UNIVERSIDADE SALVADOR - UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO PROFISSIONAL EM SISTEMAS E COMPUTAÇÃO

RAIMUNDO ARAUJO DE ALMEIDA JUNIOR

INTEGRAÇÃO DE SISTEMAS EM EMPRESAS CORPORATIVAS DO SETOR PÚBLICO: UM ESTUDO DE CASO NA SECRETARIA DE EDUCAÇÃO DO ESTADO DA BAHIA

Salvador
2009

RAIMUNDO ARAUJO DE ALMEIDA JUNIOR

INTEGRAÇÃO DE SISTEMAS EM EMPRESAS CORPORATIVAS DO SETOR PÚBLICO: UM ESTUDO DE CASO NA SECRETARIA DE EDUCAÇÃO DO ESTADO DA BAHIA

Dissertação apresentada ao Curso de Mestrado Profissional em Sistemas e Computação, Universidade Salvador - UNIFACS, como requisito parcial para obtenção do grau de Mestre.

Orientadora: Profª Drª Laís do Nascimento Salvador.

Salvador
2009

FICHA CATALOGRÁFICA
(Elaborada pelo Sistema de Bibliotecas da Universidade Salvador - UNIFACS)

Almeida Junior, Raimundo Araujo de

Integração de sistemas em empresas corporativas do setor público: um estudo de caso na Secretaria de Educação do Estado da Bahia/ Raimundo Araujo de Almeida Junior. - 2009.

98 f. : il.

Dissertação (Mestrado) - Universidade Salvador – UNIFACS.Mestrado em Sistemas de Computação, 2007.
Orientadora: Profª. Dr. Laís do Nascimento Salvador.

1. Arquitetura de redes de computadores. I. Salvador, Laís do Nascimento, orient. II. Título.

CDD: 004.65

INTEGRAÇÃO DE SISTEMAS EM EMPRESAS CORPORATIVAS DO SETOR PÚBLICO: UM ESTUDO DE CASO NA SECRETARIA DE EDUCAÇÃO DO ESTADO DA BAHIA

Dissertação de Mestrado apresentada ao programa de Pós-graduação da Universidade Salvador - UNIFACS como requisito parcial para a obtenção do título de Mestre em Sistemas e Computação.

Laís do Nascimento Salvador – Orientadora _____

Doutorado em Engenharia Elétrica, pela Poli/USP

Universidade Salvador - UNIFACS

José Maria Nazar David –_____

Doutorado em Engenharia de Sistemas e Computação pela UFRJ

Universidade Salvador - UNIFACS

Vaninha Vieira dos Santos – _____

Doutorado em Ciências da Computação pela UFPE

Universidade Federal da Bahia - UFBA

30 de setembro de 2009.

Por mais que na batalha se vença a um ou mais inimigos, a vitória sobre a si mesmo é a maior de todas as vitórias.

Sakyamuni

AGRADECIMENTOS

Em primeiro lugar, eu agradeço a Deus por fazer parte da minha vida e viabilizar a conclusão deste trabalho. Em segundo lugar, agradeço aos meus pais que sempre deram uma enorme força durante a caminhada árdua para a concepção desde estudo. E terceiro, a todos os meus familiares e amigos que colaboraram diretamente ou indiretamente.

Aos colegas de trabalho que colaboram muito para a escolha dos serviços a serem expostos com base da arquitetura proposta.

Agradeço muito a minha orientadora, Profª Drª Laís do Nascimento Salvador, que teve muita paciência nas idas e vindas durante as correções do trabalho. Tenho que agradecer também, todos os professores que durante as aulas, tiveram sua parte de colaboração. Vale também ressaltar a importante colaboração do coordenador do curso, Prof. Dr. José Augusto Suruagy Monteiro, e os demais funcionários da UNIFACS.

RESUMO

Ao longo do tempo, as organizações públicas vêm gradativamente fazendo uso da Tecnologia da Informação (TI), informatizando os serviços prestados à comunidade. Atualmente, a TI corresponde a um dos principais pilares das organizações públicas. Com o intuito de melhor prover, controlar e flexibilizar a informação, a integração entre os diversos sistemas e/ou arquiteturas diferentes é uma das grandes prioridades organizacionais. Além disso, para as instituições públicas, ou até mesmo privadas, a integração entre Sistemas de Informação (SI), vem tornando-se um dos grandes desafios em virtude dos legados que se encontram em produção utilizando tecnologias e/ou arquiteturas que permitem baixíssima taxa de reuso. A organização que foi utilizada como base para esse estudo, Secretaria da Educação do Estado da Bahia (SEC) vem, com o passar dos anos, informatizando seus serviços. Porém, esse processo foi gradativo e, inicialmente, descentralizado, resultando em um ambiente computacional corporativo muito heterogêneo. É muito comum encontrar em instituições como a utilizada neste trabalho o uso conjunto de diversos sistemas com problemas de integração e de integridade das informações. Na atual situação da SEC, a atualização e inclusão de novos sistemas e tecnologias é uma constante, tornando vital a integração dos novos sistemas com legado existente, seja, compartilhando dados, processos ou funcionalidades/requisitos. Por isso, esse trabalho vem propor uma solução/padrão para o desenvolvimento de novos sistemas, viabilizando a integração com os legados, aplicando a componentização dos requisitos, através da Arquitetura Orientada a Serviços (SOA - Service Oriented Architecture).

Palavras-chave: Arquitetura Legada. Integração. Arquitetura Orientada a Serviços. Web Service.

ABSTRACT

Over time, public organizations are increasingly making use of Information Technology (IT), computerizing services to the community. Today, TI is one of the main pillars of public organizations. As the aim of providing better, more flexible and control the information, the integration between different systems and / or different architectures is a major organizational priorities. In addition, for public institutions, or even private, the integration of Information Systems (IS), has become a major challenge because of the legacy that is in production technologies and / or architectures that allow low rate for reuse. The organization was used as the basis for this study, the Education Department of the State of Bahia (SEC) has over the years its computer services. However, this process was gradual and, initially, decentralized, resulting in a very heterogeneous enterprise computing environment. It is very common to find in institutions such as used in this study the combined use of various systems with problems of integration and integrity of information. In the current situation of the SEC, the update and inclusion of new systems and technologies is a constant, making it vital to integrate the new systems with existing legacy, either, sharing data, processes or functions / requirements. Therefore, this work proposes a solution / standard for the development of new systems, allowing integration with legacy componentization of applying the requirements, through a Services Oriented Architecture (SOA - Service Oriented Architecture).

Keywords: Legacy Architecture. Integration. Service Oriented Architecture. Web Service.

LISTA DE FIGURAS

SUMÁRIO

CAPÍTULO 1

1 INTRODUÇÃO

As tecnologias voltadas para software e hardware são alvos de um forte processo evolutivo, proporcionando ao mercado diferentes maneiras e opções de processar informações e arquitetar novos modelos de negócios. De forma geral, esse mercado é o grande beneficiário com esse processo evolutivo. Ele também é o responsável por impulsionar a indústria a desenvolver novas tecnologias. A organização que foi estudada para construir este trabalho vivenciou este mesmo processo evolutivo, no qual será detalhado na próxima seção.

1.1 HISTÓRICO E PROBLEMA

Com o passar do tempo, a Secretaria da Educação do Estado da Bahia (SEC) foi percebendo o importante papel da informatização de seus serviços. No decorrer deste período a revolução tecnológica mudou muito os paradigmas de gestão empresarial. A aceleração, inovação e automação dos processos, associada com a velocidade das informações são fatores que estão impulsionando essa revolução. Com isso, surgiram diversas arquiteturas e tecnologias dentro da instituição estudada neste trabalho.

Na década de 80, surgiram os sistemas utilizando *mainframe*, computadores de grande porte, dedicado normalmente ao processamento de um volume grande de informações. Eles têm a capacidade de oferecer serviços de processamento a muitos usuários através de milhares de terminais através de uma rede ou conectados diretamente (FONTANETTE, 2005). Nesse período a organização apenas fazia uso desse tipo de aplicação.

Em seguida, veio a arquitetura *Cliente-Servidor Desktop*, onde nela fica bem clara a divisão em duas camadas, ou seja, duas aplicações se comunicando através de uma rede. Uma camada é o servidor na qual são processadas as requisições feitas pela outra camada, que são os clientes. Geralmente, eles possuem algumas regras de negócio (BECKER, 2002, p. 8). As aplicações clientes e os servidores são

executáveis que rodam na memória. Nessa arquitetura foram utilizadas as linguagens *Visual Basic* e *Delphi* para o desenvolvimento dos sistemas, sem seguir qualquer padrão. Com isso, a diversidade de arquitetura dos legados foi surgindo e o problema de integração das informações começou a aparecer. A produção de *software* era feita por equipes distintas e sem ocorrer a devida comunicação entre elas.

Logo em seguida, surgiu o desenvolvimento voltado para WEB, onde seu funcionamento é semelhante à arquitetura *Cliente-Servior Desktop*, porém, a sua diferenciação diz respeito ao fato de que não existe uma aplicação na memória do cliente e sim um interpretador das requisições respondidas pelo servidor, através do protocolo *hypertext transmission protocol* (HTTP) (CUMMINS, 2002, p. 156). No desenvolvimento das aplicações voltadas para essa arquitetura foram utilizados o *ASP* 2.0 nativo e a combinação deste com *Microsoft Transacion Server (MTS)*. Com isso, podemos perceber o inicio no desenvolvimento baseado em componentes, porém, sem padronização e com muito pouco reuso. Dentro dessa diversidade, surgiu o embrião da padronização no desenvolvimento de sistemas que se tornará o futuro Processo de Desenvolvimento de Sistemas (PDS), que será detralhado mais adiante.

Recentemente, optamos por utilizar a Arquitetura Orientada por Modelos *Model Driven Architecture (MDA*). Segundo Maia (2006) e Gomes e outros autores (2006) *MDA* é uma proposta que enfatiza a transformação entre os modelos *Computacional Independent Model (CIM)*, ou Modelo Independente de Computação (MIC), *Platform Independent Model (PIM),* ou Modelo Independente de Plataforma (MIP) e *Platform Specific Model (PSM)*, ou Modelo para Plataforma Específica (MPE) representados através da Unified Modeling Language (UML). Essa experiência não foi muito bem sucedida. A equipe de desenvolvimento teve sérios problemas de produtividade durante a experiência com MDA, por isso, optamos por abortar a utilização dela sem a implantação de qualquer aplicação com o uso dessa arquitetura. Nesse período, já tinhamos as equipes de desenvolvimento centralizadas e com a devida comunicação entre elas.

Hoje, a SEC adotou uma arquitetura voltada para o desenvolvimento em várias camadas utilizando a tecnologia *Jboss Seam* (FARLEY, 2008) com a

definição e o planejamento adequado de cada componente baseado nas regras de negócio da organização. O *Jboss Seam* é “gestor” da integração entre os *frameworks* responsáveis pelas camadas de apresentação, negócio e persistência (FARLEY, 2008; TAIT, 2001; PACHECO, 2000). Com isso, os arquitetos de *software* da SEC ficam mais focados nas integrações das informações referentes às regras de negócio e não nos *frameworks* das camadas de integração, entre as camadas desta arquitetura. Todos os sistemas que são produzidos atualmente seguem essa aquitetura e um processo de desenvolvimento denominado *Processo de Desenvolvimento de Sistemas (PDS),* que foi concebido com base no Melhoria de Processo do Software Brasileiro (MPS-BR), Capability Maturity Model Integration (CMMI) e Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos (PMBOK) com as equipes trabalhando de forma centralizada.

Como podemos observar, no decorrer do tempo, a organização estudada passou por diversas arquiteturas de sofwares. Como isso, os softwares legados passaram a ser um grande problema no âmbito da integridade das informações. Muitas vezes, para prover uma informação concisa é necessário um dispendioso trabalho.

A convivência dos sistemas novos que estão surgindo com o parque dos legados é muitas vezes de custo muito alto, porque em muitos casos existem redundância das informações e uma grande dificuldade para integrar os dados existentes com os novos. Esse quadro acaba acarretando um maior tempo de desenvolvimento das novas aplicações, maior custo durante a execução do projeto, alto grau de divergência do planejado para o executado e um nivel elevado de manutenibilidade.

1.2 MOTIVAÇÃO

A SEC vem sofrendo com um forte processo evolutivo das tecnologias. Com o passar do tempo, Ela passou a conviver com muitas arquiteturas de software bem heterogeneas, fragilizando a informação fornecida pela instituição. Com passar do tempo essas arquiteturas legadas estão se tornando muito complexas e robustas dificultado a manutenção das mesmas.

As aplicações com compôe os legados atuais de software da SEC foram construidas de forma independente e isolada, ou seja, as aplicações eram idealizadas sem a preocupação do reuso da informação produzida, resultando em alguns casos em redundância não controlada. Porém, ao mesmo tempo e aspecto a organização necessita fornecer respostas ágeis e disponibilizar informações consistente para a sociedade, seu cliente. Por isso, há uma forte necessidade de integração entre as diversas aplicações legadas. A motivação para elaborar este trabalho surgiu para atender essa necessidade de integração entre as diversas arquiteturas heterogêneas.

1.3 OBJETIVO

A evolução dos processos e negócios das organizações, privadas ou públicas, requer uma evolução das suas aplicações legadas, mas nem sempre existe essa possibilidade. Isso é em virtude da baixa qualidade desses *softwares* conforme dito por Fontanette (2005, p. 70). A instituição estudada, Secretaria da Educação do Estado da Bahia (SEC), sofre desse mal.

Para oferecer melhor qualidade nas informações educacionais para a comunidade do Estado da Bahia, através dos sistemas desenvolvidos pela instituição estudada, que nasceu o fator motivacional com o intuito de propor uma solução que melhor atenda a esse tipo de problema. Portanto, este trabalho tem como objetivo propor uma arquitetura de *software* para melhorar o quadro atual de integração das informações entre as aplicações novas que serão construídas e os legados através da Arquitetura Orientada a Serviços, bem como, propiciar a troca de informações entre as organizações, utilizando-se da mesma arquitetura proposta por este trabalho.

1.4 ORGANIZAÇÃO

O presente trabalho possui cinco capitulos. O segundo mostra a Arquitetura Orientada a Serviços (SOA). No terceiro é apresentada a arquitetura proposta para melhorar a integração entre os sistemas. No quarto são apresentados os *cases* com o uso da arquitetura proposta por este trabalho. E para fechar este estudo as considerções finais e os trabalhos futuros.

CAPÍTULO 2

2 ARQUITETURA ORIENTADA A SERVIÇOS

Os sistemas de informação, com o passar do tempo, vêm crescendo exponencialmente, as organizações vêm projetando arquiteturas gradativamente muito mais robustas e complexas, onde as respostas a serem dadas pela instituição devem ser ágeis em relação à evolução dos negócios da mesma, procurando reduzir gradativamente seus custos e ao mesmo tempo integrar nos processos de negócio que irão surgir.

Segundo Santos Junior (2007, p. 55) e Larentis (2007, p. 59) o surgimento da Arquitetura Orientada a Serviço (SOA) é dada pelo processo evolutivo da tecnologia de software e hardware. A alta capacidade e o grande poder de processamento dos dados, aliados à velocidade com que as informações são intercambiadas, o alto grau de padronização dos protocolos, a abrangente infra-estrutura disponibilizada com o advento, principalmente, da internet que propiciou um momento para a difusão dos conceitos e práticas da arquitetura SOA.

A SOA é um novo paradigma de desenvolvimento de aplicações cujo objetivo é criar módulos funcionais chamados de serviços, com baixo acoplamento e permitindo a reutilização de código. Essa arquitetura vem se tornando muito difundida, em virtude das possibilidades de arquitetar e desenvolver novas aplicações e processo de negócios, reaproveitando principalmente códigos existentes e plataformas heterogêneas, como também estratégias proporcionadas pela arquitetura.

Segundo Santos Junior (2007, p. 62) e Larentis (2007, p. 75) a SOA tem como seu foco prover e consumir serviços, principalmente expor serviços para aplicações web.

No decorer deste capítulo serão abordados os principais conceitos que a SOA utiliza para fazer uso de sua arquitetura. Acoplamento fraco, Web Service, Webservice Description Language (WSDL), Universal Discovery Description and Integration (UDDI), Serviços, Barramento de Serviços Corporativos e Simple Object Acess Protocol (SOA) serão detalhados mais a frente.

2.1 ACOPLAMENTO FRACO

O acoplamento fraco ou *Loose coupling* tem como objetivo a redução da interdependência e possíveis riscos de conflitos entre as entidades. Existe uma crescente tendência de projetar interfaces provedoras de serviços com esse tipo de acoplamento. A razão de projetar interfaces com o mínimo de acoplamento possível (*Loose coupling*) é viabilizar maior flexibilidade, podendo adicionar e modificar as interfaces com um mínimo ou nenhum impacto na interoperabilidade entre as interfaces já existentes, como foi dito por (COSTA; CARVALHO, 2007).

Com o advento da Internet, que é caracterizada por ser uma rede muito grande de computadores que são conectados uns aos outros, através de protocolos de comunicação e transmissão de dados comuns. Através da Internet surgiram condições para o surgimento de padrões verdadeiramente abertos para interoperabilidade de computadores em ambientes distribuídos. Com isso, fortalecendo o uso do conceito de acoplamento fraco nas aplicações da internet através de padrões abertos.

Segundo Sampaio (2006, p. 42) o grau de acoplamento entre dois módulos ou aplicações também determina o grau de dificuldade de manutenibilidade dos mesmos. A Figura 1 ilustra a diferença entre um acoplamento forte e um franco. Caso o acoplamento seja alto, mudanças em um dos módulos, onde existe um acoplamento entre eles, pode afetar o outro também. Isto indica o grau de manutenção do sistema no qual os módulos são envolvidos. Estes graus de acoplamento são classificados da seguinte forma: *Dados* – ocorre quando dois módulos se comunicam apenas por meio dos parâmetros trocados. *Imagem* – quando dois ou mais módulos compartilham uma determinada área comum. *Controle* – é péssimo e ocorre quando um módulo controla a lógica do outro. *Comum* – semelhante ao de imagem, porém todos compartilham a mesma área de dados. *Conteúdo* – não muito adequado, porque está ligado ao uso de desvios incondicionais.

Com o passar do tempo, várias tentativas foram feitas para resolver os problemas da computação distribuída e orientada a objetos, mas falharam devido à falta de suporte entre os grandes fornecedores da indústria de TI e a preponderância

de padrões proprietários que permeavam esses esforços, com isso, facilitando o aumento de acoplamento entre as diversas aplicações dentro e fora das organizações. A rede aberta da Internet e a necessidade de propiciar a redução dos acoplamentos entre os módulos das aplicações corporativas propiciou o surgimento dos *Web Services*, que são componentes de *software* capazes de se comunicar uns com os outros sobre diversas redes utilizando “padrões abertos” aceitos universalmente e não proprietário. A concordância dos grandes *players* da indústria de TI em adotar os padrões de *web services* (será mais detalhado no item a seguir) emergentes criou condições para a revolução na computação, conforme dito por Pulier; Taylor; Gaffney (2008, p. 23).

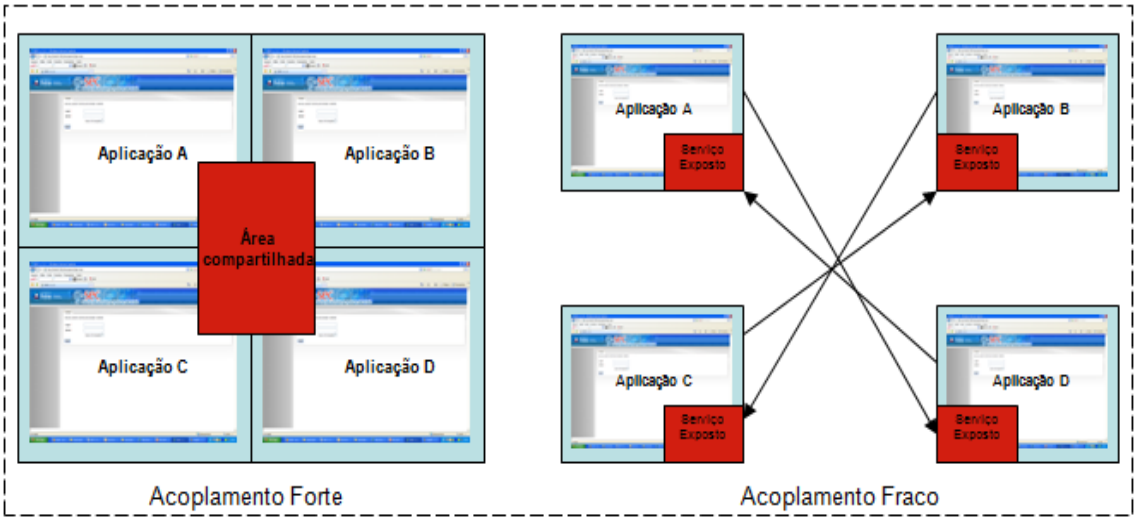

Figura 1 – Exemplo de Acoplamento fraco e forte

2.2 WEB SERVICE

A evolução tecnológica viabilizou para as instituições, públicas ou privadas, a possibilidade de enxergar seus sistemas de forma componentizada com os quais se podem integrar e interagir em um processo estruturado e organizado. Por exemplo, é possível criar *softwares* que possibilitam iniciar um processo estruturado de validar a transação em outra organização, planejar a entrega dos bens e ter um suporte pós-venda interno. Os *Web Services (WS)* são caracterizados por um conjunto estruturado de definições para a integração de diferentes tecnologias, viabilizando a componentização dos processos de negócio da organização, desde o nível dos sistemas corporativos, até aos processos organizacionais. Segundo Martins (2005,

p. 27) os WS compatibilizam as diversas tecnologias e suas soluções não integráveis, utilizando formatos comuns de trocas de informação através de um padrão aberto, o *Extensible Markup Language (XML),* e aplicando regras de interação. Através disso, é possível evitar um retrabalho de desenvolvimento obtendo a sua reutilização e a respectiva integração.

Um *Web Service* é um aplicativo Servidor que disponibiliza um ou mais serviços para seus clientes, de maneira fracamente acoplada. Geralmente, o protocolo de troca de mensagem é o SOAP. Ele expõe sua interface para os usuários usando um documento XML, conhecido como *Web Services Description Language (WSDL).* Com o uso de um WSDL podemos descobrir quais são os tipos de dados, formatos de mensagens e serviços que estão disponíveis através de um *Web Service*. Os clientes das aplicações que consomem os serviços expostos através de *Web Services* podem procurar um determinado serviço de duas maneiras distintas: quando eles têm acesso direto ao WSDL e nesse documento é definido como será criado o *Web Services*, ou utilizando o serviço via interface *Universal Discovery Description and Integration (UDDI),* conforme descrito em (SAMPAIO, 2006, p. 47).

2.3 WEBSERVICE DESCRIPTION LANGUAGE

WebService Description Language (WSDL) é um padrão baseado em XML para descrever os serviços expostos e a localização dos métodos do *WebService.* Ele funciona como um tipo de "*TypeLibrary*" do *Web Service*, ou seja, uma biblioteca das interfaces do WS. Ele também pode ser usado para a validação das chamadas dos métodos. Na figura 2 são ilustrados os principais elementos que compõem o WSDL.

O WSDL é um documento XML, projetado de acordo com os padrões especificados pela W3C, que descreve exatamente como um determinado *Web Services* funciona. Um documento WSDL é muito mais do que um mero manual de instruções de como se deve utilizar um determinado *Web Services*. Aplicações de *software* que criam *Web Services* podem processar o documento WSDL e gerar as mensagens *Simple Object Access Protocol (SOAP)* necessárias para invocar um serviço específico automaticamente (PULIER; TAYLOR; GAFFNEY, 2008, p. 58).

Figura 2 – Principais elementos de um WSDL

Fonte: Maciel (2007, p. 25).

O WSDL tem um conceito muito poderoso e por causa das suas capacidades, *web services* são conhecidos como elementos de *softwares* “auto-descritivo”. *Web Services* podem não somente inter-operar universalmente através do SOAP, como também serem descritos universalmente usando-se WSDL, conforme dito por (PULIER; TAYLOR; GAFFNEY, 2008, p. 68). Por exemplo, um desenvolvedor no Brasil pode implementar um *software* para acessar um determinado *Web Service* que está localizado em Tókio no Japão apenas lendo e processando o documento WSDL. Para o desenvolvedor realizar essa tarefa, ele não necessita entrar em contato com os demais desenvolvedores nas outras localidades, ler um manual particular ou comprar um *software* proprietário; teoricamente, o desenvolvedor precisa apenas estar ciente dos padrões a serem seguidos. Porém, neste simples exemplo não estamos levando em consideração as questões de segurança que

serão abordadas posteriormente neste trabalho. O importante é a “simplicidade” e portabilidade com que é feita esse tipo de troca de informação.

2.4 UNIVERSAL DISCOVERY, DESCRIPTION AND INTEGRATION

Universal Discovery, Dercription and Integration (UDDI) é o protocolo desenvolvido para a organização e registro de *Web Services*. Embora existam vários tipos de registros de *Web Services* disponíveis, o UDDI é identificado como sendo um padrão geral muito utilizado para esse tipo de serviço em uma rede específica. O registro de serviços deve fornecer algum mecanismo de localização dos serviços expostos para a entidade consumidora baseando-se em critérios de localização dos serviços (ROCHA, 2006, p. 24).

Os serviços expostos com a utilização do protocolo UDDI podem estar em qualquer lugar e serem chamados a qualquer momento e, de fato, a mesma função pode ser executada por um serviço diferente independentemente dos critérios de disponibilidade e preço. Nesse ambiente, operar sem um diretório com o protocolo UDDI para encontrar os serviços, obrigaria a fixar a localização dos serviços na aplicação consumidora, quebrando assim, uma das grandes vantagens da utilização de *Web Service* que é a transparência da localização dos serviços expostos.

Os Web Services possuem endereços *Internet Uniform Resource Locator (URL).* Para o computador requisitante, um *Web Service* é simplesmente uma URL. Uma vez que os *Web Services* usam protocolos Internet, podem ser acessados enviando-se uma mensagem SOAP de requisição ao endereço do *Web Service*. As URLs deles são a base de sua universalidade e transparência de rede, onde essa universalidade vem da forma padrão com que são descritos e a transparência vem da possibilidade de se utilizar um “nome lógico” na sua aplicação consumidora, que o UDDI pode converter para uma URL apropriada. Dessa maneira, caso a localização de um determinado serviço venha a ser alterado, o software consumidor não sofrerá impacto, independente de onde esteja localizado o serviço. Criar esses tipos de máscaras para as aplicações consumidoras fortalece a agilidade proposta pela tecnologia de *Web Service* (SAMPAIO, 2006; PULIER; TAYLOR; GAFFNEY, 2008).

2.5 SIMPLE OBJECT ACCESS PROTOCOL

Simple Object Access Protocol (SOAP) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML (PULIER; TAYLOR; GAFFNEY, 2008, p. 71). O que torna o SOAP especial e diferente de XML puro é que cada mensagem segue um modelo que foi especificado pelos padrões W3C. Ele foi criado, inicialmente, para posssibilitar a invocação remota de métodos através da Internet. O SOAP surgiu numa época em que as poucas alternativas para troca de mensagens eram as tecnologias DCOM, CORBA ou RMI. Os criadores do SOAP usaram um protocolo bem simples e difundido, o HTTP, e o XML como uma forma de comunicação mais padronizada e flexivel. Por esta razão ele não apresenta problemas quando encontra um *firewall*, já que não apresenta risco de segurança, tornando o uso de componentes remotos mais viáveis (SAMPAIO , 2006, p. 16).

Uma das mais importantes características do protocolo SOAP é a separação entre o formato do dado a ser transmitido e o mecanismo de nível inferior que irá transportar o dado. Há uma garantia da independência de plataforma e linguagem, pois se faz o uso de XML com HTTP que são protocolos abertos e bem difundidos na internet. Na maioria das vezes, os dados que são transportados em uma mensagem SOAP formam chamadas remotas de procedimentos *Remote Procedure Call* (*RPC)* que invocam procedimentos específicos em um provedor de serviços (MACIEL, 2007, p. 22).

Juntos os três padrões, SOAP, WSDL e UDDI, são combinados para dar a um *Web Service* as possibilidades de funcionar, autodescrever-se e ser encontrado dentro de uma rede. Na teoria, um *Web Service* não necessita de todos os três protocolos para funcionar, ele pode ser utilizado apenas com o SOAP, mas como mostrado na figura 3, para trabalhar mais eficazmente é necessário usar os três protocolos (PULIER; TAYLOR; GAFFNEY, 2008, p. 81).

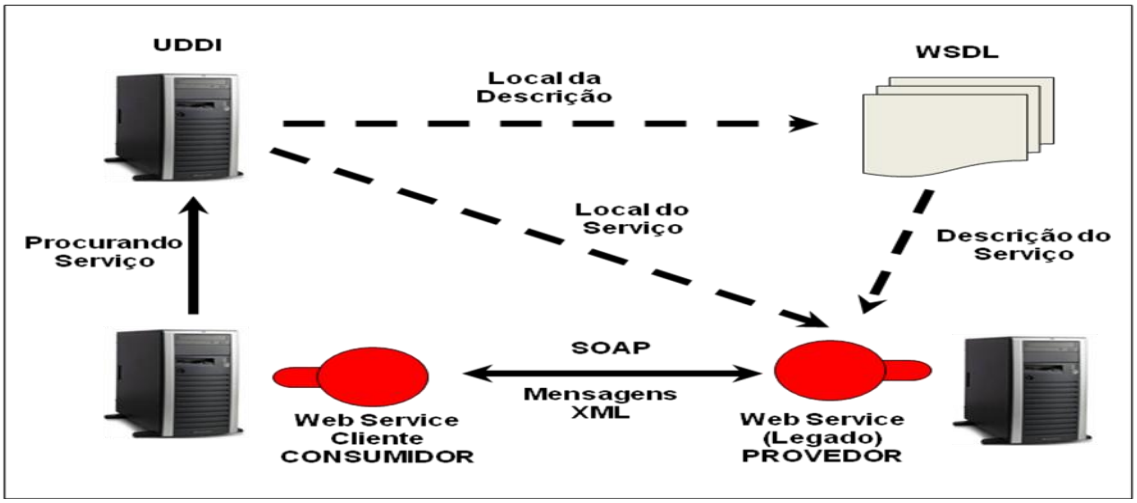

Figura 3 – Utilização dos protocolos SOAP, WSDL e UDDI
Fonte: Pulier, Taylor e Gaffney (2008, p. 82).

Segundo Rocha (2006, p. 35) uma mensagem SOAP é constituida por um “envelope” de código em XML onde é definido seu início e seu fim da mensagem. O “cabeçalho” (header) descreve de onde a mensagem foi enviada, para onde vai e como chegará no seu destino. O “corpo” (body) da mensagem SOAP possui os dados relevantes ou instruções sobre os procedimentos da requisição ou resposta da mensagem SOAP. A figura 4 ilustra uma mensage SOAP completa com “envelope” e corpo, viajando por uma rede de um consumidor (computador B) de web service para um provedor (computador A), neste exemplo um *mainframe*.

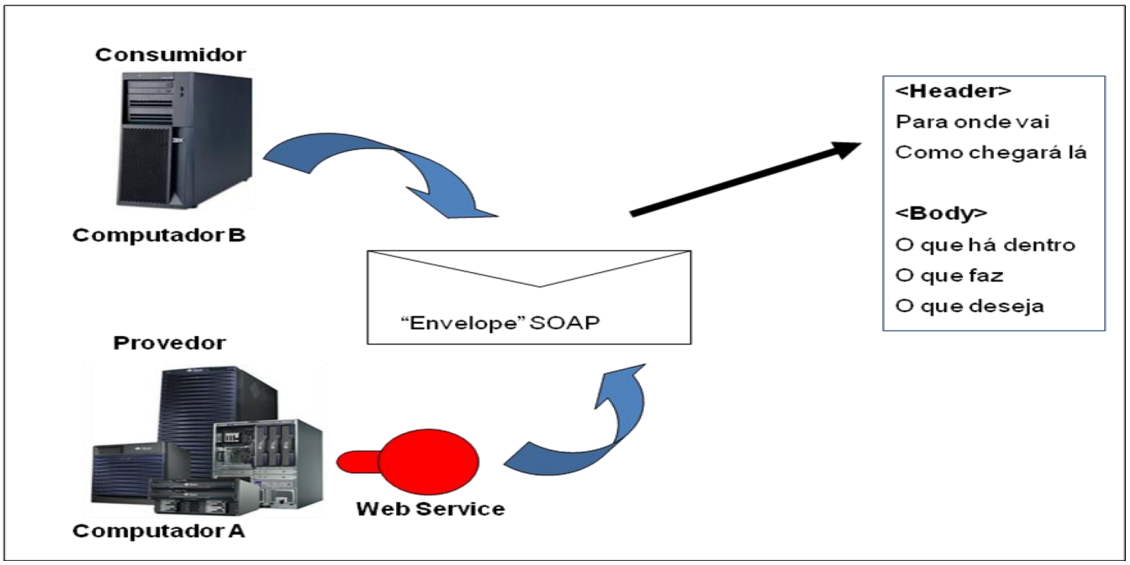

Figura 4 – Exemplo da forma de uma mensagem SOAP

Fonte: Pulier, Taylor e Gaffney (2008, p. 86).

2.6 DEFININDO SOA

A Arquitetura Orientada a Serviço (SOA) é composta por um conjunto de regras e conceitos que viabilizam a estrutura para arquitetar, desenvolver sistemas e aplicações orientadas a serviços, com o objetivo de obter o máximo de acoplamento fraco entre os serviços que são expostos. Muitas dessas regras e conceitos existentes na Arquitetura SOA, foram baseados em modelos já existentes, como por exemplo o CORBA e DCOM, conforme descrito por Rocha (2006, p. 33).

Larentis (2007), relataram que a arquitetura SOA pode ser composta por vários serviços que atuam como consumidores e/ou provedores. De forma genérica esses tipos de serviços ficam expostos para que aplicações e/ou outros serviços os acessem, fortalecendo a idéia de consumidores e provedores. A SOA também permite que haja a composição entre serviços, ou seja, provedores podem consumir serviços de outros provedores tornando-se consumidores, e vice-versa com os consumidores. Teoricamente, a composição é muito utilizada neste tipo de arquitetura, porém, os mecanismos de segurança devem ser adequados para este novo modelo de negócio. A arquitetura SOA é resumida como um conjunto de componentes que podem ser acessados e cujas interfaces podem ser divulgadas e pesquisadas, conforme figura 5.

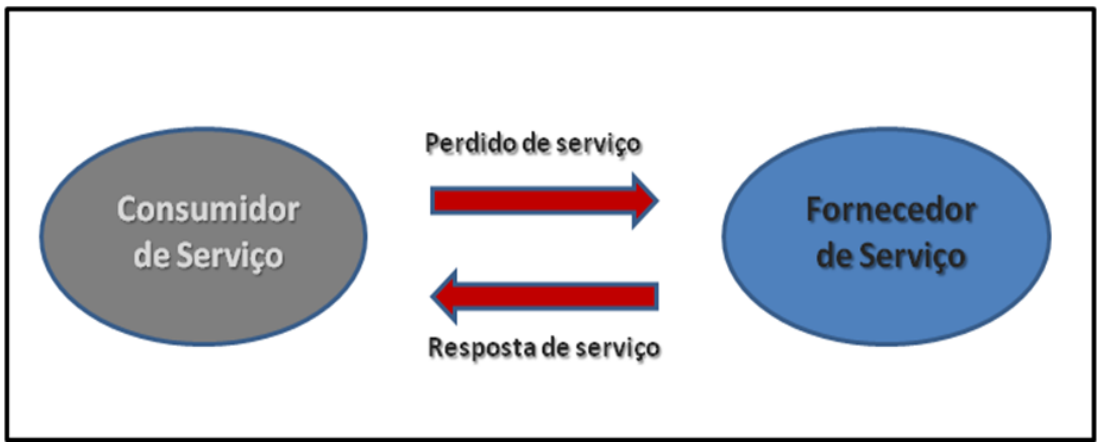

Figura 5 – Princípio básico da SOA

Um outro aspecto muito importante é que a arquitetura SOA não é somente construída sobre um conjunto de Web Services na Internet. Ela é um modelo conceitual, e não apenas um produto, aplicação ou módulo de um sistema a ser

implementado. É primordial ficar claro que a SOA é fundamentada em protocolos, conceitos, regras, padrões e acordos contratuais para a realização de trafego de informações e dados através de serviços expostos, e que Web Service está caminhando para se tornar de fato a plataforma preferida para implementar uma arquitetura SOA por ser independente de tecnologia.

O Web Service não é o único que pode ser usado para criar serviços expostos em uma arquitetura de software. Os Entreprise JavaBeans (EJB) podem ser utilizados na exposição de serviços. Porém, o EJB é basicamente um componente gerenciado que é criado, controlado e destruído pelo contêiner J2EE em que vive baseado em uma tecnologia proprietária, diferentemente, do Web Service (BOND e outros, 2003) que é baseado em padrão aberto.

A SOA proporciona uma transformação nos sistemas das organizações onde ela é implantada. Essa modificação refe-se ao fato de que os softwares passam a ser vistos como componentes e/ou serviços de uma aplicação, como exemplificado na figura 6. Neste aspecto, você pode movê-los e reconfigurá-los quando achar necessário. Com uma SOA, os arquitetos de softwares ficam livres da construção com “tijolos e pedras” que são geralmente solicitadas pelas arquiteturas corporatativas tradicionais (PULIER; TAYLOR; GAFFNEY, 2008, p. 70).

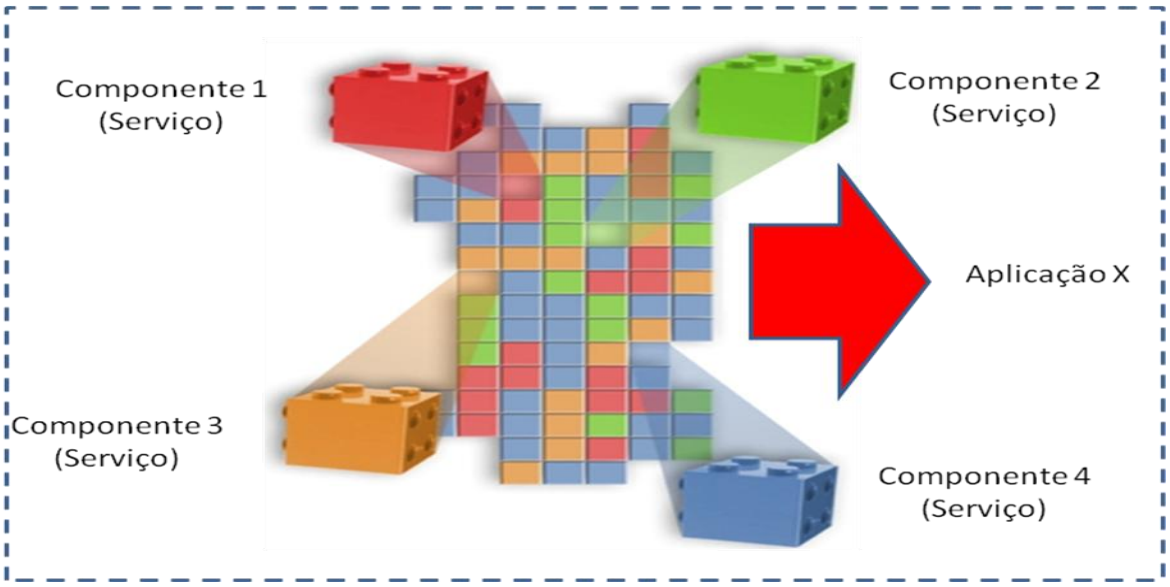

Figura 6 – Sistema composto por “legos”

A SOA pode ser considerada uma abordagem para EAI onde cada elemento importante é exposto como um “serviço”. Ela oferece um ambiente computacional distribuído com alto nível de interoperabilidade entre os sistemas. A SOA dá permissão ao arquiteto de software corporativo combinar elementos de software sem ter custos elevados de tempo e dinheiro, assumindo que os serviços expostos tenham sido implementados de forma inteligente. Em resumo, os princípios da SOA são o de não ser necessário ter o conhecimento específico e nem entender os detalhes para utilizá-los, basta combinar o que deve ser feito e a interferência do codificador será mínima ou nenhuma, é usada uma composição de diversos serviços para um objetivo maior, os fornecedores de serviços prestam um serviço muito melhor do que uma aplicação isolada e a localização dos serviços são armazenadas em um repositório que podem ser encontrados através de uma consulta.

2.6.1 Serviço

Um serviço é um componente que procurar atender a uma função de negócio específica de seus clientes. Ele recebe requisições e as responde ocultando todo o detalhamento do seu processamento. Como isso, podemos obervar que um serviço exposto é encapsulado em relação ao seu cliente, ou seja, quaisquer outras aplicações ou componentes que auxiliam o serviço em questão devem ser utilizados somente dentro do código privado do serviço, não possibilitando a exposição das regras de negócio para seus clientes (LARENTIS, 2007, p. 37).

Segundo Sampaio (2006, p. 14) um serviço deve executar unidades complementares de trabalho, não dependendo do estado de outros componentes externos. Sendo assim, outro aspecto muito importante é que esses tipos de serviços devem ser “*Stateless*”, ou seja, eles não possuem armazenamento de estado de conversação, com isso elevando o grau de reutilização. Outro aspecto diz respeito à granularidade dos serviços, que quanto maior ela for, mais adequado será o serviço. Logo um componente que atua como serviço não deve ter operações muito complexas. Resumindo, um serviço executa uma função atômica (ou transação) e todas a operações intermediárias devem ser realizadas apenas pelo serviço, e não pelo cliente consumidor.

2.6.2 Barramento de Serviços Corporativos

Um barramento de serviços corporativos *Enterprise Service Bus (ESB)* não é considerado um produto, mas sim um padrão de arquitetura. Sua função é atuar como um intermediário entre a implementação de um serviço e a forma como ele é exposto para seus consumidores. Por esta razão o barramento deve ser robusto e possuir recursos para viabilizar a exposição e composição de serviços, roteamento de mensagens, transformação e enriquecimento de dados, segurança, dentre outros aspectos, sempre utilizando padrões abertos para viabiliar esses aspectos (CUNHA, 2008; GOMES e outros, 2007, p. 4).

O ESB representa um pilar dos serviços, mensagens, comunicações, transformações e de segurança sobre o qual se pode acoplar sistemas ou simplesmente interoperar entre eles. Eles seguem os princípios da SOA, possibilitando a integração com diferentes tipos de serviços, e incluem de forma geral as seguintes áreas funcionais: *Standard-based Communication Infrastructure, Standard-based Connectivity, Standard-based Transformation Engines, Service Oriented Architecture (SOA) for application deployment e Standards-based Security* CUNHA, 2008; GOMES e outros, 2007, p. 5; MARTINS, 2005, p. 67).

O uso do ESB permite uma melhor gestão do conjunto de componentes e/ou serviços, de forma que ele será responsável por redirecionar as conexões dos serviços solicitantes para as novas localizações dos fornecedores. Na figura 7 é ilustrado o uso do ESB, de maneira que os diversos serviços expostos (componentes) das referidas aplicações são integrados através dele.

Na arquitetura SOA é um erro comum dizer que um ESB “implementa” a SOA ou o fato de se utilzar um ESB já caracteriza a arquitetura com sendo uma SOA. Um ESB é responsável por grande parte dos requisitos que a SOA estabelece, mas não todos. Não é correto afirmar que Implementar um barramento de serviços há uma aderência automática à SOA. Para ter isso vai depender da forma como são modelados e concebidos os sistemas. Não há como ter SOA somente tendo um ESB. Porém, o ESB é um componente importante em uma solução SOA, tendo em vista que toda a integração de sistemas acontece através dele (COSTA; CARVALHO, 2007, p. 3).

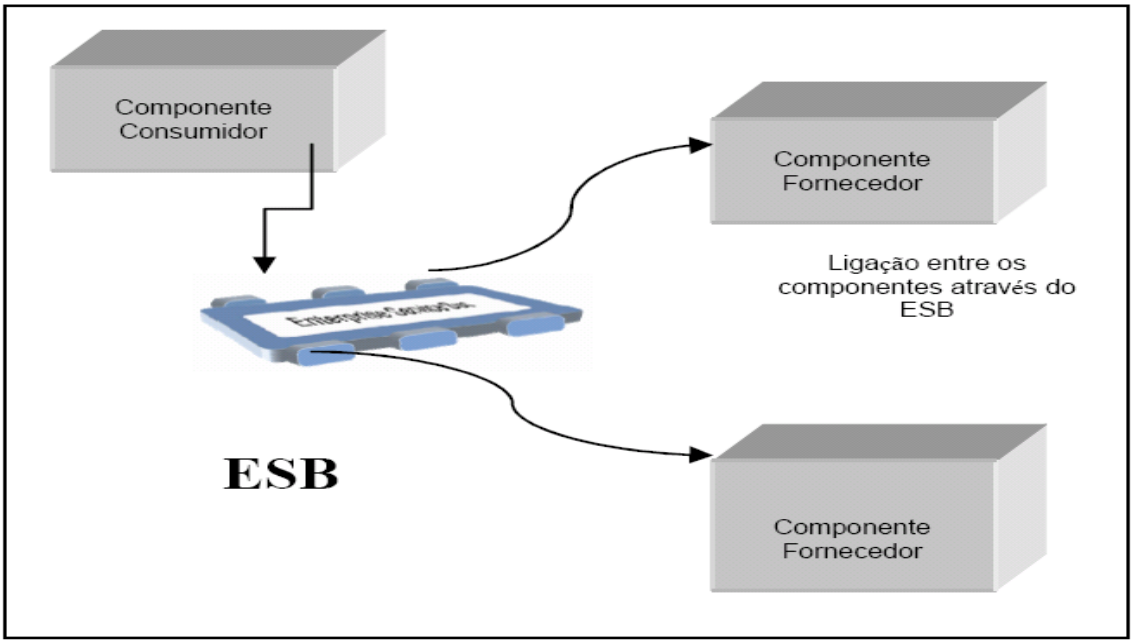

Figura 7 – Conexão dos componentes através do ESB
Fonte: Martins (2005, p. 66).

2.6.3 Relação entre WS e SOA

Os *Web Services* não são resquisitos promordiais para se obter, implementar ou estabelecer uma arquitetura SOA. Segundo Natis (2003) Web services são baseados em especificações tecnológicas, enquanto a arquitetura SOA é baseado em princípios de desenvolvimento de software. De forma notável, Web Services Description Language (WSDL) utilizado por Web Services é o padrão que define interfaces para arquitetura SOA, este é o momento que Web Services e a arquitetura SOA fundamentalmente se conectam.

Por outro lado, Rocha (2006) afirma que os *Web Services* são considerados componentes que posssibilitam às aplicações receber e enviar informações descritas no formato XML. Cada aplicação tem seu formato proprietário para descrever as suas informações, porém, ocorre uma tradução para uma liguagem universal, o formato XML. As instituições W3C e OASIS são as responsáveis pela padronização dos *Web Services* (OASIS, 2006; W3C, 2008).

Em adição às discussões levantadas por Bertoni (2006), é relevante ressaltar que os Web Services são aplicações fracamente acopladas que interagem dinamicamente através de redes TCP/IP (Internet e/ou Intranets), através da

publicação, localização e invocação destes pela Web. Assim, os serviços disponíveis podem ser descobertos pelos consumidores, possibilitando transações empresariais e comerciais pela rede, sendo este o princípio básico da arquitetura SOA (BERTONI, 2006, p. 23).

Em suma, a arquitetura SOA é considerada um padrão para se projetar sistemas, enquanto que os *Web Services* são serviços implementados através da utilização de padrões. Os *Web Services* possibilitam a implementação da arquitetura SOA e os benefícios do uso de uma arquitetura baseada em SOA através de implementações com *Web Services* propiciam o acesso a serviços expostos independentemente de plataforma e interoperabilidade entre os diversos fabricantes da industria de TI.

2.6.4 Integração de Aplicações Corporativas com SOA

Com projetos de EAI sofrendo altos custos, ciclos de vida longos e uma alta taxa de fracasso, como já foi mencionado no capítulo 2, a questão de integração vem se caracterizando como uma enorme fonte de preocupação para os gestores de TI. Pulier, Taylor e Gaffney (2008, p. 67) assim como Cummins (2002, p. 93) relatam que os projetos de EAI geralmente são iniciados com boas intenções e inteligentes. Entretanto, políticas corporativas e mudanças não planejadas nos processos de negócios acabam resultando em projetos EAI presos em linhas “ilhas de integração” incompatíveis, ou seja, projetos com baixa integração entre eles. Com isso, aumentando os obstáculos para a integração entre os diversos sistemas utilizados pela organização.

Como já foi dito anteriormente neste trabalho, os *Web Services* têm um grande potencial para reduzir as necessidades de custo, de tempo e a complexidade de EAI ao possibilitar que aplicações interajam utilizando padrões universais como SOA, WDSL e UDDI. O fato dos *Web Services* serem baseados em padrões abertos aumenta o grau de independência em relação ao fornecedor. Porém, as questões de segurança, assim como considerações políticas e organizacionais, os caracterizam como uma solução em potencial nas situações de EAI (PULIER; TAYLOR; GAFFNEY, 2008, p. 73). Baseando-se nisso, Garcia (2001, p. 18) e Cummins (2002, p. 102) relataram que os pacotes de EAI existentes, provavelmente irão perdurar

durante muito tempo, devido à sua capacidade de gerenciar o volume de transações no nível corporativo e fornecer garantia de desempenho dos sistemas legados. Porém, eles também descreveram que há uma grande probabilidade de que estes pacotes de EAI sejam gradativamente substituídos pelos padrões de *Web Services*.

Resumindo, a EAI é a forma com que algumas empresas, públicas ou privadas, adotam para prover e consumir seus serviços entre parceiros de negócios ou mesmo expor serviços dentro da sua própria infra-estrutura de TI, entre sistemas com arquiteturas e tecnologias diferentes. A SOA também tem esse fundamento como seu princípio. Na maioria das arquiteturas EAI é tratada também separadamente pelas organizações, que, inicialmente, expõem seus *Web Services*, e, em segundo lugar, tentam acoplar os mecanismos de segurança mediante as exigências do negócio (ROCHA, 2006, p. 41).

2.6.5 Segurança na SOA

Os problemas de segurança inerentes à SOA são derivados das formas com que os parâmetros de segurança utilizam os padrões abertos. Existem dois lados em relação ao problema de segurança na arquitetura SOA, pois além dos padrões não terem um proprietário, eles foram desenvolvidos sem ter segurança em mente. Os *Web Services* foram criados por um consenso da indústria de TI, como uma forma de desenvolver *software*, dentre outras coisas, permitir a criação de código reutilizável, tornar o desenvolvimento mais simples e facilitar a interoperabilidade entre aplicações mais heterogêneas. Embora seus objetivos tenham sido alcançados com o surgimento de padrões abertos, os mesmos não se preocupam com a segurança das informações trocadas. Eles por sí só não tratam as questões de segurança, sozinhos são completamentes inseguros. Os *Web Services* foram criados para quebrar a barreira imposta por firewalls, daí, a grande importância do assunto Segurança no mundo da Arquitetura Orientada a Serviços (DEGAN, 2005, p. 45).

A segurança é outro aspecto na adesão da arquitetura SOA para fins de EAI. Embora arquiteturas complexas que usam SOA pareceram impressionantes, em relação a sua complexidade, elas podem ser totalmente inseguras. Uma vez que os

Web Services utilizam protocolos de Internet para mensagens e esse tipo de mensagem é projetada para passar pelo *firewall*. Como isso, os *Web Services* são expostos a brechas de segurança.

Segundo Cummins (2002, p. 110) é importante compreender que a maioria das infra-estruturas de segurança é gerada para interações homem-máquina, enquanto que os *Web Services* trabalham com a interação máquina-máquina. Até recentemente, a maior parte da atenção e do desenvolvimento de aplicações estava voltada para o ambiente muito conhecido de acesso homem-máquina, incluindo soluções que fornecem gerenciamento de identidade e soluções de *single-sign-on* (SSO) para usuários que acessam aplicações através de *browsers*.

Na arquitetura convencional, a tecnologia de segurança do sistema, como por exemplo o *firewall*, evita que usuários não autorizados tenham acesso às informações restritas da organização. Entretanto, na arquitetura SOA ela prega flexibilidade e abertura para que seja acessada por diversos sistemas, fortalecendo a reutilização e composição de novas aplicações. Nesse ambiente as aplicações são expostas como serviços, mas um novo mecanismo de segurança não é imposto, deixando-as bem frágeis. Em virtude da natureza grosseira do mecanismo de segurança, não temos como identificar se a máquina que está solicitando a utilização do serviço exposto está autorizada ou autenticada (ROCHA, 2006, p. 23).

Apesar desses problemas há soluções para melhorar a segurança em uma Arquitetura Orientada a Serviços. Uma solução completa pode ser composta por diversas subsoluções, cada uma lidando com certo aspecto de segurança de SOA. Dependendo das suas necessidades e infra-estrutura de segurança existente, haverá uma necessdade de optar por soluções que podem diferir umas das outras. Pulier, Taylor e Gaffney (2008, p. 73) destacam os seguintes produtos de segurança de SOA:

a) **Monitoramento de mensagens SOAP** - é baseado em interceptação de mensagens SOAP, sendo uma maneira de construir o alicerce de uma solução de segurança SOA. A interceptação de SOAP é pôr um componente chamado “Interceptador SOAP”, no caminho das mensagens SOAP, que trafegam entre os provedores e consumidores de serviços

expostos. Ele analisa as identidades dos usuários contidas nos cabeçalhos das mensagens que monitora e as compara com os nomes armazenados na infra-estrutura de segurança existente. O resultado é a autenticação e autorização de remetentes e destinatários de mensagens;

b) **Autenticação federada** - autenticação federada é um processo no qual diversos parceiros entram em acordo que um determinado grupo de usuários pode ser autenticado por um dado conjunto de critérios. Para utilizar uma autenticação federada em uma segurança SOA, o interceptador de SOAP encaminha uma mensagem que chega para uma solução de segurança, que compara a identidade do usuário contida no cabeçalho da mensagem com os usuários listados no banco de dados de autenticação federada. Uma vez aprovado, a solução de segurança de SOA informa que o usuário foi autenticado;
c) **Proxy de aplicação** - uma forma altamente eficaz de melhorar a segurança de sistemas centrais é não permitir que qualquer um alcance a plataforma que hospeda o serviço. Isso pode ser feito instalando um *proxy* para *Web Services* dentro da arquitetura SOA. Uma vantagem adicional da utilização de *proxy* é a sua capacidade de reduzir a carga na infra-estrutura de segurança da organização.
d) **Gerenciamento de contrato** - um contrato é um grupo de regras que governa o uso de um determinado *Web Service*. Como por exemplo, um contrato pode estipular que quantidade e/ou tempo de acessos um usuário pode ter por dia a um *Web Service* ou a quantidade (*MBytes*) de informação que pode ser acessada. Essas medidas podem aumentar a segurança, pois os contratos são úteis para determinar se a SOA está funcionando adequadamente ou sendo mal utilizada devido às falhas de segurança;
e) **Certificados, chaves e criptografia** - nos ultimos anos, a TI apoiou diversas técnicas e tecnologias de segurança em criptografica. Na SOA podemos fazer uso desses mesmos algoritmos já existentes. Esses processos, seja, certificado, chaves e criptografia, podem desempenhar um papel muito importante na busca de uma maior segurança para os serviços expostos na sua arquitetura SOA;

f) **Criptografia de XML** - neste caso para se manter o conceito de uso dos padrões abertos na arquitetura SOA, as mensagens no formato XML são encriptadas, porém, o conteúdo só poderá ser entendido por quem possuir a chave para quebrar o código;
g) **Assinaturas digitais** - baseada em chaves, a assinatura digital é um número que captura unicamente sua identidade e o conteúdo da mensagem, passando os dois conjuntos de informações (chave e mensagens) por um algoritmo com esse fim. A diferença entre assinatura digital e o processo de criptografia, dito anteriomente, é que na assinatura digital a mensagem não é totalmente criptografada;
h) **Proteção de ataque duplicado e auditoria** - a solução de segurança SOA idendifica a mensagem SOAP com um número único, e configura a solução para bloquear mensagens duplicadas, evitando que a mesma mensagem seja enviada duas vezes. Já auditoria é um uso avançado das funcionalidades de rastreamento das mensagens.

2.6.6 Vantagens x Desvantagens da SOA

A Arquitetura Orientada a Serviço também possui suas desvantagens e vantagens como qualquer outra tecnologia. As organizações adotam a SOA com o objetivo de alcançar aumento da competitividade, melhorar a eficiência dos sistemas legados, ganhar dinamismo nos processos para melhor aproveitar as novas oportunidades de negócios, utilizar a infra-estrutura existente dos sistemas legados, reutilização dos códigos dos legados, reduzir custos, aumentar a flexibilidade dos novos sistemas, potencializar a portabilidade, dentre outras.

Por outro lado, uma das possíveis desvantagens da arquitetura SOA relatadas por Rocha (2006, p. 31) é o gerenciamento dos mecanismos de segurança dessa arquitetura. A complexidade na interoperabilidade dos mecanismos de segurança que são solicitados pelos novos processos de negócio, inevitavelmente, será um fator que constribui fortemente para dificultar a evolução da arquitetura SOA. A imaturidade dos mecanismos de segurança fornecidos pela indústria de TI, para as aplicações baseadas no padrão XML ou uso inadequado dos mecanismos de segurança citados anteriomentes neste trabalho na arquitetura SOA, podem levar as organizações a rever a aplicabilidade da SOA. Resumidamente, os mecanismos

de segurança na arquitetura SOA em relação aos processos de negócios são considerados fatores críticos na Arquitetura Orientada a Serviços.

Um outro aspecto é o gerenciamento de mudanças como uma área problemática para arquiteturas distribuídas. Se não for trata corretamente em uma SOA, pode ser tão problemática quanto em arquiteturas antigas. Mudanças são um fator cosntantes em gerenciamento de TI, e a SOA não é exceção. Conforme os requisitos e os processos de mudam, também mudam os sistemas que os suportam. Embora a SOA alivie muitos dos estresses que acompanham o gerenciamento de mudanças no modelo tradicional, é ainda um área que requer atenção de perto e gerenciamento adequado para funcionar eficazmente.

O gerenciamento de mudanças da SOA traz desafios equivalentes áqueles do modelo tradicional. Quando um *Web Service* muda, tanto os aspectos da invocação quanto da resposta do novo *Web Service* devem ter um desempenho conforme prometido ou haverá um problema. Se organização substituir o *Web Service* de aluno, por exemplo, por um novo o administrador da SOA deve ser capaz de garantir aos usuários que cada sistema que invoca o novo *Web Service* obterá os resultados de que precisa.

E por fim, o fator de mapeamento dos requisitos na SOA. A transformação dos requisitos de negócio da organização durante o processo de implantação de uma SOA não é uma tarefa tão simples. Por exemplo, temos uma determinado requisito de negócio que possui cinco regras específicas, quando houver a migração das funcionalidades para serviços podem haver redução dessas regras. Essa tarefa de determinar quais as referidas regras serão excluidas no novo serviço depende de um acordo com os gestores do respectivo serviço. Em resumo, não pedemos simplesmente mudar de funcionalidade para serviço uma determinada aplicação, deve haver um estudo e negociação junto aos gestores.

2.7 TENDÊNCIAS DE MERCADO PARA SERVIÇOS

Nas atuais organizações, públicas ou privadas, viver uma época muito empolgante para o envolvimento com tecnologia da informação corporativa. A arquitetura Orientada a Serviços, um sonho muito antigo na indústria de TI, está se

tornando uma realidade em um ritmo que poucos antecipavam há pouquíssimos anos. Os organismos de padronização continuam lutando com os conflitos sobre padrões e o movimento bem sutil de padrões proprietários sempre parece estar se esgueirando pelas sombras, ameaçando prejudicar a essência verdadeira que a SOA está tentando alcançar (PULIER; TAYLOR; GAFFNEY, 2008, p. 68).

Uma das características do serviço é a intangibilidade. Para amenizar esse tipo de problema, a indústria de TI, com especialidade em serviço, contará com o apóio de vários modelos, com o intuito de serem intermediadores entre os provedores e consumidores dos serviços expostos. Tomando como exemplo o padrão *Infrastructure Resource Management (IRM),* que fornecem diversos serviços para melhor gerir a infraestrutura de TI, como por exemplo o gerenciamento de mudanças, problemas e incidentes, com o objetivo de prever os desvios no ambiente de TI, de forma a reduzir os seus efeitos e impactos (COSTA; CARVALHO, 2007, p. 4).

Segundo Rocha (2006, p. 30) a utilização da arquitetura SOA nas organizações deve estar intimamente ligada a um processo de gestão do conhecimento em relação aos componentes reusáveis da mesma. O funcionamento adequado desse tipo de gestão está associado ao uso de uma biblioteca central com os artefatos organizados de maneira que a pesquisa possa ser feita por diversas categorias de informação, fortalecendo o reuso dos componentes de *software*.

Um dos fatores benéficos do uso da SOA é que ele utiliza a arquitetura centrada no cliente, modificando o paradigma dos modelos e práticas que tendem a ser centradas na aplicação. A arquitetura focada no cliente provê uma abordagem de configuração em relação ao usuário, ao invés de focar em um único modelo para todos os usuários ou onde há *workflows* pré empacotados (NATIS, 2003, p. 14).

A Arquitetura Orientada a Serviços é uma nova oportunidade, mas um conceito antigo – tão antigo quanto *software* em si: construir uma vez e reutilizar muitas vezes, seja dentro da própria organização ou entre parceiros (PULIER; TAYLOR; GAFFNEY, 2008, p. 73).

Segundo Costa e Carvalho (2007, p. 5) o UDDI embora seja projetado para se tornar um repositório de componentes de serviços, geralmente não é utilizado com a aderência com qual foi planejado. Uma vez os projetistas desvendando os serviços que serão expostos, eles tendem a fazer uma conexão mesmo flexível entre os componentes de serviços fazendo ligações diretas entre os componentes, conforme figura 8. E logo o uso da SOA é interessante, pois resolve esse problema com o acoplamento franco entre os componentes e a independência de tecnologia.

O uso do *Enterprise Service Bus* (ESB), ou simplesmente Barramento de Serviços, permite às organizações gerir melhor os seus conjuntos de componentes, de maneira que eles mesmos sejam responsáveis e capazes de redirecionar as conexões dos serviços consumidores para as novas localizações dos provedores. Com isso, qualquer que seja a alteração no ambiente, ou na localização dos serviços expostos, não haverá impacto no componente consumidor, pois sua interação é realizada apenas com o ESB (MACHADO, 2004, p. 44).

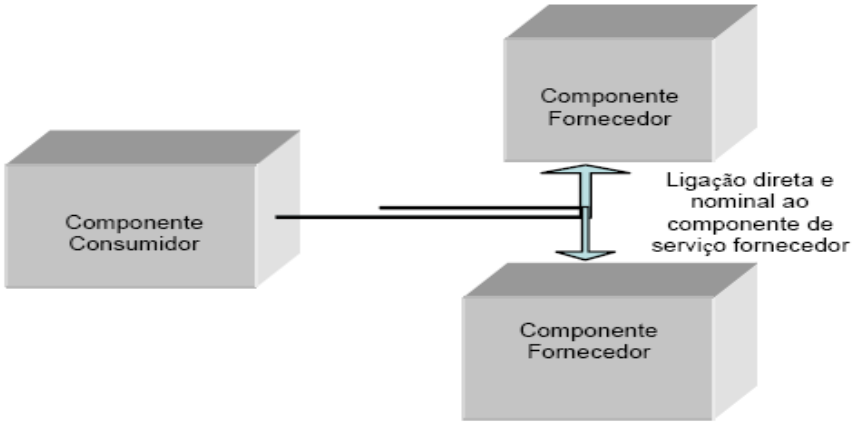

Figura 8 – Conexão entre componentes.
Fonte: Costa e Carvalho (2007, p. 7).

A tecnologia de *Web Services*, embora já seja amplamente difundida, ainda tem que percorrer muito o caminho da evolução. Esta tecnologia, comparada a outras já consolidadas na indústria da TI, como manutenção do estado transacional e composição dos serviços, ainda não está bem resolvida. A indústria da TI tem procurado trabalhar em novas especificações para tentar resolver essas questões. Como *WS-Transaction* e *WS-Coordination* (COSTA; CARVALHO, 2007, p. 8).

2.8 TRABALHOS RELACIONADOS

A exposição de serviços através de Web Service fazendo uso da SOA não é algo muito novo, visto que sua utilização possui diversas referências para implementação. Esta seção busca os principais trabalhos relacionados a esse tema e que trazem contruições para solucionar alguns problemas já encontrados, além de servirem para posiocionar o presente trabalho em relação a outros na mesma área. A seção apresenta uma solução de migração de sistemas legados para SOA denominada SMART, que não contempla aspectos de codificação. Em seguida, é apresentada uma técnica para encapsulamento de sistemas legados para reuso em SOA. Logo após, é apresentada uma arquitetura denomidada ARUBA que permite a geração de serviços a partir de sistemas legados. Por fim, apresenta duas ferramentas comerciais que permite geração de serviços.

2.8.1 Migração de sistemas legados para SOA

Segundo Lewis, Morris e Smith (2005), torna possível que um sistema legado trabalhe com Web Service é as vezes algo relativamente simples. Os Web Services são baseados em padrões abertos e são configurados para receber mensagens, transformar seu conteúdo, fazer uso de código legado, e opcionalmente alterar os resultados das mensagens retornadas ao fornecedor (LEWIS; MORRIS;, SMITH, 2005). Porém, segundo Larentis, existem características de sistemas legados que podem tornar o processo de criação de Web Service uma tarefa complicada, por exemplo:

a) Tempo de existência;
b) Linguagem de programação;
c) Arquitetura;
d) Estado futuro baseado em serviço;

A técnica SMART (*Service-Oriented Migration and Reuse Technique*) foi derivada do modelo *Options Analysis for Reengineering* (OAR) desenvolvido pelo *Software Engineering Institute* (SEI) utilizado para suporte na análise de reuso de sistemas legados (BERGEY, 2002). A SMART está dividida em cinco atividades descritas a seguir:

a) **Estabelecer o contexto com os stakehouders** - identifica características sobre o sistema legado, seu objetivo atual, e o que deverá ser migrado para serviço;
b) **Descrever as capacidades existentes** - obter dados descritivos sobre os componentes do sistema legado;
c) **Descrever o estado fututo baseado em serviços** - obter evidência sobre os serviços potencias que podem ser criados a partir dos legados;
d) **Analisar as diferênças entre o estado baseado em serviço e as capacidades existentes** - Identificar a diferença entre o estado existente e o futuro e determinar o nível de esforço e custo necessários para converter os componentes legados;
e) **Desenvolver uma estratégia para migração de serviço** - Identificar os componentes específicos para migrar, planejar e ordenar os esforços para a migração.

Resumidamente, a técnica SMART propôs um modelo para expor as funcionalidades de sistemas legados como serviços, porém, esta tecnica é muito direcionada em gerar informações que justifiquem a viabilidade deste objetivo.

2.8.2 Encapsulando sistemas legados para reuso em SOA

Em Sneed (2006) é apresentado um modelo que ajuda na identificação de quais partes de um sistema legado poderiam ser expostos com *Web Services* e uma técnica de como fazer essa transformação. Esta solução está baseada num método disponibilizado por uma ferramenta, que permite integrar o código legado dentro de uma interface XML em funções separadas, para serem oferecidas como *Web Services* para outro usuário externo. O foco deste método consiste de três passos básicos para a criação de *Web Services* a partir de um sistema legado:

a) **Preservando o código legado** - neste primeiro momento, é sugerido que seja realizada uma análise do código legado existente para identificar qual parte do código agrega valor para ser reutilizado;
b) **Encapsulando o código legado** - neste passo, é fornecer uma componente extraído do código legado através de uma interface WSDL;

c) **Tornar o código disponível como um *Web Service*** - o último passo desta solução diz respeito a fazer um *link* do *Web Services* com o processo do negócio. Isto é feito por um componente *proxy*.

Resumidamente, este trabalho tem como proposta um método para identificar quais funções serão disponibilizadas como *Web Services*. O código legado é preservado, porém a lógica de negócios que será disponibilizada deve ser extraída e salva em outro arquivo, de forma que todas as funções dependentes sejam juntamente agrupadas em uma única função sequencialmente. Após isso, são identificados os parâmetros de entrada e saída, gerendo uma XML capaz de fazer a tradução destes parâmetros e por fim, é utilizado um *proxy* para fazer a comunicação entre o cliente e o código legado alterado.

2.8.3 Uma arquitetura para geração de serviços a partir de um sistemas legados de forma intrusiva

A construção de sistemas baseados no conceito de serviços estabelece uma abstração entre os processos de negócios e as aplicações existentes nas organizações, porém, constitui uma atividade que requer alguns esforços e que pode ser realizada de diversas maneiras. Muitos são os padrões de tecnologia que podem ser utilizados para a geração desses serviços, cujo o principal objetivo é contribuir para a integração de diferentes aplicações (LARENTIS, 2007).

A arquitetura para geração de serviços a partir de uma sistema legado de forma intrusiva (ARUBA) é uma arquitetura que através de uma interface permite o mapeamento das regras de negócios para a exposição como serviços. Assim, integrando as tecnologias, WSDL, SOPA e UDDI, ao conceitos de sistemas legados não desenvolvidos sob a forma de Web Services, é possível reutilizar este código sem que haja necessidade de alterá-lo, mapeando as funcionalidades e gerar os serviços como Web Services, para reuso em outra tecnologia, como SOA (LARENTIS, 2007).

Uma das preocupações da arquitetura ARUBA é evitar que sistemas legados estáveis necessitem ser alterados para que possam ser utilizados na SOA. Buscando otimizar este cenário. Um mapeamento do código é feito incialmente,

após isso, ocorre a geração do *Web Services* por um método da arquitetura, assim como a geração de arquivos de configuração (LARENTIS, 2007).

2.8.4 Sistemas legados e SOA – Soluções comerciais

Durante a contrução deste trabalho foram encontradas ferramentas comerciais que fazem uso de Web Services para expor as funcionalidades dos sistemas legados ou novos para uso em SOA, entre eles são: IBM SOA *Foundation* e Microsoft *Biztalk Server*. Cada uma destas soluções apresenta uma abordagem diferente ou proprietária para geração de *Web Services*, possuem diferentes mecanismos para desen volvimento SOA e fornecem soluções desde integração com sistemas legados até a disponibilidade dos serviços de novas aplicações para uso por outras.

2.8.4.1 IBM SOA *Foundation*

Segundo a IBM, a SOA fornece flexibilidade e reuso dos processos de negócios ou recursos de uma empresa. A SOA propõe uma arquitetura que combina conexões adaptáveis com relações bem definidas, baseadas em tecnologias padrão para ajudar a flexibilizar infraestruturas existentes (IBM CORPORATION, 2005).

Os serviços de SOA são extensíveis, baseados na implementação de novos serviços ou recursos de TI existentes. Assim, o desenvolvimento de uma SOA começa com uma infraestrutura flexível, robusta que pode ser utilizada em conjunto com outra infraestrutura existente e recursos de TI para proporcionar mais valor ao negócio.

A plataforma IBM SOA *Foundation* foi densenvolvida para atender todas as camadas de arquitetura de referencia SOA da IBM, na qual define serviços de TI requeridos para fornecer suporte a cada um dos estágios do cliclo de vida de SOA (Modelo, Construção, Implantação, Controle e Governança/Processos). A arquitetura de referência SOA inclui um ambiente de desenvolvimento, gerenciamento de serviços, integração de aplicações e processos de serviços em tempo de execução (IBM CORPORATION, 2005).

2.8.4.2 Microsoft

A Microsoft disse que a orientação a serviços é uma abordagem para organizar recursos distribuidos de TI em uma solução integrada que desmembra grupos de informação e maximiza a organização na agilidade dos negócios. Os serviços de comunicam entre si por meio de formatos de mensagens bem definidos; isso significa que a confiabilidade do aplicativo de sistemas conectados sofrerá uma grande influência da confiabilidade da infraestrutura de mensagens que ela usa para fazer a comunicação entre os serviços. A orientação a serviços une fonte de informações autônomas construido um pacote em grande escala de sistemas operacionais, tecnologias, e protocolos de comunicações (MICROSOFT, 2006a).

O Microsoft *Biz Talk Server* é o aplicativo de destaque na colaboração no desenvolvimento de aplicações voltadas para SOA (MICROSOFT, 2006b). Ele é denominado um ambiente para desenvolver SOA. É um servidor da Microsoft que permite que os clientes integrem sistemas, funcionários e parceiros comerciais e reúne as funcionalidades de integração de aplicativos e automação de processos das tecnologias XML e *Web Services*. O *Biz Talk* Server funciona como um mecanismo de execução de processo e como um *hub* de sistema de mensagens, fornece um suporte ao consumo de serviços Web como parte do processo comercial, expondo os processos e aplicativos de linha de negócio como serviços Web. Ele fornece suporte SOAP, UDDI, WSDL entre outros, utilizando adaptadores ASMX e *Web Services Enhancements* (WSE). Ele também acrescenta a capacidade de chamar serviços da Web por meio de serviços de mensagens públicas/subestilo e fornece um adaptador *Windows Communication Foundation* (WCF) para incorporar serviços da Web WCF a processos comerciais.

CAPÍTULO 3

3 UMA ARQUITETURA PARA INTEGRAÇÃO DE SISTEMAS

Neste capítulo abordaremos a instituição estudada e uma solução proposta com base na Arquitetura Orientada a Serviços para ambientes com sistemas heterogêneos. Para isso, apresentaremos um breve histórico da organização, os objetivos desta arquitetura, os legados encontrados na organização estudada, uma proposta de componentização dos requisitos de negócio da instituição e alguns *cases* de aplicações que estão fazendo uso da arquitetura proposta por este trabalho.

3.1 A SECRETARIA DA EDUCAÇÃO DO ESTADO DA BAHIA

A informática na Secretaria da Educação do Estado da Bahia (SEC) tem como órgão central a CMO – Coordenação de Modernização, unidade administrativa da Diretoria Geral. O papel da CMO é de planejar, coordenar, avaliar e controlar a execução de atividades de informação e informática, bem como as atividades de organização e modernização administrativa, através de soluções sistêmicas, com base em recursos metodológicos e tecnológicos avançados. A CMO se estrutura da seguinte forma:

a) **Suporte Técnico de Informática** - tem o papel de Gerenciamento e Manutenção de Rede, Hospedagem de Sites e Sistemas, Suporte ao Usuário e Manutenção de Equipamentos;

b) **Área de Sistemas** - abrange as funções de Análise de Sistemas, Programação, Design, Administração de Banco de Dados e Manutenção de Sistemas de Informação;

c) **Modernização Administrativa** - implementa novas técnicas de procedimentos, da desburocratização e do melhoramento dos serviços oferecidos pelas unidades da SEC, apoiando-se no uso eficiente de novas tecnologias.

Com o passar do tempo a revolução tecnológica mudou muito os paradigmas de gestão empresarial. A aceleração, inovação, automação dos processos e a velocidade das informações são fatores que estão impulsionando essa revolução. O novo desafio é descobrir o que as organizações devem fazer para potencializar seus recursos na nova sociedade da informação. Dessa forma, as organizações necessitam investir em meios que garantam a diferenciação no mercado, resolvendo e qualificando ações que resultarão na integração e comprometimento de todos os seus componentes, a fim de se obter a melhoria contínua. Dessa forma, ações foram planejadas e executadas pela SEC.

A SEC vem avançando no contexto tecnológico e procurando responder com presteza ao desafio de universalizar o atendimento aos seus usuários com efetiva melhoria na quantidade e na qualidade de serviços. Tem implantado nos diversos setores modernos equipamentos e desenvolvido sistemas que atendam às exigências da nova Administração Pública, que prima por uma disponibilização mais efetiva de serviços ao cidadão, como também, por ferramentas de controle de ações e de gastos governamentais mais eficazes, conforme dito por Almeida Junior (2005, p. 4).

Nos últimos anos, foram muitas as ações voltadas para implementação de sistemas, com o objetivo de otimizar e modernizar os seus procedimentos administrativos. Exemplificando algumas ações, a Organização estudada implantou o Sistema de Programação de Carga Horária, que informatiza e padroniza as práticas de execução, acompanhamento, avaliação e controle de folha de pagamento da Programação Escolar. Também implantou o Sistema de Controle de Processos, que informatiza as atividades de tramitação de Processos, desde a sua entrada, no protocolo até o seu término e arquivamento, o qual permite que o servidor acompanhe o seu processo através de consulta atualizada através da Internet. Além de ter desenvolvido o Sistema de Controle e Acompanhamento de Diárias e Adiantamento, o qual torna o gerenciamento do processo mais eficiente.

A SEC implementou o Projeto SECONLINE que consiste no sistema automatizado de administração de pessoal. Através do SECONLINE, o servidor poderá consultar seu histórico funcional, assim como, a administração de pessoal poderá conceder, de forma automática, sem a necessidade de volumosos

processos, direitos e vantagens dos servidores ativos do seu quadro efetivo, tais como: substituição, freqüência, gratificações, funções, licenças, adicionais, afastamentos, remoção, certidões, carga horária, ascensões e outros serviços.

Também foi implementado o sistema de publicação, ferramenta que auxilia as unidades administrativas na publicação dos seus Atos. De forma padronizada, os atos são gerados e publicados automaticamente no Diário Oficial. As informações publicadas alimentam o banco de dados da SEC promovendo maior confiabilidade e integridade das informações.

Já há alguns anos que a matrícula da rede estadual tem sido um grande sucesso, devido ao Sistema de Matrícula (SOMAR). Ele vem dando a possibilidade de que a matrícula dos alunos seja feita nos postos, facilitando o processo de matrícula para a rede estadual.

No ano passado foi implantando o sistema Transparência da Escola, que veio para informar como estão sendo gastos os recursos disponibilizados para as escolas pelo diretor. Ele fornece um extrato disponibilizado no site da SEC com as receitas e despesas realizadas pela unidade escolar.

Preocupando-se com o crescimento das atividades, a CMO está implantando uma Metodologia de Desenvolvimento de Sistemas (MDS), a qual irá orientar a SEC, parceiros e terceiros a desenvolver sistemas dentro de determinados padrões. Assim, buscando meios mais eficazes de gerenciar e manter sistemas desenvolvidos na organização, aumentando assim, a qualidade e reduzindo os custos de desenvolvimento. Essa metodologia vem com o passar do tempo sofrendo melhorias continuas. Hoje ela é conhecida como Processo de Desenvolvimento de Sistemas (PDS), com aplicação de alguns dos conceitos de CMMI, MPS-BR e PMBOK. Conforme a figura 9 a atual metodologia é formada por oito etapas:

a) **Iniciação** - que tem por objetivo descrever a solicitação de projeto e selecionar o projeto que deverá ser iniciado;
b) **Planejamento do Projeto** - que tem como objetivo descrever as atividades necessárias para a realização de um bom planejamento de projeto e deve ser elaborado de forma rápida, objetiva e completa, onde

uma boa prática para a execução do planejamento do projeto e seu acompanhamento é seguir o *Project Management Institute* (PMI, 2004);

c) **Análise de Processo de Negócio** - que consiste em levantar a lógica dos processos atuais, além de formulários, relatórios, documentos, volumes, formas de cálculo, dentre outros, ou seja, tudo que esclareça o funcionamento dos processos de trabalho;
d) **Análise e Projeto de Sistemas** - é a fase em que o sistema passa a ser modelado e projetado;
e) **Construção** - que tem a finalidade de codificar o sistema através dos artefatos gerados na fase anterior;
f) **Homologação** - que tem o objetivo de fornecer o aceite do produto construído, onde são feitos testes para garantir que os processos de negócio e o sistema funcionem de acordo com o que foi requisitado pelo cliente;
g) **Implantação** - que tem como objetivo preparar a infra-estrutura e os treinamentos necessários para implantar o produto;
h) **Encerramento** - que tem como finalidade avaliar, medir e dar por encerrado o projeto.

Atualmente, a CMO se lançou na tarefa de revitalizar e fortalecer a Área de Modernização Administrativa, com o objetivo de promover novas técnicas de procedimentos, desburocratização, otimização de processos e o melhoramento dos serviços oferecidos pelas unidades da Organização, apoiando-se no redesenho de processos, trabalhos de conscientização e treinamento de pessoal e no uso eficiente de novas tecnologias. A equipe de desenvolvimento de projetos vem trabalhando nos seguintes projetos: Sistema do IAT, que tem a finalidade de gerenciar a logística dos eventos realizados pelo Instituto Anísio Teixeira. Gestão TOPA, que tem o objetivo de gerir os dados do Todos pela Alfabetização (TOPA). Gestão da Matrícula, com a finalidade de fornecer os requisitos para possibilitar a realização da matrícula informatizada. Almoxarifado, cujo foco é gerenciar o almoxarifado da SEC. Contratos e Convênios, cujo escopo é controlar e acompanhar os convênios e contratos firmados entre a SEC e os partícipes; bem como a implantação da nova arquitetura padronizada, a qual será mais detalhada na seção 4.6.

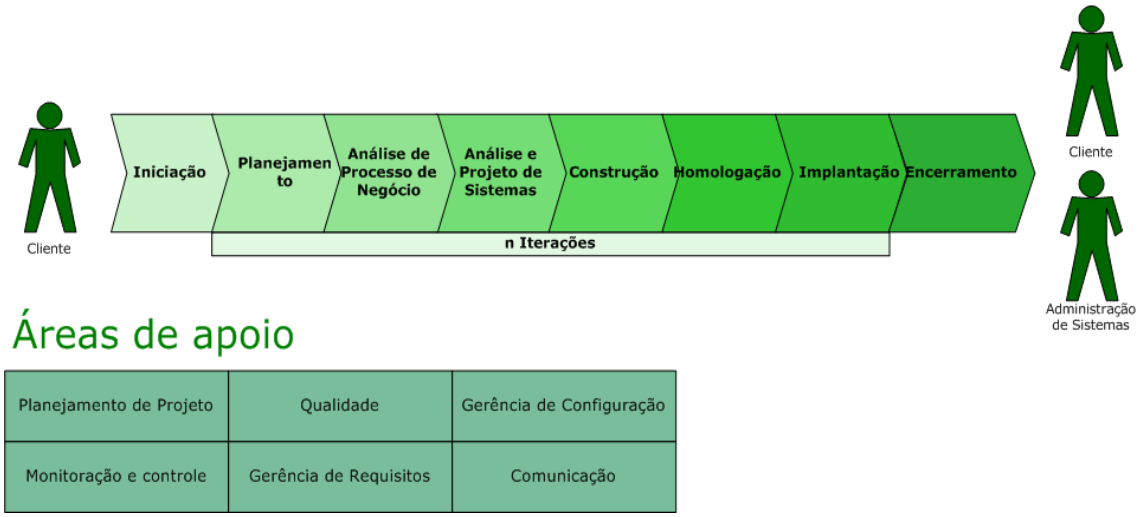

Figura 9 – Processo de Desenvolvimento de Sistemas

3.2 ARQUITETURA LEGADA

Os sistemas legados não podem ser meramente considerados como sistemas antigos. Eles incluem *software, hardware*, dados e processos corporativos e qualquer alteração em uma parte do sistema, provavelmente, envolve modificações em outras partes ou componentes.

No capítulo 1 foram abordadas as diversas arquiteturas que compõe os parques tecnológicos dos sistemas da organização estudada. A figura 10 mostra o atual panorama complexo e custoso para integrar as informações solicitadas pelos processos de negócio. Em alguns casos existem até redundância de funcionalidade não gerenciada, dificuldando as respostas ágeis que as organizações necessitam para aumentar sua competitividade ou melhorar o atendimento à sociedade como a instituição base para este trabalho.

Como podemos observar também na figura 10, os bancos de dados não estão integrados e os requisitos de negócio e/ou funcionalidades estão espalhados com suas informações distribuidas. Com isso, o problema de integridade, associado à falta de gerência e redundância dos dados, cresce exponencialmente à medida que surgiram novas aplicações.

A industria de TI, com o passar do anos, vem procurando evoluir novos padrões de integração entre as diversas funcionalidades das aplicações e as

organizações. Porém, essas instituições envolvidas não podem abandonar seus investimentos em tecnologia e nos seus atuais sistemas. A necessidade de integrar os *softwares* legados com as novas aplicações que surgem tem levado à criação de novos produtos e padrões com o intuito de permitir que aplicações existentes trabalhem juntas com as novas de uma forma fracamente acoplada. Para possibilitar essa flexibilidade entre a troca de dados das aplicações é necessário a criação de um padrão independente de plataforma (BECKER; CLARO; SOBRAL, 2002, p. 6); (MARTINS, 2005, p. 69). A arquitetura proposta viabiliza essa integração através dos Web Services dentro da arquitetura SOA, que será mais detalhada neste capítulo.

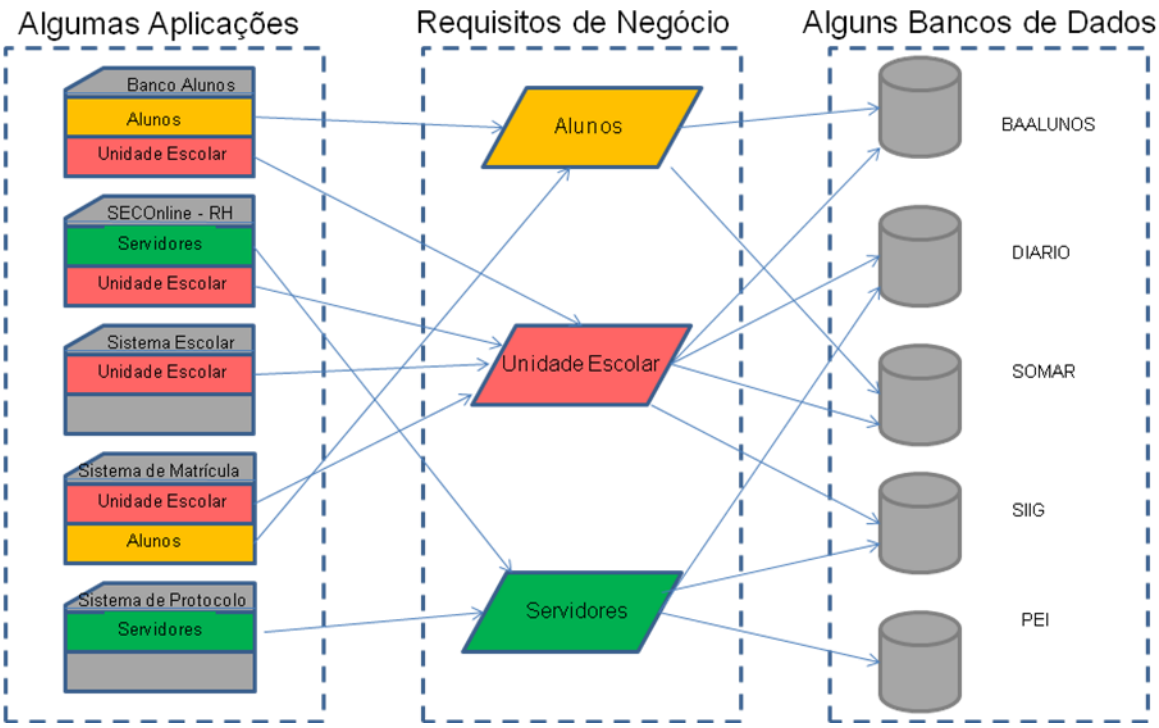

Figura 10 – Aplicações Legadas na SEC

Atualmente, a equipe que desenvolve os *software* da SEC trabalha em alguns projetos com funcionalidades que podem ser caracterizadas como requisitos compartilhados, ou seja, houve necessidade de troca de informações entre eles e com os legados existentes, como já foi relatado no capítulo 1. Com isso, fica cada vez mais evidente a necessidade de viabilizar uma nova arquitetura integrada para manter e desenvolver os sistemas da organização estudada.

3.3 ENTERPRISE APPLICATION INTEGRATION (EAI)

No passado foi cogitada a possibilidade de fazer uso da solução de *Enterprise Application Integration (EAI)* na SEC. Porém, após um estudo sobre essa arquitetura ficou evidenciado que os projetos de EAI sofrem com seus altos custos, ciclos de vida longos e uma alta taxa de fracasso, conforme dito por Pulier, Taylor, Gaffney (2008, p. 67) e Cummins (2002, p. 93). Os padrões de EAI são bastante abstratos para serem aplicados com a maioria das tecnologias de integração, mas específicos o suficiente para prover um guia ou catálogo para projetistas e arquitetos de *softwares*. Porém, as soluções tradicionais propostas pela EAI focam em uma máquina de integração centralizada e monolítica, que usa tecnologias proprietárias para integrar sistemas, e adaptadores especializados para conectar fontes de dados e sistemas legados. Outro aspecto são os altos investimentos iniciais solicitados por essas soluções tradicionais de EAI e a forte dependência dos fornecedores.

Por outro lado, Cunha (2002), Gomes e outros (2007, p. 4) disse que a abordagem tradicional de EAI não leva em consideração o atual dinamismo que a internet submete às organizações, na qual os sistemas de uma organização não podem ser isolados do resto do mundo. Neste âmbito, os requisitos de integração se modificam constantimente. As soluções tradicionais e proprietárias se tornam pouco ágeis e flexiveis e altamente custosas mediante qualquer alteração da demanda.

Em suma, pelo alto custo, ciclos de vida longos e dependência de fornecedores a *Enterprise Application Integration* (EAI) não funciona de maneira adequada para a SEC.

3.4 DESENVOLVIMENTO ORIENTADO A COMPONENTES

Nas organizações até pouco tempo os sistemas eram projetados de maneira isolada, tipicamente fechada, e, caso houvesse a necessidade de alguma extensão era preciso criar uma outra aplicação e compô-la de maneira que ficasse transparente para os usuários. Esta outra aplicação é realmente um *software* de fato independente. Um conjunto de aplicações independentes pode ser visto como uma única aplicação, onde o elo entre elas é geralmente feito pelo *menu* (COSTA; CARVALHO, 2007, p. 4).

O desenvolvimento orientado a componentes disponibiliza uma maior flexibilidade, fortalecendo a separação bem definida de algumas partes da aplicação, componentes, através de interfaces bem definidas. As principais barreiras para essa flexibilidade corporativa são o tempo e o custo para desenvolver aplicações de negócio ou modificar as já existentes. As aplicações tradicionais tendem a ser enormes e monoliticas, dificultando sua adaptação a novos requisitos, especialmente quando há modificações nos processos de negócios da organização (CUMMINS, 2002, p. 278).

Os componentes forçam uma estrutura de sistema bem-organizada. Dentro de um único sistema, muito provavelmente a funcionalidade comum será consolidade e encapsulada para que seja mais simples implementar as modificações exigidas pelos processo de negócios da organização. Com a reusabilidade proposta pelo desenvolvimento orientado a componentes pode-se obter uma melhor economia e maior flexibilidade.

Na figura 11 é ilustrado um exemplo nos quais alguns componentes são reutilizados por diversas aplicações corporativas. Como por exemplo, o componente *Pessoa* é reusado por três aplicações (G-SEC, SIAT e SIESC). Já a aplicação SIESC consome os componentes *Aluno* e *Pessoa*.

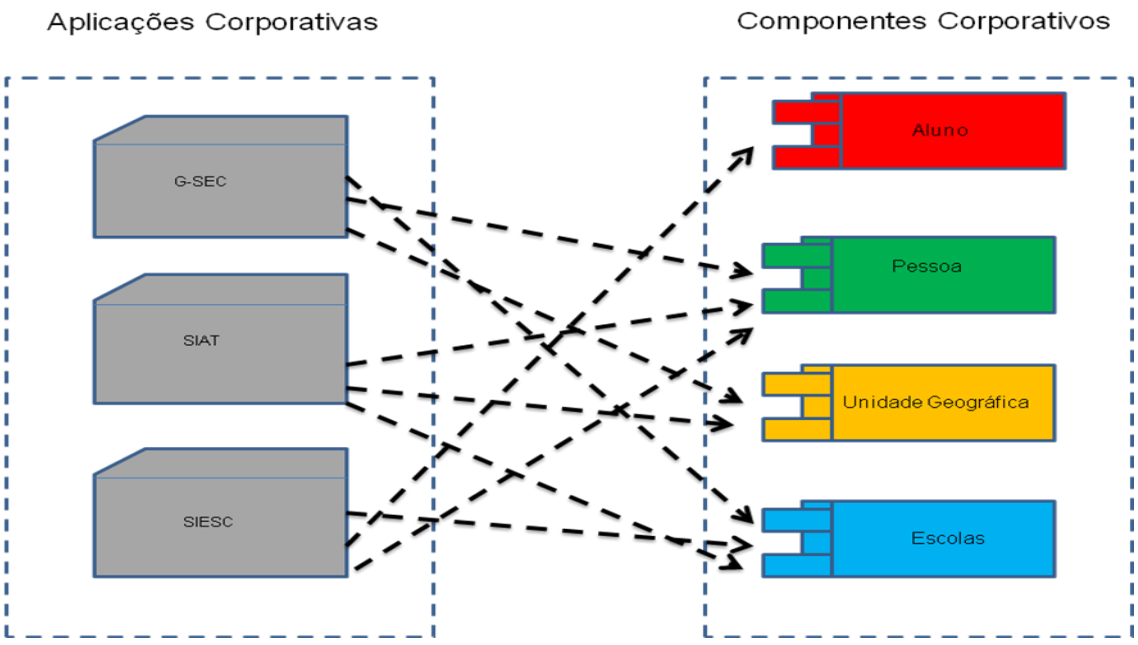

Figura 11 – Reuso dos componentes pelas aplicações

Anteriormente, neste capítulo foi relatado a variedade de arquiteturas de *softwares* que compõem parte de sistemas da organização estudada. Para minimizar esse quadro a equipe de desenvolvimento vem propondo que as novas aplicações sejam desenvolvidas com uma única arquitetura e nela sejam aplicados os conceitos de desenvolvimento baseados em componentes, como é mostrado pela figura 11, onde as aplicações são separadas por regras de negócios com o foco dos objetivos organizacionais da instituição. Com isso, poderemos reduzir o espalhamento dos requisitos que o atual parque de sistemas da organização evidencia.

3.5 FATOR MOTIVACIONAL

Na organização pesquisada para a realização deste trabalho, Secretaria da Educação do Estado da Bahia, com o passar do tempo o Sistema de Informação foi crescendo vertiginosamente, trazendo arquiteturas de sistemas cada vez mais complexas, robustas e bem heterogêneas. E ao mesmo tempo, um aumento da necessidade de fornecer respostas ágeis em relação a evolução dos seus negócios.

Segundo Pulier, Taylor e Gaffney (2008, p. 18) dentro de uma instituição pública ou privada, geralmente, as aplicações de *softwares* especializados são introduzidas por meio de esforços feitos na própria organização, da introdução de pacotes de *software* de terceiros, ou ambos. Essas aplicações rapidamente se tornam independentes, necessitando umas das outras para completar processos de negócios básicos, tais como, matricular um aluno na rede estadual de ensino. De fato, a otimização dos processos de negócio das organizações públicas ou privadas demandam rapidamente interação não somente entre os sistemas da própria organização, mas também com sistemas de outras instituições. Embora esse tipo de integração tenha sido desejada há muito tempo pela industria de TI, os custos e complexidades envolvidos resultam em dados armazenados de forma isolados e em alguns casos surgem redundâncias não controladas. Com isso, os dados só podem ser conectados por meio de um custos e esforços enormes.

Atualmente, a SEC está enfrentando exatamente o problema de integração das informações entre as diversas arquiteturas legadas. Com o intuito de tentar amenizar o atual quadro em que a organização em questão se encontra, surgiu o

fator que motivou a propor uma arquitetura para solucionar esse tipo de problema através da Arquitetura Orientada a Serviço (SOA) com a exposição dos serviços com a tecnologia *Web Services*.

3.6 DETALHES DA ARQUITETURA PARA INTEGRAÇÃO DE SISTEMAS

O presente estudo propõe viabilizar a implantação de modelo uma arquitetura de software em organizações públicas cujo existem diversos tipos de arquitetura em seus sistemas. Esse modelo proposto é baseado em SOA com o uso de *Web Services* para integrar os sistemas legados com as novas aplicações através da exposição de serviços.

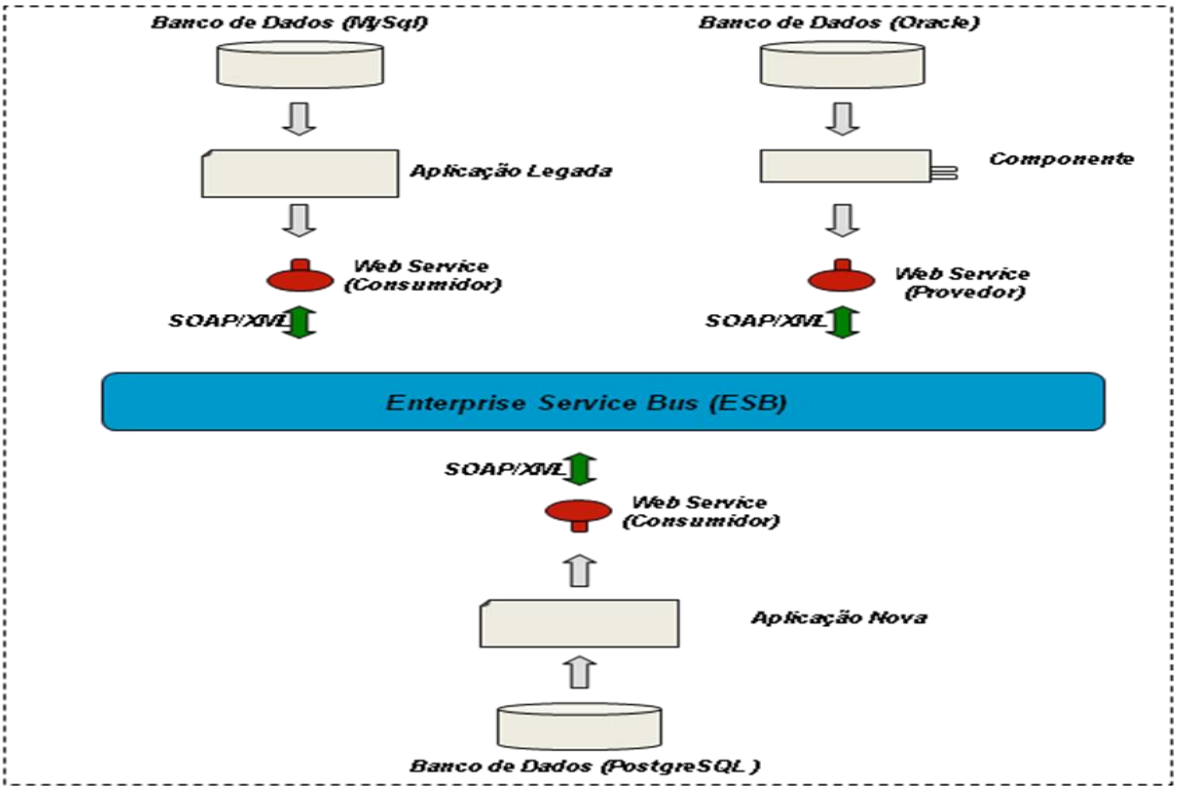

Figura 12 – Visão Geral

A arquitetura proposta foi idealizada para possibilitar a integração entre as diversas arquiteturas encontradas na TI da organização estudada. Na figura 12 é ilustrada essa arquitetura que é composta por diversos tipos de servidores de aplicação com seus respectivos bancos de dados, onde são oferecidos servicos através de *Web Services* e as trocas de mensagens entre os consumidores e

provedores de serviços são realizadas utilizando padrão SOAP com XML. Por fim, os padrões WSDL e UDDI servirão para possibilitar a descrição e localização dos serviços que são expostos pelas aplicações e/ou componentes através de um barramento de serviço.

Na figura 13 é ilustrado a localização de serviços. A “Aplicação Nova” possui um *Web Service Consumidor* que solicita informações ao *Web Service Provedor,* onde para saber a localização e a descrição do provedor o consumidor deve solicitar essa informação para o servidor de UDDI. Após saber a localização e a descrição, o consumidor, troca informações com o provedor através de SOAP baseado em XML.

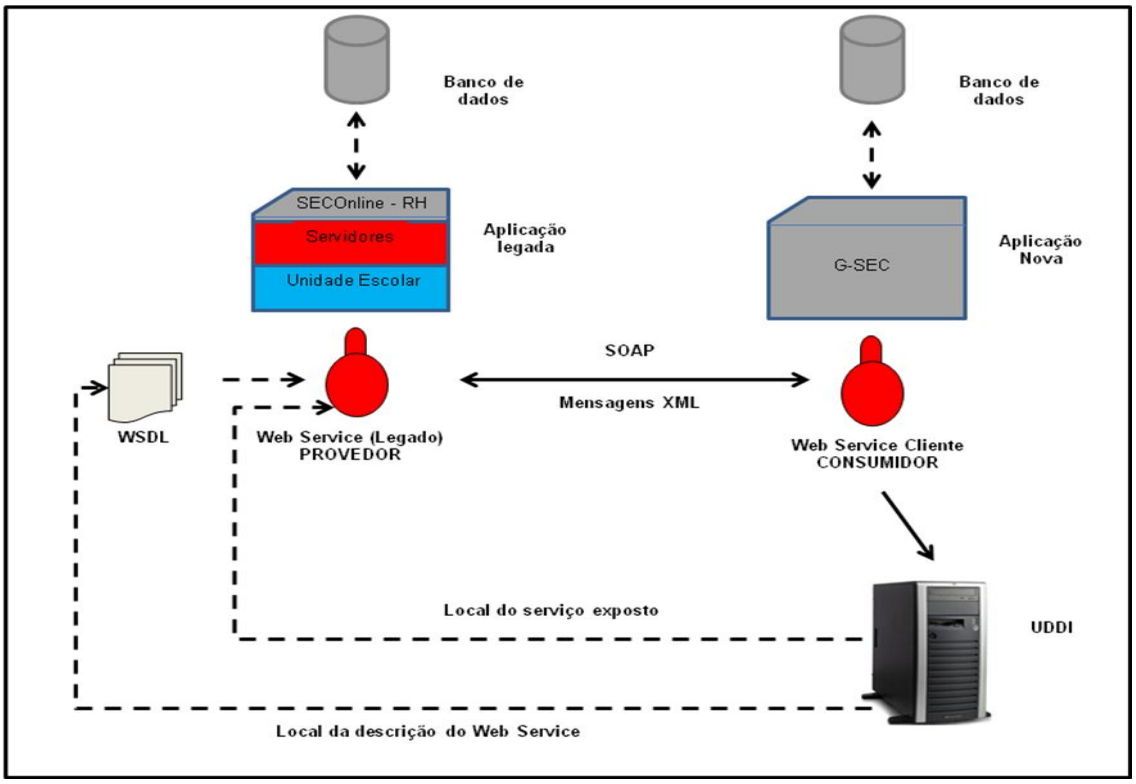

Figura 13 – Localização de serviço

Nos próximos ítens serão detalhados os aspetos da nova arquitetura proposta por este trabalho.

3.6.1 Serviços Expostos através de *Web Services*

As organizações estão gradativamente adotando a SOA com a finalidade de suportar suas aplicações legadas mais críticas. A SOA é considerada por Maciel

(2007, p. 24) como um paradigma computacional que surgiu para suprir a necessidade de colaboração e integração de componentes funcionais, disponibilizados nesse contexto como serviços, entre organizações ou dentro da própria instituição que expõe os serviços, de forma interoperável e com fraco acoplamento, como já foi mencionado no capítulo 3.

Já Barbieri (2001, p. 211) afirma que os *Web Services* são uma poderosa implementação da SOA, que podem ser basicamente definidos como interface para recursos computacionais, acessíveis através da troca de mensagens baseadas em XML e construídas com a utilização de padrões muito utilizados na internet. Em virtude disso e do que já foi relatado anteriormente sobre fraco acoplamento, viabilização de componentização dos processos de negócio da organização e compatibilização das diversas tecnologias e suas soluções não integráveis; os *Web Services* foram adotados como parte da arquitetura proposta para possibilitar uma melhora na integração entre as diversas arquiteturas do parque de sistemas encontradas durante este estudo.

3.6.2 Aplicações Novas

Atualmente, o setor responsável pela TI da SEC vem buscando fortalecer a utilização de uma única arquitetura para o desenvolvimento de sistemas. Portanto, seguindo essa premissa, foi proposta a utilização de uma única linguagem de programação, o JAVA. Seguindo com padronização no desenvolvimento de sistemas foi selecionado o framework *Jboss Seam*, que defende o conceito de integração entre vários *frameworks* específicos (FARLEY, 2008, p. 56).

As novas aplicações devem preferencialmente nascer utilizando o *framework Jboss Seam*. Porém, esses novos *softwares* produzidos serão, no futuro, considerados como aplicações legadas. Por isso, estamos propondo uma arquitetura para esses novos sistemas pensando nas possíveis integrações com outras aplicações legadas e novas também. Na figura 14 é mostrada a arquitetura proposta para o desenvolvimento de novas aplicações, dividida em três camadas:

a) **Apresentação** - nesta camada a camada responsável pela interface com o usuário, e tem o papel de expor os serviços das aplicações novas, criando os *Web Services* para esta finalidade;
b) **Negócio** - já nesta camada tem como foco a implementação das regras de negócio da aplicação;
c) **Persistência** - e por fim, a camada de persistência que tem a finalidade de persistir as informações no banco de dados e criar os *Web Services* para consumir os serviços expostos por outras aplicações.

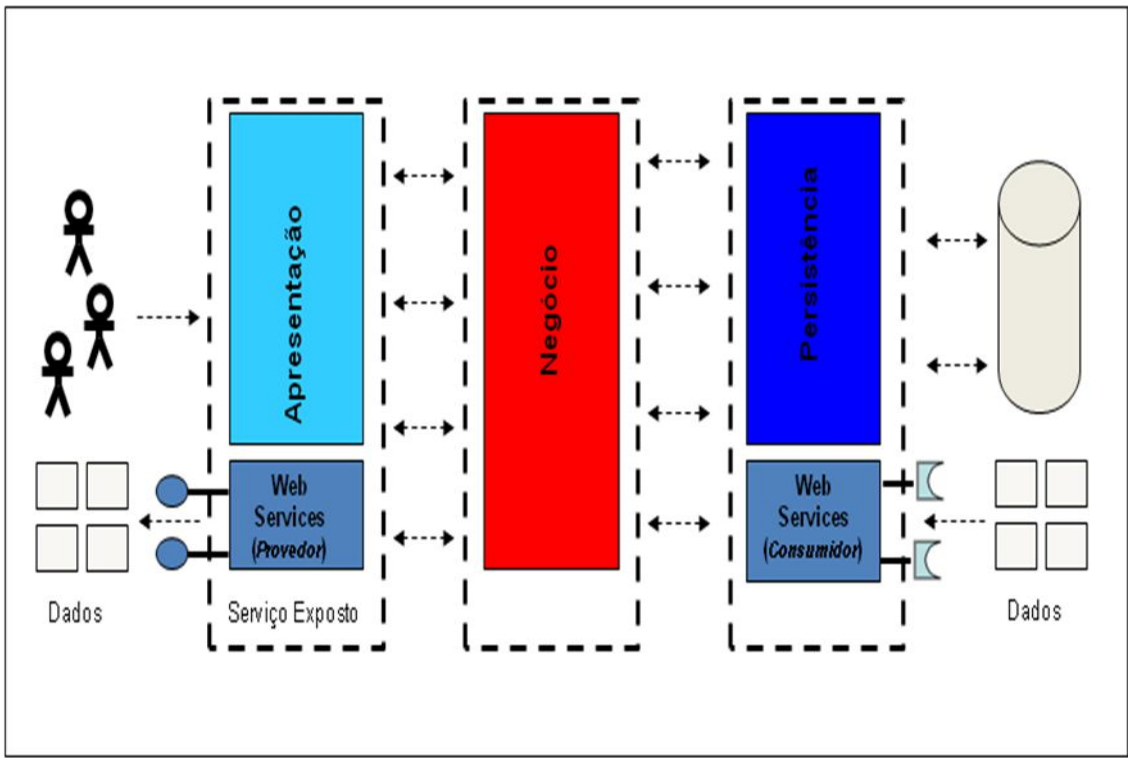

Figura 14 – Arquitetura de uma nova aplicação

O modelo em camadas proposto acima não é uma grande novidade no mundo da TI. Estas camadas já são bem conhecidas, o importante é a integração que ocorre através das camadas de apresentação e persistência. Na camada de apresentação residem os serviços expostos através de *Web Services*, ou seja, os provedores de serviços que serão consumidos por outras aplicações. Já na camada de persistência residem os consumidores dos serviços expostos por outras aplicações.

3.6.3 Aplicações Legadas

Nas aplicações novas podem haver consumidores e/ou provedores de serviços. Já para as aplicações legadas inicialmente serão criados apenas os serviços expostos (provedores). Porém, no futuro pode haver a necessidade de uma aplicação legada consumir um serviço exposto pela nova arquitetura proposta, em decorrência de um custo muito mais baixo para novas implementações de funcionalidades. Em virtude disso, podemos destacar a importância da flexibilide deste tipo de arquitetura e já podemos tanto criar um serviço exposto nas aplicações legadas, como nas novas que podem surgir, caracterizando o acoplamento fraco, conforme já relatado no capítulo 3 com base nos trabalhos de Sampaio (2006, p. 29) e Dantas (2008) Gomes e outros (2002, p. 2).

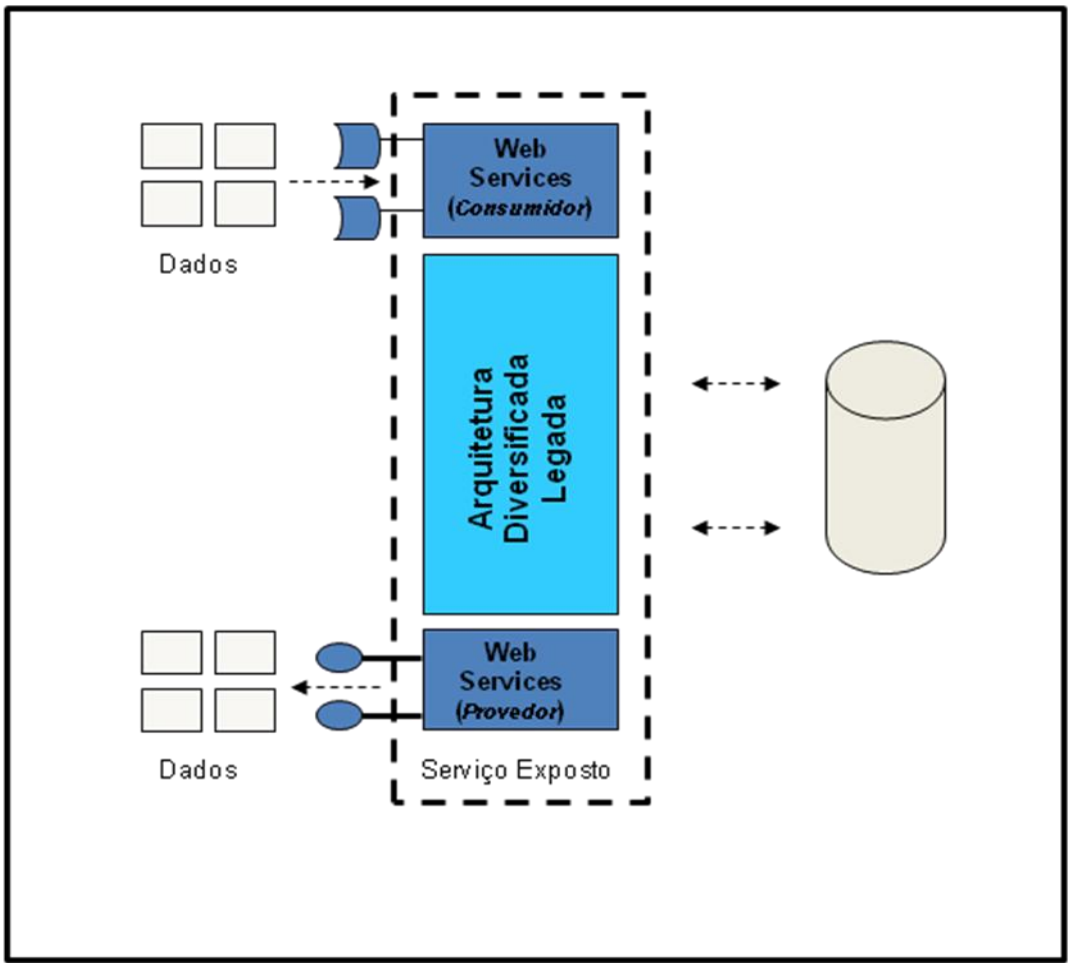

Figura 15 – Arquitetura das aplicações Legadas

Na organização estudada para a realização deste trabalho, as aplicações legadas possuem uma grande gama de funcionalidades já implementadas e em uso.

Na figura 15 podemos observar que nas aplicações legadas serão apenas implementados os serviços, sendo consumidor ou provedor. Com isso, geralmente, reduzimos os custos, que muitas vezes são onerosos e tornam as implementações das novas funcionalidades inviáveis.

No futuro, com o andamento da implantação e uso da arquitetura orientada a serviços iremos obter um conjunto de aplicações que serão interligadas através de seus serviços expostos, como mostrado na figura 16.

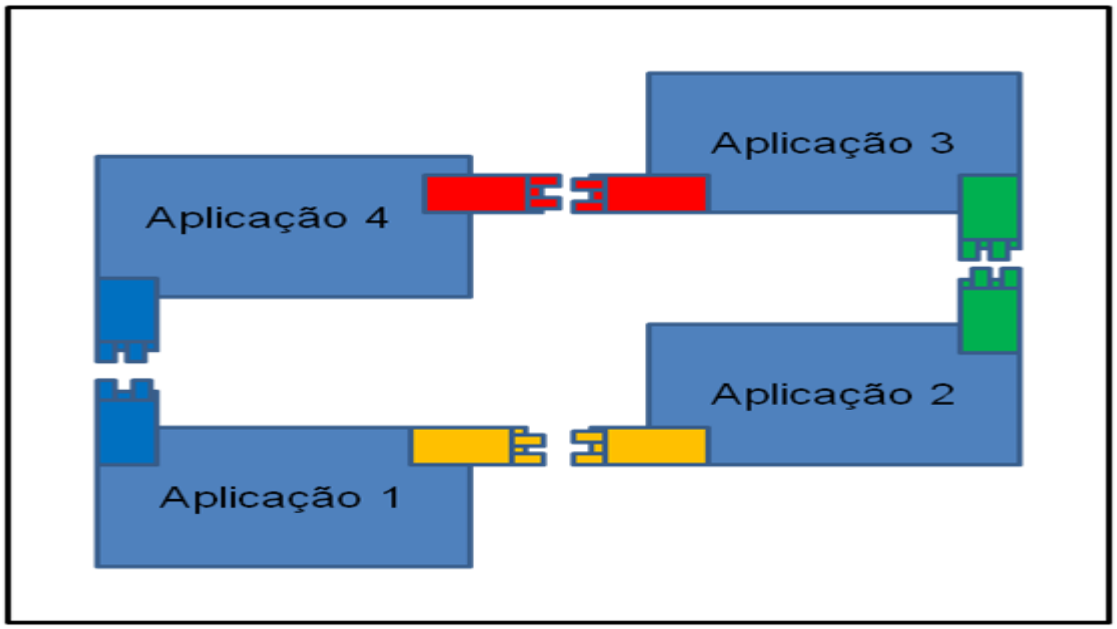

Figura 16 – Aplicações Ligadas pelos serviços

3.6.4 Protocolos utilizados

Anteriormente, no capítulo 3, foi relatado com base em Barbieri (2001, p. 189) que os *Web Services*, na teoria, podem funcionar apenas com o protocolo SOAP. Porém, os *Web Servies* trabalham mais eficazmente com o uso em conjunto dos protocolos WSDL, UDDI e SOAP. Em virtude da natureza aberta dos *Web Services*, é possível para muitos consumidores acessar o mesmo serviço exposto, independente do sistema operacional ou linguagem de programação, desde que o consumidor faça requisição de informação com a utilização do protoloco SOAP em um padrão adequado. Na figura 17 é mostrada a flexibilidade do uso do protocolo SOAP. Com base nesses aspectos, que esses protocolos foram escolhidos para faz

parte da arquitetura proposta por este trabalho com o intuito de viabilizar uma melhor integração entre as aplicações legadas e as novas.

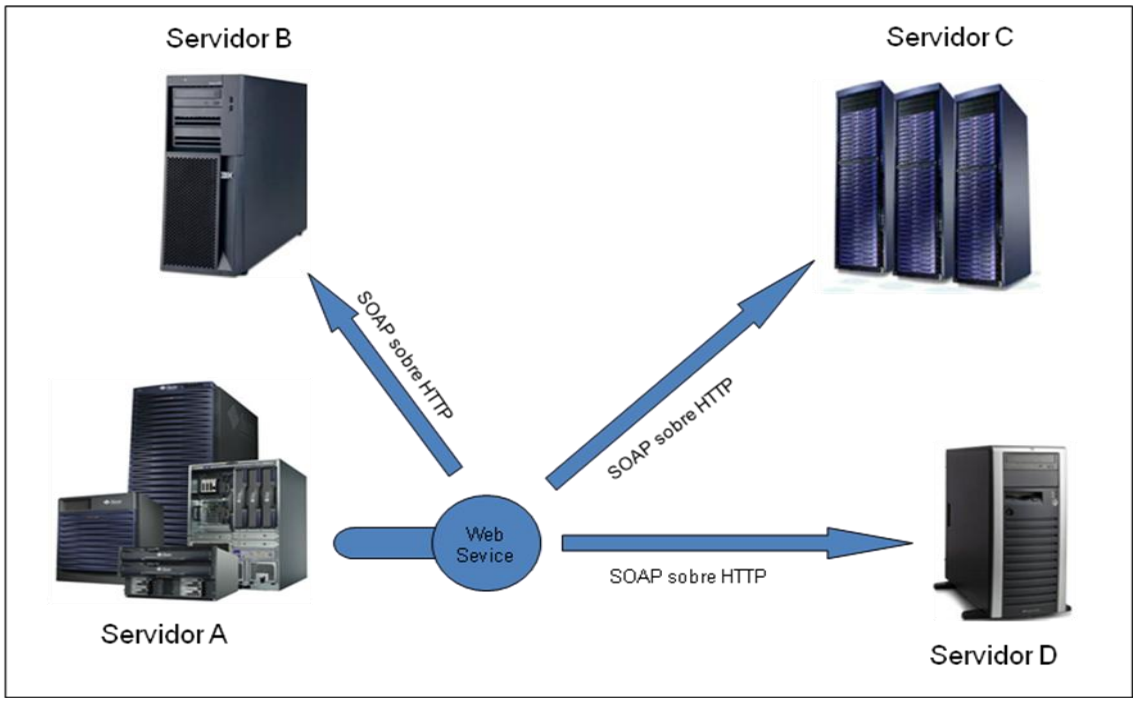

Figura 17 – Flexibilidade do protocolo SOAP.
Fonte: Martins (2005, p. 97).

3.6.5 Barramento de Serviços Corporativos

O uso do barramento de serviços corporativos (Enterprise Service Bus - ESB) facilita o gerenciamento do conjunto dos componentes coordenados que são expostos, de forma que ele tem a responsabilidade de gerenciar as conexões entre os serviços consumidores e provedores (COSTA; CARVALHO, 2007, p. 8). Dessa forma, em qualquer alteração na localização dos componentes provedores, os consumidores não sofrerão impacto, pois a interação não se dá entre os próprios componentes, e sim, através dos barramentos de serviços.

O ESB é caracterizado por ser a base dos serviços, mensagens, comunicações, tranformações e de segurança sobre a qual se pode plugar aplicações ou simplesmente interagir com elas. Ele segue os princípios de uma arquitetura orientada a serviços (SOA), possibilitando uma integração com diferentes tipos de serviços, fortalecendo a flexibilidade que a solução propõe (MARTINS, 2005, p. 98).

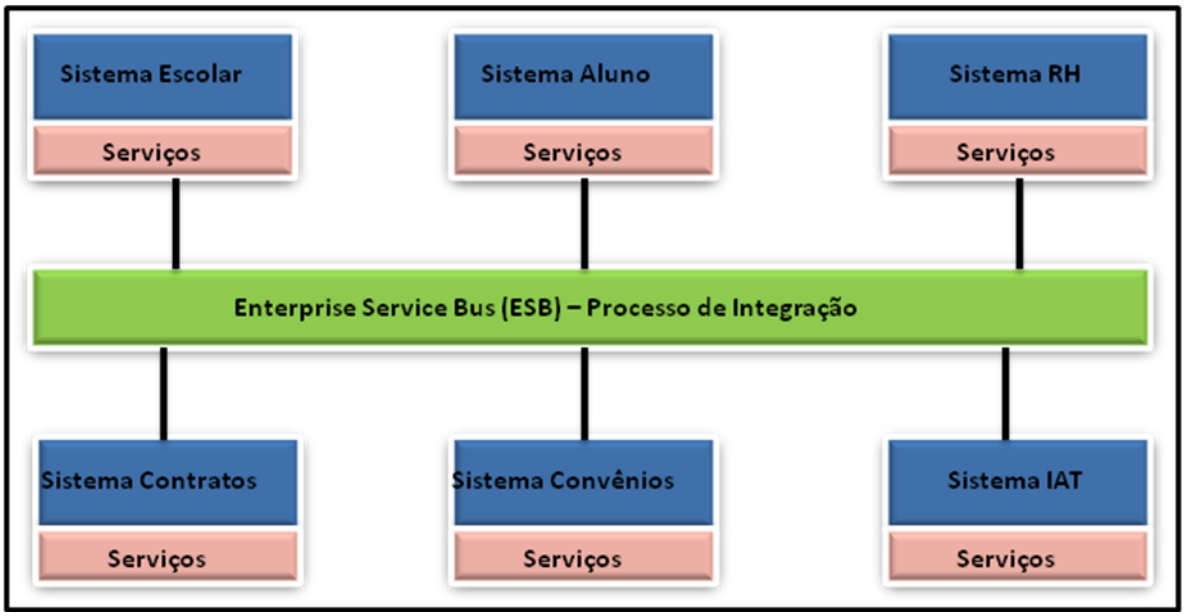

Figura 18 – Barramento de serviço proposto

Na figura 18 é ilustrado o ESB proposto pela arquitetura deste trabalho. Ele pode simplificar as integrações entre as diversas aplicações, onde antes era necessário conhecer a interface de cada diferente serviço exposto na sua arquitetura. Sendo assim, numa alteração em um serviço nas arquiteturas que não utilizam ESB, o impacto é fortemente encontrado em vários componentes. Com o advento do ESB, as aplicações só precisam conhecer uma interface (a do ESB). Todo conhecimento necessário dos padrões e tecnologias são de responsabilidade do ESB. Há um isolamento das aplicações e uma simplificação nas mudanças.

3.6.6 Segurança na Arquitetura Proposta

As tecnologias da arquitetura orientada a serviços ainda estão em fase de evolução. Elas promovem através de suas diretrizes a facilidade de comunicação entre serviços de diferentes aquiteturas, interligados por uma rede como a Internet. Considerando essa facilidade e as inovações nos processo de negócios, Rocha (2006, p. 42) afirmou que soluções de segurança são extremamente necessárias para atender os novos requisitos de segurança que surgem em virtude da arquitetura SOA.

Segundo Rocha (2006, p. 44), no momento em que os conceitos de SOA forem utilizados para expor serviços com base em *Web Services* via Internet,

utilizando os padrões abertos (XML, SOAP, WSDL, UDDI, etc), certamente será necessário prover novos mecanismos de segurança. Em *Web Services* podemos implementar a segurança com esses padrões abertos em dois níveis: mecanismos de segurança para o nível de transporte e mecanismos de segurança voltados para as mensagens. Cada um desses níveis de mecanismos de seguração dispõe de padrões e protocolos específicos com o objetivo de atender aos diferentes tipos de requisitos de segurança de cada negócio que a arquitetura SOA propõe e esses mecanismos podem ser implementados em conjunto ou separadamente.

Por outro lado, Rocha (2006, p. 49) e Pacheco (2000, p. 10) afirmaram que a segurança na arquitetura SOA é complexa e que grande parte das organizações que implementam a arquitetura orientada a serviços deixam para segundo plano os aspectos de segurança. A arquitetura proposta por este trabalho também deixou a segurança para um segundo passo. E também, inicialmente, os serviços serão expostos apenas dentro da própria instituição. Porém, os aspectos de segurança ficaram para ser viabilizados no futuro, como também, a exposição de serviços na internet.

3.6.7 Gerenciamento de mudanças

Durante o planejamento de um sistema é acordado o escopo do projeto. Porém, durante a sua execução podem ocorrer alterações de escopo. O cliente deve ficar ciente dos impactos da solicitação de mudança de requistitos. Há também, a possibilidade de alterações de requisitos que já estejam em produção. Na arquitetura proposta os serviços que já estão em uso pelos usuários ou já codificados e não implantados devem passar pelo procedimento de solicitação de mudanças. Esse procedimento é retratado pelos seguintes passos:

a) **Abrir a solicitação**, o usuário ou arquiteto responsável pelo serviço que será alterado solicita uma alteração através de um formulário que deve ser encaminhado para o grupo de arquitetos da instituição;
b) **Avaliação da solicitação**, o grupo de arquitetos recebe a solicitação e dá um parecer de atendimento da solicitação.

Execução e notificação da solicitação, o arquiteto executa a alteração do serviço e após a conclusão notifica ao usuário sobre a implantação do serviço alterado;

3.6.8 Mapeamento dos requisitos

Durante a implantação dos *cases* foi observado que podem ocorrer divergências entre os requisitos dos serviços e as funcionalidades legadas. Para resolver esse problema a equipe de arquitetos executa um questionário junto aos clientes dos serviços conflitantes com os questionamentos sobre a quantidade de requisitos atendidos pela funcionalidade legada e também a quantidade atendida pelo novo serviço. Após o resultado do questionário o grupo de arquitetos da organização faz uma avaliação e fornece ao analista de sistemas responsável pelo projeto um parecer sobre os requisitos que serão atendidos pelo novo serviço em relação aos requisitos que eram pela funcionalidade legada.

CAPÍTULO 4

4 *CASES* NA ORGANIZAÇÃO

Neste capítulo serão apresentados alguns exemplos que foram implementados fazendo uso da arquitetura proposta por este trabalho e os passos para a implementação da arquitetura proposta por este trabalho. Salientamos que os *cases* foram escolhidos pela equipe de desenvolvimento de sistemas da instituição estudada.

4.1 IMPLEMENTAÇÃO DA ARQUITETURA

Nos capítulos anteriores foram apresentados os conceitos e modelos relacionados as tecnologias e a arquitetura de integração proposta por este trabalho. Na organização estudada o ambiente arquitetural de softwware é muito heterogêneo, como já foi relatado anteriormente. Para a implementação dessa arquitetura devemos seguir os seguintes passos:

a) **Selecionar os primerios serviços** - incialmente, a equipe de arquitetos da organização seleciona as funcionalidades em comuns nas aplicações legadas e novas que são candidatas a serviço para criar um único ponto de entrada da informação;
b) **Mapear as funcionalidades** - logo após, realizamos o mapeamento das funcionalidades junto ao usuário responsável pelos dados, e realizamos o processo de transformação das regras de negócios em serviços;
c) **Codificar os provedores de serviços** - em seguida, os projetistas da organização inciam a codificação dos *Web Services* que serão os serviços mapeados anteriormente pelos arquitetos;
d) **Criar os WSDL** - o próximo passo, é criar o WSDL que servirá de guia referencial para os consumidores, em relação aos consumidores. Nele é necessário descrever todos os métodos que o serviço realizará;
e) **Implementar o ESB** - neste passo, devemos viabilizar o barramento de serviço corporativo através de alguma ferramenta, que tem a finalizade de gerenciar as conexões entre os serviços consumidores e provedores;

f) **Codificar os consumidores de serviços** - logo após, os projetistas implementam os consumidores
g) **Registrar os serviços** - em seguida, é necessário registrar os serviços criados no servidor de UDDI;
h) **Disponibilizar os serviços no ESB** - e por fim, disponibilizar os provedores e consumidores no barramento de serviços corporativo.

4.2 CASES E SERVIÇOS

Os serviços expostos foram projetados com o objetivo de minimizar os altos custos de manutenibilidade com as aplicações legadas, aumentar o reuso das informações corporativas e viabilizar uma melhora na integração entre as diversas aplicações com suas arquiteturas proprietárias.

Um dos grandes desafios encontrados durante o desenvolvimento de sistemas com base em componentes é justamente identificar que trechos de código poderão vir a ser componentizados. Possivelmente, os padrões de componentização podem facilitar a obtenção de uma uniformidade na construção dos componentes, facilitando assim, a integração e a manutenção. Na figura 19 está ilustrada a interação entre as aplicações e os componentes com seus serviços expostos. Nessa figura é observado o reuso da informação de Unidade Geográfica e Pessoa pelas aplicações G-SEC, Gestão TOPA, Almoxarifado e SIAT.

O serviço Unidade Geográfica fornece os dados geográficos com base nas regras e conceitos da organização estudada através do ESB. Por exemplo, a divisão geográfica dos municípios por DIREC (Diretoria Regional de Educação) e os dados de municípios da Bahia. Os dados de DIREC são consumidos pelo módulo de Contratos e Convênios da aplicação G-SEC. Já os dados de Municípios são consumidos pelos módulos Contratos e Convênios, Gestão TOPA e Almoxarifado da aplicação G-SEC e o módulo de Logística da aplicação SIAT. Já o serviço Pessoa que centraliza os dados dos servidores da SEC é consumido pelo módulo de Contratos e Convênios e Almoxarifado da aplicação G-SEC e o módulo de Logística da aplicação SIAT.

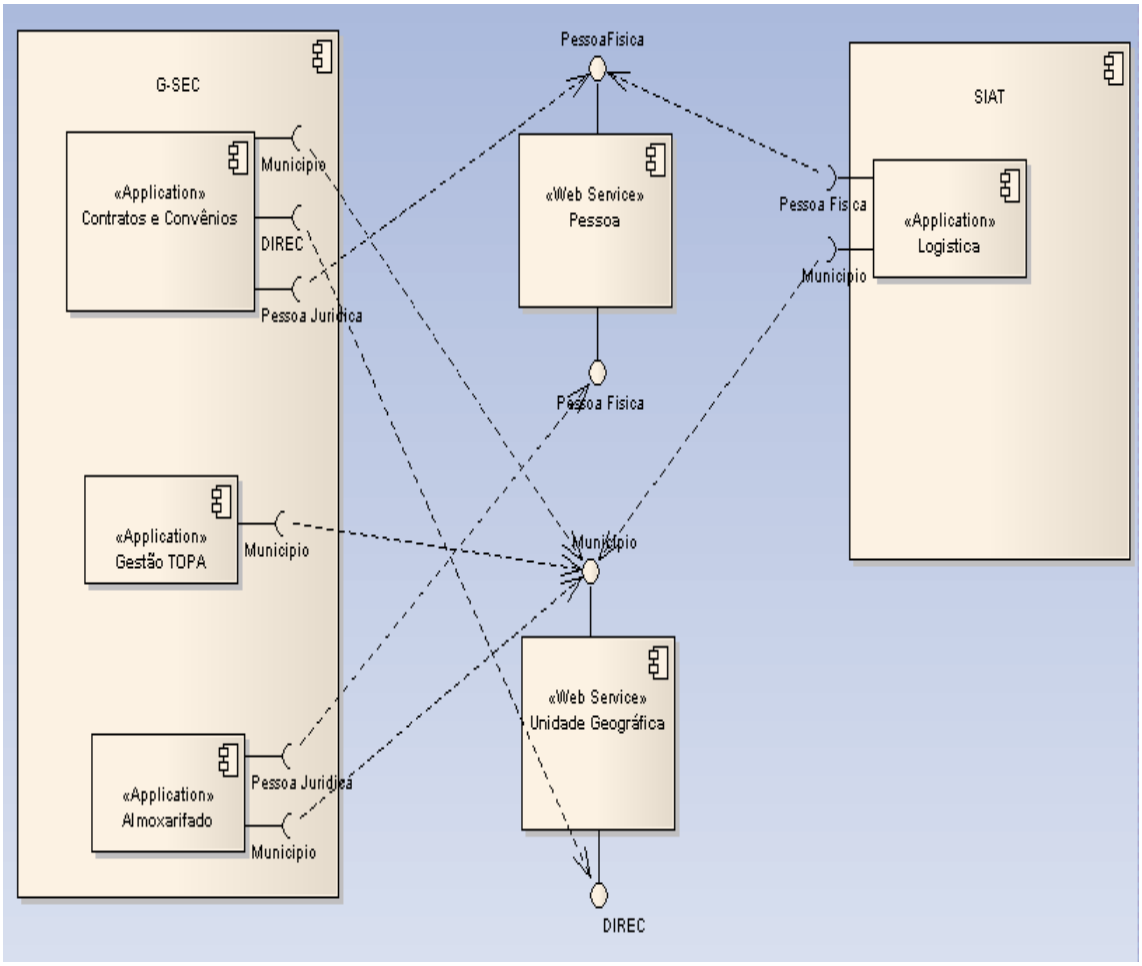

Figura 19 – interação das aplicações com os componentes

4.2.1 Unidade Geográfica

Com o passar do tempo e com a passagem de diversas empresas tercerizadas na SEC foram surgindo uma variedades de arquiteturas de *softwares*. Com isso, gerando uma série de problemas, como por exemplo, o espalhamento das informações de forma não controlada, que resultou no clássico problema de inconsistência dos dados, quando os mesmos eram solicitados pelos gestores. Um exemplo desse tipo de informação é a divisão geográfica do Estado da Bahia em relação à organização. Existem hoje diversos sistemas com as informações sobre Estado, Município, Território e Direc.

O serviço exposto de Unidade Geográfica veio para unificar as informações de Estado, Município, Território e Direc. Na figura 20 é mostrado o diagrama de classes de Unidade Geográfica.

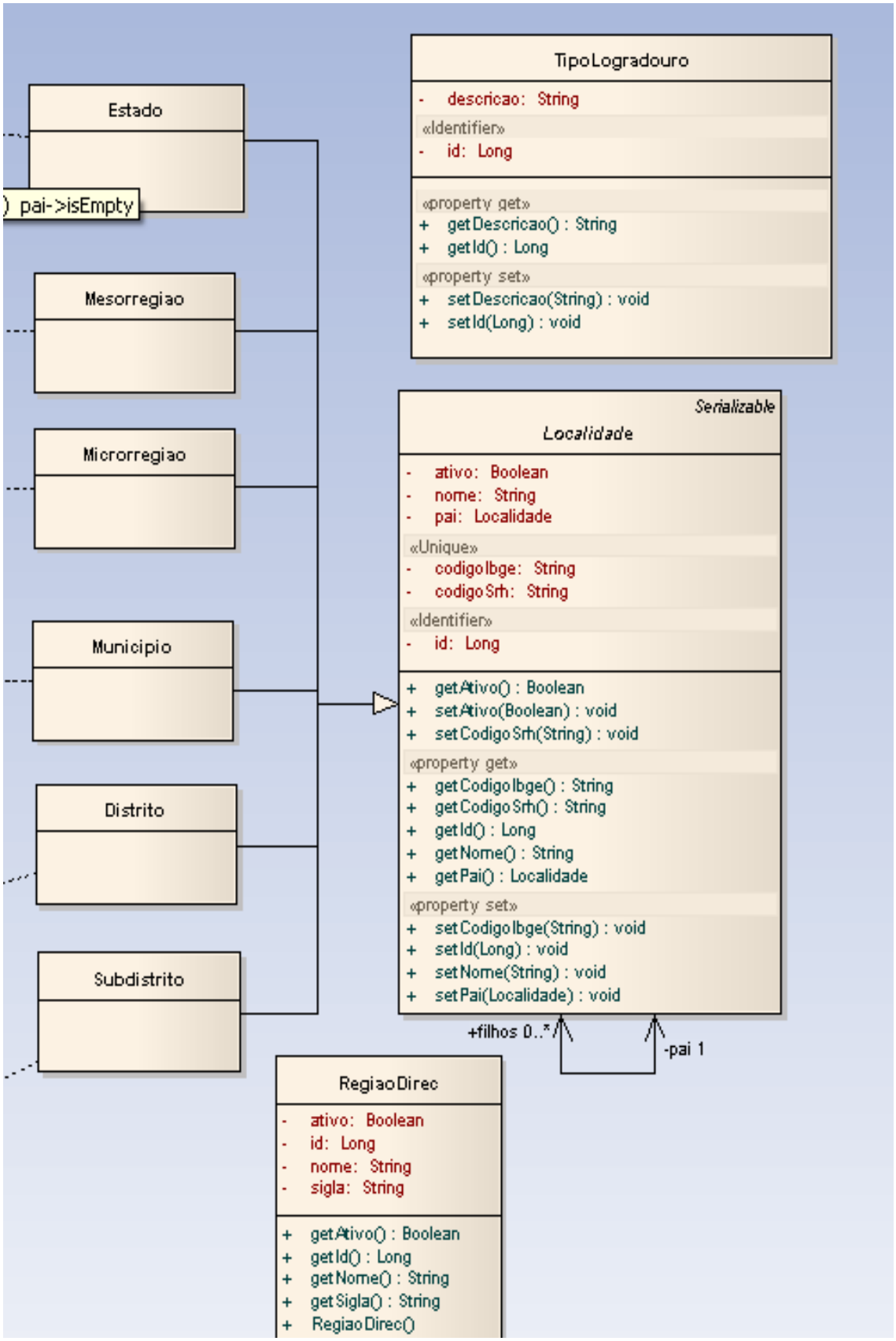

Figura 20 – O diagrama de classes de Unidade Geográfica

Atualmente, todos os sistemas em desenvolvimento ou a serem desenvolvidos, que necessitarem dessas informações, irão consumir o *Web Service* de Unidade Geográfica. Com isso, reduziremos o espelhamento das informações. Já as aplicações legadas que também necessitam fazer uso dessas informações serão gradativamente alteradas para consumir este serviço.

4.2.2 Pessoa

Outro serviço exposto é o Pessoa, cujo principal papel é fornecer informações sobre os servidores estatais. Atualmente, no parque computacional dos sistemas legados existe uma aplicação que gera todas as informações de recursos humanos dos servidores da SEC. Para uma melhor integração entre essa aplicação e as demais que estão sendo construídas foi selecionado esse serviço com o objetivo de fornecer tais informações através de *Web Services*. Por exemplo, uma aplicação necessita das informações de endereço, situação funcional, cargo, o valor de uma determinada gratificação, dentre outros, dos servidores estatais. Esse serviço é responsável por disponibilizar esse tipo de informação.

Na figura 20b é ilustrado o digrama de classes do serviço pessoa. Nesse modelo podemos observar que existe uma entidade denominada *Pessoa*, que representa a generialização de Pessoa e as entidades PessoaFisica e PessoaJuridica que representão as especializações de Pessoa.

4.2.3 G-SEC

Já mencionamos alguns dos *cases* que estão expondo serviços, como Unidade Geográfica e Pessoa. Hoje a aplicação G-SEC é a grande consumidora dos atuais serviços que estão sendo expostos com a adoção da arquitetura SOA. Essa aplicação está sendo concebida com base no conceito de componentização, onde a cada nova funcionalidade idealizada pelos usuários, é analisada pela equipe de desenvolvimento para deterninar a possibilidade de reusar algum componente. Hoje ela já possui os módulos de Almoxarifado, Contratos e Convênios e Pessoa. Ela já está consumindo os serviços de Unidade Geográfica e Pessoa mencionados neste capítulo.

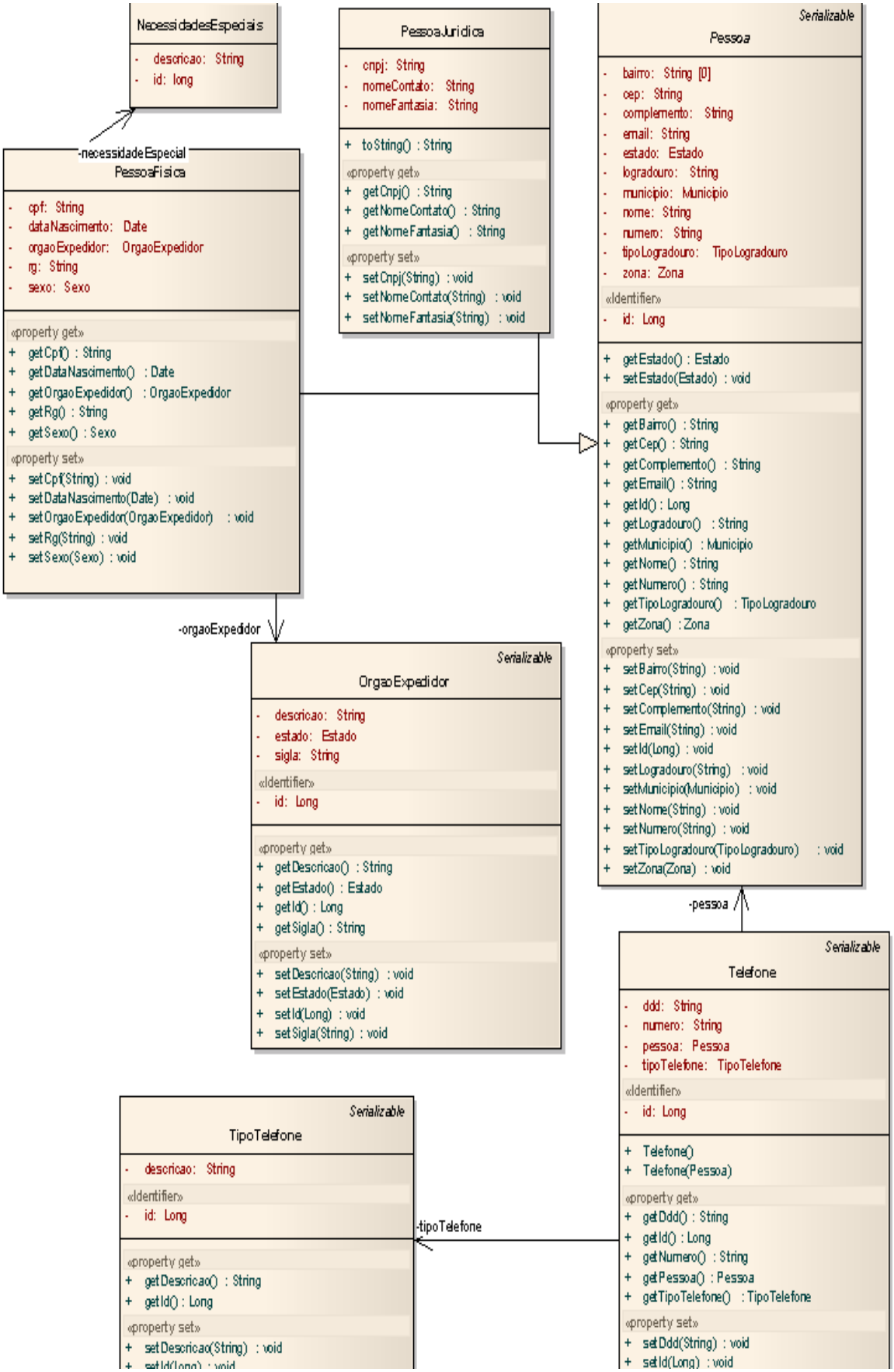

Figura 20b – O diagrama de classes de Pessoa

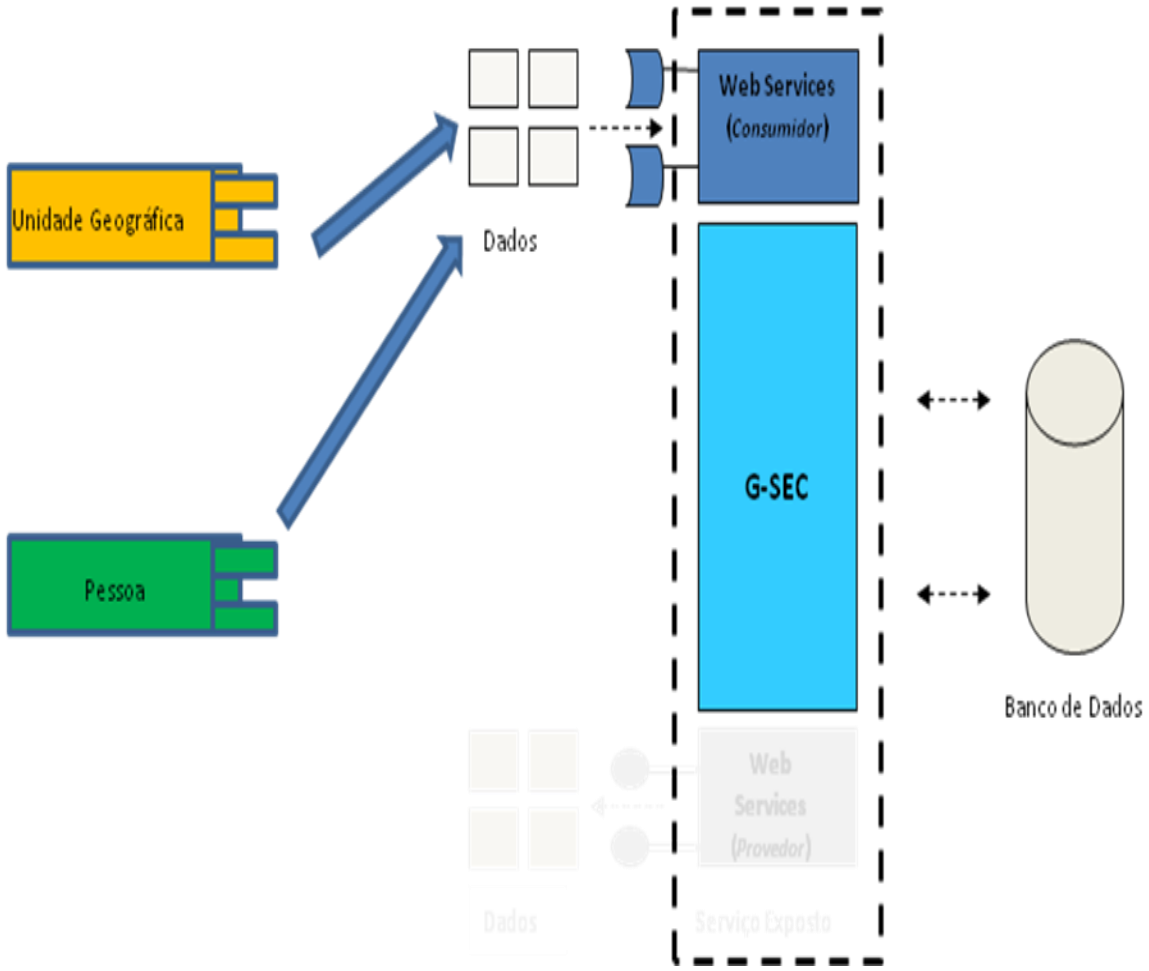

Figura 21 – Relação do G-SEC com Unidade Geográfica e Pessoa

Na figura 21 é ilustrada a relação da aplicação G-SEC com os componentes Unidade Geográfica e Pessoa. Nela os componentes fornecem dados para a aplicação G-SEC através de um Web Service.

Com o passar do tempo a aplicação corporativa G-SEC tende a crescer bastante e fortalecer o reuso das informações através dos serviços expostos. Segue a descrição dos módulos do G-SEC que fazem uso dos serviços pela arquitetura proposta.

4.2.4 Contratos e Convênios

O sistema Contratos e Convênios é mais um módulo do G-SEC, com a finalidade básica de gerir os contratos e convênios firmados entre SEC e os demais participantes, projetos e/ou programas solicitados pelo governo. Ele nasceu para suprir a necessidade da SEC de ter uma ferramenta e processos adequados para

um eficiente gerenciamento dos seus contratos e convênios, objetivando melhor acompanhamento dos mesmos. Anteriormente, a SEC encontrava-se em dificuldades para fazer a gestão dos seus contratos e convênios em virtude do volume de dados a serem controlados e da carência de uma ferramenta adequadamente informatizada, para apoio à gestão e processos adequados para melhor controle dos dados. Ele tem como requisitos principais o registro do contrato/convênio, acompanhamento físico e financeiro, alteração do contrato/convênio (aditamento, prorrogação, renovação e apostila), administração dos tipos de contrato/convênio e relatórios e distrato (quebra de um contrato) dos mesmos. Na figura 22 ilustra uma parte do diagrama de caso de uso da referida aplicação com um destaque aos relacionados com os serviços expostos. No caso de uso *Cadastrar Unidade Orçamentária* tem uma regra que descreve a relação com o serviços de *Unidade Geográfica*. Já no caso de *Cadastrar Dados Participe* faz referência ao serviços exposto *Pessoa*.

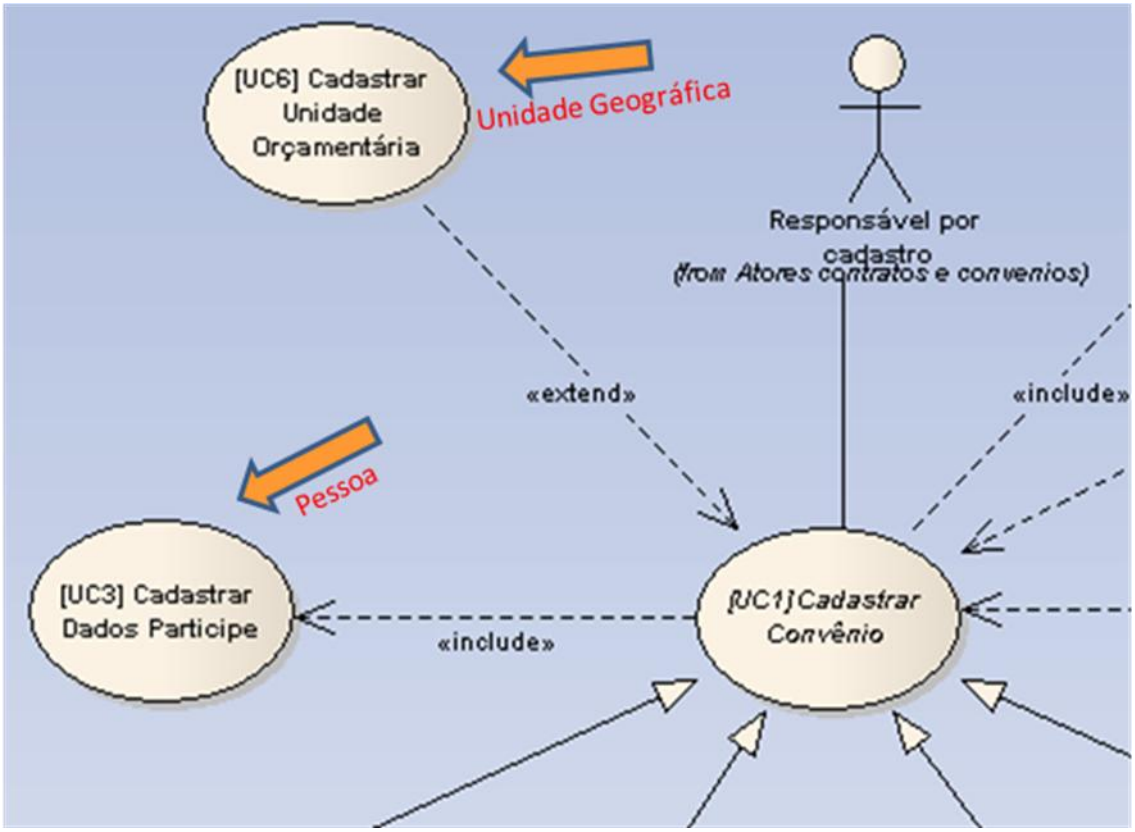

Figura 22 – Parte dos casos de uso do projeto Contratos e Convênios

Os serviços expostos através de *Web Services*, como proposto neste trabalho, podem ser consumidos por qualquer aplicação que faça parte da

arquitetura na organização. Na figura 23 também foi destacada a classe que consome o serviço exposto pelo componente de Unidade Geográfica. Já na figura 23b é ilustrada a

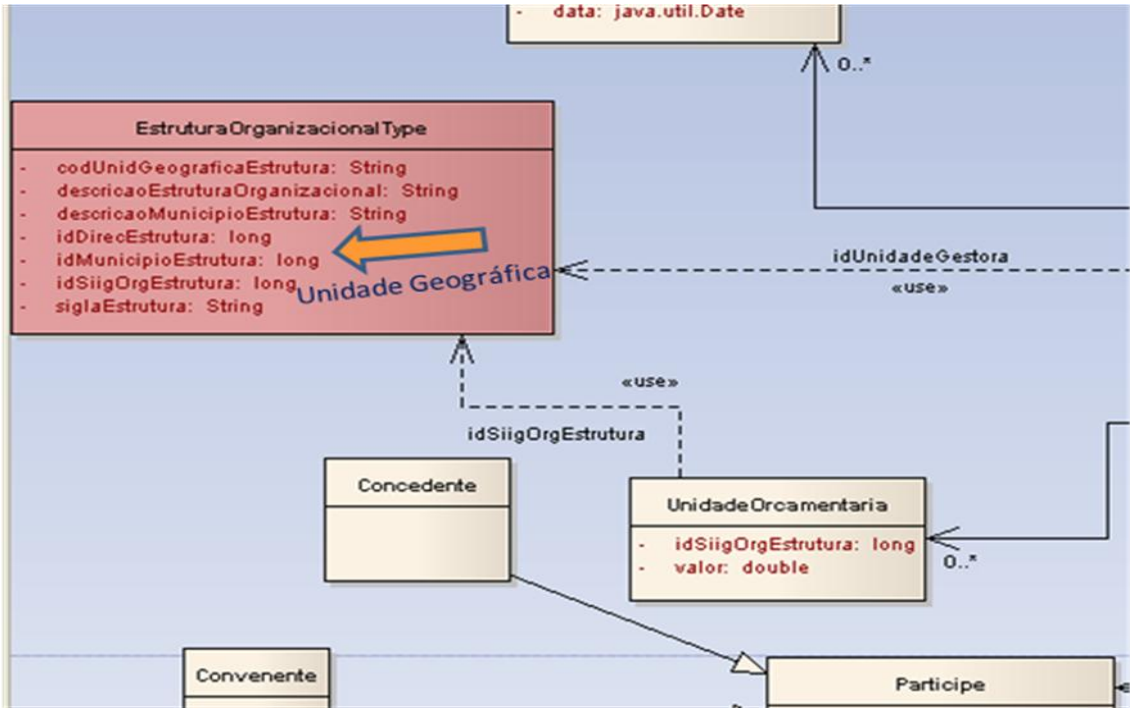

Figura 23 – Entidade de Contratos e Convênios destacada em relação ao serviço *Unidade Geográfica*

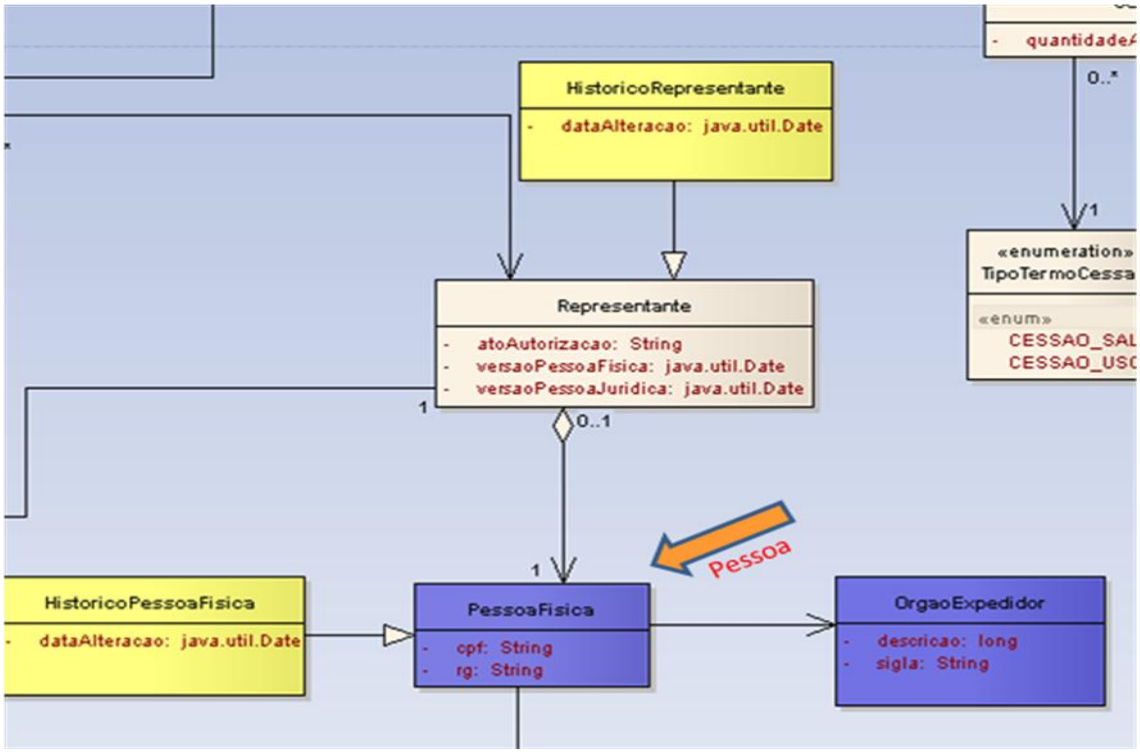

Figura 23b – Entidade de Contratos e Convênios destacada em relação ao serviço *Pessoa*

4.2.5 Almoxarifado

O módulo de Almoxarifado tem como finalidade gerenciar o almoxarifado escolar da SUPEC/DISUP integrando à Coordenação de Suprimento Escolar (CSE) com seus almoxarifados ou centros de distribuição (CD). Ele veio para melhorar seus controles e processos do almoxarifado. O sistema está disponível para outros setores da SEC que tenham interesse em sua adoção levando em consideração o fato de que o sistema contempla apenas as regras de negócio para atendimento das necessidades do almoxarifado da Ribeira, CD, recebimento e expedição de materiais dos mesmos.

O referido sistema/módulo acaba atingindo os processos de recebimento, expedição, devolução de itens, auditoria, inventário, cadastro de família de itens, cadastro e fragmentação de itens, cadastro de fabricantes, cadastro de fornecedores, cadastros de destino, movimentação de itens dentro e entre almoxarifados e centros de distribuição. Ele é integrado com os demais módulos do sistema da SEC, compartilhando informações e processos de gestão de cadastros.

O componente de Unidade Geográfica também faz integração com o módulo de Almoxarifado do G-SEC. Nas figuras 24 e 24b são é ilustrados trechos dos casos de uso com destaque aos relacionamentos entre o serviço expostos e as regras de negócio deste módulo dos casos de uso *Cadastrar Conferente, Cadastrar Almoxarifado e Cadastrar Estrutura Física do Almoxarifado* respectivamente.

Na figura 25b é destacada a entidade do diagrama de classes que faz referência ao serviço exposto de *Unidade Geográfica* nas entidades *LoteDirec e AloxarifadoMunicipio* com o objetivo de realizar o caso de uso *Cadastrar Almoxarifado e Cadastrar Estrutura Física do Almoxarifado*. Já a entidade *Conferente,* fez uso do serviço exposto do componente Pessoa, para também realizar o caso de uso *Cadastrar Conferente*, que também é ilustrado na figura 25.

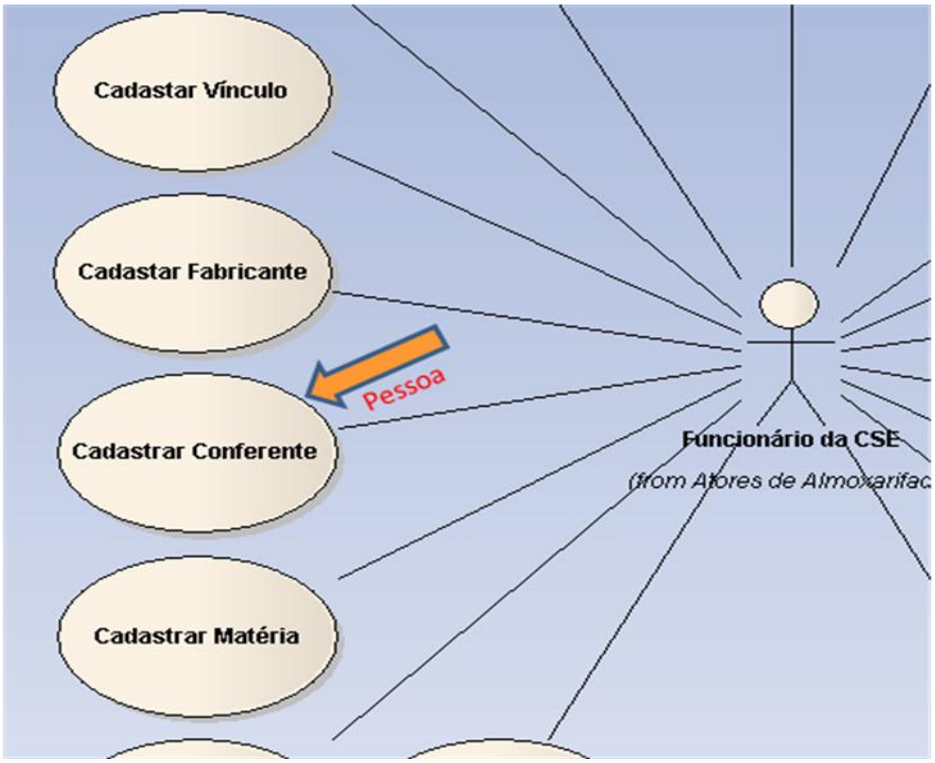

Figura 24 **–** Parte dos casos de uso do projeto Almoxarifado do serviço *Pessoa*

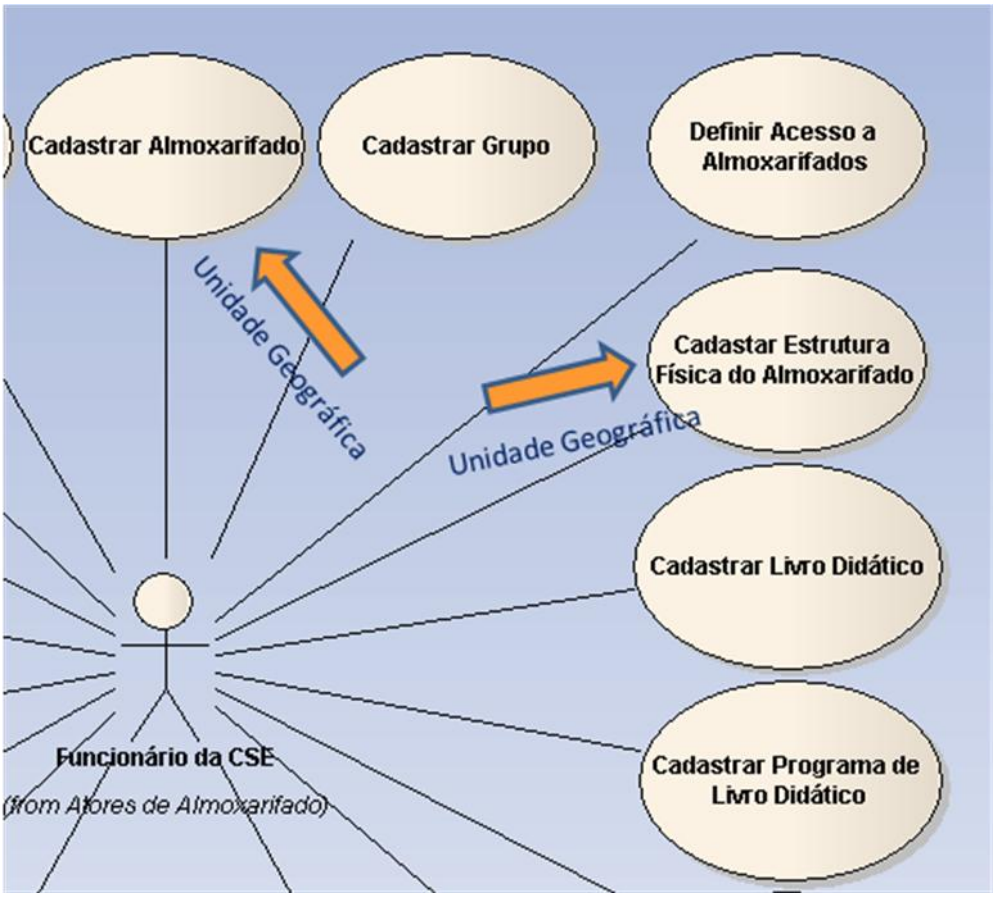

Figura 24b **–** Parte dos casos de uso do projeto Almoxarifado do serviço *Unidade Geográfica*

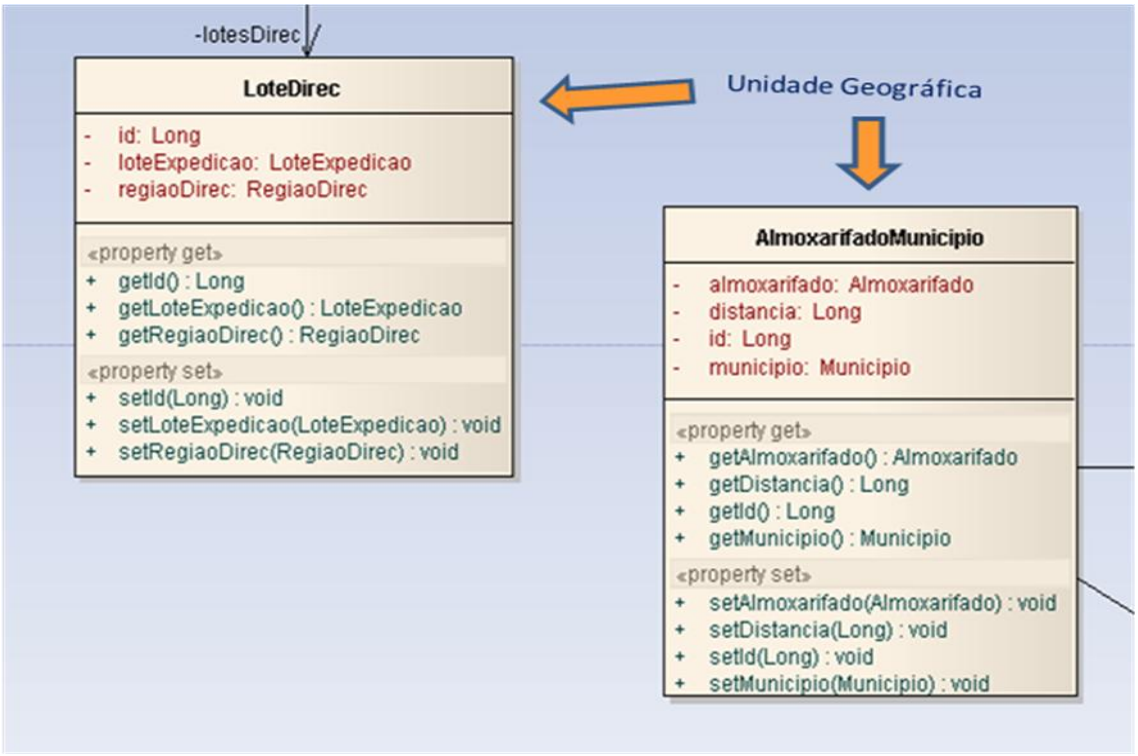

Figura 25 – Entidades de Almoxarifado em relação ao serviço *Unidade Geográfica*

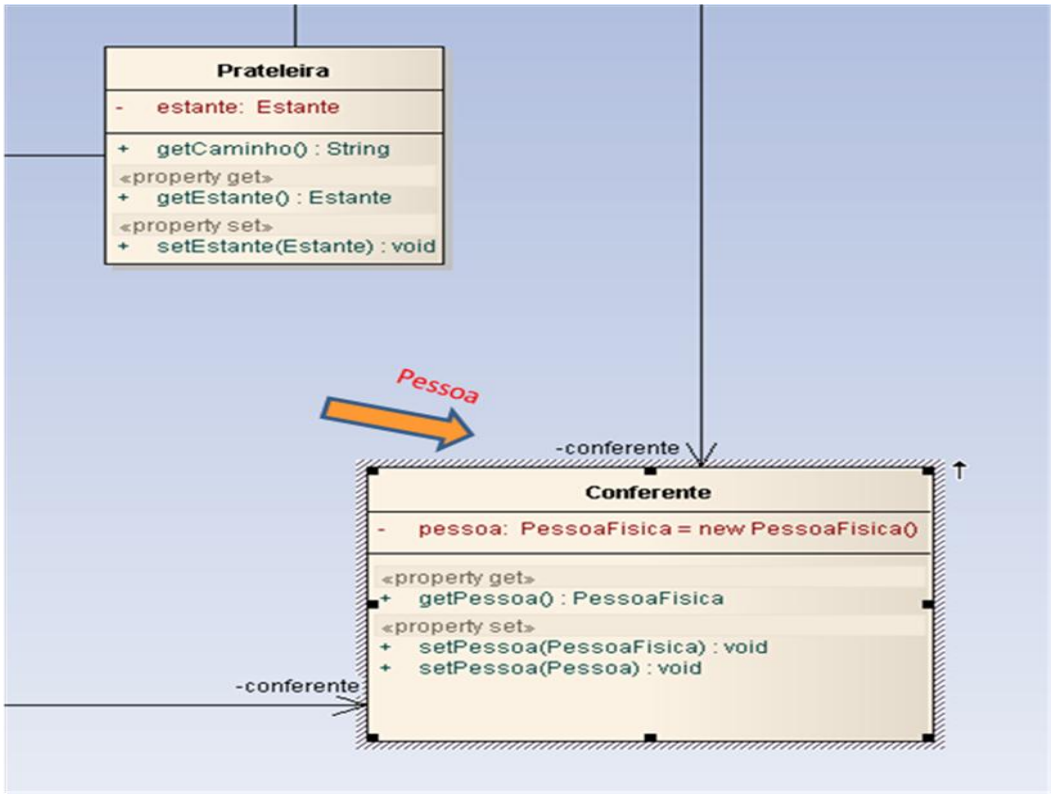

Figura 25b – Entidades de Almoxarifado em relação ao serviço *Pessoa*

4.2.6 Gestão TOPA

O módulo de Gestão TOPA é baseado no programa de governo Todos pela Alfabetização (TOPA), com a finalidade de gerir as informações extras que não são contempladas pelo sistema SBA do MEC. Este módulo possui uma integração com o SBA através de impostações periódicas.

Todos os dados do SBA devem ser importados semanalmente, exceto a taxa de analfabetismo que é importada do Censo, dos municípios da Jurisdição da DIREC e da distribuição por territórios de identidade. Isso não impede que, quando necessário, ocorram importações fora desse prazo em caráter de urgência.

O Gestão TOPA possui a integração de informações com o componente Unidade Geográfica. Na figura 26 é mostrada uma parte dos casos de uso deste módulo, onde é destacado o caso de uso que faz referência ao serviço exposto pelo componente. Já na figura 27 é ilustrada a entidade do diagrama de classes que também faz referência ao referido componente.

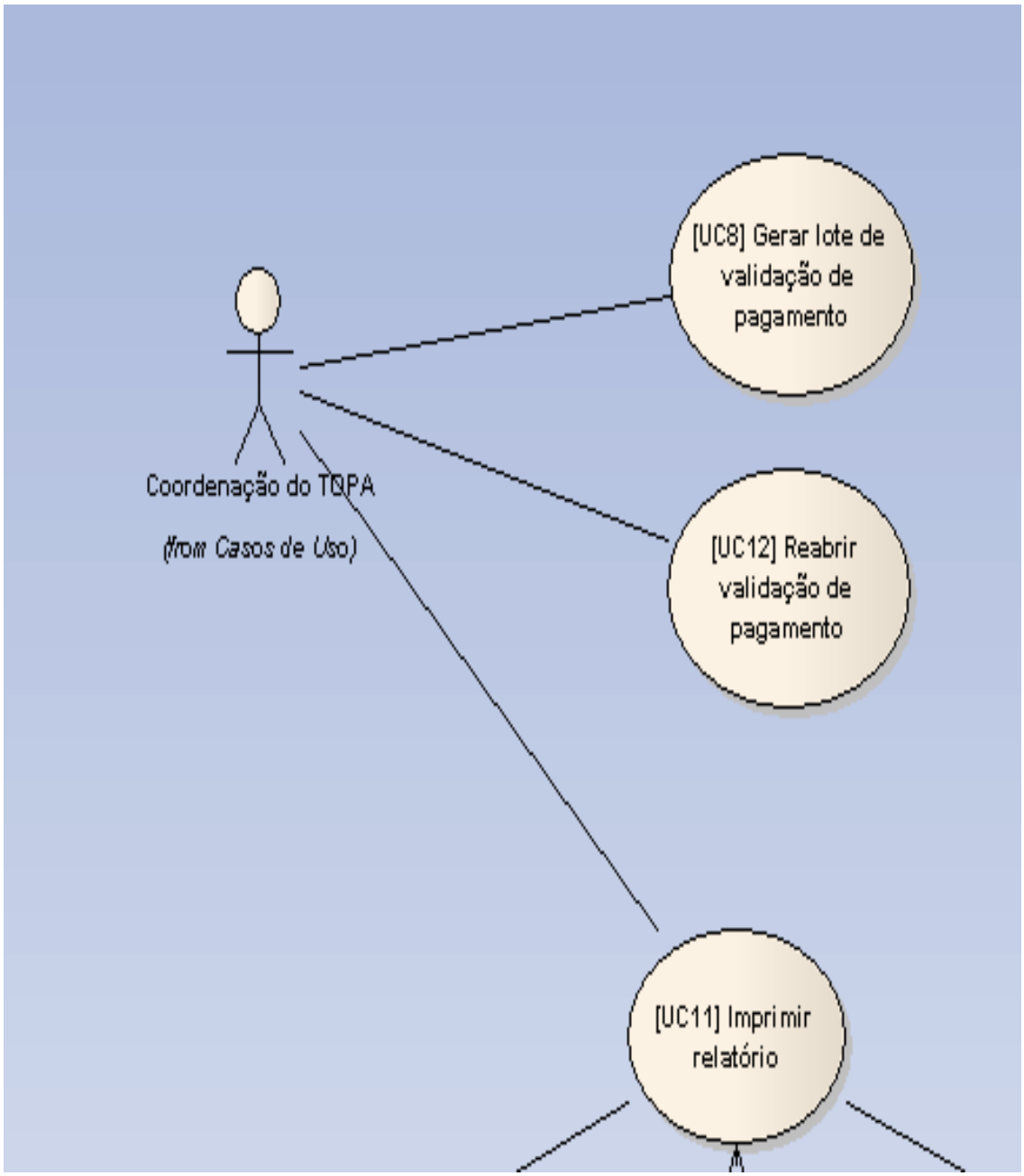

Figura 26 – Caso de uso do Gestão TOPA X Serviço Exposto

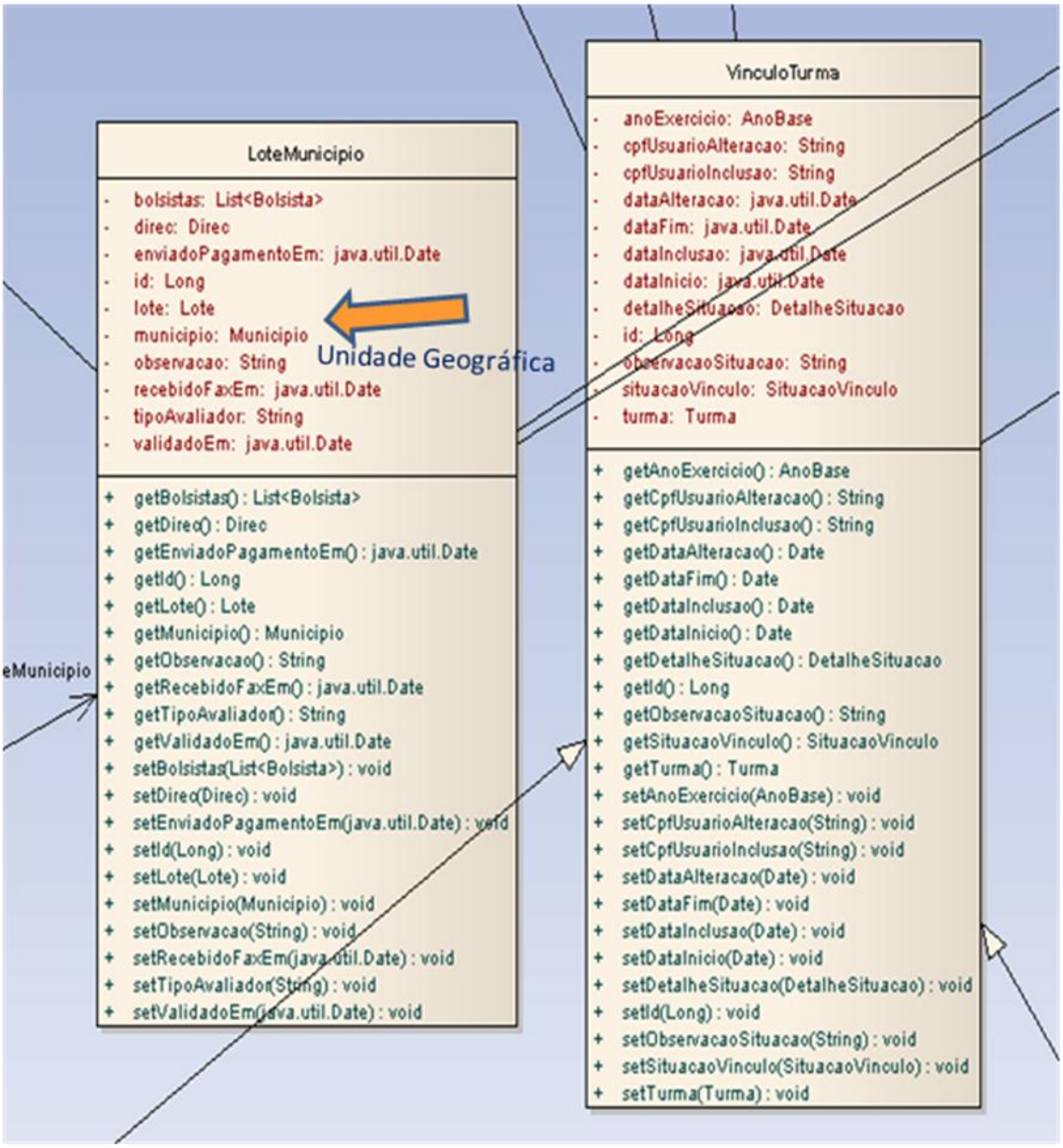

Figura 27 – Entidade do Gestão TOPA X Serviço *Unidade Geográfica*

4.2.7 SIAT

No Instituto Anísio Teixeira (IAT) há também a composição do quadro heterogêneo de arquiteturas de *software*. Com o objetivo de atender às necessidades de gerenciamento dos eventos, a SEC propiciou uma a solução com base nos novos moldes de desenvolvimento. O outro grande consumidor dos serviços é o Sistema de Gerenciamento de eventos do IAT, que tem a finalidade de gerir os eventos (cursos, palestra, reunião, etc) realizados. Essa aplicação está atualmente consumindo os serviços expostos de Unidade Geográfica e Pessoa.

O IAT é um órgão em regime especial de administração direta da SEC, que tem como finalidade planejar, coordenar, acompanhar e executar as ações de formação dos profissionais da educação de todo o Estado. O processo de Logística dos Eventos, que é o maior da instituição, não contava com qualquer ferramenta informatizada para apoiá-lo. Todas as atividades eram realizadas manualmente e a documentação gerada era arquivada em meio físico, o que faz com que o manuseio da informação e extração de relatórios se torne deficiente. Isso gerava dificuldade em atender as requisições não somente de clientes internos da Secretaria, que planejam os eventos, como também aos clientes externos a exemplo do Tribunal de Contas do Estado.

O SIAT necessita intercambiar informações de municípios e de pessoas. Por isso, houve a necessidade de consumir os serviços expostos pelos componentes de Unidade Geográfica e Pessoa. Na figura 28 está em destaque o caso de uso, que se relaciona com o serviço expostos de Unidade Geográfica.

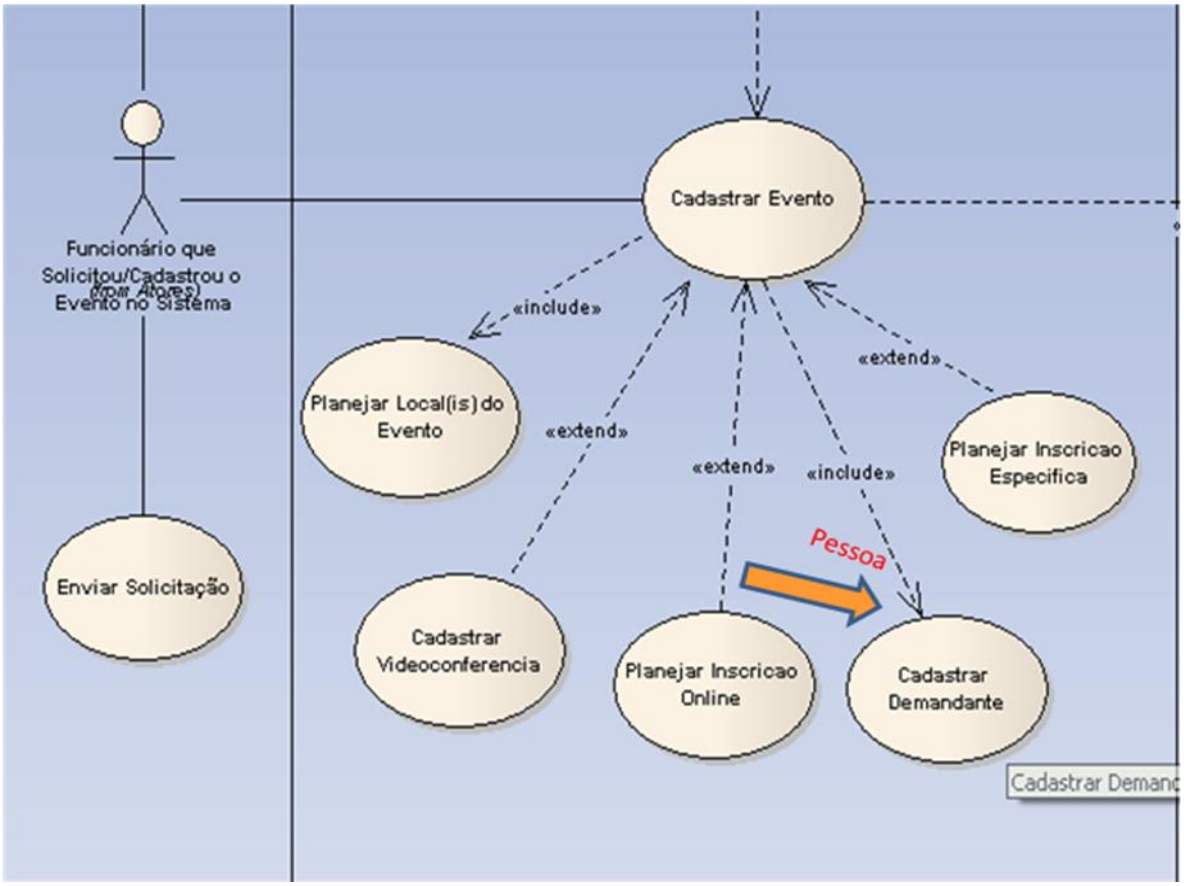

Figura 28 - Casos de usos do SIAT X Serviço *Pessoa*

Na figura 28b temos a relação entre o caso de uso *Confirmar Inscrição* com o serviço *Pessoa*, que faz relação com a entidade *Incricao* do diagrama de classes ilustrado na figura 29.

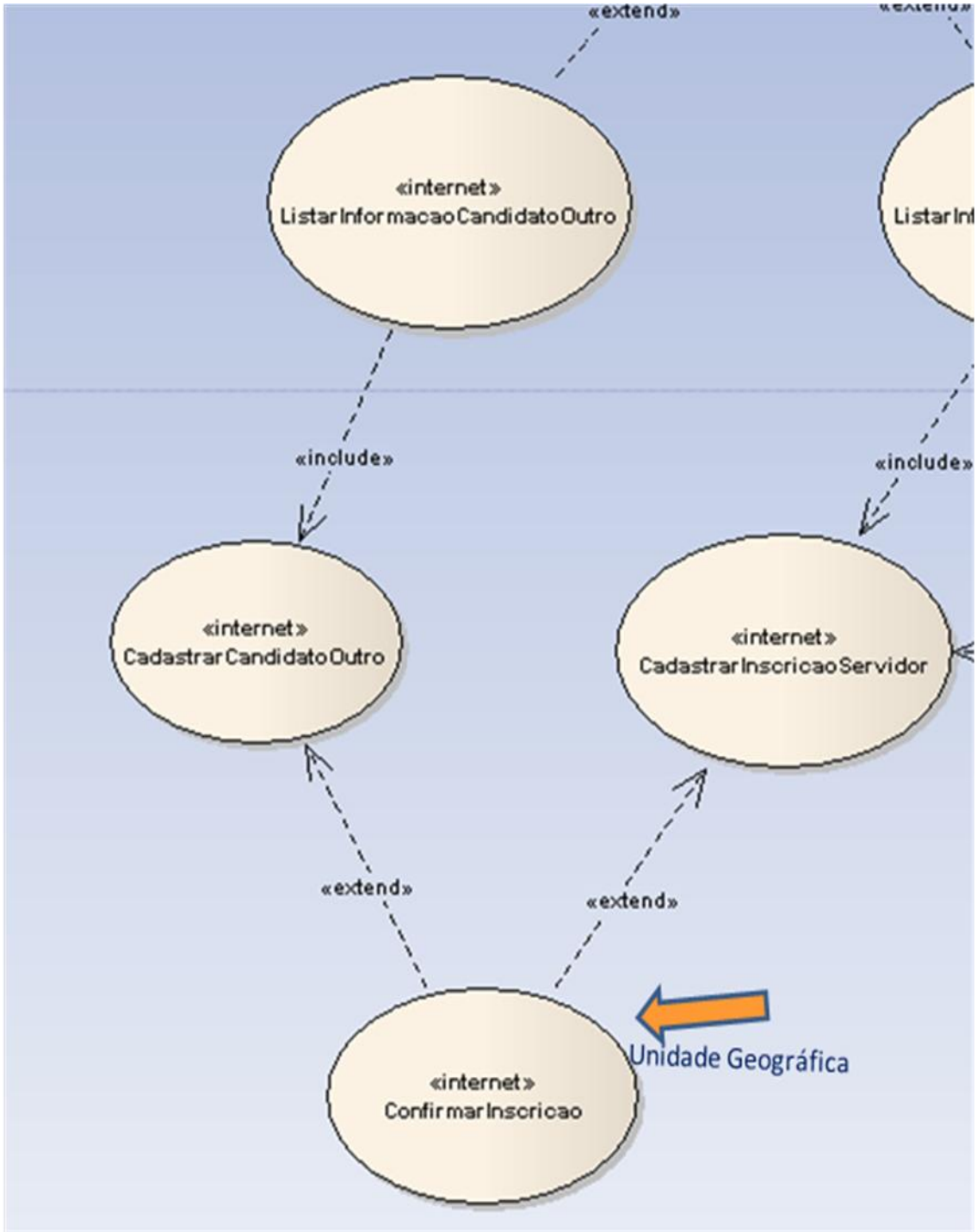

Figura 28b - Casos de usos do SIAT X Serviço *Unidade Geográfica*

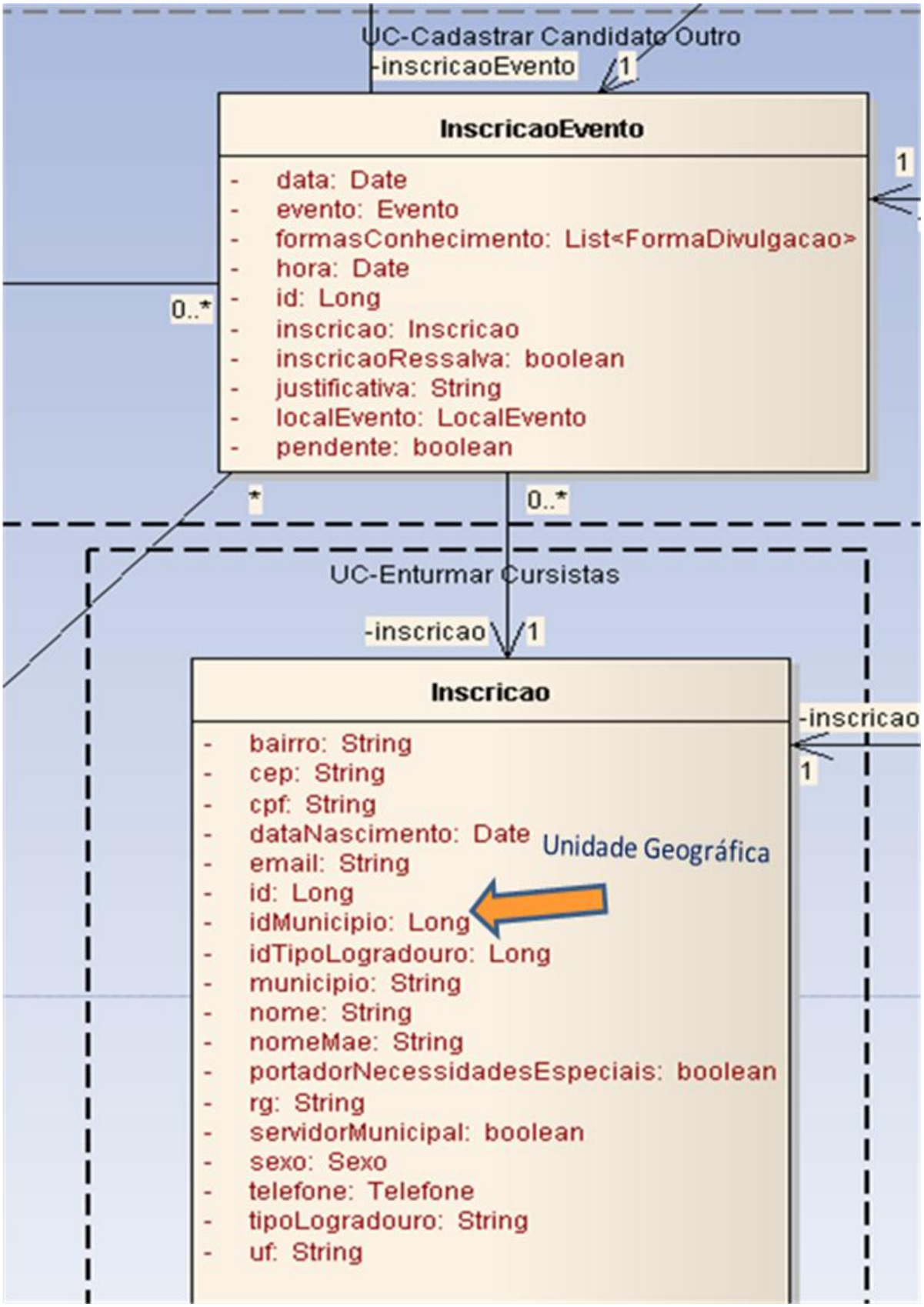

Figura 29 - Entidade do SIAT X Serviço *Unidade Geográfica*

4.3 ANÁLISE DOS RESULTADOS

Os cases que foram implementados durante a elaboração deste trabalho deram origem a alguns dados importantes. Essas informações obtidas após uma análise de comparação entre os sistemas novos e legados com base nas horas trabalhadas em cada sistema resultaram nos seguintes dados nas figuras 30 e 31:

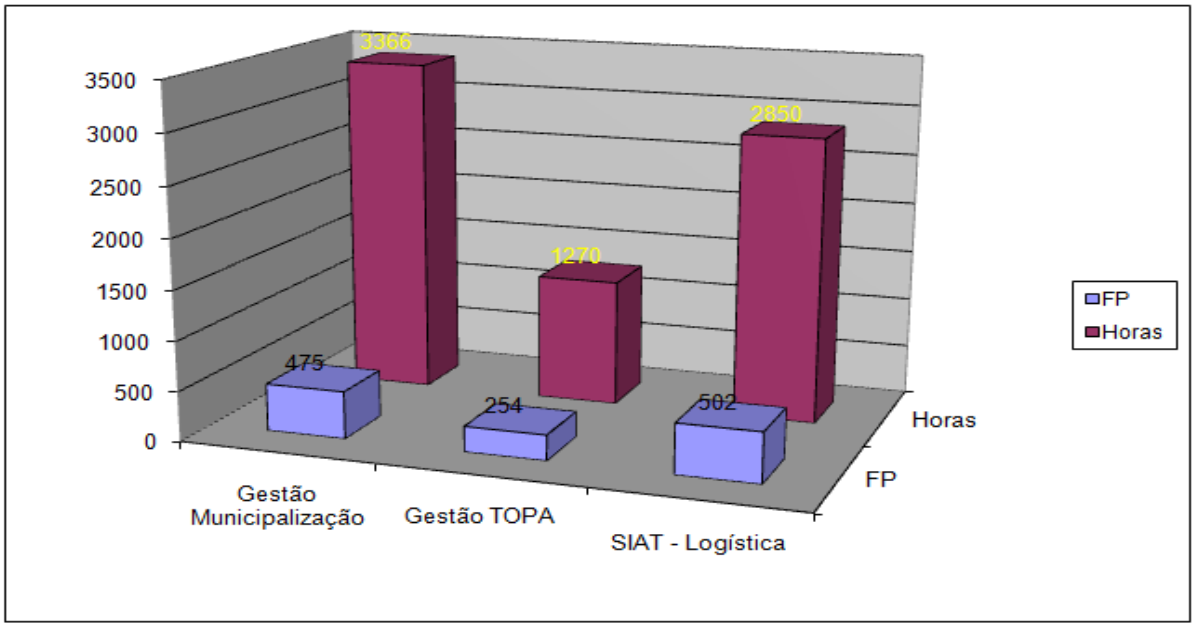

Figura 30 – Comparativo entre as aplicações com base nas horas trabalhadas

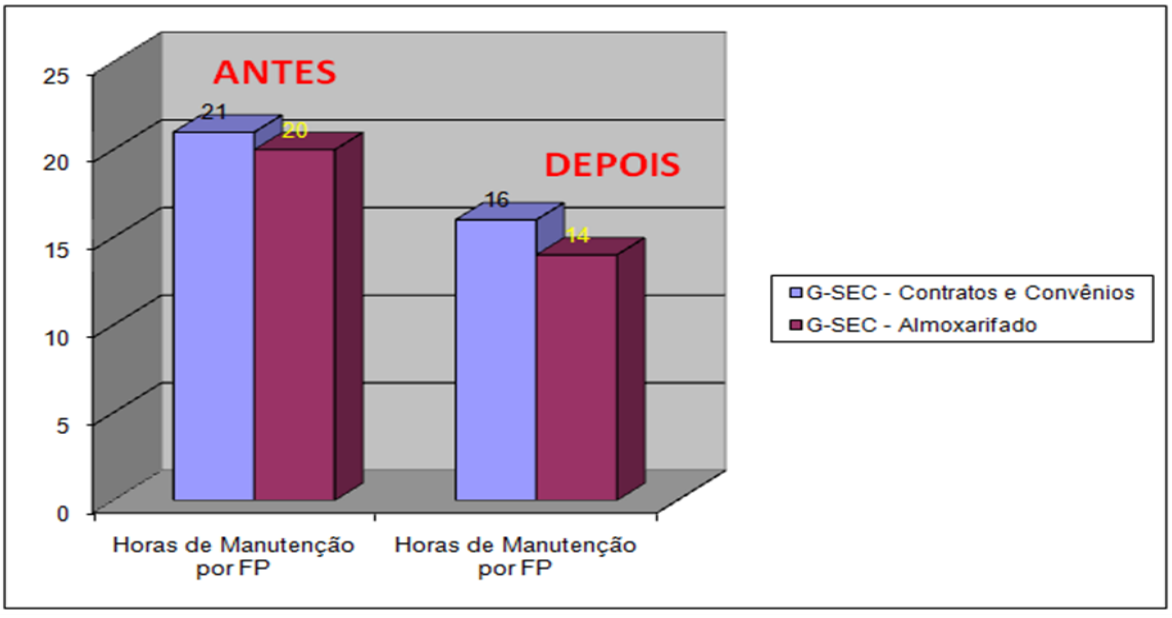

Figura 31 – Antes e depois da implantação dos serviços

Na figura 31 é ilustrado um comparativo entre três aplicações com base nos FP's (Function Point) e nas horas gastas para construir esses pontos dos sistemas. Fica evidente que as aplicações, Gestão TOPA e SIAT – Logística, que foram construídas com o reuso dos serviços gastaram um menor valor de horas em relação à aplicação legada Gestão Municipalização. Já na figura 32 é mostrado outro comparativo com base nas horas gastas para manter as aplicações legadas, antes e depois de reusar os serviços da arquitetura proposta por este trabalho. Com base nos dados levantados e analisados antes e depois dos reusos pelas aplicações dos serviços *Unidade Geográfica* e *Pessoa* foi evidenciado que há um ganho real na produtividade com a manutenção e o desenvolvimento das aplicações.

CAPÍTULO 5

5 CONCLUSÃO

Este capítulo apresenta as considerações finais sobre o uso da SOA na integração entre às aplicações legadas e as novas utilizando a SEC como *case*. Ele está dividido O PROBLEMA, A SOLUÇÃO, DIFICULDADES DE IMPLANTAÇÃO DA ARQUITETURA, IMPLANTAÇÃO GRADATIVA DA SOA, VANTAGENS E DESVANTAGENS DE SOA e TRABALHOS FUTUROS.

5.1 O PROBLEMA

A SEC, ao longo do tempo vem terceirizando sua TI. Essa estratégia adotada, com o passar do tempo, por causa dos processos licitatórios, já que se trata de uma organização pública, possibilitou o surgimento de diversas empresas de TI no setor de informática dentro da SEC. Essa diversidade foi um dos fatores que marcaram o surgimento de diversas arquiteturas de *softwares*.

Todas as organizações públicas ou privadas acabam vendo o importante papel da informatização de seus serviços. No decorrer do tempo a revolução tecnológica mudou muito os paradigmas de gestão empresarial. O crescimento exponencial, inovação e automação dos processos da organização associada com a velocidade das informações são também fatores que estão acelerando essa revolução e o surgimento de diversas arquiteturas e tecnologias dentro da SEC.

Atualmente, com o mercado de TI voltado para fornecer informações cada vez mais rápidas e confiáveis, mesmo que seja em uma instituição pública, onde seu objetivo é prestar serviços à sociedade gratuitamente, Ela também deve pensar em fornecê-los da melhor forma. Com o surgimento de uma variedade de arquiteturas de *softwares*, como já foi relatado no capítulo 1, algumas informações acabaram ficando redundantes de forma não controlada.

Com um parque de sistemas bastante heterogêneos em relação à arquitetura de *software*, a integração entre esses diversos ambientes é muito custosa. Em alguns casos, quando uma nova funcionalidade é requisitada, de custo muito alto, a organização opta por não implementá-la. Em virtude dos aspectos até aqui

apresentados foi que surgiu a motivação para propor uma nova arquitetura para melhorar a integração entre os sistemas legados e as novas aplicações que estarão por vir.

5.2 A SOLUÇÃO

Com o quadro de uma variedade de arquiteturas de softwares encontrado na SEC, propomos uma solução para melhorar a integração entre as aplicações legadas e as novas. Essa solução é baseada na adoção da arquitetura orientada a serviços.

A arquitetura proposta foi idealizada com o intuito de oferecer serviços através de *Web Services*, onde o intercâmbio entre as aplicações, troca de mensagens entre os consumidores e provedores de serviços, são realizadas utilizando padrão SOAP com XML. E os padrões WSDL e UDDI dão suporte com o intuito de possibilitar a descrição e localização dos serviços que são expostos.

Um dos fatores relatados por vários autores, como por exemplo, Pulier; Taylor; Gaffney (2008, p. 83) e Rocha (2006, p. 28) é que há um ganho na reusabilidade das funcionalidades através dos serviços expostos. Na organização estudada foi também observado esse ganho, onde o reuso das funcionalidades, principalmente das aplicações legadas, elevou a produtividade no desenvolvimento e manutenção das aplicações e uma considerável melhora no espalhamento das informações, conforme analise relatada no capítulo anterior.

Outro aspecto defendido pelos adeptos de SOA é a portabilidade das arquiteturas, ou seja, caso no futuro a instituição deseje mudar a tecnologia de seus serviços expostos, *Web Services*, as aplicações legadas não sofrerão grandes impactos, já que a troca de mensagens entre elas é realizada através de padrões abertos. Mesmo sendo esse aspecto um fator muito positivo, ele não foi detectado, já que durante a elaboração deste trabalho não houve alteração na tecnologia adotada para expor os serviços escolhidos, e, inicialmente, os mesmos serão expostos apenas na intranet da própria organização.

A flexibilidade que a SOA prega fica muito mais forte e evidenciada com o uso de ESB. Os componentes são interligados através desse barramento. Com isso,

facilita a manutenibilidade dos serviços desde que suas interfaces não sejam alteradas, ou seja, quando um serviço sofrer alteração de sua localização, os serviços que o estão consumindo não necessitarão sofrer alteração. O uso de um ESB na arquitetura fornece todo esse poder de flexibilidade, mas temos que levar em consideração os custos e a quebra de paradigma quando estivermos fazendo o planejamento de sua implementação. Em algumas organizações pode não dar muito certo.

5.3 DIFICULDADES DE IMPLANTAÇÃO DA ARQUITETURA

A engenharia de *software* vem buscando, fortemente, maneiras otimizadas para o desenvolvimento de novas aplicações. Porém, o que encontramos nas organizações, como na SEC, uma diversidade de *softwares* que não foram implementados com essa visão. A arquitetura orientada a serviços vem com a proposta de integrar esses *softwares* “antigos” com os novos através da exposição de serviços.

Para uma implantação bem sucedida temos que enfrentar alguns obstáculos. A quebra de paradigma é um desses aspectos, onde o desenvolvimento de *software* em muitos casos era departamentalizado, ou seja, a equipe de desenvolvimento não pensava em componentes corporativos. Como já relatado anteriormente, com a SOA devemos pensar nos serviços corporativos. Outro fator importante é a resistência de aquisição dos novos conhecimentos pelos colaboradores dentro da organização. A SOA vem com propostas novas, por isso, devemos ter uma preocupação em socializar o conhecimento dentro da instituição para que possamos obter o sucesso na implantação dessa nova arquitetura.

5.4 IMPLANTAÇÃO GRADATIVA DA SOA

Para propiciar um melhor sucesso na implantação da SOA dentro das organizações com o mesmo perfil da SEC, durante a realização deste trabalho foi necessário realizar a implementação dessa arquitetura de maneira gradativa, para enfrentarmos as dificuldades mencionadas anteriormente. Inicialmente, devemos definir as funcionalidades que serão os serviços. Em uma segunda etapa, devem ocorrer ás implementações dos serviços escolhidos anteriormente. Logo em

seguida, devemos implementar os consumidores. E por fim, no quarto passo, devemos criar o barramento de serviços corporativos. Seguindo essas diretrizes na ordem, a implantação da arquitetura orientada a serviços tem uma forte possibilidade de chegar ao sucesso.

5.5 VANTAGENS E DESVANTAGENS DE SOA

As organizações adotam a arquitetura orientada a serviços para fazer uso das vantagens que ela proporciona, tornando a instituição uma fornecedora de serviços com uma melhor qualidade para a sociedade. A SOA viabiliza um aumento na produtividade, eleva o reuso das aplicações legadas, propicia uma redução nos custos de manutenibilidade, dá uma maior flexibilidade às novas aplicações e potencializa a portabilidade de funções de negócio da organização.

Como nem tudo é puramente positivo, temos na SOA o problema de gerenciamento de segurança. Segundo Rocha (2006, p. 35) a complexidade na interoperabilidade dos mecanismos de segurança que surgem através dos novos processos de negócio pode contribuir para dificultar a implantação desta arquitetura. Os mecanismos de segurança estão em evolução como a própria SOA, mas eles ainda não estão solidificados para garantir uma segurança dos dados da organização. Por isso, ao optarmos pelo uso da arquitetura orientada a serviços temos que destacar um estudo de como será a segurança adotada.

5.6 TRABALHOS FUTUROS

Uma sugestão para um trabalho futuro é a realização de um estudo focado nos mecanismos de segurança que devem ser adotados em organizações com as mesmas características da que foi estudada com o objetivo de fortalecer esses mecanismos no mundo da SOA.

Outra possibilidade é realizar um estudo sobre o gerenciamento de Rede de SOA, descrevendo o quão complexo ele pode ser. Seu objetivo seria estudar o gerenciamento, transporte, roteamento e segurança de mensagens para viabilizar uma rede de SOA confiável.

E por fim, pode ser feito um estudo de uma metodologia bem definida para realizar os mapeamentos de maneira adequada das funcionalidades e/ou requisitos para os serviços.

REFERÊNCIAS

ARANTES, Juliana Amaral; WESTPHALL, Carlos Becker; CUSTÓDIO, Ricardo Felipe. **Modelo analítico para avaliar plataformas cliente/servidor e agentes móveis aplicado à gerência de redes.** Santa Catarina, 2003. Disponível em: <http://www.lbd.dcc.ufmg.br:8080/colecoes/sbrc/2002/027.pdf>. Acesso em: 19 jul. 2008.

ALMEIDA JUNIOR, Raimundo Araujo de. Alinhamento estratégico entre os objetivos organizacionais e sistemas de informação da secretaria da educação do estado da bahia. **Revista de Tecnologia Empresarial (rte)**, Salvador, p.1-15, 10 set. 2005. Semestral.

AZEVEDO, Leonardo Guerreiro et al. **Desenvolvimento baseado em componentes experiências de sucesso.** 2002. Disponível em: <http://www2.dc.ufscar.br/~valdirene/Publications/CLEI2002.pdf>. Acesso em: 29 jul. 2008.

BARBIERI, Carlos. **BI-business intelligence:** modelagem & tecnologia. Rio de Janeiro: Axcel Books do Brasil, 2001.

BECKER, Aleksader Knabben; CLARO, Daniela Barreiro; SOBRAL, João Bosco. **Web services e XML um novo paradigma da computação distribuída.** Florianópolis, 2002. Disponível em: <http://www.inf.ufsc.br/~danclaro/download/ArtigoWebServices.pdf>. Acesso em: 12 jul. 2008.

BERGEY, Jony; BRIEN, Lenthi; SMITH, Douglas. Using the optins analysis for reengineering (OAR) method for mining components for a product line. In: PROCEEDINGS OF THE SECOND SOFTWARE PRODUCT LINE CONFERENCE, 2., 2002, San Diego. **Software Produc Lines.** Berlin: Splc2, 2002. p. 1 - 147.

BERTONI, Guilherme Machado. **Uma arquitetura baseada em web services com diferenciação de serviços para integração de sistemas embutidos a outros sistemas.** 2006. 131 f. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Florianópolis, 2006.

BOND, Martins et al. **Aprenda J2EE com EJB, JSP, servilets, JNDI, JDBC e XML.** São Paulo: Makron Books, 2003.

BRAGA, Luciano Rogério Perdigão. **Curso básico de programação natural.** Disponível em: <http://www.geocities.com/guia_rapido/natural_basico.pdf>. Acesso em: 21 jul. 2008.

PMI - PROJECT MANAGEMENT INSTITUTE. **Um guia do conjunto de conhecimentos em gerenciamento de projetos:** PMBOK. São Paulo, 2004.

COSTA, Ivanir; CARVALHO, Antonio Rodrigues. **Tendências sobre a arquitetura orientada a serviços - SOA.** Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP2007_TR670485_9968.pdf>. Acesso em: 11 ago. 2008..

CUNHA, Mônica. **Desafios de integração de sistemas:** roadmap, evolução tecnológica e impactos organizacionais. Disponível em: <http://www.redenet.edu.br/publicacoes/arquivos/20071227_151130_GEST-013.pdf>. Acesso em: 03 ago. 2008.

CUMMINS, Fred A.. **Integração de sistemas – EAI (enterprise application integration).** Rio de Janeiro: Campus, 2002.

DEGAN, Joecy Otsuka Cortês. **Integração de dados corporativos:** uma proposta de arquitetura baseada em serviços de dados. 2005. 115 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2005.

DANTAS, Alexandre. **Suporte a padrões no projeto de software.** Disponível em: <://www.lbd.dcc.ufmg.br:8080/colecoes/sbes/2002/038.pdf>. Acesso em: 17 ago. 2008.

FARLEY, Jim. **Projetos práticos com JBoss Seam.** Rio de Janeiro: Ciência Moderna, 2008.

FONTANETTE, Valdirene. **AMGraA: u**ma abordagem para migração gradativa de aplicações legadas. 2005. 144 f. Dissertação (Mestrado) - Universidade Federal de São Carlos, São Carlos, 2005.

GARCIA, Simone de Souza; SHINOTSUKA, Tania Hiromi. **EAI - enterprise application integration.** Disponível em: <http://www.bax.com.br/Bax/orientacao/alunos/PSI/EAI1.pdf>. Acesso em: 29 jul. 2008.

GOMES, Maria Cristina F. et al. **Certificação da utilização de padrões de projeto no desenvolvimento orientado a modelos.** Disponível em: <http://www.lbd.dcc.ufmg.br:8080/colecoes/sbes/2006/005.pdf>. Acesso em: 20 ago. 2008.

IANSITI, Marco; CLARK, Kim Brith. **Integration and dynamic capability: e**vidence from product development in automobiles and mainframe computers. Disponível em: <http://icc.oxfordjournals.org/cgi/content/abstract/3/3/557>. Acesso em: 17 ago. 2008.

IBM CORPORATION.**IBM SOA Foundation:** providing what you need to get started with SOA. New York: Ibm Corporation, 2005.

LARENTIS, Andresa Vargas. **Aruba: u**ma arquitetura para geração de serviços a partir de sistemas legados de forma intrusiva. 2007. 104 f. Dissertação (Mestrado) - Universidade do Vale do Rio Dos Sinos, São Leopoldo, 2007.

LEWIS, Ghonn; MORRIS, Elton; SMITH, Douglas. Migration of lagacy components to service-oriented architectures. **The Dod Software Tech**, New York, v. 8, n. 3, p.1-7, 03 out. 2005. Semestral.

MACHADO, Janes Countis. **Um estudo sobre o desenvolvimento orientado a serviços.** 2004. 125 f. Dissertação (Mestrado) - Pontifícia Universidade Católica, Rio de Janeiro, 2004.

MACIEL, Diego Rabelo. **Implementação de um disco virtual seguro baseado em Web Services.** 2007. 74 f. Monografia (Graduação) - Universidade Federal do Maranhão, São Luís, 2007.

MARTINS, Victor Manuel Moreira. **Integração de sistemas de informação:** perspectivas, normas e abordagens. 2005. 218 f. Dissertação (Mestrado) - Universidade do Minho, Lisboa - Portugual, 2005.

MICROSOFT. **Learn about service oriented architecture (SOA)**. New York: Microsoft Corporation, 2006a.

MICROSOFT. **Web services and the microsoft**. New York: Microsoft Corporation, 2006b.

NATIS, Yefim Verbes. **Service-oriented architecture scenario.** Disponível em: <http://www.g2r.com/DisplayDocument?doc_cd=114358>. Acesso em: 05 ago. 2008.

MAIA, Natanael Elias Nascimento. **ODYSSEY-MDA: u**ma abordagem para transformação de modelos. Disponível em: <http://reuse.cos.ufrj.br/prometeus/publicacoes/DissertacaoMScNatanaelMaia.pdf>. Acesso em: 15 ago. 2008.

OASIS. **Reference Model for Service Oriented Architeture V 1.0.** Disponível em: <http://www.fapepi.pi.gov.br/novafapepi/ciencia/documentos/REFRAT%D4METRO.PDF>. Acesso em: 20 jul. 2008.

PACHECO, Viviane. **EAI:** a sigla da integração dos sistemas internos e externos. 2000. Disponível em: <http://www.neogrid.com.br/portugue/imprensa/imprensa/surftrade/25.../Surftrade_Brasil.html>. Acesso em: 15 jul. 2008.

PULIER, Eric; TAYLOR, Hugh; GAFFNEY, Paul. **Compreendendo SOA Corporativa.** Rio de Janeiro: Ciência Moderna, 2008.

ROCHA, Claudio Aparecido. **Um estudo sobre os desafios de segurança na adoção da Arquitetura Orientada a Serviços.** 2006. 147 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2006.

SAMPAIO, Cleuton. **SOA e Web Service em Java.** Rio de Janeiro: Brasport, 2006.

SANTOS JUNIOR, Alfredo Luiz Dos. **Integração de sistemas com Java.** Rio de Janeiro: Brasport, 2007.

SOMMERVILLE, Ivan. **Engenharia de software.** São Paulo: Engenharia de Software, 2005.

STEFANI, Eduardo; SEKI, Yuri. **Engenharia de software cliente/servidor.** Disponível em: <http://www.eduardostefani.eti.br/bennett/engsoftware/cliente-servidor.pdf>. Acesso em: 01 jul. 2008.

TAIT, Tania Fatima Calvi; PACHECO, Roberto Carlos Dos Santos. **Proposição de um modelo de arquitetura de sistemas de informação para o setor público.** Disponível em: <http://periodicos.uem.br/ojs/index.php/ActaSciTechnol/article/viewFile/2783/1841>. Acesso em: 07 jul. 2008.

W3C. **Web Service Architecture:** W3C working group. Disponível em: <http://www.w3.org/TR/ws-addr-arch/>. Acesso em: 13 jul. 2008.

APÊNDICE A – Códigos dos Serviços

UNIDADE GEOGRÁFICA

```
/*
 * UnidGeograficaService.java
 *
 */
package br.gov.ba.sec.unidgeografica.ws;

import br.gov.ba.sec.unidgeografica.dao.UnidGeograficaDao;
import br.gov.ba.sec.unidgeografica.service.UnidGeograficaService;
import br.gov.ba.sec.unidgeografica.type.EstadoType;
import br.gov.ba.sec.unidgeografica.type.MunicipioType;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.jws.WebService;
import javax.persistence.PersistenceContext;
import javax.annotation.PostConstruct;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author fcsilva
 */
@Stateless
@WebService(name="UnidGeograficaWebService", portName="UnidGeograficaServicePort")
public class UnidGeograficaWebService implements UnidGeograficaService {

    @PersistenceContext
    private EntityManager em;
    private UnidGeograficaDao unidGeograficaDao;

    @PostConstruct
    public void carregaEscola() {
        unidGeograficaDao = new UnidGeograficaDao(em);
    }

    @WebMethod
    public List<EstadoType> obterEstados() {
       return unidGeograficaDao.obterEstados();
    }

    @WebMethod
    public List<MunicipioType> obterPorCodigoIbge(@WebParam(name = "estado") String estado) {
       return unidGeograficaDao.obterPorCodigoIbgeEstado(estado);
    }

    @WebMethod
    public List<MunicipioType> obterPorNomeEstado(@WebParam(name = "estado") String estado) {
        return unidGeograficaDao.obterPorNomeEstado(estado);
    }

}

*
```

```
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package br.gov.ba.sec.unidgeografica.service;

import br.gov.ba.sec.unidgeografica.type.EstadoType;
import br.gov.ba.sec.unidgeografica.type.MunicipioType;
import java.io.Serializable;
import java.util.List;

public interface UnidGeograficaService extends Serializable{

        /**
       * Obtém uma lista de estados
        * @param escola
        * @return
        */
        List<EstadoType> obterEstados();

        /**
       * Obtém uma lista de municípios de acordo com o estado
        * @param escola
        * @return
        */
      List<MunicipioType> obterPorCodigoIbge(String estado);

        /**
       * Obtém uma lista de municípios de acordo com o estado
        * @param escola
        * @return
        */
      List<MunicipioType> obterPorNomeEstado(String estado);
}


*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package br.gov.ba.sec.unidgeografica.dao;

import br.gov.ba.sec.unidgeografica.type.EstadoType;
import br.gov.ba.sec.unidgeografica.type.MunicipioType;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.Query;

public class UnidGeograficaDao {

    private EntityManager em;

    public UnidGeograficaDao(EntityManager em) {
        this.em = em;
    }

    public List<MunicipioType> obterPorCodigoIbgeEstado(String codigoIbge) {
```

```
        if (codigoIbge == null) {
            return null;
        }
        try {
            Query query = em.createQuery("select new " +
                    "br.gov.ba.sec.unidgeografica.type.MunicipioType(m.nome, m.codigoIbge, m.pai.nome,
m.pai.codigoIbge) " +
                    "from Municipio m where m.pai.codigoIbge=:codigoIbge and m.ativo = 1");
            query.setParameter("codigoIbge", codigoIbge);
            return query.getResultList();
        } catch (NoResultException e) {
            return null;
        }

    }

    public List<MunicipioType> obterPorNomeEstado(String estado) {
        if (estado == null) {
            return null;
        }
        try {
            Query query = em.createQuery("select new " +
                    "br.gov.ba.sec.unidgeografica.type.MunicipioType(m.nome, m.codigoIbge, " +
                    "m.pai.nome, m.pai.codigoIbge) from Municipio m where upper(trim(m.pai.nome))=
upper(:estado) " +
                    "and m.ativo = 1");
            query.setParameter("estado", estado);
            return query.getResultList();
        } catch (NoResultException e) {
            return null;
        }

    }

    public List<EstadoType> obterEstados() {
        try {
            Query query = em.createQuery("select br.gov.ba.sec.unidgeografica.type.EstadoType(e.nome,
e.codigoIbge) " +
                    "from Estado e where e.ativo = 1");
            return query.getResultList();
        } catch (NoResultException e) {
            return null;
        }
    }
}
```

PESSOA

```
package br.gov.ba.sec.pessoa.ws;

import java.text.ParseException;
import java.util.List;

import javax.annotation.PostConstruct;
import javax.ejb.Stateless;
import javax.jws.WebMethod;
import javax.jws.WebParam;
```

```
import javax.jws.WebService;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import br.gov.ba.sec.pessoa.dao.pessoaDao;
import br.gov.ba.sec.pessoa.entity.pessoa;
import br.gov.ba.sec.pessoa.service.pessoaService;
import br.gov.ba.sec.pessoa.util.DateUtil;

@Stateless
@WebService(serviceName = "pessoaWS", portName = "pessoaPort")
public class pessoaWebService implements pessoaService {

	@PersistenceContext(unitName = "pessoaPU")
	private EntityManager entityManager;
	private pessoaDao pessoaDao;

	@PostConstruct
	public void CarregarEntityManegerpessoa() {
		pessoaDao = new pessoaDao(entityManager);
	}

	@WebMethod
	public Boolean isDiretor(@WebParam(name = "cadastro") Long cadastro) {
		return pessoaDao.isDiretor(cadastro);
	}

	@WebMethod
	public Boolean isProfessor(@WebParam(name = "cadastro") Long cadastro) {
		return pessoaDao.isProfessor(cadastro);
	}

   @WebMethod
   public List<pessoa> obterpessoaAtivo(@WebParam(name = "cpf") Long cpf,
        @WebParam(name = "sexo") String sexo,
        @WebParam(name = "dataNascimento") String dataNascimento) throws ParseException {

      return pessoaDao.obterpessoaAtivo(cpf, sexo, DateUtil.toDate(dataNascimento));

   }

	@WebMethod
	public pessoa obterpessoaAtivoPorCadastro(
			@WebParam(name = "cadastro") Long cadastro) {
		return pessoaDao.obterpessoaAtivoPorCadastro(cadastro);
	}

}


package br.gov.ba.sec.pessoa.service;

import br.gov.ba.sec.pessoa.entity.pessoa;
import java.util.List;


import java.text.ParseException;

public interface pessoaService {
```

```
        List<pessoa> obterpessoaAtivo (Long cpf, String sexo, String dataNascimento) throws
ParseException;
        pessoa obterpessoaAtivoPorCadastro (Long cadastro);
        Boolean isProfessor (Long cadastro);
        Boolean isDiretor (Long cadastro);
}


package br.gov.ba.sec.pessoa.dao;

import br.gov.ba.sec.pessoa.entity.pessoa;
import java.util.Date;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.NoResultException;
import javax.persistence.Query;


public class pessoaDao {

   private EntityManager entityManager;

   public pessoaDao(EntityManager entityManeger) {
      this.entityManager = entityManeger;
   }

   public pessoa obterpessoaAtivoPorCadastro(Long cadastro) {
      String hql = "select s from pessoa s " +
            "join fetch s.dadosPessoais join fetch s.planoCargo join fetch s.funcao " +
            "join fetch s.orgao join fetch s.instrucao " +
            "where s.cadastro = :cadastro and upper(s.situacao.ativo) = 'S'";
      Query query = entityManager.createQuery(hql);
      query.setParameter("cadastro", cadastro);
      try {
         return (pessoa) query.getSingleResult();
      } catch (NoResultException cause) {
         return null;
      }
   }

   @SuppressWarnings("unchecked")
   public List<pessoa> obterpessoaAtivo(Long cpf, String sexo,
         Date dataNascimento) {
      String hql = "select s from pessoa s " +
            "join fetch s.dadosPessoais join fetch s.planoCargo join fetch s.funcao " +
            "join fetch s.orgao join fetch s.instrucao " +
            "where s.dadosPessoais.dadosPessoaisPK.cpf = :cpf " +
            "and s.dadosPessoais.dadosPessoaisPK.sexo = :sexo " +
            "and s.dadosPessoais.dataNascimento = :dataNascimento " +
            "and upper(s.situacao.ativo) = 'S'";
```

```
      Query query = entityManager.createQuery(hql);
      query.setParameter("cpf", cpf);
      query.setParameter("sexo", sexo);
      query.setParameter("dataNascimento", dataNascimento);
      try {
         return query.getResultList();
      } catch (NoResultException cause) {
         return null;
      }
   }
   public Boolean isProfessor(Long cadastro) {
      String hql = "select s from pessoa s where s.cadastro = :cadastro " + " and
upper(s.situacao.ativo) = 'S' " + " and upper(s.planoCargo.cargo.cargo) like 'PROFESSOR%' " + " and
upper(s.planoCargo.cargo.ativo) = 'S' ";
      Query query = entityManager.createQuery(hql);
      query.setParameter("cadastro", cadastro);
      try {
         if (query.getSingleResult() != null) {
            return true;
         }
         return false;
      } catch (NoResultException cause) {
         return false;
      }
   }
   public Boolean isDiretor(Long cadastro) {
      String hql = "select s from pessoa s where s.cadastro = :cadastro " + " and
upper(s.situacao.ativo) = 'S' " + " and upper(s.funcao.funcao) like 'DIRETOR%' ";
      Query query = entityManager.createQuery(hql);
      query.setParameter("cadastro", cadastro);

      try {
         if (query.getSingleResult() != null) {
            return true;
         }
         return false;
      } catch (NoResultException cause) {
         return false;
      }
  }

}
```