de TI da organização, avaliando as possibilidades de reuso;

- **Análise de Realização:** fase onde são analisadas alternativas de realização possíveis para cada uma das Operações Candidatas identificadas e decide-se qual delas será adotada para realizar o serviço;

- **Atividades de Projeto:** fase na qual realiza-se o Projeto de Serviços, na qual, criam-se as especificações de serviços de acordo com os requisitos do negócio. É composta pelas quatro fases que estão descritas a seguir.

  - **Especificação dos Serviços**: fase na qual as interfaces dos serviços serão definidas, bem como as suas operações;

  - **Verificação dos Princípios:** fase na qual é feita uma série de verificações sobre a especificação de cada um dos serviços, visando avaliar a conformidade com os princípios de orientação a serviços, que serão expostos na seção 3.1.8;

  - **Revisão dos Serviços:** fase na qual é verificada a aderência aos princípios de orientação a serviços nos serviços criados, podendo demandar alterações nos mesmos;

  - **Orquestração de Serviços:** fase na qual é realizada a orquestração dos serviços construídos.

- **Atividades de Contrução:** fase na qual são desempenhadas atividades com o intuito de se obter os serviços executáveis devidamente integrados e testados, prontos para serem implantados. É composta por duas subatividades: Implementação e Testes.

- **Implementação e Teste:** fase nas quais a implantação e testes dos serviços são executados pelos desenvolvedores, que receberão as especificações dos mesmos para a realização de tais atividades.

3.1.8 Princípios SOA

Erl (2005) destaca um grupo de princípios que devem ser considerados na construção de projetos que utilizem SOA. Na figura 8, destacam-se alguns destes princípios.

Figura 8: Princípios SOA.

Fonte: Do Autor.

Descrevem-se a seguir os princípios SOA, conforme Erl (2005).

- **Reuso:** um serviço deve ser projetado no sentido de favorecer o reuso de seus recursos por diversos tipos de aplicações de software;

- **Contrato Padronizado:** serviços devem compartilhar um contrato formal bem definido que descreve o que cada serviço faz e define os termos para troca de informação;

- **Baixo Acoplamento**: um serviço deve ser projetado para interagir com outras aplicações de software, ou com outros serviços, sem manter dependência deles; de acordo com Fujita e Hirama (2012), "de uma forma geral, baixo acoplamento é uma condição em que um serviço está ciente da existência de outros serviços, mas ainda assim é independente deles (ERL, 2005)";

- **Abstrair Lógica de Negócio:** um serviço deve expor somente um Contrato ou Interface, que permita que o mesmo seja utilizado por uma aplicação de software cliente, não permitindo o acesso aos detalhes de implementação. Os contratos de serviço devem ser representados em uma linguagem ou notação bem definida, seguindo padrões que garantam a interoperabilidade do serviço;

- **Permitir Composição entre Serviços:** um serviço deve permitir ser composto por outros serviços, permitindo reusabilidade e a construção de camadas de abstração;

- **Autonomia:** a lógica empregada em um serviço deve estar encapsulada em uma fronteira explícita, a qual deve ser controlada pelo mesmo, tornando-o independente de outros serviços para executar a governança do mesmo;

- **Execução Sem Estado:** um serviço deve ser projetado para ter baixoacoplamento, autonomia e permanecer sem estado. Este princípio propicia o reuso do serviço e permite a composição de serviços;

- **Visibilidade:** um serviço deve permitir que suas descrições possam ser descobertas e sejam de fácil entendimento aos usuários do mesmo. Para isso, deve ser disponibilizado no Registro de Serviços.

Além destes princípios citados, há outros, como: separação de interesses, visões arquiteturais, acomodação de mudanças, consistência, derivação de negócio, padrões, facilitação e comunicação entre outros. Dentre eles cabe destacar a interoperabilidade; ele implica na necessidade de interação com o serviço, independente da tecnologia empregada na sua implementação (FUGITA; HIRAMA, 2012).

3.2 Tecnologias de Web Services

Nesta seção, serão apresentados os principais conceitos sobre as principais tecnologias de *Web Services* e quais elementos formam toda a arquitetura necessária para o projeto e utilização dos mesmos.

3.2.1 Definição

Um *Web Service* é uma aplicação de software desenvolvida para permitir interoperabilidade entre sistemas de software distribuídos em diferentes plataformas e/ou *frameworks*, sendo independente de linguagem de programação, sistema operacional e hardware, possibilitando baixo acoplamento entre o Consumidor e Provedor de Serviços (ENDREI *et al.*, 2004).

De acordo com a W3C (2004, p. 7):

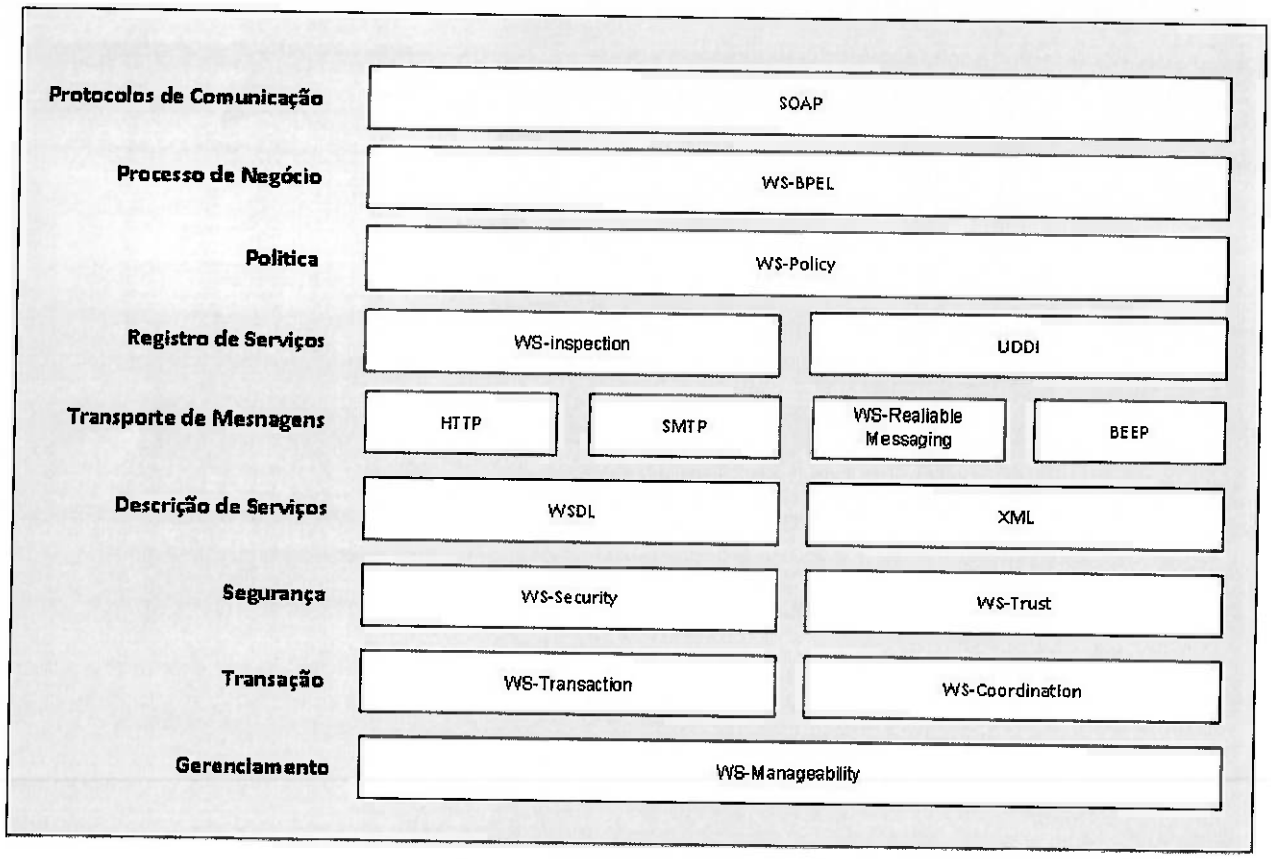
> Um *Web Service* é um sistema de software projetado para apoiar a interoperabilidade de interação entre computadores através de uma rede de computadores. Ele tem uma interface descrita em um formato processável por máquina (especificamente WSDL). Outros sistemas interagem com o *Web Service* de uma forma prescrita através da descrição do mesmo utilizando mensagens SOAP, tipicamente transportada utilizando HTTP com serialização XML em conjunto com outros padrões Web relacionados.

Com a padronização oferecida a partir da utilização do protocolo SOAP e mensagens baseadas na liguagem XML, a adoção da tecnologia de *Web Services* tem se tornado uma opção mais vantajosa em relação à utilização de protocolos

proprietários, como CORBA e DCOM, pois permite que as mensagens sejam definidas de forma que não fiquem dependentes de uma determinada plataforma ou tecnologia.

Sendo assim, a tecnologia de *Web Services* é baseada numa abordagem de computação distribuída para integração de sistemas de software heterogêneos através de redes de computadores e da Internet, tendo como principais padrões: SOAP, WSDL, XML, WS-BPEL e UDDI. A Figura 9, exibe uma lista com os padrões e tecnologias de *Web Services*.

Figura 9: Principais padrões e tecnologias de Web Services.



Fonte: baseado em Credle *et al.* (2007)

Na figura 9 é mostrada a pirâmide com os padrões de *Web Services*, relacionados aos propósitos para os quais foram projetados.

3.2.2 Extensible Markup Language (XML)

A linguagem de marcação extensível XML, do inglês *Extensible Markup Language*, é um formato baseado em texto, utilizada para descrição e troca de dados através da Internet. É portável entre diferentes ambientes computacionais e base para os para os principais padrões de *Web Services*, como: SOAP, BPEL, WSDL, UDDI entre outros (PAPAZOGLOU, 2008).

Através da notação fornecida pela linguagem XML é possível representar as informações contidas em um documento ou numa mensagem de forma estruturada, de acordo com o tipo de dados correspondente. Isto é possível porque um documento XML é composto por Elementos Nomeados, os quais representam cada um dos atributos que formam a mensagem. Esses Elementos por sua vez não são predefinidos, mas personalizados para um determinado contexto. Um conjunto desses elementos relacionados definem um Vocabulário para o contexto considerado. Na figura 10 tem-se o exemplo de um arquivo XML (PAPAZOGLOU, 2008).

Figura 10: Exemplo de um documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Cliente>
    <Nome>Júlio da Silva</Nome>
    <DataNascimento>20/02/1992</DataNascimento>
    <Endereco>
        <Logradouro>Rua Ana Jarvis</Logradouro>
        <Numero>125</Numero>
        <Complemento />
        <Cidade>Mogi das Cruzes</Cidade>
        <UF>SP</UF>
        <CEP>09225643</CEP>
    </Endereco>
    <Telefones>
        <Residencial>1155893365</Residencial>
        <Celular>11978558711</Celular>
    </Telefones>
</Cliente>
```

Fonte: Do Autor.

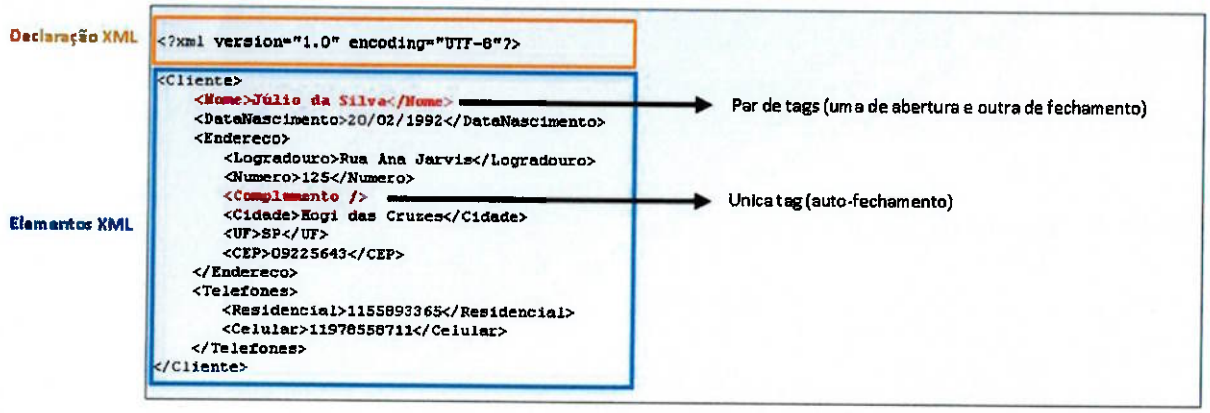
No exemplo da figura 10, uma instância de documento XML representa os dados de um determinado cliente, contendo: nome, data de nascimento, endereço e telefones. Cada um destes dados são representados pelos Elementos Nomeados - ou *tags* (etiquetas)-, onde são inseridas as informações que os mesmos estão representando (valores atribuídos aos elementos); como exemplo, o nome do cliente – Julio da Silva - está entre as *tags* <Nome> e </Nome>. Nesse exemplo, o

Vocabulário é constituído pelos seguintes Elementos Nomeados: Cliente, Nome, DataNascimento, Endereço (e seus elementos constituintes) e Telefone.

3.2.2.1 Estrutura de um Documento XML

Na figura 11 é mostrada a estrutura de um documento XML. Um documento XML é, basicamente, formado pela Declaração XML – destacada em vermelho, na primeira linha -, que contém a versão de XML que está sendo utilizada e o tipo de codificação e pelo restante do documento, formado pelos Elementos Nomeados que definem o seu vocabulário, representado pelos pares de *tags* , ou por uma única *tag* denominada *self-closing* (auto-fechamento), conforme destacado na figura.

Figura 11: Esturtura de um documento XML.



Fonte: Do Autor.

Como ilustrado na figura 11, na Declaração XML, está sendo informado o atributo *version* (versão), indicando a versão 1.0, e o atributo *encoding* (codificação), indicando que o tipo de arquivo é UTF-8. Nos Elementos XML, cada um dos elementos ou *tags*, representa uma das informações sobre o Cliente, como o <Nome></Nome>, formado por um par de *tags*, ou <Complemento />, que é formado por uma única *tag*, que se auto-fecha (*self-closing*).

3.2.2.2 Esquema XSD e os Padrões de Web Services

XSD (do inglês *XML Schema Definition Languagem*) é uma linguagem para definição e descrição de classes de documentos XML através do uso de esquemas, que

documentam o significado, o uso e os relacionamentos entre as partes que constituem o documento, tais como: tipos de dados, elementos, atributos e valores (W3C, 2004). O documento XML é uma "instância de um determinado esquema". Na troca de informação através de XML, as organizações necessitam definir um determinado vocabulário para trocarem as informações, ou seja definir qual esquema vai ser utilizado.

Através de XSD, é possível estender as capacidades de utilização dos arquivos XML por meio de mecanismos que permitem representar os elementos de forma mais específica através de atributos, como: tipo de dados, quantidade de caracteres, tamanho mínimo e máximo de cada um, dentre outros. Entretanto os esquemas definem tipos de dados não proprietários, que nem sempre coincidem com aqueles fornecidos pelas linguagens de programação comuns. Os esquemas permitem, além de descrever a estrutura dos documentos, impor regras de validação ou restrições sobre os tipos de dados, bem como sobre os relacionamentos entre as partes. Além de XSD, existem outros esquemas utilizados com XML (PAPAZOGLOU, 2008).

Neste contexto, os padrões de *Web Services* como SOAP, WS-BPEL, UDDI, WSDL entre outros, utilizam XML e XSD para estruturar e validar os elementos que compõem os arquivos que os representam. Nas figuras 12 e 13, apresentam-se exemplos de uso de XML e XSD: um arquivo SOAP e outro WSDL. A codificação XSD está destacada nas figuras, mostrando a interação entre esse esquema e a estruturação dos documentos de *Web Services,* com dois tipos diferentes de padrões de *Web Services*.

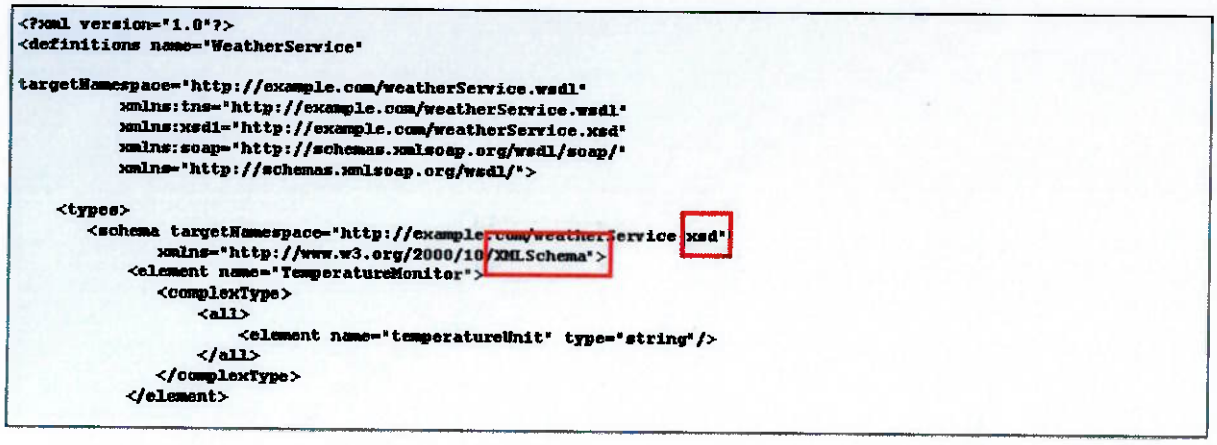
Figura 12: Exemplo de um arquivo SOAP e utilização do Esquema XSD.

```
POST /WsSchedule/ScheduleService.asmx HTTP/1.1
Host: www.efasis.com.br
Content-Type: text/xml; charset=utf-8
Content-Length: aaaa
SOAPAction: "http://www.efasis.com.br/GetScheduleList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetScheduleList xmlns="http://www.efasis.com.br/" >
            <employeeId>int</employeeId>
        </GetScheduleList>
    </soap:Body>
</soap:Envelope>
```

Fonte: Do Autor.

Figura 13: Exemplo de um arquivo WSDL e utilização do Esquema XSD.



```
<?xml version="1.0"?>
<definitions name="WeatherService"

targetNamespace="http://example.com/weatherService.wsdl"
          xmlns:tns="http://example.com/weatherService.wsdl"
          xmlns:xsd1="http://example.com/weatherService.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
      <schema targetNamespace="http://example.com/weatherService.xsd"
            xmlns="http://www.w3.org/2000/10/XMLSchema">
         <element name="TemperatureMonitor">
            <complexType>
               <all>
                  <element name="temperatureUnit" type="string"/>
               </all>
            </complexType>
         </element>
```

Fonte: Do Autor.

A criação de vocabulários em XML é livre. Apesar da flexibilidade que isso oferece, é necessário estabelecer estratégias de gerenciamento, pois isso evita o desalinhamento entre eles, bem como entre as regras de negócio envolvidas. O XML Data Custodian estabelece esse gerenciamento e cria novos esquemas à medida que as empresas necessitam deles. Os esquemas são associados a domínios lógicos, denominados *namespaces*. Nas figuras 12 e 13 eles estão identificados pela palavras reservadas `xmlns` e `TargetNamespace`. As figuras também evidenciam o uso de mais de um *namespace* em cada um dos exemplos. Essa estratégia de gerenciamento traz inúmeras vantagens: a não proliferação de domínios de dados, evita a redundância de dados e o desalinhamento entre os vocabulários e as regras de negócios envolvidas.

3.2.3 Simple Object Access Protocol (SOAP)

SOAP (*Simple Object Access Protocol*) é um protocolo de comunição, que utiliza linguagem XML e fornece um mecanismo, cuja finalidade, é mediar a troca de dados entre aplicações Web em um ambiente distribuído e descentralizado. (W3C, 2000).

Na figura 14, o exemplo mostra uma mensagem SOAP enviada através de uma requisição HTTP[1] (do inglês HTTP *Request*). A mensagem SOAP está contida entre

[1] **HTTP:** *Hypertext Transfer Protocol* ou Protocolo de Transferência de Hipertexto, é um protocolo de comunição de nível de aplicação utilizado em sistemas de informação de hipermídia colaborativos e distribuídos (LEE *et al.*, 1996).

a *tag* `soap:Envelope` e a *tag* `/soap:Envelope`. Essa construção funciona como um *container* para a mensagem que se segue. A mensagem é formada por um cabeçalho (*Header*) e por um corpo (*Body*). A figura mostra ainda o uso do *namespace* com o vocabulário SOAP a ser utilizado, identificado pela palavra reservada `xmlns:soap`.

Figura 14: Mensagem SOAP enviada através de uma Requisição HTTP.

```
POST /WsSchedule/ScheduleService.asmx HTTP/1.1
Host: www.efasis.com.br
Content-Type: text/xml; charset=utf-8
Content-Length: aaaa
SOAPAction: "http://www.efasis.com.br/GetScheduleList"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetScheduleList xmlns="http://www.efasis.com.br/" >
            <employeeId>int</employeeId>
        </GetScheduleList>
    </soap:Body>
</soap:Envelope>
```

Fonte: Do Autor.

No exemplo da figura 14, a requisição SOAP é realizada através do método chamado `GetScheduleList` (primeira linha de `<soap:Body>`), a qual é enviada ao *Web Service* chamado `ScheduleService` (primeira linha da mensagem), e deverá retornar uma lista de compromissos agendados de acordo com o número de identificação do funcionário, passado como parâmetro do método, na linha seguinte ao `GetScheduleList`.

Na figura 15, é mostrada a resposta HTTP (do inglês HTTP *Response*), referente à requisição HTTP ilustrada na figura 14.

Figura 15: Mensagem SOAP retornada de uma Resposta HTTP.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: aaaa

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Body>
      <GetScheduleListResponse xmlns="http://www.efasis.com.br/">
         <GetScheduleListResult>
            <string>14/07/2015 10:00 AM - Reunião Equipe Vendas</string>
            <string>17/07/2015 02:00 PM - Viajem a Nova York</string>
            <string>20/08/2015 09:00 AM - Participar de Congresso</string>
         </GetScheduleListResult>
      </GetScheduleListResponse>
   </soap:Body>
</soap:Envelope>
```

Fonte: Do Autor.

No exemplo da figura 15, uma resposta SOAP é retornada pelo método `GetScheduleListResponse`, com uma lista de compromissos – denominada `GetScheduleListResult` - em formato de texto (definido pelas *tags* `string` e `/string`), contendo data, horário e descrição do compromisso. No topo da mensagem HTTP, o número `200` seguido de `OK`, identifica que a requisição do método foi bem sucedida.

De acordo com a W3C (2000), o protocolo SOAP é formado por três partes:

- Envelope, onde estão contidas informações sobre o que está na mensagem e como processá-la, sendo composto por um cabeçalho (do inglês *header*) e um corpo (do inglês *body*); na figura 15 apenas o corpo foi apresentado;

- Codificação, onde ficam as regras para codificação e serialização para troca de dados, entre os tipos de dados definidos nos aplicativos de software;

- Representação, onde são definidas as regras que podem ser utilizadas para a representação das chamadas e respostas de procedimentos remotos.

3.2.3 Web Services Description Language (WSDL)

Linguagem de Descrição de Web Services (WSDL, do inglês *Web Services Description Language*), é um padrão baseado na linguagem XML, utilizado para descrever as caracteristicas funcionais de um *Web Service*, fornecendo dados sobre a finalidade, onde está publicado e como pode ser invovado tal serviço (W3C, 2001).

Através dos dados fornecidos pelo WSDL sobre as Interfaces, operações, protocolos suportados por um determinado *Web Service*, um Consumidor (Cliente), utiliza os recursos oferecidos pelo mesmo. No Quadro 2, tem-se uma lista com os elementos utilizados pelo documento WSDL, para especificação dos *Web Services*.

Quadro 2: Elementos que constituem a especificação WSDL.

Elementos WSDL	Finalidade
Tipos (Types)	Recipiente para definição de tipos de dados utilizando algum tipo de sistema, como o XSD.
Messagem (Message)	Definição abstrata e tipada dos dados que estão sendo comunicados.
Operação (Operation)	Definição abstrata de uma ação suportada pelo serviço.
Ligação (Binding)	Especificação concreta do protocolo de rede e formato de dados para uma determinada porta.
Porta (Port)	Definição de uma única porta com uma combinação de uma lingação e um endereço de rede.
Serviço (Service)	Coleção de portas (endpoints) relacionados.

Fonte: W3C (2001).

Uma definição de *Web Service* expressa em WSDL é composta por coleções de elementos primários: interfaces,mensagens, serviços e ligações.

As interfaces (`portTypes`) são equivalentes às interfaces de um componente de software. A interface é composta por um conjunto de operações (`operations`) que são equivalentes aos métodos do componente de software. Por sua vez, uma operação é constituída por um conjunto de mensagens (`message`) de entrada (`input`) ou saída (`output`). As mensagens são trocadas entre o Cliente e o Provedor do Serviço durante a execução desse serviço.

Na figura 16, tem-se o exemplo de um arquivo WSDL, com os elementos informados no quadro 2. Toda a definição do serviço está contida entre as *tags* `definitions` (segunda linha do código) e `/definitions` (última linha). Essas *tags* atuam como o *container* da definição. Dentro desse *container* os demais tipos de construção estão distribuídos.

A interface desse serviço (`portType`) é constituída por uma única operação `GetScheduleListPrice`, definida dentro da interface `EmployeeSchedulePortType`. Duas mensagens são definidas para esse serviço: `GetScheduleListInput` e `GetScheduleListOutput`. A definição da operação utiliza essas mensagens como de entrada (`input message`) e saída (`output message`), respectivamente.

A mensagem de entrada, `GetScheduleListInput`, é constituída por um tipo de dado – definido entre os `types` - como `EmployeeScheduleRequest`. Os tipos de dados em WSDL são suportados por esquemas XSD. O mesmo é feito quanto à mensagem de saída.

Figura 16: Documento WSDL contendo a especificação de um Web Service.

```
<?xml version="1.0"?>
<definitions name="EmployeeSchedule"

targetNamespace="http://www.efasis.com.br/employeeSchedule.wsdl"
          xmlns:tns="http://www.efasis.com.br/employeeSchedule.wsdl"
          xmlns:xsd1="http://www.efasis.com.br/employeeSchedule.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
       <schema targetNamespace="http://www.efasis.com.br/employeesSchedule.xsd"
             xmlns="http://www.w3.org/2000/10/XMLSchema">
          <element name="EmployeeScheduleRequest">
             <complexType>
                 <all>
                     <element name="employeeId" type="integer"/>
                 </all>
             </complexType>
          </element>
          <element name="EmployeeSchedule">
             <complexType>
                 <all>
                     <element name="ScheduleDateTime" type="string"/>
                              <element name="Description" type="string" />
                 </all>
             </complexType>
          </element>
       </schema>
    </types>
    <message name="GetScheduleListInput">
        <part name="body" element="xsd1:EmployeeScheduleRequest"/>
    </message>
    <message name="GetScheduleListOutput">
        <part name="body" element="xsd1:EmployeeSchedule"/>
    </message>
    <portType name="EmployeeSchedulePortType">
        <operation name="GetScheduleList">
           <input message="tns:GetScheduleListInput"/>
           <output message="tns:GetScheduleListOutput"/>
        </operation>
    </portType>
    <binding name="EmployeeScheduleSoapBinding" type="tns:EmployeeSchedulePortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastEmployeeSchedule">
           <soap:operation soapAction="http://www.efasis.com.br/GetScheduleList"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
                     <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
    <service name="EmployeeScheduleService">
        <documentation>Employees Schedule Service</documentation>
        <port name="EmployeeSchedulePortType" binding="tns:EmployeeScheduleBinding">
           <soap:address location="http://www.efasis.com.br/employeesSchedule.xsd"/>
        </port>
    </service>
</definitions>
```

Fonte: Do Autor.

Na figura 16, o exemplo é a especificação WSDL para um *Web Service* destinado à consulta da agenda de funcionários: `EmployeeScheduleService`. Esse serviço está especificado pelos *tags* `service` e `/service`, que se encontram entre as linhas finais do código. A especificação desse serviço incorpora a definição da interface a ser utilizada, ou seja, `EmployeeSchedulePortType`. Os Serviços (`service`) e as Ligações (`Binding`) representam a implementação do componente, continuando com a analogia iniciada. A função do elemento de ligação é estabelecer a ligação do serviço com o tipo de protocolo de comunicação (SOAP, HTTP). No caso do exemplo, comandos SOAP são usados. A palavra reservada `targetNamespace` introduz os esquemas a serem usados.

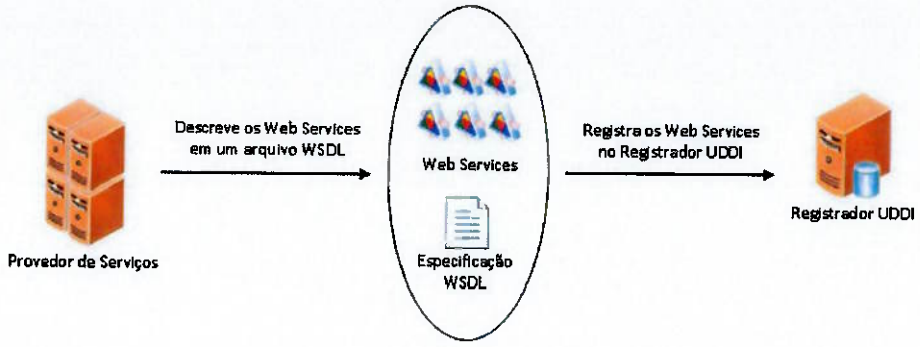
3.2.4 Universal Description, Discovery, and Integration (UDDI)

UDDI (*Universal Description, Discovery, and Integration*), é um padrão para registro de descoberta e descrição de *Web Services*, que fornece mecânismos para que os Provedores possam especificar os *Web Services* publicados, e mecanismos para que os Consumidores, também chamados de Clientes, possam descobrir as especificações de tais *Web Services* para fazer uso deles (PAPAZOGLOU, 2008).

Para publicarem os *Web Services* em um Registo UDDI, os Provedores precisam, em primeiro lugar, realizar um descrição de cada um deles através de um arquivo WSDL, no qual serão descritas as informações de negócio, como quem o implementou, a natureza do mesmo e dados técnicos sobre os métodos oferecidos e como implementá-los.

Depois de realizar tais especificações, os Provedores devem publicar os *Web Services* em um Registro UDDI, permitindo que estes serviços possam ser encontrados pelos Consumidores para que possam ser implementados e utilizados para os propósitos para o quais foram projetados. A figura 17, ilustra o processo de especificação e registro de *Web Services*.

Figura 17: Processo de especificação e registro de Web Services em um Registro UDDI.

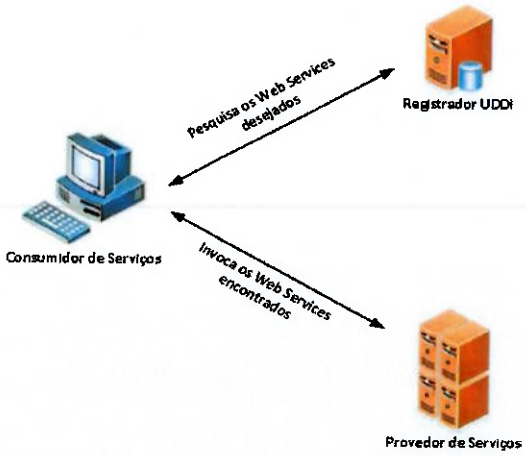


Fonte: Do Autor.

Na figura 17, os *Web Services* disponíveis pelo Provedor de Serviços, são especificados e registrados num Registro UDDI, para que possam ser descobertos e utilizados pelos Consumidores.

Para realizar a descoberta dos *Web Services* registrados, o Consumidor de Serviços (Cliente), deve realizar uma pesquisa no Registrador UDDI a fim de que seja retornada uma lista com os *Web Services* de acordo com os critérios desejados. Tal busca pode ser realizada em tempo de projeto (*design-time*), através de uma IDE[1], na qual codificam-se os comandos para busca das interfaces e operações a serem utilizadas. Na figura 18 é ilustrado como funciona o esquema de busca em um registrador UDDI.

Figura 18: Processo de pesquisa e invocação de Web Services em um Registrador UDDI.



Fonte: Do Autor.

[1] **IDE: *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, é um software de computador que permite desenvolver software com maior produtividade, através de ferramentas para edição, gerenciamente de arquivos, compilação, depuração e execução do software desenvolvido (KERRIGAN *et al.*, 2007).**

Na figura 18, os *Web Services* disponíveis pelo Provedor de Serviços, são pesquisados num Registro UDDI, e em seguida, invocados pelo Consumidor de Serviços, que implementa as interfaces e operações dos mesmos.

Sendo assim, a utilização do Registro UDDI, permite que as empresas troquem informações a partir de um registro global de *Web Services,* a partir do registro realizado pelo Provedor de Serviços, e pela pesquisa realizada pelas aplicações de software construídas pelos Consumidores (Clientes), que descobrem e acessam os detalhes a respeito das operações oferecidas pelos mesmos para sua utilização.

3.3 Composição de *Web Services*

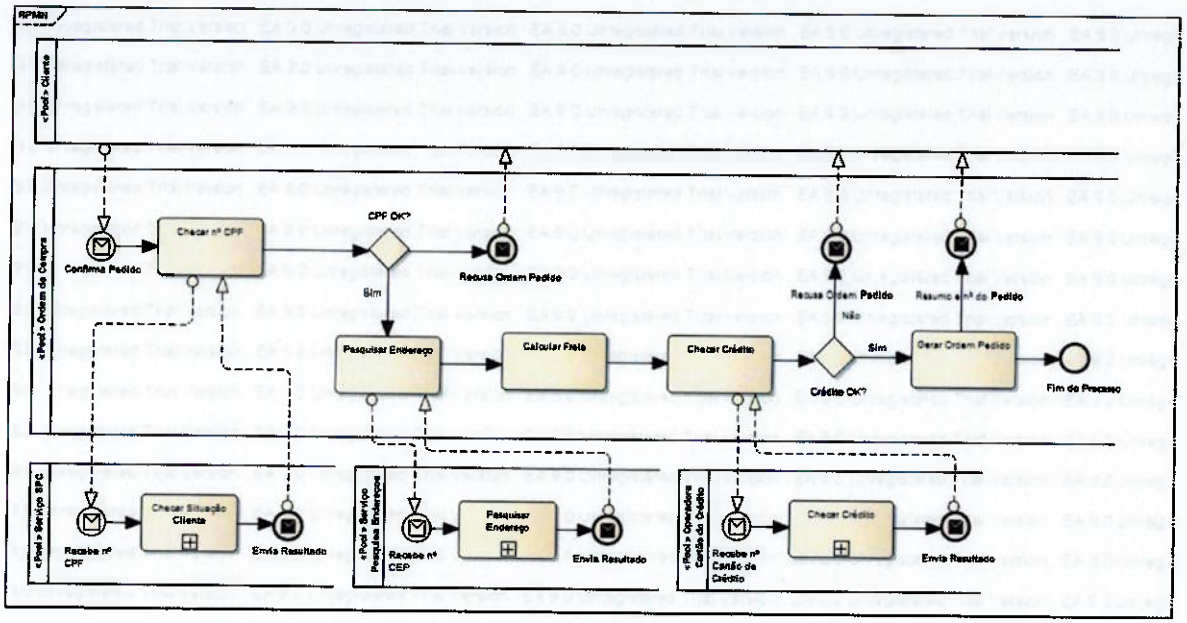
Nesta seção, serão apresentados os principais conceitos sobre Composição de *Web Services*, sobre os modelos de Orquestração e Coreografia, e sobre a linguagem para construção de fluxos de trabalho (*workflows*) baseada em *Web Services*, o WS-BPEL.

3.3.1 Definição

A Composição de *Web Services* consiste em um modelo de desenvolvimento de software, no qual, um grupo de diferentes *Web Services* são combinados e coordenados, para permitir a criação de atividades mais sofisticadas, no sentido de realizar um determinado objetivo em uma aplicação de software (HALILI *et al.*, 2013).

A partir deste modelo, é possível combinar as funcionalidades de *Web Services* fornecidos por Provedores diferentes, para compor novos *Web Services*, permitindo a integração de processos de negócios, mesmo que estes estejam separados. Um exemplo típico ocorre em um fluxo de trabalho de efetivação de pedido de compra em um site de comércio eletrônico, como ilustra a figura 19.

Figura 19: Exemplo de composição de *Web Services* em uma efetivação de pedido em um site de comércio eletrônico.



Fonte: Do Autor.

No fluxo de trabalho representado na figura 19, nas etapas de Confirmação de Pedido, Pesquisa de Endereços e Validação de Cartão de Crédito, três *Web Services* são utilizados para retornar informações necessárias no processo de efetivação de uma compra em um site de comércio eletrônico. Neste cenário, os três *Web Services* utilizados formam um modelo de Composição de serviços.

Como é possível perceber, este modelo baseia-se na análise e projetos de modelos de negócio utilizando fluxos de trabalho. Para o projeto de software através da composição de *Web Services* reutilizando serviços existentes, Sommerville (2011) propõe seis estágios, sendo eles:

1 - Formular o Fluxo de Trabalho Preliminar: a partir dos requisitos para compor os serviços, cria-se um projeto de serviço ideal, sem especificar maiores detalhes de fluxo de trabalho inicialmente.

2 - Descobrir Serviços: a partir de um catálogo de serviços existentes, indentifica-se a relação de serviços que podem ser utilizados para o projeto de fluxo de trabalhos que deseja criar.

3 - Selecionar Possíveis Serviços: a partir de uma relação de serviços candidatos que foram descobertos, selecionam-se os serviços que farão parte do fluxo de trabalho.

4 - Refinar o Fluxo de Trabalho: a partir das informações que se tem sobre os serviços selecionados, refina-se o fluxo de trabalho, adicionando ou removendo atividades ao mesmo.

5 - Criar Programa de Workflow: cria-se um programa executável que contenha os serviços e suas Interfaces. Para isto, utiliza-se uma linguagem ou tecnologia que permita a criação de serviços, como a linguagem de programação C# ou Java, ou uma linguagem própria para programação de fluxos de trabalho, como o WS-BPEL.

6 - Teste de Serviço: realizam os testes com os componentes de serviços projetados. Neste caso, não serão abordados os detalhes de fase.

Além das etapas para construção dos fluxos de trabalho, uma arquitetura de software que utiliza a composição de *Web Services*, deve conter mecanismos que identifiquem possíveis falhas e interrupções em alguma das etapas do fluxo de trabalho, devendo prover ações de desfaçam as operações não concluídas. De acordo com Sommerville (2011, p. 528), "na terminologia dos fluxos de trabalho, isso é chamado de ação de compensação".

Para que seja realizada a composição de *Web Services*, é requerido que um padrão para composição seja utilizado para controlar o fluxo de informação entre os componentes de serviços. Os dois principais padrões utilizados para este propósito são: a Orquestração e a Coreografia.

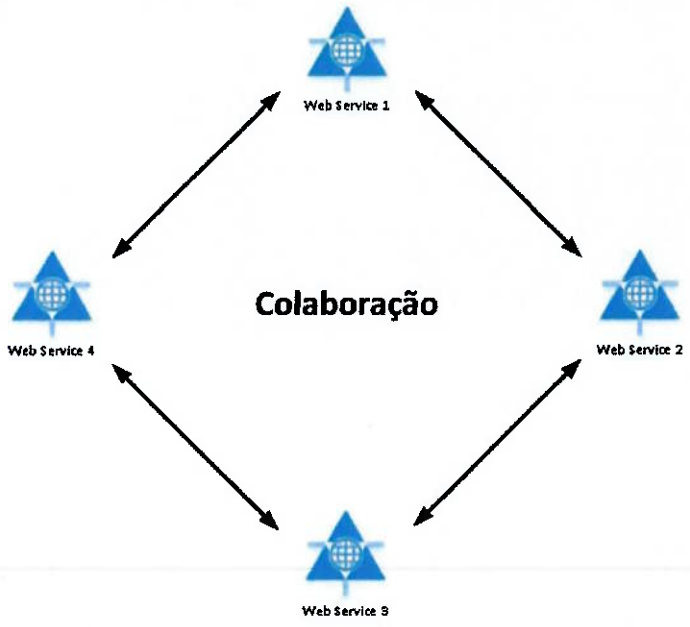
Enquanto na Orquestração o controle da execução dos *Web Services* fica centralizado em um único ponto, na Coreografia, cada *Web Service* envolvido no processo de negócio fica responsável pelo controle de parte da execução.

3.3.2 Coreografia de *Web Services*

A Coreografia é um modelo de composição de *Web Services*, no qual, cada *Web Service* envolvido, controla parte do fluxo de trabalho, invocando outros *Web Services* relacionados, formando um modelo de colaboração entre os mesmo (PELTZ, 2003).

Diferente da Orquestração, no qual o controle do fluxo de trabalho fica centralizado em um *Web Service*, na Coreografia, o controle é dividido entre cada um dos *Web Services* envolvidos. Neste modelo, um acordo é estabelecido entre as partes e diz como a colaboração deve ocorrer (HALILI, 2013). Na figura 20, tem-se um diagrama que ilustra o modelo de Coreografia de *Web Services*.

Figura 20: Composição de Web Services através de Coroegrafia.



Fonte: Halili *et al.* (2013)

No exemplo da figura 20, quatro *Web Services* atuam de maneira colaborativa no sentido de atingir um objetivo para o qual foram projetados. Neste caso, a responsabilidade na execução do processo é compartilhada entre os mesmos, não havendo um controle centralizado.

Análogo ao WS-BPEL para Orquestração, o WSCI (*Web Services Choreography Interface*), é uma especificação baseada em *Web Services*, utilizada para controlar a interação dos mesmos em um modelo arquitetural que utilize Coreografia. Este modelo descreve um ambiente observador entre os *Web Services* envolvidos. (PELTZ, 2003).

Tal especificação foi criada a partir de uma iniciativa de empresas como a Sun, SAP, BEA e Intalio, tornado-se W3C *Note,* em Agosto de 2002. Como o foco deste trabalho é realizar um estudo sobre Orquestração de *Web Services*, não serão apresentados maiores detalhes sobre o a utilização do WSCI e Coreografia de *Web Services*.

3.3.3 Orquestração de Web Services

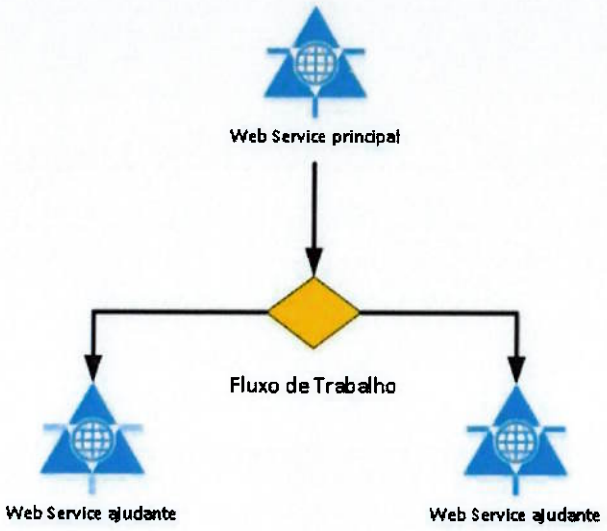
A Orquestração é um modelo de composição de *Web Services*, no qual, um *Web Service* principal, controla a execução de outros *Web Services* ajudantes, de acordo com cada uma das atividades que compõem um fluxo de trabalho (*workflow*).

Neste modelo, um motor de orquestração interpreta cada uma das atividades do fluxo de trabalho e invoca os *Web Services* destinados a realizar as operações que correspondam às mesmas. As interfaces e operações relativas aos *Web Services* ficam descritas em um arquivo WSDL, utilizado pelo orquestrador (PELTZ, 2003).

Atualmente, o WS-BPEL, é a principal linguagem utilizada na Orquestração de processos de negócio baseada em *Web Services*. Na seção 3.4, será mostrado, em detalhes, os principais fundamentos sobre tal especificação e sobre os passos para definir um orquestrador de *Web Services* utilizando a mesma.

Segundo Halili *et al.* (2013, p. 302), a "Orquestração pode ocorrer entre a aplicação e múltiplos *Web Services* ou múltiplos *Web Services* podem ser encadeados dentro de um fluxo de trabalho, para que eles possam se comunicar um com os outros". Na figura 21 tem-se um diagrama que ilustra o exemplo de um Orquestrador de *Web Services*.

Figura 21: Composição de Web Services através de Orquestração.



Fonte: Halili et al. 2013.

No exemplo da figura 21, o *Web Service* principal – localizado no topo da figura - decide, a partir de um bloco condicional, qual *Web Service* ajudante deverá ser executado para cumprir uma determinada atividade.

Os *Web Services* envolvidos no fluxo de trabalho, podem estar localizados internamente à empresa, fazendo parte da própria aplicação de software ou de um repositório de serviços da empresa, ou estar localizados externamente, sendo fornecidos por outra empresa provedora de serviços. Como exemplo, pode-se citar o caso de uma atividade que, para confirmação de um endereço, precise invocar um *Web Service* destinado à pesquisa de endereços dos clientes.

3.4 Web Services Business Process Execution Language (WS-BPEL)

Nesta seção serão apresentadas as principais características sobre a linguagem WS-BPEL e quais as etapas necessárias para projetar um Orquestrador de *Web Services* utilizando tal especificação.

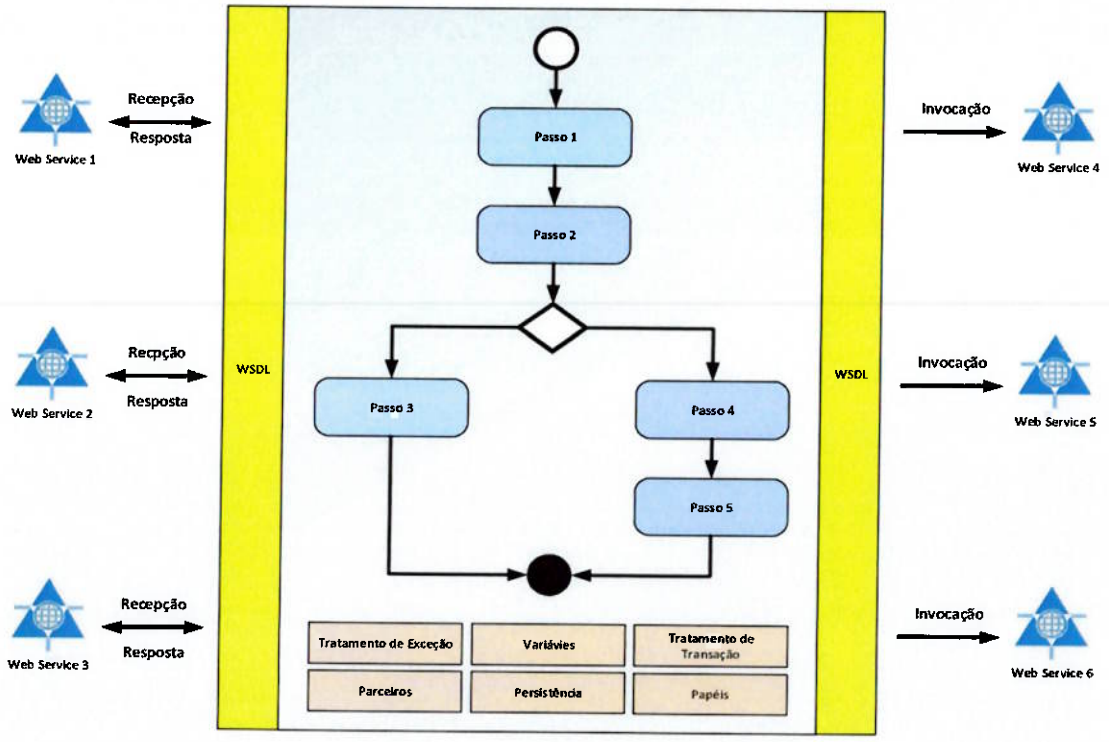
3.4.1 Definição

WS-BPEL ou somente BPEL (*Business Process Execution Language*) é uma linguagem para Orquestração de *Web Services* associados a um Processo de Negócios (OASIS, 2007).

Neste modelo, um motor de orquestração interpreta cada uma das atividades do fluxo de trabalho e invoca os *Web Services* destinados a realizar as operações que correspondem às mesmas. As interfaces e operações relativas aos *Web Services* ficam descritas em um arquivo WSDL, utilizado pelo orquestrador (PELTZ, 2003).

Além do gerenciamento dos processos e *Web Services*, a especificação WS-BPEL oferece recursos para a manipulação de exceções e transações, controle de papéis (*roles*) e parceiros (*partners*) bem como para especificar a persistência de variáveis. Na figura 22 ilustra-se o funcionamento de um Orquestrador de *Web Services* com WS-BPEL. Apesar dessa figura ser parecida com a Figura 21, ela destaca as interfaces do Orquestrador, codificadas em WSDL. Os processos parceiros se encontram além dos limites do Orquestrador.

Figura 22: Fluxo de trabalho controlado através de um Orquestrador utilizado o WS-BPEL.



Fonte: Peltz (2003).

No exemplo da figura 22, o exame do fluxo de trabalho mostra que, à medida que as atividades são executadas, os *Web Services* são acionados pelo mecanismo do WS-BPEL, através da camada WSDL, no qual os mesmos estão especificados. Abaixo do fluxo de trabalho, estão representados os elementos fornecidos pelo WS-BPEL, necessários para gerenciamento de persistência, variáveis, papéis, parceiros, tratamento de exceções e transações dentre outros.

3.4.2 História do WS-BPEL

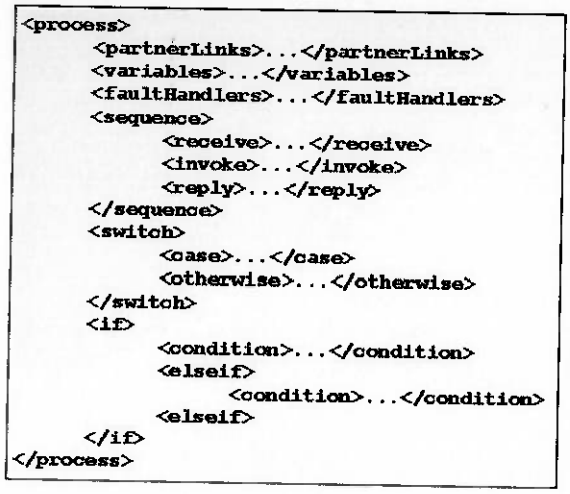
A linguagem WS-BPEL, atualmente na versão 2.0, foi definida através das revisões das versões 1.0 e 1.1 da linguagem BPEL4WS (*Business Process Execution Language for Web Services*), a qual foi projetada a partir de uma iniciativa de um comitê formado pelas empresas Microsoft, IBM, BEA, SAP e Siebel Systems.

Lançada em Julho de 2002, a linguagem BPEL4WS 1.0 foi criada a partir das linguagens XLANG (*XML Language*), da Microsoft, e WSFL (*Web Services Flow Language*), da IBM. Na segunda versão, BPEL4WS 1.1, lançada em Maio de 2003, as empresas SAP e Siebel Systems, contribuíram na implementação de novas melhorias em tal versão. A versão atual, disponível no portal de OASIS, é de 2007 (OASIS, 2007).

3.4.3 Estrutura de um Processo WS-BPEL

De acordo com a OASIS (2007, p.9, versão Primer), "um Processo BPEL é um recipiente (*container*), no qual podemos declarar relacionamentos com parceiros (*partners*) externos, declaração para processar dados, manipular para vários propósitos (exceções, transações etc.) e atividades para serem executadas". Na figura 23, apresenta-se um exemplo de como é a estrutura de um Processo WS-BPEL.

Figura 23: Estrutura de um processo WS-BPEL.



Fonte: Do Autor.

A figura 23 destaca os principais elementos do processo BPEL: `process`, partnerLink, variables, sequence dentre outros. A seguir, serão descritos cada um destes elementos.

3.4.4 **Processo (Process)**

O elemento processo (`process`) é o principal elemento que compõe a estrutura de um processo WS-BPEL. Nele, está declarado o nome do processo e os espaços de nome (*namespaces*), contendo as URIs referentes ao processo executável. Além disto, neste elemento estão contidos todos os demais elementos (OASIS, 2007). Na figura 24 apresenta-se um exemplo do elemento `process` e seus atributos de nome (`name`) *e* `namespace`.

Figura 24: Elemento `process` de um processo WS-BPEL.

```
<process name="ExemploDeProcesso"
         targetNamespace="http://oasis-open.org/WSBPEL/Primer/"
         xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable" >
      ...
</process>
```

Fonte: Do Aultor.

No exemplo da figura 24, tem-se um processo chamado ExemploDeProcesso, contendo os *namespaces* que compõem o mesmo, como o

`xmlns`="http://docs.oasis-open.org/wsbpel/2.0/process/executable", que indica que trata-se de um processo executável.

3.4.5 Parceiro (`PartnerLink`)

Parceiro (`partnerLink`) é um elemento, dentro de um processo executável WS-BPEL, que representa informação referente ao parceiro de um processo; o parceiro pode atuar como um cliente do processo ou como um prestador de serviço, dependendo da especificação do `portType` definido no *Web Service* que será acessado durante a execução do processo de negócio, representado a comunicação e troca de dados entre Cliente e Provedor (ERL, 2005).

O elemento `portType` é definido no arquivo WSDL do *Web Service* e contém a especificação das operações e mensagens que devem ser implementadas pelo WS-BPEL, equivalente a uma Interface abstrata na linguagem orientada a objetos, como C# (Csharp) ou Java. Desta maneira, a estrutura do elemento `partnerLink`, deverá corresponder à estrutura definida no elemento `portType`. O processo apresenta um atributo `myRole` e o parceiro um atributo `partnerRole` que caracterizam os papéis do processo e de seu parceiro.

Todos os elementos `partnerLink` de um processo WS-BPEL ficam agrupados dentro de um elemento de grupo, chamado `PartnerLinks`, como pode ser observado no exemplo da figura 25.

Figura 25: Elementos `partnerLink` e `partnerLinks` de um processo WS-BPEL.

```
<partnerLinks>
        <partnerLink name="CustomerPurchaseOrder"
                     partnerLinkType="ord:PurchaseOrderType"
                     myRole="PurchaseOrderServiceProvider"/>
        <partnerLink name="CheckCPF"
                     partnerLinkType="cpf:CheckCPFType"
                     partnerRole="CheckCPFServiceProvider"/>
        <partnerLink name="AddressSearch"
                     partnerLinkType="adr:AddressSearchType"
                     partnerRole="AddressSearchServiceProvider"/>
        <partnerLink name="CreditCheck"
                     partnerLinkType="ord:CreditCheckType"
                     partnerRole="CreditCheckServiceProvider"/>
        <partnerLink name="Notification"
                     partnerLinkType="not:NotificationType"
                     partnerRole="NotificationServiceProvider"/>
</partnerLinks>
```

Fonte: Do Autor.

Na figura 25, vê-se o exemplo de um grupo de `partnerLinks` definidos para acessar *Web Services*, cujas operações são necessárias em um processo de pedido de compra de um *site* de comércio eletrônico. O primeiro desses `partnerLink` especifica o papel do processo (`myRole`) e os demais, os papeis dos parceiros (`partnerRole`). Nessas especificações, o `partnerLinkType` especifica o `portType` dos elementos.

3.4.5 Variável (`Variable`)

Variável (`Variable`) é um elemento, dentro de um processo WS-BPEL, utilizado para armazenar dados que precisam ser persistidos durante as trocas de mensagens ocorridas entre os parceiros, mantendo o estado do processo de negócio durante sua execução (OASIS, 2007).

Através da utilização de variáveis, os dados obtidos durante a execução das interações das atividades dos *Web Services*, podem ser utilizados para auxiliar na manipulação dos dados processados durante o fluxo de trabalho em tempo de execução.

O elemento `variable` pode armazenar dados definidos em vários formatos como um texto, um número, um tipo complexo, uma classe, dentre outros. "Os tipos de dados que podem ser atribuídos a um `variable`, precisam ser definidos usando um destes três atributos: `messageType`, `element` ou `type`" (ERL, 2005, p. 573). Na figura 26, apresenta-se um exemplo de elemento `variable`.

Figura 26: Exemplo de um elemento `variable`.

```
<variables>
      <variable name="CustumerPOSubmission" messageType="myNS:CustomerPurchOrderSubmission" />
      <variable name="CpfCheckRequest" messageType="myNS:getCpfCheckRequestMessage" />
      <variable name="CredCheckRequest" element="myNS:getCredCheckRequestMessage" />
      <variable name="CheckAddressRequest" element="myNS:getAddressDataRequestMessage" />
      <variable name="OrderResult" type="xsd:string" />
      <variable name="NotificationResponse" type="myNS:NotificationResponseMessage" />
</variables>
```

Fonte: Do Autor.

Na figura 26, mostra-se um grupo de variáveis, definidas em WS-BPEL, para persistir os dados retornados dos *Web Services* utilizados em um processo de Ordem de Pedido de um *site* de comércio eletrônico.

3.4.6 Sequência (`Sequence`)

Sequência (`sequence`) é um elemento dentro de um processo WS-BPEL, utilizado para organizar a execução de um grupo de atividades de forma sequencial e pré-definida. Estas atividades são representadas pelo WS-BPEL, através dos elementos `receive`, `invoke` e `reply` (ERL, 2005). Na figura 27, apresenta-se um exemplo de um elemento `sequence`.

Figura 27: Exemplo de um elemento `sequence`.

```
<sequence name="CustomerMessageOrderSEQ">
    <receive name="receiveOrder" ... />
    <invoke name="invokeCheckCPF" ... />
    <invoke name="invokeCheckAddress" ... />
    <invoke name="invokeCreditCheck" ... />
    <reply name="replySendConfirmation" ... />
</sequence>
```

Fonte: Do Autor.

Na figura 27, o elemento `sequence` denominado `CustomerMessageOrderSEQ`, contém as atividades `receiveOrder` (requisição pedido), `invokeCheckCPF` (verificar a situação do CPF), `invokeCheckAddress` (consultar endereço), `invokeCheckCredit` (verificar a situação do crédito) e `replySendConfirmation` (enviar confirmação ao cliente), as quais serão executadas na ordem em que estão dispostas dentro do mesmo. Elementos desse tipo podem ser aninhados.

3.4.6.1 Invocação (`Invoke`)

`Invoke` é um elemento definido dentro de um elemento `sequence`, cuja finalidade é indenticar uma operação de um *Web Service* parceiro que será invocada durante a

execução do processo (ERL, 2005). Tal atividade é composta pelos seguintes atributos:

- **`partnerLink`:** contém o nome de um elemento `partnerLink` referente a um *Web Service*, configurado no processo WS-BPEL;

- **`portType`:** indenfica um elemento `portType` definido no arquivo WSDL do *Web Service*;

- **`operation`:** indentifica uma operação do *Web Service* definida no elemento `portType` do mesmo, para a qual será enviada uma requisição.

- **`inputVariable`:** indentifica um elemento `variable`, definido no processo WS-BPEL, utilizado para armazenar os dados de entrada passados para a operação, através da requisição do *Web Service*;

- **`outputVariable`**: indentifica um elemento `variable` definido no processo WS-BPEL, "utilizada quando a comunição é baseada em *request-response* MEP" (ERL, 2005, p. 574).

Na figura 28, apresenta-se um exemplo que ilustra um elemento `invoke`.

Figura 28: Exemplo de um elemento `invoque`.

```
<invoke name="invokeCreditCheck"
        partnerLink="CreditCheckPT"
        portType="crd:CreditInterface"
        operation="validateCreditCard"
        inputVariable="credCheckInputVar"
        outputVariable="creCheckOutputVar" />
```

Fonte: Do Autor.

No exemplo da figura 28, foi definido uma atividade `invoke` para realizar a validação do cartão de crédito de um cliente.

3.4.6.2 Recepção de Mensagens (`Receive`)

`Receive` é um elemento definido dentro de um elemento `sequence`, cuja finalidade é receber as mensagens provenientes dos *requests* feitos aos *Web Services* dos parceiros (OASIS, 2007). Tal atividade é composta pelos seguintes atributos:

- **`partnerLink`:** contém o nome de um elemento `partnerLink` referente a um *Web Service*, configurado no processo WS-BPEL;

- **`portType`:** indenfica um elemento `portType` definido no arquivo WSDL do *Web Service*;

- **`operation`:** indentifica uma operação do *Web Service* definido no elemento `portType` do mesmo;

- **`variable`:** indentifica um elemento `variable`, definido no processo WS-BPEL, para armazenar o conteúdo da mensagem proveniente da requisição feita ao *Web Service*;

- **`createInstance`:** se definido como "yes", permite ao processo que recebe a requisição criar uma nova instância do processo. Se definida como "*no*", a mensagem será consumida pela própria instância do processo.

Na figura 29, apresenta-se um exemplo que ilustra um elemento `receive`.

Figura 29: Exemplo de um elemento `receive`.

```
<receive name="receiveCustomerOrder"
         createInstance="yes"
         partnerLink="ord:CustomerPurchaseOrder"
         portType="CustomerPurchaseOrderPT"
         variable="CustomerPurchaseOrderVAR"
         operation="SendPurchaseOrder" />
```

Fonte: Do Autor.

Na figura 29, temos o exemplo de um elemento `receive`, definado para receber uma mensagem de uma requisição de pedido de compra feita por um cliente em um site de comércio eletrônico.

3.4.6.3 Resposta (`Reply`)

Resposta (`reply`) é um elemento definido dentro de um elemento `sequence`, o qual é utilizado em conjunto com um elemento `receive`, cuja finalidade é implementar uma operação *request-response* (requisição-resposta) em um `partnerLink`. "A variável especificada contém os dados da resposta, seja bem-sucedida ou não, retornando o resultado ao invocador do *Web Service*". (OASIS, 2007, p. 12). Tal atividade é composta pelos seguintes atributos:

- **`partnerLink`:** contém o nome de um elemento `partnerLink` referente a um *Web Service*, configurado no processo WS-BPEL;

- **`portType`:** indenfica um elemento `portType` exibido na atividade `receive`;

- **`operation`:** indentifica a mesma operação definida na atividade `receive`;

- **`variable`:** indentifica um elemento `variable` definido no processo WS-BPEL para armazenar o conteúdo da mensagem proveniente da requisição feita ao *Web Service*;

- **`messageExchange`:** "permite que a atividade `reply` seja explicitamente associado com uma atividade de mensagem capaz de receber a mensagem, como na atividade `receive`" (ERL, 2005, p. 576).

Nas figuras 30 e 31 mostra-se o exemplo de uma atividade `reply`, definida para interagir com uma atividade `receive`, definidas para tratarem de um pedido de cliente, em comércio eletrônico.

Figura 30: Elemento receive.definido para interagir com o elemento `reply`.

```
<reply name="replySendConfirmation"
       partnerLink="receiveOrder"
       operation="submitOrderProcess"
       variable="orderVariableResult" />
```

Fonte: Do Autor.

Figura 31: Elementos receive e reply declarados no elemento `sequence`.

```
<sequence name="CustomerMessageOrderSEQ">
    <receive name="receiveOrder" ... />
    <invoke name="invokeCheckCPF" ... />
    <invoke name="invokeCheckAddress" ... />
    <invoke name="invokeCreditCheck" ... />
    <reply name="replySendConfirmation" ... />
</sequence>
```

Fonte: Do Autor.

Na figura 30, tem-se um elemento `reply`, denominado `replySendConfirmation`, configurado para interagir com a atividade `receive` (declarada no elemento `sequence` da figura 31), através do `partnerLink receiveOrder` e da operação `submitOrderProcess`, que devolverá o resultado da operação na variável (`variable`) `replySendConfirmation`.

3.4.7 Tratamento de erros (`FaultHandlers`)

`FaultHandlers` é um elemento dentro de um processo WS-BPEL utilizado para tratar as exceções que podem ocorrer durante a execução de um processo de negócios. Dentro da estrutura do mesmo, há dois outros elementos que precisam ser configurados para permitir a manipulação de exceções, que são: `catch` e `catchAll` (OASIS, 2007). Na figura 32, apresenta-se um exemplo desse elemento.

Figura 32: Exemplo de um elemento `faultHandlers`.

```
<faultHandlers>
     <catch faultName="CustomerPurchaseOrderEX"
            faultVariable="purchaseOrderFaultVar">
     ...
     </catch>
     <catchAll>...</catchAll>
</faultHandlers>
```

Fonte: Da Autor.

3.4.8 Elementos para Controle de Fluxo `if`, `else` e `elseIf`

Igual a outras linguagens de programação, como C++ e Java, a linguagem WS-BPEL dispõe de uma série de elementos destinados à definição de estruturas de controle de fluxo de dados, como: `if`, `else` e `elseif`. Na figura 33, apresenta-se um exemplo que mostra estes elementos.

Figura 33: Exemplo de estruturas para controle de fluxo `if`, `else` e `elseif`.

```
<if xmlns:order="http://example.com/order"
    xmlns:FLT="http://example.com/creditChecher">
      <condition>
          bpel:checkCustomerCredit(custCredInfo) = True
      </condition>
      <flow>
          bpel:generateOrder('orderResult')
      </flow>
      <else>
          <condition>
              msg:sendCustomerNotification('Cancelar Operação')
          </condition>
      </else>
</if>
```

Fonte: Da Autor.

Na figura 33, apresenta-se como exemplo, uma parte de um processo de pedido de compra de um site de comércio eletrônico, no qual o crédito do cliente é validado através da operação `checkCustomerCredit`, que recebe como parâmetro os dados do cartão de crédito do mesmo. Caso a validação de crédito retorne uma confirmação positiva (*True*), a operação `generateOrder`, irá calcular e gerar o pedido. Caso a validação retorne uma confirmação negativa (*False*), o sistema irá notificar o cliente que o cartão de crédito foi recusado.

Como se pode perceber, a linguagem WS-BPEL fornece inúmeros recursos que permitem definir com completeza, qualquer processo de negócio executável e suas atividades para a construção de um Orquestrador de *Web Services*.

Além dos elementos estudados nesta seção, há outros elementos que constituem a linguagem WS-BPEL, que não serão abordados aqui, uma vez que não é objetivo deste trabalho estudar todos os detalhes desta linguagem.

3.4.9 Motores WS-BPEL

Para a construção e gerenciamento de soluções baseadas em Orquestração de *Web Services*, há um grande número de *engines* (motores), destinados à criação, interpretação e execução dos *workflows* (fluxos de trabalho) dos processos de negócios WS-BPEL.

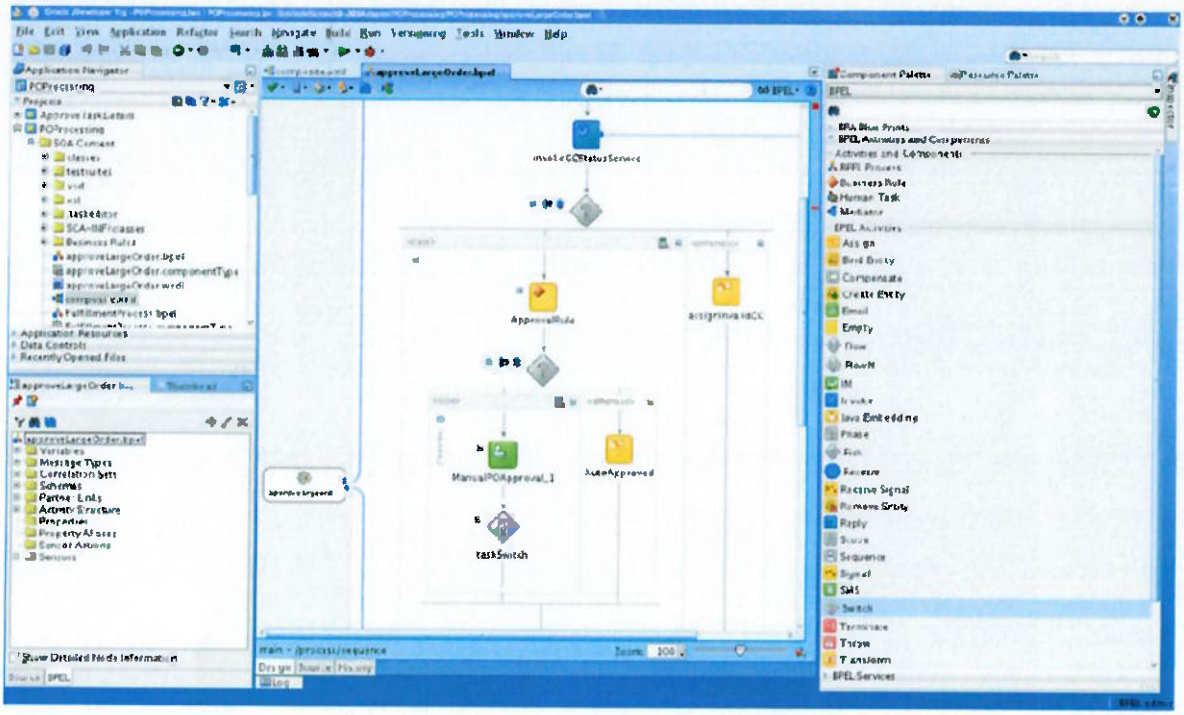
Além dos WS-BPEL *engines* proprietários, como: Biztalk Server (Microsoft), Oracle BPEL Process Manager (Oracle) e WebSphere Process Server (IBM) e ActiveOS (Active Endpoints), há os que são distruibuídos gratuitamente, como: Apache ODE (Apache), OW2 Orchestra (OW2), Open ESB (Open ESB Community), Petals BPEL Engine (Petals Link) e ActiveBPEL (Active Endpoints).

Todos esses WS-BPEL *engines* são incorporados em ambientes de desenvolvimento integrado (IDE - *Integrated Development Enviroment*), nos quais são fornecidos inúmeros componentes visuais que permitem construir de forma simples, os fluxos de trabalho (*workflows*) dos processos de negócio WS-BPEL. Além disto, os *engines* geram, automaticamente, a configuração dos arquivos WSDL e WS-BPEL associados aos processos gerados.

Os principais IDEs são: JDeveloper (Oracle), Visual Studio .NET (Microsoft), Eclipse (IBM), NetBeans (Oracle), JBoss Tools (JBoss) e Orchestra, dentre outros, que permitem a criação de processos de negócios executáveis, através de componentes gráficos.

Na figura 34, apresenta-se a imagem de um processo de negócios definido no software BPEL Designer, da JDeveloper IDE (Oracle).

Figura 34: Software JDevelper BPEL Designer da Oracle.



Fonte: http://www.oracle.com/technetwork/middleware/soasuite/bpel-01-098840.html

3.5 Sistemas de Controle de Acesso

Nesta seção, são estudados os principais conceitos sobre segurança de informação aplicado em Sistemas de Controle de Acesso Físico e Lógico e uma definição dos fluxos de trabalho (*workflows*) destes dois modelos de acesso.

3.5.1 Definição

De acordo com Kim e Solomon (2014, p. 110), "Controle de Acesso são métodos usados para restringir e permitir acesso a certos itens, como automóveis, casas, computadores ou mesmo telefone celular".

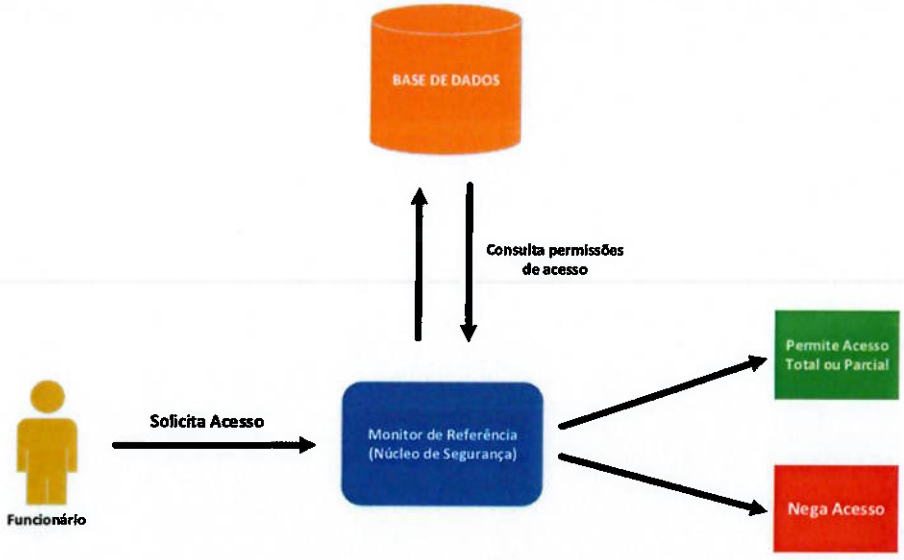
O termo Controle de Acesso vem sendo utilizado há décadas, quando se fala de garantir a segurança de acesso a algo ou um local, onde deseja-se conceder ou restringir o acesso a uma pessoa ou grupo de indivíduos, de acordo com a permissão concedida aos mesmos.

Na computação, o controle de acesso é motivado pela necessidade de divulgar acesso à informação e disponibilizar recursos computacionais e serviços somente às entidades autorizadas, entidade que se refere a um agente ativo que pode inicializar ou realizar uma tarefa computacional (BENANTAR, 2006).

Com a evolução tecnológica, tanto do software, quanto do hardware, inúmeros sistemas de software e equipamentos eletrônicos têm sido criados, para que, em conjunto, forneçam uma solução integrada que permita o controle de acesso, seja físico (acesso aos locais dentro da empresa), ou lógico (acesso aos sistemas de informação).

Assim, as empresas utilizam Sistemas de Controle de Acesso (físico ou lógico), para determinar o que os colcaboradores podem ou não fazer, quais recursos podem acessar e quais operações os mesmos podem realizar. Para isto, estes sistemas utilizam várias tecnologias, como: senhas, geradores de código (*tokens*) em *hardware*, equipamentos biométricos, certificados dentre outros (KIM; SOLOMON, 2014). Na figura 35, se tem um diagrama esquemático que ilustra o funcionamente padrão de uma Sistema de Controle de Acesso.

Figura 35: Visão geral de um Sistema de Controle de Acesso.



Fonte: Kim e Solomon (2014).

Na figura 35 o diagrama esquemático mostra um usuário que solicita acesso a um determinado ativo (físico ou lógico), no qual o controlador de acesso irá permitir ou não acesso ao mesmo.

De acordo com Kim e Solomon (2014), o controle de acesso pode ser dividido em quatro partes:

- **Autorização:** identifica-se quem está autorizado para realizar o acesso e o quê, exatamente, pode utilizar;
- **Identificação:** define-se como tais indivíduos podem ser identificados;
- **Autenticação:** define-se como suas identidades podem ser verificadas;
- **Responsabilização:** como as ações de um usuário podem ser acompanhadas, de modo a garantir que a pessoa que faz mudanças em sistemas possam ser identificadas.

Essas quatro partes do Controle de Acesso são divididas em duas fases:

Fase 1 - Definição da Política: determina quem tem acesso a quais sistemas ou que recursos pode utilizar.

Fase 2 – Imposição da Política: concede ou rejeita solicitações de acesso com base nas autorizações definidas na fase 1. Os processos de identificação, autorização e responsabilização operam nesta fase.

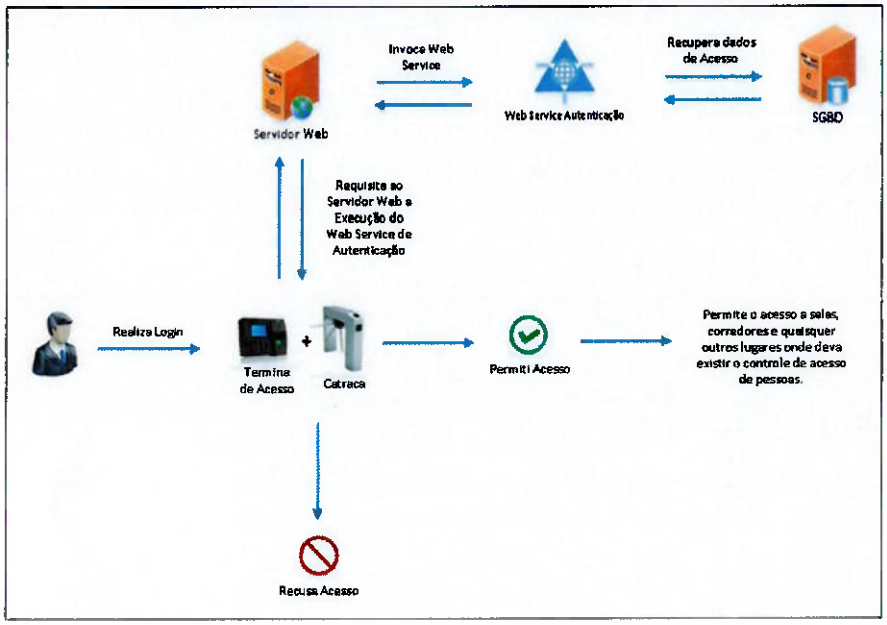
3.5.2 Tipos de Controle de Acesso

Os Sistemas de Controles de Acesso podem ser classificados em dois tipos:

- **Físico:** voltado ao controle de acesso a locais em prédios, salas ou quaisquer áreas protegidas;
- **Lógico:** voltado ao controle de acesso a sistemas computacionais ou rede de computadores.

Nas figuras 36 e 37, tem-se os dois diagramas esquemáticos que representam esses dois tipos de Controle de Acesso.

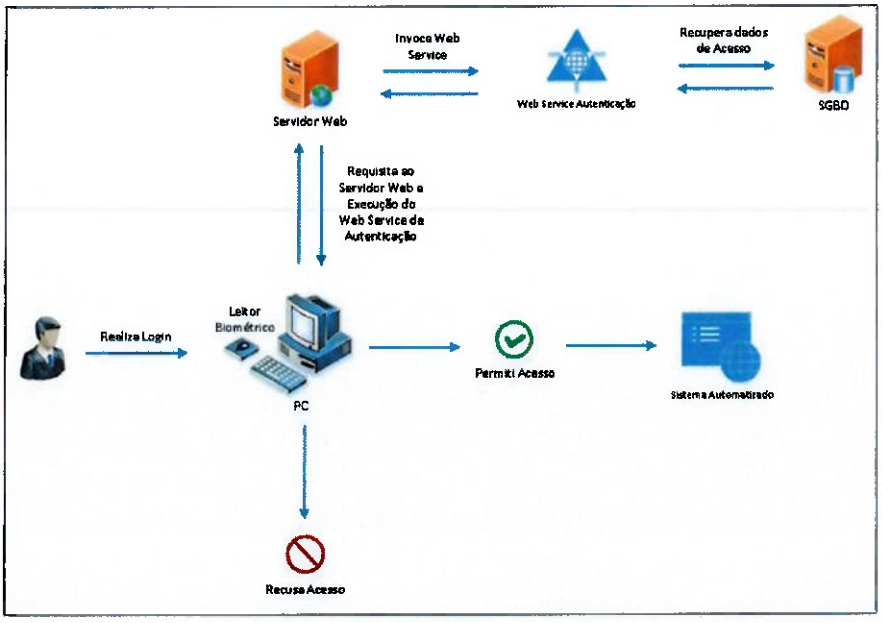
Figura 36: Diagrama esquemático de um Controle de Acesso Físico.



Fonte: Do Autor.

Na figura 36, o diagrama esquemático mostra que um usuário solicita acesso a uma área restrita (prédio, sala etc.) em uma catraca ou terminal de acesso, através de um leitor biométrico equipado nestes equipamentos.

Figura 37: Diagrama esquemático de um Controle de Acesso Lógico.



Fonte: Do Autor.

Na figura 37, tem-se um diagrama esquemático no qual um usuário solicita acesso a um sistema computacional, em um computador equipado com um leitor biométrico, através de um leitor biométrico conectado ao computador.

3.6 Considerações finais

Neste capítulo foram apresentadas todas as conceituações teóricas consideradas importantes para o trabalho a ser executado: a Arquitetura Orientada a Serviços e a tecnologia associada a ela, bem como os principais processos de desenvolvimento de sistemas que se baseiam em SOA. Detalhes dos padrões XML, XSD Schema, protocolo SOAP e os padrões WSDL e UDDI também são detalhados. Conceitos sobre Composição e Orquestração de *Web Services* foram apresentados. A finalidade dessa apresentação foi mostrar como um sistema baseado em SOA pode ser construído, apesar de que o foco desse trabalho não está na parte de Construção de aplicações e sim, na parte de Análise e Projeto dessas aplicações. Ferramentas tecnológicas para criação, interpretação e execução de *workflows* foram mencionadas.

Finalmente, encerrou-se o capítulo com uma discussão sobre Controle de Acesso, que é o foco dessa monografia. Esses conceitos são fundamentais para o Estudo de Caso que será apresentado no capítulo seguinte.

4. ESTUDO DE CASO

Para aplicação dos conceitos sobre Análise e Projeto de um Orquestrador de *Web Services* com o WS-BPEL, será realizado o estudo de caso para aplicação dos conceitos abordados durante esse trabalho.

4.1 Caso Metalúrgica Steel Auto-Peças Ltda[1]

A Metalúrgica Steel Auto-Peças Ltda, é uma empresa nacional de médio porte, que atua no ramo de produção de peças automotivas há 15 anos. Atualmente, mantém empregados cerca de 500 funcionários que atuam nas áreas de Produção, Logística, Recursos Humanos, Administrativo, Controladoria e Tecnologia. A empresa possui negócios em vários estados do território nacional.

Hoje, a empresa possui um sistema de segurança, no qual o controle de acesso de pessoas e veículos é realizado somente na portaria, através de uma guarita onde ficam alguns vigilantes terceirizados. Dentro da guarita, o monitoramente é realizado através de um sistema CFVT (Circuito Fechado de Televisão) que exibem as imagens das câmeras situadas dentro e fora da empresa.

Apesar da empresa possuir câmeras espalhadas em pontos estratégicos, não há o controle de acesso físico computadorizado em áreas que deveriam ser restritas, como: linha de produção, departamento de tecnologia, salas dos gestores e alta direção dentre outros. Também não há um sistema informatizado que registre o acesso dos funcionários nos vários ambientes dentro da empresa.

No intuito de modernizar e aumentar a segurança de acesso, a Metalúrica Steel Auto-Peças deseja implantar um Sistema de Controle de Acesso, através do qual, seja possível registrar a hora de entrada e saída dos funcionários nos principais Pontos de Acesso dentro da empresa. Cada Ponto de Acesso, deverá estar

---

[1] **A Metalúrgica Steel Auto-Peças Ltda é uma empresa fictícia criada somente como apoio ao estudo aqui realizado.**

equipado com um equipamento para leitura biométrica, que também permita a leitura de cartões magnéticos, permitindo que, tanto os Funcionários, quanto os Visitantes, consigam solicitar acesso às Áreas de Segurança.

O sistema deverá possuir uma tela de monitoramento, através da qual, seja possível monitorar em tempo real, a imagem das câmeras distribuídas dentre e fora da empresa, bem como, o registro de acesso dos funcionários em todos Pontos de Acesso. Tal recurso, será utilizado nos computadores da Sala de Segurança onde ficam os vigilantes terceirizados.

Além do monitoramento de acesso dos funcionários, o sistema deverá fornecer um módulo gerencial, através do qual, seja possível cadastrar usuários (funcionários e visitantes) e definir em quais locais de acesso os mesmos poderão acessar ou não. Deseja-se que seja possível associar os usuários em grupos específicos, de acordo com o perfil do mesmo.

O sistema deverá permitir um controle de visitantes, para os quais, deverão ser fornecidos um crachá provisório, através do qual, os mesmos poderão acessar os pontos de acesso nos locais onde forem visitar. Além disto, este controle deverá estar integrado ao Sistema de Agendamento de Visitas utilizado pela empresa.

4.2 Equipe Responsável pelo Projeto

A empresa Steel Auto-Peças possui um Departamento de Tecnologia que responde pelo desenvolvimento e manutenção de todos as aplicações de software utilizados pela mesma, incluindo a administração de infraestrutura de redes e gerenciamento de banco de dados. Sendo assim, se decidiu designar uma das equipes de desenvolvimento de software para ser responsável por todo os Ciclo de Vida do projeto do Sistema de Controle de Acesso Físico. Para apoiar a equipe na compra, implantação, manutenção e configuração dos equipamentos para controle de acesso, durante e depois da implantação do Projeto, foram contratados inicialmente, um Engenheiro Eletrônico e dois Técnicos de Eletrônica.

4.3 Metodologia Adotada

Atualmente, a maior parte das aplicações de software utilizados pela empresa, incluindo as aplicações do legado, são apoiados através de uma Arquitetura Orientada a Serviços (SOA), com Serviços integrados em um barramento (ESB - *Enterprise Service Bus*). No intuito de reutilizar os Serviços existentes através da integração com as aplicações de softwares disponíveis, decidiu-se pela utilização de SOA para a Análise e Projeto do Sistema de Controle de Acesso.

Neste sentido, o Arquiteto SOA decidiu utilizar a metodologia MAPOS, do autor Fugita (2009), para gerenciar todas as etapas do Ciclo de Vida do projeto SOA em questão. Tal decisão foi tomada levando em conta a redução de custo e facilidade de execução das etapas deste método.

Também, foram selecionados um grupo de ferramentas de software para a Análise e Projeto, que são:

- *Oracle Business Process Management Suite* 12c, da empresa Oracle, para a modelagem dos processos *as-is* e *to-be*;
- *Enterprise Architect*, da empresa Sparx Systems, para a modelagem dos Serviços e Classes de Mensagem;
- *Visual Studio .NET* 2013, da empresa Microsoft, para a implementação dos Serviços e definição dos arquivos WSDL;

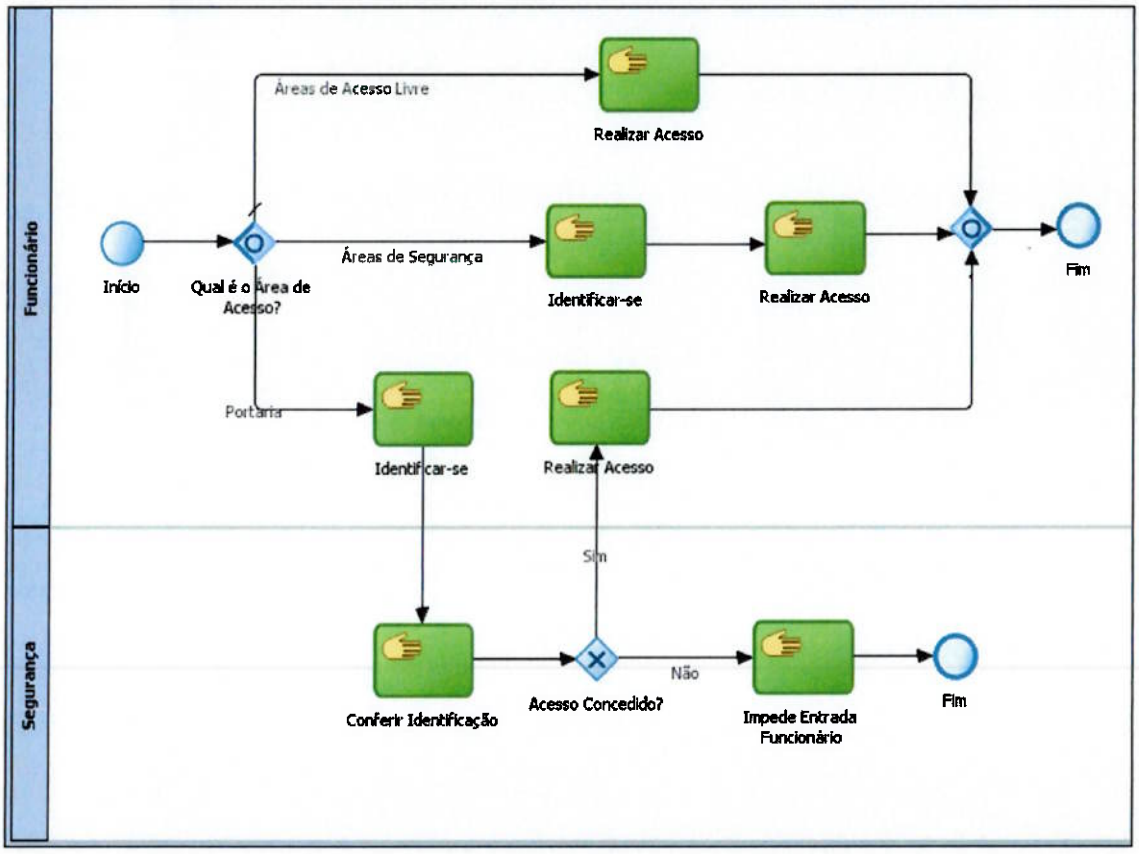
4.4 Modelagem de Processo *as-is* e *to-be*

A partir da análise realizada pelo Analista de Sistemas do atual esquema de segurança da empresa Steel Auto-Peças, e das necessidades de modernização do novo sistema de segurança a ser implantado, foram identificados os seguintes processos: Gerenciar Acesso Físico Funcionário, Gerenciar Acesso Físico Visitante, Monitorar Acesso Físico Usuários, Gerenciar Usuários, Gerenciar Visistantes, Gerenciar Funcionários, Gerenciar Cadastro Biométrico, Gerenciar Cartões de Identificação, Gerenciar Grupos de Acesso, Gerenciar Pontos de Acesso, Gerenciar Áreas de Acesso e Gerenciar Câmeras.

Para representação das etapas de Análise e Projeto, foram selecionados os dois principais processos: Gerenciar Acesso Físico Funcionário e Gerenciar Acesso Físico Visitante, construindo, para ambos os processos, os modelos *as-is* e *to-be*, fazendo uma comparação entre eles.

O modelo de processo Gerenciar Acesso Físico Funcionário, descreve os passos percorridos por um Funcionário, desde entrada na Portaria, precisando identificar-se à Segurança para solicitar entrada na empresa, até o momento em que o mesmo está dentro da empresa e precisa solicitar acesso às Áreas de Segurança (local de trabalho, salas, depertamentos etc). Na Figura 38, temos o diagrama *as-is* que representa tal processo.

Figura 38: Diagrama do modelo de processo as-is Gerenciar Acesso Físico Funcionário.



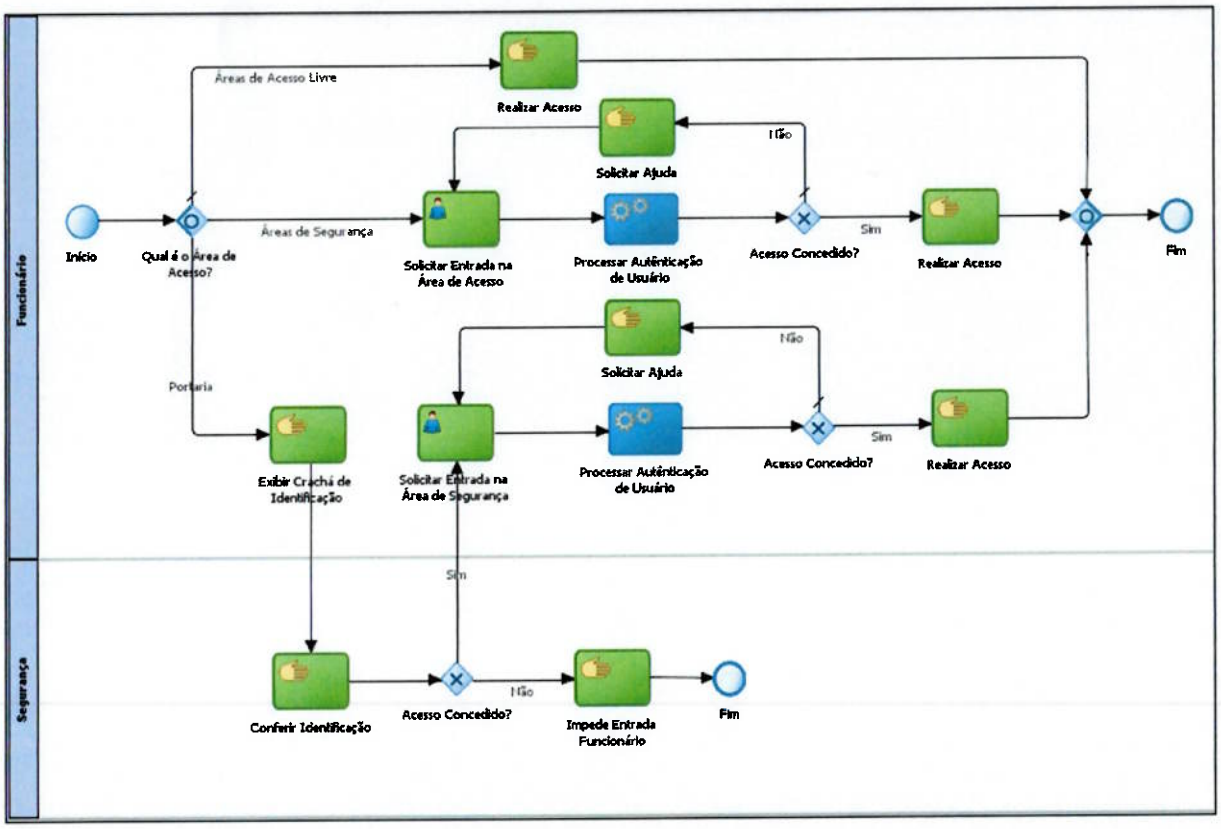
Fonte: Do Autor.

Analisando o modelo de processo de negócio *as-is* "Gerenciar Acesso Físico Funcionário", percebe-se que, em todas as atividades o pedido e concessão de acesso é realizado manualmente, não havendo utilização de equipamentos

eletrônicos ou informática para registro da movimentação do Funcionário entre os Pontos de Acesso, não sendo possível rastrear os acessos.

Quando a Área de Acesso é a Portaria, o Funcionário identifica-se mostrando à Segurança o Cartão de Identificação de Usuário (Crachá), e, caso seja autorizado a entrar na Empresa, realiza o acesso sem precisar solicitar acesso em algum equipamente de segurança. Se a Área de Acesso for uma Área de Segurança (Local de Trabalho, Salas, outros Departamentos etc), o Funcionário identifica-se e, caso autorizado, realiza o acesso. Caso seja um Local de Livre Acesso (Restaurante, Área de Lazer etc), o mesmo acesso livremente. Baseado nestes dados, foi proposto o modelo de processo de negócio *to-be* da figura 39.

Figura 39: Diagrama do modelo de processo to-be Gerenciar Acesso Físico Funcionário.



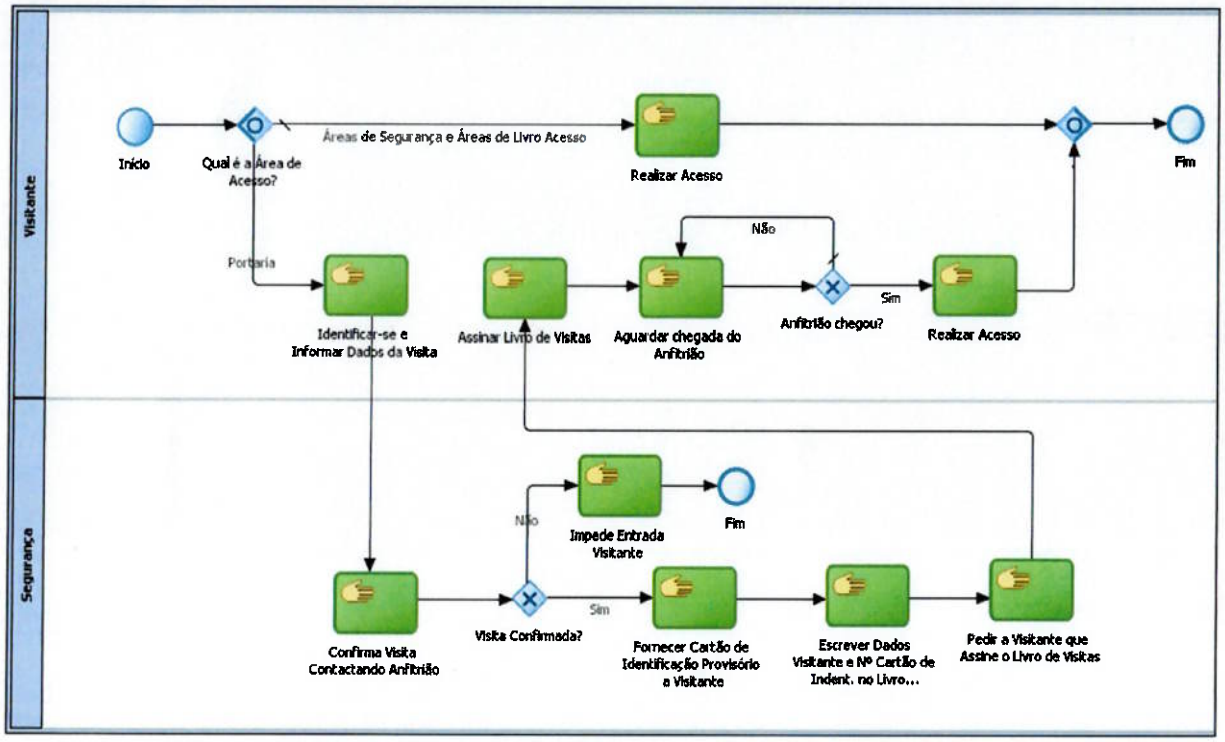
Fonte: Do Autor.

No modelo de processo de negócio *to-be*, o Funcionário precisa autênticar-se através de um leitor biométrico, em Pontos de Acesso informátizados localizados na entrada das Áreas de Segurança (Local de Trabalho, Salas, Departamentos etc), onde tal evento será registrado numa base de dados para futura auditorias. Sendo assim, todos os Funcionários deverão estar previamente cadastrados como Usuários

do Sistema de Controle de Acesso e possuirem permissão de acesso às Áreas de Segurança dentre da Empresa.

O modelo de processo Gerenciar Acesso Físico Visitante, descreve os passos percorridos por um Visitante, desde entrada na Portaria, precisando identificar-se à Segurança para solicitar entrada na empresa, até o momento em que o mesmo está dentro da empresa e precisa solicitar acesso às Áreas de Segurança (local de trabalho, salas, depertamentos etc). Ns Figura 40, temos o diagrama *as-is* que representa tal processo.

Figura 40: Diagrama do modelo de processo as-is Gerenciar Acesso Físico Visitante.



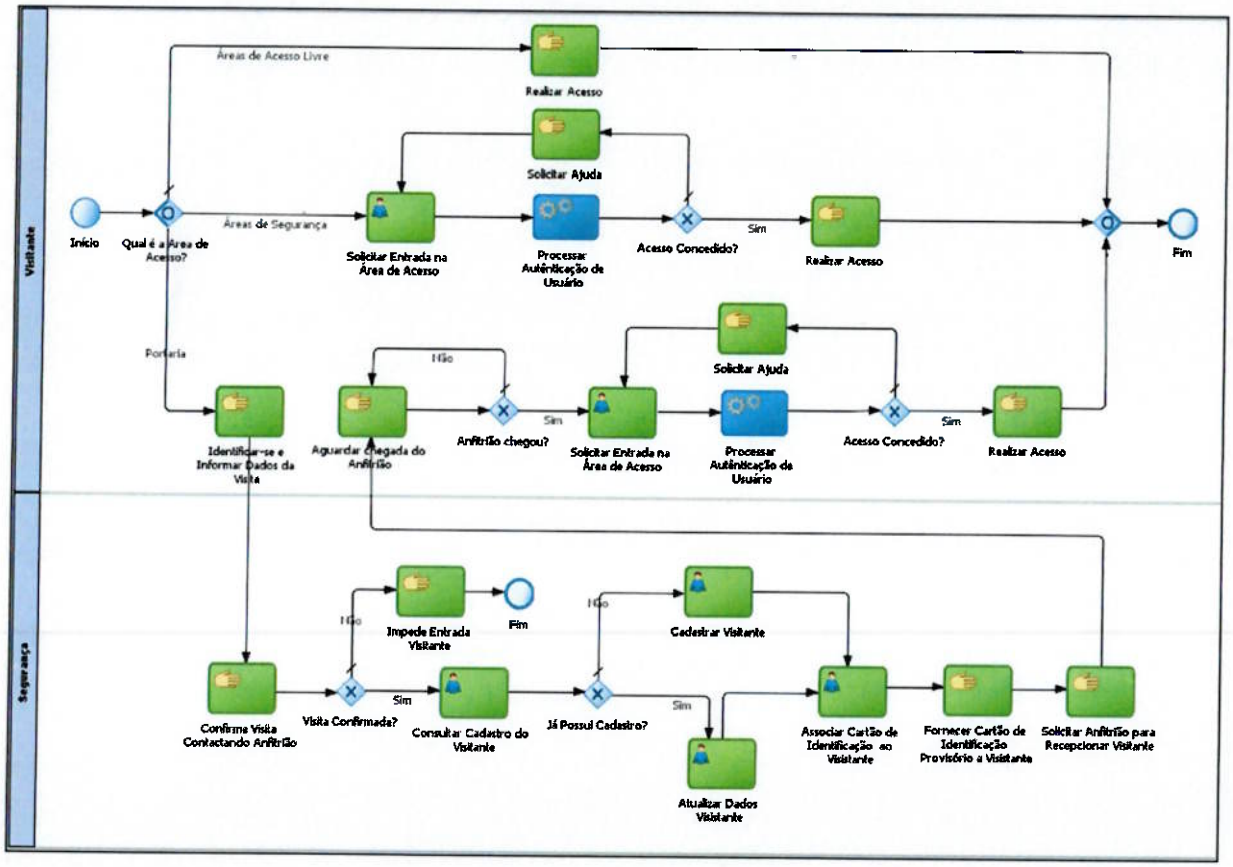
Fonte: Do Autor.

No modelo de processo de negócio *as-is*, todas as atividades são realizadas manualmente, não havendo algum controle computacional envolvido. Neste cenário, está sendo representado os dois caminhos possíveis de acesso realizado pelo Visistante: um, no qual o Visistante chega a Portaria e precisa identificar-se, e outro, quando já está dentro da Empresa e já não precisa solicitar permissão de acesso às Áreas de Segurança ou de Livre Acesso, já que está acompanhado com a pessoa que o recepcionou.

Quando o Visistante identifica-se na Portaria, o Segurança contacta a pessoa que irá recepcionar o Visitante, para confirmar, se de fato, a visita está agendada. Caso seja verdadeiro, o Segurança solicita a presença da pessoa que irá recepcionar o Visitante na Portaria, registra o nome completo do Visitante em um Livro de Visitas junto ao código do Cartão de Identificação de Visitantes fornecido ao mesmo, pede para o Visitante assinar o livro e fornece o cartão ao Visitante, pedido que o mesmo o devolva na saída da Empresa.

Com base nestes dados, os Analista de Sistemas propôs algumas mudanças no modelo de processo de negócio *as-is*, que contempla a utilização de um sistema informatizado, utilizando Pontos de Acesso eletrônicos e o registro de acessos em base de dados, como é mostrado na figura 41.

Figura 41: Diagrama do modelo de processo to-be Gerenciar Acesso Físico Visitante.



Fonte: Do Autor.

No modelo de processo de negócio *to-be*, foram inclusas algumas atividades, para as quais, é necessário a utilização de um Sistema de Controle de Acesso, para a solicitação de permissão de acesso à usuário, consulta de agendamento de visitas,

atualização e cadastro de visitantes e controle de cartões de identificação de visitantes, no qual é associado o Visistante ao Cartão de Identificação de Usuário provisório fornecido ao mesmo.

Através das atividades controladas via Sistema de Controle de Acesso, é possível um melhor controle das visitas, sendo possível gerenciar toda fluxo de acesso do Visitante às intalações da Empresa. Também, fornece recursos para gerenciamento de visitas, através do qual é possível cadastrar, atualizar, consultar e excluir agendamentos.

Uma vez definido todos os diagramas BPMN dos processos de negócio identificados, será realizado a Especificação das Tarefas das atividades do modelo de processo de negócio *to-be* "Gerenciar Acesso Físico Funcionário" e "Gerenciar Acesso Físico Visitantes, como é mostrado nos quadros 3 ao 8.

Quadro 3: Especificação da Tarefa Solicitar Entrada no Ponto de Acesso.

Tipo de Informação	**Descrição**
Nome da Tarefa	Solicitar Entrada na Área de Acesso
Objetivo	Esta tarefa descreve os passos necessários para solicitação de permissão de acesso a um determinado ambiente através do ponto de acesso eletrônico.
Tipo de Tarefa	Tarefa com interação humana.
Fluxo de Trabalho	1. A tarefa é iniciada quando o usuário autêntica-se no Ponto de Acesso. 2. Sistema recupera os dados do Usuário associado ao mesmo. 3. Sistema solicita permissão de acesso invocando o Serviço de Autêticação de Usuários, passando os dados do Usuário e código do Ponto de Acesso. 4. O Serviço processa a solicitação de acesso e retorna resultado "Acesso Permitido". 5. Sistema destrava o Ponto de Acesso, liberando a entrada do Usuário. 6. Sistema registra histórico de acesso, utilizado a data, hora, dados do Ponto de Acesso e dados do Usuário. 7. A tarefa é finalizada.
Fluxo de Exceção FE001	1. Sistema não consegue realizar leitura do cartão. 2. Sistema retorna ao passo 1 do fluxo de trabalho.
Fluxo Alternativo FA001	1. Sistema retorna resultado "Acesso Negado". 2. Sistema registra histórico de acesso, utilizado a data, hora, dados do Ponto de Acesso e dados do Usuário 3 . A tarefa é finalizada.

Fonte: Do Autor.

Quadro 4: Especificação da Tarefa Processar Autênticação Usuário.

Tipo de Informação	Descrição
Nome da Tarefa	Processar Autênticação de Usuário
Objetivo	Esta tarefa descreve os passos necessários para realizar a autenticação do usuário através dados dados e permissões de acesso concedidos ao mesmo.
Tipo de Tarefa	Tarefa com interação do computador
Fluxo de Trabalho	1. A tarefa é iniciada quando o Ponto de Acesso solicita permissão de acesso ao usuário, passando os dados do Usuário e o código do Ponto de Acesso. 2. O Sistema verifica se Usuário possui permissão de acesso a Árena de Acesso na qual deseja entrar; 3. O Sistema devolve resultado "Acesso Permitido"; 4. A tarefa é finalizada.
Fluxo Alternativo FA001	1. Sistema devolve resultado "Acesso Negado", pois Usuário não possui permissão de acesso ao ambiente que deseja entrar. 2. A tarefa é finalizada.

Fonte: Do Autor.

Quadro 5: Especificação da Tarefa Consultar Cadastro Visitante.

Tipo de Informação	Descrição
Nome da Tarefa	Consultar Cadastro do Visitante
Objetivo	Esta tarefa descreve os passos que o Segurança percorre para consulta de dados do Visitante.
Tipo de Tarefa	Tarefa com interação humana.
Fluxo de Trabalho	1. A tarefa é iniciada quando o Segurança realiza a consulta registro do Visistante a partir do seu número do documento RG. 2. O Sistema retorna registro do visitante. 3. A tarefa é finalizada.
Fluxo Alternativo FA001	1. O Sistema não encontra os dados sobre o visitante. 2. A tarefa é finalizada.

Fonte: Do Autor.

Quadro 6: Especificação da Tarefa Cadastrar Visitante.

Tipo de Informação	Descrição
Nome da Tarefa	Cadastrar Visitante
Objetivo	Esta tarefa descrê os passos necessários que o Segurança percorre para realizar o cadastro de um Visitante.

continua

continuação

Tipo de Informação	Descrição
Tipo de Tarefa	Tarefa com interação humana.
Fluxo de Trabalho	1. A tarefa é iniciada quando o Segurança inicia o Cadastro. 2. O Segurança preenche os dados sobre o Visitante. 3. O Sistema valida os dados preenchidos. 4. O Sistema gravas os dados. 5. A tarefa é finalizada.
Fluxo Alternativo FA001	1. Dados do Visitante foi preenchido de forma icorreta. 2. Sistema retorna ao passo 2 do fluxo de trabalho.

Fonte: Do Autor.

Quadro 7: Especificação da Tarefa Alterar Visitante.

Tipo de Informação	Descrição
Nome da Tarefa	Atualizar Visitante
Objetivo	Esta tarefa descrê os passos necessários que o Segurança percorre para realizar a atualização do registro de um Visitante.
Tipo de Tarefa	Tarefa com interação humana.
Fluxo de Trabalho	1. A tarefa é iniciada quando o Segurança seleciona a edição do registro do Visitante. 2. O Segurança preenche os dados que deseja alterar. 3. O Sistema valida os dados preenchidos. 4. O Sistema gravas os dados. 5. A tarefa é finalizada.
Fluxo Alternativo FA001	1. Dados do Visitante foi preenchido de forma icorreta. 2. Sistema retorna ao passo 2 do fluxo de trabalho.

Fonte: Do Autor.

Quadro 8: Especificação da Tarefa Associar Cartão de Identificação Provisório a Visistante.

Tipo de Informação	Descrição
Nome da Tarefa	Associar Cartão de Identificação ao Visitante
Objetivo	Esta tarefa descreve os passos necessários que o Segurança precisa para associar o Cartão de Identificação de Usuário ao Visitante no cadastro de Visitas.
Tipo de Tarefa	Tarefa com interação humana.
Fluxo de Trabalho	1. A tarefa é iniciada quando o Segurança seleciona a edição do registo do Visitante. 2. O Segurança consulta listagem de cartões de identificação disponíveis. 3. O Segurança seleciona um dos cartões de identificação provisório.

continua

continuação

Tipo de Informação	Descrição
	4. O Segurança seleciona a Área de Acesso inicial e Área de Acesso de destino, nas quais o Visitante precisará acessar. 5. O Sistema seleciona monta lista de Áreas de Acesso entre a Área de Acesso inicial até a Área de Acesso de destino e concende permissão de acesso a estas áreas ao Visitante. 6. O Sistema gera um registro de Usuário temporário e o associa ao Cartão de Identificação selecionado. 7. Sistema grava os dados da Visita. 8. A tarefa é finalizada.
Fluxo Alternativo FE001	1. Sistema falha ao salvar dados da Visita. 2. 6. A tarefa é finalizada.

Fonte: Do Autor.

4.5 Identificação do Serviços Candidatos

Com base nos modelos de processo *to-be* e na Especifiação das Tarefas, o Arquiteto SOA, com o apoio do Analista de Sistemas, realizou a identificação das Operações Candidatas que farão parte dos Serviços. Na tabela 5, segue listagem com as Operações identificadas.

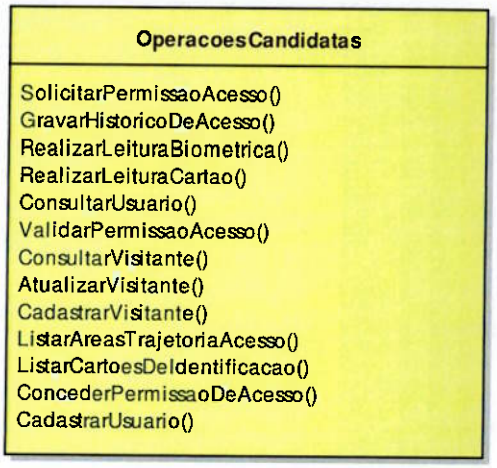
Quadro 9: Operações candidatas identificadas.

Tarefa	Operação Candidata
Solicita Entrada na Área de Segurança	Solicitar Permissão Acesso Gravar Histórico de Acesso RealizarLeituraBiometrica RealizarLeituraCartao Consultar Usuário
Processar Autênticação de Usuário	Validar Permissão de Acesso
Consultar Cadastro Visitante	Consultar Visitante
Atualizar Dados Visitante	Atualizar Visitante
Cadastrar Visitante	Cadastrar Visitante
Associar Cartão de Identificação ao Visitante	Listar Cartões de Identificação
	Listar Areas da Trajetoria de Acesso
	Conceder Permissao Acesso
	Cadastrar Usuário

Fonte: Do Autor.

Na Figura 42, segue a lista com definição das Operações Candidatas em um classe.

Figura 42: Operações candidatas identificadas.



Fonte: Do Autor.

Na etapa de Identifcação do Serviços Candidatos, o Arquiteto SOA descobriu que, algumas das Operações Candidatas identificadas, eram destinadas a operações de CRUD (*Create*, *Retrieve*, *Update* and *Delete*) como é o caso de CadastrarUsuario e Consultar Visitante. Desta forma, para conter tais operações, foram definidos os seguintes Serviços de Negócio baseado em Entidades: UsuarioService, CartaoDeIdentificacaoService, PontosDeAcessoService, FuncionarioService e AreasDeAcessoService. Também, foram identificados os Serviços de Negócio baseado em Atividades, que são: AgendamentoVisitasService e ControleDeAcessoFisicoService. E, por fim, identifcado o Serviço de Aplicação FuncoesGlobaisService, conténdo operações compartilhadas por todos os sistemas.

Na figura 43, segue a definição da inteface destes serviços.

Figura 43: Serviços Candidatos identificados.

Fonte: Do Autor.

Além da definição dos Serviços Candidatos, foram definidos as Classes de Mensagens utilizadas para passagem de parâmetros destes Serviços, como base nas informações sobre o modelo dos processos e na descrição da tarefas, como mostra a figura 44.

Figura 44: Classes de Mensagens identificadas.

Fonte: Do Autor.

4.6 Análise de *Gap*

Nesta fase, o Arquiteto SOA, auxiliado pelo Analista de Sistemas e Desenvolvedores, realizou um trabalho de identifcação das operações fornecidas através dos sistemas legados, que possam ser reutilizadas pelos Serviços Candidatos. Desta forma, foi consultado o repositório de serviços e toda documentação dos sistemas utilizados pela empresa. No Quadro 10, segue-se o resultado desta análise.

Quadro 10: Análise de gap dos Serviços Candidatos.

Serviço Candidato	Operação Candidata	Cenário de Reuso
ControleDeAcessoFisicoService	GravarHistoricoDeAcesso	Operação não existente
	ListarHistoricoAcesso	Operação não existente
	SolicitarPermissaoAcesso	Operação não existente
	ConcederPermissaoAcesso	Operação não existente
	RealizarLeituraBiometrica	Operação não existente
	ConsultarFuncionario	Operações realizadas pelo sistema legado (componente GerenciamentoFuncionarioBusiness do sistema "Gerenciamento de Funcionários", implementado em .NET)
UsuarioService	CadastrarUsuario	Operação não existente
	AtualizarUsuario	Operação não existente
	ExcluirUsuario	Operação não existente
	ConsultarUsuario	Operação não existente
	ListarFuncionarios	Operações realizadas pelo sistema legado (componente GerenciamentoFuncionarioBusiness do sistema "Gerenciamento de Funcionários", implementado em .NET)
AgendamentoVisitasService	CadastrarVisitante	Operações realizadas pelo sistema legado (componente AgendamentoVisitasNegocio do sistema "Agendamento de Visitas", implementado em .NET)
	AtualizarVisitante	
	ExcluirVisitante	
	ConsultarVisitante	
PontoDeAcessoService	CadastrarPontoDeAcesso	Operação não existente
	AtualizarPontoDeAcesso	Operação não existente
	ExcluirPontoDeAcesso	Operação não existente
	ConsultarPontoDeAcesso	Operação não existente

continua

continuação

Serviço Candidato	Operação Candidata	Cenário de Reuso
AreaDeAcessoService	CadastraAreaDeAcesso	Operação não existente
	AtualizarAreaDeAcesso	Operação não existente
	ExcluirAreaDeAcesso	Operação não existente
	ConsultarAreaDeAcesso	Operação não existente
	ListaAreaTrajetoriaDeAcesso	Operação não existente
CartaoDeIdentificacaoService	RealizarLeituraCartao	Operação não existente
	ListarCartoesIdentificacao	Operação não existente
	ConsultarCartaoDeIdentificacao	Operação não existente
	InativarCartaoDeIdentificacao	Operação não existente
	CadastrarCartaoDeIdentificacao	Operação não existente
FuncoesGlobaisService	IsCpfValido	Operações realizadas pelo sistema legado (classe "ValidacaoDocumentos", que faz parte do componente CoreLibrary, do sistema "SteelERPSystem" implementado em .NET)
	IsRGValido	

Fonte: Do Autor.

Como resultado deste trabalho, foi identificado como possibilidade de reuso, as operações oferecidas pelo componente AgendamentoVisitasNegocio, integrante da Camada de Negócio do sistema "Agendamento de Visitantes", o qual fornece todas as operações de CRUD de Visitantes e dados do Agendamento. Este componente é desenvolvido na plataforma .NET.

Para a operação de consulta de dados do Funcionário, foi identificado a possibilidade de reuso através do componente GerenciamentoFuncionarioBusiness, integrante da Camada de Negócio do sistema 'Gerenciamento de Funcionários", o qual fornece todas as operações necessários para o controle dos dados de Funcionários. Este componente é desenvolvido na plataforma .NET.

Para as operações IsCPFValido e IsRGValido, foi identificado uma classe chamada ValidacaoDocumentos, integrante do componente CoreLibrary, do sistema legado SteelERPSystem. Através desta classe, é possível realizar a validação de cpf, rg, CNPJ, Inscrição Estadual dentre outras. Este componente é desenvolvido na plataforma .NET.

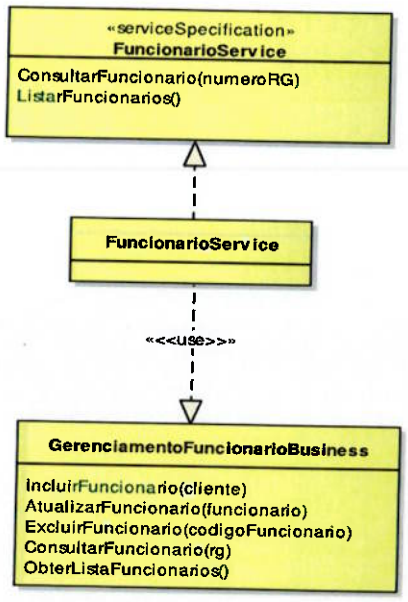
4.7 Análise de Realização

Na Análise de Realização, cada uma das Operações Candidatas são avaliadas quanto ao seu cenário de reuso, como apresentado no Quadro 10. Esta análise, foi realizado pelo Arquiteto SOA e o Analista de Sistemas, como se segue abaixo.

Serviço "ControleDeAcessoFisicoService" e "UsuarioService"

Para as operações de cadastro e atualização de funcionários, é necessário realizar uma consulta que retorne uma listagem de Funcionários, da qual, um registro é selecionado para que o código deste funcionário em questão, seja associado ao registro de Usuário do mesmo. Através da Análise de *Gap*, descobriu-se que tal funcionalidade já era oferecida pelo sistema "Gerenciamento de Funcionários", através do componente de negócio "GerenciamentoFuncionarioBusiness", mesmo caso do Serviço "ControleDeAcessoFisicoService", que também utiliza esse componente. Sendo assim, decidiu retirar as operações ConsultarFuncionarios e ListarFuncionarios dos Serviços "ControleDeAcessoFisicoService" e "UsuarioService", e passar para um novo Serviço a ser criado, chamado FuncionarioService, aumentado a coesão do projeto. Na figura 45, segue o resutado da Realização do Serviço "FuncionarioService".

Figura 45: Realização do Serviço FuncionarioService.

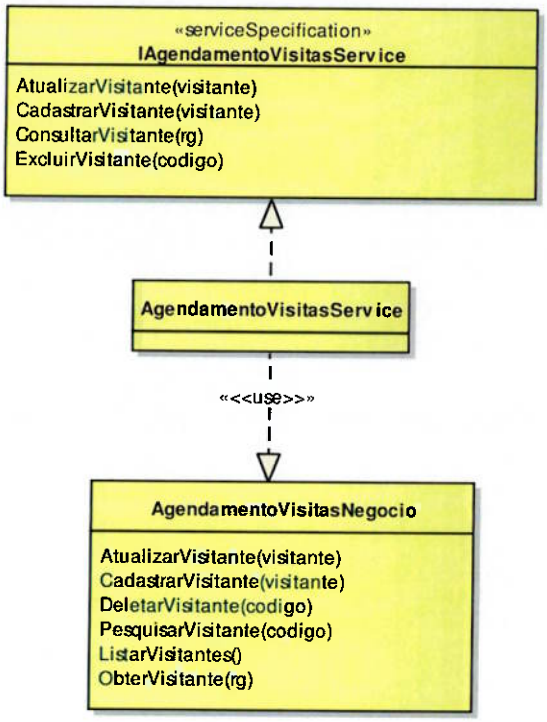


Fonte: Do Autor.

Serviço "AgendamentoVisitasService"

Na Análise de Realização Serviço "AgendamentoVisitasService", identificou-se no sistema legado que as operações deste Serviços, já estão implementadas no componente de negócio AgendamentoVisitasNegocio, integrante da Camada de Negócio do sistema "Agendamento de Visitantes". Na figura 46, segue o resutado da Realização do Serviço "AgendamentoVisitasService".

Figura 46: Realização do Serviço AgendamentoVisitasService.

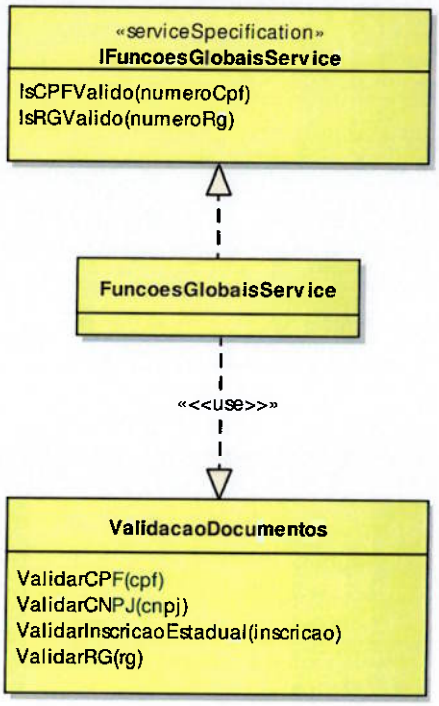


Fonte: Do Autor.

Serviço "FuncoesGlobaisService"

Na Análise de Realização, foi identificado uma classe chamada ValidacaoDocumentos, integrante do componente CoreLibrary, do sistema legado SteelERPSystem. Através desta classe, é possível realizar a validação de cpf, rg, CNPJ, Inscrição Estadual dentre outras. Sendo assim, foi possível utilizá-la para apoio ao Serviço global. Na figura 47, temos o resultado da Realização do Serviço "FuncoesGlobaisService".

Figura 47: Realização do Serviço FuncoesGlobaisService.



Fonte: Do Autor.

4.8 Projeto dos Serviços

A partir deste ponto, começa a Etapa de Projeto, começando na fase de Especificação dos Serviços, onde será realizado o detalhamento das Classes de Mensagem e das Interfaces de Serviços, bem como uma descrição técnica de cada uma das operações dos Serviços. Sendo assim, será realizado as seguintes etapas:

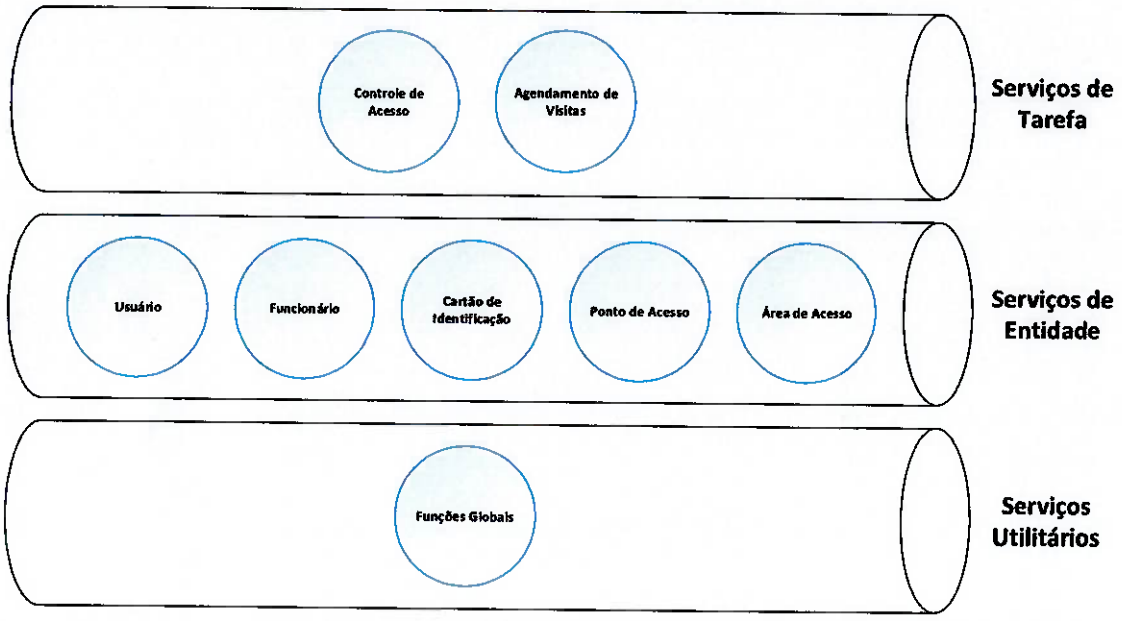
- Especificação de contrato de serviços;
- Especialização de realização de serviços;
- Veficação de princípios.

4.8.1 Especificação de Contratos de Serviços

Baseado nos Serviços e Mensagens que foram identificadas, o Arquiteto SOA realizou a especificação dos Esquemas XSD das Classes de Mensagem, e a descrição WSDL das Interfaces dos Serviços. No caso, tanto as especificações das

Classes de Mensagem, quanto, as especificações dos Serviços, são definadas nos Apêndices A e B respectivamente. Na figura 48, se segue a distribuição dos Serviços entre as camadas.

Figura 48: Distribuição dos Serviços entre as camadas de Serviços.



Fonte: Do Autor.

4.8.2 Verificação dos Princípios

Nesta etapa, for realizada a verificação de cada Serviço para analisar se os mesmos atendem aos princípios de SOA, podendo estar integrados ao portifólio de serviços da empresa. Foram envolvidos para esta atividade o Arquiteto SOA, o Analista de Sistemas e o Cordenador de Projetos.

4.8.3 Revisão dos Serviços

No intuito de garantir que todos os Serviços e Operações criados foram definidos corretamente, foi realizado uma revisão das especificações dos Serviços. Tal atividade foi realizada pelo Analista de Serviços.

4.8.4 Orquestração de Serviços

Todos os modelos de processo *to-be* definidos com o BPMN, através da ferramenta *Oracle BPM Suite*, foram utilizados para a definição do Orquestrador de *Web Services* do Sistema de Controle de Acesso Físico. Desta forma, foi utilizado a ferramenta *Oracle SOA Suite* para a construção do mesmo, utilizando como linguagem de processos de negócio o WS-BPEL.

4.8.5 Implementação e Teste

Uma vez implementados os Serviços, foram designados alguns desevolvedores para a realização de testes nos Serviços, a partir da Especificação de Requisítos fornecidas sobre os mesmo. Este teste contou com a participação do Analista de Serviços, que cordenou e orientou a realização deste trabalho.

4.8.6 Considerações do Capítulo

Neste capítulo, foi apresentado um Estudo de Caso, através do qual foi realizado todas as etapas de Análise e Projeto de uma Aplicação SOA destinado à construção de um Sistema de Controle de Acesso Físico. Para guiar estas etapas, foi utilizado o método MAPOS, do autor Fugita (2009).

Foi realizado a modelagem dos processos *as-is* e *to-be* Gerenciar Acesso Físico Funcionário e Gerenciar Acesso Físico Visitante, bem como a Especificação das Tarefas das principais atividades mapeadas no processo *to-be*. Em seguida, foi realizada a Identificação das Operações e Serviços Candidatos, para o qual, foi utilzados como insumos, os modelos de processo *to-be* e as Especificações das Tarefas.

Uma vez concluída a Identificação das Operações e Serviços Candidatos, foi realizado a Análise de *Gap* e Análise de Realização dos mesmos, tendo como objetivo, identificar a possibilidade de reuso de funcionalidades já implementadas

através dos Sistemas Legados da empresa. Em seguida, deu-se início a etapa de Projeto de Serviços, iniciando pela definição do Esquema da Classe de Mensagem e arquivo do Esquema XSD.

Na sequência, iniciou-se a etapa de definição do arquivo WSDL, para cada uma das Interfaces dos Seviços identificados anteriormente. No caso, foi realizado como exemplo, somente a especificação WSDL da Interface do Serviço IcontroleDeAcessoService, sendo que, para os demais, realizada apenas uma descrição suscinta sobre cada um destes Serviços.

Por fim, foram realizadas as etapas de Revisão dos Serviços, Orquestração dos Serviços e Implementação e Teste, para as quais somente foi informado os detalhes sobre tais etapas, como: quais ferramentos foram utilizadas e quais foram os profissionais que conduziram as mesmas.

5. ANÁLISE DOS RESULTADOS

Nesse capítulo será apresentada uma análise crítica sobre as etapas de Análise e Projeto da Aplicação SOA, voltadas a um Sistema de Controle de Acesso, destacando os resultados, benefícios e vantagens obtidas através da utilização do conceito de Serviços como sustentação da camada de negócio deste sistema.

5.1 Resultados Obtidos

Ao realizar a análise a partir da definição dos modelos de processo *as-is*, tendo como proposta de melhoria a definição dos modelos de processo *to-be*, fica notório os benefícios que podem ser obtidos através da implantação de um Sistema de Controle de Acesso Físico da entrada de colaboradores ou visitantes nos vários ambientes da empresa, para o aumento da segurança patrimonial e recintos estratégicos.

A partir dos modelos de processo *to-be* "Gerenciar Acesso Físico Funcionário" e "Gerenciar Acesso Físico Visitante", percebe-se que as vulnerabilidades existentes nos atuais processos são totalmente eliminadas, já que há um melhor controle operacional e computacional, permitindo maior controle nos acessos realizados pelos usuários (Funcionários e Visitantes), desde a entrada na Portaria, até o acesso às Áreas de Segurança dentro da empresa.

Além de potencializar a segurança de acesso às Áreas de Segurança, todos os acessos e imagens obtidas através dos Pontos de Acesso e Câmeras, são registrados através de Serviços, permitindo total rastreabilidade da movimentação de pessoas dentro e ao redor da empresa.

Os Serviços identificados através das Especificações das Tarefas, demonstra que a aplicação dos conceitos sobre utilização de Serviços como base para a construção de uma camada de negócio de um Sistema de Controle de Acesso Físico, mostra-se

muito aderente às necessidades operacionais exigidas para o funcionamento de um sistema informatizado dessa natureza.

Para que um Sistema de Controle de Acesso Físico funcione de maneira satisfatória, uma série de controles computacionais, tais como as tarefas de sistema, precisam ser executados e controlados em tempo real. Registros de acesso e imagens capturadas pelas câmeras, precisam ser armazenados a cada instante. Neste contexto, o emprego da Arquitetura Orientada a Serviços (SOA), supre essas necessidades, já que permite ser representado por aplicações desenvolvidas em diferentes plataformas, como Java e .Net.

A utilização do método MAPOS, através do ciclo de vida proposto pelo mesmo, demonstrou ser muito eficaz para a contrução deste projeto SOA, já que possui uma estrutura de fácil assimilação, que permite identificar, analisar e especificar os serviços de forma prática. Além disso, uma vantagem desse método é visar a qualidade dos serviços.

Através do estudo realizado para a construção desta aplicação SOA, pretende-se obter os seguintes benefícios:

- Tornar evidente a importância da utilização de sistemas computacionais para aumentar a segurança patrimonial e de colaboradores, como forma de incentivo às empresas que necessitem eliminar as vulnerabilidades de segurança de acesso físico, buscando como alternativa a implantação de um Sistema de Controle de Acesso Físico apoiado pela utilização de sistemas computacionais.

- Mostrar que o emprego da Arquitetura Orientada a Serviços, como solução para a construção de um Sistemas Computacional voltado ao Controle de Acesso Físico, traz inúmeras vantagens, sendo uma delas, a possibilidade de interoperabilidade entre aplicações desenvolvidas em plataformas e tecnologias diferentes, que é um dos requisítos exigidos por sistemas desta natureza; além disso, a possibilidade de se utilizar serviços prontos, fornecidos por provedores facilita a implementação do sistema e é uma forma eficiente de reuso.

- Fazer um trabalho que sirva como base para outros pesquisadores que desejem explorar as vantagens da utilização da Arquitetura Orientada a Serviços voltados à Segurança de Informação em Controle de Acesso Físico. Visando-se esse fim, procurou-se fazer na monografia uma exposição dos principais princípios envolvidos, desde a conceituação teórica até os aspectos tecnológicos atuais.

5.2 Considerações do Capítulo

Através dos processos *to-be* "Gerenciar Acesso Físico Funcionário" e "Gerenciar Acesso Físico Visitante', ficou evidente os benefícos tragos com a implantação de um Sistema de Controle de Acesso Físico como forma de pontencializar a segurança patrimonial e das pessoas dentro de uma empresa.

Concluímos que o emprego de Serviços como base para construção de um Sistema de Controle de Acessos é totalmente aderente às necessidades operacionais exigidas para o funcionamento de tal aplicação de software.

A utilização do método MAPOS demonstrou ser muito eficaz para a contrução deste projeto SOA, por possuir uma estrutura de fácil assimilação, que permite identificar, analisar e especificar os serviços de forma prática.

6. CONSIDERAÇÕES FINAIS

Este capítulo contém as considerançöes finais e contribuições do trabalho em questão, bem como as sugestões de trabalhos futuros que poderão ser desenvolvidos a partir deste trabalho.

6.1 Contribuições do Trabalho

Neste trabalho foi realizado um estudo, através do qual, se procurou mostrar os principais conceitos relacionados a SOA e Sistemas de Controle de Acesso, com especial atenção ao Controle de Acesso Físico.

Foi mostrada uma série de conceitos correlatos à SOA, como a tecnologia de *Web Services*, algumas metodologias utilizadas para a Análise e Projeto Orientado a Serviços, como o método proposto por Erl (2005), e o método MAPOS, proposto por Fugita (2009).

Ao final, foi proposto através de um Estudo de Caso, uma solução de Análise e Projeto Orientado a Serviços, voltado à construção de um Sistema de Controle de Acesso Físico, utilizando como metodologia o MAPOS, Com isto, se pretendeu mostrar as vantagens e importância na utilização da utilização de um controle computacional apoiado por SOA para o Controle de Acesso.

6.2 Trabalhos Futuros

No Estudo de Caso apresentado, foi proposto como formas de autenticação a utilização de Leitores Biométricos e Cartões de Identificação de Usuário, através dos quais, o usuário solicita acesso a uma Área de Segurança em um equipamento no Ponto de Acesso. Em trabalhos futuros, poderiam ser contempladas a utilização de formas de autenticação mais modernos, como reconhecimento de íris, voz ou traços faciais.

Outra possibilidade para evoluir esse sistema seria realizar a fase de Construção, Simulação e Testes.

Também, não se entrou em detalhes quanto a parte relativa ao tratamento das câmeras e captura de imagens, que é hoje considerada muito importante em qualquer sistema de controle de acesso.

Ao longo do estudo muito provedores de serviços foram identificados não sendo, entretanto, aqui relacionados. Porém, cada vez mais, os *Web Services* estão sendo publicados na Internet, podendo ser facilmente encontrados. Cabe, porém, realizar um estudo classificatório desses serviços.

REFERÊNCIAS

ALWADAIN, A. et al. **Integrating SOA into an Enterprise Architecture:** a comparative analysis of alternative approaches. In: 5th IFIP International Conference on Research and Practical Issues of Enterprise Information Systems, 2010. CONFENIS, 2010. 25-27 August 2010. Natal, Brazil. **Proceedings.**

BENANTAR, M. **Access Control Systems: Security, Identity Management and Trust Models.** 1st Ed. USA:

BHALLAMUDI, P.; TILLEY, S. **SOA Migration Case Studies and Lessons Learned.** In: Systems Conference (SysCon), 2011, IEEE International. Melbourne, Florida. **Proceedings.** Melbourne, Flórida: Department of Computer Sciences Florida Institute of Technology. 2011. p. 123-128.

BLUNDO, C. et al. **Validating Orchestration of Web Services with BPEL and Aggregate Signatures.** In: Sixth European Conference on Web Services, 2008. ECOWS 2008. Dublin, Ireland. 12-14 November 2008. **Proceedings.** Dublin, Ireland. 2008.

CREDLE, R. **Patterns: SOA Design Using WebSphere Message Broker and WebSphere ESB.** USA: IBM – International Business Machines Corporation, 2007. p. 10-12. Springer Science+Business Media, Inc., 2006. 261 p.

ENDREI, M. et al. **Patterns: Service-Oriented Architecture and Web Services**. International Business Machines Corporation, 2004. p. 31-33.

ERL, T. **Service-Oriented Architecture:** Concepts, Technology, and Design. 1st Ed. USA: Pearson Education, Inc., 2005. 760 p.

FERRARI, E. Access Control in Data Management Systems. Waterloo, Ca: Morgan & Claypool Publishers. 2010.

FERREIRA, C. L. P. **Maestro: Um Middleware para Suporte a Aplicações Distribuídas Baseadas em Componentes de Software**. 2001. 100 p. Dissertação (Mestrado). Escola Politécnica. Universidade de São Paulo. 2009.

FUGITA, H. S.; HIRAMA, K. **SOA: modelagem, análise e design**. Rio de Janeiro: Elsevier, 2012.

FUGITA, S. F. **MAPOS: Método de Análise e Projeto Orientado a Serviços**. 2009. 175 p. Dissertação (Mestrado). Escola Politécnica. Universidade de São Paulo. 2009.

HALILI et al. **Styles of Service Composition – Analysis and Comparison Methods,** In: Fifth International Conference on Computational Intelligence, Communication Systems and Networks, 2013. p. 302-307.

KERRIGAN, M. et al. **The Web Service Modeling Toolkit - An Integrated Development Environment for Semantic Web Services (System Description).** In: Fourth European Semantic Web Conference, 2007. ESWC 2007. Innsbruck, Austria. 3-7 June 2007. **Proceedings.** Innsbruck, Austria. 2007.

KIM, D. SOLOMON, M. G. **Fundamentos de Segurança de Sistemas de Informação.** 1 ed. BRA: LTC – Livros Técnicos e Científicos Editora Ltda, 2014. 386 p.

LEE et al. Hypertext Transfer Protocol – HTTP/1.0. Network Working Group, 1996. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc1945.txt>. Acesso em: 02/09/2015.

OASIS SOA REFERENCE MODEL TC. Service-Oriented Architecture Reference Model Version 1.0. OASIS Open, 2006. Disponível em: < http://www.pcs.usp.br/~pcs5002/oasis/soa-rm-csbr.pdf>. Acesso em: 10/06/2015.

OASIS WS-BPEL TC. Web Services Business Process Execution Language Version 2.0. OASIS Open, 2007. Disponível em: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>. Acesso em: 10/06/2015.

OASIS WS-BPEL TC. Web Services Business Process Execution Language Version 2.0 Primer. OASIS Open, 2007. Disponível em: < http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.html>. Acesso em: 14/06/2015.

PAPAZOGLOU, M. P. **Web Services:** principles and technology.1st. Ed. UK: Pearson Prentice Hall, 2008a. 752 p.

PAPAZOGLOU, M. P. ***Web Services*:** Principles and Technology. September, 23, 2007.

PELTZ, C. Web Services Orchestraion and Choreography. **IEEE Computer Society Magazine,** October 2013

SOMMERVILLE, I. **Software Engineering**. 9th. Ed. USA: Pearson Addison Wesley, 2011. 773 p.

SORATHIA, V.; LALIWALA, Z.; CHAUDHARY, S. Towards Agricultural Marketing: Web Services Orchestration Approach. In: Services Computing, 2005 IEEE International Conference, 2005. 11-15 July 2005. **Proceedings.** 2005. 260-257 v. 1.

SWEENEY, R. **Achieving Service-Oriented Architecture:** Applying an Enterprise Architecture Approach. 1st. ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2010. 11 p.

VERJUS, H.; POURRAZ, F. **A Formal Framework For Building, Checking And Evolving Service Oriented Architectures**. Fifth European Conference on Web Services. Halle (Saale), Germany. 26-28 November 2007. Proceedings. p. 245-254.

W3C – WORLD WIDE WEB CONSORTIUM. **Web Services Architecture**. 2004. Disponível em: <http://www.w3.org/TR/ws-arch>. Acesso em: 20 jan 2015.

W3C – WORLD WIDE WEB CONSORTIUM. **XML Schema Definition Language (XSD) 1.1 Part 1: Structures.** 2004. Disponível em: http://www.w3.org/TR/xmlschema11-1/#concepts>. Acesso em: 20 jan 2015.

W3C – WORLD WIDE WEB CONSORTIUM. **Simple Object Access Protocol (SOAP) 1.1.** 2000. Disponível em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>. Acesso em: 20 jan 2015.

W3C – WORLD WIDE WEB CONSORTIUM. **Web Services Description Language (WSDL) 1.1.** 2001. Disponível em: <http://www.w3.org/TR/wsdl>. Acesso em: 20 jan. 2015.

YANAI, L. P. R. **MISLP: Método de Identificação de Serviços Baseado em Linguagem de Padrões**. 2010. 124 p. Dissertação (Mestrado). Escola Politécnica. Universidade de São Paulo. 2010.

APÊNDICE A - Especificação dos Esquemas XSD das Classes de Mensagem

Classe de Mensagem "Funcionario"

Classe que contém os principais dados do Funcionário, sendo utilizada pelo Serviço FuncionarioService. Na figura 49, temos o diagrama e arquivo XSD da mesma.

Figura 49: Classe de Mensagem Funcionario e a definição do arquivo XSD.


```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Funcionario" nillable="true" type="Funcionario" />
  <xs:complexType name="Funcionario">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="codigoExterno" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="Nome" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="Cargo" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="Setor" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="xs:dateTime" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "Visitante"

Classe que contém os principais dados do Visitante, sendo utilizada pelo Serviço AgendamentoVisitasService. Na figura 50, temos o diagrama e arquivo XSD da mesma.

Figura 50: Classe de Mensagem Visitante e a definição do arquivo XSD.


```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Visitante" nillable="true" type="Visitante" />
  <xs:complexType name="Visitante">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="nome" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="telefone" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="email" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="nomeEmpresa" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="xs:dateTime" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "Usuario"

Classe que contém os principais dados do Usuario, sendo utilizada pelos Serviços UsuarioService, CartaoDeIdentificacaoService e ControleDeAcessoService. Na figura 51, temos o diagrama e arquivo XSD da mesma.

Figura 51: Classe de Mensagem Usuario e a definição do arquivo XSD.

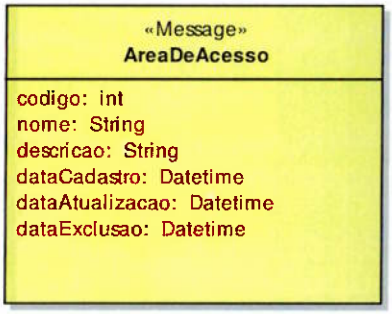
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Usuario" nillable="true" type="Usuario" />
  <xs:complexType name="Usuario">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="nome" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="codigoFuncionario" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="codigoVisitante" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "AreaDeAcesso"

Classe que contém os principais dados sobre Área de Acesso, sendo utilizada pelos Serviços AreaDeAcessoService e ControleDeAcessoService. Na figura 52, temos o diagrama e arquivo XSD da mesma.

Figura 52: Classe de Mensagem AreaDeAcesso e a definição do arquivo XSD.



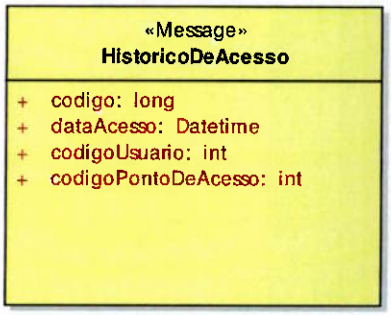

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="AreaDeAcesso" nillable="true" type="AreaDeAcesso" />
  <xs:complexType name="AreaDeAcesso">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="nome" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="descricao" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="xs:dateTime" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "HistoricoDeAcesso"

Classe que contém os principais dados sobre o Histórico de Acesso, sendo utilizada pelo Serviço ControleDeAcessoService. Na figura 53, temos o diagrama e arquivo XSD da mesma.

Figura 53: Classe de Mensagem HistoricoDeAcesso e a definição do arquivo XSD.



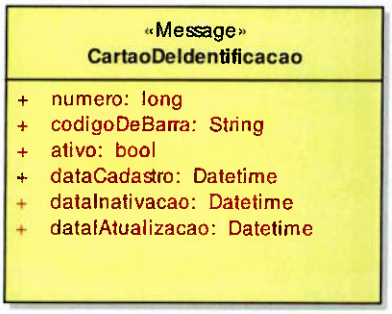

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="HistoricoDeAcesso" nillable="true" type="HistoricoDeAcesso" />
  <xs:complexType name="HistoricoDeAcesso">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:long" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAcesso" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="codigoUsuario" type="xs:int" />
      <xs:element minOccurs="1" maxOccurs="1" name="codigoPontoDeAcesso" type="xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "CartaoDeIdentificacao"

Classe que contém os principais dados sobre o Cartão de Identificacao, sendo utilizada pelo Serviço ControleDeAcessoService. Na figura 54, temos o diagrama e arquivo XSD da mesma.

Figura 54: Classe de Mensagem CartaoDeIdentificacao e a definição do arquivo XSD.



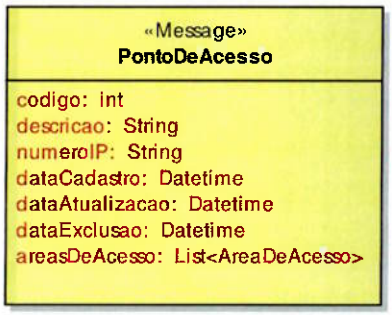
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="CartaoDeIdentificacao" nillable="true" type="CartaoDeIdentificacao" />
  <xs:complexType name="CartaoDeIdentificacao">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="numero" type="xs:long" />
      <xs:element minOccurs="0" maxOccurs="1" name="codigoDeBarra" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="ativo" type="xs:boolean" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataInativacao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataIAtualizacao" type="xs:dateTime" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "PontoDeAcesso"

Classe que contém os principais dados sobre o Ponto de Acesso, sendo utilizada pelos Serviços PontoDeAcessoService e ControleDeAcessoService. Na figura 55, temos o diagrama e arquivo XSD da mesma.

Figura 55: Classe de Mensagem PontoDeAcesso e a definição do arquivo XSD.



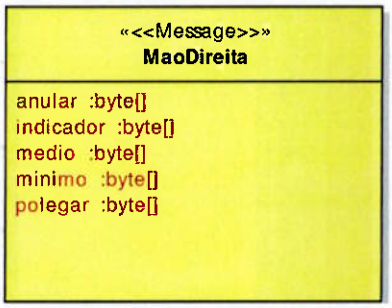

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PontoDeAcesso" nillable="true" type="PontoDeAcesso" />
  <xs:complexType name="PontoDeAcesso">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="codigo" type="xs:int" />
      <xs:element minOccurs="0" maxOccurs="1" name="descricao" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="1" name="numeroIP" type="xs:string" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="xs:dateTime" />
      <xs:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="xs:dateTime" />
      <xs:element minOccurs="0" maxOccurs="1" name="areasDeAcesso" type="ArrayOfAreaDeAcesso" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "MaoDireita"

Classe que contém os dados biométricos da Mão Direita, sendo utilizada pelo Serviço ControleDeAcessoService. Na figura 56, temos o diagrama e arquivo XSD da mesma.

Figura 56: Classe de Mensagem MaoDireita e a definição do arquivo XSD.



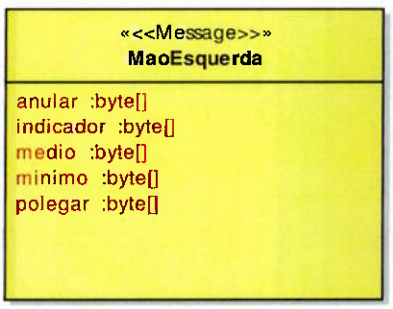

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MaoDireita" nillable="true" type="MaoDireita" />
  <xs:complexType name="MaoDireita">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="polegar" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="indicador" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="medio" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="anular" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="minimo" type="xs:base64Binary" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

Classe de Mensagem "MaoEsquerda"

Classe que contém os dados biométricos sobre da Mão Esquerda, sendo utilizada pelos Serviço ControleDeAcessoService. Na figura 57, temos o diagrama e arquivo XSD da mesma.

Figura 57: Classe de Mensagem MaoEsquerda e a definição do arquivo XSD.



```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MaoEsquerda" nillable="true" type="MaoEsquerda" />
  <xs:complexType name="MaoEsquerda">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1" name="polegar" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="indicador" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="medio" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="anular" type="xs:base64Binary" />
      <xs:element minOccurs="0" maxOccurs="1" name="minimo" type="xs:base64Binary" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Fonte: Do Autor.

APÊNDICE B - Especificação das Interfaces dos Serviços em WSDL

Serviço "IControleDeAcessoService"

Esse Serviço contém as operações necessárias para autênticação do Funcionário e Visitante no Ponto de Acesso. O Arquiteto SOA realizou a definição do arquivo WSDL referente a Interface deste Serviço. Na figura 58, segue o diagrama e arquivo WSDL do mesmo.

Figura 58: Serviço ControleDeAcessoService e a definição do arquivo WSDL.

Fonte: Do Autor.

Segue a definição da especificação da Interface deste Serviço em WSDL.

```
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://www.exemplo.com.br/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://www.exemplo.com.br/">
    <wsdl:types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://www.exemplo.com.br/">
            <s:element name="ConcederPermissaoAcesso">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1" name="areasDeAcesso" type="tns:ArrayOfAreaDeAcesso" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ArrayOfAreaDeAcesso">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded" name="AreaDeAcesso" nillable="true" type="tns:AreaDeAcesso" />
                </s:sequence>
            </s:complexType>
            <s:complexType name="AreaDeAcesso">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="codigo" type="s:int" />
                    <s:element minOccurs="0" maxOccurs="1" name="nome" type="s:string" />
                    <s:element minOccurs="0" maxOccurs="1" name="descricao" type="s:string" />
                    <s:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="s:dateTime" />
                    <s:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="s:dateTime" />
                    <s:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="s:dateTime" />
                </s:sequence>
            </s:complexType>
            <s:element name="ConcederPermissaoAcessoResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="ConcederPermissaoAcessoResult" type="s:int" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="GravarHistoricoDeAcesso">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1" name="historicoAcesso" type="tns:ArrayOfHistoricoDeAcesso" />
```

```
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:complexType name="ArrayOfHistoricoDeAcesso">
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="unbounded" name="HistoricoDeAcesso" nillable="true" type="tns:HistoricoDeAcesso" />
        </s:sequence>
    </s:complexType>
    <s:complexType name="HistoricoDeAcesso">
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="codigo" type="s:long" />
            <s:element minOccurs="1" maxOccurs="1" name="dataAcesso" type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="codigoUsuario" type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="codigoPontoDeAcesso" type="s:int" />
        </s:sequence>
    </s:complexType>
    <s:element name="GravarHistoricoDeAcessoResponse">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="1" maxOccurs="1" name="GravarHistoricoDeAcessoResult" type="s:int" />
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:element name="ListarHistoricoAcesso">
        <s:complexType />
    </s:element>
    <s:element name="ListarHistoricoAcessoResponse">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" name="ListarHistoricoAcessoResult" type="tns:ArrayOfHistoricoDeAcesso" />
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:element name="RealizarLeituraBiometrica">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" name="maoDireita" type="tns:MaoDireita" />
                <s:element minOccurs="0" maxOccurs="1" name="maoEsquerda" type="tns:MaoEsquerda" />
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:complexType name="MaoDireita">
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="polegar" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="indicador" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="medio" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="anular" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="minimo" type="s:base64Binary" />
        </s:sequence>
    </s:complexType>
    <s:complexType name="MaoEsquerda">
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="polegar" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="indicador" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="medio" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="anular" type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="minimo" type="s:base64Binary" />
        </s:sequence>
    </s:complexType>
    <s:element name="RealizarLeituraBiometricaResponse">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" name="RealizarLeituraBiometricaResult" type="tns:Usuario" />
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:complexType name="Usuario">
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="codigo" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="nome" type="s:string" />
            <s:element minOccurs="1" maxOccurs="1" name="dataCadastro" type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="dataAtualizacao" type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="dataExclusao" type="s:dateTime" />
            <s:element minOccurs="1" maxOccurs="1" name="codigoFuncionario" type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="codigoVisitante" type="s:int" />
        </s:sequence>
    </s:complexType>
    <s:element name="SolicitarPermissaoAcesso">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="1" maxOccurs="1" name="codigoPontoAcesso" type="s:int" />
                <s:element minOccurs="0" maxOccurs="1" name="usuario" type="tns:Usuario" />
```

```
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="SolicitarPermissaoAcessoResponse">
            <s:complexType>
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="SolicitarPermissaoAcessoResult" type="s:boolean" />
                </s:sequence>
            </s:complexType>
        </s:element>
    </s:schema>
</wsdl:types>
<wsdl:message name="ConcederPermissaoAcessoSoapIn">
    <wsdl:part name="parameters" element="tns:ConcederPermissaoAcesso" />
</wsdl:message>
<wsdl:message name="ConcederPermissaoAcessoSoapOut">
    <wsdl:part name="parameters" element="tns:ConcederPermissaoAcessoResponse" />
</wsdl:message>
<wsdl:message name="GravarHistoricoDeAcessoSoapIn">
    <wsdl:part name="parameters" element="tns:GravarHistoricoDeAcesso" />
</wsdl:message>
<wsdl:message name="GravarHistoricoDeAcessoSoapOut">
    <wsdl:part name="parameters" element="tns:GravarHistoricoDeAcessoResponse" />
</wsdl:message>
<wsdl:message name="ListarHistoricoAcessoSoapIn">
    <wsdl:part name="parameters" element="tns:ListarHistoricoAcesso" />
</wsdl:message>
<wsdl:message name="ListarHistoricoAcessoSoapOut">
    <wsdl:part name="parameters" element="tns:ListarHistoricoAcessoResponse" />
</wsdl:message>
<wsdl:message name="RealizarLeituraBiometricaSoapIn">
    <wsdl:part name="parameters" element="tns:RealizarLeituraBiometrica" />
</wsdl:message>
<wsdl:message name="RealizarLeituraBiometricaSoapOut">
    <wsdl:part name="parameters" element="tns:RealizarLeituraBiometricaResponse" />
</wsdl:message>
<wsdl:message name="SolicitarPermissaoAcessoSoapIn">
    <wsdl:part name="parameters" element="tns:SolicitarPermissaoAcesso" />
</wsdl:message>
<wsdl:message name="SolicitarPermissaoAcessoSoapOut">
    <wsdl:part name="parameters" element="tns:SolicitarPermissaoAcessoResponse" />
</wsdl:message>
<wsdl:portType name="ControleDeAcessoServiceSoap">
    <wsdl:operation name="ConcederPermissaoAcesso">
        <wsdl:input message="tns:ConcederPermissaoAcessoSoapIn" />
        <wsdl:output message="tns:ConcederPermissaoAcessoSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="GravarHistoricoDeAcesso">
        <wsdl:input message="tns:GravarHistoricoDeAcessoSoapIn" />
        <wsdl:output message="tns:GravarHistoricoDeAcessoSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="ListarHistoricoAcesso">
        <wsdl:input message="tns:ListarHistoricoAcessoSoapIn" />
        <wsdl:output message="tns:ListarHistoricoAcessoSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="RealizarLeituraBiomatrica">
        <wsdl:input message="tns:RealizarLeituraBiometricaSoapIn" />
        <wsdl:output message="tns:RealizarLeituraBiometricaSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="SolicitarPermissaoAcesso">
        <wsdl:input message="tns:SolicitarPermissaoAcessoSoapIn" />
        <wsdl:output message="tns:SolicitarPermissaoAcessoSoapOut" />
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ControleDeAcessoServiceSoap" type="tns:ControleDeAcessoServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="ConcederPermissaoAcesso">
        <soap:operation soapAction="http://www.exemplo.com.br/ConcederPermissaoAcesso" style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GravarHistoricoDeAcesso">
        <soap:operation soapAction="http://www.exemplo.com.br/GravarHistoricoDeAcesso" style="document" />
        <wsdl:input>
            <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal" />
```

```
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ListarHistoricoAcesso">
            <soap:operation soapAction="http://www.exemplo.com.br/ListarHistoricoAcesso" style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="RealizarLeituraBiometrica">
            <soap:operation soapAction="http://www.exemplo.com.br/RealizarLeituraBiometrica" style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="SolicitarPermissaoAcesso">
            <soap:operation soapAction="http://www.exemplo.com.br/SolicitarPermissaoAcesso" style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="ControleDeAcessoServiceSoap12" type="tns:ControleDeAcessoServiceSoap">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="ConcederPermissaoAcesso">
            <soap12:operation soapAction="http://www.exemplo.com.br/ConcederPermissaoAcesso" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="GravarHistoricoDeAcesso">
            <soap12:operation soapAction="http://www.exemplo.com.br/GravarHistoricoDeAcesso" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="ListarHistoricoAcesso">
            <soap12:operation soapAction="http://www.exemplo.com.br/ListarHistoricoAcesso" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="RealizarLeituraBiometrica">
            <soap12:operation soapAction="http://www.exemplo.com.br/RealizarLeituraBiometrica" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="SolicitarPermissaoAcesso">
            <soap12:operation soapAction="http://www.exemplo.com.br/SolicitarPermissaoAcesso" style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ControleDeAcessoService">
        <wsdl:port name="ControleDeAcessoServiceSoap" binding="tns:ControleDeAcessoServiceSoap">
            <soap:address location="http://localhost:22886/ControleDeAcessoService.asmx" />
        </wsdl:port>
```

```
        <wsdl:port name="ControleDeAcessoServiceSoap12" binding="tns:ControleDeAcessoServiceSoap12">
            <soap12:address location="http://localhost:22886/ControleDeAcessoService.asmx" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Para os demais Serviços, segue se apenas a descrição de cada um deles, sendo que, a definição da interface WSDL dos mesmos, ficaram armazenadas no software *Enterprise Architect*.

Serviço "UsuarioService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Usuários no sistema.

Serviço "FuncionarioService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Funcionários no sistema.

Serviço "CartaoDeIdentificacaoService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Cartões de Identificação no sistema.

Serviço "AgendamentoVisitasService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Agendamentos de Visita no sistema.

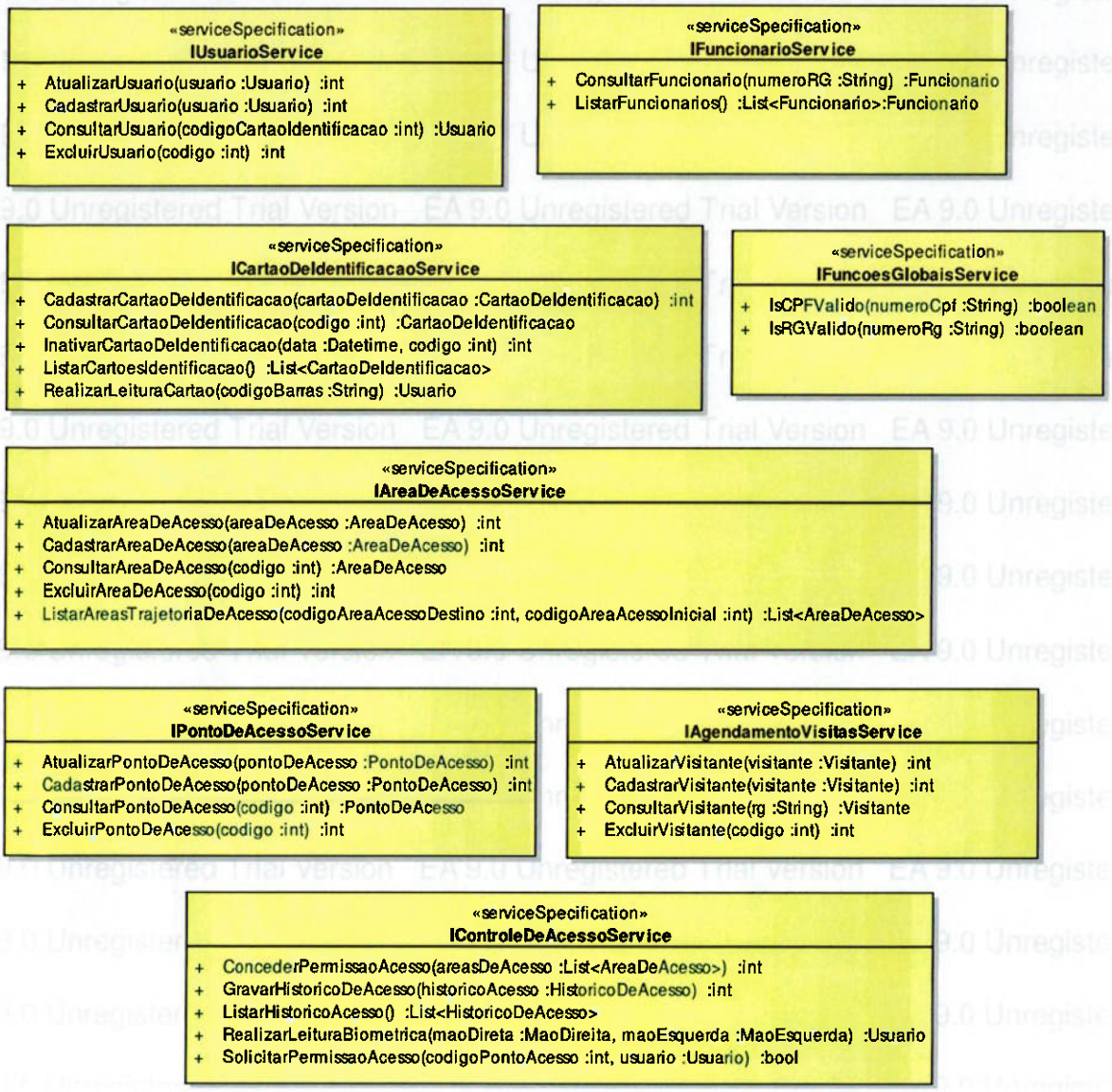
Serviço "PontoDeAcessoService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Pontos de Acesso no sistema.

Serviço "AreaDeAcessoService": Serviço que contém as operações necessárias para o cadastro, atualização, exclusão e consulta de Áreas de Acesso no sistema.

Serviço "FuncoesGlobaisService": Serviço que contém as operações globais que deverão ser compartilhadas com outros sistemas.

Na figura 59, segue o diagrama de projeto da Interface destes Serviços:

Figura 59: Diagramas de Projeto dos Serviços do Sistema de Controle de Acesso.



Fonte: Do Autor.