CENTRO UNIVERSITÁRIO FEEVALE

MARCO AURÉLIO BRAUN

MODELO DE INTEGRAÇÃO ENTRE DIFERENTES SOFTWARES APLICATIVOS POR UM BARRAMENTO DE SERVIÇOS CORPORATIVOS BASEADO EM UMA ARQUITETURA ORIENTADA A SERVIÇOS

Novo Hamburgo
2009

MARCO AURÉLIO BRAUN

MODELO DE INTEGRAÇÃO ENTRE DIFERENTES SOFTWARES APLICATIVOS POR UM BARRAMENTO DE SERVIÇOS CORPORATIVOS BASEADO EM UMA ARQUITETURA ORIENTADA A SERVIÇOS

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel
em Sistemas de Informação pelo
Centro Universitário Feevale

Orientador: Edvar Bergmann Araujo

Novo Hamburgo
2009

MARCO AURÉLIO BRAUN

Trabalho de Conclusão do Curso de Sistemas de Informação, com título Modelo de Integração Entre Diferentes Softwares Aplicativos Por um Barramento de Serviços Corporativos Baseado em Uma Arquitetura Orientada a Serviços, submetido ao corpo docente do Centro Universitário Feevale, como requisito necessário para obtenção do Grau de Bacharel em Sistemas de Informação.

Aprovado por:

_____
Professor Edvar Bergmann Araujo

_____
Professor Gabriel da Silva Simões

_____
Professor Juliano Varella de Carvalho

Novo Hamburgo, dezembro de 2009.

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

À minha esposa e filhos, minha gratidão, pelo apoio emocional e compreensão pelos momentos de ausência - dedicados à realização deste trabalho.

RESUMO

Durante anos, as empresas desenvolvedoras de software geraram uma incontável quantidade de sistemas, baseados nas mais diversas linguagens de programação e arquiteturas. Esses sistemas são o resultado da demanda gerada a partir da evolução dos negócios corporativos. Muitas empresas ainda mantêm sistemas antigos em pleno funcionamento, convivendo com softwares aplicativos baseados em tecnologias mais modernas. Uma alternativa que pode ser aplicada na integração de todos esses softwares é a Arquitetura Orientada a Serviços - *Service Oriented Architecture* (SOA). SOA combina regras de negócios entre aplicações através de serviços intercambiáveis. Um componente ou ferramenta integrante de SOA é o Barramento Corporativo de Serviços – *Enterprise Service Bus* (ESB). O ESB tem a capacidade de fazer com que sistemas heterogêneos comuniquem-se de forma transparente. Sendo assim, este trabalho tem o objetivo de estudar, definir o funcionamento e gerar um modelo de um ESB. Dessa forma, servir de referência às empresas desenvolvedoras de software, promovendo o reuso de componentes (linhas de código) e combinando recursos já existentes, no intuito de dar mais agilidade às mesmas, com objetivo de que atinjam uma vantagem estratégica.

Palavras-chave: SOA. ESB. Integração de Sistemas. Arquitetura de Software.

ABSTRACT

For many years, companies in software development created a countless number of systems, based in several programming languages and architectures. These systems are the result of demand generated from the development of corporate business. Many companies still have systems in full operation, living with software applications based on latest technology. An alternative that can be applied in the integration of all such software is the Service Oriented Architecture (SOA). SOA combines business rules between applications through services interchangeable. An integral component or tool of SOA is the Enterprise Service Bus (ESB). The ESB has the ability to communicate with heterogeneous systems that are transparent manner. Thus, this work aims to study, to define the operation and to generate a ESB model. So, it serves the companies as a reference for companies in software development, promoting the reuse of components (lines of code) and combining existing resources in order to give more flexibility to them, with the aim of reaching a strategic advantage.

Key words: SOA. ESB. Systems Integration. Software Architecture.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BPM	*Business Process Modeling*
CMMI	*Capability Maturity Model Integration*
DLL	*Dymanic Link Library*
EAI	*Enterprise Application Integration*
ESB	*Enterprise Service Bus*
FTP	*File Transfer Protocol*
HTTP	*HyperText Transfer Protocol*
J2EE	*Java 2 Enterprise Edition*
MBA	*Master in Business Administration*
SMTP	*Simple Mail Transfer Protocol*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*

SUMÁRIO

INTRODUÇÃO

Nos últimos anos, a tecnologia da informação sofreu fortes e rápidas mudanças. As empresas desenvolvedoras de software geraram uma infinidade de sistemas, a partir das mais variadas tecnologias (linguagens de programação, arquiteturas...). "O atual panorama corporativo é composto de sistemas que são o resultado da evolução dos negócios e da tecnologia com o passar dos anos." (CUMMINS, 2002, p.2).

Os sistemas legados[1] utilizam-se das mais variadas formas de integração, com o uso de tecnologias diferentes. Esse cenário é assim descrito:

> "Após muitos anos de desenvolvimento de aplicações de computador em um panorama tecnológico em evolução, a maioria das empresas está carregada de computadores e aplicações que estão interligados através de uma variedade de mecanismos específicos." (CUMMINS, 2002, p.XIII).

Essa diversidade acaba por determinar um forte acoplamento[2] entre as aplicações, com uma estrutura muito rígida de comunicação. Para cada interligação entre aplicações, existe uma tecnologia, forma ou meio diferentes. Isso acaba tornando difícil a manutenção e exigindo mais recursos.

> "Os grandes sistemas utilizam plataformas diferentes, linguagens de programação diferentes (e paradigmas de programação) e até *middleware*[3] diferente. Eles são uma bagunça de *mainframes*, clientes SAP, bancos de dados, aplicações J2EE, pequenos motores de regras, e assim por diante. Em outras palavras, eles são heterogêneos." (JOSUTTIS, 2008, p. 13).

Além dos sistemas legados, desenvolvidos por uma mesma empresa, há também os casos em que empresas diferentes têm a necessidade de realizar a integração entre seus aplicativos, para atingir um objetivo em comum, que é a satisfação da necessidade de um cliente em comum. "SOA inclui práticas e processos que são baseados no fato de que redes de sistemas distribuídos não são controlados por um único proprietário. Equipes diferentes, departamentos diferentes ou até empresas diferentes podem gerenciar os sistemas diferentes." (JOSUTTIS, 2008, p. 13).

Dessa forma, uma empresa escreve seus softwares aplicativos alicerçados em uma determinada plataforma e em uma linguagem de programação específica, além de manter os dados dos mesmos em um banco de dados em particular. Na maioria dos casos, essas

---

[1] "Termo normalmente utilizado pra fazer referência a hardware ou software antigos que não obedecem aos padrões ou níveis de desempenho mais recentes" (SAWAYA, 2002, p.261).
[2] Alta dependência entre softwares aplicativos integrados (JOSUTTIS, 2008).
[3] "O *middleware* gerencia a interação entre aplicativos por meio de plataformas de computadores heterogêneos [...]" (SAWAYA, 2002, p.296).

características não são reproduzidas em um todo pela outra empresa. Mesmo que haja semelhanças de plataforma, linguagem ou banco de dados, as maneiras de como esses dados estão estruturados são diferentes. Inclusive, existem cadastros que são repetidos em dois ou mais aplicativos, não podendo assim ser compartilhados.

Uma alternativa estudada é a *Arquitetura Orientada a Serviços - Service Oriented Architecture* (SOA). SOA pode ser conceituada como "um estilo de arquitetura de sistemas de informação que habilita a criação de aplicações que são construídas com a combinação de acoplamento leve com serviços intercambiáveis." (SANTOS, 2007, p. 3).

Josuttis (2008) afirma que SOA é baseada em três conceitos técnicos principais: serviços, interoperabilidade através de um barramento corporativo de serviços (*Enterprise Service Bus* - ESB) e acoplamento fraco.

Sendo assim, pretende-se com este trabalho, contribuir na forma de material teórico e prático sobre os temas abordados, que são de um referencial bibliográfico (livros) um tanto escasso. No meio acadêmico o assunto é relativamente novo e uma desafiadora fonte de pesquisa.

Além do que, este estudo pode ser de relevante valia às empresas desenvolvedoras de software, ajudando a minimizar os investimentos e esforços para realizar a integração entre softwares legados. Pode ainda contribuir na promoção do reuso e compartilhamento de componentes, combinando recursos de maneira rápida, dando uma resposta ágil a uma exigência de mercado, possibilitando a elas a obtenção de uma vantagem estratégica. Dentro desse contexto, têm-se como exemplo, sistemas de folha de pagamento e controle de frequência de funcionários (ponto eletrônico) que podem ser integrados de forma mais eficiente. A integração desses aplicativos pode ocorrer de forma que haja um fraco acoplamento entre eles, através de um barramento de troca de dados. Assim, um aplicativo não precisa saber da existência do outro. O barramento é o responsável por garantir que um arquivo gerado por um aplicativo seja entregue e compreendido pelo outro.

Para o desenvolvimento deste estudo, realizou-se uma pesquisa aplicada, com abordagem qualitativa e com objetivo experimental. Segundo Prodanov e Freitas (2009), do ponto de vista da sua natureza, a pesquisa aplicada tem como objetivo gerar conhecimento para aplicação prática, a fim de solucionar problemas específicos, envolvendo verdade e interesses locais. Já do ponto de vista de seus objetivos, a pesquisa experimental acontece quando "determinamos um objeto de estudo, selecionamos as variáveis que seriam capazes de influenciá-lo, definimos as formas de controle e de observação dos efeitos que a variável produz no objeto" (PRODANOV e FREITAS, 2009, p.71). Segundo Prodanov e Freitas

(2009, p.81), “na abordagem qualitativa, a pesquisa tem o ambiente como fonte direta de dados. O pesquisador mantém contato direto com o ambiente e o objeto de estudo em questão”.

Este trabalho está dividido em seis capítulos. O capítulo 1 aborda a tecnologia SOA. O capítulo 2 esmiúça as funcionalidades de um ESB, conceitua e descreve suas características. O objetivo do capítulo 3 é descrever algumas outras formas de integração entre softwares aplicativos, além de vantagens e desvantagens de cada uma delas. Já o capítulo 4 mostra estudos já realizados sobre SOA e ESB, relatando os resultados obtidos pelos mesmos, bem como os desafios e dificuldades encontrados durante as suas realizações. Como forma de identificar o principal problema enfrentado pela empresa escolhida, o capítulo 5 apresenta os processos atuais de integração utilizados por ela. Finalmente, no capítulo 6 é proposto um modelo para um ESB, utilizando-se de ferramentas para modelagem. Ainda, no mesmo capítulo é descrita a implementação de um protótipo, objeto pelo qual se realizou a avaliação de desempenho do modelo de ESB proposto.

1 SERVICE ORIENTED ARCHITECTURE (SOA)

O presente capítulo tem como objetivo apresentar os principais conceitos associados à *Service Oriented Architecture* (SOA), suas características e importância para as organizações. Além disso, apontar em quais situações essa tecnologia pode ser empregada e relacioná-la com o contexto do problema apresentado anteriormente. E ainda, referir o seu uso por meio do *Enterprise Service Bus* (ESB). Cabe salientar que, devido à escassez de materiais formais e reconhecidos sobre o assunto, a obra de Josuttis (2008) foi a principal fonte de consulta.

1.1 CONCEITO

Josuttis (2008) escreve que encontrar uma definição exata do termo SOA é difícil, isto porque existem muitas definições diferentes, porém todas concordam que SOA é um paradigma para melhorar a flexibilidade. A seguir são apresentados alguns conceitos de SOA.

Em Santos (2007, p.3), verifica-se que SOA pode ser conceituada como “um estilo de arquitetura de sistemas de informação que habilita a criação de aplicações que são construídas com a combinação de acoplamento leve com serviços intercambiáveis.”.

Já Hurwitz (2007 apud BENEDETE JUNIOR, 2007, p.7) se refere a SOA, afirmando que

> “é uma arquitetura de software voltada para a construção de aplicações que implementam processos de negócio ou serviços utilizando um conjunto de componentes “caixa-preta”, fracamente acoplados, e orquestrados para prover um nível de serviço bem definido”.

Uma terceira definição encontrada é a dada por Bierberstein (2006 apud BENEDETE JUNIOR, 2007, p.7):

> “Arquitetura orientada a serviços é uma estrutura (*framework*) para integrar processos de negócio e a infra-estrutura de TI que o suporta, na forma de componentes seguros e padronizados – serviços – que podem ser reutilizados e combinados para endereçar as mudanças de prioridade do negócio”.

Inclusive, para Josuttis (2008), SOA não é uma arquitetura concreta: é algo que conduz a uma arquitetura concreta. Pode-se chamá-la de estilo, paradigma, conceito,

perspectiva, filosofia ou representação. SOA não é uma ferramenta ou *framework*[4] que se possa comprar. É uma abordagem, uma maneira de pensar, um sistema de valores que leva a certas decisões concretas quando se projeta uma concreta arquitetura de software.

1.2 CARACTERÍSTICAS

Santos (2008, p.3) descreve algumas características de SOA:

> "Em um ambiente SOA, recursos em uma rede são disponibilizados como serviços independentes que podem ser acessados sem o conhecimento de como foi implementado internamente. [...] A chave é a independência de serviços que definem interfaces que podem ser chamadas para executar tarefas específicas por meio de um padrão, sem que o serviço tenha conhecimento da aplicação solicitante, e sem que a aplicação tenha conhecimento de como o serviço executará a tarefa".

Por outro lado, Josuttis (2008, p.12) cita a flexibilidade como principal característica esperada por SOA, quando afirma que "a principal razão para usar SOA é que ela deve ajudá-lo em seu negócio. Por exemplo, no caso de precisar de soluções de TI que guardem e gerenciem os seus dados e permitam automatizar os processos comuns que lidam com esses dados".

1.3 CARACTERÍSTICAS TÉCNICAS

A fim de melhor compreender SOA, devem-se observar suas principais características, que são: serviços, alta interoperabilidade e acoplamento fraco.

1.3.1 Serviços

Um dos principais objetivos de SOA é basear os sistemas na abstração das regras e funções de negócio. Em Josuttis (2008) tem-se que, na essência, um serviço é uma representação da TI de alguma funcionalidade do negócio. Ou seja, serviço é a representação

[4] Conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação (FAYAD, 1997).

de uma abstração. O principal objetivo de um serviço é representar um passo natural da funcionalidade do negócio. Dependendo do domínio para qual ele é fornecido, deve representar uma funcionalidade independente que corresponde a uma atividade de negócio do mundo real, como por exemplo, ‘criar um cliente’, ‘obter contratos de um cliente’, ‘transferir dinheiro’.

Nota-se que, nesse contexto, a tecnologia usada para prover o serviço (um software rodando como serviço – *service*- no sistema operacional), não faz parte dessa definição.

De acordo com Josuttis (2008), os serviços podem ser classificados em três categorias:

- **Serviços básicos** – primeiro estágio de expansão (SOA fundamental), que provê apenas serviços correspondentes a funcionalidades básicas de negócio;
- **Serviços compostos** – segundo estágio de expansão (SOA federativa), os quais representam serviços compostos por outros serviços. Na terminologia SOA, isso se chama orquestração;
- **Serviços de processos** – o terceiro estágio de expansão (SOA habilitada para processos) representa os *workflows* ou processos de negócio de longo prazo. Diferente dos serviços básicos e compostos, um serviço de processo tem um estado que se mantém estável durante múltiplas chamadas. Um exemplo disso é o carrinho de compras localizado nos sites de compras.

A figura 1.1 ilustra essa classificação.

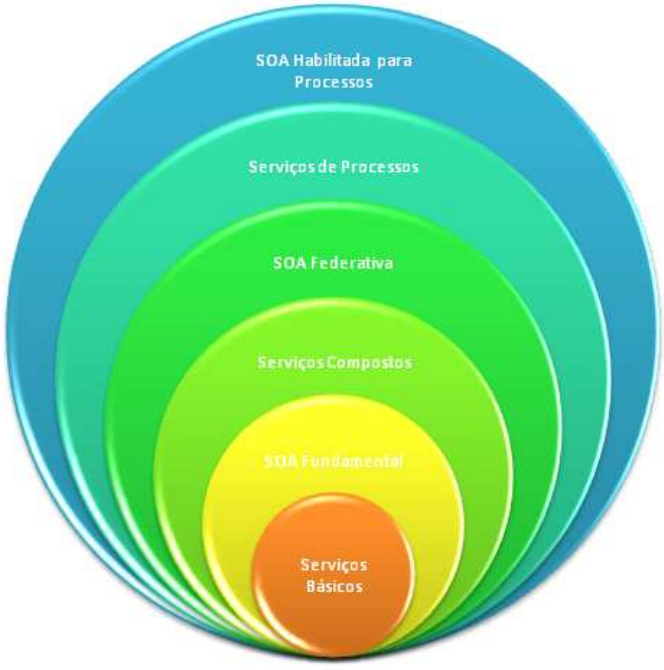

Figura 1.1 – Camadas básicas de serviços e estágios de expansão SOA
Fonte: JOSUTTIS, 2008, p.56 (adaptado pelo autor)

1.3.2 Alta interoperabilidade

Josuttis (2008) escreve que diferentes sistemas utilizam plataformas diferentes, linguagens e paradigmas de programação diferentes, assim como diferentes bancos de dados e motores de regras, ou seja, eles são heterogêneos. Uma das características mais fortes de SOA é a aceitação dessa heterogeneidade – ela lida e suporta essa diversidade.

Conforme Josuttis (2008), com os sistemas heterogêneos, o primeiro objetivo é ser capaz de conectar esses sistemas facilmente. Isso é normalmente chamado de alta interoperabilidade. Para SOA isso é o começo e não o fim. É a base a partir da qual as funcionalidades de negócio (serviços) são implementadas.

1.3.3 Acoplamento fraco

Acoplamento fraco é o conceito de minimizar as dependências. Quando as dependências estão minimizadas, as modificações têm os efeitos minimizados e os sistemas ainda executam quando partes deles estão quebradas ou indisponíveis. Minimizar as dependências contribui para tolerância à falhas e flexibilidade (JOSUTTIS, 2008).

A escalabilidade e a tolerância à falhas são as chaves para a manutenibilidade dos sistemas. Outro objetivo importante é minimizar o impacto das modificações e das falhas dentro do cenário do sistema como um todo. Dessa forma, o acoplamento fraco é um conceito-chave de SOA (JOSUTTIS, 2008).

1.4 IMPACTO NAS ORGANIZAÇÕES

Agilidade é um dos atributos mais importantes das empresas de hoje, pois possibilita aproveitar as oportunidades do mercado antes que a concorrência o faça. Para conseguir essa flexibilidade, uma possível abordagem é a empresa analisar sua operação de negócio como um conjunto de funções interconectadas (processos e serviços). As funções chaves para o negócio devem ser identificadas e trabalhadas, de forma a se tornarem um diferencial positivo no mercado (BENEDETE JUNIOR, 2007).

Segundo Benedete Junior (2007, p.17),

> "Através de sua habilidade de escalar e evoluir, SOA habilita um portfólio de TI que se adapta às várias necessidades de um domínio específico de problema ou arquitetura de processo. Portanto SOA pode prover uma sólida fundação para a agilidade e adaptabilidade do negócio".

Sendo assim, SOA permite que as empresas obtenham melhores resultados por torná-las focadas em suas competências chaves; ágeis em atender as demandas dos clientes; resilientes aos impactos das mudanças no negócio ou nas tecnologias; e integradas com seus clientes, parceiros e fornecedores (BENEDETE JUNIOR, 2007).

1.4.1 Departamentos

As estruturas dominantes das empresas são os departamentos, que mantêm sistemas específicos que cresceram ao longo dos anos (JOSUTTIS, 2008). Isso implica que, quando essas empresas necessitam de uma nova funcionalidade em algum desses sistemas, entregam a tarefa ao departamento correspondente. Por sua vez, esse departamento implementa a nova funcionalidade no seu sistema específico, a qual fica restrita ao mesmo.

Hoje em dia, a integração e a distribuição estão se tornando cada vez mais importantes. Então, embora as empresas ainda tenham estruturas monolíticas orientadas a departamentos, estes e os sistemas estão começando a trabalhar juntos (JOSUTTIS, 2008).

1.4.2 Gerenciamento

Por impactar em diversos departamentos, unidades de negócio ou empresas, o desenvolvimento de qualquer nova funcionalidade deve ser tratado como um projeto. Por esse motivo, um papel fundamental em SOA é o de gerente da solução (ou do projeto). Esse gerente tem como competência inicial, criar uma modelagem de alto nível para determinar o impacto nos sistemas existentes (JOSUTTIS, 2008).

O suporte do gerente da solução é também necessário durante a realização da mesma, porque na prática as coisas sempre se mostram diferentes do esperado. O perigo com

aplicações distribuídas também é que qualquer sistema pode quebrar toda a solução (JOSUTTIS, 2008).

1.4.3 Colaboração

Pelo fato de envolver diferentes departamentos (unidades de negócio ou empresas), a colaboração é apontada com um requisito chave. De acordo com Josuttis (2008, p.91),

> "desde a formulação de uma idéia até a manutenção da sua realização, os sistemas distribuídos requerem a colaboração. É claro que a colaboração é difícil nas organizações que consistem de departamentos que se comportam como territórios isolados. Nestes casos, mudar a cultura da empresa será o fator chave para o sucesso de SOA".

1.5 DESEMPENHO

Os sistemas de TI têm dois aspectos que repetidamente quebram planos, conceitos e modelagens: desempenho e segurança (JOSUTTIS, 2008).

Josuttis (2008, p.137) evidencia a necessidade da observância do desempenho, quando afirma que "há diferentes lugares onde o desempenho entra em jogo no contexto de serviços e infra-estrutura de SOA. Se o desempenho se torna crítico, isso é normalmente por causa do tempo de execução de um serviço".

Uma vez que SOA é um conceito para cenários de sistemas heterogêneos, não se pode simplesmente enviar dados binários de um sistema a outro, pelo motivo que o código compilado de uma aplicação não combina com o de outra. Por este motivo, obrigatoriamente, é necessário o uso de um protocolo intermediário conhecido tanto pelo consumidor[5] quanto pelo fornecedor[6].

Um formato possível é o baseado em XML[7]. Porém, de acordo com Josuttis (2008, p.138), esse formato pode ser um problema: "XML gera muita informação e o resultado disso

[5] Sistema que chama um serviço (usa um serviço fornecido) (JOSUTTIS, 2008).
[6] Sistema que implementa um serviço (uma funcionalidade de negócio) de tal forma que outros sistemas possam chamá-lo (JOSUTTIS, 2008).
[7] eXtensible Markup Language. Uma notação de propósito geral e de leitura fácil para humanos amplamente usada para a descrição e troca de dados (JOSUTTIS, 2008).

é que os dados essenciais podem ser aumentados de 4 a 20 vezes. Isto pode ter um impacto significativo na banda de rede em ambientes onde a mesma é limitada".

A adoção de um protocolo intermediário, baseado na serialização e desserialização, pode igualmente ser crítico, considerando-se o tempo que o processo de transformar uma sequência de bytes em dados complexos pode levar.

De qualquer maneira, seja qual for o protocolo utilizado, de acordo com Josuttis (2008, p.139), "é uma boa idéia colecionar as estatísticas sobre os tempos de execução de diferentes requisições de serviço e monitorar os números ao longo do tempo. Isso pode ajudá-lo a identificar os problemas mais imediatos e as tendências futuras".

1.5.1 Reusabilidade

A reusabilidade, apontada como virtude de SOA pode acabar por se tornar um problema de desempenho. Josuttis (2008) ilustra um exemplo de um sistema que fornecia um serviço responsável por retornar todos os dados do cliente - o que correspondia a centenas de atributos (Figura 1.2). Certa feita surgiu um novo sistema que necessitava de apenas alguns dados dos clientes. A fim de se atender ao requisito de usabilidade, optou-se pelo reuso do serviço existente (que retornava todos os dados do cliente).

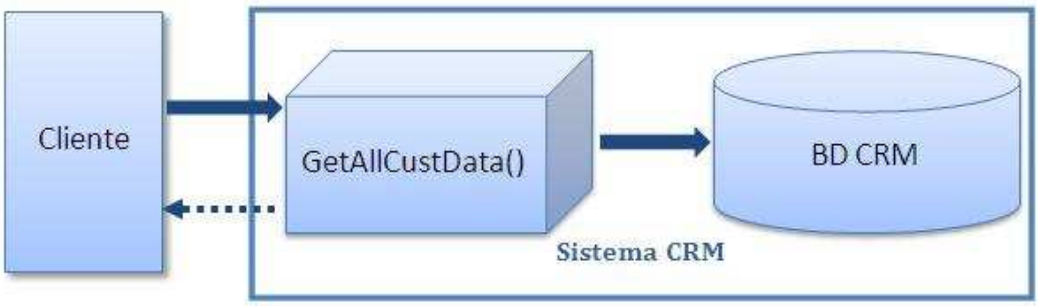

Figura 1.2 – Carregar todos os dados do cliente leva tempo
Fonte: JOSUTTIS, 2008, p.143

Ocorre que, imediatamente, o novo consumidor reclamou da demora no atendimento de sua solicitação. É claro que processar um resultado que retornava apenas os atributos desejados seria muito mais rápido do que a resposta usual que continha o portfólio completo do cliente.

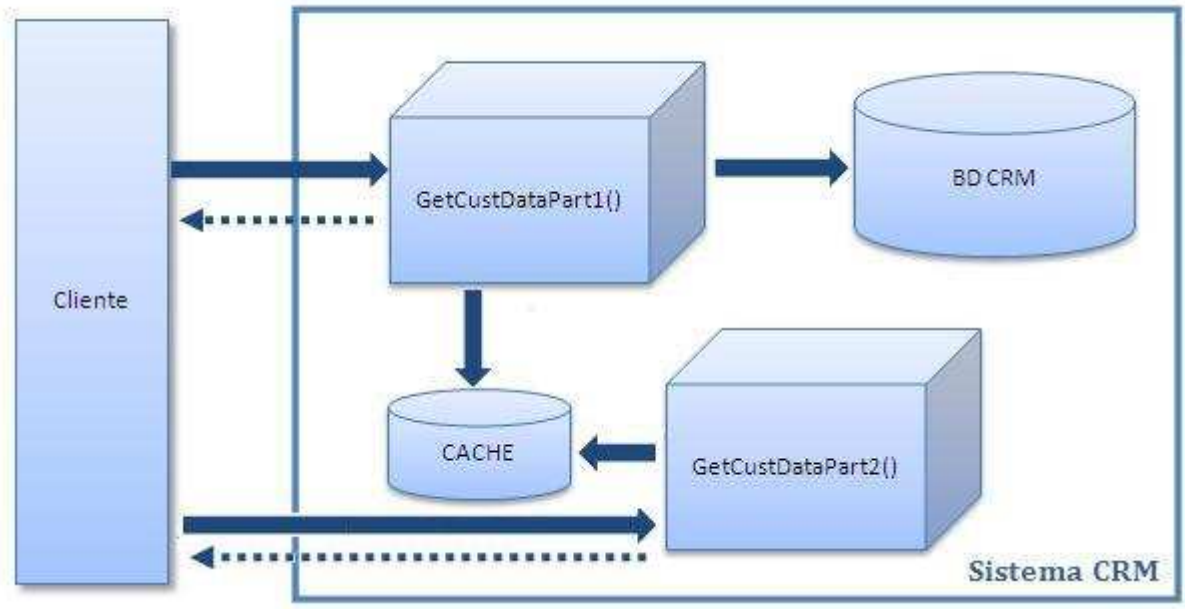

Figura 1.3 – Carregar dados de cliente em dois passos
Fonte: JOSUTTIS, 2008, p.144

A figura 1.3 mostra como o problema de desempenho pode ser resolvido, através da implementação de um método que retornasse apenas os dados de interesse do novo consumidor. Mas isso não significa que o objetivo de reusabilidade nunca possa ser atingido (JOSUTTIS, 2008).

1.5.2 Customização

O exemplo citado expõe o fato de que “sistemas diferentes freqüentemente precisam de visualizações diferentes dos mesmos dados, o que leva à criação de diferentes serviços quando o desempenho entra em jogo” (JOSUTTIS, 2008, p.143). Sendo assim, às vezes, é necessário criar serviços específicos para consumidores específicos, ou seja, customizar os serviços.

1.6 SEGURANÇA

Outro fator importante a ser considerado em uma implantação SOA é a segurança. Josuttis (2008) define a seguinte categorização para os aspectos de segurança para SOA:

- **Autenticação** – verificar identidades de usuários, dispositivos ou serviços externos;
- **Autorização** – controlar restrições e permissões para uma determinada identidade;
- **Confidencialidade** – assegurar que ninguém, além do fornecedor e consumidor, tenha acesso aos dados durante sua transferência;
- **Integridade** – garantir que os dados não sejam manipulados ou falsificados, ou que contenham erros;
- **Disponibilidade** – manter o sistema sempre disponível;
- **Contabilização** – monitorar chamadas de serviço para gerenciamento, planejamento ou outros propósitos;
- **Auditoria** – monitorar e rastrear todo o fluxo de dados relevantes para a segurança, seja para melhorar a sua confiabilidade, seja para atender a determinações legais.

A fim de atender os requisitos de autenticação e autorização "você normalmente precisa do conceito de ID de usuário e senha. Para IDs de usuário normalmente existe alguma forma indireta para que seja possível atribuir papéis a esses usuários frequentemente chamados de fornecedor de identidade" (JOSUTTIS, 2008, p.150).

Para confidencialidade e integridade, os conceitos comuns como criptografia e assinaturas digitais são usados.

1.7 WEB SERVICES

Sampaio (2007, p.139), define um *Web Service* da seguinte forma:

> "um *Web Service* é um aplicativo servidor que disponibiliza um ou mais serviços para seus clientes de maneira fracamente acoplada. [...] expõe sua interface para os usuários usando um documento XML conhecido como *Web Services Description Language* ou WSDL".

Embora, num primeiro momento, quando se ouve falar em SOA se faça uma relação direta com *Web Service*, deve-se entender que ele é apenas uma forma de implementação dessa arquitetura. Josuttis (2008, p.30), explica isso quando afirma que

> "o uso de *Web Services* não significa automaticamente que está implementando SOA. Isso pode ser apenas uma maneira para deixar alguns sistemas interagirem, sem todos os outros aspectos fornecidos e exigidos por SOA (JOSUTTIS, 2008, p.196)".

Outros conceitos ou características, citados por Josuttis (2008) são governança e maturidade. Governança pode ser definida como uma maneira de "garantir que as pessoas façam o que é certo", ou ainda de "controlar o desenvolvimento e a operação de software". O autor justifica a aplicação da governança em SOA com a finalidade de evitar o risco de caos e de desperdício de dinheiro e recursos. Assim como acontece com o modelo de referência *Capability Maturity Model Integration* (CMMI), SOA também lida com níveis de maturidade. Em Josuttis (2008), são descritos os cinco de níveis de maturidade SOA. Para maiores informações sobre esses conceitos, sugere-se consultar a obra do referido autor.

Finalizando o capítulo, é importante lembrar, principalmente, dos conceitos de acoplamento fraco, heterogeneidade e alta interoperabilidade. Sendo assim, considerando que um barramento é parte integrante e fundamental da arquitetura SOA e detentor desses conceitos, é fundamental dedicar um capítulo exclusivo ao ESB.

2 ENTERPRISE SERVICE BUS (ESB)

A finalidade deste capítulo é minudenciar as funcionalidades de um ESB, conceituar e expor suas características e desafios enfrentados. Assim como no capítulo anterior, este também está alicerçado, principalmente, na obra de Josuttis (2008).

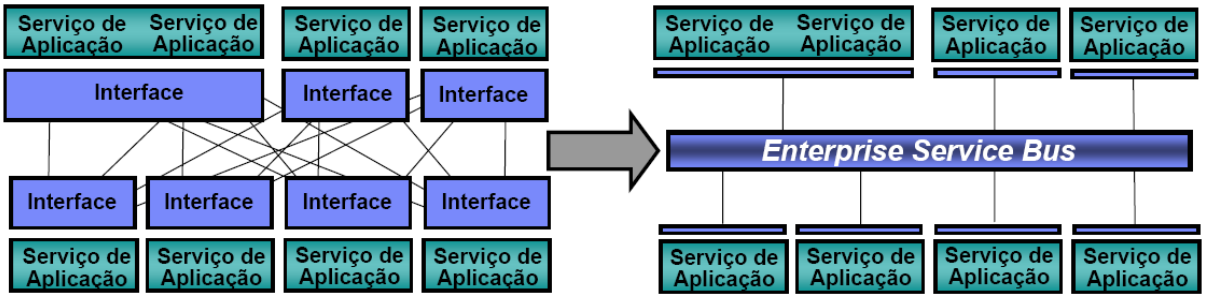

Figura 2.1 – *Enterprise Service Bus* - ESB
Fonte: VIOLA, 2006 apud BENETE JUNIOR, 2007, p. 33

A figura 2.1 mostra como um ESB é posicionado entre as aplicações, de forma a centralizar a comunicação entre as mesmas. Sem o ESB, cada aplicativo deve reconhecer a existência dos outros com quem se comunica. Isso gera um forte acoplamento entre eles. Com a adição do ESB entre esses aplicativos, esse acoplamento acaba por tornar-se fraco. Assim, não existe mais uma dependência entre eles, sendo atribuição do barramento a distribuição dos dados entre os mesmos.

2.1 CONCEITO

Uma parte de SOA é a infra-estrutura que lhe permite usar os serviços em um ambiente de sistemas produtivos. Isso é comumente chamado de Barramento de Serviços Corporativos (ESB – *Enterprise Service Bus*) (JOSUTTIS, 2008).

É interessante visualizar um conceito atribuído ao ESB à luz de outra área de conhecimento – Recursos Humanos. Silva (2009, p.58) escreve que “o barramento corporativo é uma camada que admite regras especiais, lógica específica e uma grande quantidade de softwares (programas de computador) integrados em padrões pertinentes”.

De acordo com Josuttis,

> “A infra-estrutura é a parte técnica de SOA que possibilita a alta interoperalidade. A infra-estrutura de um cenário SOA é chamada de barramento corporativo de serviços (ESB). Este termo foi tomado de integração das aplicações corporativas onde era chamado de barramento EAI ou simplesmente de barramento corporativo” (JOSUTTIS, 2008, p.17).

Dessa forma, torna-se evidente a clara relação entre SOA e ESB, uma vez que este é o responsável por possibilitar a chamada de serviços entre os sistemas heterogêneos, contribuindo de forma fundamental para a implementação dessa arquitetura.

Comparando as figuras 2.2 e 2.3, pode-se perceber como a implementação de um ESB pode melhorar a comunicação entre diferentes softwares aplicativos.

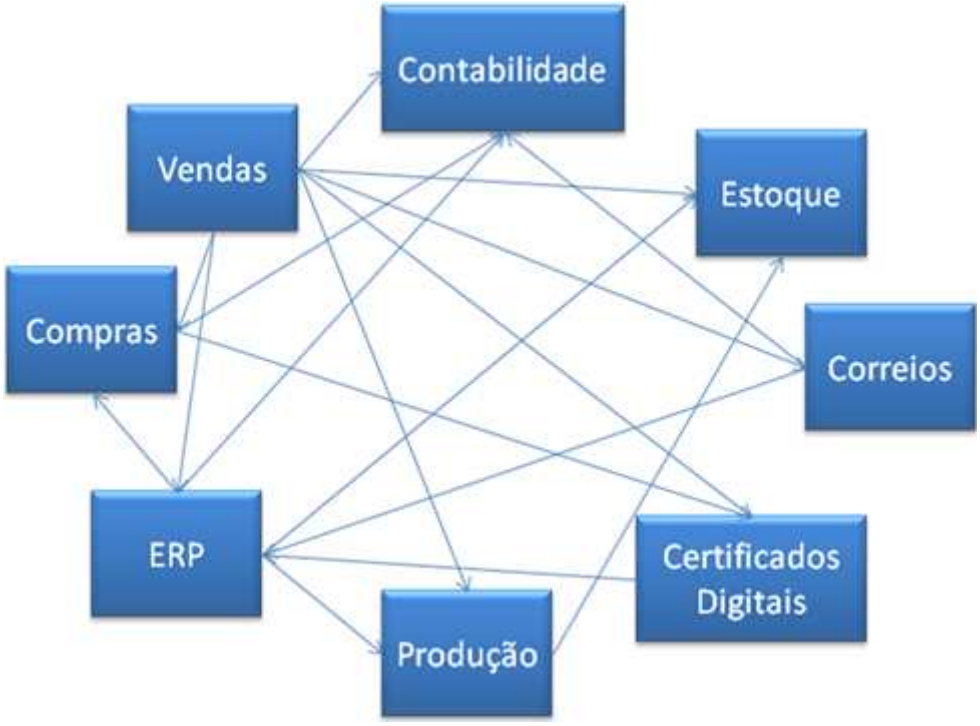

Figura 2.2 – Caos de comunicação
Fonte: MOREIRA, 2008

A figura 2.2 demonstra que sem a adoção de um ESB, cada um dos softwares aplicativos realiza a troca de dados diretamente conectado ao outro, através de uma forma específica. E, na maioria dos casos, quando um desses softwares deseja se comunicar um terceiro, uma nova forma é adotada.

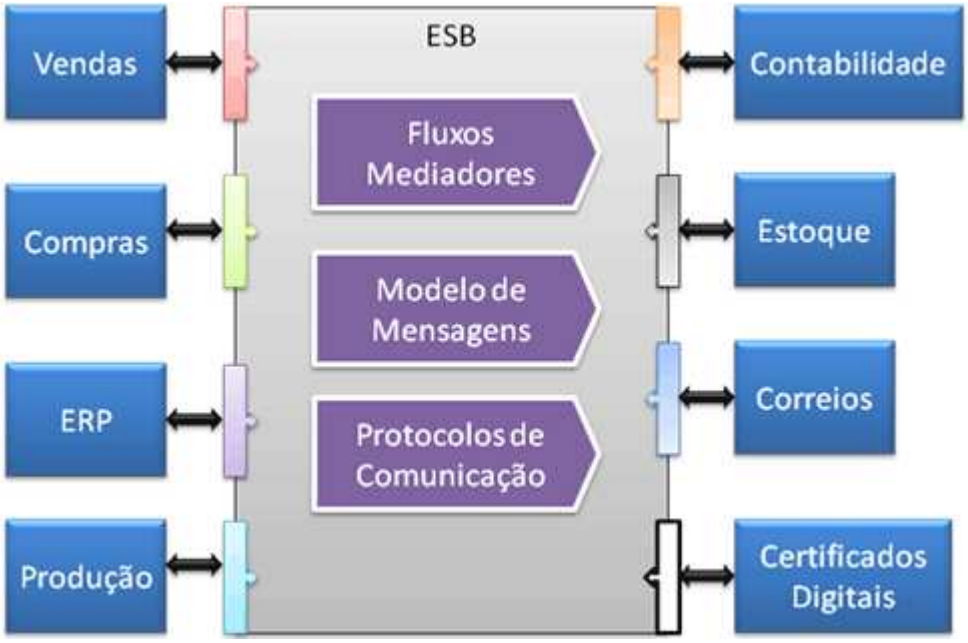

Figura 2.3 – Simplificação na comunicação através de um ESB
Fonte: http://tec.brq.com/conheca-um-pouco-mais-sobre-esb-enterprise-service-bus-e-as-ferramentas-ibm/

Já a figura 2.3 demonstra como a comunicação pode ser simplificada quando centralizada pelo barramento, podendo ser realizada de forma transparente no que diz respeito a protocolo. Isso pode ser mais bem entendido na sequência do texto.

2.1.1 Tipos de ESB

Para Josuttis

> "Conceitualmente, duas abordagens diferentes são possíveis considerando-se onde a responsabilidade de um ESB começa do ponto de vista dos consumidores e dos fornecedores. [...] abordagem orientada a protocolo[...] abordagem orientada a API [...]" (JOSUTTIS, 2008, p.48).

Na abordagem orientada a protocolo, consumidores e fornecedores enviam e recebem as mensagens de acordo com o protocolo estabelecido pelo ESB. Um exemplo disso são os *Web Services* que requerem um protocolo SOAP[8] (JOSUTTIS, 2008). Cada participante pode escolher as ferramentas que vai utilizar para atender o protocolo que foi definido.

[8] O *Simple Object Access Protocol* (SOAP) é uma tecnologia que permite distribuir objetos pela Internet (DEITEL, 2003).

Já na abordagem orientada a API[9], o ESB prevê que podem ser utilizadas as APIs específicas de cada plataforma, para que os consumidores e fornecedores implementem as chamadas de serviço (JOSUTTIS, 2008). Diferentemente da abordagem anterior, a equipe que mantém o ESB é responsável pela disponibilidade de formas de envio e recebimento das mensagens, baseadas nas APIs da plataforma utilizada.

Este trabalho segue o modelo de ESB sugerido por Josuttis (2008), que afirma que "para sinergia melhor, considere a implementação de um ESB orientado a API".

2.2 FUNCIONALIDADES

A principal responsabilidade ou funcionalidade de um ESB é possibilitar aos consumidores chamarem os serviços oferecidos pelos fornecedores (JOSUTTIS, 2008). Além disso, é possível listar as funcionalidades a seguir:

- **Conectividade** - um ESB deve ser capaz, de forma síncrona ou assíncrona, prover a conectividade entre os softwares aplicativos (fornecedor e consumidor) envolvidos;
- **Repositório** - manter arquivos ou dados em um diretório comum (unidade de disco local ou de rede) ao consumidor e fornecedor, é outra tarefa que deve ser oferecida por um ESB;
- **Transformação de dados** - talvez a mais importante das funcionalidades de um ESB seja a transformação de dados. Segundo Chappell,

> "A transformação de dados é inerentemente parte do barramento em uma distribuição ESB. Os serviços de transformação que são especializados para as necessidades das aplicações individuais ligadas no barramento podem ser localizados em qualquer lugar e acessíveis de qualquer lugar do barramento. Pelo fato de a transformação de dados ser uma parte integrante de ESB, um ESB pode ser imaginado como sendo uma resolução de desencontros de impedância entre as aplicações" (CHAPPELL 2004 apud JOSUTTIS, 2008, p.43);

- **Roteamento inteligente** - outra tarefa fundamental de ESB é o roteamento. Deve existir uma forma para que o consumidor envie uma mensagem ao fornecedor e receba uma resposta (JOSUTTIS, 2008). De acordo com o conteúdo das mensagens, o ESB deve ter a capacidade de proporcionar que sejam atribuídas

[9] "*Application Programing Interface* - Interface de Programas Aplicativos: conjunto de rotinas que os programas aplicativos utilizam para requisitar e obter os serviços de nível mais baixo oferecidos pelo sistema operacional do computador" (SAWAYA, 2002, p.26).

prioridades diferentes a cada uma delas, ou ainda, que se tomem diferentes atitudes (JOSUTTIS, 2008).

2.3 DESAFIOS

Durante a implementação de um barramento de serviços, dentro de uma abordagem SOA, alguns desafios precisam ser superados.

Deve haver um entendimento de que SOA é muito mais do que apenas uma infra-estrutura - uma cultura organizacional de colaboração deve ser criada; suporte gerencial é fundamental para o sucesso da implementação; deve-se introduzir SOA de maneira gradativa, garantindo um aprendizado.

Além desses, ainda destacam-se outros três desafios: complexidade, segurança e confidencialidade. Qualquer forma de acoplamento fraco aumenta a complexidade. Além do que, o esforço requerido para depuração e testes de sistemas distribuídos também é bem maior (JOSUTTIS, 2008). Em SOA, substitui-se as soluções individuais de conexão de cada par de sistemas por um ESB comum, afim de que cada sistema conectado ao ESB esteja conectado a todos os outros sistemas, por padrão esses sistemas estão desprotegidos (JOSUTTIS, 2008). A fim de garantir a confidencialidade dos dados transferidos, pode-se aplicar recursos de criptografia, tanto na camada de transporte, quanto na camada de mensagens.

A fim de justificar a escolha pela integração através de um ESB ao invés dos métodos convencionais, o próximo capítulo explora outras formas de se realizar a integração de softwares.

3 FORMAS ALTERNATIVAS DE INTEGRAÇÃO ENTRE SOFTWARES APLICATIVOS

Como existem outras formas de integração de softwares aplicativos, neste capítulo são arroladas algumas delas, a fim de se poder relacionar com o modo utilizado atualmente pela empresa escolhida para este trabalho. E ainda são elencadas vantagens e desvantagens desses modos de integração, comparadas às utilizadas pelo ESB e por SOA.

3.1 INTEGRAÇÃO PONTO A PONTO

No início, os aplicativos eram independentes, e não tinham nenhuma forma de comunicação com outros aplicativos. Logo as empresas sentiram a dificuldade de atualizar ou substituir esses aplicativos, e a capacidade de compartilhar passou a ser desejável (SPACKMAN e SPEAKER, 2006).

A principal forma de integração de dados, realizada ponto a ponto, dá-se através da troca de informações por arquivos texto.

> “A simplicidade dos arquivos texto fez deles uma escolha popular em diversas situações. De fato, um arquivo texto freqüentemente é a estrutura subjacente usada na implementação de arquivos seqüenciais mais elaborados, como um arquivo de funcionários” (BROOKSHEAR, 2005, p.321).

Um exemplo de integração de dados através da troca de arquivos texto pode ser visto na figura 3.1, onde arquivos são utilizados para integrar diferentes aplicativos, desde a indústria, passando pela empresa distribuidora, até o varejo.

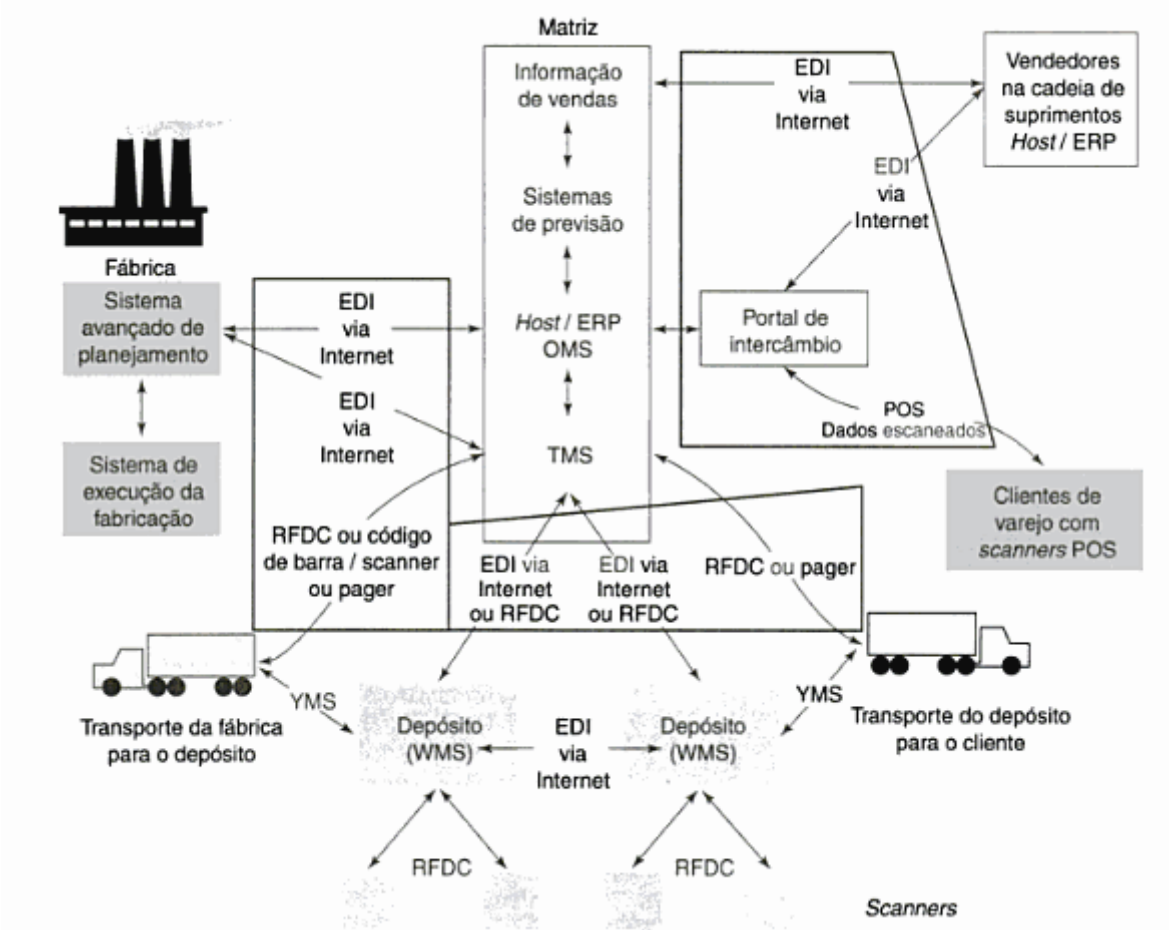

Figura 3.1 – Troca de dados por arquivo texto
Fonte: BOWERSOX et al, 2006, p.176

Como alternativa ao arquivo texto, tem-se utilizado arquivos XML. Eles são mais facilmente entendidos, uma vez que cada valor é associado a uma descrição. Cummins (2002, p. 10) conceitua um arquivo XML da seguinte forma:

> "Fundamentalmente, a XML possui um formato de dados com *tags*. Um documento XML contém vários elementos, cada qual com uma *tag* descritiva e um valor associado, todos expressos como caracteres. Caracteres especiais delimitam os elementos, de modo que os documentos e seus nomes de elemento e valores têm tamanho variável".

3.2 SOLUÇÕES DE PONTO CENTRAL

Seguindo a evolução da integração, o próximo passo foi a adoção da técnica de ponto central – "uma versão modificada da integração ponto a ponto, na qual um ponto central era responsável por direcionar os dados que estavam sendo passados entre os aplicativos" (SPACKMAN e SPEAKER, 2006, p.29).

3.2.1 Troca de mensagens

Uma forma utilizada para realizar a integração entre sistemas, através de um ponto central, é oferecida pela Integração de Aplicações Corporativas – *Enterprise Application Integration* (EAI) e apresentada na obra de Cummins (2002).

A proposta da EAI, apesar de ser muito parecida com a de um ESB, não é herdada de do mesmo conceito. Enquanto ESB é parte inerente de SOA, EAI não está contextualizada dentro de uma arquitetura específica, ou seja, é um modo isolado de se realizar uma integração entre sistemas.

Na abordagem da EAI, basicamente, um emissor envia uma mensagem para uma fila para transmissão, onde o gerente da fila a encaminha ao destino. No destino, essa mensagem é retida até que o destinatário esteja pronto para recebê-la (CUMMINS, 2002).

Assim como em um ESB, essa conexão é realizada por meio de um acoplamento fraco, uma vez que o destinatário, necessariamente, não precisa estar apto a receber a mensagem no momento em que ela é gerada pelo emissor. A figura 3.2 ilustra a ação de um intermediário de mensagens, utilizado pela EAI.

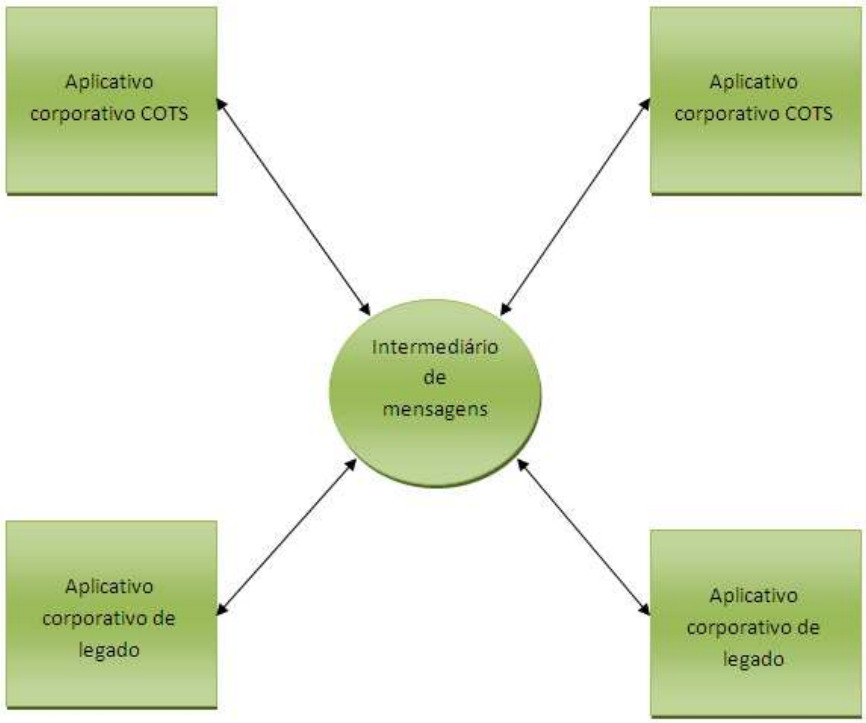

Figura 3.2 – Integração de aplicativos corporativos
Fonte: CUMMINS, 2002, p.7

3.2.2 Armazenamento compartilhado

A necessidade de integração levou, primeiramente, ao compartilhamento dos dados dos aplicativos através de um banco de dados em comum. De acordo com Spackman e Speaker (2006, p.28),

> "Sempre que uma empresa precisava compartilhar dados entre aplicativos, ela simplesmente adicionava um novo trecho de código ou um modo de visualização de bancos de dados, até que o número de aplicativos e pontos de integração finalmente tornou gerenciamento da integração praticamente impossível. Os aplicativos se tornaram tão fortemente integrados que uma pequena alteração em um trecho de código levava a uma cadeia complexa de alterações adicionais, apenas para preservar a integração, tornando o processo como um todo extremamente frágil e dispendioso".

Em Brookshear (2005, p. 339) verifica-se que, "historicamente, os bancos de dados evoluíram como forma de integrar sistemas de armazenamento de dados [...] os sistemas de banco de dados emergiram como meios de consolidar a informação armazenada e mantida por uma organização", conforme ilustrado na figura 3.3.

Figura 3.3 – Organização de um banco de dados
Fonte: BROOKSHEAR, 2005, p.339

3.3 OUTRAS FORMAS DE INTEGRAÇÃO

Oportunamente, compete ainda neste estudo citar outras formas, mesmo que ainda não tão difundidas, de proceder-se uma integração de sistemas. Três diferentes formas de integração são relacionadas em (CUMMINS, 2002):

- **Gerenciamento de fluxo de trabalho** – um processo de fluxo de trabalho gerencia a execução de atividades, que podem realizar diretamente operações limitadas ou invocar outros aplicativos. Pode variar do processamento *batch* ao processamento baseado em eventos, onde as transações comerciais são processadas à medida que ocorrem;
- **Objetos distribuídos** – essa tecnologia oferece suporte ao desenvolvimento de sistemas através de componentes distribuídos, os quais interagem como objetos, trocando mensagens em uma rede. Esses objetos podem ser serviços ou objetos compartilhados de uma aplicação de negócio;
- **Tecnologia de agentes** – programa autônomo que capta o ambiente e diante disso reage de acordo com regras internas de operação. Diferentemente de uma aplicação convencional, o próprio agente determina quando e de que maneira executará uma função. Podem ser usados para direcionar fluxos de trabalho.

O quadro 3.1, baseado nas formas de integração relacionadas neste estudo, identifica vantagens e desvantagens na aplicação de cada uma delas.

Forma	Vantagens	Desvantagens
Arquivos Texto	Simplicidade	Falta de segurança da informação
Arquivos XML	Identificação dos elementos	Tamanho do conteúdo, em virtude da duplicidade de texto
Troca de Mensagens	Acoplamento fraco	Necessidade de um centralizador de mensagens
Banco de Dados	Alta disponibilidade e performance	Entidades e atributos devem ser conhecidos pelas aplicações
Gerenciamento de Fluxo de Dados	Mapeamento dos processos	Falta de interoperabilidade
Objetos Distribuídos	Escalonabilidade	Homogeneização de plataforma
Tecnologia de Agentes	Autonomia	Difícil implementação

Quadro 3.1 – Comparativo entre as formas de integração
Fonte: do autor

Com a finalidade de se verificar o estudo e a aplicação dessas formas de integração, bem como daquela que está sendo proposta, faz-se necessária uma pesquisa sobre trabalhos correlatos, objeto do próximo capítulo.

4 TRABALHOS CORRELATOS

No intuito de auxiliar no estabelecimento de parâmetros para a modelagem do ESB, são estudados trabalhos já realizados sobre os assuntos expostos. Dessa forma, são analisados os resultados obtidos, além de desafios e dificuldades encontrados durante as suas realizações.

Primeiramente, foi realizada uma pesquisa sobre trabalhos acadêmicos, tanto em nível de graduação quanto de pós-graduação (mestrado e doutorado), que tivessem uma proposta similar a deste trabalho. A pesquisa não obteve êxito. Como alternativa, optou-se então, por realizar uma pesquisa sobre trabalhos que tivessem uma fundamentação parecida. Foram localizados trabalhos em diferentes níveis acadêmicos, fundamentados na arquitetura SOA e que, diferentemente deste trabalho, não implementaram, mas utilizaram um ESB, ou em determinado momento deram ênfase a essa tecnologia.

Comercialmente, foram localizadas diferentes implementações de ESB, tanto em softwares proprietários, como em softwares de código aberto. A seguir, encontra-se uma breve exposição de alguns dos resultados encontrados.

4.1 NÍVEL ACADÊMICO

Como resultados da pesquisa, a nível acadêmico, nesta seção são expostos trabalhos em diferentes graus (graduação e pós-graduação).

4.1.1 Graduação

Garcia et al (2008) apresentou à Universidade Federal de Itajubá – UNIFEI, de Itajubá – MG um trabalho que teve como justificativa a rápida evolução das tecnologias utilizadas para o desenvolvimento de software e a falta de padronização nesse desenvolvimento, especialmente nos sistemas utilizados na área médica. Como proposta, foi apresentada uma solução para a integração de diferentes softwares médicos, que trabalham com laudos descritos em forma de texto e com imagens obtidas em exames médicos como tomografias e ressonâncias magnéticas.

Na implementação, foram utilizados *Web Services* como barramento, para atender aos serviços de consulta de pacientes e exames. Como resultado, foi publicado que, com a organização das funcionalidades dos sistemas médicos como serviços, é alcançada a robustez que dá suporte à escalabilidade dos mesmos. E ainda, essa organização permite que se reúnam dados do paciente que estão distribuídos em sistemas distintos para obtenção de um prontuário completo.

Sobre SOA, o estudo concluiu que "é uma tecnologia adequada para a integração de ambientes heterogêneos como os ambientes da área médica", e que ela "introduz diretrizes para guiar o desenvolvimento de futuras integrações entre outros sistemas médicos".

Ainda na graduação, foi encontrado um trabalho que está em fase de conclusão na Universidade Federal de Santa Catarina (MENEZES e GALOPPINI, 2009). Ele tem como objetivo "construir um suporte para integração de sistemas utilizando *Enterprise Service Bus* (ESB), que permita a interação de sistemas legados com provedores de serviços construídos com base em uma arquitetura orientada a serviços (SOA), provendo mecanismos para transformação de dados, conectividade e tratamento de erros". Apesar de tratar sobre a integração de sistemas através de um ESB, ele não propõe a modelagem e/ou desenvolvimento de um barramento, mas sim de um *framework* para aplicação de um protocolo específico, através do uso de um barramento já existente.

4.1.2 Especialização

Benedete Junior (2007) apresentou uma monografia à Escola Politécnica da Universidade de São Paulo, a fim de obter o título de MBA[10] em Tecnologia da Informação. O título do trabalho é *Roteiro para definição de uma arquitetura SOA utilizando BPM*[11].

A motivação do trabalho está baseada na necessidade das melhorias de processos de negócio e da comunicação com a área de tecnologia da informação. Ele propõe a utilização de SOA e BPM a fim de suprir tais necessidades.

[10] Mestrado em Administração de Empresas (Especialização ou Pós-graduação Lato Sensu).

[11] Gestão de Processos de Negócio (BPM), "visto como uma disciplina de gestão, é a habilidade de continuamente otimizar aqueles processos operacionais que são mais diretamente relacionados à obtenção dos objetivos da corporação" (BENEDETE JUNIOR, 2007, p.13).

Apesar de apenas fazer referência ao ESB, o trabalho explora a aplicação de SOA de uma forma geral, em conjunto com o mapeamento dos processos de negócio, tendo sido de grande valia para este estudo, servindo inclusive como referência bibliográfica.

O autor, no capítulo 2, demonstra em uma tabela os desafios apresentados e como SOA pode superá-los. Além do que, em todo o trabalho ele procurou reforçar o potencial de SOA no alinhamento entre as áreas de TI e de negócios. Benedete Junior (2007, p.53), concluiu ainda que “a transformação da empresa, de uma estrutura departamental para uma com foco em processos corporativos, pode ser essa oportunidade de mobilização, de construção de um futuro melhor para a empresa e seus colaboradores.”.

4.1.3 Mestrado

SOA também foi apresentada e defendida em uma dissertação para obtenção de grau de Mestre em Engenharia Informática e de Computadores da Universidade Técnica de Lisboa (Portugal). Pardal (2006), motivado pela resposta à necessidade que as empresas têm em agilizar seus sistemas de informação, enfrentando o desafio de adaptação aos requisitos de negócio.

O principal objetivo da dissertação foi a avaliação de normas de segurança para *Web Services*, através de um estudo de caso e ensaios em protótipos. Em sua conclusão, o autor identificou mecanismos necessários para implementar a segurança de uma plataforma de serviços.

4.2 NÍVEL COMERCIAL

Comercialmente, existem produtos disponíveis que se propõem a realizar as funcionalidades de um ESB, tanto em forma de software proprietário, quanto na forma de software livre.

4.2.1 Código proprietário

A Microsoft comercializa o software Microsoft BizTalk Server, atualmente na quinta versão (2006). De acordo com o *DataSheet*[12] do produto, "o BizTalk Server 2006 R2 tem como alicerce o BPM e as capacidades de SOA e ESB de versões anteriores para ajudar as organizações a ampliar ainda mais as tecnologias de gerenciamento de processos". Uma simplificação do funcionamento do BizTalk é apresentada por Cambiucci (2008), através da figura 4.1.

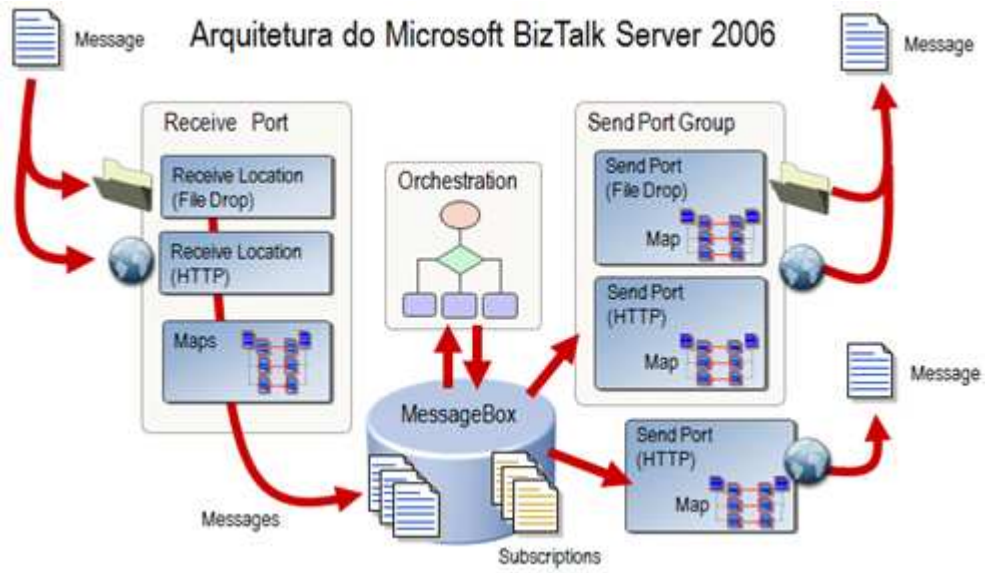

Figura 4.1 – Arquitetura do Microsoft BizTalk Server 2006
Fonte: CAMBIUCCI, 2008

O BizTalk Server 2006 R2 tem diversas edições de licenciamento pagas e uma edição sem custo, denominada de *Developer edition*, "restrita a soluções de criação, desenvolvimento e testes" (BizTalk 2006 DataSheet, p.2).

Estão disponíveis no BizTalk Server 2006 R2, além dos adaptadores comuns (FTP, HTTP, SMTP, POP3, SQL Server, ...), opções para conexão com bancos de dados IBM DB2 e Oracle, além de softwares da SAP e PeopleSoft.

Conforme tabela[13], o preço do BizTalk Server 2006 R2, pode variar de 499 dólares - versão *developer* para usuários não cadastrados -, até 34.990 dólares. Para maiores informações, consultar o site da Microsoft[14].

---

[12] Quadro de características. Especificações dos parâmetros de um dispositivo (hardware ou software), definidas pelo fabricante (SAWAYA, 2002).
[13] Tabela de preços publicada pela Microsoft em 28 de fevereiro de 2008 (http://www.microsoft.com/brasil/servidores/biztalk/howtobuy/default.mspx, acessado em 11/06/2009).
[14] http://www.microsoft.com/brasil/servidores/biztalk/default.mspx.

Outro produto disponível é o IBM WebSphere Enterprise Service Bus (WSESB). Segundo Moreira (2008), esse produto é parte integrante de outro produto da IBM, o WebSphere Application Server (WAS). Como vantagem, Moreira (2008) aponta a facilidade na implantação e o preço reduzido. Para informações adicionais, consultar o site da IBM[15].

4.2.2 Código aberto

O WSO2 ESB é um barramento de serviços corporativos, capaz de atender os mesmos requisitos dos barramentos comerciais de código proprietário. Ele é disponibilizado gratuitamente no site da empresa WSO2[16]. A WSO2, com sede no Sri Lanka, foi fundada por Sanjiva Weerawarana, ex-funcionário da IBM, em agosto de 2005. Uma ilustração do seu funcionamento pode ser encontrado no *DataSheet* do produto, reproduzida na figura 4.2.

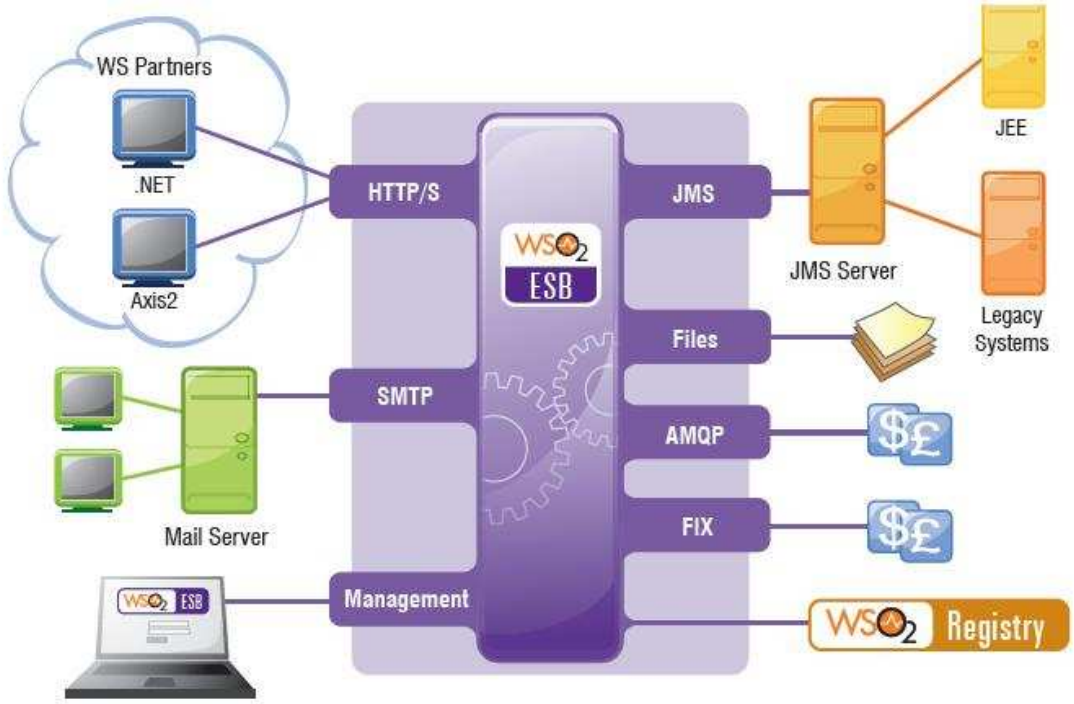

Figura 4.2 – ESB da WSO2
Fonte: WSO2 ESB DataSheet

De acordo com o *DataSheet* do produto, ele é capaz de conectar-se a qualquer banco de dados suportado por Java, além de ser multiprotocolo (FTP, HTTP, POP3, SMTP), com suporte a qualquer tipo de XML.

[15] http://www-01.ibm.com/software/br/websphere.
[16] http://wso2.com/products/connect/wso2-enterprise-service-bus.

Um dos mais conhecidos ESB *open-source* é o Mule. Desenvolvido por uma comunidade, é disponibilizado no site http://www.mulesource.org/display/MULE/Home. Além disso, no site é possível encontrar farta documentação. A figura 4.3 ilustra, resumidamente, o funcionamento do Mule e alguns conectores.

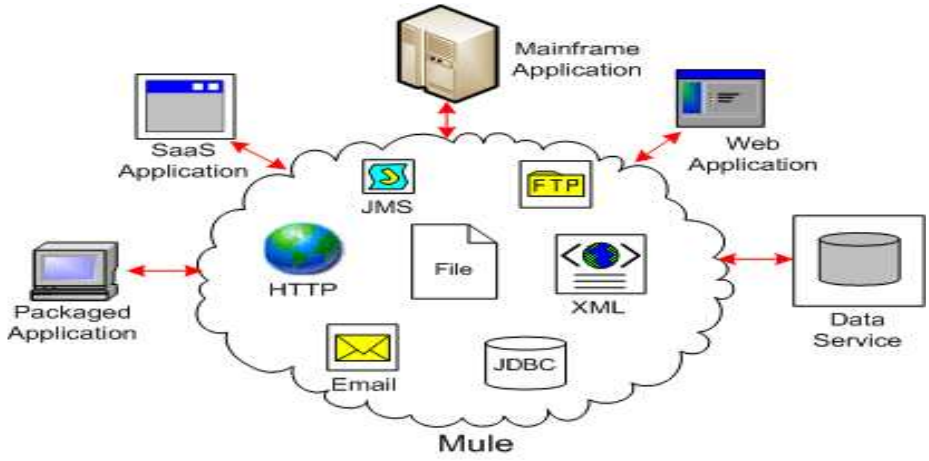

Figura 4.3 – Mule ESB
Fonte: http://www.mulesource.org/display/MULE2INTRO/What+is+Mule

Santos (2007, p.18) apresenta o Mule ESB, destacando suas características:

> "Mule é um *framework Enterprise Service Bus* (ESB) para mensagens. É escalável e funciona como um *object broker* distribuído que gerencia interações com *services* e aplicações utilizando tecnologias de transporte e mensagem. Mule foi desenhado de forma a ser leve e facilmente embutível em aplicações Java e servidores de aplicação ou rodar como um servidor *stand alone*".

O quadro 4.1 resume e destaca as principais características e diferenciais de cada ESB mencionado.

ESB	Tipo	Características
MS BizTalk	Código Proprietário	Conectores para DB2 e Oracle; Valores de licença escalonáveis; Versão grátis para desenvolvedores.
IBM WebSphere	Código Proprietário	Mais barato se já possuir IBM WAS; Conectores para SQL Server e Oracle.
WSOS ESB	Código Aberto	Gratuidade; Conexão com qualquer banco de dados suportado por Java.
Mule	Código Aberto	Gratuidade; Farta documentação.

Quadro 4.1 – Características e diferençais dos ESBs
Fonte: do autor

Ao final deste capítulo, vale salientar uma característica encontrada em cada ESB apresentado – a disponibilidade de conexão por diversos meios, chamados de conectores. É através dos conectores que as aplicações interagem com o ESB, estando ele alocado em um

mesmo computador, em uma rede local ou não. Característica esta, que foi observada durante o desenvolvimento do modelo proposto.

5 PROPOSTA

O foco deste capítulo é, inicialmente, apresentar aspectos estruturais e funcionais dos processos da empresa escolhida para a realização deste estudo. Dessa forma, por meio dessa especificação é possível indicar precisamente onde ocorre o principal problema a ser resolvido com este trabalho.

5.1 A EMPRESA ESCOLHIDA

A Sanvitron Controle e Automação Ltda foi fundada em 1997 na cidade de Igrejinha-RS, por dois formandos do curso técnico em eletrônica do Colégio CIMOL de Taquara. O primeiro projeto da empresa foi o desenvolvimento de um relógio ponto eletrônico para um cliente específico, tendo sido estendido para diversos outros clientes. A empresa desenvolveu e mantém um software para gerenciamento dos registros desses equipamentos. Além disso, o software faz a apuração de horas normais, adicional noturno, repouso remunerado, banco de horas e todos os eventos necessários para o cálculo da folha de pagamento. Esses foram os principais produtos da empresa por diversos anos.

Atualmente, além desses produtos, a Sanvitron desenvolve hardware e software para o controle de acesso através de catracas e cancelas, bem como controle e automação de diversos processos realizados por seus clientes. É preocupação constante da empresa oferecer sempre produtos de altíssima qualidade, alicerçados em tecnologia de ponta e inovação.

A opção por esta empresa se deu pelo fato de o autor deste trabalho ser colaborador da Sanvitron desde o ano de 2003, tendo trabalhado em diversos projetos durante este período na função de analista e desenvolvedor de software. Por esse motivo, o autor pode verificar pessoalmente a necessidade da melhoria nos processos de integração dos aplicativos da Sanvitron com as aplicações de folha de pagamento utilizadas pelos seus clientes.

5.2 SITUAÇÃO ATUAL

A empresa integra o resultado do processamento dos registros de ponto realizados pelos funcionários dos seus clientes, combinado com as regras de cálculo estipuladas por eles, com as mais diversas folhas de pagamento do mercado. Essa integração acontece pela configuração de um *layout* estabelecido por cada uma dessas folhas de pagamento. Quando existe uma característica no *layout* que não pode ser atendida pela ferramenta de configuração, há a necessidade de adequação do código fonte do sistema, para suprir essa necessidade.

Além da integração da apuração dos eventos, há também a integração na entrada de dados. Esses dados são constituídos de informações pertinentes aos funcionários, como código, nome, data de admissão, horários em que deve trabalhar, cargo e função atuais, dentre outros. Da mesma forma, existem diferentes formas de se fazer essa integração:

- **Arquivo texto** – é realizada a exportação dos dados na folha de pagamento, através da geração de um arquivo texto, depositado em um local previamente combinado entre as aplicações. O software da Sanvitron, denominado STSWin (*Sanvitron Time System for Windows*), ao localizar o arquivo, realiza a leitura do seu conteúdo, importando os dados que ali constam para a sua base de dados. Como cada software de folha de pagamento dispõe os dados de uma forma diferente, esse *layout* também pode ser configurado;
- **Bibliotecas** – através de bibliotecas de ligação dinâmica (DLL), pode ser realizada a integração com aplicativos desenvolvidos em linguagem de programação que as suportem;
- **Banco de dados** – menos usual, é a integração através de acesso direto à base de dados, seja do STSWin pela folha de pagamento, seja da folha de pagamento pelo STSWin.

Conforme observado, o processo de integração atual pode ser, superficialmente, exemplificado através da figura 5.1.

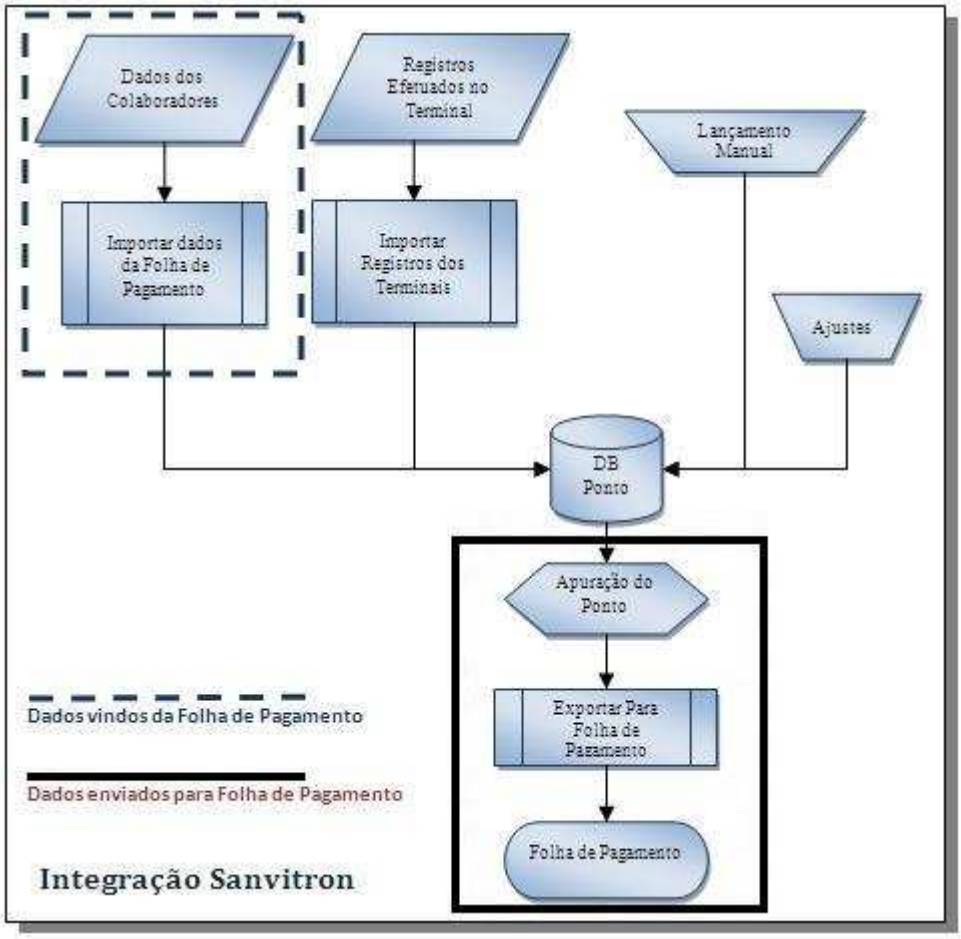

Figura 5.1 – Fluxo do processo de integração
Fonte: Do autor

Duas áreas são destacadas na figura 5.1, através de linhas pontilhadas. A primeira delas – linha tracejada -, sinaliza a entrada de dados no sistema de ponto eletrônico, através de informações enviadas pela folha de pagamento. A segunda – linha contínua -, localiza dentro do fluxo do processo de integração, o momento em que o resultado do processamento realizado é retornado à folha de pagamento. Esses dois momentos são detalhados nas figuras 5.2 e 5.3.

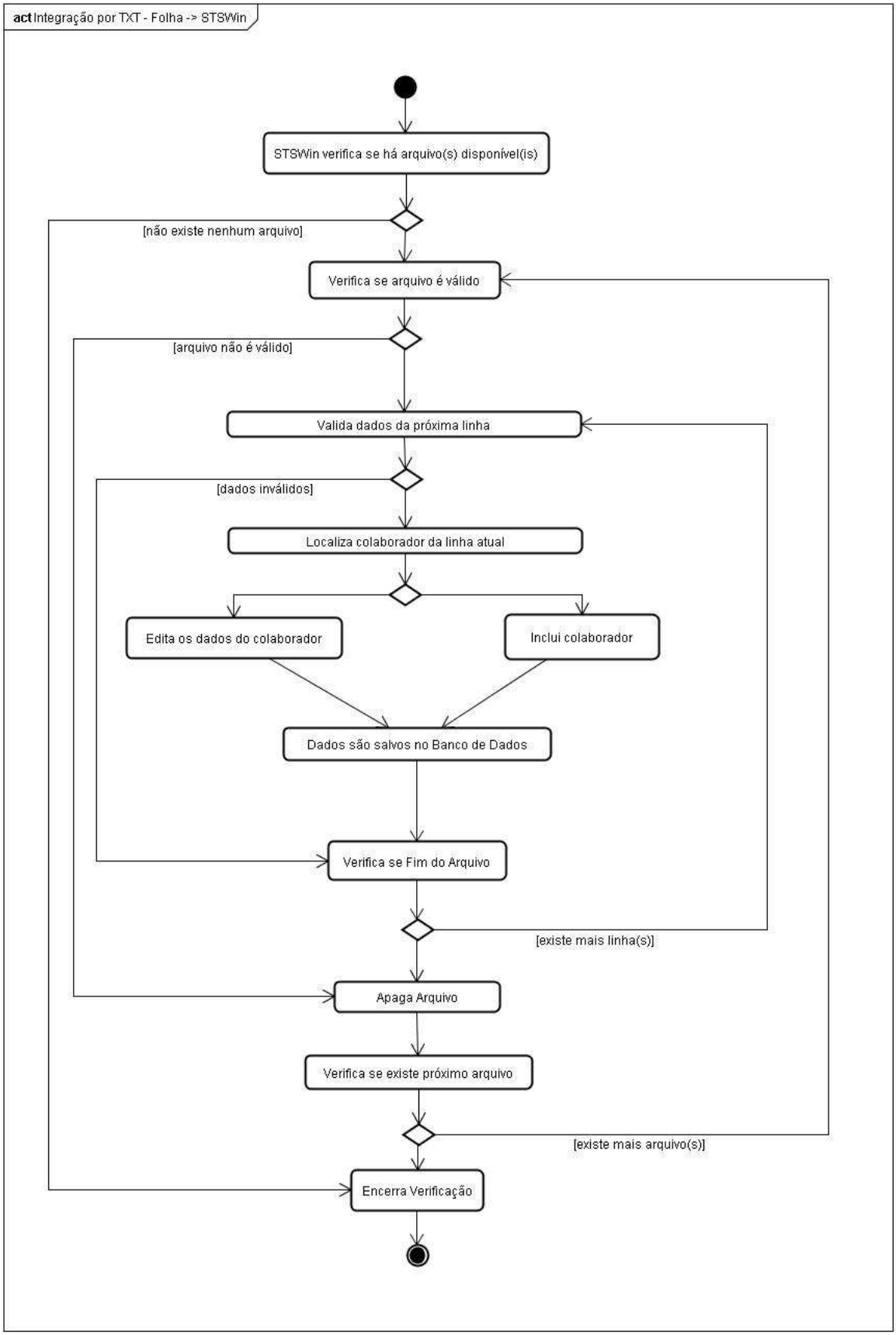

Figura 5.2 – Diagrama de atividade - importação de colaboradores pelo ponto eletrônico
Fonte: Do autor

Na figura 5.2 é demonstrado através de um diagrama de atividade, o ciclo do processo de importação de colaboradores. A folha de pagamento gera um arquivo com os dados dos colaboradores, o qual é depositado em um local combinado pelos dois aplicativos. A geração desse arquivo pode ser feita de forma manual, pelo usuário da folha de pagamento, ou automaticamente, sempre que um dado de um colaborador é modificado.

Periodicamente, o STSWin verifica a disponibilidade de um novo arquivo no local combinado. Em caso afirmativo, ele verifica se o conteúdo do mesmo está de acordo com o esperado. Ele lê esse conteúdo, linha a linha, analisando a consistência dos dados de cada uma delas, adicionando um novo colaborador, ou atualizando os seus dados, caso já esteja cadastrado. Ao final, o arquivo é apagado, para evitar que seja processado novamente.

Uma falha nesse processo, é que a folha de pagamento não tem um retorno sobre o processamento do arquivo gerado. Além disso, caso exista mais de uma instância do STSWin sendo executada, é possível que o arquivo seja processado mais de uma vez.

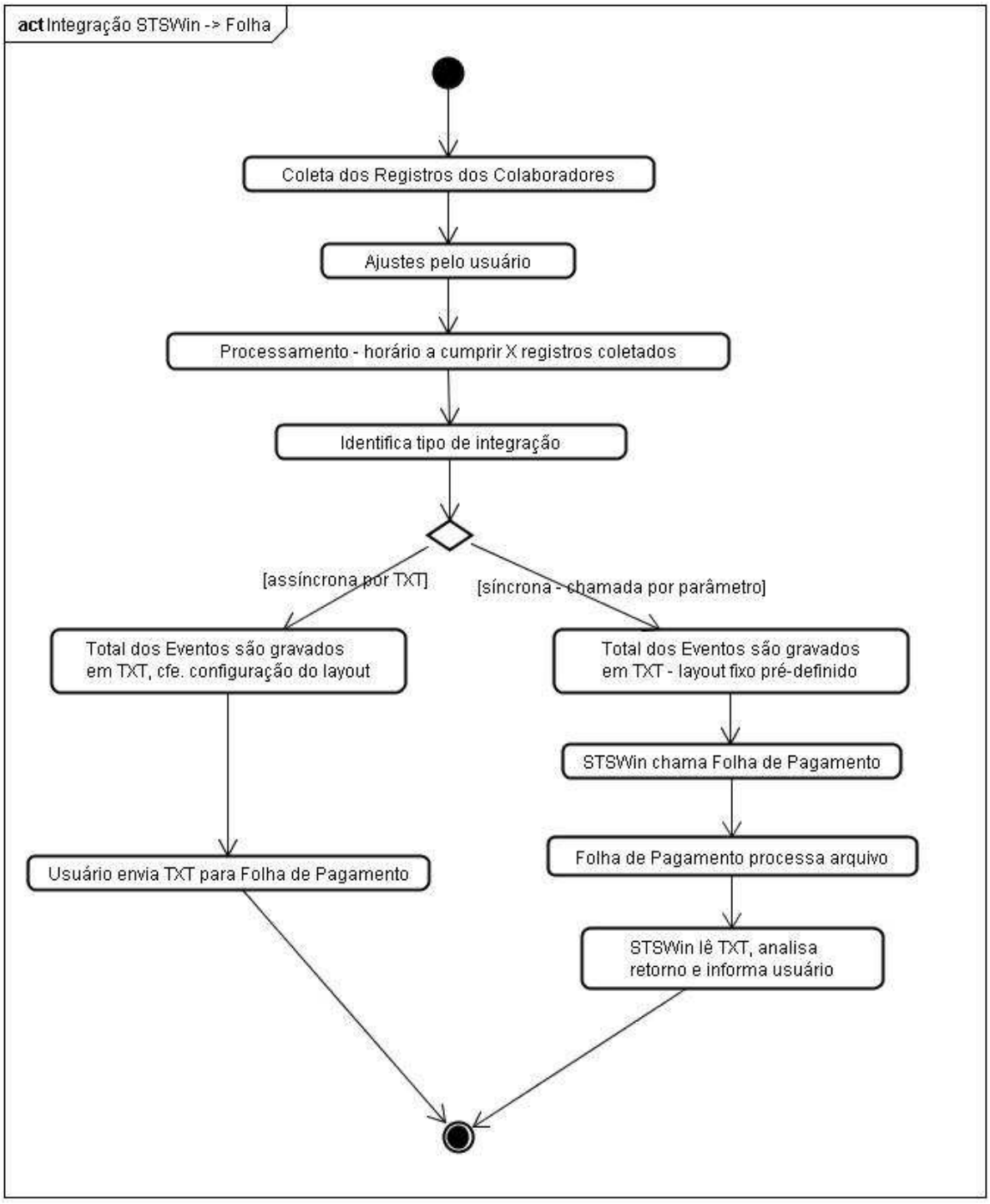

Figura 5.3 – Diagrama de atividade – exportação do resultado do cálculo para folha de pagamento
Fonte: Do autor

A figura 5.3 detalha o segundo processo, cujo sentido é inverso, onde dados do STSWin são enviados para folha de pagamento. O usuário do STSWin realiza um fechamento, o qual consiste em processar todos os registros que os colaboradores realizaram nos terminais de ponto, comparando-os com os horários que deveriam ter sido cumpridos.

Além dessas informações, existem lançamentos de gozo de férias, de feriados e de afastamentos do trabalho.

O resultado desse processamento é um conjunto de eventos totalizadores que devem ser enviados para folha de pagamento, para cálculo dos vencimentos e descontos dos colaboradores. Esses eventos são compostos por valores correspondentes a horas trabalhadas, faltas, horas extras, afastamentos do trabalho, adicional noturno, dentre outros.

Atualmente, existem duas formas de integração com folhas de pagamentos. Uma delas é assíncrona, onde um arquivo texto é gerado e enviado pelo usuário para o sistema de folha de pagamento, na própria empresa, ou no escritório de contabilidade. Esse envio é realizado da forma que o usuário julgar a mais conveniente, uma vez que a garantia de entrega passa a ser dele. A segunda é realizada de maneira síncrona. O usuário solicita ao STSWin o envio dessas totalizações para a folha de pagamento, que está instalada na sua empresa, e ao alcance do sistema de ponto eletrônico. O sistema gera um arquivo texto de acordo com o *layout* combinado, e chama o arquivo executável da folha de pagamento, informando o nome e local do arquivo gerado. A folha de pagamento faz a importação dessas totalizações, informando para cada uma delas se teve sucesso, regravando uma descrição ao final da linha. Após a conclusão, o STSWin relê esse arquivo informando o usuário sobre possíveis rejeições.

Esse processo também tem limitações, uma vez que o atendimento ao *layout* de integração é realizado diretamente no código fonte. Essa integração é realizada apenas com um sistema de folha de pagamento em específico. Caso seja necessária a implementação dessa forma de integração com outra folha de pagamento, os ajustes requeridos devem ser feitos diretamente no código fonte do sistema, e o mesmo recompilado.

Embora a manutenção do código fonte para adequação a necessidade de uma nova integração seja de fácil realização, sempre há o despendimento de recursos de tempo e pessoal, além do custo financeiro.

5.3 PROPOSTA DE MELHORIA

No sentido de minimizar a aplicação de recursos no atendimento de uma nova integração, surge a proposta deste trabalho. Através da arquitetura SOA, especificamente pelo

uso de um ESB, acredita-se que o processo de integração possa ser melhorado, de forma a garantir à empresa uma economia, tanto financeira, quanto de tempo.

Através deste trabalho, é proposta a modelagem de um ESB, o qual terá o objetivo de intermediar a integração dos dados entre o software de ponto eletrônico da Sanvitron (STSWin) e os aplicativos de folha de pagamento existentes no mercado.

Figura 5.4 – Ilustração do ESB proposto
Fonte: Do autor

Conforme ilustrado na figura 5.4, o ESB proposto prestará os serviços de conectividade, roteamento, transformação e repositório de dados. Ele deverá garantir a entrega dos dados gerados por diferentes folhas de pagamento ao sistema de ponto eletrônico. Assim como será responsável por enviar o resultado do processamento efetuado pelo fechamento do ponto aos aplicativos de folha de pagamento.

O capítulo seguinte traz em mais detalhes, a modelagem proposta para este barramento.

6 MODELAGEM DO ESB PROPOSTO

Este capítulo tem por objetivo realizar o detalhamento do modelo do ESB proposto utilizando a Linguagem de Modelagem Unificada – *Unified Modeling Language* (UML) e conceitos da engenharia de software.

6.1 A IMPORTÂNCIA DA MODELAGEM

Uma empresa que consiga desenvolver software de forma previsível e dentro de um período determinado, utilizando de forma eficaz e eficiente seus recursos, será uma empresa viável (BOOCH et al, 2002).

Inicialmente deve-se realizar um bom levantamento de requisitos. “Para entregar um software que satisfaça ao propósito pretendido, será preciso reunir-se e interagir com os usuários de uma maneira disciplinada, com a finalidade de expor os requisitos reais do sistema” (BOOCH et al, 2002, p.3).

A modelagem é parte central das atividades que levam à implantação de um bom software, utilizando-se modelos para comunicar a estrutura e o comportamento desejado do sistema (BOOCH et al, 2002).

6.2 REQUISITOS

Conforme exposto no capítulo anterior, o ESB é composto pelos principais requisitos de conectividade, roteamento, transformação e repositório de dados. Além disso existe ainda um requisito importante, que será responsável pela interface com o usuário: configurar o ESB.

Segundo Rezende (2006), “os requisitos de um software, também chamados de requerimentos de software ou de requisitos funcionais de um sistema, devem ser elaborados no início de um projeto de sistemas ou software”. Levando-se isso em consideração, a modelagem do ESB tem início pelos requisitos funcionais a seguir.

6.2.1 Requisito Serviço

O ESB deverá funcionar como serviço, sendo inicializado juntamente com o sistema operacional, sem necessidade de intervenção humana. Esse serviço deve ficar ativo, monitorando a ocorrência de eventos de chegada de dados para envio, ou verificando a disponibilidade de dados a serem recebidos.

6.2.2 Requisito Conectividade

Este requisito corresponde à garantia da conectividade entre o fornecedor e o consumidor, através dos diversos conectores disponibilizados. É o meio pelo qual a entrega dos dados será realizada.

Para atendimento deste requisito, são necessárias informações essenciais utilizadas pelos conectores, como usuário, senha e nome ou IP do servidor, por exemplo. Esses dados serão informados pelo usuário, no momento da administração de uma nova conexão do ESB, através de uma interface gráfica, pela qual é responsável o Requisito Configurar ESB.

6.2.3 Requisito Roteamento

Quando um dado é inserido no barramento, faz-se necessário identificar quem o disponibilizou (fornecedor) e a quem ele deve ser entregue (consumidor). Uma vez identificados, deve-se providenciar a remessa (Requisito Conectividade). Antes, porém, deve-se verificar se esse arquivo deverá ser interceptado e sofrer qualquer tipo de modificação, seja no seu conteúdo, seja na sua estrutura (Requisito Transformação).

6.2.4 Requisito Transformação

Podem ocorrer modificações dos dados a serem entregues. Essas modificações podem ser, tanto em seu conteúdo, quanto em sua estrutura. Essa transformação ocorre de acordo com o que foi definido pelo usuário no momento de configuração.

Como meio de transformação desses dados, é utilizada a ferramenta conhecida como *FastScript*[17]. Ela é composta por classes e componentes disponíveis para as linguagens de programação Delphi e C++. Esses componentes recebem o código-fonte implementado pelo usuário, através de texto armazenado no banco de dados da aplicação. O código-fonte é executado de forma interpretada, em tempo de execução, garantindo flexibilidade, uma vez que o código-fonte original do ESB não precisa ser alterado.

6.2.5 Requisito Repositório de Dados

O barramento se utiliza de um repositório para disposição dos dados, desde a sua recepção até a entrega. Eventualmente, de acordo com a configuração realizada, são mantidas cópias dos dados enviados, em locais específicos. Isso corresponde ao requisito Repositório de Dados.

6.2.6 Requisito Configurar ESB

A fim de se atender todos os outros requisitos, é necessário que seja realizada uma configuração por parte do usuário, para que o ESB atenda o seu desejo ou necessidade.

Como meio de configuração, deve existir uma interface gráfica, disponível ao usuário a qualquer momento. Por essa interface, o usuário pode parametrizar a ação que o ESB deve tomar em relação a determinados dados, indicando uma recepção, entrega ou

[17] http://fast-report.com/en/products/fast-script.html

transformação. Além disso, indicará por que meio (conector) essa entrega ou recepção acontecerá.

6.3 MODELAGEM DO ESB

A modelagem do ESB foi realizada utilizando UML. A UML é uma linguagem, de uso padrão, para a elaboração da estrutura de projetos de software. Entre seus usos está a visualização, a especificação, a construção e a documentação de artefatos de sistemas de software (BOOCH et al, 2002).

Existem quatro tipos de itens na UML: estruturais, comportamentais, de agrupamento e anotacionais. Além desses itens, ela ainda abrange dois tipos de blocos de construção: relacionamentos e diagramas (BOOCH et al, 2002).

A UML é uma linguagem visual utilizada para criar modelos de programas, os quais podem ser entendidos como uma representação diagramática dos programas (SHALLOWAY, 2002). Em Booch et al (2002, p.26), consta que “um diagrama é a apresentação gráfica de um conjunto de elementos, geralmente representada como gráficos de vértices e arcos”.

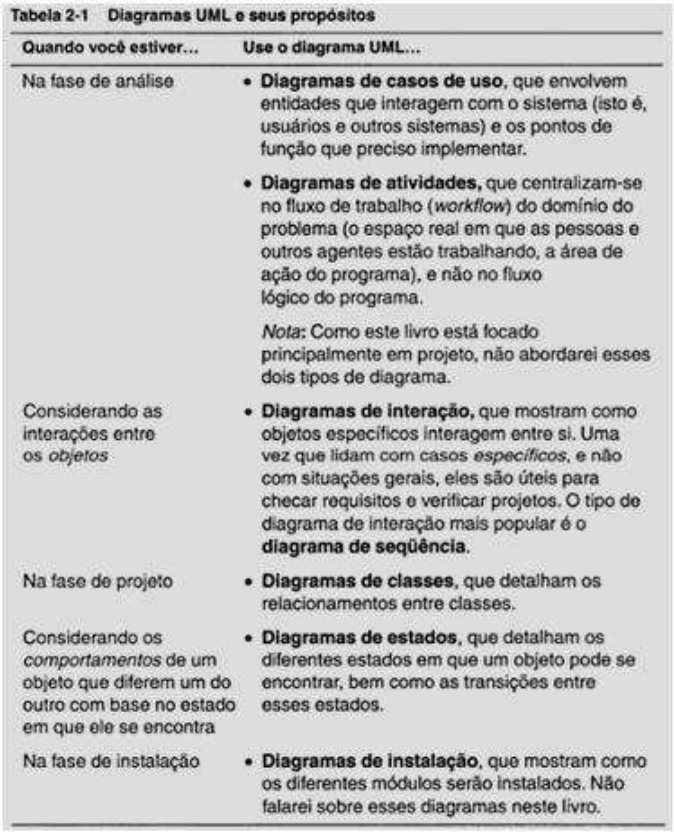

Tabela 2-1 Diagramas UML e seus propósitos

Quando você estiver...	Use o diagrama UML...
Na fase de análise	• **Diagramas de casos de uso**, que envolvem entidades que interagem com o sistema (isto é, usuários e outros sistemas) e os pontos de função que preciso implementar. • **Diagramas de atividades,** que centralizam-se no fluxo de trabalho (*workflow*) do domínio do problema (o espaço real em que as pessoas e outros agentes estão trabalhando, a área de ação do programa), e não no fluxo lógico do programa. *Nota*: Como este livro está focado principalmente em projeto, não abordarei esses dois tipos de diagrama.
Considerando as interações entre os *objetos*	• **Diagramas de interação,** que mostram como objetos específicos interagem entre si. Uma vez que lidam com casos *específicos*, e não com situações gerais, eles são úteis para checar requisitos e verificar projetos. O tipo de diagrama de interação mais popular é o **diagrama de seqüência**.
Na fase de projeto	• **Diagramas de classes**, que detalham os relacionamentos entre classes.
Considerando os *comportamentos* de um objeto que diferem um do outro com base no estado em que ele se encontra	• **Diagramas de estados**, que detalham os diferentes estados em que um objeto pode se encontrar, bem como as transições entre esses estados.
Na fase de instalação	• **Diagramas de instalação**, que mostram como os diferentes módulos serão instalados. Não falarei sobre esses diagramas neste livro.

Figura 6.1 – Diagramas UML e seus propósitos
Fonte: (SHALLOWAY, 2002, p. 54)

Dos treze diagramas relacionados por Booch et al (2002), seis são destacados por Shalloway (2002), conforme apresentado na figura 6.1. Este trabalho se utiliza dos diagramas de casos de uso, de pacotes, de classes, de atividade e de sequência.

6.3.1 Diagrama de casos de uso

Diagramas de casos de uso são um dos diagramas disponíveis na UML para a modelagem de aspectos dinâmicos de sistemas e têm um papel central para a modelagem do comportamento do sistema, subsistema ou de uma classe (BOOCH et al, 2002).

A figura 6.2 exibe o digrama de casos de uso do ESB proposto, expondo os pacotes, os casos de uso e os atores, e os relacionamentos entre eles. O sistema é dividido em dois pacotes. No pacote Configuração do ESB, estão inseridos o ator Usuário e o caso de uso Configurar ESB. Este é responsável por apresentar uma interface gráfica para a entrada de dados que fornecem parâmetros para o funcionamento do barramento.

No pacote chamado de Gestão do envio, transformação e recebimento de dados, os atores Folha de Pagamento e STSWin – sistema de ponto eletrônico - trocam informações, enviando e recebendo dados através dos casos de uso Enviar Dados e Receber Dados. Estes casos de uso se utilizam de outro, chamado Modificar Dados, o qual procede na transformação dos dados, de acordo com o que foi configurado pelo ator Usuário.

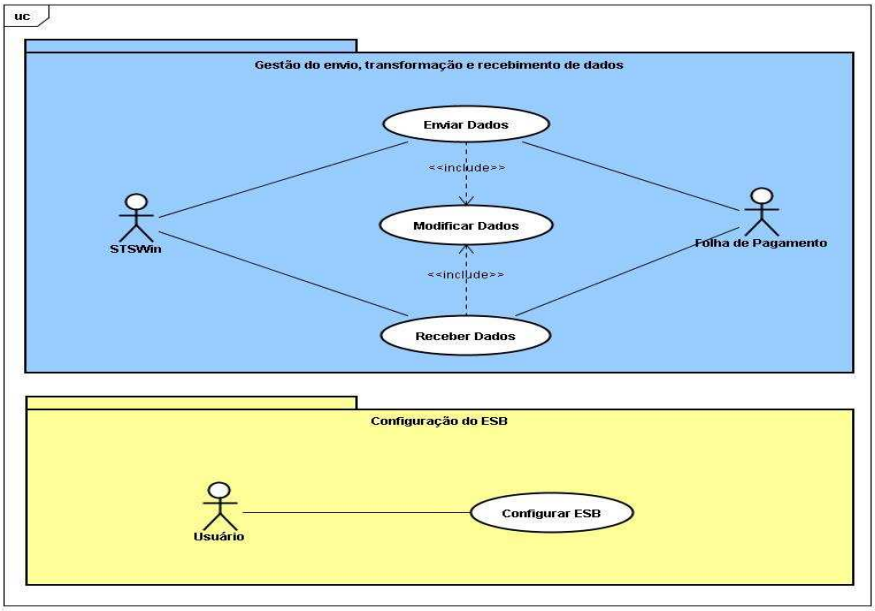

Figura 6.2 – Diagrama de casos de uso do ESB
Fonte: do Autor

Os quadros a seguir detalham cada um dos casos de uso apresentados na figura 6.2. O modelo de quadro utilizado para esse detalhamento foi adaptado a partir de várias sugestões apresentadas por Cockburn (2001).

Nome do Caso de Uso	Enviar Dados
Descrição Resumida	Controla a maneira como os dados serão enviados, do STSWin para a Folha de Pagamento ou vice-versa. O ESB verifica periodicamente os conectores configurados para envio, e monitora o local onde o arquivo será depositado, iniciando o processo quando um arquivo é detectado.
Ator Primário	STSWin ou Folha de Pagamento
Demais Atores	
Pré-condições	Dados devem estar prontos para serem enviados.
Acionadores	O ESB detecta o depósito do arquivo com os dados a serem enviados para a Folha de Pagamento ou vice-versa.
Cenário de Sucesso Principal	1. O ESB detecta que um arquivo foi depositado no local especificado no momento da configuração; 2. O ESB verifica o meio de comunicação que deverá ser utilizado, identificando o conector a ser utilizado; 3. O conector verifica se há *script* de modificação a ser executado e, caso afirmativo, o executa; 4. O ESB solicita ao conector que realize a conexão; 5. O ESB solicita ao conector que envie o arquivo; 6. O conector entrega o arquivo e, caso deva, toma a ação determinada para o arquivo de origem, apagando-o, movendo-o ou renomeando-o; 7. O ESB solicita ao conector que feche a conexão; 8. É gravado um bilhete com o resultado do envio, para fins de *log*.
Extensões	1. O *script* não é executado corretamente: 1.1. Se esgotadas as tentativas, aborta o envio; 1.2. Senão tenta novamente;

	2. Não há sucesso na conexão: 2.1. Se esgotadas as tentativas, aborta o envio; 2.2. Senão tenta novamente; 3. O conector não consegue completar a ação determinada ao arquivo de origem: 3.1. Se esgotadas as tentativas, aborta o envio; 3.2. Senão tenta novamente. 4. Não há ação configurada ou não se aplica ao tipo de conexão: 4.1. Não executa nenhuma ação.
Questões Abertas	Há casos em que, se configurado um conector para o envio, não há a necessidade de se criar um conector para recebimento, pois a ação torna-se redundante.

Quadro 6.1 – Caso de uso Enviar Dados
Fonte: Do autor

Nome do Caso de Uso	Receber Dados
Descrição Resumida	Controla a maneira como os dados serão recebidos pelo STSWin, vindos da Folha de Pagamento ou vice-versa. O ESB verifica periodicamente os conectores configurados para recebimento, e monitora o local onde o arquivo foi depositado, iniciando o processo quando um arquivo é detectado.
Ator Primário	STSWin ou Folha de Pagamento
Demais Atores	
Pré-condições	Dados devem estar prontos para serem recebidos.
Acionadores	O ESB detecta o depósito do arquivo com os dados a serem recebidos.
Cenário de Sucesso Principal	1. O ESB verifica o meio de comunicação que deverá ser utilizado, identificando o conector a ser utilizado; 2. O ESB solicita ao conector que realize a conexão; 3. O ESB verifica se há arquivo depositado no local especificado no momento da na configuração;

	4. O ESB solicita ao conector que receba o arquivo; 5. O conector verifica se há *script* de modificação a ser executado e, caso afirmativo, o executa; 6. O conector deposita o arquivo no local especificado e, caso deva, toma a ação determinada para o arquivo de origem apagando-o, movendo-o ou renomeando-o; 7. O ESB solicita ao conector que feche a conexão; 8. É gravado um bilhete com o resultado do recebimento, para fins de *log*.
Extensões	1. O *script* não é executado corretamente: 1.1. Se esgotadas as tentativas, aborta o envio; 1.2. Senão tenta novamente; 2. Não há sucesso na conexão: 2.1. Se esgotadas as tentativas, aborta o envio; 2.2. Senão tenta novamente; 3. O conector não consegue completar a ação determinada ao arquivo de origem: 3.1. Se esgotadas as tentativas, aborta o envio; 3.2. Senão tenta novamente. 4. Não há ação configurada ou não se aplica ao tipo de conexão: 4.1. Não executa nenhuma ação.
Questões Abertas	Há casos em que, se configurado um conector para o envio, não há a necessidade de se criar um conector para recebimento, pois a ação torna-se redundante.

Quadro 6.2 – Caso de uso Receber Dados
Fonte: Do autor

Nome do Caso de Uso	Modificar Dados
Descrição Resumida	Executa o *script* determinado que altera o arquivo, seja em sua estrutura, ou seja no seu conteúdo.
Ator Primário	ESB

Demais Atores	
Pré-condições	O arquivo está sendo enviado ou recebido.
Acionadores	Pelos casos de uso Enviar Dados ou Receber Dados, o ESB, o conector utilizado é solicitado a executar o *script*.
Cenário de Sucesso Principal	1. O conector identifica o *script* a ser executado; 2. O conector executa o *script*, em tempo de execução, atendendo aos comandos estipulados.
Extensões	1. O *script* está vazio – não há comandos a executar: 1.1. Aborta a execução, sem retornar erro; 2. O *script* é executado com erro em algum dos comandos: 2.1. Aborta a execução, retornando erro.
Questões Abertas	O *script* pode conter comandos que podem executar ações que vão além da modificação do arquivo, podendo realizar uma cópia do mesmo, por exemplo, ou enviá-lo, paralelamente por algum meio para algum local ou pessoa.

Quadro 6.3 – Caso de uso Modificar Dados
Fonte: Do autor

Nome do Caso de Uso	Configurar ESB
Descrição Resumida	Através de uma interface gráfica, fornece ao usuário os meios necessários para criar um novo conector ou alterar ou excluir um já existente.
Ator Primário	Usuário
Demais Atores	
Pré-condições	Execução de um arquivo executável, específico para prover a interface para configuração.
Acionadores	O usuário executa o software de configuração.
Cenário de Sucesso Principal	1. O usuário executa o software de configuração; 2. O software lista os conectores já configurados e apresenta opções para inclusão, alteração ou exclusão de conector.
Extensões	1. Inclusão de conector:

	1.1. O usuário indica a direção do conector (envio ou recebimento); 1.2. De acordo com a opção anterior, o software exibe uma lista de tipos de conexão possíveis; 1.3. O usuário seleciona o tipo desejado; 1.4. O software exibe os campos de preenchimento necessário para executar a conexão, quando solicitado; 1.5. O usuário salva ou abandona a configuração; 1.6. O software retorna para a tela inicial, exibindo os conectores já configurados. 2. Alteração de conector: 2.1. O software exibe os campos com os dados atuais do conector, permitindo sua modificação; 2.2. O usuário altera os dados que deseja; 2.3. O usuário salva ou abandona as modificações realizadas; 2.4. O software retorna para a tela inicial, exibindo os conectores já configurados. 3. Exclusão de conector: 3.1. O software solicita a confirmação para exclusão; 3.2. O usuário confirma a exclusão; 3.2.1. O software exclui o conector; 3.3. O usuário desiste da exclusão; 3.4. O software retorna para a tela inicial, exibindo os conectores já configurados.
Questões Abertas	

Quadro 6.4 – Caso de uso Configurar ESB
Fonte: Do autor

6.3.2 Diagrama de pacotes

Na UML, um pacote é um mecanismo de propósito geral, usado para organizar os elementos em grupo e dentro de uma hierarquia. Um pacote pode conter outros elementos, como classes, casos de uso, diagramas e até outros pacotes (BOOCH et al, 2002).

A figura 6.3 mostra o diagrama de pacotes do ESB proposto. O pacote Gestão do envio, transformação e recebimento de dados é dependente do pacote Configuração do ESB. Ou seja, são os dados inseridos através da configuração pelo usuário, que dão os parâmetros para o funcionamento do ESB.

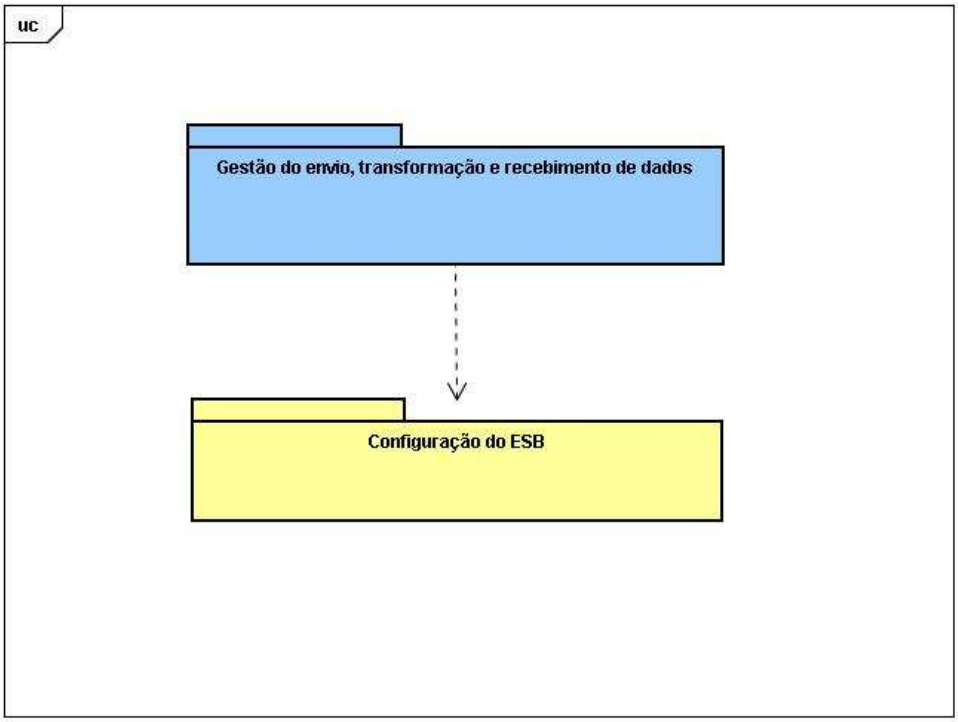

Figura 6.3 – Diagrama de pacotes
Fonte: Do autor

6.3.3 Diagrama de classes

Conforme Booch et al (2002),

> “um diagrama de classe mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. São importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistemas executáveis por intermédio de engenharia de produção e reversa”.

A figura 6.4 apresenta o diagrama de classes do ESB proposto.

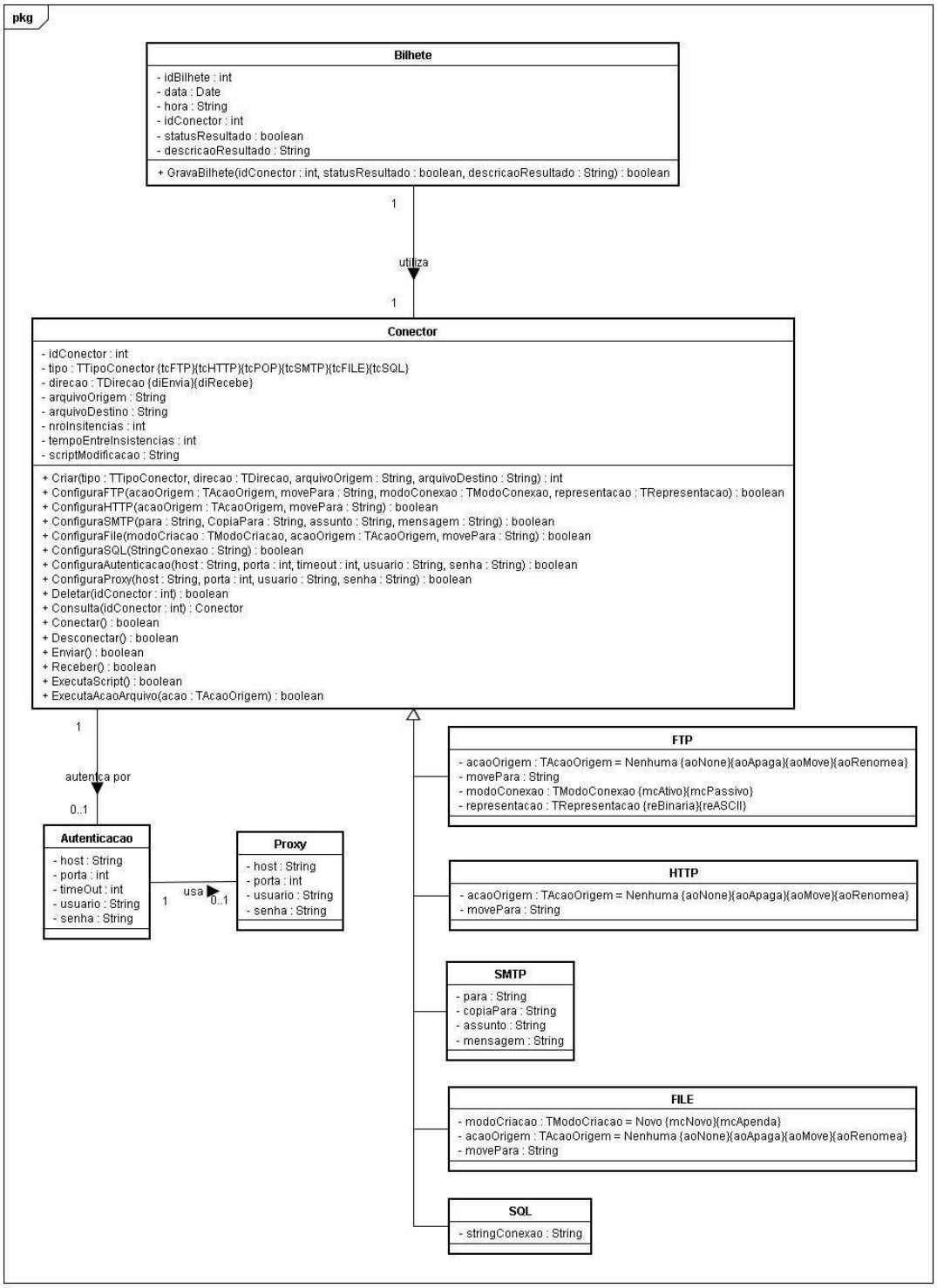

Figura 6.4 – Diagrama de classes do ESB
Fonte: do Autor

Como se pode perceber na figura 6.4, uma das principais classes é a Conector. Ela é a responsável por prover a conexão e envio e/ou recebimento de dados entre os aplicativos. Além disso, ela é quem possui o método que executa o *script* determinado para a transformação dos dados.

A classe Conector é a generalização das classes especializadas FTP, HTTP, POP3, SMTP, FILE e SQL. Essas especializações contêm atributos específicos, utilizados nas conexões através dos meios de mesmo nome.

Os atributos que guardam as informações necessárias para completar a conexão, são inerentes da classe Autenticacao. Essa por sua vez, pode fazer uso da classe Proxy, em ambientes que utilizam esse recurso.

O registro de todos os eventos de envio e recebimento, ou suas tentativas, fica contido na classe Bilhete – nome escolhido em analogia ao serviço de bilhetagem utilizado em telecomunicações. Ela possui atributos que guardam informações como data e hora em que ocorreu o evento e o seu resultado.

6.3.4 Diagrama de atividade

Diagramas de atividade são um dos diagramas utilizados na UML para a modelagem de aspectos dinâmicos de sistemas. Ele é essencialmente um gráfico mostrando o fluxo de controle de uma atividade para outra (BOOCH et al, 2002).

Como forma de complementar os fluxos dos casos de uso Enviar Dados e Receber Dados, as figuras 6.5 e 6.6 ilustram os respectivos diagramas de atividade.

A figura 6.5 mostra graficamente o fluxo descrito no caso de uso Enviar Dados. Além disso, exibe também os sub-fluxos de cada uma das exceções. O fluxo descrito no caso de uso Modificar Dados está incluído no mesmo diagrama, uma vez que ele é executado pela classe Enviar Dados.

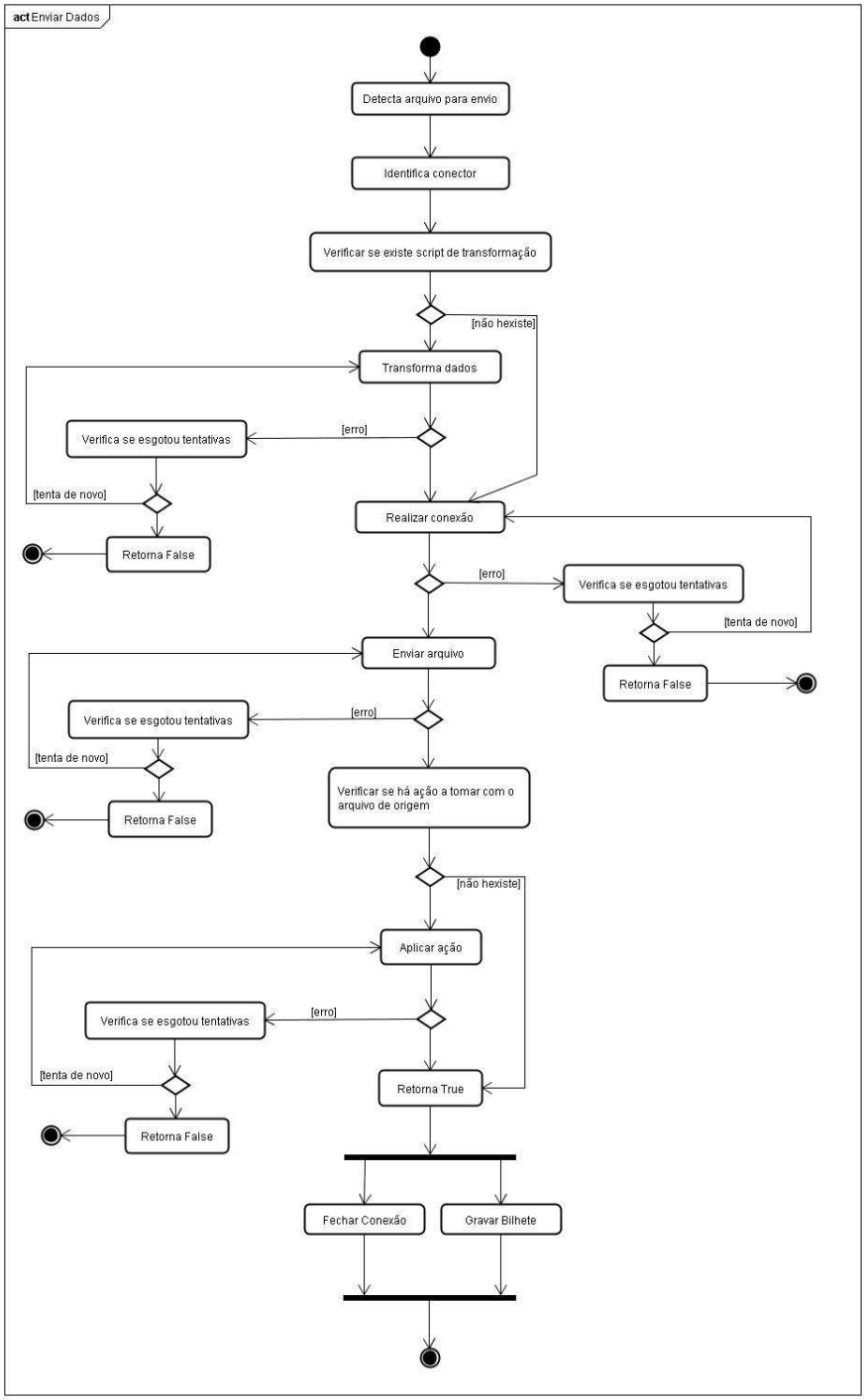

Figura 6.5 – Diagrama de atividade da classe Enviar Dados
Fonte: do Autor

Já os fluxos contidos na descrição da classe Receber Dados são detalhados de forma gráfica pelo diagrama de atividade da figura 6.6. A exemplo da figura 6.5, este diagrama também exibe os sub-fluxos de cada uma das exceções e traz de forma inerente o fluxo da classe Modificar Dados.

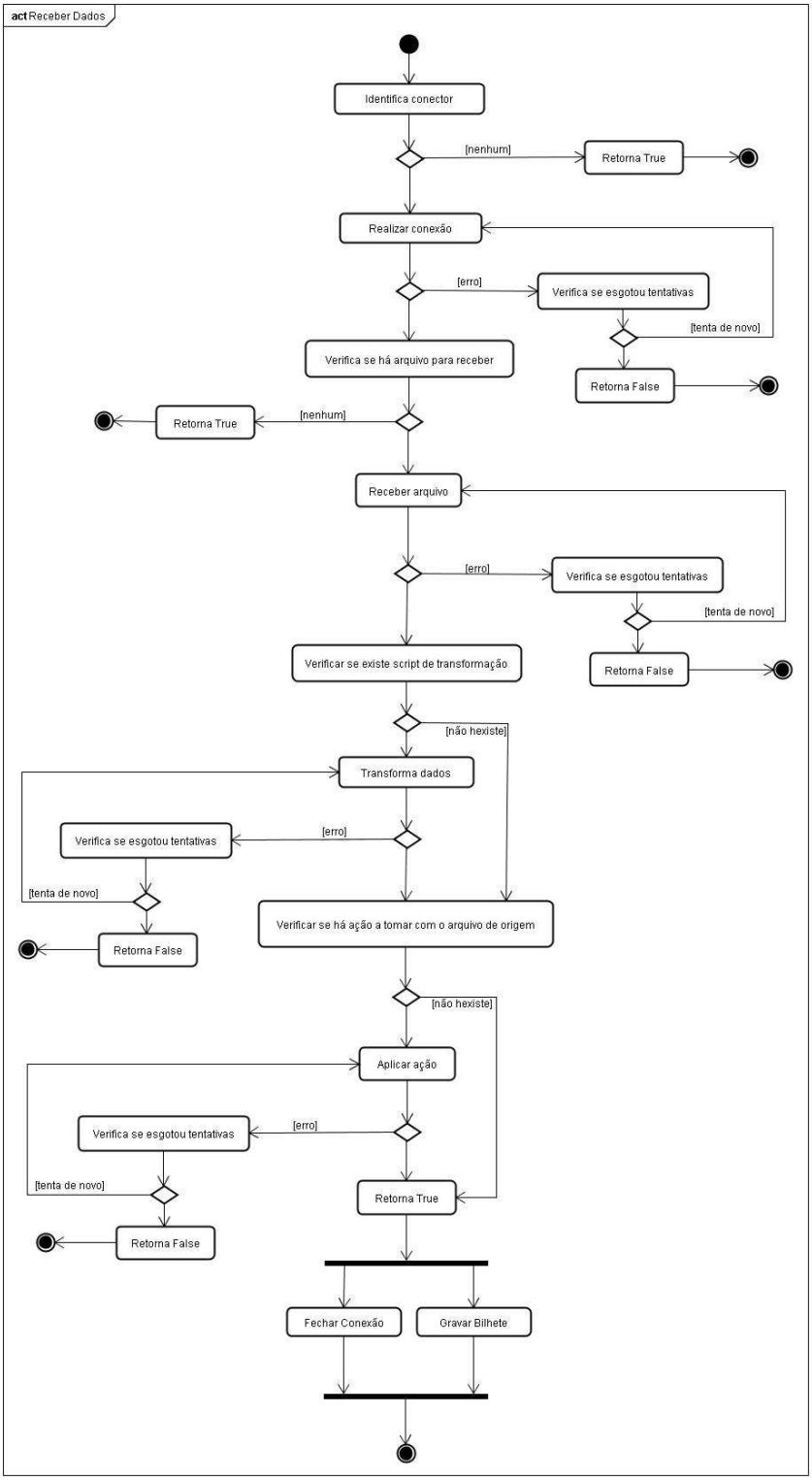

Figura 6.6 – Diagrama de atividade da classe Receber Dados
Fonte: do Autor

6.3.5 Diagrama de sequência

Assim como os diagramas de atividade, os diagramas de sequência também são utilizados para a modelagem de aspectos dinâmicos de sistemas. Ele dá ênfase à ordenação temporal das mensagens, representando graficamente uma tabela que mostra objetos distribuídos no eixo X e mensagens, em ordem crescente de tempo, no eixo Y (BOOCH et al, 2002).

Ainda em complementação dos fluxos dos casos de uso Enviar Dados e Receber Dados, as figuras 6.7 e 6.8 exibem os seus respectivos diagramas de sequência.

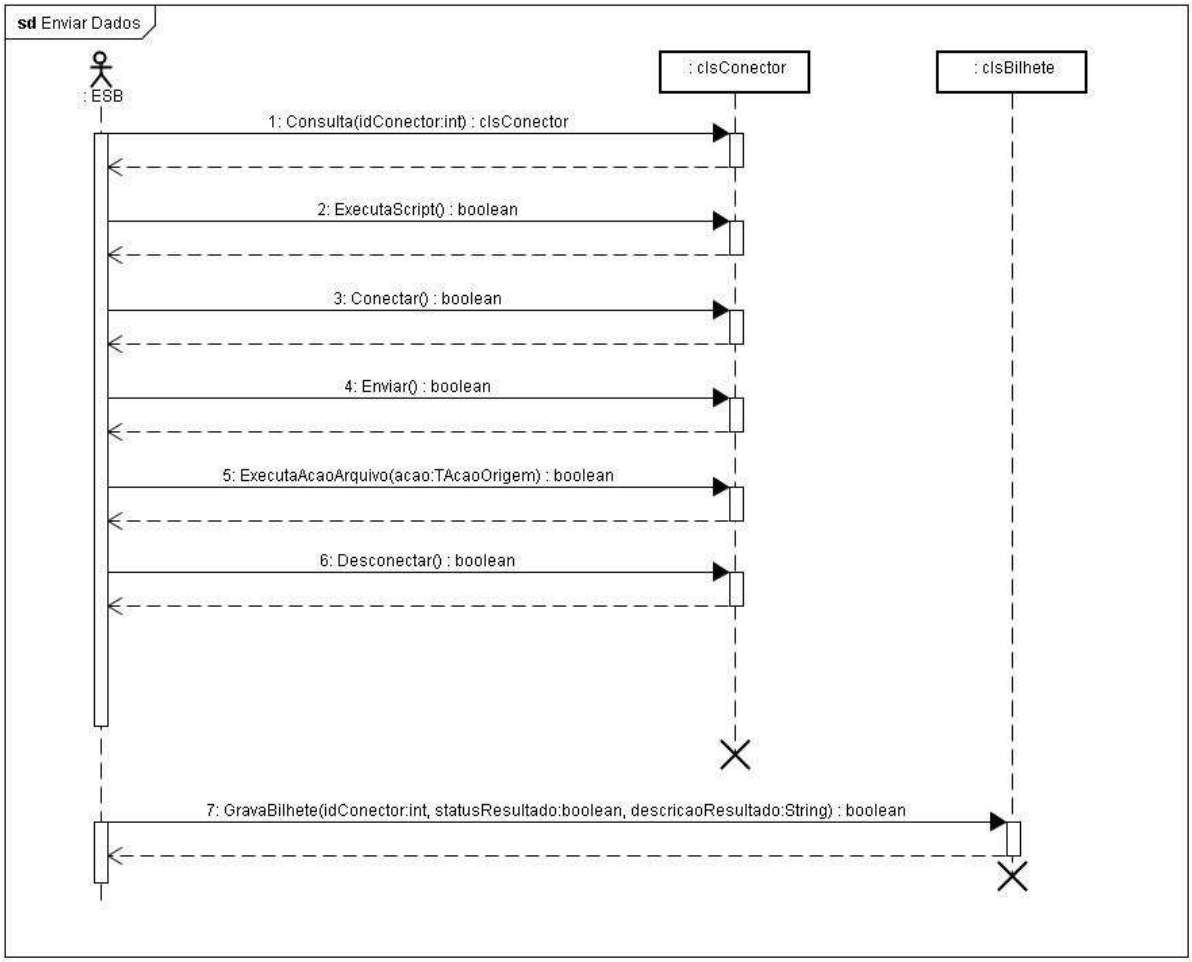

Figura 6.7 – Diagrama de sequência da classe Enviar Dados
Fonte: do Autor

A figura 6.7 exibe a troca de mensagens entre o barramento e as classes Conector e Bilhete. Desta forma é ilustrado graficamente o fluxo principal descrito no Caso de Uso Enviar Dados. Temporalmente, são exibidas as trocas de mensagens que garantem o envio dos dados, desde a consulta ao conector responsável, até a desconexão com o meio utilizado. Após finalizado o envio, o barramento interage com a classe Bilhete a fim de realizar a gravação do *log* deste envio.

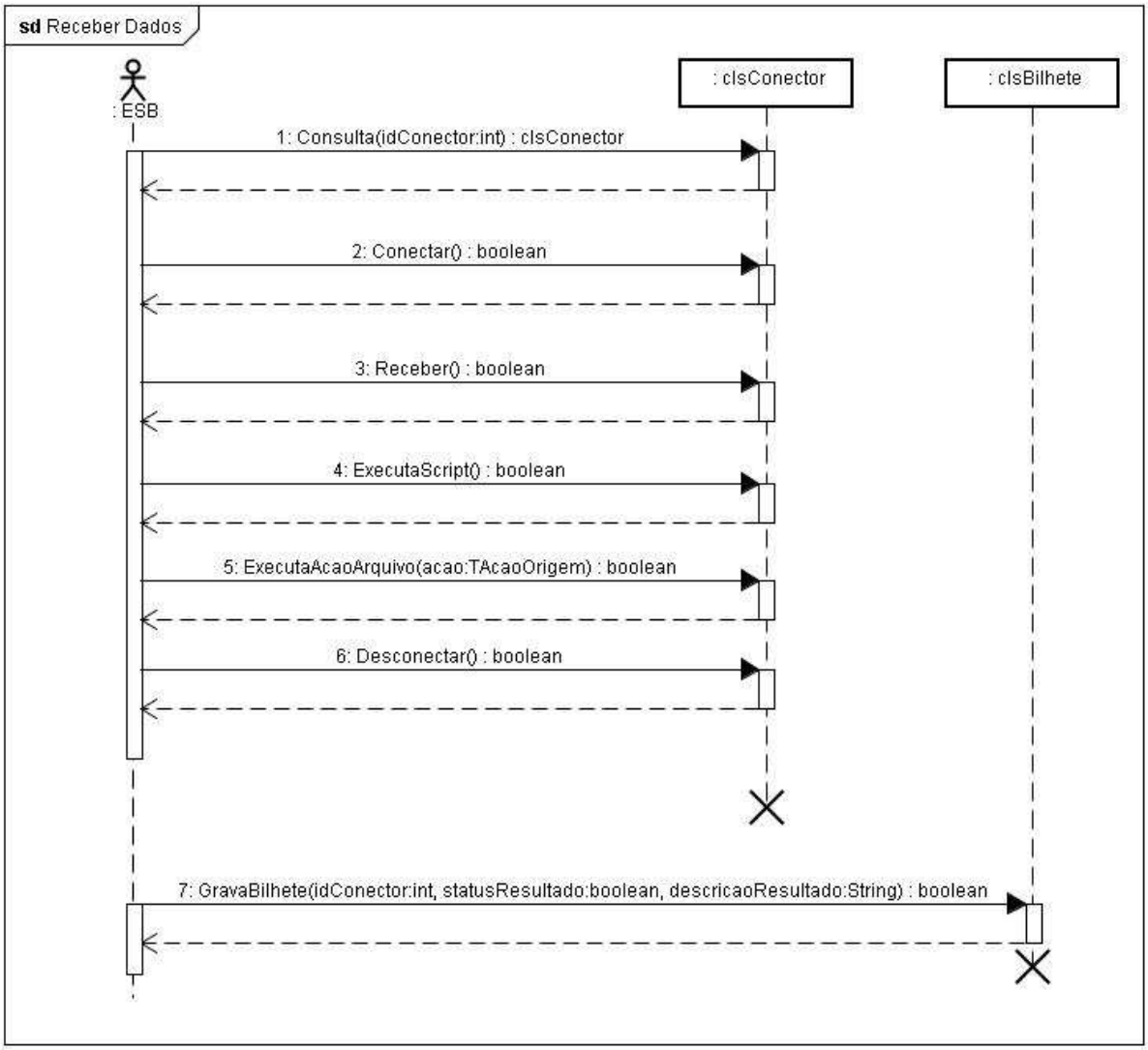

Figura 6.8 – Diagrama de sequência da classe Receber Dados
Fonte: do Autor

Com a mesma finalidade, a figura 6.8 ilustra as interações necessárias para o sucesso do Caso de Uso Receber Dados. Igualmente, exibe todo o fluxo temporal do caso de uso, entre o barramento e as classes Conector e Bilhete.

6.4 DESENVOLVIMENTO DO PROTÓTIPO

Com o intuito de se realizar a validação do modelo proposto, optou-se pelo desenvolvimento de um protótipo que reunisse os requisitos básicos do ESB, executando o envio de um determinado arquivo de dados, pela folha de pagamento ao sistema de ponto eletrônico, através de um dos meios de interconexão. Dessa forma, este protótipo contempla parte do ESB. Para completá-lo, basta apenas a implementação de algumas classes e métodos, das quais o funcionamento do protótipo não é dependente. As principais características deste protótipo são apresentadas a seguir.

6.4.1 Requisitos do teste

Como forma de testar o funcionamento do protótipo, foi proposto um teste que consiste em:

- Ler um arquivo XML disponibilizado pela folha de pagamento (Anexo A);
- Aplicar em tempo de execução, um *script* cadastrado pelo usuário, atendendo o *layout* (Anexo B) sugerido pelo sistema de ponto eletrônico, garantindo o formato esperado pelo mesmo;
- Gerar um terceiro arquivo com o novo formato e nome definidos;
- Enviar esse arquivo para um repositório em um servidor de FTP;
- Mover o arquivo original para a pasta determinada pelo usuário.

Este teste atende a todos os requisitos do ESB proposto, que são: Serviço, Conectividade, Roteamento, Transformação, Repositório de Dados e Configurar ESB. Da mesma forma, corresponde a todos os casos de uso – Enviar Dados e Receber Dados, que incluem Modificar Dados e ainda Configurar ESB.

6.4.2 Implementação do protótipo

O protótipo foi implementado utilizando o ambiente de desenvolvimento Delphi (versão 7.0). A opção pelo Delphi se deu em razão de que a atualmente a empresa escolhida utiliza esta linguagem no desenvolvimento de suas aplicações.

A figura 6.9 é uma reprodução gráfica dos componentes do Delphi que são responsáveis por atender a execução do serviço. O primeiro deles, onde os demais estão inseridos, é um componente da classe TService. Ele executa o serviço e monitora os eventos dos demais componentes. Destacam-se dois conjuntos de componentes. O primeiro conjunto, posicionado à esquerda e acima, agrega os componentes necessários à execução do *script* estipulado. O segundo, acima e à direita, contém os componentes responsáveis pela consulta aos arquivos XML. Além disso, há ainda um componente TTimer, que faz a verificação da disponibilidade de novos dados periodicamente, a cada 10 segundos. E por último, um componente que disponibiliza acesso ao servidor de FTP.

Figura 6.9 – Componentes do Delphi responsáveis pela execução do serviço
Fonte: do Autor

6.4.3 Armazenamento dos dados

As informações pertinentes à configuração e que são parâmetros para o funcionamento do ESB são mantidas em arquivos XML. Desta forma é dispensada a necessidade da instalação e utilização de um banco de dados.

A figura 6.10 ilustra o formato de um dos arquivos XML utilizados pelo protótipo. Na parte superior, há informações gerais sobre o formato do arquivo. Logo após, são exibidas

as definições dos campos do arquivo. E por último, os valores atribuídos a cada um deles, sendo que cada registro é representado por uma marcação do tipo <ROW>. Os demais arquivos XML estão dispostos como anexos a este trabalho.

```
<?xml version="1.0" standalone="yes" ?>
- <DATAPACKET Version="2.0">
  - <METADATA>
    - <FIELDS>
        <FIELD attrname="idConector" fieldtype="i4" />
        <FIELD attrname="Descricao" fieldtype="string" WIDTH="100" />
        <FIELD attrname="Tipo" fieldtype="string" WIDTH="10" />
        <FIELD attrname="Direcao" fieldtype="string" WIDTH="10" />
        <FIELD attrname="ArquivoOrigem" fieldtype="bin.hex" SUBTYPE="Binary" />
        <FIELD attrname="ArquivoDestino" fieldtype="bin.hex" SUBTYPE="Binary" />
        <FIELD attrname="NroInsistencias" fieldtype="i4" />
        <FIELD attrname="TempoEntreInsistencias" fieldtype="i4" />
        <FIELD attrname="ScriptModificacao" fieldtype="bin.hex" SUBTYPE="Binary" />
      </FIELDS>
      <PARAMS DEFAULT_ORDER="1" LCID="0" />
    </METADATA>
  - <ROWDATA>
      <ROW idConector="1" Descricao="STSWin p/ Folha - SMTP" Tipo="SMTP" Direcao="diEnvia" />
      <ROW idConector="2" Descricao="STSWin p/ Folha - File" Tipo="FILE" Direcao="diEnvia" />
      <ROW idConector="3" Descricao="Eventos do Fechamento" Tipo="FILE" />
      <ROW idConector="4" Descricao="Funcionarios com Falta - SMTP" Tipo="SMTP" />
      <ROW idConector="5" Descricao="Folha - POP3" Tipo="POP3" />
      <ROW idConector="6" Descricao="Funcionarios da Folha p/ STSWin - FTP" Tipo="FTP"
        Direcao="diEnvia"
        ArquivoOrigem="QzpcU2Fudml0cm9uXEZvbGhhIFBndG9cZnVuY2lvbmFyaW8ueG1s"
        ArquivoDestino="RnVuY2lvbmFyaW8udHh0" NroInsistencias="3" TempoEntreInsistencias="5"
        ScriptModificacao="dmFyDQogIFMgOiBTdHJpbmc7DQoNCmJlZ2luDQogIFRleHRvTm92byA6PSAnJzsNCg0K
    </ROWDATA>
</DATAPACKET>
```

Figura 6.10 – Arquivo XML contendo os dados da classe Conector
Fonte: do Autor

6.4.4 Aplicativo

O protótipo foi implementado na forma de dois aplicativos. O primeiro deles, na forma de um aplicativo gráfico, possui uma interface para a interação com o usuário. Através desse aplicativo, o usuário cria um novo Conector e define os seus parâmetros de funcionamento. Já o segundo, é o ESB propriamente dito. Foi implementado na forma de um serviço, que roda de forma transparente ao usuário. Este serviço faz o monitoramento da disponibilidade de novos dados e gera os eventos necessários para a conclusão da tarefa determinada pelo usuário.

A interface gráfica para configuração do ESB é exibida na figura 6.11. À esquerda são listados os conectores já cadastrados e os botões de tarefas. No outro lado da tela, estão dispostos os campos para entrada de dados. Destacam-se dentre eles, o campo Script de

Modificação. Nele o usuário escreve os comandos que devem ser executados pelo ESB, em tempo de execução, sem a necessidade de uma recompilação do código fonte e geração de um novo arquivo executável.

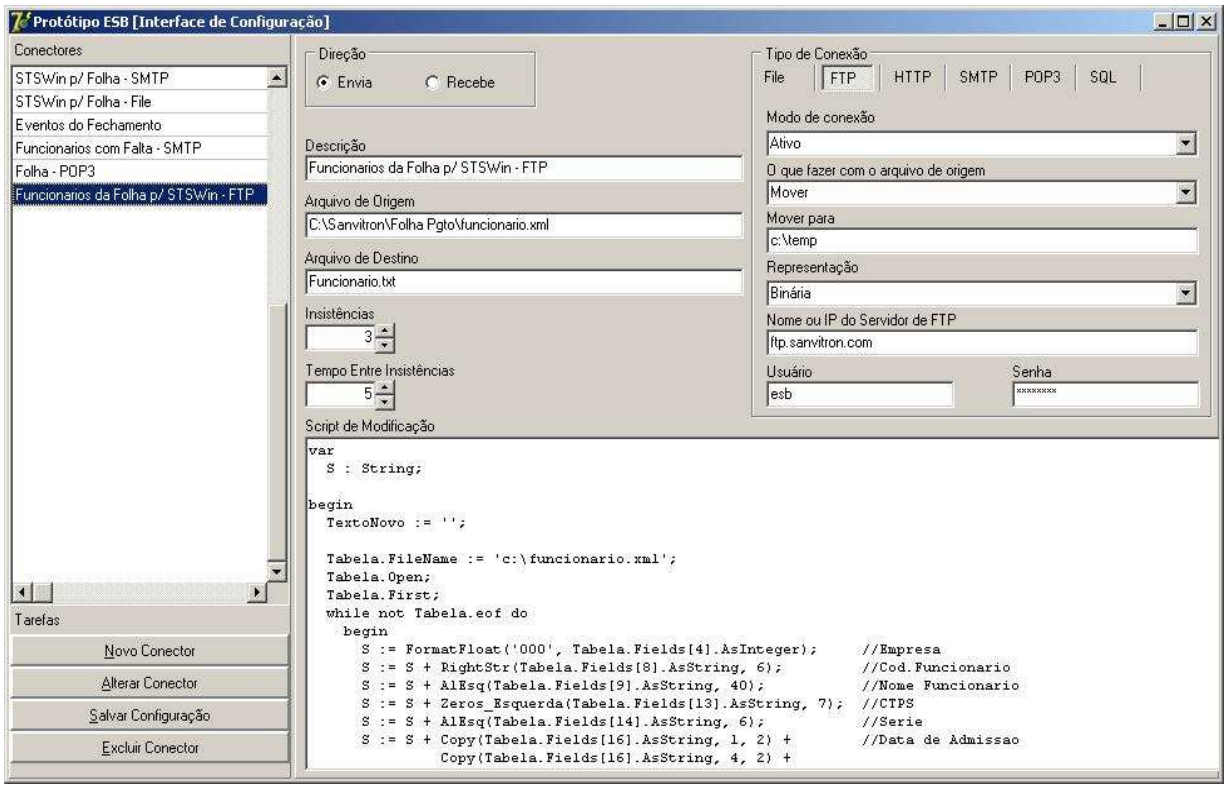

Figura 6.11 – Interface de configuração do ESB
Fonte: do Autor

Já a figura 6.12 comprova a execução do serviço implementado, com informações sobre o arquivo executável e sua localização. Além disso, exibe o status do serviço (iniciado) e sua forma de inicialização (de forma automática, junto com o sistema operacional).

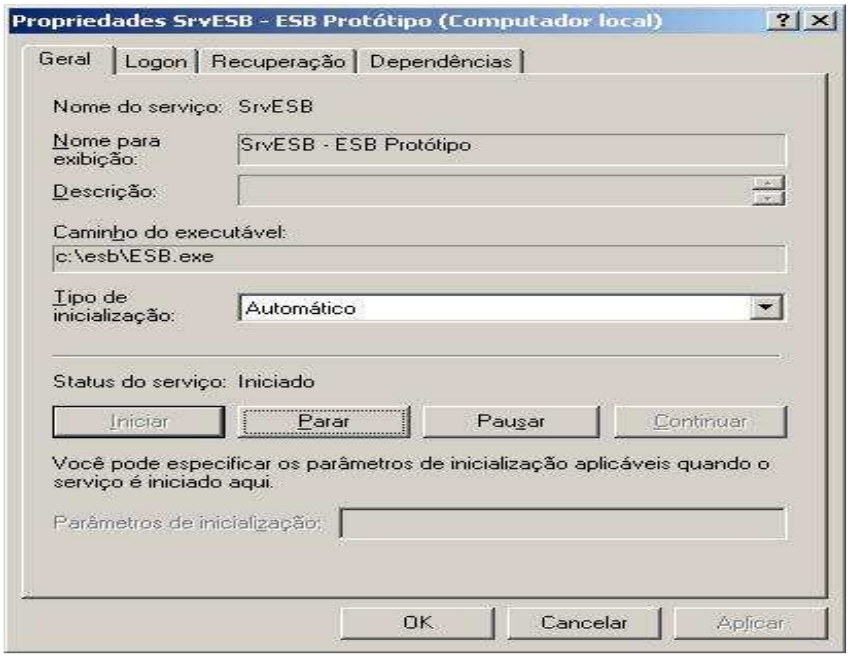

Figura 6.12 – Serviço rodando no sistema operacional
Fonte: consulta à lista de serviços do sistema operacional

6.4.5 Execução do teste

Durante o desenvolvimento do protótipo, nem todas as classes e métodos foram implementados. O foco na implementação do código fonte do protótipo foi atender o teste proposto. Através desse teste comprovou-se o atendimento de todos os requisitos arrolados. O quadro 6.5 elenca as classes e métodos modelados, e os que realmente foram implementados. Além disso, ele destaca os motivos da implementação ou não de cada um deles.

Tipo de Objeto	Nome	Pertence a	Imp.	Motivo
Classe	Conector		X	Fundamental para o funcionamento
Classe	FTP		X	Meio de comunicação dos requisitos do teste
Classe	HTTP			Fora dos requisitos do teste
Classe	SMTP			Fora dos requisitos do teste
Classe	FILE			Fora dos requisitos do teste
Classe	SQL			Fora dos requisitos do teste
Classe	POP3			Fora dos requisitos do teste
Classe	Autenticacao		X	Tem dados fundamentais para concluir a conexão

Classe	Proxy			Não há Proxy ativo no ambiente onde o teste foi realizado
Classe	Bilhete			Sem relevância para execução do teste.Deve realizar apenas o registro do *log* do envio ou recebimento
Método	Criar		X	Instancia um objeto Conector
Método	ConfiguraFTP	Conector	X	Instancia um objeto FTP
Método	ConfiguraHTTP	Conector		Fora dos requisitos do teste
Método	ConfiguraSMTP	Conector		Fora dos requisitos do teste
Método	ConfiguraFile	Conector		Fora dos requisitos do teste
Método	ConfiguraSQL	Conector		Fora dos requisitos do teste
Método	ConfiguraAutenticacao	Conector	X	Instancia um objeto Autenticacao
Método	ConfiguraProxy	Conector		Não há Proxy ativo no ambiente onde o teste foi realizado
Método	Deletar	Conector		Não houve a necessidade de apagar nenhum registro para conclusão do teste
Método	Consulta	Conector		A consulta foi realizada no arquivo XML
Método	Conectar	Conector	X	Realiza a conexão com o meio definido
Método	Desconectar	Conector	X	Desconecta do meio
Método	Enviar	Conector	X	Utilizado para enviar os dados, conforme requisitos do teste
Método	Receber	Conector		
Método	ExecutaScript	Conector	X	Usado para transformar os dados, de acordo com os requisitos do teste
Método	ExecutaAcaoArquivo	Conector	X	Conforme requisitos do teste, deve mover o arquivo. Completamente implementado, para opções de renomear e apagar também
Método	GravaBilhete	Bilhete		Sem relevância para execução do teste.Deve realizar apenas o registro do *log* do envio ou recebimento

Quadro 6.5 – Implementações realizadas ou não
Fonte: do autor

O atendimento aos requisitos de um ESB foi comprovado através do teste. A sequência do teste proposto (conforme tópico 6.4.1) foi fielmente cumprida através da implementação do fluxo principal descrito no caso de uso Enviar Dados. Conseqüentemente, o caso de uso Receber Dados também foi validado, haja vista que existe apenas uma pequena modificação, invertendo o sentido dos dados.

Então, através do teste realizado, o arquivo XML (Anexo A) do aplicativo da folha de pagamento foi transformado pelo *script* (Apêndices A) em atendimento ao layout estabelecido (Anexo B) e postado no servidor de FTP, estando à disposição do aplicativo de ponto eletrônico. Todo o código fonte do protótipo está anexado a este trabalho.

CONCLUSÃO

A Arquitetura Orientada a Serviços tem se destacado pela forma como orquestra a integração entre diferentes aplicativos. SOA abstrai as regras e funções de negócio, disponibilizando-as em forma de serviços. Constatou-se que, por natureza os aplicativos são heterogêneos, e que uma das características mais fortes de SOA é lidar com esta heterogeneidade. Ainda, SOA minimiza as dependências entre os aplicativos, proporcionando um fraco acoplamento entre os mesmos. Isso contribui para a tolerância a falhas e a flexibilidade na integração. Outros pontos fortes puderam ser observados, como o impacto positivo sobre as organizações, através da reusabilidade, customização e segurança na integração de dados.

Como parte integrante e fundamental de SOA, o ESB mostrou-se muito eficaz na condução da integração entre aplicativos. Observou-se, efetivamente, como o ESB posiciona-se entre as aplicações, centralizando a comunicação entre as mesmas e a maneira como desempenha as suas funções, principalmente de conectividade e transformação de dados.

O ESB confirmou-se como uma forma mais elegante de integração, em comparação com as demais maneiras estudadas. Além disso, através dos estudos realizados, pode-se verificar que a combinação SOA/ESB tem sido utilizada em projetos comerciais e acadêmicos, onde se destacam por suas vantagens frente a formas tradicionais de integração.

Grande parte dos benefícios do ESB pode ser observada na prática, através do desenvolvimento do protótipo. Com certeza, com a completa implementação e implantação do modelo proposto, todos os benefícios esperados (interoperabilidade, redução de dependência, flexibilidade, agilidade e reuso) estarão confirmados.

A contribuição deste trabalho é efetiva, no sentido de que fornece subsídios para o real aprimoramento dos processos de integração da Sanvitron. Dando-lhe condições reais de competitividade, ora pelo ganho de desempenho proporcionado por essa tecnologia, ora pela redução de custos, através do emprego da reusabilidade e agilidade no desenvolvimento. Além disso, o tempo dispensado na modelagem serviu para o aperfeiçoamento do conhecimento de uma parcela dos recursos de mão-de-obra da empresa, haja vista que o autor do presente trabalho é colaborador da mesma. E esse conhecimento pode agora ser aplicado na rotina de trabalho da equipe de desenvolvimento.

Sugere-se para trabalhos futuros, a completa implementação do modelo proposto e extensões de uso, que garantam a integração de outros aplicativos. Aliado a isso, pode-se realizar a aplicação dos conceitos apresentados neste trabalho na confecção de um

WebService, o que garantiria a integração de aplicativos que não estão residentes no mesmo ambiente físico. Pode-se ainda realizar a implementação deste modelo através de outras linguagens de programação, que utilizem código aberto, contribuindo para a disseminação do conhecimento através do software livre..

REFERÊNCIAS BIBLIOGRÁFICAS

BENEDETE JUNIOR, Antonio Carlos. **Roteiro para a definição de uma arquitetura SOA utilizando BPM**. Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de MBA em Tecnologia da Informação. São Paulo, 2007.

BOOCH, Grady. ET al. **UML Guia do Usuário**. 2. ed. Rio de Janeiro: Campus, 2002.

BOWERSOX, Donald J. ET al. **Gestão Logística de Cadeias de Suprimentos**. Bookman, 2006.

BROOKSHEAR, J. Glenn. **Ciência da Computação: Uma visão abrangente**. 7. ed. Bookman, 2005.

CAMBIUCCI, Waldemir. **Microsoft BizTalk Server 2006 R2 e o modelo ESB – Enterprise Service Bus.** 2008. Disponível em < http://blogs.msdn.com/wcamb/archive/2008/10/04/ microsoft-biztalk-server-2006-r2-e-o-modelo-esb-enterprise-service-bus.aspx>. Acesso em 25 de maio de 2009.

CARVALHO, Davi. **ESB: Mitos e Principais Funcionalidades**. Disponível em <http://soasimples.com/blog/?cat=11>. Acesso em 27 de maio de 2009.

COCKBURN, Alistair. **Escrevendo Casos de Uso Eficazes**. Um guia prático para desenvolvedores de software. Porto Alegre: Bookman, 2001.

CUMMINS, Fred A. **Integração de sistemas**: EAI (Enterprise application integration): arquiteturas para integração de sistemas e aplicações corporativas. Rio de Janeiro: Campus, 2002.

DEITEL, Harvey M. **XML: como programar**. Bookman, 2003.

FAYAD, Mohamed; Schmidt, Douglas. **Object-Oriented Application Frameworks**. Communications of the ACM, New York, v. 40, n. 10, p. 32-38, Oct. 1997.

GARCIA et al. **Suporte da Arquitetura Orientada a Serviços na Integração de Sistemas Médicos**. Dissertação apresentada como requisito à obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Itajubá. Itajubá, 2008.

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

MENEZES, Gabriel Nunes; GALOPPINI, Pedro Castello Branco. **Onix. Framework para Integração de Sistemas com Enterprise Service Bus**. Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação. Florianópolis, 2009.

MOREIRA, Mike. **Conheça um pouco mais sobre ESB (Enterprise Service Bus) e as Ferramentas IBM**, 2009. Disponível em <http://tec.brq.com/conheca-um-pouco-mais-sobre-esb-enterprise-service-bus-e-as-ferramentas-ibm/>. Acesso em 27 de maio de 2009.

PARDAL, Miguel Filipe Leitão. **Segurança de aplicações empresariais em arquitecturas de serviços**. Dissertação para obtenção do Grau de Mestre em Engenharia Informática e de Computadores. Lisboa, 2006.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **METODOLOGIA DO TRABALHO CIENTÍFICO**. Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. Novo Hamburgo: Feevale, 2009.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informações**. 3. ed. Rio de Janeiro: Brasport, 2005.

SAMPAIO, Cleuton. **Guia do Java Enterprise Edition 5** - Desenvolvendo aplicações corporativas. Rio de Janeiro: Brasport, 2007.

SANTOS, Alfredo Luiz dos. **Integração de Sistemas com Java**. Rio de Janeiro: Brasport, 2007.

SAWAYA, Márcia Regina. **Dicionário de informática & Internet**: inglês/português. Nobel, 2002.

SHALLOWAY, Alan. **Explicando padrões de projeto**: uma nova perspectiva em projeto orientado a objeto. Porto Alegre: Bookman, 2002.

SILVA, Roberto Ferreira Lima. **e-RH em um ambiente global e multicultural**. Distrito Federal: Senac, 2009.

SPACKMAN, Devin; SPEAKER, Mark. **Soluções de Integração Empresarial**. Tradução de João Tortello. São Paulo: Bookman, 2006.

ANEXOS

ANEXO A – Arquivo XML gerado pela Folha de Pagamento

ANEXO B – Layout definido pelo sistema de ponto eletrônico

ANEXO A – Arquivo XML gerado pela Folha de Pagamento

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <DATAPACKET Version="2.0">
- <METADATA>
- <FIELDS>
  <FIELD attrname="IDFuncionario" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="IDFuncao" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="IDSetor" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="Identidade" fieldtype="string" SUBTYPE="FixedChar" WIDTH="15" />
  <FIELD attrname="CodEmpresa" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="Empresa" fieldtype="string" SUBTYPE="FixedChar" WIDTH="60" />
  <FIELD attrname="Inativo" fieldtype="string" SUBTYPE="FixedChar" WIDTH="3" />
  <FIELD attrname="Cracha" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="CodFuncionario" fieldtype="string" SUBTYPE="FixedChar" WIDTH="20" />
  <FIELD attrname="Nome" fieldtype="string" SUBTYPE="FixedChar" WIDTH="60" />
  <FIELD attrname="IDCargo" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="codescala" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="Sexo" fieldtype="string" SUBTYPE="FixedChar" WIDTH="1" />
  <FIELD attrname="Ctps" fieldtype="string" SUBTYPE="FixedChar" WIDTH="16" />
  <FIELD attrname="Serie" fieldtype="string" SUBTYPE="FixedChar" WIDTH="12" />
  <FIELD attrname="EstadoCtps" fieldtype="string" SUBTYPE="FixedChar" WIDTH="2" />
  <FIELD attrname="DataAdm" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="DataNasc" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="Pis" fieldtype="string" SUBTYPE="FixedChar" WIDTH="12" />
  <FIELD attrname="Matricula" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="PermitirSaidaInt" fieldtype="string" SUBTYPE="FixedChar" WIDTH="3" />
  <FIELD attrname="Bloquear" fieldtype="string" SUBTYPE="FixedChar" WIDTH="3" />
  <FIELD attrname="DataAfastamento" fieldtype="string" SUBTYPE="FixedChar" WIDTH="10" />
  <FIELD attrname="NomeSetor" fieldtype="string" SUBTYPE="FixedChar" WIDTH="1" />
    </FIELDS>
  <PARAMS DEFAULT_ORDER="10" LCID="0" />
    </METADATA>
- <ROWDATA>
  <ROW IDFuncionario="0000000016" IDFuncao="0000000001" IDSetor="0000000001"
     Identidade="9074336621" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000016" CodFuncionario="00000000000000000016" Nome="ALEXANDRE
     PEREIRA DA ROSA." IDCargo="0000000002" codescala="0000000004" Sexo="M" Ctps="40369"
     Serie="00060" EstadoCtps="RS" DataAdm="27/01/2004" DataNasc="02/12/1978" Pis="12715757672"
     Matricula="16" PermitirSaidaInt="Sim" Bloquear="Não" DataAfastamento="" NomeSetor="" />
  <ROW IDFuncionario="0000000003" IDFuncao="0000000001" IDSetor="0000000001"
     Identidade="2066498136" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000003" CodFuncionario="00000000000000000003" Nome="ALINE MAGNI
     TRESSOLDI DE OLIVEIRA" IDCargo="0000000002" codescala="0000000004" Sexo="F" Ctps="0063738"
     Serie="00057" EstadoCtps="RS" DataAdm="03/01/2000" DataNasc="17/08/1978" Pis="12709637679"
     Matricula="3" PermitirSaidaInt="Sim" Bloquear="Não" DataAfastamento="" NomeSetor="" />
  <ROW IDFuncionario="0000009024" IDFuncao="0000000006" IDSetor="0000000001"
     Identidade="8109846587" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000029" CodFuncionario="00000000000000000029" Nome="CARLA FERNANDA
     DE OLIVEIRA JOHANN" IDCargo="0000000004" codescala="0000000004" Sexo="M" Ctps="6916355"
     Serie="001" EstadoCtps="0" DataAdm="01/07/2008" DataNasc="03/04/1986" Pis="12986809520"
     Matricula="29" PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor="" />
  <ROW IDFuncionario="0000009007" IDFuncao="0000000000" IDSetor="0000000000" Identidade="0"
     CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas" Inativo="Não"
     Cracha="0000000034" CodFuncionario="00000000000000000034" Nome="Cleuza Marina Magni Tressoldi"
     IDCargo="0000000000" codescala="0000000002" Sexo="F" Ctps="0" Serie="0" EstadoCtps="0"
     DataAdm="13/02/2006" DataNasc="19/09/1956" Pis="0" Matricula="0" PermitirSaidaInt="Sim"
     Bloquear="Não" NomeSetor="" />
  <ROW IDFuncionario="0000000004" IDFuncao="0000000001" IDSetor="0000000002"
     Identidade="1075349595" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000004" CodFuncionario="00000000000000000004" Nome="FREDERICO
     TIMMEN" IDCargo="0000000002" codescala="0000000004" Sexo="M" Ctps="0045479" Serie="00058"
     EstadoCtps="RS" DataAdm="02/03/2001" DataNasc="20/08/1981" Pis="12720355692" Matricula="4"
     PermitirSaidaInt="Sim" Bloquear="Não" DataAfastamento="" NomeSetor="" />
  <ROW IDFuncionario="0000000001" IDFuncao="0000000003" IDSetor="0000000004"
     Identidade="2043434782" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000001" CodFuncionario="00000000000000000001" Nome="GUIBSON HAAS DOS
     SANTOS" IDCargo="0000000005" codescala="0000000004" Sexo="M" Ctps="0" Serie="" EstadoCtps="RS"
     DataAdm="01/01/1997" DataNasc="24/12/1975" Pis="11383588-761" Matricula="1"
     PermitirSaidaInt="Sim" Bloquear="Não" DataAfastamento="" NomeSetor="" />
  <ROW IDFuncionario="0000009005" IDFuncao="0000000002" IDSetor="0000000007"
     Identidade="2057605764" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
     Inativo="Não" Cracha="0000000020" CodFuncionario="00000000000000000020" Nome="MARCIO CARDOSO"
     IDCargo="0000000001" codescala="0000000100" Sexo="M" DataAdm="01/08/2005"
```

```
   DataNasc="21/09/1974" Pis="0" Matricula="0" PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor=""
   />
<ROW IDFuncionario="0000009004" IDFuncao="0000000001" IDSetor="0000000001"
   Identidade="6070167918" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000019" CodFuncionario="00000000000000000019" Nome="MARCIO MACIEL DA
   ROSA" IDCargo="0000000002" codescala="0000000004" Sexo="M" DataAdm="02/01/2006"
   DataNasc="28/06/1979" Pis="0" Matricula="0" PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor=""
   />
<ROW IDFuncionario="0000000015" IDFuncao="0000000002" IDSetor="0000000007"
   Identidade="4056110192" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000015" CodFuncionario="00000000000000000015" Nome="MARCO AURELIO
   BRAUN" IDCargo="0000000001" codescala="0000000005" Sexo="M" Ctps="0" Serie="" EstadoCtps="RS"
   DataAdm="01/10/2003" DataNasc="09/06/1975" Pis="12385823812" Matricula="15"
   PermitirSaidaInt="Sim" Bloquear="Não" DataAfastamento="" NomeSetor="" />
<ROW IDFuncionario="0000009027" IDFuncao="0000000006" IDSetor="0000000001"
   Identidade="4086082478" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000035" CodFuncionario="00000000000000000035" Nome="Pablo Alexandre
   Sesterheim" IDCargo="0000000004" codescala="0000000004" Sexo="M" Ctps="1072432" Serie="001"
   EstadoCtps="0" DataAdm="06/07/2009" DataNasc="10/01/1984" Pis="000" Matricula="0035"
   PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor="" />
<ROW IDFuncionario="0000009003" IDFuncao="0000000006" IDSetor="0000000001"
   Identidade="1082816628" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000017" CodFuncionario="00000000000000000017" Nome="PABLO LEONARDO
   BRAUN" IDCargo="0000000004" codescala="0000000005" Sexo="M" Ctps="8600091" Serie="001-0"
   EstadoCtps="RS" DataAdm="11/07/2005" DataNasc="15/01/1986" Pis="20102657011" Matricula="17"
   PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor="" />
<ROW IDFuncionario="0000009006" IDFuncao="0000000000" IDSetor="0000000000"
   Identidade="5097328594" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000027" CodFuncionario="00000000000000000027" Nome="RICIERY A.
   FOSCARINI SPARDELOTTI" IDCargo="0000000000" codescala="0000000004" Sexo="M" Ctps="9524016"
   Serie="001-0" EstadoCtps="RS" DataAdm="30/01/2006" DataNasc="05/06/1991" Pis="1300253670-8"
   Matricula="9002" PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor="" />
<ROW IDFuncionario="0000009021" IDFuncao="0000000005" IDSetor="0000000008"
   Identidade="7087876129" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000030" CodFuncionario="00000000000000009010" Nome="SUSANA VENITES
   DE SOUZA" IDCargo="0000000003" codescala="0000000004" Sexo="F" DataAdm="01/08/2008"
   DataNasc="23/02/1984" Pis="1" Matricula="9010" PermitirSaidaInt="Sim" Bloquear="Não"
   NomeSetor="" />
<ROW IDFuncionario="0000009020" IDFuncao="0000000006" IDSetor="0000000001"
   Identidade="3057100749" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000028" CodFuncionario="00000000000000000028" Nome="VAGNER SILVEIRA"
   IDCargo="0000000004" codescala="0000000004" Sexo="M" DataAdm="11/02/2008"
   DataNasc="17/12/1974" Pis="1212" Matricula="1" PermitirSaidaInt="Sim" Bloquear="Não"
   NomeSetor="" />
<ROW IDFuncionario="0000009013" IDFuncao="0000000000" IDSetor="0000000000"
   CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas" Inativo="Não"
   Cracha="0000000023" CodFuncionario="00000000000000000023" Nome="VANDERLISE RAQUEL LAUFER"
   IDCargo="0000000000" codescala="0000000004" Sexo="F" DataAdm="16/04/2007"
   DataNasc="01/01/1983" Pis="343423423423" Matricula="09004" PermitirSaidaInt="Sim"
   Bloquear="Não" NomeSetor="" />
<ROW IDFuncionario="0000009026" IDFuncao="0000000001" IDSetor="0000000001"
   Identidade="3085688871" CodEmpresa="0000000001" Empresa="Sanvitron Soluções Eletrônicas"
   Inativo="Não" Cracha="0000000033" CodFuncionario="00000000000000000033" Nome="William Henrique
   Boff" IDCargo="0000000002" codescala="0000000004" Sexo="M" Ctps="1692986" Serie="002-0"
   EstadoCtps="RS" DataAdm="15/06/2009" DataNasc="27/10/1989" Pis="1303393371-1" Matricula="33"
   PermitirSaidaInt="Sim" Bloquear="Não" NomeSetor="" />
  </ROWDATA>
  </DATAPACKET>
```

ANEXO B – Layout definido pelo sistema de ponto eletrônico

Sanvitron - Layout Para Importação de Funcionários

Pos.Ini	Pos.Fin.	Tamanho	Conteúdo	Formato	Valor Fixo	Observação
1	3	3	Código da Empresa	Numérico		
4	9	6	Código Funcionário	Numérico		
10	49	40	Nome	Alfanumérico		
50	56	7	CTPS	Numérico		
57	62	6	Série CTPS	Alfanumérico		
63	70	8	Data Admissão	Data		ddmmaaaa
71	71	1	Tipo Salário	Numérico		1=Horista 2=Mensalista
72	72	1	Estado Atual	Numérico		0=Inativo 1=Ativo
73	80	8	Data da rescisão	Data		ddmmaaaa
81	83	3	Código da Escala de Trabalho	Numérico		
84	91	8	Data da Alteração da Escala	Data		ddmmaaaa. Data em que será aplicada a nova escala de trabalho em caso de troca
92	101	10	RG	Numérico		
102	113	12	PIS	Numérico		
114	121	8	Data Nascimento	Data		ddmmaaaa
122	122	1	Sexo	Alfanumérico		M ou F
123	128	6	Código do Crachá	Numérico		
129	134	6	Código Vínculo	Numérico		
135	164	30	Descrição Vínculo	Alfanumérico		
165	170	6	Código do Turno	Numérico		
171	200	30	Descrição do Turno	Alfanumérico		Exemplos: Diurno Noturno Manhã
201	206	6	Código da Categoria	Numérico		
207	236	30	Descrição da Categoria	Alfanumérico		
237	246	10	Matrícula	Alfanumérico		A matrícula é um código alternativo de integração que pode ser igual ao código
247	252	6	Código do Centro de Custo	Numérico		
253	292	40	Descrição do Centro de Custo	Alfanumérico		
293	298	6	Código do Setor	Numérico		
299	338	40	Descrição do Setor	Alfanumérico		
339	344	6	Código da Função	Numérico		
345	384	40	Descrição da Função	Alfanumérico		
385	390	6	CBO da função	Numérico		
391	396	6	Código do Cargo	Numérico		
397	436	40	Descrição do Cargo	Alfanumérico		
437	442	6	CBO do Cargo	Numérico		
443	448	6	Código do Departamento	Numérico		
449	488	40	Descrição do Departamento	Alfanumérico		
489	489	1	Permite Saída Intermediária	Numérico		0=Não 1=Sim
490	490	1	Adicionar no Arquivo da Folha	Numérico		0=Não 1=Sim. Campo que permite fazer com que o funcionário seja ou não incluído no arquivo do fechamento mensal
491	491	1	Colaborador Bloqueado	Numérico		0=Não 1=Sim. O bloqueio significa que o funcionário não poderá passar o cartão no(s) relógio(s)
492	501	10	Grupo de Terminais	Numérico		Cadastro feito no ponto que permite que os funcionários utilizem somente determinados relógios
502	509	8	Data Início de Férias	Data		ddmmaaaa. Caso o funcionário está sendo só para cadastro ou alteração, deixar Inicio e Final de férias em branco
510	517	8	Data final de férias	Data		ddmmaaaa. Caso houver alteração do código cadastral do funcionário, informar nesse campo o código anterior
518	523	6	Código Anterior	Numérico		
524	524	1	Excluir Funcionário	Numérico		0=Não; 1=Excluir somente o funcionário; 2=Excluir o funcionário e todos os dados relacionados a ele (registros, férias...)
525	532	8	Valor Hora do Salário	Numérico		Valor correspondente à hora trabalhada do colaborador. Formato numérico com 2 casas decimais separadas por vírgula - ex: 00001,99
533	533	1	Turno da Troca	Numérico		Quando ocorrer uma troca de função, setor ou cargo, deve-se informar o turno em que a troca ocorreu
534	541	8	Data da Troca	Data		ddmmaaaa. Quando ocorrer uma troca de função, setor ou cargo, deve-se informar a data em que a troca ocorreu
542	545	4	Hora da Troca	Hora		hhmm. Quando informar a Data da Troca, informar também a hora em que ela ocorreu

APÊNDICES

APÊNDICE A – Script executado durante o teste do protótipo

```
var
  S : String;

begin
  TextoNovo := '';

  Tabela.FileName := 'c:\funcionario.xml';
  Tabela.Open;
  Tabela.First;
  while not Tabela.eof do
    begin
      S := FormatFloat('000', Tabela.Fields[4].AsInteger);     //Empresa
      S := S + RightStr(Tabela.Fields[8].AsString, 6);         //Cod.Funcionario
      S := S + AlEsq(Tabela.Fields[9].AsString, 40);           //Nome Funcionario
      S := S + Zeros_Esquerda(Tabela.Fields[13].AsString, 7);  //CTPS
      S := S + AlEsq(Tabela.Fields[14].AsString, 6);           //Serie
      S := S + Copy(Tabela.Fields[16].AsString, 1, 2) +        //Data de Admissao
                Copy(Tabela.Fields[16].AsString, 4, 2) +
                Copy(Tabela.Fields[16].AsString, 7, 4);
      S := S + '1';                                            //Tipo de salario: 1 - horista  (informacao nao consta no XML)
      S := S + '1';                                            //Ativo  (informacao nao consta no XML)
      S := S + AlEsq('', 8);                                   //data rescisao - em branco  (informacao nao consta no XML)
      S := S + RightStr(Tabela.Fields[11].AsString, 3);        //Cod.Escala
      S := S + AlEsq('', 8);                                   //data alteracao escala - em branco  (informacao nao consta no XML)

      S := S + Zeros_Esquerda(Tabela.Fields[3].AsString, 10);  //RG
      S := S + Zeros_Esquerda(Tabela.Fields[18].AsString, 12); //PIS
      S := S + Copy(Tabela.Fields[17].AsString, 1, 2) +        //Data de Admissao
                Copy(Tabela.Fields[17].AsString, 4, 2) +
                Copy(Tabela.Fields[17].AsString, 7, 4);
      S := S + Tabela.Fields[12].AsString;                     //Sexo
      S := S + RightStr(Tabela.Fields[7].AsString, 6);         //Cracha
      S := S + '000001';                                       //Vinculo: 1 - empregado  (informacao nao consta no XML)
      S := S + AlEsq('', 30);                                  //descricao do vinculo  (informacao nao consta no XML)
      S := S + '000000';                                       //codigo do turno  (informacao nao consta no XML)
      S := S + AlEsq('', 30);                                  //descricao do turno  (informacao nao consta no XML)
      S := S + '000000';                                       //codigo da categoria  (informacao nao consta no XML)
      S := S + AlEsq('', 30);                                  //descricao da categoria  (informacao nao consta no XML)
      S := S + AlEsq(Tabela.Fields[19].AsString, 10);          //matricula
      S := S + '000000';                                       //codigo do centro de custo  (informacao nao consta no XML)
      S := S + AlEsq('', 40);                                  //descricao do centro de custo  (informacao nao consta no XML)
      S := S + '000000';                                       //codigo do setor  (informacao nao consta no XML)
      S := S + AlEsq('', 40);                                  //descricao do setor  (informacao nao consta no XML)
      S := S + '000000';                                       //codigo da funcao  (informacao nao consta no XML)
      S := S + AlEsq('', 40);                                  //descricao da funcao  (informacao nao consta no XML)
```

```
      S := S + '000000';                                //CBO da funcao  (informacao nao consta no XML)
      S := S + '000000';                                //codigo do cargo  (informacao nao consta no XML)
      S := S + AlEsq('', 40);                           //descricao do cargo  (informacao nao consta no XML)
      S := S + '000000';                                //CBO do cargo  (informacao nao consta no XML)
      S := S + '000000';                                //codigo do departamento  (informacao nao consta no XML)
      S := S + AlEsq('', 40);                           //descricao do departamento  (informacao nao consta no XML)

      if Tabela.Fields[20].AsString = 'Não' then        //Permite realizar saidas intermediarias
        S := S + '0'
      else
        S := S + '1';

      S := S + '1';                                     //Adiciona arquivo folha: 1 - Sim

      if Tabela.Fields[21].AsString = 'Não' then        //funcionario bloqueado
        S := S + '0'
      else
        S := S + '1';

      S := S + Zeros_Esquerda('1', 10);                 //Grupo de terminais - grupo 1
      S := S + AlEsq('', 8);                            //data de início das férias  (informacao nao consta no XML)
      S := S + AlEsq('', 8);                            //data de fim das férias  (informacao nao consta no XML)
      S := S + Zeros_Esquerda('0', 6);                  //Codigo anterior
      S := S + '0';                                     //excluir funcionario: 0 - Não
      S := S + Zeros_Esquerda('0,00', 8);               //salário - valor hora  (informacao nao consta no XML)
      S := S + '0';                                     //turno da troca (informacao nao consta no XML)
      S := S + AlEsq('', 8);                            //data da troca  (informacao nao consta no XML)
      S := S + AlEsq('', 4);                            //hora da troca  (informacao nao consta no XML)


      TextoNovo := TextoNovo + S + #13#10;

      Tabela.Next;
  end;
end.
```

APÊNDICE B – Arquivo texto gerado pela transformação no ESB

```
001000016ALEXANDRE PEREIRA DA ROSA.              004036900060 2701200411       004       907433662101271575767202121978M000016000001
001000003ALINE MAGNI TRESSOLDI  DE OLIVEIRA     006373800057 0301200011       004       206649813601270963767917081978F000003000001
001000029CARLA FERNANDA DE OLIVEIRA JOHANN       6916355001   0107200811       004       810984658701298680952003041986M000029000001
001000034Cleuza Marina Magni Tressoldi           00000000     1302200611       002       000000000000000000000019091956F000034000001
001000004FREDERICO TIMMEN                        004547900058 0203200111       004       107534959501272035569220081981M000004000001
001000001GUIBSON HAAS DOS SANTOS                 0000000      0101199711       004       204343478211383588-76124121975M000001000001
001000020MARCIO CARDOSO                          0000000      0108200511       100       205760576400000000000021091974M000020000001
001000019MARCIO MACIEL DA ROSA                   0000000      0201200611       004       607016791800000000000028061979M000019000001
001000015MARCO AURELIO BRAUN                     0000000      0110200311       005       405611019201238582381209061975M000015000001
001000035Pablo Alexandre Sesterheim              1072432001   0607200911       004       408608247800000000000010011984M000035000001
001000017PABLO LEONARDO BRAUN                    8600091001-0 1107200511       005       108281662802010265701115011986M000017000001
001000027RICIERY A. FOSCARINI SPARDELOTTI        9524016001-0 3001200611       004       50973285941300253670-805061991M000027000001
001009010SUSANA VENITES DE SOUZA                 0000000      0108200811       004       708787612900000000000123021984F000030000001
001000028VAGNER SILVEIRA                         0000000      1102200811       004       305710074900000000121217121974M000028000001
001000023VANDERLISE RAQUEL LAUFER                0000000      1604200711       004       000000000034342342342301011983F000023000001
001000033William Henrique Boff                   1692986002-0 1506200911       004       308568887113033933 71-127101989M000033000001
```

APÊNDICE C – Código fonte da interface de configuração

```
unit UConfigProt;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, StdCtrls, ExtCtrls, DBCtrls, DB, DBClient, Mask,
  ComCtrls;

type
  TfrmConfigProt = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Splitter1: TSplitter;
    Label1: TLabel;
    Panel3: TPanel;
    Label2: TLabel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    DBRadioGroup1: TDBRadioGroup;
    dtsConector: TClientDataSet;
    dsoConector: TDataSource;
    dtsConectoridConector: TIntegerField;
    dtsConectorDescricao: TStringField;
    dtsConectorTipo: TStringField;
    dtsConectorDirecao: TStringField;
    dtsConectorArquivoOrigem: TBlobField;
    dtsConectorArquivoDestino: TBlobField;
    dtsConectorNroInsistencias: TIntegerField;
    dtsConectorTempoEntreInsistencias: TIntegerField;
    dtsConectorScriptModificacao: TBlobField;
    dtsTemp: TClientDataSet;
    Label3: TLabel;
    DBEdit1: TDBEdit;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    DBEdit2: TDBEdit;
    Label7: TLabel;
    DBEdit3: TDBEdit;
    Label8: TLabel;
    DBEdit4: TDBMemo;
    DBEdit5: TDBMemo;
    DBMemo1: TDBMemo;
    UpDown1: TUpDown;
```

```
UpDown2: TUpDown;
GroupBox1: TGroupBox;
PageControl1: TPageControl;
tbsFile: TTabSheet;
tbsFTP: TTabSheet;
tbsHTTP: TTabSheet;
tbsSMTP: TTabSheet;
tbsSQL: TTabSheet;
tbsPOP3: TTabSheet;
dtsFile: TClientDataSet;
dtsFTP: TClientDataSet;
dtsHTTP: TClientDataSet;
dtsSMTP: TClientDataSet;
dtsSQL: TClientDataSet;
dsoFile: TDataSource;
dsoFTP: TDataSource;
dsoHTTP: TDataSource;
dsoSMTP: TDataSource;
dsoSQL: TDataSource;
dtsFileidFile: TIntegerField;
dtsFileidConector: TIntegerField;
dtsFileModoCriacao: TStringField;
dtsFileAcaoOrigem: TStringField;
dtsFileMovePara: TBlobField;
dtsFTPidFTP: TIntegerField;
dtsFTPidConector: TIntegerField;
dtsFTPAcaoOrigem: TStringField;
dtsFTPMovePara: TBlobField;
dtsFTPModoConexao: TStringField;
dtsFTPRepresentacao: TStringField;
dtsHTTPidHTTP: TIntegerField;
dtsHTTPidConector: TIntegerField;
dtsHTTPAcaoOrigem: TStringField;
dtsHTTPMovePara: TBlobField;
dtsSMTPidSMTP: TIntegerField;
dtsSMTPidConector: TIntegerField;
dtsSMTPPara: TStringField;
dtsSMTPCopiaPara: TStringField;
dtsSMTPAssunto: TStringField;
dtsSMTPMensagem: TBlobField;
dtsSQLidSQL: TIntegerField;
dtsSQLConector: TIntegerField;
dtsSQLStringConexao: TBlobField;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
DBEdit8: TDBEdit;
DBComboBox1: TDBComboBox;
DBComboBox2: TDBComboBox;
Label12: TLabel;
```

```
    DBComboBox3: TDBComboBox;
    Label13: TLabel;
    DBComboBox4: TDBComboBox;
    Label14: TLabel;
    Label15: TLabel;
    DBComboBox5: TDBComboBox;
    Panel7: TPanel;
    DBMemo2: TDBMemo;
    DBGrid1: TDBGrid;
    Label16: TLabel;
    DBMemo3: TDBMemo;
    DBMemo4: TDBMemo;
    Label17: TLabel;
    mdtSenha: TEdit;
    Label18: TLabel;
    dtsAutenticacao: TClientDataSet;
    dsoAutenticacao: TDataSource;
    dtsAutenticacaoidConector: TIntegerField;
    dtsAutenticacaoHost: TStringField;
    dtsAutenticacaoPorta: TIntegerField;
    dtsAutenticacaoTimeOut: TIntegerField;
    dtsAutenticacaoUsuario: TStringField;
    dtsAutenticacaoSenha: TStringField;
    function AbreTabela(Tabela : TClientDataSet) : Boolean;
    function SalvaTabela(Tabela : TClientDataSet) : Boolean;
    procedure FormCreate(Sender: TObject);
    procedure Panel4Click(Sender: TObject);
    procedure dtsConectorAfterScroll(DataSet: TDataSet);
    procedure Panel5Click(Sender: TObject);
    procedure Panel6Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmConfigProt: TfrmConfigProt;

implementation

{$R *.dfm}

function TfrmConfigProt.AbreTabela(Tabela: TClientDataSet): Boolean;
var
  Arquivo : String;

  function CorrigeXML : Boolean;
  var
    i : Integer;
```

```
  begin
    Result := False;

    dtsTemp.Close;
    dtsTemp.FileName := Arquivo;
    dtsTemp.Open;

    Tabela.Close;
    Tabela.FieldDefs.Clear;
    Tabela.CreateDataSet;

    dtsTemp.First;
    while not dtsTemp.Eof do
      begin
        Tabela.Append;
        for i := 0 to dtsTemp.FieldCount-1 do
          Tabela.FieldByName(dtsTemp.Fields[i].FieldName).AsString := dtsTemp.Fields[i].AsString;
        Tabela.Post;

        dtsTemp.Next;
      end;

    Result := SalvaTabela(Tabela);
  end;

begin
  if Tabela.State = dsInsert then
    SalvaTabela(Tabela);

  Tabela.Close;

  Arquivo := Tabela.Name;

  System.Delete(Arquivo, 1, 3);

  Arquivo := ExtractFilePath(Application.ExeName) + Arquivo + '.xml';

  Tabela.FileName := Arquivo;

  try
    if FileExists(Arquivo) then
      begin
        try
          Tabela.Open;
          Result := True;
        except
          on E : Exception do
            begin
              if (Pos('Field', E.Message) > 0) and (Pos('not found', E.Message) > 0) then
                begin
```

```
                if CorrigeXML then
                  begin
                    Tabela.Open;
                    Result := True;
                  end
                else
                  Result := False;
              end;
            end;
        end;
      end
    else
      Result := False;
  except
    Result := False;
  end;

  if not Result then
    Tabela.CreateDataSet;
end;

function TfrmConfigProt.SalvaTabela(Tabela: TClientDataSet): Boolean;
var
  Arquivo : String;
begin
  if Tabela.State in [dsInsert, dsEdit] then
    Tabela.Post;

  Arquivo := Tabela.Name;

  System.Delete(Arquivo, 1, 3);

  Arquivo := ExtractFilePath(Application.ExeName) + Arquivo + '.xml';

  try
    Tabela.MergeChangeLog;
    Tabela.SaveToFile(Arquivo, dfXMLUTF8);
    Result := True;
  except
    on E : Exception do
      Result := False;
  end;
end;

procedure TfrmConfigProt.FormCreate(Sender: TObject);
begin
  AbreTabela(dtsConector);

  AbreTabela(dtsFTP);
```

```
  AbreTabela(dtsAutenticacao);
end;

procedure TfrmConfigProt.Panel4Click(Sender: TObject);
begin
  if tbsFile.Showing then
    dtsConectorTipo.AsString := 'FILE'
  else
  if tbsFTP.Showing then
    dtsConectorTipo.AsString := 'FTP'
  else
  if tbsHTTP.Showing then
    dtsConectorTipo.AsString := 'HTTP'
  else
  if tbsSMTP.Showing then
    dtsConectorTipo.AsString := 'SMTP'
  else
  if tbsSQL.Showing then
    dtsConectorTipo.AsString := 'SQL'
  else
  if tbsPOP3.Showing then
    dtsConectorTipo.AsString := 'POP3';

  dtsConectoridConector.AsInteger := dtsConector.RecNo;

  if dtsConectorTipo.AsString = 'FTP' then
    begin
      dtsFTP.Edit;

      dtsFTPidConector.AsString := dtsConectoridConector.AsString;

      SalvaTabela(dtsFTP);


      dtsAutenticacao.Edit;
      dtsAutenticacaoidConector.AsString := dtsConectoridConector.AsString;
      dtsAutenticacaoSenha.AsString := mdtSenha.Text;
      SalvaTabela(dtsAutenticacao);
    end;

  SalvaTabela(dtsConector);

  AbreTabela(dtsConector);
end;

procedure TfrmConfigProt.dtsConectorAfterScroll(DataSet: TDataSet);
begin
  if dtsConectorTipo.AsString <> '' then
    PageControl1.TabIndex := TTabSheet(frmConfigProt.FindComponent('tbs' + dtsConectorTipo.AsString)).TabIndex;
```

```
  if tbsFTP.Showing then
    begin
      AbreTabela(dtsAutenticacao);

      if (dtsAutenticacao.RecordCount = 0) then
        dtsAutenticacao.Append
      else
        dtsAutenticacao.Edit;

      mdtSenha.Text := dtsAutenticacaoSenha.AsString;
    end;
end;

procedure TfrmConfigProt.Panel5Click(Sender: TObject);
begin
  dtsConector.Append;
end;

procedure TfrmConfigProt.Panel6Click(Sender: TObject);
begin
  dtsConector.Edit;
end;

end.
```

APÊNDICE D – Código fonte do serviço

```
unit UESB;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, SvcMgr, Dialogs,
  DB, DBClient, IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,
  IdFTP, fs_ipascal, fs_idbctrlsrtti, fs_ibdertti, fs_iformsrtti,
  fs_iinterpreter, ExtCtrls, UConector, Forms;

type
  TSrvESB = class(TService)
    fsScript1: TfsScript;
    fsFormsRTTI1: TfsFormsRTTI;
    fsBDERTTI1: TfsBDERTTI;
    fsDBCtrlsRTTI1: TfsDBCtrlsRTTI;
    fsPascal1: TfsPascal;
    fsBDERTTI2: TfsBDERTTI;
    IdFTP1: TIdFTP;
    Tabela: TClientDataSet;
    TabelaIDFuncionario: TStringField;
    TabelaIDFuncao: TStringField;
    TabelaIDSetor: TStringField;
    TabelaIdentidade: TStringField;
    TabelaCodEmpresa: TStringField;
    TabelaEmpresa: TStringField;
    TabelaInativo: TStringField;
    TabelaCracha: TStringField;
    TabelaCodFuncionario: TStringField;
    TabelaNome: TStringField;
    TabelaIDCargo: TStringField;
    Tabelacodescala: TStringField;
    TabelaSexo: TStringField;
    TabelaCtps: TStringField;
    TabelaSerie: TStringField;
    TabelaEstadoCtps: TStringField;
    TabelaDataAdm: TStringField;
    TabelaDataNasc: TStringField;
    TabelaPis: TStringField;
    TabelaMatricula: TStringField;
    TabelaPermitirSaidaInt: TStringField;
    TabelaBloquear: TStringField;
    TabelaDataAfastamento: TStringField;
    TabelaNomeSetor: TStringField;
    dtsAutenticacao: TClientDataSet;
    dtsAutenticacaoidConector: TIntegerField;
    dtsAutenticacaoHost: TStringField;
```

```
    dtsAutenticacaoPorta: TIntegerField;
    dtsAutenticacaoTimeOut: TIntegerField;
    dtsAutenticacaoUsuario: TStringField;
    dtsAutenticacaoSenha: TStringField;
    dtsFTP: TClientDataSet;
    dtsFTPidFTP: TIntegerField;
    dtsFTPidConector: TIntegerField;
    dtsFTPAcaoOrigem: TStringField;
    dtsFTPMovePara: TBlobField;
    dtsFTPModoConexao: TStringField;
    dtsFTPRepresentacao: TStringField;
    dtsConector: TClientDataSet;
    dtsConectoridConector: TIntegerField;
    dtsConectorDescricao: TStringField;
    dtsConectorTipo: TStringField;
    dtsConectorDirecao: TStringField;
    dtsConectorArquivoOrigem: TBlobField;
    dtsConectorArquivoDestino: TBlobField;
    dtsConectorNroInsistencias: TIntegerField;
    dtsConectorTempoEntreInsistencias: TIntegerField;
    dtsConectorScriptModificacao: TBlobField;
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    function GetServiceController: TServiceController; override;
    { Public declarations }
  end;

var
  SrvESB: TSrvESB;
  Conector : TConector;


implementation

{$R *.DFM}

procedure ServiceController(CtrlCode: DWord); stdcall;
begin
  SrvESB.Controller(CtrlCode);
end;

function TSrvESB.GetServiceController: TServiceController;
begin
  Result := ServiceController;
end;

procedure TSrvESB.Timer1Timer(Sender: TObject);
```

```
var
  dtsFuncionario : TClientDataSet;
  F : TextFile;
  DirTemp : String;
  Tentativa : Integer;

begin
  if not dtsConector.Active then
    begin
      dtsConector.FileName := ExtractFilePath(Application.ExeName) + 'Conector.xml';
      dtsConector.Open;
    end;

  dtsConector.First;
  while not dtsConector.Eof do
    begin
      Conector := TConector.Create;
      try
        Conector.Criar(dtsConectorTipo.AsString, dtsConectorDirecao.AsString, dtsConectorArquivoOrigem.AsString,
dtsConectorArquivoDestino.AsString);

        Conector.scriptModificacao := dtsConectorScriptModificacao.AsString;
        Conector.nroInsistencias := dtsConectorNroInsistencias.AsInteger;
        Conector.tempoEntreInsistencias := dtsConectorTempoEntreInsistencias.AsInteger;

        if (dtsConectorDirecao.AsString = 'diEnvia') and (dtsConectorTipo.AsString = 'FTP') then
          begin
            dtsFTP.Close;

            dtsFTP.FileName := ExtractFilePath(Application.ExeName) + 'FTP.xml';
            dtsFTP.Filtered := False;
            dtsFTP.Filter := 'IDConector = ' + dtsConectoridConector.AsString;
            dtsFTP.Filtered := True;
            dtsFTP.Open;

            dtsAutenticacao.Close;
            dtsAutenticacao.FileName := ExtractFilePath(Application.ExeName) + 'Autenticacao.xml';
            dtsAutenticacao.Filtered := False;
            dtsAutenticacao.Filter := 'IDConector = ' + dtsConectoridConector.AsString;
            dtsAutenticacao.Filtered := True;
            dtsAutenticacao.Open;

            if Conector.ConfigurarFTP(dtsFTPAcaoOrigem.AsString, dtsFTPMovePara.AsString, dtsFTPModoConexao.AsString,
dtsFTPRepresentacao.AsString) and
               Conector.ConfigurarAutenticacao(dtsAutenticacaoHost.AsString, dtsAutenticacaoUsuario.AsString, dtsAutenticacaoSenha.AsString, 21,
15000) then
              begin
                if dtsConectorScriptModificacao.AsString <> '' then
                  begin
                    //busca texto do arquivo de origem
```

```
AssignFile(F, dtsConectorArquivoOrigem.AsString);
Reset(F);
Read(F, Conector.TextoOrigem);
CloseFile(F);

Tentativa := 0;
while Tentativa <= Conector.nroInsistencias do
  begin
    Inc(Tentativa);

    if Conector.ExecutaScript then
      begin
        DirTemp := ExtractFilePath(Application.ExeName) + 'Temp\';

        if not DirectoryExists(DirTemp) then
          ForceDirectories(DirTemp);

        //salva texto do arquivo de destino
        AssignFile(F, DirTemp + dtsConectorArquivoDestino.AsString);
        Rewrite(F);
        Write(F, Conector.TextoNovo);
        CloseFile(F);

        Conector.ArquivoTransformado := DirTemp + dtsConectorArquivoDestino.AsString;

        Break;

        Sleep(Conector.tempoEntreInsistencias * 1000);
      end;
  end;

if (Tentativa <= Conector.nroInsistencias) then
  begin
    Tentativa := 0;
    while Tentativa <= Conector.nroInsistencias do
      begin
        Inc(Tentativa);

        if Conector.Conectar then
          Break;

        Sleep(Conector.tempoEntreInsistencias * 1000);
      end;

    if (Tentativa <= Conector.nroInsistencias) then
      begin
        Tentativa := 0;
        while Tentativa <= Conector.nroInsistencias do
          begin
            Inc(Tentativa);
```

```
                        if Conector.Enviar then
                          Break;

                        Sleep(Conector.tempoEntreInsistencias * 1000);
                      end;
                  end;

                if (Tentativa <= Conector.nroInsistencias) then
                  begin
                    Tentativa := 0;
                    while Tentativa <= Conector.nroInsistencias do
                      begin
                        Inc(Tentativa);

                        if Conector.ExecutaAcaoArquivo(Conector.FTP.AcaoOrigem) then
                          Break;

                        Sleep(Conector.tempoEntreInsistencias * 1000);
                      end;
                  end;

                Conector.Desconectar;
              end;
            end;
          end;
        end;
      finally
        Conector.Free;
      end;
    dtsConector.Next;
  end;
end;

end.
```

APÊNDICE E – Código fonte das Classes Conector, FTP e Autenticacao

```
unit UAutenticacao;

interface

type
  TAutenticacao = class(TObject)
  private
    Fporta: Integer;
    FtimeOut: Integer;
    Fusuario: String;
    Fhost: String;
    Fsenha: String;
    procedure Sethost(const Value: String);
    procedure Setporta(const Value: Integer);
    procedure Setsenha(const Value: String);
    procedure SettimeOut(const Value: Integer);
    procedure Setusuario(const Value: String);
  public
  published
    property host : String read Fhost write Sethost;
    property porta : Integer read Fporta write Setporta;
    property timeOut : Integer read FtimeOut write SettimeOut;
    property usuario : String read Fusuario write Setusuario;
    property senha : String read Fsenha write Setsenha;
  end;


implementation

uses UESB;

{ TAutenticacao }


{ TAutenticacao }

procedure TAutenticacao.Sethost(const Value: String);
begin
  Fhost := Value;
end;

procedure TAutenticacao.Setporta(const Value: Integer);
begin
  Fporta := Value;
end;

procedure TAutenticacao.Setsenha(const Value: String);
begin
  Fsenha := Value;
end;

procedure TAutenticacao.SettimeOut(const Value: Integer);
begin
  FtimeOut := Value;
end;

procedure TAutenticacao.Setusuario(const Value: String);
begin
  Fusuario := Value;
end;

end.
```

```
unit UConector;

interface

uses UFTP, SysUtils, fs_iInterpreter, StdCtrls, fs_iBDERTTI, fs_iFormsRTTI,
  fs_iDBCtrlsRTTI, fs_ipascal, DB, DBClient, UAutenticacao, Windows;
```

```
type
  TConector = class(TObject)
  private
    FnroInsistencias: Integer;
    FtempoEntreInsistencias: Integer;
    FidConector: Integer;
    FarquivoDestino: String;
    FarquivoOrigem: String;
    Fdirecao: String;
    FscriptModificacao: String;
    Ftipo: String;
    FFTP: TFTP;
    FAutenticacao: TAutenticacao;
    FarquivoTransformado: String;
    procedure SetarquivoDestino(const Value: String);
    procedure SetarquivoOrigem(const Value: String);
    procedure Setdirecao(const Value: String);
    procedure SetidConector(const Value: Integer);
    procedure SetnroInsistencias(const Value: Integer);
    procedure SetscriptModificacao(const Value: String);
    procedure SettempoEntreInsistencias(const Value: Integer);
    procedure Settipo(const Value: String);
    procedure SetFTP(const Value: TFTP);
    function CallMethod(Instance: TObject; ClassType: TClass;const MethodName: String; var
Params: Variant): Variant;
    procedure SetAutenticacao(const Value: TAutenticacao);
    procedure SetarquivoTransformado(const Value: String);
  public
    TextoOrigem : String;
    TextoNovo   : String;

    function RightStr(const AText: String; const ACount: Integer): AnsiString;
    function LeftStr(const AText: String; const ACount: Integer): String;
    function AlDir(S:String; Largura:Integer; LimitaLargura : Boolean=True):String;
    function AlEsq(S:String; Largura:Integer; LimitaLargura : Boolean=True):String;
    function Zeros_Esquerda(Texto : String; Tamanho : Integer) : String;
  published
    property idConector : Integer read FidConector write SetidConector;
    property tipo : String read Ftipo write Settipo;
    property direcao : String read Fdirecao write Setdirecao;
    property arquivoOrigem: String read FarquivoOrigem write SetarquivoOrigem;
    property arquivoDestino: String read FarquivoDestino write SetarquivoDestino;
    property arquivoTransformado : String read FarquivoTransformado write
SetarquivoTransformado;
    property nroInsistencias: Integer read FnroInsistencias write SetnroInsistencias;
    property tempoEntreInsistencias : Integer read FtempoEntreInsistencias write
SettempoEntreInsistencias;
    property scriptModificacao : String read FscriptModificacao write SetscriptModificacao;
    property FTP : TFTP read FFTP write SetFTP;
    property Autenticacao : TAutenticacao read FAutenticacao write SetAutenticacao;

    procedure Criar(sTipo, sDirecao, sArquivoOrigem, sArquivoDestino : String);
    function ConfigurarFTP(sAcaoOrigem, sMovePara, sModoConexao, sRepresentacao : String) :
Boolean;
    function ConfigurarAutenticacao(sHost, sUsuario, sSenha : String; iPorta, iTimeOut :
Integer) : Boolean;
    function ExecutaScript : Boolean;
    function Conectar : Boolean;
    function Enviar : Boolean;
    function ExecutaAcaoArquivo(AcaoOrigem : String) : Boolean;
    function Desconectar : Boolean;
  end;

implementation

uses IdTCPConnection, UESB;


{ TConector }

function TConector.CallMethod(Instance: TObject; ClassType: TClass;
  const MethodName: String; var Params: Variant): Variant;
begin
    { dispatch the method call }
  if MethodName = 'EXTRACTFILEPATH' then //ExtractFilePath
      Result := ExtractFilePath(Params[0])
  else
```

```
  if MethodName = 'RIGHTSTR' then
    Result := RightStr(Params[0], Params[1])
  else
  if MethodName = 'LEFTSTR' then
    Result := LeftStr(Params[0], Params[1])
  else
  if MethodName = 'ALDIR' then
    Result := AlDir(Params[0], Params[1], Params[2])
  else
  if MethodName = 'ALESQ' then
    Result := AlEsq(Params[0], Params[1], Params[2])
  else
  if MethodName = 'ZEROS_ESQUERDA' then
    Result := Zeros_Esquerda(Params[0], Params[1]);
end;

function TConector.ConfigurarFTP(sAcaoOrigem, sMovePara, sModoConexao,
  sRepresentacao: String): Boolean;
begin
  Result := True;

  try
    FTP.AcaoOrigem := sAcaoOrigem;
    FTP.MovePara := sMovePara;
    FTP.ModoConexao := sModoConexao;
    FTP.Representacao := sRepresentacao;

    if (FTP.MovePara <> '') and (RightStr(FTP.MovePara, 1) <> '\') then
      FTP.MovePara := FTP.MovePara + '\';
  except
    Result := False;
  end;
end;

procedure TConector.Criar(sTipo, sDirecao, sArquivoOrigem,
  sArquivoDestino: String);
begin
  tipo := sTipo;
  direcao := sDirecao;
  arquivoOrigem := sArquivoOrigem;
  arquivoDestino := sArquivoDestino;

  if tipo = 'FTP' then
    begin
      FTP := TFTP.Create;
      Autenticacao := TAutenticacao.Create;
    end;
end;

function TConector.ExecutaScript: Boolean;
begin
  Result := False;

  try
    { clear all items }
    SrvESB.fsScript1.Clear;
    { script text }

    SrvESB.fsScript1.Lines.Text := Conector.scriptModificacao;

    { frGlobalUnit contains standard types and functions }
    SrvESB.fsScript1.Parent := fsGlobalUnit;

    SrvESB.fsScript1.AddClass(TDataSet, TDataSet.ClassParent.ClassName);
    SrvESB.fsScript1.AddClass(TCustomClientDataSet,
TCustomClientDataSet.ClassParent.ClassName);
    SrvESB.fsScript1.AddClass(TClientDataSet, TClientDataSet.ClassParent.ClassName);


    SrvESB.fsScript1.AddForm(SrvESB);
    SrvESB.fsScript1.AddObject('Tabela', SrvESB.Tabela);

    TextoOrigem := '';
    TextoNovo := '';

    SrvESB.fsScript1.AddVariable('TextoOrigem', 'String', TextoOrigem);
    SrvESB.fsScript1.AddVariable('TextoNovo', 'String', TextoNovo);
```

```
    SrvESB.fsScript1.AddMethod('function ExtractFilePath(const FileName: string): string',
CallMethod);
    SrvESB.fsScript1.AddMethod('function RightStr(const AText: String; const ACount: Integer):
String', CallMethod);
    SrvESB.fsScript1.AddMethod('function LeftStr(const AText: String; const ACount: Integer):
String;', CallMethod);
    SrvESB.fsScript1.AddMethod('function AlDir(S:String; Largura:Integer; LimitaLargura :
Boolean=True):String', CallMethod);
    SrvESB.fsScript1.AddMethod('function AlEsq(S:String; Largura:Integer; LimitaLargura :
Boolean=True):String', CallMethod);
    SrvESB.fsScript1.AddMethod('function Zeros_Esquerda(Texto : String; Tamanho : Integer) :
String;', CallMethod);



    if SrvESB.fsScript1.Compile then
      begin
        SrvESB.fsScript1.Execute;

        TextoNovo := SrvESB.fsScript1.Variables['TextoNovo'];

        Result := True;
      end;
  except
  end;
end;

procedure TConector.SetarquivoDestino(const Value: String);
begin
  FarquivoDestino := Value;
end;

procedure TConector.SetarquivoOrigem(const Value: String);
begin
  FarquivoOrigem := Value;
end;

procedure TConector.Setdirecao(const Value: String);
begin
  Fdirecao := Value;
end;

procedure TConector.SetFTP(const Value: TFTP);
begin
  FFTP := Value;
end;

procedure TConector.SetidConector(const Value: Integer);
begin
  FidConector := Value;
end;

procedure TConector.SetnroInsistencias(const Value: Integer);
begin
  FnroInsistencias := Value;
end;

procedure TConector.SetscriptModificacao(const Value: String);
begin
  FscriptModificacao := Value;
end;

procedure TConector.SettempoEntreInsistencias(const Value: Integer);
begin
  FtempoEntreInsistencias := Value;
end;

procedure TConector.Settipo(const Value: String);
begin
  Ftipo := Value;
end;

function TConector.RightStr(const AText: String;
  const ACount: Integer): AnsiString;
begin
```

```
  Result := Copy(WideString(AText), Length(WideString(AText)) + 1 - ACount, ACount);
end;

function TConector.LeftStr(const AText: String;
  const ACount: Integer): String;
begin
  Result := Copy(WideString(AText), 1, ACount);
end;

function TConector.AlDir(S: String; Largura: Integer;
  LimitaLargura: Boolean): String;
var
  i :integer;
begin
  result := '';

  if (length(S) > Largura) and LimitaLargura then
    S := copy(S, 1, Largura);

  if length(S) > Largura then
    begin
      for i:= 1 to Largura do
        begin
          result := result + '*';
        end;
    end
  else
    begin
      for i:=1 to (largura - length(S)) do
        begin
          result := result + ' ';
        end;
      result := result + S;
    end;
end;

function TConector.AlEsq(S: String; Largura: Integer;
  LimitaLargura: Boolean): String;
var
 i : integer;

begin
  result := '';

  if (length(S) > Largura) and LimitaLargura then
    S := copy(S, 1, Largura);

  if length(S) > Largura then
    begin
      for i:= 1 to Largura do
        begin
          result := result + '*';
        end;
    end
  else
    begin
      for i:=1 to (largura - length(S)) do
        begin
          result := result + ' ';
        end;
      result := S + result;
    end;
end;

function TConector.Zeros_Esquerda(Texto: String; Tamanho: Integer): String;
var
  i : Integer;
begin
  result := '';

  Texto := Trim(texto);
  if Tamanho >= Length(Texto) then begin
    for i:= 1 to Tamanho - Length(Texto) do
      begin
        result := result + '0';
      end;
    result := result + Texto;
```

```
  end
  else
    Result := RightStr(Texto, Tamanho);
end;


function TConector.Conectar: Boolean;
begin
  Result := False;

  SrvESB.IdFTP1.Host := Conector.Autenticacao.host;
  SrvESB.IdFTP1.Port := Conector.Autenticacao.porta;
  SrvESB.IdFTP1.Username := Conector.Autenticacao.usuario;
  SrvESB.IdFTP1.Password := Conector.Autenticacao.senha;

  if Conector.FTP.ModoConexao = 'Passivo' then
    SrvESB.IdFTP1.Passive := True
  else
    SrvESB.IdFTP1.Passive := False;

  SrvESB.IdFTP1.Connect(True, Conector.Autenticacao.timeOut);

  Result := SrvESB.IdFTP1.Connected;
end;

function TConector.Enviar: Boolean;
begin
  Result := True;

  try
    if Conector.scriptModificacao <> '' then
      SrvESB.IdFTP1.Put(Conector.arquivoTransformado, Conector.arquivoDestino)
    else
      SrvESB.IdFTP1.Put(Conector.arquivoOrigem, Conector.arquivoDestino);
  except
    Result := False;
  end;
end;

function TConector.ExecutaAcaoArquivo(AcaoOrigem: String): Boolean;
begin
  Result := True;

  if Conector.FTP.AcaoOrigem = 'Apagar' then
    DeleteFile(PChar(Conector.arquivoOrigem))
  else
  if Conector.FTP.AcaoOrigem = 'Renomear' then
    RenameFile(Conector.arquivoOrigem, Conector.arquivoOrigem + '_old')
  else
  if Conector.FTP.AcaoOrigem = 'Mover' then
    begin
      CopyFile(PChar(Conector.arquivoOrigem), PChar(Conector.FTP.MovePara +
ExtractFileName(Conector.arquivoOrigem)), False);
      DeleteFile(PChar(Conector.arquivoOrigem));
    end;
end;

procedure TConector.SetAutenticacao(const Value: TAutenticacao);
begin
  FAutenticacao := Value;
end;

function TConector.ConfigurarAutenticacao(sHost, sUsuario, sSenha: String;
  iPorta, iTimeOut: Integer): Boolean;
begin
  Result := True;

  try
    Autenticacao.host := sHost;
    Autenticacao.usuario := sUsuario;
    Autenticacao.senha := sSenha;
    Autenticacao.porta := iPorta;
    Autenticacao.timeOut := iTimeOut;
  except
    Result := False;
  end;
end;
```

```
procedure TConector.SetarquivoTransformado(const Value: String);
begin
  FarquivoTransformado := Value;
end;


function TConector.Desconectar: Boolean;
begin
  try
    SrvESB.IdFTP1.Disconnect;
  except
  end;

  Result := True;
end;

end.
```

```
unit UFTP;

interface

type
  TFTP = class(TObject)
  private
    FModoConexao: String;
    FMovePara: String;
    FAcaoOrigem: String;
    FRepresentacao: String;
    procedure SetAcaoOrigem(const Value: String);
    procedure SetModoConexao(const Value: String);
    procedure SetMovePara(const Value: String);
    procedure SetRepresentacao(const Value: String);
  public
  published
    property AcaoOrigem : String read FAcaoOrigem write SetAcaoOrigem;
    property MovePara : String read FMovePara write SetMovePara;
    property ModoConexao : String read FModoConexao write SetModoConexao;
    property Representacao : String read FRepresentacao write SetRepresentacao;
  end;


implementation

{ TFTP }

procedure TFTP.SetAcaoOrigem(const Value: String);
begin
  FAcaoOrigem := Value;
end;

procedure TFTP.SetModoConexao(const Value: String);
begin
  FModoConexao := Value;
end;

procedure TFTP.SetMovePara(const Value: String);
begin
  FMovePara := Value;
end;

procedure TFTP.SetRepresentacao(const Value: String);
begin
  FRepresentacao := Value;
end;

end
```

APÊNDICE F – Diagrama ER