JOSÉ HENRIQUE MADEIRA CIMINO

ANÁLISE DA ARQUITETURA DE SOFTWARE PARA INTEROPERABILIDE ENTRE SISTEMAS VIA ARQUIVOS

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

Área de Concentração: Tenologia de Software

Orientador: Prof[a].Dr[a]. Gabriela María Cabel Barbarán

São Paulo
2015

Catalogação-na-publicação

Cimino, José Henrique Madeira
ANÁLISE DA ARQUITETURA DE SOFTWARE PARA INTEROPERABILIDE ENTRE SISTEMAS VIA ARQUIVOS / J. H. M. Cimino -- São Paulo, 2016.
45 p.

Monografia (MBA em Tecnologia de Software) - Escola Politécnica da Universidade de São Paulo. PECE – Programa de Educação Continuada em Engenharia.

1.Tecnologia de Software I.Universidade de São Paulo. Escola Politécnica. PECE – Programa de Educação Continuada em Engenharia II.t.

AGRADECIMENTOS

Agradeço à Prof^{a}.Dr^{a}. Gabriela María Cabel Barbarán por me ensinar e orientar na formação deste trabalho e principalmente calma e paciência.

Á Tecnologia Única que me deu a oportunidade de estudar e finalizar o curso.

Agradeço à minha família que sempre acreditou em mim e pela compreensão na minha ausência nesse último ano.

Agradeço aos meus amigos pelo apoio e insentivo que me deram.

RESUMO

Este trabalho teve como objetivo analisar uma arquitetura de software para a interoperabilidade via arquivos entre sistemas, utilizando o padrão em camadas. Para isso, foi feito uma revisão bibliográfica relacionada aos temas de arquitetura, padrões de design, interoperabilidade, interoperabilidades por arquivos, no intuito de identificar os benefícios da definição de uma arquitetura, assim também, foi descrito um caso real de uma empresa de seguros. Após a análise dos processos de negócio da empresa e dos estilos arquiteturais apresentados, foi possível identificar que a arquitetura em camadas, foi a mais adequada para atender os requisitos de qualidade da empresa e também as políticas internas de negócio.

Palavras chaves: arquitetura, interoperabilidade, troca de arquivos, arquitetura em camadas, empresa de seguros.

ABSTRACT

This study aimed to analyze a software architecture for interoperability via files between systems using the layers architecture patterns. For this, it made a literature review related to architectural themes, design patters, interoperability, interoperabilities for files in order to identify the benefits of defining an architecture, thus also described a real case of an insurance company. After analyzing the company's business process and presented architectural could be identified that the layered architecture was the most suitable to meet the company's quality requirements, and internal business policies.

Key words: architecture, interoperability, exchange files, layered architecture, insurance company.

LISTA DE ABREVIATURAS E SIGLAS

B2B	Business-to-Business
B2C	Business-to-Customer
B2E	Business-to-Employee
DBA	Data Base Administrator
ETL	Extract, Transform and Load
FTP	File Transport Protocol
HTML	Hypertext Markup Language
IP	Internet Protocol
SQL	Structured Query Language
SSIS	SQL Server Integration Services
TCP	Transmission Control Protocol
UoD	Universe of Discourse
XML	EXtensible Markup Language

LISTA DE FIGURAS

Pág

LISTA DE QUADROS

Pág

SUMÁRIO

Pág.

1. INTRODUÇÃO

1.1 Motivações

Devido à complexidade dos sistemas de software, uma arquitetura bem definida e que consiga atender as necessidades do cliente é muito importante. A arquitetura de software proporciona vantagens para o desenvolvimento do software e seu produto final; dentre elas estão: maior facilidade de compreender inteiramente o produto (o relacionamento entre os elementos do produto e com os elementos externos) e de identificar mais facilmente as falhas.

Devido à necessidade de se adaptar aos sistemas legados de outras empresas, a integração entre o sistema de uma seguradora e seus parceiros é feita por troca de arquivos contendo as informações para a atualização das bases de dados.

Uma arquitetura de software que propõe a integração de sistemas por trocas de arquivos tem algumas facilidades e, dentre elas, tem-se: interagem em qualquer plataforma, pode ser desenvolvida em qualquer linguagem de programação, e não depende de regras de negócio.

A motivação para este trabalho está em analisar uma arquitetura que permita a interoperabilidade entre empresas, passando por cima das dificuldades e diferenças nas culturas empresariais.

1.2 Objetivo

Este trabalho tem como objetivo analisar uma arquitetura de software, pré definida, para a integração de sistemas através de arquivos para uma empresa de seguros e o funcionamento desta arquitetura no ambiente da seguradora.

1.3 Justificativas

Com o avanço tecnológico, as empresas estão utilizando cada vez mais os sitemas computacionais para facilitar, resolver e cadenciar necessidades e tarefas do dia-a-dia. Uma dessas tarefas é a troca de informações entre empresas, proporcionando dados, para que haja uma sincronização entre elas, devido a suas necessidades. Existem vários meios de compartilhar informação e todos devem levar em conta a diversidade de aplicações existentes, tecnologias, linguagens de programação e meios físicos de comunicação.

A interoperabilidade é a capacidade de um sistema para interagir e comunicar com outro de forma mais transparente possível. O conceito parece bem simples, porém na prática o que se vê muitas vezes são desenvolvimentos bastante complexos, a fim de possibilitar esta comunicação. (MARTINS,2010).

Uma iniciativa para essa interoperabilidade, ou integração, seria a troca de arquivos, levando em conta a necessidade de processamento de dados em lotes de uma empresa do ramo de seguros, recebendo informações diariamente e em horários definidos para o processamento de um todo.
As principais dificuldades desse tipo de integração estão na definição do formato para os arquivos e na sincronização das informações existentes com as novas, depois de cada importação de dados (MARTINS,2010).

No entanto, o autor também identifica como uma vantagem da integração por troca de arquivos, dispensar o software intermediário de integração, pois o sistema operacional já fornece quase toda a infra-estrutura necessária, requerendo talvez apenas um analisador para o formado escolhido, por exemplo, um parser XML.
Neste sentido, esta pesquisa procura trabalhar sobre a problemática: Como integrar vários sistemas diferentes para que eles trabalhem em conjunto e possam trocar informações?

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos, base para a construção do modelo de arquitetura proposto.

2.1 Arquiteturas de software

De acordo com Pressman (2009), a arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles.

De acordo com Martin Fowler (2006), na arquitetura existem dois elementos comuns: um é a decomposição de alto nível de um sistema em suas partes; o outro são decisões difíceis de alterar. Não há apenas um modo de especificar arquitetura de um sistema.

Uma arquitetura de software deve ter um estilo, tipo de padrão, que expresse a organização estrutural e o esquema fundamental para sistemas de software. Um estilo fornece o conjunto de elementos, suas responsabilidades e a relação entre eles.

De acordo com Rozanski e Woods (2008) o ponto chave sobre um estilo de arquitetura é que ele forneça um conjunto de princípios organizacionais para o sistema como um todo. Existem padrões de design, chamados de Design Patterns, que fornecem um esquema para aprimorar os elementos de um sistema de software ou as relações entre eles.

2.1.1 Estilos arquiteturais

De acordo com Rozanski e Woods (2008) padrões de software são organizados em três grupos: estilos arquitetônicos, padrões de design (design patterns) e expressões idiomáticas da língua que capituram soluções uteis para problemas específicos da linguagem.

Nick Rozanski e Eoin Woods (2008) dizem que um estilo arquitetônico expressa um esquema estrutural organizacional de nível de sistema. Não fornece os detalhes de partes do sistema e sim o demonstra como um todo, os tipos de elementos, as interfaces entre eles, restrições da forma que se conectam e como devem ser combinados.

De acordo com Pressman (2009), um estilo arquitetural é uma transformação imposta no projeto de software inteiro e tem como objetivo estabelecer uma estrutura para todos os componentes do software. Os componentes de software reestruturados para cumprir um estilo arquitetural resultará em mudanças fundamentais e incluirá novas funcionalidades dos componentes.

Existem vários estilos arquiteturais, um exemplo de deles é o estilo em camadas.

Estilos em camadas é um dos mais comuns de acordo com Fowler (2006), nele pode-se visualizar cada camada separadamente, imaginando subsistemas principais no software. A camada mais acima é chamada de camada de apresentação, essa camada utiliza os serviços disponibilizados pela acamada abaixo da camanda de apresentação, chamada de camada de negócio. A camanda de negócio utiliza os serviços disponibilizados pela camada abaixo da camada de negócio, chamada de camada de domínio e assim por diante como pode ser observado na Figura 1.

Figura 1. Visão de camadas de aplicação

Fonte: [Adriano de Pinho Tavares, 2013]

De acordo com Fowler (2006), dividir um sistema em camadas tem uma série de benefícios importantes:

- Pode-se compreender uma única camada como um todo, sem precisar entrar em detalhes das outras camadas.

- Pode-se substituir camadas por implementações alternativas dos mesmos serviços básicos.

- Pode-se minimizar as dependências entre as camadas.

- As camadas podem ter padrões para definirem bem como devem operar.

As camadas têm seu lado negativo também, como por exemplo:

- Qualquer alteração em alguma das camadas pode resultar em alteração em cascata, necessitando alteração em todas as outras, como por exemplo, em uma aplicação corporativa pode-se ter um acréscimo de um campo novo.

- O uso excessivo de camadas pode prejudicar o desempenho, pois em cada camada os dados precisariam ser transformados para a utilização em uma outra camada.

De acordo com Fowler (2006), a parte mais difícil de uma arquitetura em camadas é decidir quais camadas são necessárias e quais as responsabilidades de cada uma. O estilo em camadas mais utilizado é o que contém três camadas principais: apresentação, negócio ou domínio e acesso a dados ou fonte de dados.

A camada de aprensetação diz respeito de como tratar a interação entre o usuário e o software através de uma interface. Essa interface pode ser uma página web, uma aplicação no Windows ou até mesmo uma simples linha de comando. De acordo com Fowler (2006) as responsabilidades primárias da camada de apresentação são as exibições de informações para o usuário e traduzir comandos e ações do usuário sobre a camada de domínio.

A camada de acesso a dados ou fonte de dados diz respeito à comunicação com outros sistemas que executam as tarefas do interesse da aplicação em geral. Podem ser outras aplicações, sistemas de mensagens, sistemas de transação de dados, banco de dados e outros. A maioria das aplicações corporativas utilizam banco de dados como a camada de acesso a dados, deixando a camada responsável por armanezar os dados persistentes. Essa camada pode ser responsável também por enviar e/ou receber dados entres aplicações corporativas.

A camada de negócio, esta camada é responsável por tratar os dados, fazer cálculos baseado nas entradas e nos dados armazenados, validar os dados vindos da camada de acesso aos dados e tratar os dados de saída de acordo com a necessidade da regra de negócio.

Para este trabalho será analisada uma arquitetura em camadas com apenas duas camadas, camada de fonte de dados ou acesso aos dados utilizando a camada de File Transfer Protocol (FTP) como transação dos dados e camada de domínio ou de negócio utilizando um banco de dados e procedures para validações e aplicações das regras de negócio.

2.1.2 Padrões de design (*Design patterns)*

“Um padrão de design é uma solução para um problema muito mais específico relacionado com a estrutura de uma parte particular de um sistema”. (Nick Rozanski e Eoin Woods, 2008)

“Expressões linguísticas são o tipo mais específico de padrão de software, aplicando a situações em que uma linguagem de programação especial está em uso”. (Nick Rozanski e Eoin Woods, 2008)

De acordo com Fowler (2006) não existe uma definição universalmente aceita de padrões; segundo o autor: “Cada padrão descreve um problema que ocorre repetidamente no nosso ambiente e então descreve a essência da solução desse problema, de tal forma que você possa usar essa solução um milhão de vezes, sem jamais fazê-lo exatamente da mesma forma” (Alexander et al,2013). Alexander (2013) diz também que cada padrão é uma regra de três partes que expressa uma relação entre um certo contexto, um problema e uma solução.

Em relação à arquitetura de software, um padrão arquitetural expressa a forma de organizar as estruturas que compõem o sistema como um todo. De acordo com Buschmann e Henney (2007) padrões tornaram-se populares no design de um software: mudaram o modo que desenvolvedores de software pensam e praticam o design, fornecem com o design um vocabulário de software, ajudam a resolver problemas recorrentes construtivamente e baseada em soluções comprovadas e apoiam na compreensão da arquitetura de um sistema.

Os padrões arquiteturais definem um conjunto de sub-sistemas, as responsabilidades de cada sub-sistema e as regras de relacionamento entre eles. Um sistema pode ter vários padrões diferentes para situações diferentes.

Fowler(2006) citou vários padrões arquiteturais, como o padrão modelo de domínio, banco de dados relacionais, integração de sistemas por troca de arquivos, entre outros.

O padrão para a integração de sistemas é relacionado à troca de arquivos, que foca em uma comunicação off-line, sem a necessidade de serviços e acessos a web, conhecida como back-end; não possui a camada de apresentação, apenas a camada de negócio e de acesso aos dados, utilizando o estilo arquitetônico de camadas. Para este trabalho será adotado o padrão de troca de arquivos.

2.2 Visão corporativa

As aplicações corporativas, de acordo com Fowler(2006), dizem respeito à exibição, manipulação e armazenamento de grandes quantidades de dados, frequentemente complexos, e ao suporte ou automação dos processos de negócio que fazem uso desses dados.

De acordo com Polgar, Bram e Polgar (2006), existem três tipos de aplicações corporativas: Business-to-Business (B2B), Business-to-Costumer (B2C) e Business-to-Employee (B2E). Para este trabalho o tipo de aplicação corporativa utilizada foi o B2B.

2.2.1 Business-to-Business (B2B)

O Business-to-Business está associado às operações entre empresas. O Business-to-Business pode também ser definido como a troca de mensagem estruturada com outros parceiros comerciais, a partir de redes privadas ou da internet, para criar e transformar suas relações de negócios.

De acordo com Gandhi (2004), o desafio na interação B2B está na integração e interoperabilidade entre as duas aplicações e dados. Isto é devido aos sistemas serem heterogêneos. A heterogeneidade abrange tanto os hardwares quanto os softwares.

2.2.2 Interoperabilidade

Interoperabilidade é a capacidade de um sistema para interagir e comunicar-se com outro da forma mais transparente possível. O conceito parece bem simples, porém na prática o que se vê muitas vezes são desenvolvimentos bastante complexos a fim de possibilitar esta comunicação. (MARTINS, 2010)

As informações sobre como o ambiente em que as aplicações estão inseridas, a plataforma, a linguagem de programação e os protocolos de comunicações são essenciais para se buscar a melhor técnica a ser empregada nesta integração.

Segundo Goranson (1997), integrar é obter uma operação mais eficaz dos processos de negócio de uma empresa, compreendendo, entre eles, pessoas, máquinas e informação, de acordo com os seus objetivos.

Em outras palavras, a integração serve para facilitar o acesso à informação e, consequentemente, para melhorar a comunicação, cooperação e coordenação dentro da empresa, de forma que ela se comporte como um todo integrado (VERNADAT, 1996).

Basicamente, existem quatro formas de proporcionar a integração entre diferentes sistemas:

1) Transferência de Arquivos, em que cada aplicação produz arquivos de dados compartilhados para alimentar outras aplicações e vice-versa;

2) Base de Dados Compartilhada, em que as aplicações armazenam os dados que elas querem compartilhar em uma base de dados comum;

3) Chamada de Procedimento Remoto, em que cada aplicação disponibiliza alguns dos seus procedimentos para que eles possam ser chamados remotamente;

4) Troca de mensagens, em que cada aplicação se conecta a um sistema de troca de mensagens, através do qual pode trocar dados.

Em uma empresa, mesmo que pequena, muito dificilmente existe apenas uma aplicação. Mesmo que se opte por desenvolver uma aplicação única, diversos seriam os desafios que acabariam por inviabilizar a estratégia.

Para se prover uma experiência unificada, ou seja , uma transparência ao usuário sobre diversos sistemas como se fossem apenas um para funcionários, parceiros e clientes, deve-se considerar que integrações podem ocorrer entre soluções estruturadas em plataformas distintas, separadas geograficamente dentro e fora do escopo da organização e que usam tecnologias distintas.

De acordo com Hohpe e Woolf (2003) deve-se escolher uma abordagem de integração, levando em consideração alguns aspectos:

a) Formato de Dado: Para se integrarem, aplicações devem concordar em um formato de dado. Considerando que alterar todas as aplicações da organização para considerar um formato de dado único pode ser inviável, tradutores intermediários podem ser empregados. Outro assunto relacionado é como a evolução do formato de dados ao longo do tempo pode impactar as aplicações dependentes.

b) Seleção de Tecnologia: Diferentes abordagens de integração requerem diferentes quantidades de licenças de software e hardware. Tais ferramentas podem ser

caras e podem levar à dependência da organização com fornecedores específicos e ao aumento da curva de aprendizado dos desenvolvedores.

c) Exposição de funcionalidades: Muitas abordagens de integração permitem que aplicações compartilhem não apenas dado, mas também funcionalidades. Tal compartilhamento é interessante, pois gera um nível maior de abstração entre as aplicações envolvidas.

d) Tempo para Atualização: Integrações devem ser estruturadas pensando na minimização do tempo de defasagem de dado. Idealmente, aplicações consumidoras de dado deveriam ser informadas assim que o dado estivesse pronto para consumo. Quanto mais tempo se leva para o compartilhamento do dado, maior a probabilidade de falta de sincronismo de dados.

e) Processamento assíncrono: A chamada de funcionalidades remotas de forma síncrona pode ser algo custoso para a aplicação consumidora. A capacidade de realizar tarefas assíncronas traz diversas vantagens como, por exemplo, escalabilidade. Porém, tal solução tem design, desenvolvimento e depuração mais complexos.

f) Disponibilidade: Conexões remotas não são apenas lentas, mas também são muito menos confiáveis do que a execução de procedimentos locais. Aplicações remotas podem não estar disponíveis ou a rede pode estar temporariamente indisponível. Comunicações assíncronas e confiáveis permitem que a aplicação origem realize outras tarefas, de forma confiante que a aplicação destino receberá a informação.

g) Acoplamento entre aplicações: Aplicações integradas devem minimizar as dependências entre si, de forma que cada uma possa evoluir sem causar problemas para as demais. Integrações devem ser específicas o suficiente para cumprir seu papel, porém, genéricas o suficiente para garantir que mudanças não façam com que as aplicações dependentes parem.

h) Intrusividade: Integrações devem causar o mínimo de impacto em códigos existentes e devem requerer pouca codificação.

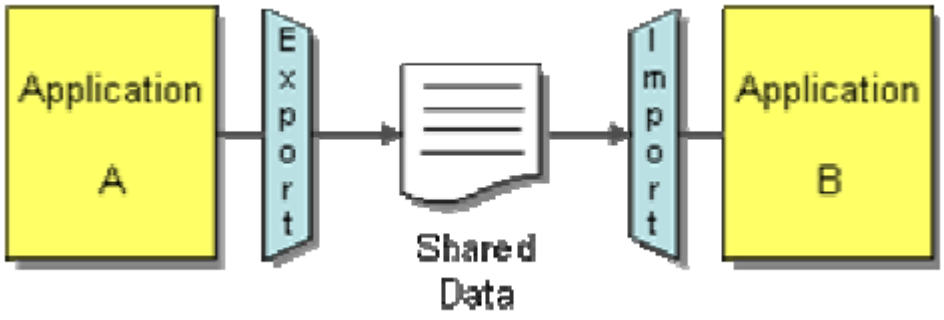
Levando em conta esses aspectos citados e também as necessidades da seguradora e seus parceiros, a integração por arquivos é a melhor opção de interoperabilidade. O formato de dado pode ser definido em qualquer sistema e facilmente tratado no momento da leitura, independe da tecnologia utilizada, grande facilidade no processamento assíncrono, o esforço para manutenção e desenvolvimento é baixo, instrusividade é baixa, as aplicações podem facilmente mudar sem qualquer dependencia entre elas.

Para este trabalho é utilizada a forma de integração por transferência de arquivos.

2.2.3 Transferência por arquivos

Na integração por transferência de arquivos, cada aplicação é responsável por produzir um arquivo que contenha as informações necessárias para a outra aplicação e de uma forma que ela seja capaz de interpretar essas informações. Esses arquivos são produzidos em intervalos regulares de acordo com a regra de negócio das empresas.
A Figura 2 mostra o fluxo da troca de arquivos entre a aplicação A e a aplicação B.

Figura 2. Transferência por arquivos



Fonte: (Hohpe e Woolf, 2003)

Os arquivos podem ser criados em qualquer sistema operacional, fazem parte de uma linguagem e modo de armazenamento universal. Mas, de acordo com Hohpe e Woolf (2003) uma decisão importante que deve ser tomada para a criação do arquivo, é o seu formato. As aplicações não precisam saber como cada uma funciona, apenas tratarem da mesma estrutura de arquivo, o local onde deve ser salvo, a periodicidade em que o evento de leitura ou escrita do arquivo ocorra, entre outras coisas que precisam ser levadas em consideração.

Quanto à estrutura do arquivo, é importante que as duas aplicações envolvidadas na comunicação concordem sobre seu formato. Existem alguns padrões de formato: arquivos com colunas delimitadas por caractere, arquivos com coluna fixa e arquivos estruturados por metadados, como por exemplo o Extensible Markup Language (XML).

De acordo com a W3C (2015) o XML é uma linguagem de marcação muito parecida com HTML (HyperText Markup Language) que foi projetada para armazenar e transportar dados de forma estruturada, podendo assumir várias estruturas que o usuário necessitar.

Figura 3. Nota em XML estruturada de uma nota de Tove para Jani

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Fonte: Introduction to XML. Disponível em: <http://www.w3schools.com/xml/xml_whatis.asp>

As aplicações precisam também estar de acordo com a semântica dos dados do arquivo. Cada campo requer uma análise para indentificar problemas no entendimento do modelo definido. Uma chave externa de uma aplicação pode ter o

modelo numérico e para outra pode ter o modelo alfanumérico, podendo haver problemas no processamento do arquivo.

Outra preocupação, que as aplicações devem ter, é o local em que o arquivo será gravado. Quando as aplicações estão no mesmo servidor ou mesma rede, o diretório pode estar localmente compartilhado. Quando as aplicações estão em redes e servidores diferentes há a necessidade de uma camada de transporte para transportar os arquivo. Um recurso muito comum utilizado na camada de transporte de arquivos é o File Transport Protocol (FTP).

O FTP é uma norma Transmission Control Protocol / Internet Protocol (TCP / IP), usado para efetuar login em uma rede, listar diretórios e copiar arquivos. Ou seja, ele fornece autenticação do usuário e permite a transferência de arquivos, listar diretórios, apagar e renomear arquivos na máquina estrangeira. (Gartner Group IT Dictionary, 2012)

Outro acordo, que deve ser feito entre as aplicações, é a periodicidade em que os arquivos serão gerados, disponibilizados e processados. A aplicação deve também se preocupar se o arquivo, que está sendo processado, já não foi processado anteriormente e assim deletar o arquivo quando tiver cumprido o seu papel.

De acordo com Hohpe e Woolf (2003), a grande vantagem da troca de arquivos é que as aplicações envolvidas não precisam conhecer os detalhes internos umas das outras, tendo assim uma solução de baixo acoplamento. Todas as preocupações estão diretamente relacionadas ao arquivo, por isso, as mudanças internas nas aplicações não acarretam problemas, desde que o processo de geração e leitura dos arquivos permaneça executando da mesma forma.

2.2.4 Leitura e armazenamento do arquivo

Date (2003) define um sistema de banco de dados como um sistema computadorizado de manutenção de registros; em outras palavras, é um sistema

computadorizado, cuja finalidade é armazenar informações e permitir que os usuários busquem e atualizem essas informações.

Elmasri (2011) definem um banco de dados como uma coelação de dados. Essa definição de banco de dados é bastante genérica; porém, o uso comum do termo banco de dados normalmente é mais restrito e tem as seguintes propriedades implícitas:

- Um banco de dados representa um aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – Universe of Discourse). As mudanças no minimundo são refletidas no banco de dados.
- Um banco de dados é uma coleção logicamente coerente de dados com algum significado inerente. Uma variedade aleatória de dados não pode ser corretamente chamada de banco de dados.
- Um banco de dados é projetado, construído e populado com dados para uma finalidade específica. Ele possui um grupo definido de usuários e algumas aplicações previamente concebidas nas quais esses usuários estão interessados.

Segundo Date (2003), um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa. É comum referir-se aos dados em um banco de dados como persistentes. Persistente sugeri intuitivamente que os dados desse banco de dados diferem em espécie de outros dados mais efêmeros, como dados de entrada, dados de saída, filas de trabalho, blocos de controle de software, etc.

Segundo Gonçalves (2012), ETL é uma sigla para Extract, Transform and Load, ou seja, extração, transformação e carga. ETL, em uma definição mais completa, é uma técnica ou processo de banco de dados em que os dados propriamente ditos são movidos das mais diversas origens e armazenadas em outros locais.

A extração é a primeira parte do processo de ETL, que diz respeito à extração das informações, determinando a origem, seus metadados, tamanhos e disponibilidades. Se a origem não cumprir determinadas regras, a carga já pode ser rejeitada na etapa

de extração. A segunda parte é a transformação, em que regras do processo podem ser aplicadas, seja selecionando apenas algumas colunas de dados de origem, conversão de caracteres de um determinado campo em outro padrão, derivando novas colunas, ordenando as informações, enfim, é neste ponto que a necessidade de negócio deve ser determinada. Por fim, a carga é o passo final, em que os dados são carregados no destino. Porém, sob diversos aspectos, pode ser muito interessante que a primeira parte do processo ocorra em uma área de stage, antes de chegar ao destino propriamente dito, evitando assim problemas de indisponibilidade.

LeBlanc (2013) define a ferramenta Microsoft SQL Server Integration Services (SSIS) como uma plataforma que permite a construção de estruturas de extração, transformação e carregamento (ETL – Extract, Transform, Load) de alto desempenho. Na maioria dos casos o SSIS é utilizado para ETL; entretanto, ela oferece várias tarefas e transformações que ampliam sua utilização muito além da ETL. Com o SSIS, é possível exportar ou importar dados de várias fontes rapidamente, incluindo Excel, outros arquivos de texto, Oracle e DB2.

Neste trabalho é utilizado, para leitura e armazenamento dos arquivos, um sistema de gerenciamento de banco de dado, o Microsoft SQL Server, utilizando-se também da ferramenta de ETL Microsoft SQL Server Integration Services para tratar das tarefas de Extract, Transform, Load (ETL - Extração, transformação e carga).

3. ANÁLISE DA VISÃO CORPORATIVA

Neste capítulo são apresentados os processos de negócio de uma seguradora e a proposta de arquitetura de software para atender as necessidades destes processos.

3.1 Visão corporativa

Uma seguradora de grande porte possui vários tipos de aplicações e essas aplicações estão distribuídas entre as várias áreas da empresa e também na web para clientes finais. Além das aplicações que exibem, manipulam e armazenam esses dados da própria corporação, há também os parceiros comerciais, outras corporações que trabalham em conjunto com a seguradora. Esses parceiros comerciais precisam trocar informações com a seguradora para que fiquem alinhados e coerentes em relação a seus clientes. Essas outras corporações podem ser bancos, outras seguradoras, grupos financeiros para soluções de créditos e outros.

Uma das características de negócio da seguradora é a venda de seus produtos via parceiros, como citados no processo de vendas. Esses parceiros são chamados de Sponsor ou parceiros patrocinadores, que cuidam de toda a venda dos produtos da seguradora. Outra característica é a utilização de uma aplicação web onde os clientes podem comprar produtos diretamente do site, sem a necessidade de um corretor para finalizar a venda.

A comercialização via Sponsor é muito comum no ramo, por isso é muito importante que a interoperabilidade entre os Sponsor e a seguradora seja feita de forma consistente, tendo em vista que o cliente poderá solicitar o uso do produto comprado na seguradora e não no Sponsor.

Dentre os processos de negócio da seguradora podem ser citados:

-Processo de vendas: No processo de vendas o seguro é oferecido ao cliente pelo parceiro, através de seus corretores; uma vez que o cliente aceita o produto vendido, o parceiro é responsável por gerar um arquivo com todas as vendas feitas no dia e enviar, para a seguradora, as informações de cada um dos clientes. Uma vez que esse arquivo é processado o cliente se encontra ativo na seguradora.

-Processo de cancelamento: O cliente pode pedir o cancelamento de seu seguro a qualquer momento. Neste processo o cancelamento pode ser feito tanto no parceiro que efetivou a venda ao cliente, quanto no call center da própria seguradora. Caso o cancelamento seja feito através do parceiro, ele fica responsável por gerar um arquivo de cancelamento com todas os clientes que o pediram e enviar para a seguradora.
No caso do cancelamento ser feito no call center da seguradora, cabe a ela gerar um arquivo de cancelamento no layout do parceiro, com todas as informações pertinentes ao parceiro para que esse segurado seja cancelado e que não haja mais cobranças ao cliente.

-Processos de cobrança: O processo de cobrança é controlado pelo parceiro, ele é o responsável por cobrar do cliente o valor do seguro mensalmente. O modelo de cobrança foi pré determinado no momento da venda. Diariamente o parceiro deve enviar para a seguradora um arquivo com todas as cobranças realizadas no dia anterior. Todo começo de mês a seguradora emite um documento com todas as cobranças recebidas pelo parceiro, para que sejam conferidos os valores, uma vez que o documento esteja correto o parceiro faz o pagamento de todo o arrecadado no mês para a seguradora.

Para este trabalho será analisádo o processo de vendas.

3.2 Gestão de TI

A seguradora tem uma área de tecnologia da informação, contando com infra-estrutura, arquiteto de software, área de desenvolvimento de novos projetos e uma área para sustentação dos projetos já implantados.

A área de infra-estrutura cuida da comunicação da seguradora com os Sponsor, diponibilizando acessos de FTP, permissões, links, cadastros e logins de usuários internos e outros. Cuida dos acessos aos usuários da própria seguradora, e conta também com um administrador de banco de dados, ou DBA (Data Base Administrator), que administra todos os bancos de dados da seguradora.

Na área de arquitetura de software, a seguradora conta com um arquiteto, responsável por todos os sistemas e sub-sistemas da empresa. O arquiteto deve conhecer as estruturas e deve participar das reuniões de novos projetos.

A área de novos projetos é composta por um gerente e de dois a três analistas de sistemas, que fazem o levantamento dos requisitos e necessidades dos usuários, além de acompanhar o desenvolvimento. O desenvolvimento é terceirizado, a segurado contrata empresas terceiras para o desenvolvimento do software, passando a documentação necessária.

A área de sustentação é composta por um gerente e de dois a três analistas de sistemas. A área é responsável por ajustes e correções dos sitemas já implantados e que estão em produção, ativos na seguradora. O papel da área é garantir o bom funcionamento dos sistemas meio ao dia-a-dia e falhas, erros ou defeitos que podem ocorrer.

3.3 Sistema Loader da Seguradora

O sistema responsável por todos os processos, citados na seção 3.1, se chama Loader. O Loader é uma base de dados inteligente, responsável pelo processamento dos arquivos recebidos, conversões, retornos, validação de regras de negócio, aceitação ou recusa de clientes e outros. O sistema Loader é maestro de toda a interoperabilidade entre os parceiros e os sistemas internos da seguradora, a entrada de dados passa por ele. O Loader não possui uma interface com usuário, por isso ele tem apenas duas camadas, camada de acesso a dados e camada de negócio. Como o usuário não tem uma interface com o Loader, ele fica

responsável por emitir alguns alertas importantes através de e-mails para o usuário, como por exemplo: erros no processamento de arquivos, clientes recusados pela regra de negócio e outros.

Outra particularidade do sistema Loader é a concentração de regras de negócio; contempla as regras de negócio da seguradora e também regras particulares de cada parceiro. O sistema Loader identifica o arquivo recebido sendo de algum parceiro específico e busca, em sua base, as regras desse parceiro e aplica para cada segurado do arquivo.

3.3.1 Requisitos funcionais

Os requisitos funcionais definem as funções de um sistema e seus resultados. O quadro 1 apresenta uma lista de requisitos funcionais criados para o sistema Loader, responsável por receber as informações de vendas e validar.

Quadro 1. Requisistos funcionais – Processo de vendas.

ID	Descrição
RF01	Leitura do arquivos de vendas do parceiro
RF02	Validação das informações via regra de negócio
RF03	Processamento das adesões e ativação dos clientes
RF04	Geração de relatório de vendas analítico
RF05	Geração de arquivos de retorno para os parceiros
RF06	Emissão de preview de faturamento para análise do parceiro
RF07	Emissão de alertas de erros no layout do arquivo para as áreas responsáveis
RF08	Emissão de alertas de adesões recusadas pela regra de negócio para as áreas responsáveis
RF09	Conversão de arquivos de adesão para o layout V8
RF10	Processamento dos arquivos de pagamento recebidos dos parceiros
RF11	Geração de parcelas para clientes pagantes
RF12	Emissão do faturamento mensal para outros sistemas internos da seguradora
RF13	Processamento dos arquivos de cancelamento recebidos dos parceiros
RF14	Efetivar o cancelamento dos clientes, que vieram no arquivo do parceiro
RF15	Gerar arquivo de cancelamento para os parceiros, de clientes cancelados internamente na seguradora via call center

Quadro 1. Requisitos funcionais do sistema Loader

3.3.2 Requisitos não funcionais

Os requisitos não funcionais estão relacionados às qualidades de software, com por exemplo: desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.

O quadro 2 apresenta uma lista de requisistos não funcionais criados para o sistema Loader.

Quadro 2. Requisitos não funcionais – Processo de vendas

ID	**Descrição**
RNF01	Interoperabilidade: O sistema deverá se comunicar com o SQL Server.
RNF02	Padrão: Toda a regra de negócio está implementada através de stored procedures no SQL Server
RNF03	Portabilidade: Todos os arquivos textos recebidos deverão ter o formato DOS\Windows ANSI
RNF04	Confiabilidade: O sistema Loader deve receber e processar os arquivos de vendas 7 dias por semana, 2 vezes ao dia.
RNF05	Segurança: usuários externos só terão acesso aos arquivos recebidos, processados e retornados pelo Loader, se lhes forem dadas a permissão pelo adimistrador do sistema junto ao gestor responsável pelo usuário.
RNF06	Segurança: usuários internos terão total acesso aos arquivos recebidos, processados e retornados pelo Loader.

Quadro 2. Requisitos não funcionais do sistema Loader

3.4 Descrição da arquitetura de Software do sistema de vendas

A seguradora hoje trabalha com um estilo arquitetônico de duas camadas (como citado na seção **2.1.1**), a primeira camanda é de acesso aos dados e a segunda camanda é a de negócio.

A camada de acesso aos dados é responsável por transitar as informações dos Sponsors para a seguradora, dentro da própropria seguradora e também da seguradora para os Sponsors. Os dados são transitados via arquivos em sua maioria no formato em extensão ".txt" (extensão para arquivos no formato de texto).

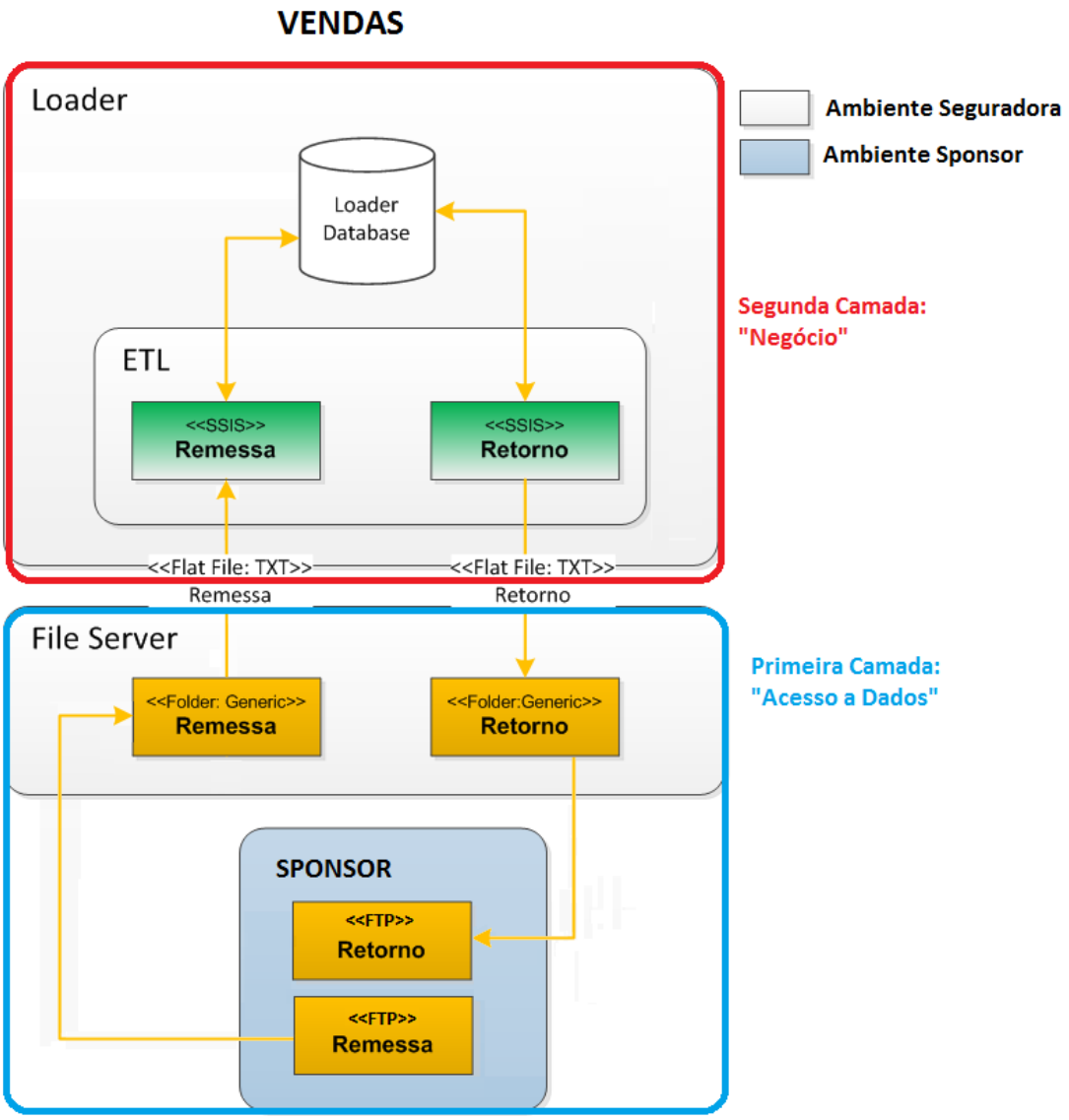
A camada de negócio é responsável por tratar os dados recebidos, transformando em informações e validar essas informações de acordo com as regras de negócio de

cada produto da seguradora. Esse processamento e essas regras de negócio são de resposabilidade do próprio banco de dados. Esse tipo de arquitetura não contém a camada de apresentação (citada na seção **2.1.1**).

No cenário atual, esse sistemas é responsável apenas pela interoperabilidade, não cabe a ele apresentar as informações, apenas transitar e validar elas.

A figura 4 apresenta a arquitetura de vendas.

Figura 4. Arquitetura de vendas



Fonte: [Arquiteto responsável pelos sistemas da seguradora]

3.4.1 Uso das camadas

Na primeira camada da arquitetura temos a transferência de informações que é feita entre a seguradora e os Sponsor vir arquivo (File transfer). Cada Sponsor ganha acesso, autenticado por um usuário e senha, a uma pasta no servidor da seguradora. Essa pasta tem um formato padrão, composta por alguns níveis:
-Nível 1, pasta "Receber": pasta onde o Sponsor deposita os arquivos com novas vendas.
- Nível 1, pasta "Retornar": pasta onde a seguradora depositar todos os arquivos de retorno para o Sponsor

-Nível 2, pasta "Receber -> Processados": pasta que contém todos os arquivos já processados pela seguradora.

-Nível 2, parta "Receber -> Erro" : pasta que contém todos os arquivos que tiveram erro de processamento por qualquer motivo.

-Nível 2, pasta "Retorno -> Processados: pasta onde o Sponsor move todos os arquivos de retorno já processados.

-Nível 2, parta "Retorno -> Erro" : pasta onde o Sponsor move todos os arquivos de retorno que tiveram erro de processamento por qualquer motivo.

As figuras 5 e 6 mostram a estrutura de pastas onde os Sponsors tem acesso e postam os arquivos de novas vendas por exemplo. Os arquivos ficam na pasta de nível 1 e são processados, caso o processamento seja feito com sucesso, ele é movido para a pasta de nível 2 - Processados, caso tenha algum erro com o arquivo ele é movido para a pasta de nível 2- Erro.

Figura 5. exemplo da estrutura de pastas nível 1- Receber

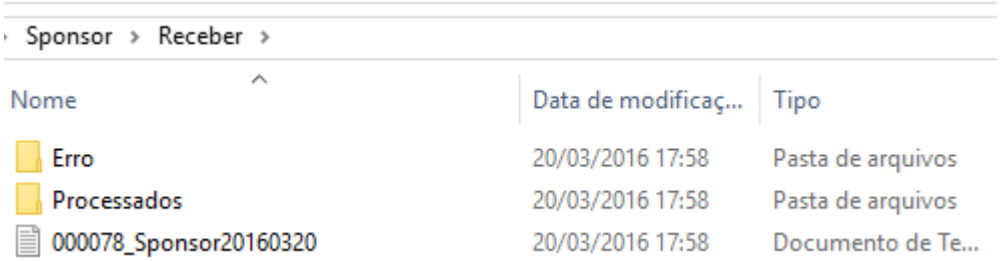
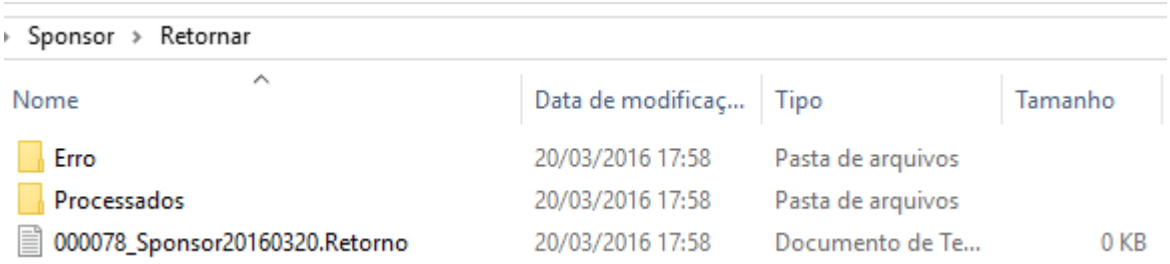


Figura 6. exemplo da estrutura de pastas nível 1- Retornar



3.4.2 Arquivos

Os arquivos são compostos por linhas delimitadas por tamanhos de colunas, onde cada posição do arquivo representa uma informação. Esse conjunto de linhas formam o layout oficial da seguradora, que hoje se encontra na versão 8 e é chamado de "V8".

Na figura 7 observa-se parte do documento que representa a formação dos arquivos onde cada posição representa uma informação.

Figura 7. Parte do documento oficial da seguradora para o layout de arquivos txt- V8

Detail (Informações Adicionais do Titular)							
No.	Nome do Campo	Comentário	Tipo	Tam	Obrig.	Inicio	Fim
01	Tipo de registro	3 – Informações adicionais do titular	N	0001	S	0001	0001
02	Id Segurado	Identificação do segurado titular.	C	0015	S	0002	0016
03	Carteira Nacional de Saúde	Carteira Nacional de Saúde.	C	0015	N	0017	0031
04	Declaração de Nascido Vivo	Número da Declaração de Nascido Vivo	C	0020	N	0032	0051
05	sExternalSponsorID	Preenchido pelo sponsor (Integração Sales Force)	C	0050	N	0052	0101
06	Flag Certificado	Flag identificando envio de certificado on-line	C	0001	N	0102	0102
07	Centro de Custo	Identificação da Concessionária / Terceiro	C	0010	N	0103	0112
08	Filler	Reservado	C	1888	N	0113	2000

No campo **01** tem-se o tipo de registro, ele tem início na coluna 1 do arquivo e termina na coluna 1 e seu tamanho é de 1 caracter. O layout informa também o tipo do registro, como por exemplo numérico, alfanumérico, data e outros. Outra informação importante que o layout mostra é a obrigatoriedade da informação. O Sponsor gerar um arquivo com extenção ".txt" apartir do documento "V8".

3.4.3 Carga dos arquivos

Uma vez que os arquivos são processados com sucesso, entra a segunda camada da arquitetura, que realiza a carga e leitura dos arquivos e a validação das informações através das regras de negócio.

A carga dos arquivos é realizada por um sistema intermediário, o Microsoft SQL Server Integration Services, que é o responsável pela execução de uma série de procedimentos (execução de stored procedures, carga de dados, scripts de linguagens de programação) para validação, persistência dos dados e movimentação de arquivos.

Após este processamento ETL, os dados validados são persistidos em uma base dentro do servidor de banco de dados SQL Server e é gerado um arquivo de retorno ao sponsor de acordo com o layout previamente definido entre as partes. Este arquivo segue pelo fluxo de retorno (citado no item **3.4**).

4 ANÁLISE DA ARQITETURA

Neste capítulo é feita a análise da adequação da arquitetura de software, para os negócios da empresa. Esta análise será dividida em análise técnica e análise de negócio.

4.1 Análise Técnica

A análise técnica leva em consideração os aspectos propostos por Hohpe e Woolf (2003):

a. Formato de Dado:

   As aplicações devem concordar sobre o formato de dados, uma vez que a troca de informações entre a seguradora e o parceiro é essencial para a definição dos formatos através de um documento de layout pré-definido. Este layout deve conter as informações do tipo de cada campo que será composto o registrado arquivo.

b. Seleção da Tecnologia:

   Para este tipo de arquitetura, a seleção da tecnologia é livre. O mais importante é que a tecnologia escolhida seja capaz de fornecer um arquivo texto no layout pré-definido e que possa eventualmente ler este arquivo e transformar suas informações em dados para sua base.

   A transação deste arquivo é feito pelo FTP, portanto os únicos pré-requisitos de tecnologia abordado nesta arquitetura é o arquivo no formato text e uma conexão via FTP.

c. Exposição de funcionalidades:

   Para este contexto e esse tipo de arquitetura não é necessário a exposição de funcionalidades, ou seja, caso a aplicação do parceiro queria invocar uma funcionalidade da aplicação da seguradora, este tipo de arquitetura não permite.

Portanto devem ser implementadas funcionalidades em cada uma das aplicações, caso seja necessário.

d. Processamento assíncrono:

O processamento assíncrono implica na não dependência dos sistemas estarem trabalhando ao mesmo tempo. O parceiro em qualquer momento pode gerar um novo arquivo de dados e disponibilizar para que a seguradora o processe assim que lhe for pertinente.

O sistema do parceiro pode estar indisponível em qualquer momento, não afetando o desempenho do sistema da seguradora que por sua vez trabalha independente do parceiro.

e. Acoplamento entre aplicações:

As aplicações neste formato de arquitetura são fracamente acopladas, não fazendo inúmeras suposições sobre como as aplicações funcionam. A aplicação do parceiro pode mudar radicalmente de formato que não irá afetar a aplicação da seguradora por exemplo.

f. Tempo para troca de dados:

De acordo com Hohpe e Woolf (2003), a arquitetura deve minimizar o período de tempo entre a integração dos dados entre as aplicações, o que significa disponibilizar pequenos lotes de dados com frequência. Esta arquitetura permite isso, porém por uma questões políticas internas, neste cenário entre seguradora e parceiro, a troca de arquivos ocorre uma vez ao dia, com lotes grandes de vendas.

4.2 Benefícios esperados - Análise do ponto de vista do Negócio

Tem-se, a seguir, uma lista de benefícios que a arquitetura pode trazer para a empresa:

- A troca de arquivos é uma forma de interoperabilidade muito utilizada entre os Sponsor e faz parte da cultura da seguradora; portanto, há pouco gasto no desenvolvimento de novas parcerias.
- Os Sponsors ganham acesso ao FTP através de um usuário e senha, e só terão permissão de Leitura\Escrita nas pastas que lhe forem pertinentes, o que torna o sistema seguro.
- Esse tipo de arquitetura permite que a seguradora e os Sponsors tenham seus próprios sistemas operacionais, não dependendo hardware e podem ter ambientes computacionais totalmente distintos e, ainda assim, é possível utilizar este tipo de arquitetura.

4.3 Dificuldades esperadas - Análise do ponto de vista do Negócio

Este tipo de arquitetura possui alguns dificuldades, tem-se, a seguir, uma lista das dificuldades esperadas:

- A diferença entre os layouts de arquivos é sempre uma dificuldade para a segurados. Nem todos os Sponsors estão abertos a utilizar o layout do arquivo proposto pela seguradora, o que acarreta uma implementação interna para diferentes tipos de layout. Este tipo de implementação exige tempo e dinheiro.
- Apesar de utilizar um acesso direto ao FTP, é possível que haja falhas na transmissão do arquivo e corromper as informações. Neste caso cabe à seguradora e ao Sponsor terem uma boa comunicação e pontos de alertas para esse tipo de problema seja corrido de forma rápida.
- Qualquer alteração no layout já utilizado, por exemplo inclusão de novos campos, exige uma implementação dos dois lados, seguradora e Sponsor.

- A regra de negócio, leitura dos arquivos e geração do retornos ficam centralizados na base de dados; portanto um bom servidor e um bom banco de dados são essenciais.

5. CONSIDERAÇÕES FINAIS

Este capítulo apresenta a conclusão da monografia, as contribuições que a monografia oferece em termos de pesquisa e uma sugestão de trabalhos futuros na mesma linha da pesquisa.

5.1. Conclusão

Este trabalho teve como objetivo identificar e mostrar os benefícios de uma arquitetura de software para a integração de sistemas por arquivos para uma empresa de seguros e seu funcionamento.

Após a revisão bibliográfica verifiocu-se que é importante a definição de uma arquitetura de software que foque na interoperabilidade, para uma empresa de grande porte, que trabalha com seguros e possui vários tipos de aplicações e essas aplicações estão distribuídas entre as várias áreas da empresa. Este tipo de arquitetura trará os seguintes benefícios: acompanhar o padrão de interoperabilidade dos parceiros; fácil e rápido desenvolvimento, baixo acoplamento entre as aplicações, baixo custo de manutenção, nenhuma dependência na seleção da tecnologia utilizada internamente em seus sistemas, processamentos assíncronos do parceiro entre outros.

Verificou-se que existem diferentes estilos arquiteturais; no entanto, a escolha de um determinado estilo deve levar em consideração a cultura da empresa, suas necessidades, e no caso da seguradora, a cultura do parceiro também foi levado em consideração. Assim, considerando as características de a venda de seus produtos via parceiros, a dependência da troca de informações entre seguradora e parceiro e outras, o estilo arquitetural mais adequado é o estilo em duas camadas.

A escolha de uma arquitetura em duas camadas trouxe os seguintes benefícios: simplicidade na manutenção, rapidez nos desenvolvimentos de novos parceiros, segurança nas informações que estão em diferentes níveis das camadas. Esta arquitetura fornece rapidez nas trocas de informações, mas não é totalmente

aproveitada pela seguradora por políticas internas. O principal item de benefício deste arquitetura para a seguradora é a interoperabilidade entre sistemas via arquivos, que é uma cultura da empresa e de seus parceiros.

5.2. Contribuição

A contribuição principal desta monografia foi analisar uma arquitetura de software utilizada em uma seguradora, capaz de resolver o problema da interoperabilidade entre os sistemas da seguradora com seus parceiros, levando em conta as tecnologias já utilizadas pelos parceiros e a cultura da seguradora.

5.3. Trabalhos futuros

Considerando que este trabalho focou nas camadas de dados e de negócio da arquitetura, fica como sugestão, para trabalhos futuros, propor um projeto para o desenvolvimento e tratamento da camada de interface com o usuário.

REFERÊNCIAS

ALEXANDER, Christopher; et al. Uma linguagem de Padrões: A Pattern Language. Porto Alegre: Bookman, 2013

BUSCHMANN, Frank; HENNEY, Kevlin In: Pattern-Oriented Software Architecture. 2007

CUNHA, Mônica Ximenes Carneiro da; JÚNIOR, Marcílio Ferreira de Souza; ALMEIDA, Hyggo Oliveira de In: Dificuldades com integração e interoperabilidade de sistemas de informações nas instituições públicas de ensino - um estudo de caso no CEFET-AL

DATE, C.K. Introdução a sistemas de banco de dados. Elsevier, 2003.

ELMASRI, Ramez; NAVA, Shamkant B. In: Sistemas de banco de dados. Pearson Addison Wesley, 2011

FIL- File Transfer. Disponível em < http://www.eaipatterns.com/FileTransferIntegration.html>.Acesso em 16 de Dezembro de 2015.

FOWLER, Martin. **Padrões de Arquitetura de Aplicações Corporativas**. Bookman, 2006

GANDHI, Saumil In: A Serive-Oriented Approach to B2B Integration Using Web Services, 2004.

Gartner Group . Garter IT Dictionary. Disponível em: <http://www.gartner.com/it-glossary>. Acessado em: 15 de julho de 2016

GONÇALVES, RODRIGO RIBEIRO. Integração de dados na prática: técnicas de ETL para Business Intelligence com Microsoft Integration Services 2012. Érica, 2012.

GORANSON, H. T. Human factors and enterprise integration. Workshop 1 Report ICEIMT, 1997

HOHPE, Gregor; WOOLF , Bobby. Indotruction, In: **Enterprise Integration Patterns.** Addison Wesley. 2003.

LEBLANC, P. Microsoft SQL Server 2012 – Passo a Passo. Bookman, 2014.

MARTINS, Ricielli Pelissoli In: INTEROPERABILIDADE ENTRE APLICAÇÕES .NET, 2010

MICROSOFT In: Integration Patterns, 2004

POLGAR, J.; BRAM, R. M.; POLGAR A. Building and Managing Enterprise Wide-Portals. 2006

PRESSMAN, R. **Engenharia de Software 6ª Edição**. Markron Books. 2009

ROZANSKI, N; WOODS, E In: **Software Systems Architecture**. 2008

SORDI, José Osvaldo De; MARINHO, Bernadete Lourdes In: Integração entre Sistemas: Análise das Abordagens Praticadas pelas Corporações Brasileiras, 2006

W3C. W3Schools. XML. Disponível em: <http://www.w3schools.com/xml/xml_whatis.asp>. Acessado em 02 de Fevereiro de 2016